IPDL Case Studies

Kristina Sojakova Mihai Codescu Joshua Gancher

December 13, 2023

Abstract

We present here the full proofs of our IPDL case studies: Authenticated-To-Secure Channel in Section 3; Diffie-Hellman Key Exchange (DHKE) in Section 4; DHKE + One-Time Pad in 5; El Gamal Encryption in Section 6; Oblivious Transfer: 1-Out-Of-2 Pre-Processing in Section 7; Multi-Party Coin Toss in Section 8; and Two-Party GMW Protocol in Section 9.

Acknowledgement

This project was funded through the NGI Assure Fund, a fund established by NLnet with financial support from the European Commission's Next Generation Internet programme, under the aegis of DG Communications Networks, Content and Technology under grant agreement No. 957073.

1 Symmetric-Key Encryption: CPA Security

Semantic security, also known as *security against chosen-plaintext attacks (CPA)*, for a symmetric-key encryption schema states that encoding q fixed messages with a secret key is computationally indistinguishable from encoding any other q messages. Or equivalently, encoding q fixed messages with a secret key is computationally indistinguishable from encoding the same message q times. We now show this equivalence in IPDL.

Formally, we assume types $\mathsf{key}, \mathsf{msg}, \mathsf{ctxt}$ of keys, messages, and ciphertexts, respectively; a chosen message $\mathsf{zeros} : 1 \to \mathsf{msg}$; a probabilistic key generation algorithm $\mathsf{gen}_{\mathsf{key}} : 1 \twoheadrightarrow \mathsf{key}$; and a probabilistic encryption algorithm $\mathsf{enc} : \mathsf{msg} \times \mathsf{key} \twoheadrightarrow \mathsf{ctxt}$ that takes a message and a key, and returns a ciphertext. We will write $\mathsf{enc}(m, k)$ in place of $\mathsf{enc}\ (m, k)$.

We can express the standard (*left-right*) version of the CPA cryptographic assumption as the following protocol-level axiom: in the channel context $\{\mathsf{Msg}_L(i) : \mathsf{msg}\}_i$, $\{\mathsf{Msg}_R(i) : \mathsf{msg}\}_i$, $\mathsf{Key} : \mathsf{key}$, $\{\mathsf{Enc}(i) : \mathsf{ctxt}\}_i$ where $i := 1, \ldots, q$, the protocol

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m_L, k)$ for $0 \leqslant i < q$

with inputs $\mathsf{Msg}_L(-)$, $\mathsf{Msg}_R(-)$, outputs $\mathsf{Enc}(-)$, and an internal channel Key rewrites approximately to the protocol

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m_R, k)$ for $0 \leqslant i < q$

The *chosen-message* version of CPA security is another protocol-level axiom: in the channel context $\{\mathsf{Msg}(i) : \mathsf{msg}\}_i, \mathsf{Key} : \mathsf{key}, \{\mathsf{Enc}(i) : \mathsf{ctxt}\}_i$ where $i := 1, \ldots, q$, the protocol

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{Msg}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m, k)$ for $0 \leqslant i < q$

with inputs $\mathsf{Msg}(-)$, outputs $\mathsf{Enc}(-)$, and an internal channel Key rewrites approximately to the protocol

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$

- $\mathsf{Enc}(i) := m \leftarrow \mathsf{Msg}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(\mathsf{zeros}, k)$ for $0 \leqslant i < q$

To prove that these are equivalent, assume first the *left-right* version of CPA security. The protocol

- $\mathsf{Key} := \mathsf{samp\ gen_{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{Msg}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(m, k)$ for $0 \leqslant i < q$

can be factored as the composition of the protocol

- $\mathsf{Key} := \mathsf{samp\ gen_{key}}$
- $\mathsf{Enc}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(m_L, k)$ for $0 \leqslant i < q$

(the left-hand side of our CPA assumption) and the protocol

- $\mathsf{Msg}_L(i) := \mathsf{read\ Msg}(i)$ for $0 \leqslant i < q$
- $\mathsf{Msg}_R(i) := \mathsf{ret\ zeros}$ for $0 \leqslant i < q$

followed by the hiding of the channels $\mathsf{Msg}_L(-)$, $\mathsf{Msg}_R(-)$.

Applying the CPA assumption yields the composition of the protocol

- $\mathsf{Key} := \mathsf{samp\ gen_{key}}$
- $\mathsf{Enc}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(m_R, k)$ for $0 \leqslant i < q$

and the (unchanged) protocol

- $\mathsf{Msg}_L(i) := \mathsf{read\ Msg}(i)$ for $0 \leqslant i < q$
- $\mathsf{Msg}_R(i) := \mathsf{ret\ zeros}$ for $0 \leqslant i < q$

followed by the hiding of the channels $\mathsf{Msg}_L(-)$, $\mathsf{Msg}_R(-)$.

Reversing the factorization process yields the protocol

- $\mathsf{Key} := \mathsf{samp\ gen_{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{Msg}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(\mathsf{zeros}, k)$ for $0 \leqslant i < q$

as desired.

For the other direction, assume the *chosen-message* version of CPA security. The protocol

- $\mathsf{Key} := \mathsf{samp\ gen_{key}}$
- $\mathsf{Enc}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(m_L, k)$ for $0 \leqslant i < q$

can be factored as the composition of the protocol

- $\mathsf{Key} := \mathsf{samp\ gen_{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{Msg}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(m, k)$ for $0 \leqslant i < q$

(the left-hand side of our CPA assumption) and the protocol

- $\mathsf{Msg}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ \mathsf{ret}\ m_L$ for $0 \leqslant i < q$

followed by the hiding of the channels $\mathsf{Msg}(-)$.

Applying the CPA assumption yields the composition of the protocol

- $\mathsf{Key} := \mathsf{samp\ gen_{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{Msg}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(\mathsf{zeros}, k)$ for $0 \leqslant i < q$

and the (unchanged) protocol

- $\mathsf{Msg}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ \mathsf{ret}\ m_L$ for $0 \leqslant i < q$

followed by the hiding of the channels $\mathsf{Msg}(-)$.

Substituting the channels $\mathsf{Msg}(-)$ away yields the protocol

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(\mathsf{zeros}, k)$ for $0 \leqslant i < q$

The above can be factored as the composition of the protocol

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{Msg}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(\mathsf{zeros}, k)$ for $0 \leqslant i < q$

(the right-hand side of our CPA assumption) and the protocol

- $\mathsf{Msg}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ \mathsf{ret}\ m_R$ for $0 \leqslant i < q$

followed by the hiding of the channels $\mathsf{Msg}(-)$.

Applying the CPA assumption again (this time from right to left) yields the composition of the protocol

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{Msg}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m, k)$ for $0 \leqslant i < q$

and the (unchanged) protocol

- $\mathsf{Msg}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ \mathsf{ret}\ m_R$ for $0 \leqslant i < q$

followed by the hiding of the channels $\mathsf{Msg}(-)$.

Reversing the factorization process yields the protocol

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m_R, k)$ for $0 \leqslant i < q$

as desired.

2 Symmetric-Key Encryption: CPA$-To-CPA Security

Many CPA-secure encryption schemas also satisfy the stronger property of having *pseudo-random ciphertexts in the presence of chosen-plaintext attacks (CPA$)*. This latter notion states that encoding q fixed messages with a secret key is computationally indistinguishable from generating q random ciphertexts. We now show that CPA$ security implies CPA security in IPDL.

Formally, we assume types $\mathsf{key}, \mathsf{msg}, \mathsf{ctxt}$ of keys, messages, and ciphertexts, respectively; a chosen message $\mathsf{zeros} : 1 \to \mathsf{msg}$; a uniform distribution $\mathsf{unif}_{\mathsf{ctxt}} : 1 \twoheadrightarrow \mathsf{ctxt}$ on ciphertexts; a probabilistic key generation algorithm $\mathsf{gen}_{\mathsf{key}} : 1 \twoheadrightarrow \mathsf{key}$; and a probabilistic encryption algorithm $\mathsf{enc} : \mathsf{msg} \times \mathsf{key} \twoheadrightarrow \mathsf{ctxt}$ that takes a message and a key, and returns a ciphertext. We will write $\mathsf{enc}(m, k)$ in place of $\mathsf{enc}\ (m, k)$.

We can express the CPA$ cryptographic assumption as the following protocol-level axiom: in the channel context $\{\mathsf{Msg}(i) : \mathsf{msg}\}_i$, $\mathsf{Key} : \mathsf{key}$, $\{\mathsf{Enc}(i) : \mathsf{ctxt}\}_i$ where $i := 1, \ldots, q$, the protocol

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{Msg}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m, k)$ for $0 \leqslant i < q$

with inputs $\mathsf{Msg}(-)$, outputs $\mathsf{Enc}(-)$, and an internal channel Key rewrites approximately to the protocol

- $\mathsf{Enc}(i) := m \leftarrow \mathsf{Msg};\ \mathsf{samp}\ \mathsf{unif}_{\mathsf{ctxt}}$ for $0 \leqslant i < q$

We now show that the above axiom implies CPA security. The protocol

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m_L, k)$ for $0 \leqslant i < q$

can be factored as the composition of the protocol

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{Msg}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m, k)$ for $0 \leqslant i < q$

(the left-hand side of the CPA\$ assumption) and the protocol

- $\mathsf{Msg}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ \mathsf{ret}\ m_L$ for $0 \leqslant i < q$

followed by the hiding of the channels $\mathsf{Msg}(-)$.

Applying the CPA\$ assumption yields the composition of the protocol

- $\mathsf{Enc}(i) := m \leftarrow \mathsf{Msg};\ \mathsf{samp}\ \mathsf{unif}_{\mathsf{ctxt}}$ for $0 \leqslant i < q$

and the (unchanged) protocol

- $\mathsf{Msg}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ \mathsf{ret}\ m_L$ for $0 \leqslant i < q$

followed by the hiding of the channels $\mathsf{Msg}(-)$.

Substituting the channels $\mathsf{Msg}(-)$ away yields the protocol

- $\mathsf{Enc}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ \mathsf{samp}\ \mathsf{unif}_{\mathsf{ctxt}}$ for $0 \leqslant i < q$

The above can be factored as the composition of the protocol

- $\mathsf{Enc}(i) := m \leftarrow \mathsf{Msg}(i);\ \mathsf{samp}\ \mathsf{unif}_{\mathsf{ctxt}}$ for $0 \leqslant i < q$

(the right-hand side of the CPA\$ assumption) and the protocol

- $\mathsf{Msg}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ \mathsf{ret}\ m_R$ for $0 \leqslant i < q$

followed by the hiding of the channels $\mathsf{Msg}(-)$.

Applying the CPA\$ assumption again (this time from right to left) yields the composition of the protocol

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{Msg}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m, k)$ for $0 \leqslant i < q$

and the (unchanged) protocol

- $\mathsf{Msg}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ \mathsf{ret}\ m_R$ for $0 \leqslant i < q$

followed by the hiding of the channels $\mathsf{Msg}(-)$.

Reversing the factorization process yields the protocol

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m_L \leftarrow \mathsf{Msg}_L(i);\ m_R \leftarrow \mathsf{Msg}_R(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m_R, k)$ for $0 \leqslant i < q$

as desired.

3 Symmetric-Key Encryption: Authenticated-To-Secure Channel

Alice wants to communicate q messages to Bob using an authenticated channel. The authenticated channel is not secure: it leaks each message to Eve, and waits to receive an ok message back from her before delivering the in-flight message. Thus, Eve cannot modify any of the messages but can read and delay them for any amount of time. To transmit information securely, Alice sends encryptions of her messages, which Bob decrypts using a shared key not known to Eve.

Formally, we assume the types $\mathsf{key}, \mathsf{msg}, \mathsf{ctxt}$ of keys, messages, and ciphertexts, respectively; a chosen message $\mathsf{zeros} : 1 \to \mathsf{msg}$; a probabilistic key generation algorithm $\mathsf{gen}_{\mathsf{key}} : 1 \twoheadrightarrow \mathsf{key}$; a probabilistic encryption algorithm $\mathsf{enc} : \mathsf{msg} \times \mathsf{key} \twoheadrightarrow \mathsf{ctxt}$ that takes a message and a key, and returns a ciphertext; and a decryption algorithm $\mathsf{dec} : \mathsf{ctxt} \times \mathsf{key} \to \mathsf{msg}$ that takes a ciphertext and a key, and returns a message. We will write $\mathsf{enc}(m, k)$ and $\mathsf{dec}(c, k)$ in place of $\mathsf{enc}\ (m, k)$ and $\mathsf{dec}\ (c, k)$.

3.1 The Assumptions

The *correctness* assumption on the encryption schema states that encoding and decoding a single message with the same key yields the original message. We express this as a protocol-level axiom: in the channel context $\mathsf{In} : \mathsf{msg}, \mathsf{Key} : \mathsf{key}, \mathsf{Enc} : \mathsf{ctxt}, \mathsf{Dec} : \mathsf{msg}$ the protocol

- $\mathsf{Enc} := m \leftarrow \mathsf{In};\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(m, k)$
- $\mathsf{Dec} := c \leftarrow \mathsf{Enc};\ k \leftarrow \mathsf{Key};\ \mathsf{ret\ dec}(c, k)$

with inputs In, Key and outputs Enc, Dec rewrites strictly to the protocol

- $\mathsf{Enc} := m \leftarrow \mathsf{In};\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(m, k)$
- $\mathsf{Dec} := \mathsf{read\ In}$

Applying this assumption q times, we get that the protocol

- $\mathsf{Enc}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(m, k)$ for $0 \leqslant i < q$
- $\mathsf{Dec}(i) := c \leftarrow \mathsf{Enc}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{ret\ dec}(c, k)$ for $0 \leqslant i < q$

rewrites strictly to the protocol

- $\mathsf{Enc}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(m, k)$ for $0 \leqslant i < q$
- $\mathsf{Dec}(i) := \mathsf{read\ In}(i)$ for $0 \leqslant i < q$

The CPA cryptographic assumption states that if the key is secret, encoding q fixed messages is computationally indistinguishable from encoding the chosen message q times: in the channel context $\{\mathsf{In}(i) : \mathsf{msg}\}_i$, $\mathsf{Key} : \mathsf{key}$, $\{\mathsf{Enc}(i) : \mathsf{ctxt}\}_i$ where $i := 1, \ldots, q$, the protocol

- $\mathsf{Key} := \mathsf{samp\ gen_{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(m, k)$ for $0 \leqslant i < q$

with inputs $\mathsf{In}(-)$, outputs $\mathsf{Enc}(-)$, and an internal channel Key rewrites approximately to the protocol

- $\mathsf{Key} := \mathsf{samp\ gen_{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(\mathsf{zeros}, k)$ for $0 \leqslant i < q$

3.2 The Ideal Functionality

The ideal functionality reads the input message, leaks a confirmation to Eve to signal that the message has been received, and, upon the approval from Eve, outputs the message:

- $\mathsf{LeakMsgRcvd}(i)^{\mathsf{id}}_{\mathsf{adv}} := m \leftarrow \mathsf{In}(i);\ \mathsf{ret}\ \checkmark$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkMsg}(i)^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{read\ In}(i)$ for $0 \leqslant i < q$

3.3 The Real Protocol

The real-world protocol consists of Alice, Bob, the key-generating functionality, and the authenticated channel. The functionality starts by calling the key generation algorithm:

- $\mathsf{Key} := \mathsf{samp\ gen_{key}}$

Alice encrypts each input with the provided key, samples a ciphertext from the resulting distribution, and sends it to the authenticated channel:

- $\mathsf{Send}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp\ enc}(m, k)$

The authenticated channel leaks each ciphertext received from Alice to Eve, and, upon receiving the okay from Eve, forwards the ciphertext to Bob:

- $\mathsf{LeakCtxt}(i)^{\mathsf{net}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Send}(i)$
- $\mathsf{Recv}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{Send}(i)$

Bob decrypts each ciphertext with the shared key and outputs the result:

- $\mathsf{Out}(i) := c \leftarrow \mathsf{Recv}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{ret}\ \mathsf{dec}(c, k)$

Composing all of this together and hiding the internal communication yields the real-world protocol.

3.4 The Simulator

The simulator turns the adversarial inputs and outputs of the real world protocol into the adversarial inputs and outputs of the ideal functionality, thereby converting any adversary for the real-world protocol into an adversary for the ideal functionality. This means that the channels $\mathsf{LeakMsgRcvd}(-)^{\mathsf{id}}_{\mathsf{adv}}, \mathsf{OkCtxt}(-)^{\mathsf{adv}}_{\mathsf{net}}$ are inputs to the simulator and the channels $\mathsf{LeakCtxt}(-)^{\mathsf{net}}_{\mathsf{adv}}, \mathsf{OkMsg}(-)^{\mathsf{adv}}_{\mathsf{id}}$ are the outputs. Hence, upon receiving the empty message from the ideal functionality to indicate that a message has been received, the simulator must conjure up a ciphertext to leak to Eve. This is accomplished by generating a key and encrypting the chosen message:

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{LeakCtxt}(i)^{\mathsf{net}}_{\mathsf{adv}} := \_ \leftarrow \mathsf{LeakMsgRcvd}(i)^{\mathsf{id}}_{\mathsf{adv}};\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(\mathsf{zeros}, k)$ for $0 \leqslant i < q$

Upon receiving the approval from Eve for the generated ciphertext, the simulator gives the approval to the functionality to output the message:

- $\mathsf{OkMsg}(i)^{\mathsf{adv}}_{\mathsf{id}} := \mathsf{read}\ \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}}$ for $0 \leqslant i < q$

Putting this all together yields the following code for the simulator:

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{LeakCtxt}(i)^{\mathsf{net}}_{\mathsf{adv}} := \_ \leftarrow \mathsf{LeakMsgRcvd}(i)^{\mathsf{id}}_{\mathsf{adv}};\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(\mathsf{zeros}, k)$ for $0 \leqslant i < q$
- $\mathsf{OkMsg}(i)^{\mathsf{adv}}_{\mathsf{id}} := \mathsf{read}\ \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}}$ for $0 \leqslant i < q$

3.5 Real ≈ Ideal + Simulator

Composing the simulator with the ideal functionality and hiding the internal communication yields the following:

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{LeakCtxt}(i)^{\mathsf{net}}_{\mathsf{adv}} := \_ \leftarrow \mathsf{LeakMsgRcvd}(i)^{\mathsf{id}}_{\mathsf{adv}};\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(\mathsf{zeros}, k)$ for $0 \leqslant i < q$
- $\mathsf{OkMsg}(i)^{\mathsf{adv}}_{\mathsf{id}} := \mathsf{read}\ \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}}$ for $0 \leqslant i < q$
- $\mathsf{LeakMsgRcvd}(i)^{\mathsf{id}}_{\mathsf{adv}} := m \leftarrow \mathsf{In}(i);\ \mathsf{ret}\ \checkmark$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkMsg}(i)^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

The internal channels $\mathsf{LeakMsgRcvd}(-)^{\mathsf{id}}_{\mathsf{adv}}$ and $\mathsf{OkMsg}(-)^{\mathsf{adv}}_{\mathsf{id}}$ that originally served as a line of communication for Eve can now be substituted away:

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{LeakCtxt}(i)^{\mathsf{net}}_{\mathsf{adv}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(\mathsf{zeros}, k)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

Next we move on to simplifying the real protocol. Explicitly, we have the code below:

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Send}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m, k)$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{net}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Send}(i)$ for $0 \leqslant i < q$
- $\mathsf{Recv}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{Send}(i)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := c \leftarrow \mathsf{Recv}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{ret}\ \mathsf{dec}(c, k)$ for $0 \leqslant i < q$

We first substitute away the internal channels $\mathsf{Recv}(-)$:

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Send}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m, k)$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{net}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Send}(i)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ c \leftarrow \mathsf{Send}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{ret}\ \mathsf{dec}(c, k)$ for $0 \leqslant i < q$

Next we conceptually separate the encryption and decryption actions from the message-passing in the real-world by introducing new internal channels $\mathsf{Enc}(-)$ and $\mathsf{Dec}(-)$:

- $\mathsf{Key} := \mathsf{samp}\ \ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m, k)$ for $0 \leqslant i < q$
- $\mathsf{Send}(i) := \mathsf{read}\ \mathsf{Enc}(i)$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{net}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Send}(i)$ for $0 \leqslant i < q$
- $\mathsf{Dec}(i) := c \leftarrow \mathsf{Send}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{ret}\ \mathsf{dec}(c, k)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{Dec}(i)$ for $0 \leqslant i < q$

We can now substitute away the internal channels $\mathsf{Send}(-)$ as well:

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m, k)$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{net}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Enc}(i)$ for $0 \leqslant i < q$
- $\mathsf{Dec}(i) := c \leftarrow \mathsf{Enc}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{ret}\ \mathsf{dec}(c, k)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{Dec}(i)$ for $0 \leqslant i < q$

As we observed earlier, the subprotocol

- $\mathsf{Enc}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m, k)$ for $0 \leqslant i < q$
- $\mathsf{Dec}(i) := c \leftarrow \mathsf{Enc}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{ret}\ \mathsf{dec}(c, k)$ for $0 \leqslant i < q$

strictly rewrites to the following protocol snippet:

- $\mathsf{Enc}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m, k)$ for $0 \leqslant i < q$
- $\mathsf{Dec}(i) := \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

Our original protocol thus becomes:

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m, k)$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{net}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Enc}(i)$ for $0 \leqslant i < q$
- $\mathsf{Dec}(i) := \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{Dec}(i)$ for $0 \leqslant i < q$

Since the channel Key is only used in the channels $\mathsf{Enc}(-)$, we can extract the following subprotocol, where Key is hidden:

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(m, k)$ for $0 \leqslant i < q$

The CPA assumption allows us to approximately rewrite the above protocol snippet to

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(\mathsf{zeros}, k)$ for $0 \leqslant i < q$

Plugging this back into the original protocol yields the following:

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(\mathsf{zeros}, k)$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{net}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Enc}(i)$ for $0 \leqslant i < q$
- $\mathsf{Dec}(i) := \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{Dec}(i)$ for $0 \leqslant i < q$

Finally, we can fold away the internal channels $\mathsf{Enc}(-)$ and $\mathsf{Dec}(-)$:

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{gen}_{\mathsf{key}}$
- $\mathsf{LeakCtxt}(i)^{\mathsf{net}}_{\mathsf{adv}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{Key};\ \mathsf{samp}\ \mathsf{enc}(\mathsf{zeros}, k)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

This is precisely the simplified composition of the ideal functionality and the simulator from the beginning of this section.

4 Diffie-Hellman Key Exchange (DHKE)

In symmetric-key encryption, the sender (Alice) and the receiver (Bob) need to agree on a shared secret key. One such key-agreement protocol is the Diffie-Hellman Key Exchange (DHKE), which we now prove secure in IPDL.

Formally, we assume a type key of secret keys (elements of $\{0, \ldots, p-1\}$, where p is a prime); a type msg of messages, also serving as public keys (elements of a cyclic group $G = \{g^0, \ldots, g^{p-1}\}$); a uniform distribution $\mathsf{unif}_{\mathsf{key}} : 1 \twoheadrightarrow \mathsf{key}$ on keys; a uniform distribution $\mathsf{unif}_{\mathsf{msg}} : 1 \twoheadrightarrow \mathsf{msg}$ on messages; the generator $\mathsf{g} : 1 \rightarrow \mathsf{msg}$ of G; and the exponentiation function $\mathsf{exp} : \mathsf{msg} \times \mathsf{key} \rightarrow \mathsf{msg}$ that raises an element of G to the power of $k \in \{0, \ldots, p-1\}$. We will write m^k in place of $\mathsf{exp}\ (m, k)$.

4.1 The Assumptions

At the level of expressions, we only need to know that exponents commute:

- $k : \mathsf{key}, l : \mathsf{key} \vdash (\mathsf{g}^l)^k = (\mathsf{g}^k)^l : \mathsf{msg}$, and

At the level of distributions, we need to know that sampling a random secret key k from $\mathsf{unif}_{\mathsf{key}}$ and returning g^k is the same as sampling from $\mathsf{unif}_{\mathsf{msg}}$ directly (a consequence of $\mathsf{unif}_{\mathsf{key}}$ being uniform),

- $\cdot \vdash \left(k \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^k\right) = \mathsf{unif}_{\mathsf{msg}} : \mathsf{msg}$

Finally, at the level of protocols we need the *decisional Diffie-Hellman (DDH)* cryptographic assumption: as long as the secret keys k, l are generated uniformly, even if the adversary knows the values g^k and g^l, they will be unable to distinguish $(g^k)^l$ from a uniformly generated element of G. The corresponding protocol-level axiom states that in the channel context $\mathsf{DDH}_3 : \mathsf{msg} \times (\mathsf{msg} \times \mathsf{msg})$, the single-reaction no-input protocol

- $\mathsf{DDH}_3 \coloneqq k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \left(\mathsf{g}^{k_A}, \left(\mathsf{g}^{k_B}, (\mathsf{g}^{k_A})^{k_B}\right)\right)$

rewrites approximately to the protocol

- $\mathsf{DDH}_3 \coloneqq k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ k_R \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \left(\mathsf{g}^{k_A}, \left(\mathsf{g}^{k_B}, \mathsf{g}^{k_R}\right)\right)$

4.2 The Ideal Functionality

The ideal functionality for the key exchange generates the shared key randomly, and, upon the approval from the adversary, sends it to the respective party:

- $\mathsf{SharedKey} \coloneqq \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$
- $\mathsf{Key(Alice)} \coloneqq \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{read}\ \mathsf{SharedKey}$
- $\mathsf{Key(Bob)} \coloneqq \_ \leftarrow \mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{read}\ \mathsf{SharedKey}$

Here the channel $\mathsf{SharedKey}$ is internal.

4.3 The Real Protocol

The real-world protocol consists of Alice, Bob, and two authenticated channels. Alice randomly generates a secret key k_A and sends the value g^{k_A} as her public key to Bob using one of the authenticated channels:

- $\mathsf{SecretKey(Alice)} \coloneqq \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{Send(Alice, Bob)} \coloneqq k_A \leftarrow \mathsf{SecretKey(Alice)};\ \mathsf{ret}\ \mathsf{g}^{k_A}$

Symmetrically, Bob generates a secret key k_B and sends the value g^{k_B} to Alice using the second authenticated channel:

- $\mathsf{SecretKey(Bob)} \coloneqq \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{Send(Bob, Alice)} \coloneqq k_B \leftarrow \mathsf{SecretKey(Bob)};\ \mathsf{ret}\ \mathsf{g}^{k_B}$

Each authenticated channel leaks the corresponding public key to the adversary and waits for their approval before forwarding the key to the other party:

- $\mathsf{LeakPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}} \coloneqq \mathsf{read}\ \mathsf{Send(Alice, Bob)}$
- $\mathsf{LeakPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}} \coloneqq \mathsf{read}\ \mathsf{Send(Bob, Alice)}$
- $\mathsf{Recv(Alice, Bob)} \coloneqq \_ \leftarrow \mathsf{OkPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{Send(Alice, Bob)}$
- $\mathsf{Recv(Bob, Alice)} \coloneqq \_ \leftarrow \mathsf{OkPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{Send(Bob, Alice)}$

Upon receiving Bob's public key g^{k_B}, Alice computes the shared key as the value $(g^{k_B})^{k_A}$:

- $\mathsf{Key}(\mathsf{Alice}) := p_B \leftarrow \mathsf{Recv}(\mathsf{Bob}, \mathsf{Alice});\ k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ \mathsf{ret}\ p_B^{k_A}$

Analogously, upon receiving Alice's public key g^{k_A}, Bob computes the shared key as the value $(g^{k_A})^{k_B}$:

- $\mathsf{Key}(\mathsf{Bob}) := p_A \leftarrow \mathsf{Recv}(\mathsf{Alice}, \mathsf{Bob});\ k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ p_A^{k_B}$

Thus, we have the following code for Alice:

- $\mathsf{SecretKey}(\mathsf{Alice}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{Send}(\mathsf{Alice}, \mathsf{Bob}) := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ \mathsf{ret}\ \mathsf{g}^{k_A}$
- $\mathsf{Key}(\mathsf{Alice}) := p_B \leftarrow \mathsf{Recv}(\mathsf{Bob}, \mathsf{Alice});\ k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ \mathsf{ret}\ p_B^{k_A}$

The code for Bob has the symmetric form:

- $\mathsf{SecretKey}(\mathsf{Bob}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{Send}(\mathsf{Bob}, \mathsf{Alice}) := k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{Key}(\mathsf{Bob}) := p_A \leftarrow \mathsf{Recv}(\mathsf{Alice}, \mathsf{Bob});\ k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ p_A^{k_B}$

At last we have the two authenticated channels: from Alice to Bob,

- $\mathsf{LeakPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{Send}(\mathsf{Alice}, \mathsf{Bob})$
- $\mathsf{Recv}(\mathsf{Alice}, \mathsf{Bob}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{Send}(\mathsf{Alice}, \mathsf{Bob})$

and from Bob to Alice:

- $\mathsf{LeakPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{Send}(\mathsf{Bob}, \mathsf{Alice})$
- $\mathsf{Recv}(\mathsf{Bob}, \mathsf{Alice}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{Send}(\mathsf{Bob}, \mathsf{Alice})$

Composing all of this together and hiding the internal communication yields the Diffie-Hellman Key Exchange protocol.

4.4 The Simulator

The channels $\mathsf{OkPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}}$ and $\mathsf{OkPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}}$ are inputs to the simulator, whereas the channels $\mathsf{LeakPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}}$, $\mathsf{LeakPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}}$ and $\mathsf{OkKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{id}}$, $\mathsf{OkKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{id}}$ are the outputs.

The simulator constructs the public keys for Alice and Bob exactly as in the real protocol: it randomly generates the respective secret keys k_A and k_B, and computes the corresponding public keys as g^{k_A} and g^{k_B}. In the real-world Diffie-Hellman Key Exchange, Alice can only construct the shared key once the adversary approves the forwarding of Bob's public key to her. Hence, the ideal functionality is only allowed to forward the shared key to Alice once the approval for Bob's public key clears, and vice versa.

- $\mathsf{SecretKey}(\mathsf{Alice}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SecretKey}(\mathsf{Bob}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{PublicKey}(\mathsf{Alice}) := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ \mathsf{ret}\ \mathsf{g}^{k_A}$
- $\mathsf{PublicKey}(\mathsf{Bob}) := k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Alice})$
- $\mathsf{LeakPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Bob})$
- $\mathsf{OkKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{id}} := \mathsf{read}\ \mathsf{OkPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}}$
- $\mathsf{OkKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{id}} := \mathsf{read}\ \mathsf{OkPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}}$

In the above, the channels $\mathsf{SecretKey}(\mathsf{Alice})$, $\mathsf{SecretKey}(\mathsf{Bob})$ and $\mathsf{PublicKey}(\mathsf{Alice})$, $\mathsf{PublicKey}(\mathsf{Bob})$ are internal.

4.5 Real $\approx$ Ideal + Simulator

Plugging the simulator into the ideal functionality and substituting away the internal channels $\mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}}$, $\mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}}$ that originally served as a line of communication for the adversary yields the following:

- $\mathsf{SecretKey(Alice)} := \mathsf{samp\ unif_{key}}$
- $\mathsf{SecretKey(Bob)} := \mathsf{samp\ unif_{key}}$
- $\mathsf{SharedKey} := \mathsf{samp\ unif_{msg}}$
- $\mathsf{PublicKey(Alice)} := k_A \leftarrow \mathsf{SecretKey(Alice)};\ \mathsf{ret\ g}^{k_A}$
- $\mathsf{PublicKey(Bob)} := k_B \leftarrow \mathsf{SecretKey(Bob)};\ \mathsf{ret\ g}^{k_B}$
- $\mathsf{LeakPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read\ PublicKey(Alice)}$
- $\mathsf{LeakPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read\ PublicKey(Bob)}$
- $\mathsf{Key(Alice)} := \_ \leftarrow \mathsf{OkPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read\ SharedKey}$
- $\mathsf{Key(Bob)} := \_ \leftarrow \mathsf{OkPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read\ SharedKey}$

Next we move on to simplifying the real protocol. Explicitly, we have the code below:

- $\mathsf{SecretKey(Alice)} := \mathsf{samp\ unif_{key}}$
- $\mathsf{SecretKey(Bob)} := \mathsf{samp\ unif_{key}}$
- $\mathsf{Send(Alice, Bob)} := k_A \leftarrow \mathsf{SecretKey(Alice)};\ \mathsf{ret\ g}^{k_A}$
- $\mathsf{Send(Bob, Alice)} := k_B \leftarrow \mathsf{SecretKey(Bob)};\ \mathsf{ret\ g}^{k_B}$
- $\mathsf{LeakPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read\ Send(Alice, Bob)}$
- $\mathsf{LeakPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read\ Send(Bob, Alice)}$
- $\mathsf{Recv(Alice, Bob)} := \_ \leftarrow \mathsf{OkPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read\ Send(Alice, Bob)}$
- $\mathsf{Recv(Bob, Alice)} := \_ \leftarrow \mathsf{OkPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read\ Send(Bob, Alice)}$
- $\mathsf{Key(Alice)} := p_B \leftarrow \mathsf{Recv(Bob, Alice)};\ k_A \leftarrow \mathsf{SecretKey(Alice)};\ \mathsf{ret}\ p_B^{k_A}$
- $\mathsf{Key(Bob)} := p_A \leftarrow \mathsf{Recv(Alice, Bob)};\ k_B \leftarrow \mathsf{SecretKey(Bob)};\ \mathsf{ret}\ p_A^{k_B}$

The internal channels $\mathsf{Send(Alice, Bob)}$, $\mathsf{Send(Bob, Alice)}$ and $\mathsf{Recv(Alice, Bob)}$, $\mathsf{Recv(Bob, Alice)}$ can be substituted away:

- $\mathsf{SecretKey(Alice)} := \mathsf{samp\ unif_{key}}$
- $\mathsf{SecretKey(Bob)} := \mathsf{samp\ unif_{key}}$
- $\mathsf{LeakPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}} := k_A \leftarrow \mathsf{SecretKey(Alice)};\ \mathsf{ret\ g}^{k_A}$
- $\mathsf{LeakPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}} := k_B \leftarrow \mathsf{SecretKey(Bob)};\ \mathsf{ret\ g}^{k_B}$
- $\mathsf{Key(Alice)} := \_ \leftarrow \mathsf{OkPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}};\ k_B \leftarrow \mathsf{SecretKey(Bob)};\ k_A \leftarrow \mathsf{SecretKey(Alice)};\ \mathsf{ret}\ (\mathsf{g}^{k_B})^{k_A}$
- $\mathsf{Key(Bob)} := \_ \leftarrow \mathsf{OkPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}};\ k_A \leftarrow \mathsf{SecretKey(Alice)};\ k_B \leftarrow \mathsf{SecretKey(Bob)};\ \mathsf{ret}\ (\mathsf{g}^{k_A})^{k_B}$

Exponents commute by assumption, so we can rewrite the channel $\mathsf{Key(Alice)}$ equivalently as

- $\mathsf{SecretKey(Alice)} := \mathsf{samp\ unif_{key}}$

- $\mathsf{SecretKey}(\mathsf{Bob}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{LeakPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}} := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ \mathsf{ret}\ \mathsf{g}^{k_A}$
- $\mathsf{LeakPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}} := k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{Key}(\mathsf{Alice}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}};\ k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ \mathsf{ret}\ (\mathsf{g}^{k_A})^{k_B}$
- $\mathsf{Key}(\mathsf{Bob}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}};\ k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ (\mathsf{g}^{k_A})^{k_B}$

After rearranging, the channels $\mathsf{Key}(\mathsf{Alice})$ and $\mathsf{Key}(\mathsf{Bob})$ construct the exact same shared key:

- $\mathsf{SecretKey}(\mathsf{Alice}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SecretKey}(\mathsf{Bob}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{LeakPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}} := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ \mathsf{ret}\ \mathsf{g}^{k_A}$
- $\mathsf{LeakPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}} := k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{Key}(\mathsf{Alice}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}};\ k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ (\mathsf{g}^{k_A})^{k_B}$
- $\mathsf{Key}(\mathsf{Bob}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}};\ k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ (\mathsf{g}^{k_A})^{k_B}$

We can extract the key construction into a new internal channel $\mathsf{SharedKey}$:

- $\mathsf{SecretKey}(\mathsf{Alice}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SecretKey}(\mathsf{Bob}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SharedKey} := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ (\mathsf{g}^{k_A})^{k_B}$
- $\mathsf{LeakPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}} := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ \mathsf{ret}\ \mathsf{g}^{k_A}$
- $\mathsf{LeakPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}} := k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{Key}(\mathsf{Alice}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$
- $\mathsf{Key}(\mathsf{Bob}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$

Similarly, we can extract the public key construction for Alice and Bob into new internal channels $\mathsf{PublicKey}(\mathsf{Alice})$ and $\mathsf{PublicKey}(\mathsf{Bob})$:

- $\mathsf{SecretKey}(\mathsf{Alice}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SecretKey}(\mathsf{Bob}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SharedKey} := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ (\mathsf{g}^{k_A})^{k_B}$
- $\mathsf{PublicKey}(\mathsf{Alice}) := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ \mathsf{ret}\ \mathsf{g}^{k_A}$
- $\mathsf{PublicKey}(\mathsf{Bob}) := k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Alice})$
- $\mathsf{LeakPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Bob})$
- $\mathsf{Key}(\mathsf{Alice}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$
- $\mathsf{Key}(\mathsf{Bob}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$

Adding a gratuitous dependency on the channel $\mathsf{SecretKey}(\mathsf{Bob})$ into the channel $\mathsf{PublicKey}(\mathsf{Alice})$, and, analogously, on the channel $\mathsf{SecretKey}(\mathsf{Alice})$ into the channel $\mathsf{PublicKey}(\mathsf{Bob})$ yields

- $\mathsf{SecretKey}(\mathsf{Alice}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SecretKey}(\mathsf{Bob}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SharedKey} := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ (\mathsf{g}^{k_A})^{k_B}$
- $\mathsf{PublicKey}(\mathsf{Alice}) := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ \mathsf{g}^{k_A}$
- $\mathsf{PublicKey}(\mathsf{Bob}) := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Alice})$
- $\mathsf{LeakPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Bob})$
- $\mathsf{Key}(\mathsf{Alice}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$
- $\mathsf{Key}(\mathsf{Bob}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$

We can now extract the DDH triple into a new internal channel DDH_3:

- $\mathsf{SecretKey}(\mathsf{Alice}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SecretKey}(\mathsf{Bob}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{DDH}_3 := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ \left(\mathsf{g}^{k_A}, \left(\mathsf{g}^{k_B}, (\mathsf{g}^{k_A})^{k_B}\right)\right)$
- $\mathsf{SharedKey} := \left(k_A, \left(k_B, k\right)\right) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ k$
- $\mathsf{PublicKey}(\mathsf{Alice}) := \left(k_A, \left(k_B, k\right)\right) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ k_A$
- $\mathsf{PublicKey}(\mathsf{Bob}) := \left(k_A, \left(k_B, k\right)\right) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ k_B$
- $\mathsf{LeakPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Alice})$
- $\mathsf{LeakPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Bob})$
- $\mathsf{Key}(\mathsf{Alice}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$
- $\mathsf{Key}(\mathsf{Bob}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$

The internal channels $\mathsf{SecretKey}(\mathsf{Alice})$ and $\mathsf{SecretKey}(\mathsf{Bob})$ are now only used in the channel DDH_3, so we can fold them in:

- $\mathsf{DDH}_3 := k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \left(\mathsf{g}^{k_A}, \left(\mathsf{g}^{k_B}, (\mathsf{g}^{k_A})^{k_B}\right)\right)$
- $\mathsf{SharedKey} := \left(k_A, \left(k_B, k\right)\right) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ k$
- $\mathsf{PublicKey}(\mathsf{Alice}) := \left(k_A, \left(k_B, k\right)\right) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ k_A$
- $\mathsf{PublicKey}(\mathsf{Bob}) := \left(k_A, \left(k_B, k\right)\right) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ k_B$
- $\mathsf{LeakPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Alice})$
- $\mathsf{LeakPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Bob})$
- $\mathsf{Key}(\mathsf{Alice}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$
- $\mathsf{Key}(\mathsf{Bob}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$

Using the DDH assumption, we can approximately rewrite the above protocol to

- $\mathsf{DDH}_3 := k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ k_R \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \left(\mathsf{g}^{k_A}, \left(\mathsf{g}^{k_B}, \mathsf{g}^{k_R}\right)\right)$
- $\mathsf{SharedKey} := \left(k_A, \left(k_B, k\right)\right) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ k$

- $\mathsf{PublicKey(Alice)} := \big(k_A, \big(k_B, k\big)\big) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ k_A$
- $\mathsf{PublicKey(Bob)} := \big(k_A, \big(k_B, k\big)\big) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ k_B$
- $\mathsf{LeakPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read\ PublicKey(Alice)}$
- $\mathsf{LeakPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read\ PublicKey(Bob)}$
- $\mathsf{Key(Alice)} := \_ \leftarrow \mathsf{OkPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read\ SharedKey}$
- $\mathsf{Key(Bob)} := \_ \leftarrow \mathsf{OkPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read\ SharedKey}$

Unfolding the three samplings from the channel DDH_3 into new internal channels SecretKey(Alice), SecretKey(Bob), Key yields

- $\mathsf{Key} := \mathsf{samp\ unif}_{\mathsf{key}}$
- $\mathsf{SecretKey(Alice)} := \mathsf{samp\ unif}_{\mathsf{key}}$
- $\mathsf{SecretKey(Bob)} := \mathsf{samp\ unif}_{\mathsf{key}}$
- $\mathsf{DDH}_3 := k_A \leftarrow \mathsf{SecretKey(Alice)};\ k_B \leftarrow \mathsf{SecretKey(Bob)};\ k_R \leftarrow \mathsf{Key};\ \mathsf{ret}\ \big(\mathsf{g}^{k_A}, \big(\mathsf{g}^{k_B}, \mathsf{g}^{k_R}\big)\big)$
- $\mathsf{SharedKey} := \big(k_A, \big(k_B, k\big)\big) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ k$
- $\mathsf{PublicKey(Alice)} := \big(k_A, \big(k_B, k\big)\big) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ k_A$
- $\mathsf{PublicKey(Bob)} := \big(k_A, \big(k_B, k\big)\big) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ k_B$
- $\mathsf{LeakPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read\ PublicKey(Alice)}$
- $\mathsf{LeakPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read\ PublicKey(Bob)}$
- $\mathsf{Key(Alice)} := \_ \leftarrow \mathsf{OkPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read\ SharedKey}$
- $\mathsf{Key(Bob)} := \_ \leftarrow \mathsf{OkPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read\ SharedKey}$

The internal channel DDH_3 can now be substituted away:

- $\mathsf{Key} := \mathsf{samp\ unif}_{\mathsf{key}}$
- $\mathsf{SecretKey(Alice)} := \mathsf{samp\ unif}_{\mathsf{key}}$
- $\mathsf{SecretKey(Bob)} := \mathsf{samp\ unif}_{\mathsf{key}}$
- $\mathsf{SharedKey} := k_A \leftarrow \mathsf{SecretKey(Alice)};\ k_B \leftarrow \mathsf{SecretKey(Bob)};\ k_R \leftarrow \mathsf{Key};\ \mathsf{ret}\ \mathsf{g}^{k_R}$
- $\mathsf{PublicKey(Alice)} := k_A \leftarrow \mathsf{SecretKey(Alice)};\ k_B \leftarrow \mathsf{SecretKey(Bob)};\ k_R \leftarrow \mathsf{Key};\ \mathsf{ret}\ \mathsf{g}^{k_A}$
- $\mathsf{PublicKey(Bob)} := k_A \leftarrow \mathsf{SecretKey(Alice)};\ k_B \leftarrow \mathsf{SecretKey(Bob)};\ k_R \leftarrow \mathsf{Key};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read\ PublicKey(Alice)}$
- $\mathsf{LeakPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read\ PublicKey(Bob)}$
- $\mathsf{Key(Alice)} := \_ \leftarrow \mathsf{OkPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read\ SharedKey}$
- $\mathsf{Key(Bob)} := \_ \leftarrow \mathsf{OkPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read\ SharedKey}$

Dropping the gratuitous dependencies – on channels SecretKey(Alice), SecretKey(Bob) in the channel SharedKey, on channels SecretKey(Bob), Key in the channel PublicKey(Alice), and on channels SecretKey(Alice), Key in the channel PublicKey(Bob) – yields the following:

- $\mathsf{Key} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SecretKey}(\mathsf{Alice}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SecretKey}(\mathsf{Bob}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SharedKey} := k_R \leftarrow \mathsf{Key};\ \mathsf{ret}\ \mathsf{g}^{k_R}$
- $\mathsf{PublicKey}(\mathsf{Alice}) := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ \mathsf{ret}\ \mathsf{g}^{k_A}$
- $\mathsf{PublicKey}(\mathsf{Bob}) := k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Alice})$
- $\mathsf{LeakPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Bob})$
- $\mathsf{Key}(\mathsf{Alice}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$
- $\mathsf{Key}(\mathsf{Bob}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$

Since the channel Key is now only used in the channel SharedKey, we can fold it in:

- $\mathsf{SecretKey}(\mathsf{Alice}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SecretKey}(\mathsf{Bob}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SharedKey} := k_R \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_R}$
- $\mathsf{PublicKey}(\mathsf{Alice}) := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ \mathsf{ret}\ \mathsf{g}^{k_A}$
- $\mathsf{PublicKey}(\mathsf{Bob}) := k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Alice})$
- $\mathsf{LeakPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Bob})$
- $\mathsf{Key}(\mathsf{Alice}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$
- $\mathsf{Key}(\mathsf{Bob}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$

By assumption, sampling a random secret key k_R from $\mathsf{unif}_{\mathsf{key}}$ and returning g^{k_R} is the same as sampling from $\mathsf{unif}_{\mathsf{msg}}$ directly, so we can simplify the channel SharedKey as follows:

- $\mathsf{SecretKey}(\mathsf{Alice}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SecretKey}(\mathsf{Bob}) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SharedKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$
- $\mathsf{PublicKey}(\mathsf{Alice}) := k_A \leftarrow \mathsf{SecretKey}(\mathsf{Alice});\ \mathsf{ret}\ \mathsf{g}^{k_A}$
- $\mathsf{PublicKey}(\mathsf{Bob}) := k_B \leftarrow \mathsf{SecretKey}(\mathsf{Bob});\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Alice})$
- $\mathsf{LeakPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{PublicKey}(\mathsf{Bob})$
- $\mathsf{Key}(\mathsf{Alice}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Bob})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$
- $\mathsf{Key}(\mathsf{Bob}) := \_ \leftarrow \mathsf{OkPublicKey}(\mathsf{Alice})^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{SharedKey}$

This is precisely the simplified composition of the ideal functionality and the simulator from the beginning of this section.

5 Diffie-Hellman Key Exchange (DHKE) + One-Time Pad (OTP)

We now use the Diffie-Hellman Key Exchange protocol from Section 4 to turn an authenticated channel into a one-time pad that delivers a single secret message from Alice to Bob.

Formally, we assume a type key of secret keys (elements of $\{0, \ldots, p-1\}$, where p is a prime); a type msg of messages, also serving as public keys (elements of a cyclic group $G = \{g^0, \ldots, g^{p-1}\}$); a uniform distribution $\mathsf{unif}_{\mathsf{key}} : 1 \twoheadrightarrow \mathsf{key}$ on keys; a uniform distribution $\mathsf{unif}_{\mathsf{msg}} : 1 \twoheadrightarrow \mathsf{msg}$ on messages; the generator $\mathsf{g} : 1 \to \mathsf{msg}$ of G; the group multiplication function $\mathsf{mul} : \mathsf{msg} \times \mathsf{msg} \to \mathsf{msg}$, where we write $m * k$ in place of $\mathsf{mul}\ (m, k)$; the group inverse function $\mathsf{inv} : \mathsf{msg} \to \mathsf{msg}$, where we write k^{-1} in place of $\mathsf{inv}\ k$; and the exponentiation function $\mathsf{exp} : \mathsf{msg} \times \mathsf{key} \to \mathsf{msg}$ that raises an element of G to the power of $k \in \{0, \ldots, p-1\}$. We will write m^k in place of $\mathsf{exp}\ (m, k)$.

We will structure our proof modularly: in the first step, we establish that the Diffie-Hellman Key Exchange can be replaced with an idealization. In the second step, we prove that the resulting One-Time Pad protocol itself reduces to an idealization.

5.1 The Assumptions

In addition to the assumptions from Section 4, we will need the fact that inverses cancel on the right:

- $m : \mathsf{msg}, k : \mathsf{msg} \vdash (m * k) * k^{-1} = m : \mathsf{msg}$

Additionally, we need to know that that the distribution $\mathsf{unif}_{\mathsf{msg}}$ on messages is invariant under the operation of multiplication with a fixed message (a consequence of $\mathsf{unif}_{\mathsf{msg}}$ being uniform):

- $m : \mathsf{msg} \vdash \big(k \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{ret}\ m * k\big) = \mathsf{unif}_{\mathsf{msg}} : \mathsf{msg}$

5.2 The Ideal Functionality

The ideal functionality reads the input message, leaks a confirmation to the adversary to signal that the message has been received, and, upon the approval from the adversary, outputs the message:

- $\mathsf{LeakMsgRcvd}^{\mathsf{id}}_{\mathsf{adv}} := m \leftarrow \mathsf{In};\ \mathsf{ret}\ \checkmark$
- $\mathsf{Out} := \_ \leftarrow \mathsf{OkMsg}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{read}\ \mathsf{In}$

5.3 The Real Protocol

The real-world protocol consists of Alice, Bob, and three authenticated channels: two for carrying out the Diffie-Hellman Key Exchange, as described in Section 4, and one for communicating the secret message from Alice to Bob. The complete code for Alice is the composition of Alice's key exchange part with the single-reaction protocol

- $\mathsf{Send} := m \leftarrow \mathsf{In};\ k_A \leftarrow \mathsf{Key(Alice)};\ \mathsf{ret}\ m * k_A$

In the above, Alice encrypts the input message by multiplying it with the shared key produced by the key exchange, and sends the resulting ciphertext to Bob via the third authenticated channel. The complete code for Bob is the composition of Bob's key exchange part with the single-reaction protocol

- $\mathsf{Out} := c \leftarrow \mathsf{Recv};\ k_B \leftarrow \mathsf{Key(Bob)};\ \mathsf{ret}\ c * k_B^{-1}$

In the above, Bob decrypts the ciphertext received from Alice by multiplying it with the inverse of the shared key, and outputs the result. For communicating the ciphertext from Alice to Bob through the adversary, we have the third authenticated channel:

- $\mathsf{LeakCtxt}^{\mathsf{net}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Send}$
- $\mathsf{Recv} := \_ \leftarrow \mathsf{OkCtxt}^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{Send}$

The real protocol is a composition of the two parties and the three authenticated channels, followed by the hiding of the internal communication.

5.4 The Simulator

The inputs to the simulator are channels $\mathsf{OkPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}}$, $\mathsf{OkPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}}$, $\mathsf{OkCtxt}^{\mathsf{adv}}_{\mathsf{net}}$, $\mathsf{LeakMsgRcvd}^{\mathsf{id}}_{\mathsf{adv}}$, whereas the outputs are channels $\mathsf{LeakPublicKey(Alice)}^{\mathsf{adv}}_{\mathsf{net}}$, $\mathsf{LeakPublicKey(Bob)}^{\mathsf{adv}}_{\mathsf{net}}$, $\mathsf{LeakCtxt}^{\mathsf{net}}_{\mathsf{adv}}$, $\mathsf{OkMsg}^{\mathsf{adv}}_{\mathsf{id}}$. We define the simulator as the composition of the key exchange simulator and the protocol

- $\mathsf{LeakCtxt}^{\mathsf{net}}_{\mathsf{adv}} := \_ \leftarrow \mathsf{LeakMsgRcvd}^{\mathsf{id}}_{\mathsf{adv}};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{OkMsg}^{\mathsf{adv}}_{\mathsf{id}} := \_ \leftarrow \mathsf{OkCtxt}^{\mathsf{adv}}_{\mathsf{net}};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ \_ \leftarrow \mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{ret}\ \checkmark$

followed by the hiding of the channels $\mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}}$ and $\mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}}$.

5.5 Real ≈ Ideal + Simulator: One-Time Pad

Plugging the simulator into the ideal functionality and substituting away the internal channels $\mathsf{LeakMsgRcvd}^{\mathsf{id}}_{\mathsf{adv}}$ and $\mathsf{OkMsg}^{\mathsf{adv}}_{\mathsf{id}}$ that originally served as a line of communication for the adversary yields the composition of the key exchange simulator with the protocol

- $\mathsf{LeakCtxt}^{\mathsf{net}}_{\mathsf{adv}} := m \leftarrow \mathsf{In};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{Out} := \_ \leftarrow \mathsf{OkCtxt}^{\mathsf{adv}}_{\mathsf{net}};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ \_ \leftarrow \mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{read\ In}$

followed by the hiding of the channels $\mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}}$ and $\mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}}$.

Next we move on to simplifying the real protocol. The real protocol can be refactored as the composition of the key exchange part and the protocol

- $\mathsf{Send} := m \leftarrow \mathsf{In};\ k_A \leftarrow \mathsf{Key(Alice)};\ \mathsf{ret}\ m * k_A$
- $\mathsf{LeakCtxt}^{\mathsf{net}}_{\mathsf{adv}} := \mathsf{read\ Send}$
- $\mathsf{Recv} := \_ \leftarrow \mathsf{OkCtxt}^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read\ Send}$
- $\mathsf{Out} := c \leftarrow \mathsf{Recv};\ k_B \leftarrow \mathsf{Key(Bob)};\ \mathsf{ret}\ c * k_B^{-1}$

where the channels Send and Recv are internal, followed by the hiding of the channels Key(Alice) and Key(Bob).

The channels Send and Recv can be substituted away, turning our protocol into the composition of the key exchange part and the protocol

- $\mathsf{LeakCtxt}^{\mathsf{net}}_{\mathsf{adv}} := m \leftarrow \mathsf{In};\ k_A \leftarrow \mathsf{Key(Alice)};\ \mathsf{ret}\ m * k_A$
- $\mathsf{Out} := \_ \leftarrow \mathsf{OkCtxt}^{\mathsf{adv}}_{\mathsf{net}};\ m \leftarrow \mathsf{In};\ k_A \leftarrow \mathsf{Key(Alice)};\ k_B \leftarrow \mathsf{Key(Bob)};\ \mathsf{ret}\ (m * k_A) * k_B^{-1}$

followed by the hiding of the channels Key(Alice) and Key(Bob).

We can approximately rewrite the key exchange part as the composition of the key exchange idealization and the key exchange simulator, followed by the hiding of the channels $\mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}}$ and $\mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}}$. Thus, we can express the real protocol as the composition of the key exchange idealization, the key exchange simulator, and the protocol

- $\mathsf{LeakCtxt}^{\mathsf{net}}_{\mathsf{adv}} := m \leftarrow \mathsf{In};\ k_A \leftarrow \mathsf{Key(Alice)};\ \mathsf{ret}\ m * k_A$
- $\mathsf{Out} := \_ \leftarrow \mathsf{OkCtxt}^{\mathsf{adv}}_{\mathsf{net}};\ m \leftarrow \mathsf{In};\ k_A \leftarrow \mathsf{Key(Alice)};\ k_B \leftarrow \mathsf{Key(Bob)};\ \mathsf{ret}\ (m * k_A) * k_B^{-1}$

all followed by the hiding of the channels $\mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}}$, $\mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}}$, and Key(Alice), Key(Bob).

Composing the above protocol snippet with the key exchange idealization, and substituting away the internal channels Key(Alice) and Key(Bob) that originally served as a line of communication between the key exchange part and the one-time pad yields the protocol

- $\mathsf{SharedKey} := \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{LeakCtxt}^{\mathsf{net}}_{\mathsf{adv}} := m \leftarrow \mathsf{In};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ k \leftarrow \mathsf{SharedKey};\ \mathsf{ret}\ m * k$
- $\mathsf{Out} := \_ \leftarrow \mathsf{OkCtxt}^{\mathsf{adv}}_{\mathsf{net}};\ m \leftarrow \mathsf{In};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ \_ \leftarrow \mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}};\ k \leftarrow \mathsf{SharedKey};\ \mathsf{ret}\ (m * k) * k^{-1}$

where the channel $\mathsf{SharedKey}$ is internal. Canceling out k with its inverse in the channel Out yields

- $\mathsf{Out} := \_ \leftarrow \mathsf{OkCtxt}^{\mathsf{adv}}_{\mathsf{net}};\ m \leftarrow \mathsf{In};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ \_ \leftarrow \mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}};\ k \leftarrow \mathsf{SharedKey};\ \mathsf{ret}\ m$

Dropping the gratuitous dependency on the channel $\mathsf{SharedKey}$ yields

- $\mathsf{Out} := \_ \leftarrow \mathsf{OkCtxt}^{\mathsf{adv}}_{\mathsf{net}};\ m \leftarrow \mathsf{In};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ \_ \leftarrow \mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{ret}\ m$

which simplifies to

- $\mathsf{Out} := \_ \leftarrow \mathsf{OkCtxt}^{\mathsf{adv}}_{\mathsf{net}};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ \_ \leftarrow \mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{read\ In}$

Summarizing, the cleaned-up version of the real protocol is the composition of the the key exchange simulator and the protocol

- $\mathsf{SharedKey} := \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{LeakCtxt}^{\mathsf{net}}_{\mathsf{adv}} := m \leftarrow \mathsf{In};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ k \leftarrow \mathsf{SharedKey};\ \mathsf{ret}\ m * k$
- $\mathsf{Out} := \_ \leftarrow \mathsf{OkCtxt}^{\mathsf{adv}}_{\mathsf{net}};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ \_ \leftarrow \mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{read\ In}$

where the channel $\mathsf{SharedKey}$ is internal, followed by the hiding of the channels $\mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}}$ and $\mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}}$.

Since the channel $\mathsf{SharedKey}$ is now only used in the channel $\mathsf{LeakCtxt}^{\mathsf{net}}_{\mathsf{adv}}$, we can fold it in:

- $\mathsf{LeakCtxt}^{\mathsf{net}}_{\mathsf{adv}} := m \leftarrow \mathsf{In};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ k \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{ret}\ m * k$
- $\mathsf{Out} := \_ \leftarrow \mathsf{OkCtxt}^{\mathsf{adv}}_{\mathsf{net}};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ \_ \leftarrow \mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{read\ In}$

By assumption, the distribution $\mathsf{unif}_{\mathsf{msg}}$ on messages is invariant under the operation of multiplication with a fixed message, so we can simplify the channel $\mathsf{LeakCtxt}^{\mathsf{net}}_{\mathsf{adv}}$ as follows:

- $\mathsf{LeakCtxt}^{\mathsf{net}}_{\mathsf{adv}} := m \leftarrow \mathsf{In};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{samp\ unif}_{\mathsf{msg}}$

Thus, the final version of the real protocol is the composition of the key exchange simulator and the protocol

- $\mathsf{LeakCtxt}^{\mathsf{net}}_{\mathsf{adv}} := m \leftarrow \mathsf{In};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{Out} := \_ \leftarrow \mathsf{OkCtxt}^{\mathsf{adv}}_{\mathsf{net}};\ \_ \leftarrow \mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}};\ \_ \leftarrow \mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{read\ In}$

followed by the hiding of the channels $\mathsf{OkKey(Alice)}^{\mathsf{adv}}_{\mathsf{id}}$ and $\mathsf{OkKey(Bob)}^{\mathsf{adv}}_{\mathsf{id}}$. But this is precisely the simplified composition of the ideal functionality and the simulator from the beginning of this section.

6 Public-Key Encryption: El Gamal

The El Gamal protocol is a CPA-secure public-key encryption schema based on the Diffie-Hellman Key Exchange. As in our previous case studies, Alice wants to communicate q messages to Bob using an authenticated channel. In the symmetric-key setting, the two parties had to agree on a secret shared key that was used both for encryption and decryption. In the public-key setting of El Gamal, Bob generates two keys: the public key is used for encryption and is known to both Alice and the adversary, whereas the secret key is used for decryption and known only to Bob.

Formally, we assume a type key of secret keys (elements of $\{0, \ldots, p-1\}$, where p is a prime); a type msg of messages, also serving as public keys (elements of a cyclic group $G = \{g^0, \ldots, g^{p-1}\}$); a uniform distribution $\mathsf{unif}_{\mathsf{key}} : 1 \twoheadrightarrow \mathsf{key}$ on keys; a uniform distribution $\mathsf{unif}_{\mathsf{msg}} : 1 \twoheadrightarrow \mathsf{msg}$ on messages; the generator $\mathsf{g} : 1 \rightarrow \mathsf{msg}$ of G; the group multiplication function $\mathsf{mul} : \mathsf{msg} \times \mathsf{msg} \rightarrow \mathsf{msg}$, where we write $m * k$ in place of $\mathsf{mul}\ (m, k)$; the group inverse function $\mathsf{inv} : \mathsf{msg} \rightarrow \mathsf{msg}$, where we write k^{-1} in place of $\mathsf{inv}\ k$; and the exponentiation function $\mathsf{exp} : \mathsf{msg} \times \mathsf{key} \rightarrow \mathsf{msg}$ that raises an element of G to the power of $k \in \{0, \ldots, p-1\}$. We will write m^k in place of $\mathsf{exp}\ (m, k)$.

6.1 The Assumptions

At the level of expressions, we only need to know that exponents commute and inverses cancel on the right:

- $k : \mathsf{key}, l : \mathsf{key} \vdash (\mathsf{g}^l)^k = (\mathsf{g}^k)^l : \mathsf{msg}$, and
- $m : \mathsf{msg}, k : \mathsf{key} \vdash (m * k) * k^{-1} = m : \mathsf{msg}$

At the level of distributions, we need to know that sampling a random secret key k from $\mathsf{unif}_{\mathsf{key}}$ and returning g^k is the same as sampling from $\mathsf{unif}_{\mathsf{msg}}$ directly (a consequence of $\mathsf{unif}_{\mathsf{key}}$ being uniform),

- $\cdot \vdash \left(k \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^k\right) = \mathsf{unif}_{\mathsf{msg}} : \mathsf{msg}$

and that the distribution $\mathsf{unif}_{\mathsf{msg}}$ on messages is invariant under the operation of multiplication with a fixed message (a consequence of $\mathsf{unif}_{\mathsf{msg}}$ being uniform),

- $m : \mathsf{msg} \vdash \left(k \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{ret}\ m * k\right) = \mathsf{unif}_{\mathsf{msg}} : \mathsf{msg}$.

Finally, at the level of protocols we need the *decisional Diffie-Hellman (DDH)* cryptographic assumption: as long as the secret keys k, l are generated uniformly, even if the adversary knows the values g^k and g^l, they will be unable to distinguish $(g^k)^l$ from a uniformly generated element of G. The corresponding protocol-level axiom states that in the channel context $\mathsf{DDH}_3 : \mathsf{msg} \times (\mathsf{msg} \times \mathsf{msg})$, the single-reaction no-input protocol

- $\mathsf{DDH}_3 := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \left(\mathsf{g}^{k_B}, \left(\mathsf{g}^{k_A}, (\mathsf{g}^{k_B})^{k_A}\right)\right)$

rewrites approximately to the protocol

- $\mathsf{DDH}_3 := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ k_R \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \left(\mathsf{g}^{k_B}, \left(\mathsf{g}^{k_A}, \mathsf{g}^{k_R}\right)\right)$

We now show that if the DDH assumption holds, the protocol

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{DDH}_2 := p_B \leftarrow \mathsf{PublicKey};\ k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \left(\mathsf{g}^{k_A}, p_B^{k_A}\right)$

rewrites approximately to the protocol

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{DDH}_2 := k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ k_R \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \left(\mathsf{g}^{k_A}, \mathsf{g}^{k_R}\right)$

To show this, we first unfold the key samplings from the channels PublicKey and DDH_2 into new internal channels SecretKey and SecretEphemeralKey, respectively:

- $\mathsf{SecretKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SecretEphemeralKey} := \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{DDH}_2 := p_B \leftarrow \mathsf{PublicKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey};\ \mathsf{ret}\ \left(\mathsf{g}^{k_A}, p_B^{k_A}\right)$

Next we substitute the channel PublicKey into the channel DDH_2, yielding

- $\mathsf{SecretKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SecretEphemeralKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{DDH}_2 := k_B \leftarrow \mathsf{SecretKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey};\ \mathsf{ret}\ \left(\mathsf{g}^{k_A}, (\mathsf{g}^{k_B})^{k_A}\right)$

We subsequently add a gratuitous dependency on the channel SecretEphemeralKey into the channel PublicKey:

- $\mathsf{SecretKey} := \mathsf{samp\ unif_{key}}$
- $\mathsf{SecretEphemeralKey} := \mathsf{samp\ unif_{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey};\ \mathsf{ret\ g}^{k_B}$
- $\mathsf{DDH}_2 := k_B \leftarrow \mathsf{SecretKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey};\ \mathsf{ret}\ \big(\mathsf{g}^{k_A}, (\mathsf{g}^{k_B})^{k_A}\big)$

We can now extract the DDH triple into a new internal channel DDH_3:

- $\mathsf{SecretKey} := \mathsf{samp\ unif_{key}}$
- $\mathsf{SecretEphemeralKey} := \mathsf{samp\ unif_{key}}$
- $\mathsf{DDH}_3 := k_B \leftarrow \mathsf{SecretKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey};\ \mathsf{ret}\ \big(\mathsf{g}^{k_B}, \big(\mathsf{g}^{k_A}, (\mathsf{g}^{k_B})^{k_A}\big)\big)$
- $\mathsf{PublicKey} := \big(p_B, ddh_2\big) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ p_B$
- $\mathsf{DDH}_2 := \big(p_B, ddh_2\big) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ ddh_2$

The internal channels SecretKey and SecretEphemeralKey are now only used in the channel DDH_3, so we can fold them in:

- $\mathsf{DDH}_3 := k_B \leftarrow \mathsf{unif_{key}};\ k_A \leftarrow \mathsf{unif_{key}};\ \mathsf{ret}\ \big(\mathsf{g}^{k_B}, \big(\mathsf{g}^{k_A}, (\mathsf{g}^{k_B})^{k_A}\big)\big)$
- $\mathsf{PublicKey} := \big(p_B, ddh_2\big) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ p_B$
- $\mathsf{DDH}_2 := \big(p_B, ddh_2\big) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ ddh_2$

Using the DDH assumption, we can approximately rewrite the above protocol to

- $\mathsf{DDH}_3 := k_B \leftarrow \mathsf{unif_{key}};\ k_A \leftarrow \mathsf{unif_{key}};\ k_R \leftarrow \mathsf{unif_{key}};\ \mathsf{ret}\ \big(\mathsf{g}^{k_B}, \big(\mathsf{g}^{k_A}, \mathsf{g}^{k_R}\big)\big)$
- $\mathsf{PublicKey} := \big(p_B, ddh_2\big) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ p_B$
- $\mathsf{DDH}_2 := \big(p_B, ddh_2\big) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ ddh_2$

Unfolding the three samplings from the channel DDH_3 into new internal channels SecretKey, SecretEphemeralKey, Key yields

- $\mathsf{SecretKey} := \mathsf{samp\ unif_{key}}$
- $\mathsf{SecretEphemeralKey} := \mathsf{samp\ unif_{key}}$
- $\mathsf{Key} := \mathsf{samp\ unif_{key}}$
- $\mathsf{DDH}_3 := k_B \leftarrow \mathsf{SecretKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey};\ k_R \leftarrow \mathsf{Key};\ \mathsf{ret}\ \big(\mathsf{g}^{k_B}, \big(\mathsf{g}^{k_A}, \mathsf{g}^{k_R}\big)\big)$
- $\mathsf{PublicKey} := \big(p_B, ddh_2\big) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ p_B$
- $\mathsf{DDH}_2 := \big(p_B, ddh_2\big) \leftarrow \mathsf{DDH}_3;\ \mathsf{ret}\ ddh_2$

The internal channel DDH_3 can now be substituted away:

- $\mathsf{SecretKey} := \mathsf{samp\ unif_{key}}$
- $\mathsf{SecretEphemeralKey} := \mathsf{samp\ unif_{key}}$
- $\mathsf{Key} := \mathsf{samp\ unif_{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey};\ k_R \leftarrow \mathsf{Key};\ \mathsf{ret\ g}^{k_B}$
- $\mathsf{DDH}_2 := k_B \leftarrow \mathsf{SecretKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey};\ k_R \leftarrow \mathsf{Key};\ \mathsf{ret}\ \big(\mathsf{g}^{k_A}, \mathsf{g}^{k_R}\big)$

After dropping the gratuitous dependencies – on channels $\mathsf{SecretEphemeralKey}$, Key in the channel $\mathsf{PublicKey}$, and channel $\mathsf{SecretKey}$ in the channel DDH_2 – we end up with the following:

- $\mathsf{SecretKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{SecretEphemeralKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{Key} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{DDH}_2 := k_A \leftarrow \mathsf{SecretEphemeralKey};\ k_R \leftarrow \mathsf{Key};\ \mathsf{ret}\ \left(\mathsf{g}^{k_A}, \mathsf{g}^{k_R}\right)$

The channel $\mathsf{SecretKey}$ is now only used in the channel $\mathsf{PublicKey}$, and the channels $\mathsf{SecretEphemeralKey}$ and Key are only used in the channel DDH_2. Performing the appropriate foldings results in the protocol

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{DDH}_2 := k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ k_R \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \left(\mathsf{g}^{k_A}, \mathsf{g}^{k_R}\right)$

and we are done. Generalizing this argument to q sessions, we have just shown that the protocol

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{DDH}_2(i) := p_B \leftarrow \mathsf{PublicKey};\ k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \left(\mathsf{g}^{k_A}, p_B^{k_A}\right)$ for $0 \leqslant i < q$

rewrites approximately to the protocol

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{DDH}_2(i) := k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ k_R \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \left(\mathsf{g}^{k_A}, \mathsf{g}^{k_R}\right)$ for $0 \leqslant i < q$

6.2 The Ideal Functionality

The ideal functionality reads the input message, leaks a confirmation to the adversary to signal that the message has been received, and, upon the approval from the adversary, outputs the message:

- $\mathsf{LeakMsgRcvd}(i)^{\mathsf{id}}_{\mathsf{adv}} := m \leftarrow \mathsf{In}(i);\ \mathsf{ret}\ \checkmark$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkMsg}(i)^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

6.3 The Real Protocol

The real-world protocol consists of Alice, Bob, and an authenticated channel. Bob randomly generates a secret key k_B visible only to him:

- $\mathsf{SecretKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$

He computes the value g^{k_B} as the public key visible to everybody, including the adversary; we model the adversary's access to the public key by an explicit leakage function:

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$

Using the public key g^{k_B}, Alice encrypts each message by randomly generating a secret ephemeral key k_A and constructing the ciphertext in two parts. The first is the public ephemeral key g^{k_A}, and the second is the encoding of the input message by multiplying it with the session key $(g^{k_B})^{k_A}$:

- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$

- $\mathsf{SessionKey}(i) := p_B \leftarrow \mathsf{PublicKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ p_B^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{Send}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$

The authenticated channel leaks each ciphertext to the adversary and waits for their approval before forwarding it to Bob:

- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{Send}(i)$ for $0 \leqslant i < q$
- $\mathsf{Recv}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{Send}(i)$ for $0 \leqslant i < q$

At last, Bob decrypts the ciphertext by multiplying the message encoding (the second half) by the inverse of the session key $(g^{k_A})^{k_B}$. He constructs the session key from the the public ephemeral key (the first half) and his own secret key k_B:

- $\mathsf{Out}(i) := \big(p_A, c\big) \leftarrow \mathsf{Recv}(i);\ k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ c * \big(p_A^{k_B}\big)^{-1}$ for $0 \leqslant i < q$

Thus, we have the following code for Alice:

- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{SessionKey}(i) := p_B \leftarrow \mathsf{PublicKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ p_B^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{Send}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$

The code for Bob is below:

- $\mathsf{SecretKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{Out}(i) := \big(p_A, c\big) \leftarrow \mathsf{Recv}(i);\ k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ c * \big(p_A^{k_B}\big)^{-1}$ for $0 \leqslant i < q$

Finally, we recall the code for the authenticated channel:

- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read}\ \mathsf{Send}(i)$ for $0 \leqslant i < q$
- $\mathsf{Recv}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{Send}(i)$ for $0 \leqslant i < q$

Composing all of this together and hiding the internal communication yields the real-world protocol.

6.4 The Simulator

The channels $\mathsf{OkCtxt}(-)^{\mathsf{adv}}_{\mathsf{net}}$, $\mathsf{LeakMsgRcvd}(-)^{\mathsf{id}}_{\mathsf{adv}}$ are inputs to the simulator, whereas the channels $\mathsf{OkMsg}(i)^{\mathsf{adv}}_{\mathsf{id}}$, $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}}$, $\mathsf{LeakCtxt}(-)^{\mathsf{adv}}_{\mathsf{net}}$ are the outputs. The simulator constructs the public key and the public ephemeral key exactly as in the real world: by randomly generating the secret key k_A and the secret ephemeral key k_B, and computing g^{k_A} and g^{k_B}. Upon receiving the empty message from the ideal functionality to indicate that a message has been received, the simulator leaks the ciphertext by pairing up the public ephemeral key with a random element of G:

- $\mathsf{SecretKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := \_ \leftarrow \mathsf{LeakMsgRcvd}(i)^{\mathsf{id}}_{\mathsf{adv}};\ c \leftarrow \mathsf{unif}_{\mathsf{msg}};\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, c\big)$ for $0 \leqslant i < q$
- $\mathsf{OkMsg}(i)^{\mathsf{adv}}_{\mathsf{id}} := \mathsf{read}\ \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}}$ for $0 \leqslant i < q$

6.5 Real ≈ Ideal + Simulator

Plugging the simulator into the ideal functionality and substituting away the internal channels $\mathsf{LeakMsgRcvd}(-)^{\mathsf{id}}_{\mathsf{adv}}$ and $\mathsf{OkMsg}(-)^{\mathsf{adv}}_{\mathsf{id}}$ that originally served as a line of communication for the adversary yields the following:

- $\mathsf{SecretKey} := \mathsf{samp\ unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret\ g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp\ unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret\ g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := \_ \leftarrow \mathsf{In}(i);\ c \leftarrow \mathsf{unif}_{\mathsf{msg}};\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, c\big)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read\ In}(i)$ for $0 \leqslant i < q$

Next we move on to simplifying the real protocol. Explicitly, we have the code below:

- $\mathsf{SecretKey} := \mathsf{samp\ unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret\ g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp\ unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret\ g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{SessionKey}(i) := p_B \leftarrow \mathsf{PublicKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ p_B^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{Send}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := \mathsf{read\ Send}(i)$ for $0 \leqslant i < q$
- $\mathsf{Recv}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read\ Send}(i)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \big(p_A, c\big) \leftarrow \mathsf{Recv}(i);\ k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ c * \big(p_A^{k_B}\big)^{-1}$ for $0 \leqslant i < q$

We first substitute away the internal channels $\mathsf{Send}(-)$ and $\mathsf{Recv}(-)$:

- $\mathsf{SecretKey} := \mathsf{samp\ unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret\ g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp\ unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret\ g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{SessionKey}(i) := p_B \leftarrow \mathsf{PublicKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ p_B^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ k_B \leftarrow \mathsf{SecretKey};$ $\mathsf{ret}\ (m * k) * \big(p_A^{k_B}\big)^{-1}$ for $0 \leqslant i < q$

Substituting the channels $\mathsf{PublicKey}$, $\mathsf{SessionKey}(i)$, $\mathsf{PublicEphemeralKey}(i)$ into the channel $\mathsf{Out}(i)$ yields

- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ m \leftarrow \mathsf{In}(i);\ k_B \leftarrow \mathsf{SecretKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey}(i);$ $\mathsf{ret}\ \big(m * (\mathsf{g}^{k_B})^{k_A}\big) * \big((\mathsf{g}^{k_A})^{k_B}\big)^{-1}$ for $0 \leqslant i < q$

Since exponents commute, we can rewrite the channel $\mathsf{Out}(i)$ equivalently as

- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ m \leftarrow \mathsf{In}(i);\ k_B \leftarrow \mathsf{SecretKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey}(i);$ $\mathsf{ret}\ \big(m * (\mathsf{g}^{k_A})^{k_B}\big) * \big((\mathsf{g}^{k_A})^{k_B}\big)^{-1}$ for $0 \leqslant i < q$

We can now cancel out the two applications of $*$ to get:

- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ m \leftarrow \mathsf{In}(i);\ k_B \leftarrow \mathsf{SecretKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ m$ for $0 \leqslant i < q$

Dropping the gratuitous dependencies on the channels $\mathsf{SecretKey}$ and $\mathsf{SecretEphemeralKey}(i)$ yields

- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ m \leftarrow \mathsf{In}(i);\ \mathsf{ret}\ m$ for $0 \leqslant i < q$

which simplifies to

- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

To summarize, our cleaned-up version of the real protocol looks as follows:

- $\mathsf{SecretKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{SessionKey}(i) := p_B \leftarrow \mathsf{PublicKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ p_B^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

The channel $\mathsf{SecretKey}$ is now only used in $\mathsf{PublicKey}$, so we can fold it in:

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{SessionKey}(i) := p_B \leftarrow \mathsf{PublicKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ p_B^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

Adding a gratuitous dependency on the channel $\mathsf{PublicKey}$ into the channel $\mathsf{PublicEphemeralKey}(i)$ yields

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := p_B \leftarrow \mathsf{PublicKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$ for $0 \leqslant i < q$

- $\mathsf{SessionKey}(i) := p_B \leftarrow \mathsf{PublicKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ p_B^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

We can now combine the construction of the public ephemeral key and the session key into a new internal channel $\mathsf{DDH}_2(i)$:

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{DDH}_2(i) := p_B \leftarrow \mathsf{PublicKey};\ k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \big(\mathsf{g}^{k_A}, p_B^{k_A}\big)$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := \big(p_A, k\big) \leftarrow \mathsf{DDH}_2(i);\ \mathsf{ret}\ p_A$ for $0 \leqslant i < q$
- $\mathsf{SessionKey}(i) := \big(p_A, k\big) \leftarrow \mathsf{DDH}_2(i);\ \mathsf{ret}\ k$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

The channel $\mathsf{SecretEphemeralKey}(i)$ is now only used in $\mathsf{DDH}_2(i)$, so we can fold it in:

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{DDH}_2(i) := p_B \leftarrow \mathsf{PublicKey};\ k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \big(\mathsf{g}^{k_A}, p_B^{k_A}\big)$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := \big(p_A, k\big) \leftarrow \mathsf{DDH}_2(i);\ \mathsf{ret}\ p_A$ for $0 \leqslant i < q$
- $\mathsf{SessionKey}(i) := \big(p_A, k\big) \leftarrow \mathsf{DDH}_2(i);\ \mathsf{ret}\ k$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

As we observed earlier, the protocol snippet

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{DDH}_2(i) := p_B \leftarrow \mathsf{PublicKey};\ k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \big(\mathsf{g}^{k_A}, p_B^{k_A}\big)$ for $0 \leqslant i < q$

rewrites approximately to

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{DDH}_2(i) := k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ k_R \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \big(\mathsf{g}^{k_A}, \mathsf{g}^{k_R}\big)$ for $0 \leqslant i < q$

Our real-world protocol therefore approximately rewrites to the protocol

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{DDH}_2(i) := k_A \leftarrow \mathsf{unif}_{\mathsf{key}};\ k_R \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \big(\mathsf{g}^{k_A}, \mathsf{g}^{k_R}\big)$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := \big(p_A, k\big) \leftarrow \mathsf{DDH}_2(i);\ \mathsf{ret}\ p_A$ for $0 \leqslant i < q$

- $\mathsf{SessionKey}(i) := \big(p_A, k\big) \leftarrow \mathsf{DDH}_2(i);\ \mathsf{ret}\ k$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

Unfolding the two samplings from $\mathsf{DDH}_2(i)$ into new internal channels $\mathsf{SecretEphemeralKey}(i)$ and $\mathsf{Key}(i)$ yields

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{Key}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{DDH}_2(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ k_R \leftarrow \mathsf{Key}(i);\ \mathsf{ret}\ \big(\mathsf{g}^{k_A}, \mathsf{g}^{k_R}\big)$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := \big(p_A, k\big) \leftarrow \mathsf{DDH}_2(i);\ \mathsf{ret}\ p_A$ for $0 \leqslant i < q$
- $\mathsf{SessionKey}(i) := \big(p_A, k\big) \leftarrow \mathsf{DDH}_2(i);\ \mathsf{ret}\ k$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

The channel $\mathsf{DDH}_2(i)$ can now be substituted away:

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{Key}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ k_R \leftarrow \mathsf{Key}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{SessionKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ k_R \leftarrow \mathsf{Key}(i);\ \mathsf{ret}\ \mathsf{g}^{k_R}$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

After dropping the gratuitous dependencies – on channel $\mathsf{Key}(i)$ from the channel $\mathsf{PublicEphemeralKey}(i)$, and channel $\mathsf{SecretEphemeralKey}(i)$ from the channel $\mathsf{SessionKey}(i)$ – we end up with the following:

- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{Key}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{SessionKey}(i) := k_R \leftarrow \mathsf{Key}(i);\ \mathsf{ret}\ \mathsf{g}^{k_R}$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

We now unfold the sampling from $\mathsf{PublicKey}$ into a new internal channel $\mathsf{SecretKey}$:

- $\mathsf{SecretKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{Key}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{SessionKey}(i) := k_R \leftarrow \mathsf{Key}(i);\ \mathsf{ret}\ \mathsf{g}^{k_R}$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

The channel $\mathsf{Key}(i)$ is now only used in $\mathsf{SessionKey}(i)$, so we can fold it in:

- $\mathsf{SecretKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{SessionKey}(i) := k_R \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ \mathsf{g}^{k_R}$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

By assumption, sampling a random secret key k_R from $\mathsf{unif}_{\mathsf{key}}$ and returning g^{k_R} is the same as sampling from $\mathsf{unif}_{\mathsf{msg}}$ directly, so we can simplify the channel $\mathsf{SessionKey}(i)$ as follows:

- $\mathsf{SecretKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{SessionKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ \big(p_A, m * k\big)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

We can now extract the encoding of the input message into a new internal channel $\mathsf{Enc}(i)$:

- $\mathsf{SecretKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ \mathsf{g}^{k_B}$

- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{SessionKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$ for $0 \leqslant i < q$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{SessionKey}(i);\ \mathsf{ret}\ m * k$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := c \leftarrow \mathsf{Enc}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ (p_A, c)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

At this point, the channel $\mathsf{SessionKey}(i)$ is only used in $\mathsf{Enc}(i)$, so we can fold it in:

- $\mathsf{SecretKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{Enc}(i) := m \leftarrow \mathsf{In}(i);\ k \leftarrow \mathsf{unif}_{\mathsf{key}};\ \mathsf{ret}\ m * k$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := c \leftarrow \mathsf{Enc}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ (p_A, c)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

By assumption, the distribution $\mathsf{unif}_{\mathsf{msg}}$ on messages is invariant under the operation of multiplication with a fixed message, so we can simplify the channel $\mathsf{Enc}(i)$ as follows:

- $\mathsf{SecretKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{Enc}(i) := \_ \leftarrow \mathsf{In}(i);\ \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := c \leftarrow \mathsf{Enc}(i);\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ (p_A, c)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

The channel $\mathsf{Enc}(i)$ is now only used in $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}}$, so we can fold it in:

- $\mathsf{SecretKey} := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$
- $\mathsf{PublicKey} := k_B \leftarrow \mathsf{SecretKey};\ \mathsf{ret}\ \mathsf{g}^{k_B}$
- $\mathsf{LeakPublicKey}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{PublicKey}$
- $\mathsf{SecretEphemeralKey}(i) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{key}}$ for $0 \leqslant i < q$
- $\mathsf{PublicEphemeralKey}(i) := k_A \leftarrow \mathsf{SecretEphemeralKey}(i);\ \mathsf{ret}\ \mathsf{g}^{k_A}$ for $0 \leqslant i < q$
- $\mathsf{LeakCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}} := \_ \leftarrow \mathsf{In}(i);\ c \leftarrow \mathsf{unif}_{\mathsf{key}};\ p_A \leftarrow \mathsf{PublicEphemeralKey}(i);\ \mathsf{ret}\ (p_A, c)$ for $0 \leqslant i < q$
- $\mathsf{Out}(i) := \_ \leftarrow \mathsf{OkCtxt}(i)^{\mathsf{adv}}_{\mathsf{net}};\ \mathsf{read}\ \mathsf{In}(i)$ for $0 \leqslant i < q$

But this is precisely the simplified composition of the ideal functionality and the simulator from the beginning of this section.

7 Oblivious Transfer: 1-Out-Of-2 Pre-Processing

In this case study, Alice and Bob carry out a 1-Out-Of-2 Oblivious Transfer (OT) separated into an *offline* phase, where Alice and Bob exchange a key using a single idealized 1-Out-Of-2 OT instance, and an *online* phase that relies on the shared key and requires no cryptographic assumptions at all, thereby being very fast. We prove the protocol semi-honest secure in the case when the receiver is corrupt. Formally, we assume a type msg of messages; a coin-flip distribution $\mathsf{flip} : 1 \twoheadrightarrow \mathsf{Bool}$; a uniform distribution $\mathsf{unif}_{\mathsf{msg}} : 1 \twoheadrightarrow \mathsf{msg}$ on messages; and a bitwise xor function $\oplus_{\mathsf{msg}} : \mathsf{msg} \times \mathsf{msg} \to \mathsf{msg}$ on messages. We will write $\oplus\,(x, y)$ as $x \oplus y$.

7.1 The Assumptions

At the expression level, we assume that the operation of bitwise xor with a fixed message is self-inverse:

- $x : \mathsf{msg}, y : \mathsf{msg} \vdash (x \oplus y) \oplus y = x : \mathsf{msg}$.

At the distribution level, we assume that the distribution $\mathsf{unif}_{\mathsf{msg}}$ on messages is invariant under the operation of xor-ing with a fixed message (a consequence of $\mathsf{unif}_{\mathsf{msg}}$ being uniform):

- $x : \mathsf{msg} \vdash \big(y \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{ret}\ x \oplus y\big) = \mathsf{unif}_{\mathsf{msg}} : \mathsf{msg}$.

7.2 The Ideal Functionality

In its basic form, the ideal functionality reads two messages m_0, m_1 from the sender, and one Boolean c from the receiver, and outputs the following message:

$$\begin{cases} m_0 & \text{if } c = \mathsf{false} \\ m_1 & \text{if } c = \mathsf{true} \end{cases}$$

In code:

- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0$

Since the sender is honest, only the timing information for their messages is leaked:

- $\mathsf{MsgRcvd}(0)^{\mathsf{id}}_{\mathsf{adv}} := m_0 \leftarrow \mathsf{Msg}(0);\ \mathsf{ret}\ \checkmark$
- $\mathsf{MsgRcvd}(1)^{\mathsf{id}}_{\mathsf{adv}} := m_1 \leftarrow \mathsf{Msg}(1);\ \mathsf{ret}\ \checkmark$

Since the receiver is semi-honest, the choice as well as selected message are leaked to the adversary:

- $\mathsf{Chc}^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Chc}$
- $\mathsf{Out}^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}$

7.3 The Real Protocol

For the offline phase, we assume an ideal OT functionality. Alice randomly generates a new pair of messages, to be treated as keys, and sends them to the OT functionality:

- $\mathsf{Key}(0) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$
- $\mathsf{Key}(1) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$
- $\mathsf{OTMsg}(0) := \mathsf{read}\ \mathsf{Key}(0)$
- $\mathsf{OTMsg}(1) := \mathsf{read}\ \mathsf{Key}(1)$

Bob flips a coin to decide which key he will ask for and informs the adversary:

- $\mathsf{Flip} := \mathsf{samp}\ \mathsf{flip}$

- $\mathsf{Flip}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Flip}$

He subsequently sends his choice to the OT functionality:

- $\mathsf{OTChc} := \mathsf{read}\ \mathsf{Flip}$

The OT functionality selects the corresponding key and sends it to Bob, accompanied by the requisite leakages:

- $\mathsf{OTOut} := m_0 \leftarrow \mathsf{OTMsg}(0);\ m_1 \leftarrow \mathsf{OTMsg}(1);\ c \leftarrow \mathsf{OTChc};\ \mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0$
- $\mathsf{OTMsgRcvd}(0)^{\mathsf{ot}}_{\mathsf{adv}} := m_0 \leftarrow \mathsf{OTMsg}(0);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTMsgRcvd}(1)^{\mathsf{ot}}_{\mathsf{adv}} := m_1 \leftarrow \mathsf{OTMsg}(1);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChc}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}$
- $\mathsf{OTOut}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}$

Bob stores the result of the OT exchange as the key shared between him and Alice:

- $\mathsf{SharedKey} := \mathsf{read}\ \mathsf{OTOut}$

This ends the offline phase. The online phase starts by Bob's informing the adversary about his choice of message:

- $\mathsf{Chc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Chc}$

Bob subsequently encrypts this choice by xor-ing it with the shared key established in the pre-processing phase, and sends the encryption to Alice while leaking its value:

- $\mathsf{ChcEnc} := f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{false}\ \mathsf{else}\ \mathsf{true}\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{true}\ \mathsf{else}\ \mathsf{false}\big)$

Upon receiving Bob's encrypted choice, Alice encrypts her messages by bitwise xor-ing them with the keys - either their own respective keys in case Bob's encrypted choice is false, or the mutually-swapped keys if Bob's encrypted choice is true:

- $\mathsf{MsgEnc}(0) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ e \leftarrow \mathsf{ChcEnc};$
  $\mathsf{if}\ e\ \mathsf{then}\ \mathsf{ret}\ m_0 \oplus k_1\ \mathsf{else}\ \mathsf{ret}\ m_0 \oplus k_0$
- $\mathsf{MsgEnc}(1) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ e \leftarrow \mathsf{ChcEnc};$
  $\mathsf{if}\ e\ \mathsf{then}\ \mathsf{ret}\ m_1 \oplus k_0\ \mathsf{else}\ \mathsf{ret}\ m_1 \oplus k_1$

After receiving Alice's encrypted messages, Bob leaks them to the adversary:

- $\mathsf{MsgEnc}(0)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{MsgEnc}(0)$
- $\mathsf{MsgEnc}(1)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{MsgEnc}(1)$

He then selects the encryption of the message he wants, decrypts it by xor-ing it with the shared key, and outputs the result while leaking its value:

- $\mathsf{Out} := e_0 \leftarrow \mathsf{MsgEnc}(0);\ e_1 \leftarrow \mathsf{MsgEnc}(1);\ k \leftarrow \mathsf{SharedKey};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ e_1 \oplus k\ \mathsf{else}\ \mathsf{ret}\ e_0 \oplus k$
- $\mathsf{Out}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}$

Thus, we have the following code for Alice:

- $\mathsf{Key}(0) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$
- $\mathsf{Key}(1) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$
- $\mathsf{OTMsg}(0) := \mathsf{read}\ \mathsf{Key}(0)$
- $\mathsf{OTMsg}(1) := \mathsf{read}\ \mathsf{Key}(1)$

- $\mathsf{MsgEnc}(0) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ e \leftarrow \mathsf{ChcEnc};$ $\mathsf{if}\ e\ \mathsf{then\ ret}\ m_0 \oplus k_1\ \mathsf{else\ ret}\ m_0 \oplus k_0$
- $\mathsf{MsgEnc}(1) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ e \leftarrow \mathsf{ChcEnc};$ $\mathsf{if}\ e\ \mathsf{then\ ret}\ m_1 \oplus k_0\ \mathsf{else\ ret}\ m_1 \oplus k_1$

The code for Bob has the following form:

- $\mathsf{Flip} := \mathsf{samp\ flip}$
- $\mathsf{Flip}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTChc} := \mathsf{read\ Flip}$
- $\mathsf{SharedKey} := \mathsf{read\ OTOut}$
- $\mathsf{Chc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Chc}$
- $\mathsf{ChcEnc} := f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then\ false\ else\ true}\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then\ true\ else\ false}\big)$
- $\mathsf{ChcEnc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ ChcEnc}$
- $\mathsf{MsgEnc}(0)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ MsgEnc}(0)$
- $\mathsf{MsgEnc}(1)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ MsgEnc}(1)$
- $\mathsf{Out} := e_0 \leftarrow \mathsf{MsgEnc}(0);\ e_1 \leftarrow \mathsf{MsgEnc}(1);\ k \leftarrow \mathsf{SharedKey};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then\ ret}\ e_1 \oplus k\ \mathsf{else\ ret}\ e_0 \oplus k$
- $\mathsf{Out}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Out}$

Finally, we recall the code for the OT functionality:

- $\mathsf{OTOut} := m_0 \leftarrow \mathsf{OTMsg}(0);\ m_1 \leftarrow \mathsf{OTMsg}(1);\ c \leftarrow \mathsf{OTChc};\ \mathsf{if}\ c\ \mathsf{then\ ret}\ m_1\ \mathsf{else\ ret}\ m_0$
- $\mathsf{OTMsgRcvd}(0)^{\mathsf{ot}}_{\mathsf{adv}} := m_0 \leftarrow \mathsf{OTMsg}(0);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTMsgRcvd}(1)^{\mathsf{ot}}_{\mathsf{adv}} := m_1 \leftarrow \mathsf{OTMsg}(1);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChc}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ OTChc}$
- $\mathsf{OTOut}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ OTOut}$

Composing all of this together and hiding the internal communication yields the real-world protocol.

7.4 Real = Ideal + Simulator

Our goal is to simplify the real protocol until it becomes clear how to separate it out into the ideal functionality part and the simulator part. We recall the code:

- $\mathsf{Key}(0) := \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{Key}(1) := \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{Flip} := \mathsf{samp\ flip}$
- $\mathsf{Flip}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTMsg}(0) := \mathsf{read\ Key}(0)$
- $\mathsf{OTMsg}(1) := \mathsf{read\ Key}(1)$
- $\mathsf{OTChc} := \mathsf{read\ Flip}$
- $\mathsf{OTOut} := m_0 \leftarrow \mathsf{OTMsg}(0);\ m_1 \leftarrow \mathsf{OTMsg}(1);\ c \leftarrow \mathsf{OTChc};\ \mathsf{if}\ c\ \mathsf{then\ ret}\ m_1\ \mathsf{else\ ret}\ m_0$

- $\mathsf{OTMsgRcvd}(0)^{\mathsf{ot}}_{\mathsf{adv}} := m_0 \leftarrow \mathsf{OTMsg}(0);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTMsgRcvd}(1)^{\mathsf{ot}}_{\mathsf{adv}} := m_1 \leftarrow \mathsf{OTMsg}(1);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChc}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}$
- $\mathsf{OTOut}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}$
- $\mathsf{SharedKey} := \mathsf{read}\ \mathsf{OTOut}$
- $\mathsf{Chc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Chc}$
- $\mathsf{ChcEnc} := f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{false}\ \mathsf{else}\ \mathsf{true}\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{true}\ \mathsf{else}\ \mathsf{false}\big)$
- $\mathsf{ChcEnc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{ChcEnc}$
- $\mathsf{MsgEnc}(0) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ e \leftarrow \mathsf{ChcEnc};$
  $\mathsf{if}\ e\ \mathsf{then}\ \mathsf{ret}\ m_0 \oplus k_1\ \mathsf{else}\ \mathsf{ret}\ m_0 \oplus k_0$
- $\mathsf{MsgEnc}(1) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ e \leftarrow \mathsf{ChcEnc};$
  $\mathsf{if}\ e\ \mathsf{then}\ \mathsf{ret}\ m_1 \oplus k_0\ \mathsf{else}\ \mathsf{ret}\ m_1 \oplus k_1$
- $\mathsf{MsgEnc}(0)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{MsgEnc}(0)$
- $\mathsf{MsgEnc}(1)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{MsgEnc}(1)$
- $\mathsf{Out} := e_0 \leftarrow \mathsf{MsgEnc}(0);\ e_1 \leftarrow \mathsf{MsgEnc}(1);\ k \leftarrow \mathsf{SharedKey};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ e_1 \oplus k\ \mathsf{else}\ \mathsf{ret}\ e_0 \oplus k$
- $\mathsf{Out}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}$

We start by eliminating the internal OT channels by substituting them into the OT leakage channels and the channel SharedKey:

- $\mathsf{Key}(0) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$
- $\mathsf{Key}(1) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$
- $\mathsf{Flip} := \mathsf{samp}\ \mathsf{flip}$
- $\mathsf{Flip}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Flip}$
- $\mathsf{OTMsgRcvd}(0)^{\mathsf{ot}}_{\mathsf{adv}} := k_0 \leftarrow \mathsf{Key}(0);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTMsgRcvd}(1)^{\mathsf{ot}}_{\mathsf{adv}} := k_1 \leftarrow \mathsf{Key}(1);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChc}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Flip}$
- $\mathsf{OTOut}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SharedKey}$
- $\mathsf{SharedKey} := k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ f \leftarrow \mathsf{Flip};\ \mathsf{if}\ f\ \mathsf{then}\ \mathsf{ret}\ k_1\ \mathsf{else}\ \mathsf{ret}\ k_0$
- $\mathsf{Chc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Chc}$
- $\mathsf{ChcEnc} := f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{false}\ \mathsf{else}\ \mathsf{true}\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{true}\ \mathsf{else}\ \mathsf{false}\big)$
- $\mathsf{ChcEnc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{ChcEnc}$
- $\mathsf{MsgEnc}(0) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ e \leftarrow \mathsf{ChcEnc};$
  $\mathsf{if}\ e\ \mathsf{then}\ \mathsf{ret}\ m_0 \oplus k_1\ \mathsf{else}\ \mathsf{ret}\ m_0 \oplus k_0$
- $\mathsf{MsgEnc}(1) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ e \leftarrow \mathsf{ChcEnc};$
  $\mathsf{if}\ e\ \mathsf{then}\ \mathsf{ret}\ m_1 \oplus k_0\ \mathsf{else}\ \mathsf{ret}\ m_1 \oplus k_1$
- $\mathsf{MsgEnc}(0)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{MsgEnc}(0)$

- $\mathsf{MsgEnc}(1)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{MsgEnc}(1)$

- $\mathsf{Out} := e_0 \leftarrow \mathsf{MsgEnc}(0);\ e_1 \leftarrow \mathsf{MsgEnc}(1);\ k \leftarrow \mathsf{SharedKey};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ e_1 \oplus k\ \mathsf{else}\ \mathsf{ret}\ e_0 \oplus k$

- $\mathsf{Out}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}$

Substituting the channel ChcEnc into MsgEnc(0) and MsgEnc(1) yields

- $\mathsf{MsgEnc}(0) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_0 \oplus k_0\ \mathsf{else}\ \mathsf{ret}\ m_0 \oplus k_1\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_0 \oplus k_1\ \mathsf{else}\ \mathsf{ret}\ m_0 \oplus k_0\big)$

- $\mathsf{MsgEnc}(1) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_1 \oplus k_1\ \mathsf{else}\ \mathsf{ret}\ m_1 \oplus k_0\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_1 \oplus k_0\ \mathsf{else}\ \mathsf{ret}\ m_1 \oplus k_1\big)$

Substituting the channel SharedKey into Out yields

- $\mathsf{Out} := e_0 \leftarrow \mathsf{MsgEnc}(0);\ e_1 \leftarrow \mathsf{MsgEnc}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ e_1 \oplus k_1\ \mathsf{else}\ \mathsf{ret}\ e_0 \oplus k_1\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ e_1 \oplus k_0\ \mathsf{else}\ \mathsf{ret}\ e_0 \oplus k_0\big)$

Further substituting the channels MsgEnc(0) and MsgEnc(1) into Out yields

- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ (m_1 \oplus k_1) \oplus k_1\ \mathsf{else}\ \mathsf{ret}\ (m_0 \oplus k_1) \oplus k_1\big)$
  $\mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ (m_1 \oplus k_0) \oplus k_0\ \mathsf{else}\ \mathsf{ret}\ (m_0 \oplus k_0) \oplus k_0\big)$

We can now cancel out the two applications of $\oplus$ since they are mutually inverse by assumption:

- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

We can now drop the gratuitous dependencies on channels Key(0), Key(1):

- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

Since both branches are the same, we can also not flip:

- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0$

After dropping the gratuitous dependency on the channel Flip, we get

- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0$

Summarizing, the cleaned-up version of the real protocol is below:

- $\mathsf{Key}(0) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$

- $\mathsf{Key}(1) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$

- $\mathsf{Flip} := \mathsf{samp}\ \mathsf{flip}$

- $\mathsf{Flip}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Flip}$

- $\mathsf{OTMsgRcvd}(0)^{\mathsf{ot}}_{\mathsf{adv}} := k_0 \leftarrow \mathsf{Key}(0);\ \mathsf{ret}\ \checkmark$

- $\mathsf{OTMsgRcvd}(1)^{\mathsf{ot}}_{\mathsf{adv}} := k_1 \leftarrow \mathsf{Key}(1);\ \mathsf{ret}\ \checkmark$

- $\mathsf{OTChc}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Flip}$

- $\mathsf{OTOut}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SharedKey}$

- $\mathsf{SharedKey} := k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ f \leftarrow \mathsf{Flip};\ \mathsf{if}\ f\ \mathsf{then}\ \mathsf{ret}\ k_1\ \mathsf{else}\ \mathsf{ret}\ k_0$

- $\mathsf{Chc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Chc}$

- $\mathsf{ChcEnc} := f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then\ false\ else\ true}\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then\ true\ else\ false}\big)$
- $\mathsf{ChcEnc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{ChcEnc}$
- $\mathsf{MsgEnc}(0) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then\ ret}\ m_0 \oplus k_0\ \mathsf{else\ ret}\ m_0 \oplus k_1\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then\ ret}\ m_0 \oplus k_1\ \mathsf{else\ ret}\ m_0 \oplus k_0\big)$
- $\mathsf{MsgEnc}(1) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then\ ret}\ m_1 \oplus k_1\ \mathsf{else\ ret}\ m_1 \oplus k_0\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then\ ret}\ m_1 \oplus k_0\ \mathsf{else\ ret}\ m_1 \oplus k_1\big)$
- $\mathsf{MsgEnc}(0)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{MsgEnc}(0)$
- $\mathsf{MsgEnc}(1)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{MsgEnc}(1)$
- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then\ ret}\ m_1\ \mathsf{else\ ret}\ m_0$
- $\mathsf{Out}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}$

Since both keys are generated from the same distribution, the coin flip that distinguishes between them can be eliminated (*"decoupling"*). To show this, we introduce an internal channel KeyPair that constructs the pair of two keys, where the first one is shared and the second one is private:

- $\mathsf{Key}(0) := \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{Key}(1) := \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{Flip} := \mathsf{samp\ flip}$
- $\mathsf{Flip}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTMsgRcvd}(0)^{\mathsf{ot}}_{\mathsf{adv}} := k_0 \leftarrow \mathsf{Key}(0);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTMsgRcvd}(1)^{\mathsf{ot}}_{\mathsf{adv}} := k_1 \leftarrow \mathsf{Key}(1);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChc}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTOut}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ SharedKey}$
- $\mathsf{KeyPair} := k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ f \leftarrow \mathsf{Flip};\ \mathsf{if}\ f\ \mathsf{then\ ret}\ (k_1, k_0)\ \mathsf{else\ ret}\ (k_0, k_1)$
- $\mathsf{SharedKey} := \big(k_s, k_p\big) \leftarrow \mathsf{KeyPair};\ \mathsf{ret}\ k_s$
- $\mathsf{Chc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Chc}$
- $\mathsf{ChcEnc} := f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then\ false\ else\ true}\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then\ true\ else\ false}\big)$
- $\mathsf{ChcEnc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{ChcEnc}$
- $\mathsf{MsgEnc}(0) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ \big(k_s, k_p\big) \leftarrow \mathsf{KeyPair};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_0 \oplus k_p\ \mathsf{else\ ret}\ m_0 \oplus k_s$
- $\mathsf{MsgEnc}(1) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ \big(k_s, k_p\big) \leftarrow \mathsf{KeyPair};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_1 \oplus k_s\ \mathsf{else\ ret}\ m_1 \oplus k_p$
- $\mathsf{MsgEnc}(0)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{MsgEnc}(0)$
- $\mathsf{MsgEnc}(1)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{MsgEnc}(1)$
- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_1\ \mathsf{else\ ret}\ m_0$
- $\mathsf{Out}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}$

The internal channels $\mathsf{Key}(0)$ and $\mathsf{Key}(1)$ are now only used in the channel $\mathsf{KeyPair}$. We can therefore fold them in:

- $\mathsf{Flip} := \mathsf{samp\ flip}$
- $\mathsf{Flip}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTMsgRcvd}(0)^{\mathsf{ot}}_{\mathsf{adv}} := k_0 \leftarrow \mathsf{Key}(0);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTMsgRcvd}(1)^{\mathsf{ot}}_{\mathsf{adv}} := k_1 \leftarrow \mathsf{Key}(1);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChc}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTOut}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ SharedKey}$
- $\mathsf{KeyPair} := k_0 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ k_1 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ f \leftarrow \mathsf{Flip};\ \mathsf{if}\ f\ \mathsf{then\ ret}\ (k_1, k_0)\ \mathsf{else\ ret}\ (k_0, k_1)$
- $\mathsf{SharedKey} := (k_s, k_p) \leftarrow \mathsf{KeyPair};\ \mathsf{ret}\ k_s$
- $\mathsf{Chc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Chc}$
- $\mathsf{ChcEnc} := f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then\ false\ else\ true}\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then\ true\ else\ false}\big)$
- $\mathsf{ChcEnc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ ChcEnc}$
- $\mathsf{MsgEnc}(0) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ \big(k_s, k_p\big) \leftarrow \mathsf{KeyPair};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_0 \oplus k_p\ \mathsf{else\ ret}\ m_0 \oplus k_s$
- $\mathsf{MsgEnc}(1) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ \big(k_s, k_p\big) \leftarrow \mathsf{KeyPair};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_1 \oplus k_s\ \mathsf{else\ ret}\ m_1 \oplus k_p$
- $\mathsf{MsgEnc}(0)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ MsgEnc}(0)$
- $\mathsf{MsgEnc}(1)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ MsgEnc}(1)$
- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then\ ret}\ m_1\ \mathsf{else\ ret}\ m_0$
- $\mathsf{Out}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Out}$

Rearranging the order of bindings inside $\mathsf{KeyPair}$ yields

- $\mathsf{KeyPair} := f \leftarrow \mathsf{Flip};\ k_0 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ k_1 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{if}\ f\ \mathsf{then\ ret}\ (k_1, k_0)\ \mathsf{else\ ret}\ (k_0, k_1)$

Since sampling and branching are interchangeable, the three reaction snippets

- $k_0 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ k_1 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{if}\ f\ \mathsf{then\ ret}\ (k_1, k_0)\ \mathsf{else\ ret}\ (k_0, k_1)$
- $\mathsf{if}\ f\ \mathsf{then}\ \big(k_0 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ k_1 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{ret}\ (k_1, k_0)\big)\ \mathsf{else}\ \big(k_0 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ k_1 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{ret}\ (k_0, k_1)\big)$
- $\mathsf{if}\ f\ \mathsf{then}\ \big(k_1 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ k_0 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{ret}\ (k_1, k_0)\big)\ \mathsf{else}\ \big(k_0 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ k_1 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{ret}\ (k_0, k_1)\big)$

are equivalent. But the last snippet amounts to doing the same thing either way, so we might just as well not flip:

- $\mathsf{KeyPair} := f \leftarrow \mathsf{Flip};\ k_0 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ k_1 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{ret}\ (k_0, k_1)$

Getting rid of the gratuitous dependency on the channel Flip gives us

- $\mathsf{KeyPair} := k_0 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ k_1 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{ret}\ (k_0, k_1)$

Unfolding the samplings back thus yields the following protocol:

- $\mathsf{Key}(0) := \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{Key}(1) := \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{Flip} := \mathsf{samp\ flip}$

- $\mathsf{Flip}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTMsgRcvd}(0)^{\mathsf{ot}}_{\mathsf{adv}} := k_0 \leftarrow \mathsf{Key}(0);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTMsgRcvd}(1)^{\mathsf{ot}}_{\mathsf{adv}} := k_1 \leftarrow \mathsf{Key}(1);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChc}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTOut}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ SharedKey}$
- $\mathsf{KeyPair} := k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ \mathsf{ret}\ (k_0, k_1)$
- $\mathsf{SharedKey} := (k_s, k_p) \leftarrow \mathsf{KeyPair};\ \mathsf{ret}\ k_s$
- $\mathsf{Chc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Chc}$
- $\mathsf{ChcEnc} := f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ f\ \mathsf{then}\ (\mathsf{if}\ c\ \mathsf{then\ false\ else\ true})\ \mathsf{else}\ (\mathsf{if}\ c\ \mathsf{then\ true\ else\ false})$
- $\mathsf{ChcEnc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ ChcEnc}$
- $\mathsf{MsgEnc}(0) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ (k_s, k_p) \leftarrow \mathsf{KeyPair};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_0 \oplus k_p\ \mathsf{else\ ret}\ m_0 \oplus k_s$
- $\mathsf{MsgEnc}(1) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ (k_s, k_p) \leftarrow \mathsf{KeyPair};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_1 \oplus k_s\ \mathsf{else\ ret}\ m_1 \oplus k_p$
- $\mathsf{MsgEnc}(0)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ MsgEnc}(0)$
- $\mathsf{MsgEnc}(1)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ MsgEnc}(1)$
- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then\ ret}\ m_1\ \mathsf{else\ ret}\ m_0$
- $\mathsf{Out}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Out}$

The internal channel KeyPair can now be substituted away:

- $\mathsf{Key}(0) := \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{Key}(1) := \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{Flip} := \mathsf{samp\ flip}$
- $\mathsf{Flip}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTMsgRcvd}(0)^{\mathsf{ot}}_{\mathsf{adv}} := k_0 \leftarrow \mathsf{Key}(0);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTMsgRcvd}(1)^{\mathsf{ot}}_{\mathsf{adv}} := k_1 \leftarrow \mathsf{Key}(1);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChc}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTOut}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ SharedKey}$
- $\mathsf{SharedKey} := \mathsf{read\ Key}(0)$
- $\mathsf{Chc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Chc}$
- $\mathsf{ChcEnc} := f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ f\ \mathsf{then}\ (\mathsf{if}\ c\ \mathsf{then\ false\ else\ true})\ \mathsf{else}\ (\mathsf{if}\ c\ \mathsf{then\ true\ else\ false})$
- $\mathsf{ChcEnc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ ChcEnc}$
- $\mathsf{MsgEnc}(0) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_0 \oplus k_1\ \mathsf{else\ ret}\ m_0 \oplus k_0$

- $\mathsf{MsgEnc}(1) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_1 \oplus k_0\ \mathsf{else}\ \mathsf{ret}\ m_1 \oplus k_1$
- $\mathsf{MsgEnc}(0)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{MsgEnc}(0)$
- $\mathsf{MsgEnc}(1)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{MsgEnc}(1)$
- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0$
- $\mathsf{Out}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}$

The second key is now only referenced in the channels $\mathsf{MsgEnc}(0)$ and $\mathsf{MsgEnc}(1)$, where we use it to encrypt either the first or the second message, respectively. This encryption process can be extracted out into a new internal channel $\mathsf{PrivMsg}$:

- $\mathsf{Key}(0) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$
- $\mathsf{Key}(1) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$
- $\mathsf{Flip} := \mathsf{samp}\ \mathsf{flip}$
- $\mathsf{Flip}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Flip}$
- $\mathsf{OTMsgRcvd}(0)^{\mathsf{ot}}_{\mathsf{adv}} := k_0 \leftarrow \mathsf{Key}(0);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTMsgRcvd}(1)^{\mathsf{ot}}_{\mathsf{adv}} := k_1 \leftarrow \mathsf{Key}(1);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChc}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Flip}$
- $\mathsf{OTOut}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SharedKey}$
- $\mathsf{SharedKey} := \mathsf{read}\ \mathsf{Key}(0)$
- $\mathsf{Chc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Chc}$
- $\mathsf{ChcEnc} := f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{false}\ \mathsf{else}\ \mathsf{true}\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then}\ \mathsf{true}\ \mathsf{else}\ \mathsf{false}\big)$
- $\mathsf{ChcEnc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{ChcEnc}$
- $\mathsf{PrivMsg} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_1 \leftarrow \mathsf{Key}(1);\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_0 \oplus k_1\ \mathsf{else}\ \mathsf{ret}\ m_1 \oplus k_1$
- $\mathsf{MsgEnc}(0) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ m_p \leftarrow \mathsf{PrivMsg};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_p\ \mathsf{else}\ \mathsf{ret}\ m_0 \oplus k_0$
- $\mathsf{MsgEnc}(1) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ m_p \leftarrow \mathsf{PrivMsg};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_1 \oplus k_0\ \mathsf{else}\ \mathsf{ret}\ m_p$
- $\mathsf{MsgEnc}(0)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{MsgEnc}(0)$
- $\mathsf{MsgEnc}(1)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{MsgEnc}(1)$
- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0$
- $\mathsf{Out}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}$

We can now fold the internal channel $\mathsf{Key}(1)$ into the channel $\mathsf{PrivMsg}$:

- $\mathsf{Key}(0) := \mathsf{samp}\ \mathsf{unif}_{\mathsf{msg}}$
- $\mathsf{Flip} := \mathsf{samp}\ \mathsf{flip}$
- $\mathsf{Flip}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Flip}$

- $\mathsf{OTMsgRcvd}(0)^{\mathsf{ot}}_{\mathsf{adv}} := k_0 \leftarrow \mathsf{Key}(0);\ \mathsf{ret}\ \checkmark$

- $\mathsf{OTMsgRcvd}(1)^{\mathsf{ot}}_{\mathsf{adv}} := k_1 \leftarrow \mathsf{Key}(1);\ \mathsf{ret}\ \checkmark$

- $\mathsf{OTChc}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ Flip}$

- $\mathsf{OTOut}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ SharedKey}$

- $\mathsf{SharedKey} := \mathsf{read\ Key}(0)$

- $\mathsf{Chc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Chc}$

- $\mathsf{ChcEnc} := f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then\ false\ else\ true}\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then\ true\ else\ false}\big)$

- $\mathsf{ChcEnc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ ChcEnc}$

- $\mathsf{PrivMsg} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_1 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_0 \oplus k_1\ \mathsf{else\ ret}\ m_1 \oplus k_1$

- $\mathsf{MsgEnc}(0) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ m_p \leftarrow \mathsf{PrivMsg};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_p\ \mathsf{else\ ret}\ m_0 \oplus k_0$

- $\mathsf{MsgEnc}(1) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ m_p \leftarrow \mathsf{PrivMsg};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_1 \oplus k_0\ \mathsf{else\ ret}\ m_p$

- $\mathsf{MsgEnc}(0)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ MsgEnc}(0)$

- $\mathsf{MsgEnc}(1)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ MsgEnc}(1)$

- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then\ ret}\ m_1\ \mathsf{else\ ret}\ m_0$

- $\mathsf{Out}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Out}$

Rearranging the order of bindings inside PrivMsg yields

- $\mathsf{PrivMsg} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};\ k_1 \leftarrow \mathsf{unif}_{\mathsf{msg}};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_0 \oplus k_1\ \mathsf{else\ ret}\ m_1 \oplus k_1$

Since sampling and branching are interchangeable, and by assumption $\mathsf{unif}_{\mathsf{msg}}$ is invariant under xor-ing with a fixed message, the three reaction snippets

- $k_1 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{if}\ c\ \mathsf{then\ ret}\ m_0 \oplus k_1\ \mathsf{else\ ret}\ m_1 \oplus k_1$
- $\mathsf{if}\ c\ \mathsf{then}\ \big(k_1 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{ret}\ m_0 \oplus k_1\big)\ \mathsf{else}\ \big(k_1 \leftarrow \mathsf{unif}_{\mathsf{msg}};\ \mathsf{ret}\ m_1 \oplus k_1\big)$
- $\mathsf{if}\ c\ \mathsf{then\ samp\ unif}_{\mathsf{msg}}\ \mathsf{else\ samp\ unif}_{\mathsf{msg}}$

are equivalent. So we may just as well not branch:

- $\mathsf{PrivMsg} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};\ \mathsf{samp\ unif}_{\mathsf{msg}}$

Unfolding the sampling back gives us:

- $\mathsf{Key}(0) := \mathsf{samp\ unif}_{\mathsf{msg}}$

- $\mathsf{Key}(1) := \mathsf{samp\ unif}_{\mathsf{msg}}$

- $\mathsf{Flip} := \mathsf{samp\ flip}$

- $\mathsf{Flip}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Flip}$

- $\mathsf{OTMsgRcvd}(0)^{\mathsf{ot}}_{\mathsf{adv}} := k_0 \leftarrow \mathsf{Key}(0);\ \mathsf{ret}\ \checkmark$

- $\mathsf{OTMsgRcvd}(1)^{\mathsf{ot}}_{\mathsf{adv}} := k_1 \leftarrow \mathsf{Key}(1);\ \mathsf{ret}\ \checkmark$

- $\mathsf{OTChc}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTOut}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ SharedKey}$
- $\mathsf{SharedKey} := \mathsf{read\ Key}(0)$
- $\mathsf{Chc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Chc}$
- $\mathsf{ChcEnc} := f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then\ false\ else\ true}\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then\ true\ else\ false}\big)$
- $\mathsf{ChcEnc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ ChcEnc}$
- $\mathsf{PrivMsg} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};\ \mathsf{read\ Key}(1)$
- $\mathsf{MsgEnc}(0) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ m_p \leftarrow \mathsf{PrivMsg};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_p\ \mathsf{else\ ret}\ m_0 \oplus k_0$
- $\mathsf{MsgEnc}(1) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ m_p \leftarrow \mathsf{PrivMsg};\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_1 \oplus k_0\ \mathsf{else\ ret}\ m_p$
- $\mathsf{MsgEnc}(0)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ MsgEnc}(0)$
- $\mathsf{MsgEnc}(1)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ MsgEnc}(1)$
- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then\ ret}\ m_1\ \mathsf{else\ ret}\ m_0$
- $\mathsf{Out}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Out}$

The internal channel PrivMsg can now be substituted away, yielding the final version of the real protocol:

- $\mathsf{Key}(0) := \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{Key}(1) := \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{Flip} := \mathsf{samp\ flip}$
- $\mathsf{Flip}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTMsgRcvd}(0)^{\mathsf{ot}}_{\mathsf{adv}} := k_0 \leftarrow \mathsf{Key}(0);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTMsgRcvd}(1)^{\mathsf{ot}}_{\mathsf{adv}} := k_1 \leftarrow \mathsf{Key}(1);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChc}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTOut}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ SharedKey}$
- $\mathsf{SharedKey} := \mathsf{read\ Key}(0)$
- $\mathsf{Chc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Chc}$
- $\mathsf{ChcEnc} := f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then\ false\ else\ true}\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then\ true\ else\ false}\big)$
- $\mathsf{ChcEnc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ ChcEnc}$
- $\mathsf{MsgEnc}(0) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ k_1\ \mathsf{else\ ret}\ m_0 \oplus k_0$
- $\mathsf{MsgEnc}(1) := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ c \leftarrow \mathsf{Chc};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m_1 \oplus k_0\ \mathsf{else\ ret}\ k_1$
- $\mathsf{MsgEnc}(0)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ MsgEnc}(0)$
- $\mathsf{MsgEnc}(1)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ MsgEnc}(1)$
- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then\ ret}\ m_1\ \mathsf{else\ ret}\ m_0$
- $\mathsf{Out}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Out}$

7.5 The Simulator

The channel

- $\mathsf{Out} := m_0 \leftarrow \mathsf{Msg}(0);\ m_1 \leftarrow \mathsf{Msg}(1);\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ c\ \mathsf{then\ ret}\ m_1\ \mathsf{else\ ret}\ m_0$

can now be separated out as coming from the functionality, and the remainder of the protocol is turned into the simulator below. Plugging in the simulator into the ideal functionality and substituting away the internal channels $\mathsf{Chc}^{\mathsf{id}}_{\mathsf{adv}}$ and $\mathsf{Out}^{\mathsf{id}}_{\mathsf{adv}}$ that originally served as a line of communication for the adversary yields the final version of the real protocol, as desired.

- $\mathsf{Key}(0) := \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{Key}(1) := \mathsf{samp\ unif}_{\mathsf{msg}}$
- $\mathsf{Flip} := \mathsf{samp\ flip}$
- $\mathsf{Flip}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTMsgRcvd}(0)^{\mathsf{ot}}_{\mathsf{adv}} := k_0 \leftarrow \mathsf{Key}(0);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTMsgRcvd}(1)^{\mathsf{ot}}_{\mathsf{adv}} := k_1 \leftarrow \mathsf{Key}(1);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChc}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ Flip}$
- $\mathsf{OTOut}^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ SharedKey}$
- $\mathsf{SharedKey} := \mathsf{read\ Key}(0)$
- $\mathsf{Chc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Chc}^{\mathsf{id}}_{\mathsf{adv}}$
- $\mathsf{ChcEnc} := f \leftarrow \mathsf{Flip};\ c \leftarrow \mathsf{Chc};\ \mathsf{if}\ f\ \mathsf{then}\ \big(\mathsf{if}\ c\ \mathsf{then\ false\ else\ true}\big)\ \mathsf{else}\ \big(\mathsf{if}\ c\ \mathsf{then\ true\ else\ false}\big)$
- $\mathsf{ChcEnc}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ ChcEnc}$
- $\mathsf{MsgEnc}(0) := m \leftarrow \mathsf{Out}^{\mathsf{id}}_{\mathsf{adv}};\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ c \leftarrow \mathsf{Chc}^{\mathsf{id}}_{\mathsf{adv}};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ k_1\ \mathsf{else\ ret}\ m \oplus k_0$
- $\mathsf{MsgEnc}(1) := m \leftarrow \mathsf{Out}^{\mathsf{id}}_{\mathsf{adv}};\ k_0 \leftarrow \mathsf{Key}(0);\ k_1 \leftarrow \mathsf{Key}(1);\ c \leftarrow \mathsf{Chc}^{\mathsf{id}}_{\mathsf{adv}};$
  $\mathsf{if}\ c\ \mathsf{then\ ret}\ m \oplus k_0\ \mathsf{else\ ret}\ k_1$
- $\mathsf{MsgEnc}(0)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ MsgEnc}(0)$
- $\mathsf{MsgEnc}(1)^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ MsgEnc}(1)$
- $\mathsf{Out}^{\mathsf{rec}}_{\mathsf{adv}} := \mathsf{read\ Out}^{\mathsf{id}}_{\mathsf{adv}}$

8 Multi-Party Coin Toss

In this section we implement a protocol where $N+2$ parties labeled $0, \dots, N+1$ reach a Boolean consensus. We prove the protocol secure against a malicious attacker in the case when party N is corrupt, party $N+1$ is honest, and any other party is arbitrarily honest or corrupt. Formally, we assume a coin-flip distribution $\mathsf{flip} : 1 \twoheadrightarrow \mathsf{Bool}$ and a Boolean sum function $\oplus : \mathsf{Bool} \times \mathsf{Bool} \to \mathsf{Bool}$, where we write $x \oplus y$ in place of $\oplus\,(x, y)$.

8.1 The Assumptions

At the expression level, we assume that the operation of Boolean sum with a fixed bit is self-inverse:

- $x : \mathsf{Bool}, y : \mathsf{Bool} \vdash (x \oplus y) \oplus y = x : \mathsf{Bool}.$

At the distribution level, we assume that the distribution flip on bits is invariant under the operation of Boolean sum with a fixed bit (as is indeed the case when flip is uniform):

- $x : \mathsf{Bool} \vdash \big(y \leftarrow \mathsf{flip};\ \mathsf{ret}\ x \oplus y\big) = \mathsf{flip} : \mathsf{Bool}$

8.2 The Ideal Protocol

The ideal functionality generates a random Boolean, leaks it to the adversary on behalf of every corrupt party, and, upon the approval from the adversary, outputs it on behalf of every honest party:

- $\mathsf{Flip} := \mathsf{samp\ flip}$
- $\begin{cases} \mathsf{LeakFlip}(i)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read\ Flip} & \text{if } i \leqslant N+1 \text{ corrupt} \\ \mathsf{LeakFlip}(i)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read\ LeakFlip}(i)^{\mathsf{id}}_{\mathsf{adv}} & \text{if } i \leqslant N+1 \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{Out}(i) := \_ \leftarrow \mathsf{Ok}^{\mathsf{adv}}_{\mathsf{id}};\ \mathsf{read\ Flip} & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) := \mathsf{read\ Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

The output of every corrupted party diverges, since in the malicious setting the external outputs of corrupted parties provide no useful information.

8.3 The Real Protocol

We assume that each party has an associated *commitment functionality* that broadcasts information, and that all broadcast communication is visible to the adversary. At the start of the protocol, each honest party i commits to a randomly generated Boolean and sends it to its commitment functionality:

- $\mathsf{Commit}(i) := \mathsf{samp\ flip}$

In the malicious setting, we assume that the adversary supplies inputs to each corrupted party in lieu of the party's own internal computation. Thus, each corrupted party i commits to the Boolean of the adversary's choice:

- $\mathsf{Commit}(i) := \mathsf{read\ AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)}$

To uniformly cover all cases, we assume channels $\mathsf{AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)}$ as inputs to the real protocol, for all $i \leqslant N+1$ even if i is honest; in this case the corresponding input simply goes unused.

Upon receiving the commit from the party, each commitment functionality broadcasts the fact that a commit happened – but not its value – to everybody, including the adversary:

- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark$
- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read\ Committed}(i)$

Each honest party i inductively keeps track of all parties that have already committed:

- $\begin{cases} \mathsf{AllCommitted}(i,0) := \mathsf{ret}\ \checkmark \\ \mathsf{AllCommitted}(i,j+1) := \_ \leftarrow \mathsf{AllCommitted}(i,j);\ c_j \leftarrow \mathsf{Committed}(j);\ \mathsf{ret}\ \checkmark \text{ for } j \leqslant N+1 \end{cases}$

After all parties have committed, each honest party lets the commitment functionality open its commit for everybody else to see:

- $\mathsf{Open}(i) := \_ \leftarrow \mathsf{AllCommitted}(i, N+2);\ \mathsf{ret}\ \checkmark$

A corrupted party i opens its commit when the adversary says so:

- $\mathsf{Open}(i) := \mathsf{read\ AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)}$

We again assume channels $\mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)}$ as inputs to the real protocol for all $i \leqslant N+1$.

Upon receiving the party's decision to open the commit, each commitment functionality broadcasts the value of the commit to everybody, including the adversary:

- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read\ Commit}(i)$
- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read\ Opened}(i)$

Each honest party i inductively sums up the commits of all parties once they have been opened:

- $\begin{cases} \mathsf{SumOpened}(i, 0) := \mathsf{ret\ false} \\ \mathsf{SumOpened}(i, j+1) := x_j \leftarrow \mathsf{SumOpened}(i, j);\ o_j \leftarrow \mathsf{Opened}(j);\ \mathsf{ret}\ x_j \oplus o_j \text{ for } j \leqslant N+1 \end{cases}$

Finally, each honest party i outputs the consensus - the Boolean sum of all commits:

- $\mathsf{Out}(i) := \mathsf{read\ SumOpened}(i, N+2)$

The output of each corrupted party i diverges:

- $\mathsf{Out}(i) := \mathsf{read\ Out}(i)$

Thus, we have the following code for each honest party i:

- $\mathsf{Commit}(i) := \mathsf{samp\ flip}$
- $\begin{cases} \mathsf{AllCommitted}(i, 0) := \mathsf{ret}\ \checkmark \\ \mathsf{AllCommitted}(i, j+1) := \_ \leftarrow \mathsf{AllCommitted}(i, j);\ c_j \leftarrow \mathsf{Committed}(j);\ \mathsf{ret}\ \checkmark \text{ for } j \leqslant N+1 \end{cases}$
- $\mathsf{Open}(i) := \_ \leftarrow \mathsf{AllCommitted}(i, N+2);\ \mathsf{ret}\ \checkmark$
- $\begin{cases} \mathsf{SumOpened}(i, 0) := \mathsf{ret\ false} \\ \mathsf{SumOpened}(i, j+1) := x_j \leftarrow \mathsf{SumOpened}(i, j);\ o_j \leftarrow \mathsf{Opened}(j);\ \mathsf{ret}\ x_j \oplus o_j \text{ for } j \leqslant N+1 \end{cases}$
- $\mathsf{Out}(i) := \mathsf{read\ SumOpened}(i, N+2)$

The code for a corrupted party i has the following form:

- $\mathsf{Commit}(i) := \mathsf{read\ AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)}$
- $\mathsf{Open}(i) := \mathsf{read\ AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)}$
- $\mathsf{Out}(i) := \mathsf{read\ Out}(i)$

Finally, the code for the commitment functionality for party i is below:

- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark$
- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read\ Committed}(i)$
- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read\ Commit}(i)$
- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read\ Opened}(i)$

Composing all of the above together and hiding the internal communication yields the real protocol.

8.4 The Simulator

In the real protocol, the consensus is the Boolean sum of all parties' commits. The simulator, however, gets the value of the consensus from the ideal functionality. To preserve the invariant that the consensus is the sum of all commits, we adjust the last party's commit: it is no longer a random Boolean, but rather the sum of all other commits plus the consensus. Hence, in the simulator, the last commit only happens after all the other commits, unlike in the real world where the last commit has no dependencies. This is okay – the last party is by assumption honest, so there is no leakage that would need to happen right away – but requires some care. Specifically, the announcement that the last party committed must be independent of the timing of the other commits, so we cannot let it actually depend on the last commit as it does in the real world. Instead, we manually postulate no dependencies. The simulator gives the ok message to the functionality once all the commits (except the last, which we explicitly construct) and all the requests to open have been made.

- $\begin{cases} \mathsf{Commit}(i) := \mathsf{samp\ flip} & \text{if } i \leqslant N \text{ honest} \\ \mathsf{Commit}(i) := \mathsf{read\ AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N \text{ corrupt} \end{cases}$

- $\mathsf{LastCommit} := x_{N+1} \leftarrow \mathsf{SumCommit}(N+1);\ f \leftarrow \mathsf{LeakFlip}(N)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_{N+1} \oplus f$
- $\begin{cases} \mathsf{SumCommit}(0) := \mathsf{ret\ false} \\ \mathsf{SumCommit}(j+1) := x_j \leftarrow \mathsf{SumCommit}(j);\ c_j \leftarrow \mathsf{Commit}(j);\ \mathsf{ret}\ x_j \oplus c_j \text{ for } j \leqslant N \end{cases}$
- $\mathsf{SumCommit}(N+2) := x_{N+1} \leftarrow \mathsf{SumCommit}(N+1);\ c_{N+1} \leftarrow \mathsf{LastCommit};\ \mathsf{ret}\ x_{N+1} \oplus c_{N+1}$
- $\mathsf{Committed}(i) := c_j \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark \text{ for } i \leqslant N$
- $\mathsf{Committed}(N+1) := \mathsf{ret}\ \checkmark$
- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Committed}(i) \text{ for } i \leqslant N+1$
- $\begin{cases} \mathsf{Open}(i) := x_{N+1} \leftarrow \mathsf{SumCommit}(N+2);\ \mathsf{ret}\ \checkmark & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Open}(i) := \mathsf{read}\ \mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$
- $\begin{cases} \mathsf{AllOpen}(0) := \mathsf{ret}\ \checkmark \\ \mathsf{AllOpen}(j+1) := \_ \leftarrow \mathsf{AllOpen}(j);\ \_ \leftarrow \mathsf{Open}(j);\ \mathsf{ret}\ \checkmark \text{ for } j \leqslant N+1 \end{cases}$
- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i) \text{ for } i \leqslant N+1$
- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Opened}(i) \text{ for } i \leqslant N+1$
- $\mathsf{Ok}^{\mathsf{adv}}_{\mathsf{id}} := \_ \leftarrow \mathsf{AllOpen}(N+2);\ x_{N+1} \leftarrow \mathsf{SumCommit}(N+1);\ \mathsf{ret}\ \checkmark$

8.5 Real = Ideal + Simulator

In the real protocol, the composition of all commitment functionalities has the following form:

- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark \text{ for } i \leqslant N+1$
- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Committed}(i) \text{ for } i \leqslant N+1$
- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i) \text{ for } i \leqslant N+1$
- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Opened}(i) \text{ for } i \leqslant N+1$

Currently, each honest party i keeps its own track of who committed. This is of course unnecessary, as each party has the same information, so we can add new internal channels $\mathsf{AllCommitted}(-)$ that inductively keep a global track of commitment:

- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark \text{ for } i \leqslant N+1$
- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Committed}(i) \text{ for } i \leqslant N+1$
- $\begin{cases} \mathsf{AllCommitted}(0) := \mathsf{ret}\ \checkmark \\ \mathsf{AllCommitted}(j+1) := \_ \leftarrow \mathsf{AllCommitted}(j);\ c_j \leftarrow \mathsf{Committed}(j);\ \mathsf{ret}\ \checkmark \text{ for } j \leqslant N+1 \end{cases}$
- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i) \text{ for } i \leqslant N+1$
- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Opened}(i) \text{ for } i \leqslant N+1$

In the presence of the above, we can inductively rewrite the code of each honest party i to the following:

- $\mathsf{Commit}(i) := \mathsf{samp\ flip}$
- $\mathsf{AllCommitted}(i,j) := \mathsf{read}\ \mathsf{AllCommitted}(j) \text{ for } j \leqslant N+2$
- $\mathsf{Open}(i) := \_ \leftarrow \mathsf{AllCommitted}(i, N+2);\ \mathsf{ret}\ \checkmark$

- $\begin{cases} \mathsf{SumOpened}(i,0) := \mathsf{ret\ false} \\ \mathsf{SumOpened}(i,j+1) := x_j \leftarrow \mathsf{SumOpened}(i,j);\ o_j \leftarrow \mathsf{Opened}(j);\ \mathsf{ret}\ x_j \oplus o_j \text{ for } j \leqslant N+1 \end{cases}$

- $\mathsf{Out}(i) := \mathsf{read\ SumOpened}(i, N+2)$

After substituting the channel $\mathsf{AllCommitted}(i, N+2)$ into $\mathsf{Open}(i)$, the internal channels $\mathsf{AllCommitted}(i,-)$ become unused and we can eliminate them entirely:

- $\mathsf{Commit}(i) := \mathsf{samp\ flip}$

- $\mathsf{Open}(i) := \_ \leftarrow \mathsf{AllCommitted}(N+2);\ \mathsf{ret}\ \checkmark$

- $\begin{cases} \mathsf{SumOpened}(i,0) := \mathsf{ret\ false} \\ \mathsf{SumOpened}(i,j+1) := x_j \leftarrow \mathsf{SumOpened}(i,j);\ o_j \leftarrow \mathsf{Opened}(j);\ \mathsf{ret}\ x_j \oplus o_j \text{ for } j \leqslant N+1 \end{cases}$

- $\mathsf{Out}(i) := \mathsf{read\ SumOpened}(i, N+2)$

By the same token, we can add new internal channels $\mathsf{SumOpened}(-)$ to the composition of functionalities that inductively keep a global track of the sum of all commits once they have been opened:

- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark \text{ for } i \leqslant N+1$

- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read\ Committed}(i) \text{ for } i \leqslant N+1$

- $\begin{cases} \mathsf{AllCommitted}(0) := \mathsf{ret}\ \checkmark \\ \mathsf{AllCommitted}(j+1) := \_ \leftarrow \mathsf{AllCommitted}(j);\ c_j \leftarrow \mathsf{Committed}(j);\ \mathsf{ret}\ \checkmark \text{ for } j \leqslant N+1 \end{cases}$

- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read\ Commit}(i) \text{ for } i \leqslant N+1$

- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read\ Opened}(i) \text{ for } i \leqslant N+1$

- $\begin{cases} \mathsf{SumOpened}(0) := \mathsf{ret\ false} \\ \mathsf{SumOpened}(j+1) := x_j \leftarrow \mathsf{SumOpened}(j);\ o_j \leftarrow \mathsf{Opened}(j);\ \mathsf{ret}\ x_j \oplus o_j \text{ for } j \leqslant N+1 \end{cases}$

In the presence of the above, we can inductively rewrite the code of each honest party i to the following:

- $\mathsf{Commit}(i) := \mathsf{samp\ flip}$

- $\mathsf{Open}(i) := \_ \leftarrow \mathsf{AllCommitted}(N+2);\ \mathsf{ret}\ \checkmark$

- $\mathsf{SumOpened}(i,j) := \mathsf{read\ SumOpened}(j) \text{ for } j \leqslant N+2$

- $\mathsf{Out}(i) := \mathsf{read\ SumOpened}(i, N+2)$

After substituting the channel $\mathsf{SumOpened}(i, N+2)$ into $\mathsf{Out}(i)$, the internal channels $\mathsf{SumOpened}(i,-)$ become unused and we can eliminate them entirely:

- $\mathsf{Commit}(i) := \mathsf{samp\ flip}$

- $\mathsf{Open}(i) := \_ \leftarrow \mathsf{AllCommitted}(N+2);\ \mathsf{ret}\ \checkmark$

- $\mathsf{Out}(i) := \mathsf{read\ SumOpened}(N+2)$

The combined code for the real protocol after the aforementioned changes is thus as follows:

- $\begin{cases} \mathsf{Commit}(i) := \mathsf{samp\ flip} & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Commit}(i) := \mathsf{read\ AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark \text{ for } i \leqslant N+1$

- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Committed}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{AllCommitted}(0) := \mathsf{ret}\ \checkmark \\ \mathsf{AllCommitted}(j+1) := \_ \leftarrow \mathsf{AllCommitted}(j);\ c_j \leftarrow \mathsf{Committed}(j);\ \mathsf{ret}\ \checkmark \text{ for } j \leqslant N+1 \end{cases}$

- $\begin{cases} \mathsf{Open}(i) := \_ \leftarrow \mathsf{AllCommitted}(N+2);\ \mathsf{ret}\ \checkmark & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Open}(i) := \mathsf{read}\ \mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i)$ for $i \leqslant N+1$

- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Opened}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{SumOpened}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumOpened}(j+1) := x_j \leftarrow \mathsf{SumOpened}(j);\ o_j \leftarrow \mathsf{Opened}(j);\ \mathsf{ret}\ x_j \oplus o_j \text{ for } j \leqslant N+1 \end{cases}$

- $\begin{cases} \mathsf{Out}(i) := \mathsf{read}\ \mathsf{SumOpened}(N+2) & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) := \mathsf{read}\ \mathsf{Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

Instead of summing up the commits once they have been opened, we can sum them up at the beginning, as done in the simulator, using new internal channels $\mathsf{SumCommit}(-)$:

- $\begin{cases} \mathsf{Commit}(i) := \mathsf{samp}\ \mathsf{flip} & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Commit}(i) := \mathsf{read}\ \mathsf{AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\begin{cases} \mathsf{SumCommit}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumCommit}(j+1) := x_j \leftarrow \mathsf{SumCommit}(j);\ c_j \leftarrow \mathsf{Commit}(j);\ \mathsf{ret}\ x_j \oplus c_j \text{ for } j \leqslant N+1 \end{cases}$

- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark$ for $i \leqslant N+1$

- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Committed}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{AllCommitted}(0) := \mathsf{ret}\ \checkmark \\ \mathsf{AllCommitted}(j+1) := \_ \leftarrow \mathsf{AllCommitted}(j);\ c_j \leftarrow \mathsf{Committed}(j);\ \mathsf{ret}\ \checkmark \text{ for } j \leqslant N+1 \end{cases}$

- $\begin{cases} \mathsf{Open}(i) := \_ \leftarrow \mathsf{AllCommitted}(N+2);\ \mathsf{ret}\ \checkmark & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Open}(i) := \mathsf{read}\ \mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i)$ for $i \leqslant N+1$

- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Opened}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{SumOpened}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumOpened}(j+1) := x_j \leftarrow \mathsf{SumOpened}(j);\ o_j \leftarrow \mathsf{Opened}(j);\ \mathsf{ret}\ x_j \oplus o_j \text{ for } j \leqslant N+1 \end{cases}$

- $\begin{cases} \mathsf{Out}(i) := \mathsf{read}\ \mathsf{SumOpened}(N+2) & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) := \mathsf{read}\ \mathsf{Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

In the presence of these new channels, the channels $\mathsf{AllCommitted}(-)$ can be simplified:

- $\begin{cases} \mathsf{Commit}(i) := \mathsf{samp}\ \mathsf{flip} & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Commit}(i) := \mathsf{read}\ \mathsf{AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\begin{cases} \mathsf{SumCommit}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumCommit}(j+1) := x_j \leftarrow \mathsf{SumCommit}(j);\ c_j \leftarrow \mathsf{Commit}(j);\ \mathsf{ret}\ x_j \oplus c_j \text{ for } j \leqslant N+1 \end{cases}$

- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark$ for $i \leqslant N+1$

- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Committed}(i)$ for $i \leqslant N+1$

- $\mathsf{AllCommitted}(j) := c_j \leftarrow \mathsf{SumCommit}(j);\ \mathsf{ret}\ \checkmark$ for $j \leqslant N+2$

- $\begin{cases} \mathsf{Open}(i) := \_ \leftarrow \mathsf{AllCommitted}(N+2);\ \mathsf{ret}\ \checkmark & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Open}(i) := \mathsf{read}\ \mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i)$ for $i \leqslant N+1$

- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Opened}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{SumOpened}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumOpened}(j+1) := x_j \leftarrow \mathsf{SumOpened}(j);\ o_j \leftarrow \mathsf{Opened}(j);\ \mathsf{ret}\ x_j \oplus o_j \text{ for } j \leqslant N+1 \end{cases}$

- $\begin{cases} \mathsf{Out}(i) := \mathsf{read}\ \mathsf{SumOpened}(N+2) & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) := \mathsf{read}\ \mathsf{Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

After substituting the channel $\mathsf{AllCommitted}(N+2)$ into the channels $\mathsf{Open}(i)$ for $i \leqslant N+1$ honest, the internal channels $\mathsf{AllCommitted}(-)$ become unused and we can eliminate them entirely:

- $\begin{cases} \mathsf{Commit}(i) := \mathsf{samp}\ \mathsf{flip} & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Commit}(i) := \mathsf{read}\ \mathsf{AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\begin{cases} \mathsf{SumCommit}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumCommit}(j+1) := x_j \leftarrow \mathsf{SumCommit}(j);\ c_j \leftarrow \mathsf{Commit}(j);\ \mathsf{ret}\ x_j \oplus c_j \text{ for } j \leqslant N+1 \end{cases}$

- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark$ for $i \leqslant N+1$

- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Committed}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{Open}(i) := x_{N+2} \leftarrow \mathsf{SumCommit}(N+2);\ \mathsf{ret}\ \checkmark & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Open}(i) := \mathsf{read}\ \mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i)$ for $i \leqslant N+1$

- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Opened}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{SumOpened}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumOpened}(j+1) := x_j \leftarrow \mathsf{SumOpened}(j);\ o_j \leftarrow \mathsf{Opened}(j);\ \mathsf{ret}\ x_j \oplus o_j \text{ for } j \leqslant N+1 \end{cases}$

- $\begin{cases} \mathsf{Out}(i) := \mathsf{read}\ \mathsf{SumOpened}(N+2) & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) := \mathsf{read}\ \mathsf{Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

Proceeding further, we can keep track of the decisions to open the commits just as the simulator does, using new internal channels $\mathsf{AllOpen}(-)$:

- $\begin{cases} \mathsf{Commit}(i) := \mathsf{samp}\ \mathsf{flip} & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Commit}(i) := \mathsf{read}\ \mathsf{AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\begin{cases} \mathsf{SumCommit}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumCommit}(j+1) := x_j \leftarrow \mathsf{SumCommit}(j);\ c_j \leftarrow \mathsf{Commit}(j);\ \mathsf{ret}\ x_j \oplus c_j \text{ for } j \leqslant N+1 \end{cases}$

- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark$ for $i \leqslant N+1$

- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Committed}(i)$ for $i \leqslant N+1$
- $\begin{cases} \mathsf{Open}(i) := x_{N+2} \leftarrow \mathsf{SumCommit}(N+2);\ \mathsf{ret}\ \checkmark & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Open}(i) := \mathsf{read}\ \mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$
- $\begin{cases} \mathsf{AllOpen}(0) := \mathsf{ret}\ \checkmark \\ \mathsf{AllOpen}(j+1) := \_ \leftarrow \mathsf{AllOpen}(j);\ \_ \leftarrow \mathsf{Open}(j);\ \mathsf{ret}\ \checkmark \text{for } j \leqslant N+1 \end{cases}$
- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i)$ for $i \leqslant N+1$
- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Opened}(i)$ for $i \leqslant N+1$
- $\begin{cases} \mathsf{SumOpened}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumOpened}(j+1) := x_j \leftarrow \mathsf{SumOpened}(j);\ o_j \leftarrow \mathsf{Opened}(j);\ \mathsf{ret}\ x_j \oplus o_j \text{ for } j \leqslant N+1 \end{cases}$
- $\begin{cases} \mathsf{Out}(i) := \mathsf{read}\ \mathsf{SumOpened}(N+2) & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) := \mathsf{read}\ \mathsf{Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

In the presence of these new channels, the channels $\mathsf{SumOpened}(-)$ can be simplified:

- $\begin{cases} \mathsf{Commit}(i) := \mathsf{samp}\ \mathsf{flip} & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Commit}(i) := \mathsf{read}\ \mathsf{AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$
- $\begin{cases} \mathsf{SumCommit}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumCommit}(j+1) := x_j \leftarrow \mathsf{SumCommit}(j);\ c_j \leftarrow \mathsf{Commit}(j);\ \mathsf{ret}\ x_j \oplus c_j \text{ for } j \leqslant N+1 \end{cases}$
- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark$ for $i \leqslant N+1$
- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Committed}(i)$ for $i \leqslant N+1$
- $\begin{cases} \mathsf{Open}(i) := x_{N+2} \leftarrow \mathsf{SumCommit}(N+2);\ \mathsf{ret}\ \checkmark & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Open}(i) := \mathsf{read}\ \mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$
- $\begin{cases} \mathsf{AllOpen}(0) := \mathsf{ret}\ \checkmark \\ \mathsf{AllOpen}(j+1) := \_ \leftarrow \mathsf{AllOpen}(j);\ \_ \leftarrow \mathsf{Open}(j);\ \mathsf{ret}\ \checkmark \text{ for } j \leqslant N+1 \end{cases}$
- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i)$ for $i \leqslant N+1$
- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Opened}(i)$ for $i \leqslant N+1$
- $\mathsf{SumOpened}(j) := \_ \leftarrow \mathsf{AllOpen}(j);\ \mathsf{read}\ \mathsf{SumCommit}(j)$ for $j \leqslant N+2$
- $\begin{cases} \mathsf{Out}(i) := \mathsf{read}\ \mathsf{SumOpened}(N+2) & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) := \mathsf{read}\ \mathsf{Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

After substituting the channel $\mathsf{SumOpened}(N+2)$ into the channels $\mathsf{Out}(i)$ for $i \leqslant N$ honest, the internal channels $\mathsf{SumOpened}(-)$ become unused and we can eliminate them entirely:

- $\begin{cases} \mathsf{Commit}(i) := \mathsf{samp}\ \mathsf{flip} & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Commit}(i) := \mathsf{read}\ \mathsf{AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$
- $\begin{cases} \mathsf{SumCommit}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumCommit}(j+1) := x_j \leftarrow \mathsf{SumCommit}(j);\ c_j \leftarrow \mathsf{Commit}(j);\ \mathsf{ret}\ x_j \oplus c_j \text{ for } j \leqslant N+1 \end{cases}$
- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark$ for $i \leqslant N+1$

- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Committed}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{Open}(i) \coloneqq x_{N+2} \leftarrow \mathsf{SumCommit}(N+2);\ \mathsf{ret}\ \checkmark & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Open}(i) \coloneqq \mathsf{read}\ \mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\begin{cases} \mathsf{AllOpen}(0) \coloneqq \mathsf{ret}\ \checkmark \\ \mathsf{AllOpen}(j+1) \coloneqq \_ \leftarrow \mathsf{AllOpen}(j);\ \_ \leftarrow \mathsf{Open}(j);\ \mathsf{ret}\ \checkmark \text{ for } j \leqslant N+1 \end{cases}$

- $\mathsf{Opened}(i) \coloneqq \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i)$ for $i \leqslant N+1$

- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Opened}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{Out}(i) \coloneqq \_ \leftarrow \mathsf{AllOpen}(N+2);\ \mathsf{read}\ \mathsf{SumCommit}(N+2) & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) \coloneqq \mathsf{read}\ \mathsf{Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

This is the cleaned-up version of the real protocol. Plugging the simulator into the ideal protocol and substituting away the channels $\mathsf{LeakFlip}(N)^{\mathsf{id}}_{\mathsf{adv}}$ and $\mathsf{Ok}^{\mathsf{adv}}_{\mathsf{id}}$ that have now become internal yields the following:

- $\mathsf{Flip} \coloneqq \mathsf{samp}\ \mathsf{flip}$

- $\begin{cases} \mathsf{Commit}(i) \coloneqq \mathsf{samp}\ \mathsf{flip} & \text{if } i \leqslant N \text{ honest} \\ \mathsf{Commit}(i) \coloneqq \mathsf{read}\ \mathsf{AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N \text{ corrupt} \end{cases}$

- $\mathsf{LastCommit} \coloneqq x_{N+1} \leftarrow \mathsf{SumCommit}(N+1);\ f \leftarrow \mathsf{Flip};\ \mathsf{ret}\ x_{N+1} \oplus f$

- $\begin{cases} \mathsf{SumCommit}(0) \coloneqq \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumCommit}(j+1) \coloneqq x_j \leftarrow \mathsf{SumCommit}(j);\ c_j \leftarrow \mathsf{Commit}(j);\ \mathsf{ret}\ x_j \oplus c_j \text{ for } j \leqslant N \end{cases}$

- $\mathsf{SumCommit}(N+2) \coloneqq x_{N+1} \leftarrow \mathsf{SumCommit}(N+1);\ c_{N+1} \leftarrow \mathsf{LastCommit};\ \mathsf{ret}\ x_{N+1} \oplus c_{N+1}$

- $\mathsf{Committed}(i) \coloneqq c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark$ for $i \leqslant N$

- $\mathsf{Committed}(N+1) \coloneqq \mathsf{ret}\ \checkmark$

- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Committed}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{Open}(i) \coloneqq x_{N+2} \leftarrow \mathsf{SumCommit}(N+2);\ \mathsf{ret}\ \checkmark & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Open}(i) \coloneqq \mathsf{read}\ \mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\begin{cases} \mathsf{AllOpen}(0) \coloneqq \mathsf{ret}\ \checkmark \\ \mathsf{AllOpen}(j+1) \coloneqq \_ \leftarrow \mathsf{AllOpen}(j);\ \_ \leftarrow \mathsf{Open}(j);\ \mathsf{ret}\ \checkmark \text{ for } j \leqslant N+1 \end{cases}$

- $\mathsf{Opened}(i) \coloneqq \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i)$ for $i \leqslant N+1$

- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Opened}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{Out}(i) \coloneqq \_ \leftarrow \mathsf{AllOpen}(N+2);\ x_{N+1} \leftarrow \mathsf{SumCommit}(N+1);\ \mathsf{read}\ \mathsf{Flip} & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) \coloneqq \mathsf{read}\ \mathsf{Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

Substituting the channel $\mathsf{LastCommit}$ into the channel $\mathsf{SumCommit}(N+2)$ yields:

- $\mathsf{SumCommit}(N+2) \coloneqq x_{N+1} \leftarrow \mathsf{SumCommit}(N+1);\ f \leftarrow \mathsf{Flip};\ \mathsf{ret}\ x_{N+1} \oplus (x_{N+1} \oplus f)$

By assumption, we can cancel out the Boolean sum:

- $\mathsf{SumCommit}(N+2) \coloneqq x_{N+1} \leftarrow \mathsf{SumCommit}(N+1);\ \mathsf{read}\ \mathsf{Flip}$

In the presence of this simplified definition, we can rewrite the channels $\mathsf{Out}(-)$ to the following:

- $\begin{cases} \mathsf{Out}(i) := \_ \leftarrow \mathsf{AllOpen}(N+2);\ \mathsf{read}\ \mathsf{SumCommit}(N+2) & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) := \mathsf{read}\ \mathsf{Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

The original formulation of $\mathsf{SumCommit}(N+2)$ will be more convenient for our purposes, so we rewrite it back to end up with the following protocol:

- $\mathsf{Flip} := \mathsf{samp}\ \mathsf{flip}$
- $\begin{cases} \mathsf{Commit}(i) := \mathsf{samp}\ \mathsf{flip} & \text{if } i \leqslant N \text{ honest} \\ \mathsf{Commit}(i) := \mathsf{read}\ \mathsf{AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N \text{ corrupt} \end{cases}$
- $\mathsf{LastCommit} := x_{N+1} \leftarrow \mathsf{SumCommit}(N+1);\ f \leftarrow \mathsf{Flip};\ x_{N+1} \oplus f$
- $\begin{cases} \mathsf{SumCommit}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumCommit}(j+1) := x_j \leftarrow \mathsf{SumCommit}(j);\ c_j \leftarrow \mathsf{Commit}(j);\ x_j \oplus c_j \text{ for } j \leqslant N \end{cases}$
- $\mathsf{SumCommit}(N+2) := x_{N+1} \leftarrow \mathsf{SumCommit}(N+1);\ c_{N+1} \leftarrow \mathsf{LastCommit};\ x_{N+1} \oplus c_{N+1}$
- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark \text{ for } i \leqslant N$
- $\mathsf{Committed}(N+1) := \mathsf{ret}\ \checkmark$
- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Committed}(i) \text{ for } i \leqslant N+1$
- $\begin{cases} \mathsf{Open}(i) := x_{N+2} \leftarrow \mathsf{SumCommit}(N+2);\ \mathsf{ret}\ \checkmark & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Open}(i) := \mathsf{read}\ \mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$
- $\begin{cases} \mathsf{AllOpen}(0) := \mathsf{ret}\ \checkmark \\ \mathsf{AllOpen}(j+1) := \_ \leftarrow \mathsf{AllOpen}(j);\ \_ \leftarrow \mathsf{Open}(j);\ \mathsf{ret}\ \checkmark \text{ for } j \leqslant N+1 \end{cases}$
- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i) \text{ for } i \leqslant N+1$
- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Opened}(i) \text{ for } i \leqslant N+1$
- $\begin{cases} \mathsf{Out}(i) := \_ \leftarrow \mathsf{AllOpen}(N+2);\ \mathsf{read}\ \mathsf{SumCommit}(N+2) & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) := \mathsf{read}\ \mathsf{Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

The channel Flip now only occurs in the channel $\mathsf{LastCommit}$, so we can fold it in:

- $\begin{cases} \mathsf{Commit}(i) := \mathsf{samp}\ \mathsf{flip} & \text{if } i \leqslant N \text{ honest} \\ \mathsf{Commit}(i) := \mathsf{read}\ \mathsf{AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N \text{ corrupt} \end{cases}$
- $\mathsf{LastCommit} := x_{N+1} \leftarrow \mathsf{SumCommit}(N+1);\ f \leftarrow \mathsf{samp}\ \mathsf{flip};\ x_{N+1} \oplus f$
- $\begin{cases} \mathsf{SumCommit}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumCommit}(j+1) := x_j \leftarrow \mathsf{SumCommit}(j);\ c_j \leftarrow \mathsf{Commit}(j);\ x_j \oplus c_j \text{ for } j \leqslant N \end{cases}$
- $\mathsf{SumCommit}(N+2) := x_{N+1} \leftarrow \mathsf{SumCommit}(N+1);\ c_{N+1} \leftarrow \mathsf{LastCommit};\ x_{N+1} \oplus c_{N+1}$
- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark \text{ for } i \leqslant N$
- $\mathsf{Committed}(N+1) := \mathsf{ret}\ \checkmark$
- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Committed}(i) \text{ for } i \leqslant N+1$
- $\begin{cases} \mathsf{Open}(i) := x_{N+2} \leftarrow \mathsf{SumCommit}(N+2);\ \mathsf{ret}\ \checkmark & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Open}(i) := \mathsf{read}\ \mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\begin{cases} \mathsf{AllOpen}(0) := \mathsf{ret}\ \checkmark \\ \mathsf{AllOpen}(j+1) := \_ \leftarrow \mathsf{AllOpen}(j);\ \_ \leftarrow \mathsf{Open}(j);\ \mathsf{ret}\ \checkmark\ \text{for } j \leqslant N+1 \end{cases}$
- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i)$ for $i \leqslant N+1$
- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Opened}(i)$ for $i \leqslant N+1$
- $\begin{cases} \mathsf{Out}(i) := \_ \leftarrow \mathsf{AllOpen}(N+2);\ \mathsf{read}\ \mathsf{SumCommit}(N+2) & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) := \mathsf{read}\ \mathsf{Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

By assumption, the distribution flip is invariant under taking a Boolean sum with a fixed bit:

- $\mathsf{LastCommit} := x_{N+1} \leftarrow \mathsf{SumCommit}(N+1);\ \mathsf{samp}\ \mathsf{flip}$

We can unfold the sampling back into a new internal channel $\mathsf{Commit}(N+1)$:

- $\begin{cases} \mathsf{Commit}(i) := \mathsf{samp}\ \mathsf{flip} & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Commit}(i) := \mathsf{read}\ \mathsf{AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$
- $\mathsf{LastCommit} := x_{N+1} \leftarrow \mathsf{SumCommit}(n);\ \mathsf{read}\ \mathsf{Commit}(N+1)$
- $\begin{cases} \mathsf{SumCommit}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumCommit}(j+1) := x_j \leftarrow \mathsf{SumCommit}(j);\ c_j \leftarrow \mathsf{Commit}(j);\ x_j \oplus c_j\ \text{for } j \leqslant N \end{cases}$
- $\mathsf{SumCommit}(N+2) := x_{N+1} \leftarrow \mathsf{SumCommit}(N+1);\ c_{N+1} \leftarrow \mathsf{LastCommit};\ x_{N+1} \oplus c_{N+1}$
- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark$ for $i \leqslant N$
- $\mathsf{Committed}(N+1) := \mathsf{ret}\ \checkmark$
- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Committed}(i)$ for $i \leqslant N+1$
- $\begin{cases} \mathsf{Open}(i) := x_{N+2} \leftarrow \mathsf{SumCommit}(N+2);\ \mathsf{ret}\ \checkmark & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Open}(i) := \mathsf{read}\ \mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$
- $\begin{cases} \mathsf{AllOpen}(0) := \mathsf{ret}\ \checkmark \\ \mathsf{AllOpen}(j+1) := \_ \leftarrow \mathsf{AllOpen}(j);\ \_ \leftarrow \mathsf{Open}(j);\ \mathsf{ret}\ \checkmark\ \text{for } j \leqslant N+1 \end{cases}$
- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i)$ for $i \leqslant N+1$
- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Opened}(i)$ for $i \leqslant N+1$
- $\begin{cases} \mathsf{Out}(i) := \_ \leftarrow \mathsf{AllOpen}(N+2);\ \mathsf{read}\ \mathsf{SumCommit}(N+2) & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) := \mathsf{read}\ \mathsf{Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

The internal channel LastCommit can now be substituted away:

- $\begin{cases} \mathsf{Commit}(i) := \mathsf{samp}\ \mathsf{flip} & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Commit}(i) := \mathsf{read}\ \mathsf{AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$
- $\begin{cases} \mathsf{SumCommit}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumCommit}(j+1) := x_j \leftarrow \mathsf{SumCommit}(j);\ c_j \leftarrow \mathsf{Commit}(j);\ x_j \oplus c_j\ \text{for } j \leqslant N+1 \end{cases}$
- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark$ for $i \leqslant N$
- $\mathsf{Committed}(N+1) := \mathsf{ret}\ \checkmark$
- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Committed}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{Open}(i) := x_{N+2} \leftarrow \mathsf{SumCommit}(N+2);\ \mathsf{ret}\ \checkmark & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Open}(i) := \mathsf{read}\ \mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\begin{cases} \mathsf{AllOpen}(0) := \mathsf{ret}\ \checkmark \\ \mathsf{AllOpen}(j+1) := \_ \leftarrow \mathsf{AllOpen}(j);\ \_ \leftarrow \mathsf{Open}(j);\ \mathsf{ret}\ \checkmark \text{for } j \leqslant N+1 \end{cases}$

- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i)$ for $i \leqslant N+1$

- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Opened}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{Out}(i) := \_ \leftarrow \mathsf{AllOpen}(N+2);\ \mathsf{read}\ \mathsf{SumCommit}(N+2) & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) := \mathsf{read}\ \mathsf{Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

Finally, we rewrite the channel $\mathsf{Committed}(N+1)$ to include a gratuitous dependency on $\mathsf{Commit}(N+1)$:

- $\begin{cases} \mathsf{Commit}(i) := \mathsf{samp}\ \mathsf{flip} & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Commit}(i) := \mathsf{read}\ \mathsf{AdvCommit}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\begin{cases} \mathsf{SumCommit}(0) := \mathsf{ret}\ \mathsf{false} \\ \mathsf{SumCommit}(j+1) := x_j \leftarrow \mathsf{SumCommit}(j);\ c_j \leftarrow \mathsf{Commit}(j);\ x_j \oplus c_j \text{ for } j \leqslant N+1 \end{cases}$

- $\mathsf{Committed}(i) := c_i \leftarrow \mathsf{Commit}(i);\ \mathsf{ret}\ \checkmark$ for $i \leqslant N+1$

- $\mathsf{LeakCommitted}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Committed}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{Open}(i) := x_{N+2} \leftarrow \mathsf{SumCommit}(N+2);\ \mathsf{ret}\ \checkmark & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Open}(i) := \mathsf{read}\ \mathsf{AdvOpen}^{\mathsf{adv}}_{\mathsf{party}(i)} & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

- $\begin{cases} \mathsf{AllOpen}(0) := \mathsf{ret}\ \checkmark \\ \mathsf{AllOpen}(j+1) := \_ \leftarrow \mathsf{AllOpen}(j);\ \_ \leftarrow \mathsf{Open}(j);\ \mathsf{ret}\ \checkmark \text{ for } j \leqslant N+1 \end{cases}$

- $\mathsf{Opened}(i) := \_ \leftarrow \mathsf{Open}(i);\ \mathsf{read}\ \mathsf{Commit}(i)$ for $i \leqslant N+1$

- $\mathsf{LeakOpened}(i)^{\mathsf{comm}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Opened}(i)$ for $i \leqslant N+1$

- $\begin{cases} \mathsf{Out}(i) := \_ \leftarrow \mathsf{AllOpen}(N+2);\ \mathsf{read}\ \mathsf{SumCommit}(N+2) & \text{if } i \leqslant N+1 \text{ honest} \\ \mathsf{Out}(i) := \mathsf{read}\ \mathsf{Out}(i) & \text{if } i \leqslant N+1 \text{ corrupt} \end{cases}$

But this is precisely the cleaned-up version of the real protocol.

9 Two-Party GMW Protocol

In the two-party GMW protocol, Alice and Bob jointly compute the value of a given Boolean circuit built out of *xor*-, *and*-, and *not* gates. The inputs to the circuit are divided between Alice and Bob, and neither party has access to the inputs of the other. For each gate, Alice and Bob maintain their respective *shares* of the actual value v computed by the gate, with Alice's share computed only from the information available to Alice, and analogously for Bob. The respective shares for Alice and Bob sum up to v. We prove the protocol secure against a semi-honest attacker in the case when Alice is corrupt and Bob is honest.

Formally, we assume a coin-flip distribution $\mathsf{flip} : 1 \twoheadrightarrow \mathsf{Bool}$; a Boolean sum function $\oplus : \mathsf{Bool} \times \mathsf{Bool} \rightarrow \mathsf{Bool}$, where we write $x \oplus y$ in place of $\oplus\,(x, y)$; a Boolean multiplication function $* : \mathsf{Bool} \times \mathsf{Bool} \rightarrow \mathsf{Bool}$, where we write $x * y$ in place of $*\,(x, y)$; and a Boolean negation function $\neg : \mathsf{Bool} \rightarrow \mathsf{Bool}$, where we write $\neg x$ in place of $\neg\ x$.

We represent Boolean circuits using the syntax below, where we assume Alice has $N \geqslant 0$ inputs labeled $\{0, \ldots, N-1\}$, and Bob has $M \geqslant 0$ inputs labeled $\{0, \ldots, M-1\}$. Starting from the empty circuit ϵ, we add one gate at a time: an *input* gate allows us to plug into a specified input i of Alice or Bob; a *not* gate negates the value carried on wire k; an *xor* gate computes the Boolean sum of the two values carried on wires k and l; and an *and* gate does the same for Boolean product.

Parties	$p := \mathsf{A}, \mathsf{B}$		
Inputs	$i \in \mathbb{N}$		
Wires	$k, l \in \mathbb{N}$		
Circuits	C	$::=$	$\epsilon \mid C;\ \textit{input-gate}(p, i) \mid C;\ \textit{not-gate}(k) \mid C;\ \textit{xor-gate}(k, l) \mid C;\ \textit{and-gate}(k, l)$

A circuit C with $n \in \mathbb{N}$ wires is considered well-formed if each logical gate combines previously defined wires only:

$$\frac{}{\epsilon\ \mathsf{circuit}(0)} \qquad \frac{C\ \mathsf{circuit}(n) \qquad i < N}{C;\ \textit{input-gate}(\mathsf{A}, i)\ \mathsf{circuit}(n+1)} \qquad \frac{C\ \mathsf{circuit}(n) \qquad i < M}{C;\ \textit{input-gate}(\mathsf{B}, i)\ \mathsf{circuit}(n+1)}$$

$$\frac{C\ \mathsf{circuit}(n) \qquad 0 \leqslant k < n}{C;\ \textit{not-gate}(k)\ \mathsf{circuit}(n+1)} \qquad \frac{C\ \mathsf{circuit}(n) \qquad 0 \leqslant k < n \qquad 0 \leqslant l < n}{C;\ \textit{xor-gate}(k, l)\ \mathsf{circuit}(n+1)}$$

$$\frac{C\ \mathsf{circuit}(n) \qquad 0 \leqslant k < n \qquad 0 \leqslant l < n}{C;\ \textit{and-gate}(k, l)\ \mathsf{circuit}(n+1)}$$

We now fix an ambient Boolean circuit C with K wires $\{0, \ldots, K-1\}$, a subset of which is designated as outputs.

9.1 The Assumptions

At the expression level, we assume that the Boolean sum and product operations are commutative and associative:

- $x : \mathsf{Bool}, y : \mathsf{Bool} \vdash x \oplus y = y \oplus x : \mathsf{Bool}$,
- $x : \mathsf{Bool}, y : \mathsf{Bool} \vdash x * y = y * x : \mathsf{Bool}$,
- $x : \mathsf{Bool}, y : \mathsf{Bool}, z : \mathsf{Bool} \vdash (x \oplus y) \oplus z = x \oplus (y \oplus z) : \mathsf{Bool}$, and
- $x : \mathsf{Bool}, y : \mathsf{Bool}, z : \mathsf{Bool} \vdash (x * y) * z = x * (y * z) : \mathsf{Bool}$.

Furthermore, Boolean multiplication distributes over Boolean sum:

- $x : \mathsf{Bool}, y : \mathsf{Bool}, z : \mathsf{Bool} \vdash (x \oplus y) * z = (x * z) \oplus (y * z) : \mathsf{Bool}$.

Summing up a Boolean with itself yields false and summing up a Boolean with false yields the original Boolean:

- $x : \mathsf{Bool} \vdash x \oplus x = \mathsf{false} : \mathsf{Bool}$, and
- $x : \mathsf{Bool} \vdash x \oplus \mathsf{false} = x : \mathsf{Bool}$.

Negating a Boolean equals summing it up with true:

- $x : \mathsf{Bool} \vdash x \oplus \mathsf{true} = \neg x : \mathsf{Bool}$.

Finally, multiplying a Boolean with false or true yields false or the original Boolean, respectively:

- $x : \mathsf{Bool} \vdash x * \mathsf{false} = \mathsf{false} : \mathsf{Bool}$, and
- $x : \mathsf{Bool} \vdash x * \mathsf{true} = x : \mathsf{Bool}$.

At the distribution level, we assume that the distribution flip on Booleans is invariant under the operation of Boolean sum with a fixed Boolean (as is indeed the case when flip is uniform):

- $x : \mathsf{Bool} \vdash \big(y \leftarrow \mathsf{flip};\ \mathsf{ret}\ x \oplus y\big) = \mathsf{flip} : \mathsf{Bool}$

9.2 The Ideal Protocol

The leakage from the ideal functionality includes the timing information for Bob's inputs, plus the value of each of Alice's inputs, as she is semi-honest:

- $\mathsf{In}(\mathsf{A}, i)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{In}(\mathsf{A}, i)$ for $i < N$
- $\mathsf{InRcvd}(\mathsf{B}, i)^{\mathsf{id}}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{ret}\ \checkmark$ for $i < M$

In the inductive phase, the functionality computes the value carried by each wire $k < K$ of the ambient circuit by induction on the circuit:

- $\mathsf{Wires}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Wires}\big(C;\ \textit{input-gate}(\mathsf{A}, i), K+1\big)$ is the composition of $\mathsf{Wires}(C, K)$ with the protocol
  - $\mathsf{Wire}(K) := \mathsf{read}\ \mathsf{In}(\mathsf{A}, i)$
- $\mathsf{Wires}\big(C;\ \textit{input-gate}(\mathsf{B}, i), K+1\big)$ is the composition of $\mathsf{Wires}(C, K)$ with the protocol
  - $\mathsf{Wire}(K) := \mathsf{read}\ \mathsf{In}(\mathsf{B}, i)$
- $\mathsf{Wires}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Wires}(C, K)$ with the protocol
  - $\mathsf{Wire}(K) := x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ \neg x$
- $\mathsf{Wires}\big(C;\ \textit{xor-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Wires}(C, K)$ with the protocol
  - $\mathsf{Wire}(K) := x \leftarrow \mathsf{Wire}(k);\ y \leftarrow \mathsf{Wire}(l);\ \mathsf{ret}\ x \oplus y$
- $\mathsf{Wires}\big(C;\ \textit{and-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Wires}(C, K)$ with the protocol
  - $\mathsf{Wire}(K) := x \leftarrow \mathsf{Wire}(k);\ y \leftarrow \mathsf{Wire}(l);\ \mathsf{ret}\ x * y$

After performing the above computation, the ideal functionality outputs the computed value for each wire marked as an output, and leaks the outputs to the adversary on behalf of Alice:

- $\begin{cases} \mathsf{Out}(\mathsf{A}, k) := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{A}, k) := \mathsf{read}\ \mathsf{Out}(\mathsf{A}, k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{Out}(\mathsf{B}, k) := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{B}, k) := \mathsf{read}\ \mathsf{Out}(\mathsf{B}, k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\mathsf{Out}(\mathsf{A}, k)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(\mathsf{A}, k)$ for $k < K$

Finally, the channels $\mathsf{Wire}(-)$ coming from the inductive protocol $\mathsf{Wires}(C, K)$ are designated as internal.

9.3 The Real Protocol

The real protocol consists of the two parties, plus an instance of the ideal 1-Out-Of-4 Oblivious Transfer (OT) functionality for each gate, with Alice the sender and Bob the receiver. The code for each party is separated into three parts: in the initial phase, each party computes and distributes everyone's shares for each of its inputs. In the inductive phase, each party computes their share of each wire by induction on the ambient circuit. At last, in the final phase, Alice and Bob send their shares of each output wire to each other and add them up to compute the result.

9.3.1 Alice: Initial Phase

As Alice is semi-honest, she leaks the value of each of her inputs:

- $\mathsf{In}(\mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(\mathsf{A}, i)$ for $i < N$

She next randomly generates shares for each of her inputs:

- $\mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i) := x \leftarrow \mathsf{In}(\mathsf{A}, i);\ \mathsf{samp\ flip}$ for $i < N$
- $\mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i)$ for $i < N$

Bob's share is computed by summing up the input i with Alice's share:

- $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i) := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i);\ x \leftarrow \mathsf{In}(\mathsf{A}, i);\ \mathsf{ret}\ x_A \oplus x$ for $i < N$
- $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i)$ for $i < N$

Alice's share of Bob's inputs is sent over to her by Bob, and she shares the value with the adversary:

- $\mathsf{RcvdInShare}(\mathsf{A}, \mathsf{B}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(\mathsf{A}, \mathsf{B}, i)$ for $i < M$

Alice now records her input shares – those computed as above, as well as those received from Bob:

- $\mathsf{InShare}(\mathsf{A}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{A}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{SendInShare}(\mathsf{A}, \mathsf{B}, i)$ for $i < M$
- $\mathsf{InShare}(\mathsf{A}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{A}, \mathsf{B}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{B}, i)$ for $i < M$

At last, Alice sends over the shares she computed for Bob:

- $\mathsf{SendInShare}(\mathsf{B}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{SendInShare}(\mathsf{B}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(\mathsf{B}, \mathsf{A}, i)$ for $i < N$

The channels

- $\mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i)$ for $i < N$,
- $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i)$ for $i < M$

are declared as internal.

9.3.2 Bob: Initial Phase

Since Bob is honest, the only leakage from him is the timing of his inputs:

- $\mathsf{InRcvd}(\mathsf{B}, i)^{\mathsf{B}}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{ret}\ \checkmark$ for $i < M$

Bob next randomly generates Alice's shares for each of his inputs:

- $\mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{B}, i) := x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{samp\ flip}$ for $i < M$

Bob's share is computed by summing up the input i with Alice's share:

- $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{B}, i) := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{B}, i);\ x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{ret}\ x_A \oplus x$ for $i < M$

Bob now records his input shares – those computed as above, as well as those received from Alice:

- $\mathsf{InShare}(\mathsf{B}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{SendInShare}(\mathsf{B}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{B}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{B}, i)$ for $i < M$

At last, Bob sends over the shares he computed for Alice:

- $\mathsf{SendInShare}(\mathsf{A}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{B}, i)$ for $i < M$

The channels

- $\mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{B}, i)$ for $i < N$,
- $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{B}, i)$ for $i < M$

are declared as internal.

9.3.3 Alice: Inductive Phase

In the case of an *input* gate, Alice uses her corresponding input share from the initial phase. In the case of a *not* gate, she simply copies her share x_A of the incoming wire. If the gate is an *xor* gate, the resulting share is the sum $x_A \oplus y_A$ of the shares of the incoming two wires. The case of an *and* gate is the most complex. The sum of Alice's and Bob's shares must equal $(x_A \oplus x_B) * (y_A \oplus y_B)$, where x_A, y_A and x_B, y_B are the respective shares of Alice and Bob on the incoming two wires. We have

$$(x_A \oplus x_B) * (y_A \oplus y_B) = (x_A * y_A) \oplus (x_A * y_B) \oplus (x_B * y_A) \oplus (x_B * y_B)$$

and the quantity $(x_A * y_B) \oplus (x_B * y_A)$ cannot be directly computed by either Alice or Bob, as neither of them has access to the shares of the other. Instead, Alice and Bob engage in an idealized 1-Out-Of-4 OT exchange: there are four possible combinations of values that x_B, y_B can take, and Alice computes the value of $(x_A * y_B) \oplus (x_B * y_A)$ for each. This offers Bob four messages to choose from, and he selects the one corresponding to the actual values of x_B, y_B. A small caveat: in the exchange as described above, Bob would still be able to infer the value of Alice's shares in certain cases: *e.g.*, if $x_B = 0$ and $y_B = 1$, Bob gets the share x_A as the result of the exchange. To prevent this, Alice encodes her messages by xor-ing them with a random Boolean b that only she knows. To offset for the presence of this Boolean, she includes it in her share $b \oplus (x_A * y_A)$.

To stay consistent throughout the cases, we set up our protocol so that each gate induces the same set of outputs, even though some of these channels may not be relevant to the specific gate in question. These irrelevant channels will simply diverge, which makes them effectively nonexistent. For wire K, the relevant non-adversarial outputs for Alice are among the following:

- $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$ for storing the masking Boolean is relevant for an *and* gate,
- $\mathsf{Share}(\mathsf{A}, K)$ for storing the share is always relevant,
- $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)$, $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)$, $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)$, $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)$ are relevant for an *and* gate.

We define Alice's inductive part as follows:

- $\mathsf{Circ}_\mathsf{A}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Circ}_\mathsf{A}\big(C;\, \textit{input-gate}(\mathsf{A}, i), K + 1\big)$ is the composition of $\mathsf{Circ}_\mathsf{A}(C, K)$ with the following protocol. Alice's share is the input share as determined in the initial part of the protocol:
  - $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{A}, i)$
  - $\mathsf{Share}(\mathsf{A}, K)^\mathsf{A}_\mathsf{adv} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$

  As we said earlier, the 1-Out-Of-4 OT exchange with Bob is vacuous:
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)$

  Since no OT exchange is taking place, no masking Boolean is needed:
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^\mathsf{A}_\mathsf{adv} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^\mathsf{A}_\mathsf{adv}$
- $\mathsf{Circ}_\mathsf{A}\big(C;\, \textit{input-gate}(\mathsf{B}, i), K + 1\big)$ is the composition of $\mathsf{Circ}_\mathsf{A}(C, K)$ with the following protocol. Alice's share is the input share as determined in the initial part of the protocol:
  - $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{B}, i)$
  - $\mathsf{Share}(\mathsf{A}, K)^\mathsf{A}_\mathsf{adv} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$

The 1-Out-Of-4 OT exchange with Bob is again vacuous:

- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$

Since no OT exchange is taking place, no masking Boolean is needed:

- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$

• $\mathsf{Circ}_{\mathsf{A}}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Circ}_{\mathsf{A}}(C,K)$ with the following protocol. Alice's share is the share on wire k:

- $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k)$
- $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$

The 1-Out-Of-4 OT exchange with Bob is once again vacuous:

- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$

Since no OT exchange is taking place, no masking Boolean is needed:

- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$

• $\mathsf{Circ}_{\mathsf{A}}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}_{\mathsf{A}}(C,K)$ with the following protocol. Alice's share is the sum of shares on wires k and l:

- $\mathsf{Share}(\mathsf{A},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ x_A \oplus y_A$
- $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$

The 1-Out-Of-4 OT exchange with Bob is once more vacuous:

- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$

Since no OT exchange is taking place, no masking Boolean is needed:

- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$

• $\mathsf{Circ}_{\mathsf{A}}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}_{\mathsf{A}}(C,K)$ with the following protocol. First, Alice generates a random Boolean for the OT exchange with Bob:

- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{samp}\ \mathsf{flip}$
- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$

She carries out the 1-Out-Of-4 OT exchange with Bob:

- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A$
- $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A$
- $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus y_A$
- $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A \oplus y_A$

Alice's share is computed by adding the masking Boolean to the product of shares on wires k and l:

- $\mathsf{Share}(\mathsf{A},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ (x_A * y_A) \oplus b_A$
- $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$

Finally, the channels

- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},k)$ for $k < K$

are declared as internal.

9.3.4 Bob: Inductive Phase

In the case of an *input* gate, Bob uses his corresponding input share from the initial phase. In the case of a *not* gate, the resulting share is a negation of the share x_B of the incoming wire. If the gate is an *xor* gate, the resulting share is the sum $x_B \oplus y_B$ of the shares of the incoming two wires. Finally, in the case of an *and* gate, Bob engages in an idealized 1-Out-Of-4 exchange with Alice as described in the previous section. To compute his share, he adds the result of the OT exchange to the product $x_B * y_B$ of the shares of the incoming two wires.

For wire K, the relevant non-adversarial outputs for Bob are among the following:

- $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$ for receiving the result of the OT exchange is relevant for an *and* gate,
- $\mathsf{Share}(\mathsf{B},K)$ for storing the share is always relevant,
- $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$, $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K)$ are relevant for an *and* gate.

We define Bob's inductive part as follows:

- $\mathsf{Circ}_{\mathsf{B}}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Circ}_{\mathsf{B}}\big(C;\ \textit{input-gate}(\mathsf{A},i),K+1\big)$ is the composition of $\mathsf{Circ}_{\mathsf{B}}(C,K)$ with the following protocol. Bob's share is the input share as determined in the initial part of the protocol:
  - $\mathsf{Share}(\mathsf{B},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{B},\mathsf{A},i)$

  As we said earlier, the 1-Out-Of-4 OT exchange with Alice is vacuous:
  - $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K)$

  Since no OT exchange is taking place, there is nothing to receive from the OT functionality:
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
- $\mathsf{Circ}_{\mathsf{B}}\big(C;\ \textit{input-gate}(\mathsf{B},i),K+1\big)$ is the composition of $\mathsf{Circ}_{\mathsf{B}}(C,K)$ with the following protocol. Bob's share is the input share as determined in the initial part of the protocol:
  - $\mathsf{Share}(\mathsf{B},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{B},\mathsf{B},i)$

  The 1-Out-Of-4 OT exchange with Alice is again vacuous:
  - $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K)$

Since no OT exchange is taking place, there is nothing to receive from the OT functionality:

- $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$

- $\mathsf{Circ_B}\big(C;\ \textit{not-gate}(k), K+1\big)$ the composition of $\mathsf{Circ_B}(C, K)$ with the following protocol. Bob's share is the negation of the share on wire k:

  - $\mathsf{Share}(\mathsf{B}, K) := x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ \mathsf{ret}\ \neg x_B$

  The 1-Out-Of-4 OT exchange with Alice is once again vacuous:

  - $\mathsf{OTChc}_0(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{OTChc}_1(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A}, \mathsf{B}, K)$

  Since no OT exchange is taking place, there is nothing to receive from the OT functionality:

  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$

- $\mathsf{Circ_B}\big(C;\ \textit{xor-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Circ_B}(C, K)$ with the following protocol. Bob's share is the sum of shares on wires k and l:

  - $\mathsf{Share}(\mathsf{B}, K) := x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ x_B \oplus y_B$

  The 1-Out-Of-4 OT exchange with Bob is once more vacuous:

  - $\mathsf{OTChc}_0(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{OTChc}_1(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A}, \mathsf{B}, K)$

  Since no OT exchange is taking place, there is nothing to receive from the OT functionality:

  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$

- $\mathsf{Circ_B}\big(C;\ \textit{and-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Circ_B}(C, K)$ with the following protocol. First, Bob carries out the 1-Out-Of-4 OT exchange with Alice:

  - $\mathsf{OTChc}_0(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{Share}(\mathsf{B}, k)$
  - $\mathsf{OTChc}_1(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{Share}(\mathsf{B}, l)$

  He records the Boolean he received from the 1-Out-Of-4 OT exchange:

  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{OTOut}(\mathsf{A}, \mathsf{B}, K)$

  Bob's share is computed by adding the above Boolean to the product of shares on wires k and l:

  - $\mathsf{Share}(\mathsf{B}, K) := b_B \leftarrow \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ b_B \oplus (x_B * y_B)$

Finally, the channels

- $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, k)$ for $k < K$

are declared as internal.

9.3.5 Alice: Final Phase

For each output wire, Alice sends her output share to Bob:

- $\begin{cases} \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k) := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)$ for $k < K$

Each output share received from Bob is forwarded to the adversary:

- $\mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{A},\mathsf{B},k)$ for $k < K$

Alice now records the output shares – her own, as well as those received from Bob:

- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{A},k) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k) := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\mathsf{OutShare}(\mathsf{A},\mathsf{B},k) := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{A},\mathsf{B},k)$ for $k < K$

- $\mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)$ for $k < K$

- $\mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)$ for $k < K$

Finally, Alice declares the output to be the sum of the output shares:

- $\mathsf{Out}(\mathsf{A},k) := x_A \leftarrow \mathsf{OutShare}(\mathsf{A},\mathsf{A},k);\ x_B \leftarrow \mathsf{OutShare}(\mathsf{A},\mathsf{B},k);\ \mathsf{ret}\ x_A \oplus x_B$ for $k < K$

- $\mathsf{Out}(\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k)$ for $k < K$

The channels

- $\mathsf{OutShare}(\mathsf{A},\mathsf{A},k)$ for $k < K$,

- $\mathsf{OutShare}(\mathsf{A},\mathsf{B},k)$ for $k < k$

are declared as internal.

9.3.6 Bob: Final Phase

For each output wire, Bob sends his output share to Alice:

- $\begin{cases} \mathsf{SendOutShare}(\mathsf{A},\mathsf{B},k) := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(\mathsf{A},\mathsf{B},k) := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{A},\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

Bob now records the output shares – his own, as well as those received from Alice:

- $\mathsf{OutShare}(\mathsf{B},\mathsf{A},k) := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)$ for $k < K$

- $\begin{cases} \mathsf{OutShare}(\mathsf{B},\mathsf{B},k) := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{B},\mathsf{B},k) := \mathsf{read}\ \mathsf{OutShare}(\mathsf{B},\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

Finally, Bob declares the output to be the sum of the output shares:

- $\mathsf{Out}(\mathsf{B}, k) := x_A \leftarrow \mathsf{OutShare}(\mathsf{B}, \mathsf{A}, k);\ x_B \leftarrow \mathsf{OutShare}(\mathsf{B}, \mathsf{B}, k);\ \mathsf{ret}\ x_A \oplus x_B$ for $k < K$

The channels

- $\mathsf{OutShare}(\mathsf{B}, \mathsf{A}, k)$ for $k < K$,
- $\mathsf{OutShare}(\mathsf{B}, \mathsf{B}, k)$ for $k < K$

are declared as internal.

9.3.7 1-Out-Of-4 Oblivious Transfer Functionality

For each wire $k < K$ we have a separate idealized 1-Out-Of-4 Oblivious Transfer functionality $\mathsf{1OutOf4OT}(k)$, which we now describe. Since Alice is semi-honest, the functionality leaks the value of all messages received from Alice:

- $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, k)$
- $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, k)$
- $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, k)$
- $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, k)$

Since Bob is honest, the functionality only lets the adversary know that a message from Bob has been received:

- $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, k)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A}, \mathsf{B}, k);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, k)^{\mathsf{ot}}_{\mathsf{adv}} := c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A}, \mathsf{B}, k);\ \mathsf{ret}\ \checkmark$

The functionality then selects the appropriate message:

- $\mathsf{OTOut}(\mathsf{A}, \mathsf{B}, k) := m_0 \leftarrow \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, k);\ m_1 \leftarrow \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, k);\ m_2 \leftarrow \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, k);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, k);\ c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A}, \mathsf{B}, k);\ c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A}, \mathsf{B}, k);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

9.3.8 The Real Protocol

The complete code for Alice arises as the composition of her initial, inductive, and final phases, followed by the hiding of the communication internal to Alice - namely, the channels

- $\mathsf{InShare}(\mathsf{A}, \mathsf{A}, i)$ for $i < N$,
- $\mathsf{InShare}(\mathsf{A}, \mathsf{B}, i)$ for $i < M$,
- $\mathsf{Share}(\mathsf{A}, k)$ for $k < K$.

Analogously, the complete code for Bob arises as the composition of his initial, inductive, and final phases, followed by the hiding of the communication internal to Bob - namely, the channels

- $\mathsf{InShare}(\mathsf{B}, \mathsf{A}, i)$ for $i < N$,
- $\mathsf{InShare}(\mathsf{B}, \mathsf{B}, i)$ for $i < M$,
- $\mathsf{Share}(\mathsf{B}, k)$ for $k < K$.

The real protocol is a composition of the two parties, plus an instance of the circuit-wide OT functionality

- $\mathsf{1OutOf4OT}(\mathsf{A}, \mathsf{B}, k)$ for $k < K$,

all followed by the hiding of the internal communication among the two parties and the functionality – namely the channels

- $\mathsf{SendInShare}(\mathsf{A}, \mathsf{B}, i)$ for $i < M$,
- $\mathsf{SendInShare}(\mathsf{B}, \mathsf{A}, i)$ for $i < N$,
- $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, k)$ for $k < K$,
- $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, k)$ for $k < K$,
- $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, k)$ for $k < K$,
- $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, k)$ for $k < K$,
- $\mathsf{OTChc}_0(\mathsf{A}, \mathsf{B}, k)$ for $k < K$,
- $\mathsf{OTChc}_1(\mathsf{A}, \mathsf{B}, k)$ for $k < K$,
- $\mathsf{OTOut}(\mathsf{A}, \mathsf{B}, k)$ for $k < K$,
- $\mathsf{SendOutShare}(\mathsf{A}, \mathsf{B}, k)$ for $k < K$,
- $\mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)$ $k < K$.

9.4 Real = Ideal + Simulator

Our goal is to keep simplifying the real protocol until it becomes clear how to extract out a suitable simulator. We first restructure the entire protocol as a composition of an initial part, an inductive part, and a final part, all followed by the hiding of the channels

- $\mathsf{InShare}(\mathsf{A}, \mathsf{A}, i)$ for $i < N$,
- $\mathsf{InShare}(\mathsf{A}, \mathsf{B}, i)$ for $i < M$,
- $\mathsf{InShare}(\mathsf{B}, \mathsf{A}, i)$ for $i < N$,
- $\mathsf{InShare}(\mathsf{B}, \mathsf{B}, i)$ for $i < M$,
- $\mathsf{Share}(\mathsf{A}, k)$ for $k < K$,
- $\mathsf{Share}(\mathsf{B}, k)$ for $k < K$.

The initial part of the real protocol arises by composing together the respective initial parts for each party, and declaring their communication as internal. Specifically, we have the following protocol Init:

- $\mathsf{In}(\mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(\mathsf{A}, i)$ for $i < N$
- $\mathsf{InRcvd}(\mathsf{B}, i)^{\mathsf{B}}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{ret}\ \checkmark$ for $i < M$
- $\mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{A}, i) := x \leftarrow \mathsf{In}(\mathsf{A}, i);\ \mathsf{samp}\ \mathsf{flip}$ for $i < N$
- $\mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{B}, i) := x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{samp}\ \mathsf{flip}$ for $i < M$
- $\mathsf{InShare}\text{-}\$(\mathsf{B}, \mathsf{A}, i) := x_A \leftarrow \mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{A}, i);\ x \leftarrow \mathsf{In}(\mathsf{A}, i);\ \mathsf{ret}\ x_A \oplus x$ for $i < N$
- $\mathsf{InShare}\text{-}\$(\mathsf{B}, \mathsf{B}, i) := x_A \leftarrow \mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{B}, i);\ x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{ret}\ x_A \oplus x$ for $i < M$
- $\mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}\text{-}\$(\mathsf{B}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(\mathsf{B}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{SendInShare}(\mathsf{B}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$(\mathsf{B}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{SendInShare}(\mathsf{A}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{B}, i)$ for $i < M$
- $\mathsf{SendInShare}(\mathsf{B}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(\mathsf{B}, \mathsf{A}, i)$ for $i < N$

- $\mathsf{RcvdInShare}(\mathsf{A},\mathsf{B},i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(\mathsf{A},\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}(\mathsf{A},\mathsf{A},i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{A},\mathsf{B},i) := \mathsf{read}\ \mathsf{SendInShare}(\mathsf{A},\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}(\mathsf{B},\mathsf{A},i) := \mathsf{read}\ \mathsf{SendInShare}(\mathsf{B},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{B},\mathsf{B},i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{B},\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}(\mathsf{A},\mathsf{A},i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{A},\mathsf{B},i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{B},i)$ for $i < M$

This is followed by the hiding of the channels

- $\mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i)$ for $i < N$,
- $\mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{B},i)$ for $i < M$,
- $\mathsf{InShare\text{-}\$}(\mathsf{B},\mathsf{A},i)$ for $i < N$,
- $\mathsf{InShare\text{-}\$}(\mathsf{B},\mathsf{B},i)$ for $i < M$,
- $\mathsf{SendInShare}(\mathsf{B},\mathsf{A},i)$ for $i < N$,
- $\mathsf{SendInShare}(\mathsf{A},\mathsf{B},i)$ for $i < M$.

The inductive part of the real protocol arises by composing together the respective inductive parts for each party plus the circuit-wide OT functionality, and declaring the communication with the OT functionality as internal. Specifically, we have the following protocol $\mathsf{Circ}(C,K)$:

- $\mathsf{Circ}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Circ}\big(C;\ \textit{input-gate}(\mathsf{A},i),K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{A},i)$
  - $\mathsf{Share}(\mathsf{B},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{B},\mathsf{A},i)$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ \mathsf{ret}\ \checkmark$

- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) := m_0 \leftarrow \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K);\ m_1 \leftarrow \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K);\ m_2 \leftarrow \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K);\ c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

- $\mathsf{Circ}\big(C;\ \textit{input-gate}(\mathsf{B},i), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{B},i)$
  - $\mathsf{Share}(\mathsf{B},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{B},\mathsf{B},i)$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) := m_0 \leftarrow \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K);\ m_1 \leftarrow \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K);\ m_2 \leftarrow \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K);$
    $m_3 \leftarrow \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K);\ c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K);$
    $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

- $\mathsf{Circ}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k)$
  - $\mathsf{Share}(\mathsf{B},K) := x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ \neg x_B$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$

- $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) := m_0 \leftarrow \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K);\ m_1 \leftarrow \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K);\ m_2 \leftarrow \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K);\ c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

- $\mathsf{Circ}\big(C;\ \mathit{xor\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ x_A \oplus y_A$
  - $\mathsf{Share}(\mathsf{B},K) := x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ x_B \oplus y_B$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) := m_0 \leftarrow \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K);\ m_1 \leftarrow \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K);\ m_2 \leftarrow \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K);$
    $m_3 \leftarrow \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K);\ c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K);$
    $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

- $\mathsf{Circ}\big(C;\ \mathit{and\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{samp}\ \mathsf{flip}$
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{OTOut}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{Share}(\mathsf{A},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ (x_A * y_A) \oplus b_A$
  - $\mathsf{Share}(\mathsf{B},K) := b_B \leftarrow \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ b_B \oplus (x_B * y_B)$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus y_A$

- $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A \oplus y_A$
- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k)$
- $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{Share}(\mathsf{B},l)$
- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) := m_0 \leftarrow \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K);\ m_1 \leftarrow \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K);\ m_2 \leftarrow \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K);\ c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

This is followed by the hiding of the channels

- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},k)$ for $k < K$,
- $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},k)$ for $k < K$,
- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},k)$ for $k < K$,
- $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},k)$ for $k < K$,
- $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},k)$ for $k < K$,
- $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},k)$ for $k < K$,
- $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},k)$ for $k < K$
- $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},k)$ for $k < K$,
- $\mathsf{OTOut}(\mathsf{A},\mathsf{B},k)$ for $k < K$.

The final part of the real protocol arises by composing together the respective initial parts for each party, and declaring their communication as internal. Specifically, we have the following protocol Fin:

- $\begin{cases} \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k) := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{SendOutShare}(\mathsf{A},\mathsf{B},k) := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(\mathsf{A},\mathsf{B},k) := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{A},\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)$ for $k < K$
- $\mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{A},\mathsf{B},k)$ for $k < K$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{A},k) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k) := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\mathsf{OutShare}(\mathsf{A}, \mathsf{B}, k) := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{A}, \mathsf{B}, k)$ for $k < K$
- $\mathsf{OutShare}(\mathsf{B}, \mathsf{A}, k) := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)$ for $k < K$
- $\begin{cases} \mathsf{OutShare}(\mathsf{B}, \mathsf{B}, k) := \mathsf{read}\ \mathsf{Share}(\mathsf{B}, k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{B}, \mathsf{B}, k) := \mathsf{read}\ \mathsf{OutShare}(\mathsf{B}, \mathsf{B}, k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\mathsf{OutShare}(\mathsf{A}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A}, \mathsf{A}, k)$ for $k < K$
- $\mathsf{OutShare}(\mathsf{A}, \mathsf{B}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A}, \mathsf{B}, k)$ for $k < K$
- $\mathsf{Out}(\mathsf{A}, k) := x_A \leftarrow \mathsf{OutShare}(\mathsf{A}, \mathsf{A}, k);\ x_B \leftarrow \mathsf{OutShare}(\mathsf{A}, \mathsf{B}, k);\ \mathsf{ret}\ x_A \oplus x_B$ for $k < K$
- $\mathsf{Out}(\mathsf{B}, k) := x_A \leftarrow \mathsf{OutShare}(\mathsf{B}, \mathsf{A}, k);\ x_B \leftarrow \mathsf{OutShare}(\mathsf{B}, \mathsf{B}, k);\ \mathsf{ret}\ x_A \oplus x_B$ for $k < K$
- $\mathsf{Out}(\mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(\mathsf{A}, k)$ for $k < K$

This is followed by the hiding of the channels

- $\mathsf{OutShare}(\mathsf{A}, \mathsf{A}, k)$ for $k < K$,
- $\mathsf{OutShare}(\mathsf{A}, \mathsf{B}, k)$ for $k < K$,
- $\mathsf{OutShare}(\mathsf{B}, \mathsf{A}, k)$ for $k < K$,
- $\mathsf{OutShare}(\mathsf{B}, \mathsf{B}, k)$ for $k < K$,
- $\mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)$ for $k < K$,
- $\mathsf{SendOutShare}(\mathsf{A}, \mathsf{B}, k)$ for $k < K$.

9.4.1 Simplifying The Real Protocol: Initial Phase

The internal channels $\mathsf{SendInShare}(\mathsf{B}, \mathsf{A}, -)$, $\mathsf{SendInShare}(\mathsf{A}, \mathsf{B}, -)$ can be substituted away, which yields the following simplified version Init of the initial part of the real protocol:

- $\mathsf{In}(\mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(\mathsf{A}, i)$ for $i < N$
- $\mathsf{InRcvd}(\mathsf{B}, i)^{\mathsf{B}}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{ret}\ \checkmark$ for $i < M$
- $\mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i) := x \leftarrow \mathsf{In}(\mathsf{A}, i);\ \mathsf{samp}\ \mathsf{flip}$ for $i < N$
- $\mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{B}, i) := x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{samp}\ \mathsf{flip}$ for $i < M$
- $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i) := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i);\ x \leftarrow \mathsf{In}(\mathsf{A}, i);\ \mathsf{ret}\ x_A \oplus x$ for $i < N$
- $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{B}, i) := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{B}, i);\ x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{ret}\ x_A \oplus x$ for $i < M$
- $\mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{SendInShare}(\mathsf{B}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{RcvdInShare}(\mathsf{A}, \mathsf{B}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{B}, i)$ for $i < M$
- $\mathsf{InShare}(\mathsf{A}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{A}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{B}, i)$ for $i < M$
- $\mathsf{InShare}(\mathsf{B}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i)$ for $i < N$

- $\mathsf{InShare}(\mathsf{B},\mathsf{B},i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$(\mathsf{B},\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}(\mathsf{A},\mathsf{A},i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{A},\mathsf{B},i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{B},i)$ for $i < M$

This is followed by the hiding of the channels

- $\mathsf{InShare}\text{-}\$(\mathsf{A},\mathsf{A},i)$ for $i < N$,
- $\mathsf{InShare}\text{-}\$(\mathsf{A},\mathsf{B},i)$ for $i < M$,
- $\mathsf{InShare}\text{-}\$(\mathsf{B},\mathsf{A},i)$ for $i < N$,
- $\mathsf{InShare}\text{-}\$(\mathsf{B},\mathsf{B},i)$ for $i < M$.

9.4.2 Simplifying The Real Protocol: Inductive Phase

Our next order of business is to eliminate all channels interacting with the OT functionality. In the case of *input*-, *not*-, and *xor* gates, the OT channels are divergent and only appear in the corresponding leakage channels. The leakage channels themselves are therefore divergent. For instance, the channel

- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$

reads from the divergent channel

- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$

and so we may equivalently write the following:

- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

Similarly, the channel

- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ \mathsf{ret}\ \checkmark$

reads from the divergent channel

- $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$

and so we may equivalently write the following:

- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

Finally, the channel

- $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) := m_0 \leftarrow \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K);\ m_1 \leftarrow \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K);\ m_2 \leftarrow \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K);\ c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

reads from the divergent channel

- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$

and so we may equivalently write the following:

- $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTOut}(\mathsf{A},\mathsf{B},K)$

In the case of an *and* gate, we start by eliminating any mention of the OT channels from the leakage channels. For instance, substituting the channel

- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A$

into the channel

- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$

yields the following:

- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A$

Analogously, we have the following:

- $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A$
- $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus y_A$
- $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A \oplus y_A$

We treat the receiver channels similarly. For instance, substituting the channel

- $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k)$

into the channel

- $\mathsf{OTChcRcvd}_0(\mathsf{B},\mathsf{A},K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ \mathsf{ret}\ \checkmark$

yields the following:

- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ \checkmark$

Analogously, we have the following:

- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ \checkmark$

At this point, none of the leakage channels refer to any of the OT channels anymore. So outside of the OT channels themselves, the only place where we still refer to an OT channel is in the channel

- $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{OTOut}(\mathsf{A},\mathsf{B},K)$

that stores the result of the OT exchange between Alice and Bob. Substituting the channel

- $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) := m_0 \leftarrow \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K);\ m_1 \leftarrow \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K);\ m_2 \leftarrow \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K);\ c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

into the above thus yields the somewhat more verbose

- $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := m_0 \leftarrow \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K);\ m_1 \leftarrow \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K);\ m_2 \leftarrow \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K);\ c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

with the advantage that it no longer mentions $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K)$. We may further substitute the channels

- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A$
- $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A$
- $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus y_A$
- $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A \oplus y_A$

as well as the channels

- $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k)$
- $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{Share}(\mathsf{B},l)$

to obtain the following:

- $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);$ $y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{if}\ x_B\ \mathsf{then}\ \big(\mathsf{if}\ y_B\ \mathsf{then\ ret}\ b_A{\oplus}x_A{\oplus}y_A\ \mathsf{else\ ret}\ b_A{\oplus}y_A\big)\ \mathsf{else}\ \big(\mathsf{if}\ y_B\ \mathsf{then\ ret}\ b_A{\oplus}x_A\ \mathsf{else\ ret}\ b_A\big)$

Here we no longer refer to any of the OT channels. We can express the above more concisely as follows:

- $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);$ $x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$

Summarizing, we can rewrite the inductive part $\mathsf{Circ}(C,K)$ of the real protocol as follows:

- $\mathsf{Circ}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Circ}\big(C;\ \mathit{input\text{-}gate}(\mathsf{A},i),K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{A},i)$
  - $\mathsf{Share}(\mathsf{B},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{B},\mathsf{A},i)$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTOut}(\mathsf{A},\mathsf{B},K)$
- $\mathsf{Circ}\big(C;\ \mathit{input\text{-}gate}(\mathsf{B},i),K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{B},i)$
  - $\mathsf{Share}(\mathsf{B},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{B},\mathsf{B},i)$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$

- $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTOut}(\mathsf{A},\mathsf{B},K)$

- $\mathsf{Circ}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k)$
  - $\mathsf{Share}(\mathsf{B},K) := x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \ \mathsf{ret}\ \neg x_B$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTOut}(\mathsf{A},\mathsf{B},K)$

- $\mathsf{Circ}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ \ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \ \mathsf{ret}\ x_A \oplus y_A$
  - $\mathsf{Share}(\mathsf{B},K) := x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \ \mathsf{ret}\ x_B \oplus y_B$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$

- $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)$
- $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTChc}_0(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A}, \mathsf{B}, K)$
- $\mathsf{OTChc}_1(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A}, \mathsf{B}, K)$
- $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTOut}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{OTOut}(\mathsf{A}, \mathsf{B}, K)$

- $\mathsf{Circ}\big(C;\ \mathit{and\text{-}gate}(k, l), K+1\big)$ is the composition of $\mathsf{Circ}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{samp\ flip}$
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
  - $\mathsf{Share}(\mathsf{A}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ (x_A * y_A) \oplus b_A$
  - $\mathsf{Share}(\mathsf{B}, K) := b_B \leftarrow \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ b_B \oplus (x_B * y_B)$
  - $\mathsf{Share}(\mathsf{A}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A$
  - $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A \oplus x_A$
  - $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A \oplus y_A$
  - $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A \oplus x_A \oplus y_A$
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A$
  - $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A \oplus x_A$
  - $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A \oplus y_A$
  - $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A \oplus x_A \oplus y_A$
  - $\mathsf{OTChc}_0(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{Share}(\mathsf{B}, k)$
  - $\mathsf{OTChc}_1(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{Share}(\mathsf{B}, l)$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{OTOut}(\mathsf{A}, \mathsf{B}, K) := m_0 \leftarrow \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K);\ m_1 \leftarrow \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K);\ m_2 \leftarrow \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K);$
    $m_3 \leftarrow \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K);\ c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A}, \mathsf{B}, K);\ c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A}, \mathsf{B}, K);$
    $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then\ ret}\ m_3\ \mathsf{else\ ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then\ ret}\ m_1\ \mathsf{else\ ret}\ m_0\big)$

We now split the protocol $\mathsf{Circ}(C, K)$ into three parts. The first protocol $\mathsf{Shares}(C, K)$ performs the computation of shares and is defined as follows:

- $\mathsf{Shares}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \mathit{input\text{-}gate}(\mathsf{A}, i), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$

- $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{A}, i)$
- $\mathsf{Share}(\mathsf{B}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{B}, \mathsf{A}, i)$

- $\mathsf{Shares}\big(C;\ \textit{input-gate}(\mathsf{B}, i), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
  - $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{B}, i)$
  - $\mathsf{Share}(\mathsf{B}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{B}, \mathsf{B}, i)$
- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
  - $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, k)$
  - $\mathsf{Share}(\mathsf{B}, K) := x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ \mathsf{ret}\ \neg x_B$
- $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
  - $\mathsf{Share}(\mathsf{A}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ x_A \oplus y_A$
  - $\mathsf{Share}(\mathsf{B}, K) := x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ x_B \oplus y_B$
- $\mathsf{Shares}\big(C;\ \textit{and-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{samp}\ \mathsf{flip}$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
  - $\mathsf{Share}(\mathsf{A}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ (x_A * y_A) \oplus b_A$
  - $\mathsf{Share}(\mathsf{B}, K) := b_B \leftarrow \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ b_B \oplus (x_B * y_B)$

The second protocol $\mathsf{Adv}(C, K)$ performs all leakages and is defined as follows:

- $\mathsf{Adv}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Adv}\big(C;\ \textit{input-gate}(\mathsf{A}, i), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{input-gate}(\mathsf{B}, i), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$

- $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

- $\mathsf{Adv}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

- $\mathsf{Adv}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

- $\mathsf{Adv}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus y_A$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A \oplus y_A$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ \checkmark$

The third protocol $\mathsf{1OutOf4OT}(C,K)$ performs all Oblivious Transfer exchanges and is defined as follows:

- $\mathsf{1OutOf4OT}(\epsilon,0)$ is the protocol $\mathsf{0}$

- $\mathsf{1OutOf4OT}\big(C;\ \textit{input-gate}(\mathsf{A},i), K+1\big)$ is the composition of $\mathsf{1OutOf4OT}(C,K)$ with the protocol
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$

- $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K)$
- $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTOut}(\mathsf{A},\mathsf{B},K)$

- $\mathsf{1OutOf4OT}\big(C;\ \textit{input-gate}(\mathsf{B},i), K+1\big)$ is the composition of $\mathsf{1OutOf4OT}(C,K)$ with the protocol
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTOut}(\mathsf{A},\mathsf{B},K)$
- $\mathsf{1OutOf4OT}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{1OutOf4OT}(C,K)$ with the protocol
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTOut}(\mathsf{A},\mathsf{B},K)$
- $\mathsf{1OutOf4OT}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{1OutOf4OT}(C,K)$ with the protocol
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{OTOut}(\mathsf{A},\mathsf{B},K)$
- $\mathsf{1OutOf4OT}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{1OutOf4OT}(C,K)$ with the protocol
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K) \coloneqq b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K) \coloneqq b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K) \coloneqq b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus y_A$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K) \coloneqq b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A \oplus y_A$
  - $\mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{Share}(\mathsf{B},k)$
  - $\mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{Share}(\mathsf{B},l)$
  - $\mathsf{OTOut}(\mathsf{A},\mathsf{B},K) \coloneqq m_0 \leftarrow \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K);\ m_1 \leftarrow \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K);\ m_2 \leftarrow \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K);$
    $m_3 \leftarrow \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K);\ c_0 \leftarrow \mathsf{OTChc}_0(\mathsf{A},\mathsf{B},K);\ c_1 \leftarrow \mathsf{OTChc}_1(\mathsf{A},\mathsf{B},K);$
    $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

At this point, none of the channels defined by $\mathsf{1OutOf4OT}(C,K)$ are utilized anywhere outside of $\mathsf{1OutOf4OT}(C,K)$ and as such we may discard this protocol entirely. The inductive part of the real protocol therefore consists of the protocols $\mathsf{Shares}(C,K)$ and $\mathsf{Adv}(C,K)$, followed by the hiding of the channels

- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},k)$ for $k < K$,
- $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},k)$ for $k < K$.

9.4.3 Simplifying The Real Protocol: Final Phase

If wire k is not an output, then the channels

- $\mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)$
- $\mathsf{OutShare}(\mathsf{B}, \mathsf{A}, k) := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)$

read from the divergent channel

- $\mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k) := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)$

and thus we may equivalently write the following:

- $\mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}}$ for $k < K$
- $\mathsf{OutShare}(\mathsf{B}, \mathsf{A}, k) := \mathsf{read}\ \mathsf{OutShare}(\mathsf{B}, \mathsf{A}, k)$

If wire k is an output, then the channels

- $\mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)$
- $\mathsf{OutShare}(\mathsf{B}, \mathsf{A}, k) := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)$

read from the channel

- $\mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k) := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, k)$

so we may substitute:

- $\mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, k)$
- $\mathsf{OutShare}(\mathsf{B}, \mathsf{A}, k) := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, k)$

Summarizing the above, we get the following for channels $\mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, -)^{\mathsf{A}}_{\mathsf{adv}}$ and $\mathsf{OutShare}(\mathsf{B}, \mathsf{A}, -)$:

- $\begin{cases} \mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{B}, \mathsf{A}, k) := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{B}, \mathsf{A}, k) := \mathsf{read}\ \mathsf{OutShare}(\mathsf{B}, \mathsf{A}, k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

If wire k is not an output, then the channels

- $\mathsf{RcvdOutShare}(\mathsf{A}, \mathsf{B}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{A}, \mathsf{B}, k)$
- $\mathsf{OutShare}(\mathsf{A}, \mathsf{B}, k) := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{A}, \mathsf{B}, k)$

read from the divergent channel

- $\mathsf{SendOutShare}(\mathsf{A}, \mathsf{B}, k) := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{A}, \mathsf{B}, k)$

and thus we may equivalently write the following:

- $\mathsf{RcvdOutShare}(\mathsf{A}, \mathsf{B}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(\mathsf{A}, \mathsf{B}, k)^{\mathsf{A}}_{\mathsf{adv}}$ for $k < K$
- $\mathsf{OutShare}(\mathsf{A}, \mathsf{B}, k) := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A}, \mathsf{B}, k)$

If wire k is an output, then the channels

- $\mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{A},\mathsf{B},k)$
- $\mathsf{OutShare}(\mathsf{A},\mathsf{B},k) := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{A},\mathsf{B},k)$

read from the channel

- $\mathsf{SendOutShare}(\mathsf{A},\mathsf{B},k) := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k)$

so we may substitute:

- $\mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k)$
- $\mathsf{OutShare}(\mathsf{A},\mathsf{B},k) := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k)$

Summarizing the above, we get the following for channels $\mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}}$ and $\mathsf{OutShare}(\mathsf{A},\mathsf{B},-)$:

- $\begin{cases} \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{B},k) := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k) := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

At this point, the internal channels $\mathsf{SendOutShare}(\mathsf{B},\mathsf{A},-)$, $\mathsf{SendOutShare}(\mathsf{A},\mathsf{B},-)$ are unused and can be eliminated. Our simplified version Fin of the final part of the real protocol is therefore as follows:

- $\begin{cases} \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{A},k) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k) := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{B},k) := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k) := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{B},\mathsf{A},k) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{B},\mathsf{A},k) := \mathsf{read}\ \mathsf{OutShare}(\mathsf{B},\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $$\begin{cases} \mathsf{OutShare}(\mathsf{B},\mathsf{B},k) := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{B},\mathsf{B},k) := \mathsf{read}\ \mathsf{OutShare}(\mathsf{B},\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$$
- $\mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)$ for $k < K$
- $\mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)$ for $k < K$
- $\mathsf{Out}(\mathsf{A},k) := x_A \leftarrow \mathsf{OutShare}(\mathsf{A},\mathsf{A},k);\ x_B \leftarrow \mathsf{OutShare}(\mathsf{A},\mathsf{B},k);\ \mathsf{ret}\ x_A \oplus x_B$ for $k < K$
- $\mathsf{Out}(\mathsf{B},k) := x_A \leftarrow \mathsf{OutShare}(\mathsf{B},\mathsf{A},k);\ x_B \leftarrow \mathsf{OutShare}(\mathsf{B},\mathsf{B},k);\ \mathsf{ret}\ x_A \oplus x_B$ for $k < K$
- $\mathsf{Out}(\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k)$ for $k < K$

This is followed by the hiding of the channels

- $\mathsf{OutShare}(\mathsf{A},\mathsf{A},k)$ for $k < K$,
- $\mathsf{OutShare}(\mathsf{A},\mathsf{B},k)$ for $k < K$,
- $\mathsf{OutShare}(\mathsf{B},\mathsf{A},k)$ for $k < K$,
- $\mathsf{OutShare}(\mathsf{B},\mathsf{B},k)$ for $k < K$.

9.4.4 Timing of Shares I

Since Bob is by assumption honest, the simulator does not have access to his inputs. Therefore, any computation that depends on the value of Bob's inputs must be eliminated. In particular, all of Bob's shares must be eliminated.

By design, summing up the respective shares $x_A \oplus x_B$ of each party on a given wire yields the actual value x carried by the wire. If we got our hands on x, for example by inductively computing the circuit the same way the ideal functionality does, we could replace Bob's share x_B by the sum $x \oplus x_A$. For this strategy to work, however, we need to arrange the timing so that Bob computes his shares after Alice.

To this end, we introduce new internal channels

- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{A},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{A},\mathsf{A},i);\ \mathsf{ret}\ \checkmark$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{B},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{A},\mathsf{B},i);\ \mathsf{ret}\ \checkmark$ for $i < M$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{A},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{B},\mathsf{A},i);\ \mathsf{ret}\ \checkmark$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{B},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{B},\mathsf{B},i);\ \mathsf{ret}\ \checkmark$ for $i < M$

for the timing of input shares in the initial part of the protocol and

- $\mathsf{Share}\text{-}\checkmark(\mathsf{A},k) := \_ \leftarrow \mathsf{Share}(\mathsf{A},k);\ \mathsf{ret}\ \checkmark$ for $k < K$
- $\mathsf{Share}\text{-}\checkmark(\mathsf{B},k) := \_ \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ \checkmark$ for $k < K$

for the timing of shares in the inductive part of the protocol.

Since the primary job of the simulator is to construct the appropriate leakage, we start by eliminating any mention of the Bob's shares from the leakage channels. Upon carefully examining the inductive part of the real protocol, we see that the only place where we leak information depending on Bob's shares is when we leak the timing of his shares on behalf of the OT functionality in the case of an *and* gate. But even in this case, the *value* of the shares is immaterial - it is only the timing information that matters.

Specifically, take the protocol $\mathsf{Adv}(C,K)$ in the case of an *and* gate. We can write the channels

- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ \checkmark$

- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ \checkmark$

equivalently as follows:

- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(\mathsf{B},k)$
- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(\mathsf{B},l)$

So the inductive part of the real protocol now has $\mathsf{Adv}(C,K)$ looking like so:

- $\mathsf{Adv}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Adv}\big(C;\ \textit{input-gate}(\mathsf{A},i),K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{input-gate}(\mathsf{B},i),K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{not-gate}(k),K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{xor-gate}(k,l),K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

- $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

- $\mathsf{Adv}\big(C;\ \mathit{and\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus y_A$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A \oplus y_A$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(\mathsf{B},k)$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(\mathsf{B},l)$

We now amend Bob's shares with a gratuitous dependency on timing, which we will later convert into a dependency on the sum of shares of parties $0,\dots,N$. Specifically, in the presence of the channels $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{A},-)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{B},-)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{A},-)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{B},-)$ as well as the channels $\mathsf{Share}\text{-}\checkmark(\mathsf{A},-)$, $\mathsf{Share}\text{-}\checkmark(\mathsf{B},-)$ we can express the protocol $\mathsf{Shares}(C,K)$ equivalently as follows:

- $\mathsf{Shares}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \mathit{input\text{-}gate}(\mathsf{A},i), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{A},i)$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{A},i);\ \mathsf{read}\ \mathsf{InShare}(\mathsf{B},\mathsf{A},i)$
- $\mathsf{Shares}\big(C;\ \mathit{input\text{-}gate}(\mathsf{B},i), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{B},i)$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{B},i);\ \mathsf{read}\ \mathsf{InShare}(\mathsf{B},\mathsf{B},i)$
- $\mathsf{Shares}\big(C;\ \mathit{not\text{-}gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k)$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B},k);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ \neg x_B$
- $\mathsf{Shares}\big(C;\ \mathit{xor\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ x_A \oplus y_A$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B},k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B},l);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ x_B \oplus y_B$

- $\mathsf{Shares}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{samp\ flip}$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
  - $\mathsf{Share}(\mathsf{A},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ (x_A * y_A) \oplus b_A$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B},k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B},l);$
    $b_B \leftarrow \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ b_B \oplus (x_B * y_B)$

Clearly, we can make the timing of shares independent of their value; this was the point of introducing the timing channels in the first place. For input shares, the timing only depends on the timing of the corresponding input, so we introduce new internal channels

- $\mathsf{In}\text{-}\checkmark(\mathsf{A},i) := \_ \leftarrow \mathsf{read}\ \mathsf{In}(\mathsf{A},i);\ \mathsf{ret}\ \checkmark$ for $i < N$
- $\mathsf{In}\text{-}\checkmark(\mathsf{B},i) := \_ \leftarrow \mathsf{read}\ \mathsf{In}(\mathsf{B},i);\ \mathsf{ret}\ \checkmark$ for $i < M$

to keep track of whether an input has arrived. For a wire share, the timing depends on the timing of every input that recursively feeds into the wire. We can easily compute this in a new protocol $\mathsf{Wires}\text{-}\checkmark(C,K)$:

- $\mathsf{Wires}\text{-}\checkmark(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \textit{input-gate}(\mathsf{A},i), K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{Wire}\text{-}\checkmark(K) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{A},i)$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \textit{input-gate}(\mathsf{B},i), K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{Wire}\text{-}\checkmark(K) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{B},i)$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{Wire}\text{-}\checkmark(K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \mathsf{ret}\ \checkmark$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{Wire}\text{-}\checkmark(K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(l);\ \mathsf{ret}\ \checkmark$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{Wire}\text{-}\checkmark(K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(l);\ \mathsf{ret}\ \checkmark$

Our goal now is to show that the timing channels can be equivalently characterized as follows:

- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{A},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{B},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{A},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{B},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{B},i)$ for $i < M$
- $\mathsf{Share}\text{-}\checkmark(\mathsf{A},k) := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$ for $k < K$
- $\mathsf{Share}\text{-}\checkmark(\mathsf{B},k) := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$ for $k < K$

To this end, we introduce new internal channels

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A},\mathsf{B},i)$ for $i < M$

- $\mathsf{InShare\text{-}\$\text{-}\checkmark}(\mathsf{B},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare\text{-}\$\text{-}\checkmark}(\mathsf{B},\mathsf{B},i)$ for $i < M$

to keep track of the timing information in the protocol Init and

- $\mathsf{SendBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},k)$ for $k < K$
- $\mathsf{RcvdBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},k)$ for $k < K$

to keep track of the timing information in the protocol $\mathsf{Shares}(C, K)$. Call the following protocol fragment $\mathsf{Init\text{-}\checkmark}$:

- $\mathsf{InShare\text{-}\$\text{-}\checkmark}(\mathsf{A},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare\text{-}\$\text{-}\checkmark}(\mathsf{A},\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare\text{-}\$\text{-}\checkmark}(\mathsf{B},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare\text{-}\$\text{-}\checkmark}(\mathsf{B},\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare\text{-}\checkmark}(\mathsf{A},\mathsf{A},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{A},\mathsf{A},i);\ \mathsf{ret}\ \checkmark$ for $i < N$
- $\mathsf{InShare\text{-}\checkmark}(\mathsf{A},\mathsf{B},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{A},\mathsf{B},i);\ \mathsf{ret}\ \checkmark$ for $i < M$
- $\mathsf{InShare\text{-}\checkmark}(\mathsf{B},\mathsf{A},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{B},\mathsf{A},i);\ \mathsf{ret}\ \checkmark$ for $i < N$
- $\mathsf{InShare\text{-}\checkmark}(\mathsf{B},\mathsf{B},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{B},\mathsf{B},i);\ \mathsf{ret}\ \checkmark$ for $i < M$

Call the following protocol fragment $\mathsf{Shares\text{-}\checkmark}(C, K)$:

- $\mathsf{SendBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},k)$ for $k < K$
- $\mathsf{RcvdBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},k)$ for $k < K$
- $\mathsf{Share\text{-}\checkmark}(\mathsf{A},k) := \_ \leftarrow \mathsf{Share}(\mathsf{A},k);\ \mathsf{ret}\ \checkmark$ for $k < K$
- $\mathsf{Share\text{-}\checkmark}(\mathsf{B},k) := \_ \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ \checkmark$ for $k < K$

In the presence of the channels $\mathsf{InShare\text{-}\checkmark}(\mathsf{A},\mathsf{A},-)$, $\mathsf{InShare\text{-}\checkmark}(\mathsf{A},\mathsf{B},-)$, $\mathsf{InShare\text{-}\checkmark}(\mathsf{B},\mathsf{A},-)$, $\mathsf{InShare\text{-}\checkmark}(\mathsf{B},\mathsf{B},-)$ and the protocol $\mathsf{Shares}(C, K)$ we can define the protocol $\mathsf{Shares\text{-}\checkmark}(C, K)$ equivalently as follows:

- $\mathsf{Shares\text{-}\checkmark}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares\text{-}\checkmark}\big(C;\ \textit{input-gate}(\mathsf{A}, i), K+1\big)$ is the composition of $\mathsf{Shares\text{-}\checkmark}(C, K)$ with the protocol
  - $\mathsf{SendBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{RcvdBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{Share\text{-}\checkmark}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare\text{-}\checkmark}(\mathsf{A},\mathsf{A},i)$
  - $\mathsf{Share\text{-}\checkmark}(\mathsf{B},K) := \mathsf{read}\ \mathsf{InShare\text{-}\checkmark}(\mathsf{B},\mathsf{A},i)$
- $\mathsf{Shares\text{-}\checkmark}\big(C;\ \textit{input-gate}(\mathsf{B}, i), K+1\big)$ is the composition of $\mathsf{Shares\text{-}\checkmark}(C, K)$ with the protocol
  - $\mathsf{SendBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{RcvdBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{Share\text{-}\checkmark}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare\text{-}\checkmark}(\mathsf{A},\mathsf{B},i)$
  - $\mathsf{Share\text{-}\checkmark}(\mathsf{B},K) := \mathsf{read}\ \mathsf{InShare\text{-}\checkmark}(\mathsf{B},\mathsf{B},i)$
- $\mathsf{Shares\text{-}\checkmark}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares\text{-}\checkmark}(C, K)$ with the protocol
  - $\mathsf{SendBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{RcvdBit\text{-}\checkmark}(\mathsf{A},\mathsf{B},K)$

- $\mathsf{Share}\text{-}\checkmark(\mathsf{A}, K) := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(\mathsf{A}, k)$
- $\mathsf{Share}\text{-}\checkmark(\mathsf{B}, K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B}, k);\ \mathsf{ret}\ \checkmark$

- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{Share}\text{-}\checkmark(\mathsf{A}, K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, l);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{Share}\text{-}\checkmark(\mathsf{B}, K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B}, k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B}, l);\ \mathsf{ret}\ \checkmark$

- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, l);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \_ \leftarrow \mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, l);$ $\_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B}, k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B}, l);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{Share}\text{-}\checkmark(\mathsf{A}, K) := \_ \leftarrow \mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, l);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{Share}\text{-}\checkmark(\mathsf{B}, K) := \_ \leftarrow \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B}, k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B}, l);\ \mathsf{ret}\ \checkmark$

We also observe that in the presence of the protocol Init we can define the protocol $\mathsf{Init}\text{-}\checkmark$ equivalently as follows:

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A}, \mathsf{A}, i) := \_ \leftarrow \mathsf{In}\text{-}\checkmark(\mathsf{A}, i);\ \mathsf{ret}\ \checkmark$ for $i < N$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A}, \mathsf{B}, i) := \_ \leftarrow \mathsf{In}\text{-}\checkmark(\mathsf{B}, i);\ \mathsf{ret}\ \checkmark$ for $i < M$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B}, \mathsf{A}, i) := \_ \leftarrow \mathsf{In}\text{-}\checkmark(\mathsf{A}, i);\ \_ \leftarrow \mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A}, \mathsf{A}, i);\ \mathsf{ret}\ \checkmark$ for $i < N$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B}, \mathsf{B}, i) := \_ \leftarrow \mathsf{In}\text{-}\checkmark(\mathsf{B}, i);\ \_ \leftarrow \mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A}, \mathsf{B}, i);\ \mathsf{ret}\ \checkmark$ for $i < M$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A}, \mathsf{B}, i)$ for $i < M$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B}, \mathsf{B}, i)$ for $i < M$

Furthermore, in the presence of the channels $\mathsf{In}\text{-}\checkmark(\mathsf{A}, -)$, $\mathsf{In}\text{-}\checkmark(\mathsf{B}, -)$ we can express the protocol $\mathsf{Init}\text{-}\checkmark$ equivalently as follows:

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{B}, i)$ for $i < M$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{B}, i)$ for $i < M$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{B}, i)$ for $i < M$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{B}, i)$ for $i < M$

Lastly, in the presence of the channels $\mathsf{InShare}\text{-}\checkmark(\mathsf{A}, \mathsf{A}, -)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, -)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{B}, \mathsf{A}, -)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{B}, \mathsf{B}, -)$ as well as the protocol $\mathsf{Wires}\text{-}\checkmark(C, K)$ we can express the protocol $\mathsf{Shares}\text{-}\checkmark(C, K)$ equivalently as the channels

- $\mathsf{Share}\text{-}\checkmark(\mathsf{A}, k) := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$ for $k < K$

- $\mathsf{Share}\text{-}\checkmark(\mathsf{B}, k) := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$ for $k < K$

together with the following protocol $\mathsf{SendRcvdBits}\text{-}\checkmark(C, K)$:

- $\mathsf{SendRcvdBits}\text{-}\checkmark(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{SendRcvdBits}\text{-}\checkmark\big(C;\ \textit{input-gate}(\mathsf{A}, i), K + 1\big)$ is the composition of $\mathsf{SendRcvdBits}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
- $\mathsf{SendRcvdBits}\text{-}\checkmark\big(C;\ \textit{input-gate}(\mathsf{B}, i), K + 1\big)$ is the composition of $\mathsf{SendRcvdBits}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
- $\mathsf{SendRcvdBits}\text{-}\checkmark\big(C;\ \textit{not-gate}(k), K + 1\big)$ is the composition of $\mathsf{SendRcvdBits}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
- $\mathsf{SendRcvdBits}\text{-}\checkmark\big(C;\ \textit{xor-gate}(k, l), K + 1\big)$ is the composition of $\mathsf{SendRcvdBits}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
- $\mathsf{SendRcvdBits}\text{-}\checkmark\big(C;\ \textit{and-gate}(k, l), K + 1\big)$ is the composition of $\mathsf{SendRcvdBits}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(l);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(l);\ \mathsf{ret}\ \checkmark$

We now revisit the case of an *and* gate in the protocol $\mathsf{Adv}(C, K)$. We can write the channels

- $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(\mathsf{B}, k)$
- $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(\mathsf{B}, l)$

equivalently as follows:

- $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$
- $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(l)$

So the inductive part of the real protocol now has $\mathsf{Adv}(C, K)$ looking like so:

- $\mathsf{Adv}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Adv}\big(C;\ \textit{input-gate}(\mathsf{A}, i), K + 1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$

- $\mathsf{Adv}\big(C;\ \textit{input-gate}(\mathsf{B}, i), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{xor-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{and-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{Share}(\mathsf{A}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A$
  - $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A \oplus x_A$
  - $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A \oplus y_A$
  - $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A \oplus x_A \oplus y_A$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(l)$

In the presence of the channels $\mathsf{InShare}\text{-}\checkmark(\mathsf{A}, \mathsf{A}, -)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, -)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{B}, \mathsf{A}, -)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{B}, \mathsf{B}, -)$ and the channels $\mathsf{Share}\text{-}\checkmark(\mathsf{A}, -)$, $\mathsf{Share}\text{-}\checkmark(\mathsf{B}, -)$ as well as the protocols $\mathsf{Wires}\text{-}\checkmark$ and $\mathsf{Shares}\text{-}\checkmark$ we can express the protocol $\mathsf{Shares}(C, K)$ equivalently as follows:

- $\mathsf{Shares}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(\mathsf{A}, i), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{A},i)$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(K);\ \mathsf{read}\ \mathsf{InShare}(\mathsf{B},\mathsf{A},i)$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(\mathsf{B}, i), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{B},i)$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(K);\ \mathsf{read}\ \mathsf{InShare}(\mathsf{B},\mathsf{B},i)$
- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k)$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ \neg x_B$
- $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ x_A \oplus y_A$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ x_B \oplus y_B$
- $\mathsf{Shares}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{samp}\ \mathsf{flip}$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
  - $\mathsf{Share}(\mathsf{A},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ (x_A * y_A) \oplus b_A$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(K);\ b_B \leftarrow \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ b_B \oplus (x_B * y_B)$

9.4.5 Timing of Shares II

We now revisit the real protocol in the form we had at the beginning of Section 9.4.4. We first consider the protocol $\mathsf{Adv}(C, K)$ in the case of an *and* gate. We want to write the channels

- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ \checkmark$

in the form below:

- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ \checkmark$

So the inductive part of the real protocol now has $\mathsf{Adv}(C, K)$ looking like so:

- $\mathsf{Adv}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Adv}\big(C;\ \textit{input-gate}(\mathsf{A}, i), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{input-gate}(\mathsf{B}, i), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{xor-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{and-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol

- $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
- $\mathsf{Share}(\mathsf{A}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$
- $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A$
- $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A \oplus x_A$
- $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A \oplus y_A$
- $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ b_A \oplus x_A \oplus y_A$
- $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ \checkmark$

We now amend Bob's shares with a gratuitous dependency on Alice's shares. Specifically, we modify the protocol $\mathsf{Shares}(C, K)$ as follows:

- $\mathsf{Shares}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \mathit{input\text{-}gate}(\mathsf{A}, i), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
  - $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{A}, i)$
  - $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ \mathsf{read}\ \mathsf{InShare}(\mathsf{B}, \mathsf{A}, i)$
- $\mathsf{Shares}\big(C;\ \mathit{input\text{-}gate}(\mathsf{B}, i), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
  - $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{B}, i)$
  - $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ \mathsf{read}\ \mathsf{InShare}(\mathsf{B}, \mathsf{B}, i)$
- $\mathsf{Shares}\big(C;\ \mathit{not\text{-}gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
  - $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, k)$
  - $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ \mathsf{ret}\ \neg x_B$
- $\mathsf{Shares}\big(C;\ \mathit{xor\text{-}gate}(k, l), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
  - $\mathsf{Share}(\mathsf{A}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ x_A \oplus y_A$
  - $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ x_B \oplus y_B$
- $\mathsf{Shares}\big(C;\ \mathit{and\text{-}gate}(k, l), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{samp}\ \mathsf{flip}$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
  - $\mathsf{Share}(\mathsf{A}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ (x_A * y_A) \oplus b_A$
  - $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ b_B \leftarrow \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ b_B \oplus (x_B * y_B)$

It is not at all clear that the aforementioned amendments of $\mathsf{Adv}(C,K)$ and $\mathsf{Shares}(C,K)$ are sound. In the rest of this section we justify their soundness.

We start by introducing the channels

- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{A},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{A},\mathsf{A},i);\ \mathsf{ret}\ \checkmark$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{B},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{A},\mathsf{B},i);\ \mathsf{ret}\ \checkmark$ for $i < M$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{A},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{B},\mathsf{A},i);\ \mathsf{ret}\ \checkmark$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{B},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{B},\mathsf{B},i);\ \mathsf{ret}\ \checkmark$ for $i < M$

for the timing of input shares in the initial part of the protocol and

- $\mathsf{Share}\text{-}\checkmark(\mathsf{A},k) := \_ \leftarrow \mathsf{Share}(\mathsf{A},k);\ \mathsf{ret}\ \checkmark$ for $k < K$
- $\mathsf{Share}\text{-}\checkmark(\mathsf{B},k) := \_ \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ \checkmark$ for $k < K$

for the timing of shares in the inductive part of the protocol.

Take the protocol $\mathsf{Adv}(C,K)$ in the case of an *and* gate. We can write the channels

- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ \checkmark$
- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ \checkmark$

equivalently as follows:

- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(\mathsf{A},k)$
- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(\mathsf{A},l)$

So the inductive part of the real protocol now has $\mathsf{Adv}(C,K)$ looking like so:

- $\mathsf{Adv}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Adv}\big(C;\ \mathit{input\text{-}gate}(\mathsf{A},i),K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \mathit{input\text{-}gate}(\mathsf{B},i),K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

- $\mathsf{Adv}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus y_A$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A \oplus y_A$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(\mathsf{A},k)$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(\mathsf{A},l)$

We now amend Bob's shares further: in the presence of the channels $\mathsf{Share}\text{-}\checkmark(\mathsf{A},-)$ we can turn the gratuitous dependency on Alice's shares into a dependency on the corresponding timing channel. Specifically, we can express the protocol $\mathsf{Shares}(C,K)$ equivalently as follows:

- $\mathsf{Shares}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(\mathsf{A},i), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{A},i)$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A},K);\ \mathsf{read}\ \mathsf{InShare}(\mathsf{B},\mathsf{A},i)$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(\mathsf{B},i), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol

- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
- $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
- $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{B},i)$
- $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A},K);\ \mathsf{read}\ \mathsf{InShare}(\mathsf{B},\mathsf{B},i)$

- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k)$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A},K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ \neg x_B$
- $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ x_A \oplus y_A$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A},K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ x_B \oplus y_B$
- $\mathsf{Shares}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{samp\ flip}$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);$ $x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
  - $\mathsf{Share}(\mathsf{A},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ (x_A * y_A) \oplus b_A$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A},K);\ b_B \leftarrow \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K);$ $x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ b_B \oplus (x_B * y_B)$

We now carry out largely the same steps as before to make the timing of shares independent of their value. To this end, we introduce new internal channels

- $\mathsf{In}\text{-}\checkmark(\mathsf{A},i) := \_ \leftarrow \mathsf{read}\ \mathsf{In}(\mathsf{A},i);\ \mathsf{ret}\ \checkmark$ for $i < N$
- $\mathsf{In}\text{-}\checkmark(\mathsf{B},i) := \_ \leftarrow \mathsf{read}\ \mathsf{In}(\mathsf{B},i);\ \mathsf{ret}\ \checkmark$ for $i < M$

to keep track of whether an input has arrived, and a new protocol $\mathsf{Wires}\text{-}\checkmark(C,K)$ that keeps track of the timing of wire shares:

- $\mathsf{Wires}\text{-}\checkmark(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \textit{input-gate}(\mathsf{A},i), K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{Wire}\text{-}\checkmark(K) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{A},i)$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \textit{input-gate}(\mathsf{B},i), K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{Wire}\text{-}\checkmark(K) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{B},i)$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{Wire}\text{-}\checkmark(K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \mathsf{ret}\ \checkmark$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{Wire}\text{-}\checkmark(K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(l);\ \mathsf{ret}\ \checkmark$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C,K)$ with the protocol

- $\mathsf{Wire}\text{-}\checkmark(K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(l);\ \mathsf{ret}\ \checkmark$

Our goal is again to show that the timing channels can be equivalently characterized as follows:

- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{A},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{B},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{A},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{B},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{B},i)$ for $i < M$
- $\mathsf{Share}\text{-}\checkmark(\mathsf{A},k) := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$ for $k < K$
- $\mathsf{Share}\text{-}\checkmark(\mathsf{B},k) := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$ for $k < K$

To this end, we introduce new internal channels

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A},\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B},\mathsf{B},i)$ for $i < M$

to keep track of the timing information in the protocol Init and

- $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},k)$ for $k < K$
- $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},k)$ for $k < K$

to keep track of the timing information in the protocol $\mathsf{Circ}(C,K)$. Call the following protocol fragment $\mathsf{Init}\text{-}\checkmark$:

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A},\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B},\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{A},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{A},\mathsf{A},i);\ \mathsf{ret}\ \checkmark$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{B},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{A},\mathsf{B},i);\ \mathsf{ret}\ \checkmark$ for $i < M$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{A},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{B},\mathsf{A},i);\ \mathsf{ret}\ \checkmark$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{B},i) := \_ \leftarrow \mathsf{InShare}(\mathsf{B},\mathsf{B},i);\ \mathsf{ret}\ \checkmark$ for $i < M$

Call the following protocol fragment $\mathsf{Shares}\text{-}\checkmark(C,K)$:

- $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},k)$ for $k < K$
- $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},k)$ for $k < K$
- $\mathsf{Share}\text{-}\checkmark(\mathsf{A},k) := \_ \leftarrow \mathsf{Share}(\mathsf{A},k);\ \mathsf{ret}\ \checkmark$ for $k < K$
- $\mathsf{Share}\text{-}\checkmark(\mathsf{B},k) := \_ \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ \checkmark$ for $k < K$

In the presence of the channels $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{A},-)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{B},-)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{A},-)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{B},-)$ as well as the protocol $\mathsf{Shares}(C,K)$ we can define the protocol $\mathsf{Shares}\text{-}\checkmark(C,K)$ equivalently as follows:

- $\mathsf{Shares}\text{-}\checkmark(\epsilon,0)$ is the protocol $\mathsf{0}$

- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{input-gate}(\mathsf{A}, i), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{Share}\text{-}\checkmark(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}\text{-}\checkmark(\mathsf{A}, \mathsf{A}, i)$
  - $\mathsf{Share}\text{-}\checkmark(\mathsf{B}, K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, K);\ \mathsf{read}\ \mathsf{InShare}\text{-}\checkmark(\mathsf{B}, \mathsf{A}, i)$
- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{input-gate}(\mathsf{B}, i), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{Share}\text{-}\checkmark(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, i)$
  - $\mathsf{Share}\text{-}\checkmark(\mathsf{B}, K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, K);\ \mathsf{read}\ \mathsf{InShare}\text{-}\checkmark(\mathsf{B}, \mathsf{B}, i)$
- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{Share}\text{-}\checkmark(\mathsf{A}, K) := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(\mathsf{A}, k)$
  - $\mathsf{Share}\text{-}\checkmark(\mathsf{B}, K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B}, k);\ \mathsf{ret}\ \checkmark$
- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{xor-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{Share}\text{-}\checkmark(\mathsf{A}, K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, l);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{Share}\text{-}\checkmark(\mathsf{B}, K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B}, k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B}, l);\ \mathsf{ret}\ \checkmark$
- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{and-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, l);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K) := \_ \leftarrow \mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, l);$ $\_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B}, k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B}, l);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{Share}\text{-}\checkmark(\mathsf{A}, K) := \_ \leftarrow \mathsf{SendBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, l);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{Share}\text{-}\checkmark(\mathsf{B}, K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{A}, K);\ \_ \leftarrow \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, K);$ $\_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B}, k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(\mathsf{B}, l);\ \mathsf{ret}\ \checkmark$

We also observe that in the presence of the protocol Init we can define the protocol $\mathsf{Init}\text{-}\checkmark$ equivalently as follows:

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A}, \mathsf{A}, i) := \_ \leftarrow \mathsf{In}\text{-}\checkmark(\mathsf{A}, i);\ \mathsf{ret}\ \checkmark$ for $i < N$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A}, \mathsf{B}, i) := \_ \leftarrow \mathsf{In}\text{-}\checkmark(\mathsf{B}, i);\ \mathsf{ret}\ \checkmark$ for $i < M$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B}, \mathsf{A}, i) := \_ \leftarrow \mathsf{In}\text{-}\checkmark(\mathsf{A}, i);\ \_ \leftarrow \mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A}, \mathsf{A}, i);\ \mathsf{ret}\ \checkmark$ for $i < N$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B}, \mathsf{B}, i) := \_ \leftarrow \mathsf{In}\text{-}\checkmark(\mathsf{B}, i);\ \_ \leftarrow \mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A}, \mathsf{B}, i);\ \mathsf{ret}\ \checkmark$ for $i < M$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A}, \mathsf{B}, i)$ for $i < M$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B}, \mathsf{B}, i)$ for $i < M$

Furthermore, in the presence of the channels $\mathsf{In}\text{-}\checkmark(\mathsf{A},-)$, $\mathsf{In}\text{-}\checkmark(\mathsf{B},-)$ we can express the protocol $\mathsf{Init}\text{-}\checkmark$ equivalently as follows:

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A},\mathsf{A},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{A},\mathsf{B},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B},\mathsf{A},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(\mathsf{B},\mathsf{B},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{A},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{B},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{A},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{B},i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(\mathsf{B},i)$ for $i < M$

In the presence of the channels $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{A},-)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{A},\mathsf{B},-)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{A},-)$, $\mathsf{InShare}\text{-}\checkmark(\mathsf{B},\mathsf{B},-)$ as well as the protocol $\mathsf{Wires}\text{-}\checkmark(C,K)$ we can define the protocol $\mathsf{Shares}\text{-}\checkmark(C,K)$ equivalently as the channels

- $\mathsf{Share}\text{-}\checkmark(\mathsf{A},k) := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$ for $k < K$
- $\mathsf{Share}\text{-}\checkmark(\mathsf{B},k) := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$ for $k < K$

together with the following protocol $\mathsf{SendRcvdBits}\text{-}\checkmark(C,K)$:

- $\mathsf{SendRcvdBits}\text{-}\checkmark(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{SendRcvdBits}\text{-}\checkmark\big(C;\ \mathit{input\text{-}gate}(\mathsf{A},i), K+1\big)$ is the composition of $\mathsf{SendRcvdBits}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K)$
- $\mathsf{SendRcvdBits}\text{-}\checkmark\big(C;\ \mathit{input\text{-}gate}(\mathsf{B},i), K+1\big)$ is the composition of $\mathsf{SendRcvdBits}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K)$
- $\mathsf{SendRcvdBits}\text{-}\checkmark\big(C;\ \mathit{not\text{-}gate}(k), K+1\big)$ is the composition of $\mathsf{SendRcvdBits}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K)$
- $\mathsf{SendRcvdBits}\text{-}\checkmark\big(C;\ \mathit{xor\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{SendRcvdBits}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K)$
- $\mathsf{SendRcvdBits}\text{-}\checkmark\big(C;\ \mathit{and\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{SendRcvdBits}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(l);\ \mathsf{ret}\ \checkmark$
  - $\mathsf{RcvdBit}\text{-}\checkmark(\mathsf{A},\mathsf{B},K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(l);\ \mathsf{ret}\ \checkmark$

We now revisit the case of an *and* gate in the protocol $\mathsf{Adv}(C,K)$. We can write the channels

- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(\mathsf{A},k)$
- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(\mathsf{A},l)$

equivalently as follows:

- $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$
- $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(l)$

So the inductive part of the real protocol now has $\mathsf{Adv}(C, K)$ looking like so:

- $\mathsf{Adv}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Adv}\big(C;\ \textit{input-gate}(\mathsf{A}, i), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{input-gate}(\mathsf{B}, i), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \textit{xor-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

- $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Adv}\big(C;\ \mathit{and\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$
  - $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A$
  - $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A$
  - $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus y_A$
  - $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A \oplus y_A$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire\text{-}}\checkmark(k)$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire\text{-}}\checkmark(l)$

Finally, in the presence of the channels $\mathsf{Share\text{-}}\checkmark(\mathsf{A},-)$ we can characterize the protocol $\mathsf{Shares}(C,K)$ equivalently as follows:

- $\mathsf{Shares}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \mathit{input\text{-}gate}(\mathsf{A},i), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{A},i)$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Wire\text{-}}\checkmark(K);\ \mathsf{read}\ \mathsf{InShare}(\mathsf{B},\mathsf{A},i)$
- $\mathsf{Shares}\big(C;\ \mathit{input\text{-}gate}(\mathsf{B},i), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{B},i)$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Wire\text{-}}\checkmark(K);\ \mathsf{read}\ \mathsf{InShare}(\mathsf{B},\mathsf{B},i)$
- $\mathsf{Shares}\big(C;\ \mathit{not\text{-}gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k)$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Wire\text{-}}\checkmark(K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ \neg x_B$
- $\mathsf{Shares}\big(C;\ \mathit{xor\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
  - $\mathsf{Share}(\mathsf{A},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ x_A \oplus y_A$
  - $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Wire\text{-}}\checkmark(K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ x_B \oplus y_B$
- $\mathsf{Shares}\big(C;\ \mathit{and\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{samp}\ \mathsf{flip}$

– $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);$ $x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
– $\mathsf{Share}(\mathsf{A},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ (x_A * y_A) \oplus b_A$
– $\mathsf{Share}(\mathsf{B},K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(K);\ b_B \leftarrow \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K);$ $x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ b_B \oplus (x_B * y_B)$

The resulting real protocol is identical to the form of the real protocol we had at the end of Section 9.4.4, which justifies the amendments to $\mathsf{Adv}(C,K)$ and $\mathsf{Shares}(C,K)$ we made at the beginning of this section.

9.4.6 Sum Of Shares

We continue to simplify the real protocol after amending $\mathsf{Adv}(C,K)$ and $\mathsf{Shares}(C,K)$ as indicated in Section 9.4.5.

We start by adding new internal channels

- $\mathsf{Share}\text{-}\Sigma(k) := x_A \leftarrow \mathsf{Share}\text{-}\Sigma(\mathsf{A},k);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ x_A \oplus x_B$ for $k < K$

that keep track of the sum of shares on each wire k.

We now revisit the final part of the real protocol. We can express the channels

- $\mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)$ for $k < K$
- $\mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)$ for $k < K$

equivalently as follows:

- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

We can express the channels

- $\mathsf{Out}(\mathsf{A},k) := x_A \leftarrow \mathsf{OutShare}(\mathsf{A},\mathsf{A},k);\ x_B \leftarrow \mathsf{OutShare}(\mathsf{A},\mathsf{B},k);\ \mathsf{ret}\ x_A \oplus x_B$ for $k < K$
- $\mathsf{Out}(\mathsf{B},k) := x_A \leftarrow \mathsf{OutShare}(\mathsf{B},\mathsf{A},k);\ x_B \leftarrow \mathsf{OutShare}(\mathsf{B},\mathsf{B},k);\ \mathsf{ret}\ x_A \oplus x_B$ for $k < K$

equivalently as follows:

- $\begin{cases} \mathsf{Out}(\mathsf{A},k) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ x_A \oplus x_B \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{Out}(\mathsf{B},k) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ x_A \oplus x_B \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Out}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

In presence of the channels $\mathsf{Share}\text{-}\Sigma(-)$ we can express the above equivalently as follows:

- $\begin{cases} \mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Out}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

At this point, the internal channels $\mathsf{OutShare}(\mathsf{A},\mathsf{A},-)$, $\mathsf{OutShare}(\mathsf{A},\mathsf{B},-)$, $\mathsf{OutShare}(\mathsf{B},\mathsf{A},-)$, $\mathsf{OutShare}(\mathsf{B},\mathsf{B},-)$ are unused and can be eliminated. The simplified version Fin of the final part of the real protocol is therefore as follows:

- $\begin{cases} \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Out}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\mathsf{Out}(\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k)$ for $k < K$

We now show that for each wire, the respective shares of the two parties add up to the value carried by the wire. At the same time we express Bob's shares in a closed form, as the sum of Alice's shares plus the value on the wire. We proceed by an induction on circuits: in the presence of the protocol Init we can express the channels

- $\mathsf{Share}\text{-}\Sigma(k) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ \mathsf{ret}\ x_A \oplus x_B$ for $k < K$

together with the protocol $\mathsf{Shares}(C,K)$ equivalently as the channels

- $\mathsf{Share}(\mathsf{B}, k) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ \mathsf{ret}\ x_A \oplus x$ for $k < K$

together with the following new form of the protocol $\mathsf{Shares}(C, K)$:

- $\mathsf{Shares}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(\mathsf{A}, i), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
  - $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{A}, i)$
  - $\mathsf{Share}\text{-}\Sigma(K) := \mathsf{In}(\mathsf{A}, i)$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(\mathsf{B}, i), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
  - $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{B}, i)$
  - $\mathsf{Share}\text{-}\Sigma(K) := \mathsf{In}(\mathsf{B}, i)$
- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
  - $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, k)$
  - $\mathsf{Share}\text{-}\Sigma(K) := x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ \mathsf{ret}\ \neg x$
- $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
  - $\mathsf{Share}(\mathsf{A}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ x_A \oplus y_A$
  - $\mathsf{Share}\text{-}\Sigma(K) := x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ y \leftarrow \mathsf{Share}\text{-}\Sigma(l);\ \mathsf{ret}\ x \oplus y$
- $\mathsf{Shares}\big(C;\ \textit{and-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{samp}\ \mathsf{flip}$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
  - $\mathsf{Share}(\mathsf{A}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ (x_A * y_A) \oplus b_A$
  - $\mathsf{Share}\text{-}\Sigma(K) := x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ y \leftarrow \mathsf{Share}\text{-}\Sigma(l);\ \mathsf{ret}\ x * y$

To see why this works, we consider each gate in turn.

- In the case of an *input* gate for Alice's input, we start by substituting the inductive form of the channel
  - $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ \mathsf{read}\ \mathsf{InShare}(\mathsf{B}, \mathsf{A}, i)$

  into the closed form of the channel
  - $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, K);\ \mathsf{ret}\ x_A \oplus x_B$

  which yields the following:
  - $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{InShare}(\mathsf{B}, \mathsf{A}, i);\ \mathsf{ret}\ x_A \oplus x_B$

By canceling out two applications of $x_A \oplus -$ we can reformulate the channel $\mathsf{Share}(\mathsf{B}, K)$ as follows:

- $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{InShare}(\mathsf{B}, \mathsf{A}, i);\ \mathsf{ret}\ x_A \oplus (x_A \oplus x_B)$

The above can be expressed more concisely as

- $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(K);\ \mathsf{ret}\ x_A \oplus x$

and this is the desired closed form of the channel $\mathsf{Share}(\mathsf{B}, K)$.
We can now turn our attention to the sum of shares. In the presence of the channel

- $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{A}, i)$

and the channel

- $\mathsf{InShare}(\mathsf{A}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{A}, i)$

from the protocol Init we can write the channel

- $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{InShare}(\mathsf{B}, \mathsf{A}, i);\ \mathsf{ret}\ x_A \oplus x_B$

equivalently as follows:

- $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{A}, i);\ x_B \leftarrow \mathsf{InShare}(\mathsf{B}, \mathsf{A}, i);\ \mathsf{ret}\ x_A \oplus x_B$

In the presence of the channels

- $\mathsf{InShare}(\mathsf{B}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$(\mathsf{B}, \mathsf{A}, i)$
- $\mathsf{InShare}\text{-}\$(\mathsf{B}, \mathsf{A}, i) := x_A \leftarrow \mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{A}, i);\ x \leftarrow \mathsf{In}(\mathsf{A}, i);\ \mathsf{ret}\ x_A \oplus x$

from the protocol Init we can further write the channel $\mathsf{Share}\text{-}\Sigma(K)$ as follows:

- $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{A}, i);\ x \leftarrow \mathsf{In}(\mathsf{A}, i);\ \mathsf{ret}\ x_A \oplus (x_A \oplus x)$

We can cancel out the two applications of $x_A \oplus -$:

- $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{A}, i);\ \mathsf{read}\ \mathsf{In}(\mathsf{A}, i)$

This is almost what we want except for the extra dependency on the channel $\mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{A}, i)$. But it is easy to see that this dependency can be dropped because the channel

- $\mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{A}, i) := x \leftarrow \mathsf{In}(\mathsf{A}, i);\ \mathsf{samp}\ \mathsf{flip}$

only reads from the channel $\mathsf{In}(\mathsf{A}, i)$, which $\mathsf{Share}\text{-}\Sigma(K)$ reads from as well:

- $\mathsf{Share}\text{-}\Sigma(K) := \mathsf{read}\ \mathsf{In}(\mathsf{A}, i)$

But this is precisely the desired inductive form of the channel $\mathsf{Share}\text{-}\Sigma(K)$.

- In the case of an *input* gate for Bob's input, we start by substituting the inductive form of the channel

  - $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ \mathsf{read}\ \mathsf{InShare}(\mathsf{B}, \mathsf{B}, i)$

  into the closed form of the channel

  - $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, K);\ \mathsf{ret}\ x_A \oplus x_B$

  which yields the following:

  - $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{InShare}(\mathsf{B}, \mathsf{B}, i);\ \mathsf{ret}\ x_A \oplus x_B$

  By canceling out two applications of $x_A \oplus -$ we can reformulate the channel $\mathsf{Share}(\mathsf{B}, K)$ as follows:

  - $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{InShare}(\mathsf{B}, \mathsf{B}, i);\ \mathsf{ret}\ x_A \oplus (x_A \oplus x_B)$

The above can be expressed more concisely as

– $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(K);\ \mathsf{ret}\ x_A \oplus x$

and this is the desired closed form of the channel $\mathsf{Share}(\mathsf{B}, K)$.

We can now turn our attention to the sum of shares. In the presence of the channel

– $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{B}, i)$

and the channel

– $\mathsf{InShare}(\mathsf{A}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{B}, i)$

from the protocol Init we can write the channel

– $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{InShare}(\mathsf{B}, \mathsf{B}, i);\ \mathsf{ret}\ x_A \oplus x_B$

equivalently as follows:

– $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{B}, i);\ x_B \leftarrow \mathsf{InShare}(\mathsf{B}, \mathsf{B}, i);\ \mathsf{ret}\ x_A \oplus x_B$

In the presence of the channels

– $\mathsf{InShare}(\mathsf{B}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$(\mathsf{B}, \mathsf{B}, i)$
– $\mathsf{InShare}\text{-}\$(\mathsf{B}, \mathsf{B}, i) := x_A \leftarrow \mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{B}, i);\ x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{ret}\ x_A \oplus x$

from the protocol Init we can further write the channel $\mathsf{Share}\text{-}\Sigma(K)$ as follows:

– $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{B}, i);\ x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{ret}\ x_A \oplus (x_A \oplus x)$

We can cancel out the two applications of $x_A \oplus -$:

– $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{B}, i);\ \mathsf{read}\ \mathsf{In}(\mathsf{B}, i)$

This is almost what we want except for the extra dependency on the channel $\mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{B}, i)$. But it is easy to see that this dependency can be dropped because the channel

– $\mathsf{InShare}\text{-}\$(\mathsf{A}, \mathsf{B}, i) := x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{samp}\ \mathsf{flip}$

only reads from the channel $\mathsf{In}(\mathsf{B}, i)$, which $\mathsf{Share}\text{-}\Sigma(K)$ reads from as well:

– $\mathsf{Share}\text{-}\Sigma(K) := \mathsf{read}\ \mathsf{In}(\mathsf{B}, i)$

But this is precisely the desired inductive form of the channel $\mathsf{Share}\text{-}\Sigma(K)$.

- In the case of a *not* gate, we start by substituting the inductive form of the channel

  – $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ \mathsf{ret}\ \neg x_B$

  into the closed form of the channel

  – $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, K);\ \mathsf{ret}\ x_A \oplus x_B$

  which yields the following:

  – $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ \mathsf{ret}\ x_A \oplus (\neg x_B)$

  By canceling out two applications of $x_A \oplus -$ we can reformulate the channel $\mathsf{Share}(\mathsf{B}, K)$ as follows:

  – $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ \mathsf{ret}\ x_\Sigma \oplus (x_\Sigma \oplus (\neg x_B))$

  The above can be expressed more concisely as

  – $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(K);\ \mathsf{ret}\ x_A \oplus x$

and this is the desired closed form of the channel $\mathsf{Share}(\mathsf{B}, K)$.

We can now turn our attention to the sum of shares. Substituting the channel

- $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, k)$

into the channel

- $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ \mathsf{ret}\ x_A \oplus (\neg x_B)$

yields the following:

- $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ \mathsf{ret}\ x_A \oplus (\neg x_B)$

The negation can be brought to the top level:

- $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ \mathsf{ret}\ \neg(x_A \oplus x_B)$

But this is precisely what we get if we substitute the closed form of the channel

- $\mathsf{Share}\text{-}\Sigma(k) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ \mathsf{ret}\ x_A \oplus x_B$

into the desired inductive form of the channel

- $\mathsf{Share}\text{-}\Sigma(K) := x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ \mathsf{ret}\ \neg x$

so we are done.

- In the case of an *xor* gate, we start by substituting the inductive form of the channel

  - $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ x_B \oplus y_B$

  into the closed form of the channel

  - $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, K);\ \mathsf{ret}\ x_A \oplus x_B$

  which yields the following:

  - $\mathsf{Share}\text{-}\Sigma(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ x_A \oplus (x_B \oplus y_B)$

  By canceling out two applications of $x_A \oplus -$ we can reformulate the channel $\mathsf{Share}(B, K)$ as follows:

  - $\mathsf{Share}(B, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ x_A \oplus \big(x_A \oplus (x_A \oplus y_B)\big)$

  The above can be expressed more concisely as

  - $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(K);\ \mathsf{ret}\ x_A \oplus x$

  and this is the desired closed form of the channel $\mathsf{Share}(\mathsf{B}, K)$.

  We can now turn our attention to the sum of shares. Substituting the channel

  - $\mathsf{Share}(\mathsf{A}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ x_A \oplus y_A$

  into the channel

  - $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ x_A \oplus (x_B \oplus y_B)$

  yields the following:

  - $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ (x_A \oplus y_A) \oplus (x_B \oplus y_B)$

  After a slight rearrangement we get the following:

- $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);$
  $y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ (x_A \oplus x_B) \oplus (y_A \oplus y_B)$

But this is precisely what we get if we substitute the channels

- $\mathsf{Share}\text{-}\Sigma(k) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ \mathsf{ret}\ x_A \oplus x_B$
- $\mathsf{Share}\text{-}\Sigma(l) := y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ y_A \oplus y_B$

into the desired inductive form of the channel

- $\mathsf{Share}\text{-}\Sigma(K) := x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ y \leftarrow \mathsf{Share}\text{-}\Sigma(l);\ \mathsf{ret}\ x \oplus y$

so we are done.

- In the case of an *and* gate, we start by substituting the inductive form of the channel

  - $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ b_B \leftarrow \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ b_B \oplus (x_B * y_B)$

  into the closed form of the channel

  - $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, K);\ \mathsf{ret}\ x_A \oplus x_B$

  which yields the following:

  - $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ b_B \leftarrow \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ x_A \oplus (b_B \oplus (x_B * y_B))$

  By canceling out two applications of $x_A \oplus -$ we can reformulate the channel $\mathsf{Share}(\mathsf{B}, K)$ as follows:

  - $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ b_B \leftarrow \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ x_A \oplus \big(x_A \oplus (b_B \oplus (x_B * y_B))\big)$

  The above can be expressed more concisely as

  - $\mathsf{Share}(\mathsf{B}, K) := x_A \leftarrow \mathsf{Share}\text{-}\Sigma(\mathsf{A}, K);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(K);\ \mathsf{ret}\ x_A \oplus x$

  and this is the desired closed form of the channel $\mathsf{Share}(\mathsf{B}, K)$.

  We can now turn our attention to the sum of shares. Substituting the channels

  - $\mathsf{Share}(\mathsf{A}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ (x_A * y_A) \oplus b_A$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$

  into the channel

  - $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, K);\ b_B \leftarrow \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ x_A \oplus (b_B \oplus (x_B * y_B))$

  yields the following:

  - $\mathsf{Share}\text{-}\Sigma(K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);$
    $y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ \big((x_A * y_A) \oplus b_A\big) \oplus \big(\big(b_A \oplus (x_A * y_B) \oplus (x_B * y_A)\big) \oplus (x_B * y_B)\big)$

  After a slight rearrangement we get the following:

  - $\mathsf{Share}\text{-}\Sigma(K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);$
    $y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ (x_A * y_A) \oplus \big(b_A \oplus \big(b_A \oplus (x_A * y_B) \oplus (x_B * y_A)\big)\big) \oplus (x_B * y_B)$

  Canceling out the two applications of $b_A \oplus -$ yields the following:

- $\mathsf{Share}\text{-}\Sigma(K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);$ $x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ (x_A * y_A) \oplus (x_A * y_B) \oplus (x_B * y_A) \oplus (x_B * y_B)$

We can drop the dependency on the unused channel

- $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{samp\ flip}$

because it only reads from the channels $\mathsf{Share}(\mathsf{A}, k)$ and $\mathsf{Share}(\mathsf{A}, l)$, which $\mathsf{Share}\text{-}\Sigma(K)$ reads from as well:

- $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);$ $y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ (x_A * y_A) \oplus (x_A * y_B) \oplus (x_B * y_A) \oplus (x_B * y_B)$

After a slight rearrangement we get the following:

- $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);$ $y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ (x_A * y_A) \oplus (x_A * y_B) \oplus (x_B * y_A) \oplus (x_B * y_B)$

The above is equivalent to the following:

- $\mathsf{Share}\text{-}\Sigma(K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);$ $y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ (x_A \oplus x_B) * (y_A \oplus y_B)$

But this is precisely what we get if we substitute the channels

- $\mathsf{Share}\text{-}\Sigma(k) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x_B \leftarrow \mathsf{Share}(B, k);\ \mathsf{ret}\ x_A \oplus x_B$
- $\mathsf{Share}\text{-}\Sigma(l) := y_A \leftarrow \mathsf{Share}(A, l);\ y_B \leftarrow \mathsf{Share}(B, l);\ \mathsf{ret}\ y_A \oplus y_B$

into the desired inductive form of the channel

- $\mathsf{Share}\text{-}\Sigma(k) := x \leftarrow \mathsf{Share}(k);\ y \leftarrow \mathsf{Share}(l);\ \mathsf{ret}\ x * y$

so we are done.

We have now shown that summing up Alice's and Bob's respective shares $x_A \oplus x_B$ on a given wire yields the actual value x carried by the wire. Currently the computation is performed by the channels $\mathsf{Share}\text{-}\Sigma(-)$ but we can extract it out into a separate protocol $\mathsf{Wires}(C, K)$ as defined in the ideal functionality. Specifically, we introduce new internal channels

- $\mathsf{Wire}(k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(k)$ for $k < K$

and observe that together with the protocol $\mathsf{Shares}(C, K)$ we can express them equivalently as the channels

- $\mathsf{Share}\text{-}\Sigma(k) := \mathsf{read}\ \mathsf{Wire}(k)$ for $k < K$

together with the aforementioned protocol $\mathsf{Wires}(C, K)$ and the following new form of the protocol $\mathsf{Shares}(C, K)$:

- $\mathsf{Shares}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(\mathsf{A}, i), K + 1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
  - $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{A}, i)$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(\mathsf{B}, i), K + 1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
  - $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{B}, i)$
- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K + 1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol

- $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
- $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
- $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, k)$

- $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k, l), K + 1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K)$
  - $\mathsf{Share}(\mathsf{A}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ x_A \oplus y_A$
- $\mathsf{Shares}\big(C;\ \textit{and-gate}(k, l), K + 1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{samp\ flip}$
  - $\mathsf{RcvdBit}(\mathsf{B}, \mathsf{A}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);$
    $x_B \leftarrow \mathsf{Share}(\mathsf{B}, k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B}, l);\ \mathsf{ret}\ b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
  - $\mathsf{Share}(\mathsf{A}, K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A}, l);\ \mathsf{ret}\ (x_A * y_A) \oplus b_A$

9.4.7 Eliminating Bob's Shares

We begin by eliminating the channels $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, -)$, $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{B}, -)$, $\mathsf{InShare}(\mathsf{B}, \mathsf{A}, -)$, $\mathsf{InShare}(\mathsf{B}, \mathsf{B}, -)$ from the initial part of the protocol. Substituting the channels

- $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i) := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i);\ x \leftarrow \mathsf{In}(\mathsf{A}, i);\ \mathsf{ret}\ x_A \oplus x$ for $i < N$

into the channels

- $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{SendInShare}(\mathsf{B}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i)$ for $i < N$

yields the following:

- $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i);\ x \leftarrow \mathsf{In}(\mathsf{A}, i);\ \mathsf{ret}\ x_A \oplus x$ for $i < N$
- $\mathsf{SendInShare}(\mathsf{B}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i);\ x \leftarrow \mathsf{In}(\mathsf{A}, i);\ \mathsf{ret}\ x_A \oplus x$ for $i < N$

Substituting the channels

- $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i) := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i);\ x \leftarrow \mathsf{In}(\mathsf{A}, i);\ \mathsf{ret}\ x_A \oplus x$ for $i < N$
- $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{B}, i) := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{B}, i);\ x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{ret}\ x_A \oplus x$ for $i < M$

into the channels

- $\mathsf{InShare}(\mathsf{B}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{B}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{B}, i)$ for $i < M$

yields the following:

- $\mathsf{InShare}(\mathsf{B}, \mathsf{A}, i) := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i);\ x \leftarrow \mathsf{In}(\mathsf{A}, i);\ \mathsf{ret}\ x_A \oplus x$ for $i < N$
- $\mathsf{InShare}(\mathsf{B}, \mathsf{B}, i) := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{B}, i);\ x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{ret}\ x_A \oplus x$ for $i < M$

At this point, the internal channels $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, -)$, $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{B}, -)$ are unused and can be eliminated. The top-level internal channels $\mathsf{InShare}(\mathsf{B}, \mathsf{A}, -)$, $\mathsf{InShare}(\mathsf{B}, \mathsf{B}, -)$ are unused by the rest of the protocol and can also be eliminated. The resulting version Init of the initial part of the real protocol is therefore as follows:

- $\mathsf{In}(\mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(\mathsf{A}, i)$ for $i < N$
- $\mathsf{InRcvd}(\mathsf{B}, i)^{\mathsf{B}}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(\mathsf{B}, i);\ \mathsf{ret}\ \checkmark$ for $i < M$

- $\mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i) \coloneqq x \leftarrow \mathsf{In}(\mathsf{A},i);\ \mathsf{samp\ flip}$ for $i < N$
- $\mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{B},i) \coloneqq x \leftarrow \mathsf{In}(\mathsf{B},i);\ \mathsf{samp\ flip}$ for $i < M$
- $\mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i)^{\mathsf{A}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare\text{-}\$}(\mathsf{B},\mathsf{A},i)^{\mathsf{A}}_{\mathsf{adv}} \coloneqq x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i);\ x \leftarrow \mathsf{In}(\mathsf{A},i);\ \mathsf{ret}\ x_A \oplus x$ for $i < N$
- $\mathsf{SendInShare}(\mathsf{B},\mathsf{A},i)^{\mathsf{A}}_{\mathsf{adv}} \coloneqq x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i);\ x \leftarrow \mathsf{In}(\mathsf{A},i);\ \mathsf{ret}\ x_A \oplus x$ for $i < N$
- $\mathsf{RcvdInShare}(\mathsf{A},\mathsf{B},i)^{\mathsf{A}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}(\mathsf{A},\mathsf{A},i) \coloneqq \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{A},\mathsf{B},i) \coloneqq \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}(\mathsf{A},\mathsf{A},i)^{\mathsf{A}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{A},\mathsf{B},i)^{\mathsf{A}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{B},i)$ for $i < M$

This is followed by the hiding of the channels

- $\mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i)$ for $i < N$,
- $\mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{B},i)$ for $i < M$.

We now eliminate the channels $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},-)$, $\mathsf{Share}(\mathsf{B},-)$ from the inductive part of the real protocol. We can extract the computation of the channels $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},-)$ into a separate protocol $\mathsf{RcvdBits}(C,K)$:

- $\mathsf{RcvdBits}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{RcvdBits}\big(C;\ \textit{input-gate}(\mathsf{A},i), K+1\big)$ is the composition of $\mathsf{RcvdBits}(C,K)$ with the protocol
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
- $\mathsf{RcvdBits}\big(C;\ \textit{input-gate}(\mathsf{B},i), K+1\big)$ is the composition of $\mathsf{RcvdBits}(C,K)$ with the protocol
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
- $\mathsf{RcvdBits}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{RcvdBits}(C,K)$ with the protocol
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
- $\mathsf{RcvdBits}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{RcvdBits}(C,K)$ with the protocol
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K)$
- $\mathsf{RcvdBits}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{RcvdBits}(C,K)$ with the protocol
  - $\mathsf{RcvdBit}(\mathsf{B},\mathsf{A},K) \coloneqq b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);$ $x_B \leftarrow \mathsf{Share}(\mathsf{B},k);\ y_B \leftarrow \mathsf{Share}(\mathsf{B},l);\ \mathsf{ret}\ b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$

After the extraction, the protocol $\mathsf{Shares}(C,K)$ is left looking as follows:

- $\mathsf{Shares}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(\mathsf{A},i), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{Share}(\mathsf{A},K) \coloneqq \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{A},i)$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(\mathsf{B},i), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol

- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
- $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{B},i)$

- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k)$
- $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  - $\mathsf{Share}(\mathsf{A},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ x_A \oplus y_A$
- $\mathsf{Shares}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{samp\ flip}$
  - $\mathsf{Share}(\mathsf{A},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ (x_A * y_A) \oplus b_A$

None of the channels defined by $\mathsf{RcvdBits}(C,K)$ are utilized anywhere outside of $\mathsf{RcvdBits}(C,K)$ and as such we may discard this protocol fragment entirely. This in particular eliminates all references to the channels $\mathsf{Share}(\mathsf{B},-)$ from the inductive part of the protocol. To summarize, the inductive part of the real protocol now consists of the protocols $\mathsf{Shares}(C,K)$ and $\mathsf{Adv}(C,K)$, followed by the hiding of the channels

- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},k)$ for $k < K$.

We recall that on the top level we also have the protocol $\mathsf{Wires}(C,K)$ and the channels below:

- $\mathsf{Share}\text{-}\Sigma(k) := \mathsf{read}\ \mathsf{Wire}(k)$ for $k < K$
- $\mathsf{Share}(\mathsf{B},k) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ \mathsf{ret}\ x_A \oplus x$ for $k < K$

We now eliminate all references to Bob's shares from the final part of the protocol. If wire k is an output, then the channels

- $\mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k)$
- $\mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{B},k)$

read from the channel

- $\mathsf{Share}(\mathsf{B},k) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ \mathsf{ret}\ x_A \oplus x$

so we may substitute:

- $\mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ \mathsf{ret}\ x_A \oplus x$
- $\mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ \mathsf{ret}\ x_A \oplus x$

We thus get the following for channels $\mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},-)^{\mathsf{A}}_{\mathsf{adv}}$ and $\mathsf{OutShare}(\mathsf{A},\mathsf{B},-)^{\mathsf{A}}_{\mathsf{adv}}$:

- $$\begin{cases} \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$$
- $$\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$$

The top-level internal channels $\mathsf{Share}(\mathsf{B}, -)$ are now unused by the rest of the protocol and can be eliminated. The resulting version Fin of the final part of the real protocol is as follows:

- $\begin{cases} \mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{RcvdOutShare}(\mathsf{A}, \mathsf{B}, k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(\mathsf{A}, \mathsf{B}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(\mathsf{A}, \mathsf{B}, k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A}, \mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A}, \mathsf{B}, k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A}, \mathsf{B}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A}, \mathsf{B}, k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{Out}(\mathsf{A}, k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{A}, k) := \mathsf{read}\ \mathsf{Out}(\mathsf{A}, k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{Out}(\mathsf{B}, k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{B}, k) := \mathsf{read}\ \mathsf{Out}(\mathsf{B}, k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\mathsf{Out}(\mathsf{A}, k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(\mathsf{A}, k)$ for $k < K$

As a final step before the extraction of the simulator, we eliminate any reference to the channels $\mathsf{Share}\text{-}\Sigma(-)$ from the final part of the real protocol. If wire k is an output, then we can substitute the channel

- $\mathsf{Share}\text{-}\Sigma(k) := \mathsf{read}\ \mathsf{Wire}(k)$

into the channels

- $\mathsf{RcvdOutShare}(\mathsf{A}, \mathsf{B}, k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ \mathsf{ret}\ x_A \oplus x$
- $\mathsf{OutShare}(\mathsf{A}, \mathsf{B}, k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(k);\ \mathsf{ret}\ x_A \oplus x$

which yields the following:

- $\mathsf{RcvdOutShare}(\mathsf{A}, \mathsf{B}, k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_A \oplus x$
- $\mathsf{OutShare}(\mathsf{A}, \mathsf{B}, k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A}, k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_A \oplus x$

We thus get the following for channels $\mathsf{RcvdOutShare}(\mathsf{A}, \mathsf{B}, -)^{\mathsf{A}}_{\mathsf{adv}}$ and $\mathsf{OutShare}(\mathsf{A}, \mathsf{B}, -)^{\mathsf{A}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

If wire k is an output, then the channels

- $\mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(k)$
- $\mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(k)$

read from the channel

- $\mathsf{Share}\text{-}\Sigma(k) := \mathsf{read}\ \mathsf{Wire}(k)$

so we may substitute:

- $\mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Wire}(k)$
- $\mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Wire}(k)$

We thus get the following for channels $\mathsf{Out}(\mathsf{A},-)$ and $\mathsf{Out}(\mathsf{B},-)$:

- $\begin{cases} \mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Out}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

The top-level internal channels $\mathsf{Share}\text{-}\Sigma(-)$ are now unused by the rest of the protocol and can be eliminated. The resulting version Fin of the final part of the real protocol is as follows:

- $\begin{cases} \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Out}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\mathsf{Out}(\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k)$ for $k < K$

9.4.8 Extracting The Simulator

We are now ready to extract the simulator. The internal protocol $\mathsf{Wires}(C, K)$ together with the output channels

- $\begin{cases} \mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Out}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

will be factored out as coming from the ideal functionality. In particular, this leaves us with following version Fin of the final part of the soon-to-be simulator:

- $\begin{cases} \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\mathsf{Out}(\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k)$ for $k < K$

In the remainder of the soon-to-be simulator, we must eliminate any references to the channels $\mathsf{In}(\mathsf{A},-)$, $\mathsf{Wire}(-)$, $\mathsf{Out}(\mathsf{A},-)$, $\mathsf{Out}(\mathsf{B},-)$. We begin with the initial part of the real protocol. Recall the leakage

- $\mathsf{In}(\mathsf{A},i)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{In}(\mathsf{A},i)$ for $i < N$
- $\mathsf{InRcvd}(\mathsf{B},i)^{\mathsf{id}}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(\mathsf{B},i);\ \mathsf{ret}\ \checkmark$ for $i < M$

from the ideal functionality. In the presence of these channels we can write the protocol Init – forming the initial part of the simulator – equivalently as follows:

- $\mathsf{In}(\mathsf{A},i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(\mathsf{A},i)^{\mathsf{id}}_{\mathsf{adv}}$ for $i < N$
- $\mathsf{InRcvd}(\mathsf{B},i)^{\mathsf{B}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(\mathsf{B},i)^{\mathsf{id}}_{\mathsf{adv}}$ for $i < M$
- $\mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i) := x \leftarrow \mathsf{In}(\mathsf{A},i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{samp\ flip}$ for $i < N$
- $\mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{B},i) := \_ \leftarrow \mathsf{InRcvd}(\mathsf{B},i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{samp\ flip}$ for $i < M$
- $\mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare\text{-}\$}(\mathsf{B},\mathsf{A},i)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i);\ x \leftarrow \mathsf{In}(\mathsf{A},i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_A \oplus x$ for $i < N$
- $\mathsf{SendInShare}(\mathsf{B},\mathsf{A},i)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i);\ x \leftarrow \mathsf{In}(\mathsf{A},i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_A \oplus x$ for $i < N$
- $\mathsf{RcvdInShare}(\mathsf{A},\mathsf{B},i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}(\mathsf{A},\mathsf{A},i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{A},\mathsf{B},i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{B},i)$ for $i < M$
- $\mathsf{InShare}(\mathsf{A},\mathsf{A},i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{A},i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{A},\mathsf{B},i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(\mathsf{A},\mathsf{B},i)$ for $i < M$

This is followed by the hiding of the channels

- $\mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{A},i)$ for $i < N$,
- $\mathsf{InShare\text{-}\$}(\mathsf{A},\mathsf{B},i)$ for $i < M$.

We continue with the final part of the soon-to-be simulator. Recall the definition of the output channels

- $\begin{cases} \mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(\mathsf{B},k) := \mathsf{read}\ \mathsf{Out}(\mathsf{B},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

in the ideal functionality. If wire k is an output, then the channels

- $\mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_A \oplus x$
- $\mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_A \oplus x$

can be expressed equivalently as

- $\mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k);\ \mathsf{ret}\ x_A \oplus x$
- $\mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k);\ \mathsf{ret}\ x_A \oplus x$

since we have the definition below:

- $\mathsf{Out}(\mathsf{A},k) := \mathsf{read}\ \mathsf{Wire}(k)$

We thus get the following for channels $\mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},-)^{\mathsf{A}}_{\mathsf{adv}}$ and $\mathsf{OutShare}(\mathsf{A},\mathsf{B},-)^{\mathsf{A}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k);\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k);\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

The latest version Fin of the final part of the soon-to-be simulator is therefore as follows:

- $\begin{cases} \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k);\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k);\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\mathsf{Out}(\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k)$ for $k < K$

We now recall the leakage

- $\mathsf{Out}(\mathsf{A},k)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k)$ for $k < K$

from the ideal functionality. If wire k is an output, then the channels

- $\mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k);\ \mathsf{ret}\ x_A \oplus x$
- $\mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k);\ \mathsf{ret}\ x_A \oplus x$

can be expressed equivalently as

- $\mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_A \oplus x$
- $\mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_A \oplus x$

We thus get the following for channels $\mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},-)^{\mathsf{A}}_{\mathsf{adv}}$ and $\mathsf{OutShare}(\mathsf{A},\mathsf{B},-)^{\mathsf{A}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

Finally, the channels

- $\mathsf{Out}(\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k)$ for $k < K$

can be expressed equivalently as

- $\mathsf{Out}(\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k)^{\mathsf{id}}_{\mathsf{adv}}$ for $k < K$

The final version Fin of the final part of the simulator is therefore as follows:

- $\begin{cases} \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\mathsf{Out}(\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k)^{\mathsf{id}}_{\mathsf{adv}}$ for $k < K$

9.4.9 The Simulator

The simulator consists of three parts. In the initial phase, we have the protocol Init:

- $\mathsf{In}(\mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(\mathsf{A}, i)^{\mathsf{id}}_{\mathsf{adv}}$ for $i < N$
- $\mathsf{InRcvd}(\mathsf{B}, i)^{\mathsf{B}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(\mathsf{B}, i)^{\mathsf{id}}_{\mathsf{adv}}$ for $i < M$
- $\mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i) := x \leftarrow \mathsf{In}(\mathsf{A}, i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{samp\ flip}$ for $i < N$
- $\mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{B}, i) := \_ \leftarrow \mathsf{InRcvd}(\mathsf{B}, i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{samp\ flip}$ for $i < M$
- $\mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare\text{-}\$}(\mathsf{B}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i);\ x \leftarrow \mathsf{In}(\mathsf{A}, i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_A \oplus x$ for $i < N$
- $\mathsf{SendInShare}(\mathsf{B}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i);\ x \leftarrow \mathsf{In}(\mathsf{A}, i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_A \oplus x$ for $i < N$
- $\mathsf{RcvdInShare}(\mathsf{A}, \mathsf{B}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{B}, i)$ for $i < M$
- $\mathsf{InShare}(\mathsf{A}, \mathsf{A}, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{A}, \mathsf{B}, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{B}, i)$ for $i < M$
- $\mathsf{InShare}(\mathsf{A}, \mathsf{A}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{A}, i)$ for $i < N$
- $\mathsf{InShare}(\mathsf{A}, \mathsf{B}, i)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{B}, i)$ for $i < M$

This is followed by the hiding of the channels

- $\mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{A}, i)$ for $i < N$,
- $\mathsf{InShare\text{-}\$}(\mathsf{A}, \mathsf{B}, i)$ for $i < M$.

In the inductive phase, we have the protocol $\mathsf{Circ}(C, K)$, obtained by merging the two protocols $\mathsf{Shares}(C, K)$ and $\mathsf{Adv}(C, K)$ from earlier:

- $\mathsf{Circ}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Circ}\big(C;\ \textit{input-gate}(\mathsf{A}, i), K + 1\big)$ is the composition of $\mathsf{Circ}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{A}, i)$
  - $\mathsf{Share}(\mathsf{A}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A}, K)$
  - $\mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
  - $\mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A}, \mathsf{B}, K)^{\mathsf{ot}}_{\mathsf{adv}}$
- $\mathsf{Circ}\big(C;\ \textit{input-gate}(\mathsf{B}, i), K + 1\big)$ is the composition of $\mathsf{Circ}(C, K)$ with the protocol
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)$
  - $\mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A}, \mathsf{B}, K)^{\mathsf{A}}_{\mathsf{adv}}$
  - $\mathsf{Share}(\mathsf{A}, K) := \mathsf{read}\ \mathsf{InShare}(\mathsf{A}, \mathsf{B}, i)$

– $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$

– $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
– $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
– $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
– $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

– $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
– $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

- $\mathsf{Circ}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol

  – $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  – $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$

  – $\mathsf{Share}(\mathsf{A},K) := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k)$
  – $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$

  – $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  – $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  – $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  – $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

  – $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  – $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

- $\mathsf{Circ}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol

  – $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$
  – $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}}$

  – $\mathsf{Share}(\mathsf{A},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ x_A \oplus y_A$
  – $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$

  – $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  – $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  – $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  – $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

  – $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$
  – $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}}$

- $\mathsf{Circ}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol

  – $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K) := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{samp\ flip}$
  – $\mathsf{SendBit}(\mathsf{A},\mathsf{B},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(\mathsf{A},\mathsf{B},K)$

  – $\mathsf{Share}(\mathsf{A},K) := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ (x_A * y_A) \oplus b_A$
  – $\mathsf{Share}(\mathsf{A},K)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},K)$

  – $\mathsf{OTMsg}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A$
  – $\mathsf{OTMsg}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A$
  – $\mathsf{OTMsg}_2(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus y_A$
  – $\mathsf{OTMsg}_3(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := b_A \leftarrow \mathsf{SendBit}(\mathsf{A},\mathsf{B},K);\ x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ y_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ b_A \oplus x_A \oplus y_A$

  – $\mathsf{OTChcRcvd}_0(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ \mathsf{ret}\ \checkmark$

– $\mathsf{OTChcRcvd}_1(\mathsf{A},\mathsf{B},K)^{\mathsf{ot}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},l);\ \mathsf{ret}\ \checkmark$

This is followed by the hiding of the channels

- $\mathsf{SendBit}(\mathsf{A},\mathsf{B},k)$ for $k < K$.

In the final phase, we have the protocol Fin:

- $\begin{cases} \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(\mathsf{B},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(\mathsf{A},k) \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := x_A \leftarrow \mathsf{Share}(\mathsf{A},k);\ x \leftarrow \mathsf{Out}(\mathsf{A},k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_A \oplus x \\ \quad \text{for } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(\mathsf{A},\mathsf{B},k)^{\mathsf{A}}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\mathsf{Out}(\mathsf{A},k)^{\mathsf{A}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(\mathsf{A},k)^{\mathsf{id}}_{\mathsf{adv}}$ for $k < K$

The composition of the three parts is followed by the hiding of the channels

- $\mathsf{InShare}(\mathsf{A},\mathsf{A},i)$ for $i < N$,
- $\mathsf{InShare}(\mathsf{A},\mathsf{B},i)$ for $i < M$,
- $\mathsf{Share}(\mathsf{A},k)$ for $k < K$.

Composing the ideal protocol with the simulator, and substituting away the ideal leakage

- $\mathsf{In}(\mathsf{A},i)^{\mathsf{id}}_{\mathsf{adv}}$ for $i < N$,
- $\mathsf{InRcvd}(\mathsf{B},i)^{\mathsf{id}}_{\mathsf{adv}}$ for $i < M$,
- $\mathsf{Out}(\mathsf{A},k)^{\mathsf{id}}_{\mathsf{adv}}$ for $k < K$,
- $\mathsf{Out}(\mathsf{B},k)^{\mathsf{id}}_{\mathsf{adv}}$ for $k < K$

as indicated in Section 9.4.8 yields precisely the version of the real protocol we had at the end of Section 9.4.7.

10 Multi-Party GMW Protocol

In the multi-party GMW protocol, $N+2$ parties labeled $0,\ldots,N+1$ jointly compute the value of a given Boolean circuit built out of *xor*-, *and*-, and *not* gates. The inputs to the circuit are divided among the parties, and no party has access to the inputs of any other. Analogously to the two-party case, party n maintains its share of the actual value v computed by each gate, and summing up the shares $n := 0,\ldots,N+1$ yields back v. We prove the protocol secure in the case when party N is semi-honest, party $N+1$ is honest, and any other party is arbitrarily honest or semi-honest. When carrying out the 1-Out-Of-4 Oblivious Transfer between parties $n < m$, we assume n is the sender and m is the receiver.

Formally, we assume a coin-flip distribution $\mathsf{flip} : \mathsf{1} \twoheadrightarrow \mathsf{Bool}$; a Boolean sum function $\oplus : \mathsf{Bool} \times \mathsf{Bool} \to \mathsf{Bool}$, where we write $x \oplus y$ in place of $\oplus\ (x,y)$; a Boolean multiplication function $* : \mathsf{Bool} \times \mathsf{Bool} \to \mathsf{Bool}$, where we write $x * y$ in place of $*\ (x,y)$; and a Boolean negation function $\neg : \mathsf{Bool} \to \mathsf{Bool}$, where we write $\neg x$ in place of $\neg\ x$.

We represent Boolean circuits using the syntax below, where we assume that party $p := 0,\ldots,N+1$ has $I_p \geqslant 0$ inputs labeled $\{0,\ldots,I_p - 1\}$. Starting from the empty circuit ϵ, we add one gate at a time: an *input* gate allows us to plug into a specified input i of party p; a *not* gate negates the value carried on wire k; an *xor* gate computes the Boolean sum of the two values carried on wires k and l; and an *and* gate does the same for Boolean product.

$$
\begin{array}{lll}
\text{Parties} & p \in \mathbb{N} & \\
\text{Inputs} & i \in \mathbb{N} & \\
\text{Wires} & k, l \in \mathbb{N} & \\
\text{Circuits} & C & ::= \ \epsilon \mid C;\ \textit{input-gate}(p,i) \mid C;\ \textit{not-gate}(k) \mid C;\ \textit{xor-gate}(k,l) \mid C;\ \textit{and-gate}(k,l)
\end{array}
$$

A circuit C with $n \in \mathbb{N}$ wires is considered well-formed if each logical gate combines previously defined wires only:

$$
\frac{}{\epsilon\ \mathsf{circuit}(0)} \qquad \frac{C\ \mathsf{circuit}(n) \qquad 0 \leqslant p \leqslant N+1 \qquad i < I_p}{C;\ \textit{input-gate}(p,i)\ \mathsf{circuit}(n+1)} \qquad \frac{C\ \mathsf{circuit}(n) \qquad k < n}{C;\ \textit{not-gate}(k)\ \mathsf{circuit}(n+1)}
$$

$$
\frac{C\ \mathsf{circuit}(n) \qquad k < n \qquad 0 \leqslant l < n}{C;\ \textit{xor-gate}(k,l)\ \mathsf{circuit}(n+1)} \qquad \frac{C\ \mathsf{circuit}(n) \qquad k < n \qquad 0 \leqslant l < n}{C;\ \textit{and-gate}(k,l)\ \mathsf{circuit}(n+1)}
$$

We now fix an ambient Boolean circuit C with K wires $\{0,\ldots,K-1\}$, a subset of which is designated as outputs.

10.1 The Assumptions

At the expression level, we assume that the Boolean sum and product operations are commutative and associative:

- $x : \mathsf{Bool}, y : \mathsf{Bool} \vdash x \oplus y = y \oplus x : \mathsf{Bool}$,
- $x : \mathsf{Bool}, y : \mathsf{Bool} \vdash x * y = y * x : \mathsf{Bool}$,
- $x : \mathsf{Bool}, y : \mathsf{Bool}, z : \mathsf{Bool} \vdash (x \oplus y) \oplus z = x \oplus (y \oplus z) : \mathsf{Bool}$, and
- $x : \mathsf{Bool}, y : \mathsf{Bool}, z : \mathsf{Bool} \vdash (x * y) * z = x * (y * z) : \mathsf{Bool}$.

Furthermore, Boolean multiplication distributes over Boolean sum:

- $x : \mathsf{Bool}, y : \mathsf{Bool}, z : \mathsf{Bool} \vdash (x \oplus y) * z = (x * z) \oplus (y * z) : \mathsf{Bool}$.

Summing up a Boolean with itself yields false and summing up a Boolean with false yields the original Boolean:

- $x : \mathsf{Bool} \vdash x \oplus x = \mathsf{false} : \mathsf{Bool}$, and
- $x : \mathsf{Bool} \vdash x \oplus \mathsf{false} = x : \mathsf{Bool}$.

Negating a Boolean equals summing it up with true:

- $x : \mathsf{Bool} \vdash x \oplus \mathsf{true} = \neg x : \mathsf{Bool}$.

Finally, multiplying a Boolean with false or true yields false or the original Boolean, respectively:

- $x : \mathsf{Bool} \vdash x * \mathsf{false} = \mathsf{false} : \mathsf{Bool}$, and
- $x : \mathsf{Bool} \vdash x * \mathsf{true} = x : \mathsf{Bool}$.

At the distribution level, we assume that the distribution flip on Booleans is invariant under the operation of Boolean sum with a fixed Boolean (as is indeed the case when flip is uniform):

- $x : \mathsf{Bool} \vdash \big(y \leftarrow \mathsf{flip};\ \mathsf{ret}\ x \oplus y\big) = \mathsf{flip} : \mathsf{Bool}$

10.2 The Ideal Protocol

The leakage from the ideal functionality includes the value of each input i belonging to a semi-honest party n, plus the timing information for each input i belonging to an honest party n:

- $\begin{cases} \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i) \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$

In the inductive phase, the functionality computes the value carried by each wire $k < K$ of the ambient circuit by induction on the circuit:

- $\mathsf{Wires}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Wires}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Wires}(C,K)$ with the protocol
  - $\mathsf{Wire}(K) := \mathsf{read}\ \mathsf{In}(p,i)$
- $\mathsf{Wires}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Wires}(C,K)$ with the protocol
  - $\mathsf{Wire}(K) := x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ \neg x$
- $\mathsf{Wires}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Wires}(C,K)$ with the protocol
  - $\mathsf{Wire}(K) := x \leftarrow \mathsf{Wire}(k);\ y \leftarrow \mathsf{Wire}(l);\ \mathsf{ret}\ x \oplus y$
- $\mathsf{Wires}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Wires}(C,K)$ with the protocol
  - $\mathsf{Wire}(K) := x \leftarrow \mathsf{Wire}(k);\ y \leftarrow \mathsf{Wire}(l);\ \mathsf{ret}\ x * y$

After performing the above computation, the ideal functionality outputs the computed value for each wire marked as an output, and leaks the outputs to the adversary on behalf of each semi-honest party:

- $\begin{cases} \mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } n \leqslant n+1 \text{ and } k < K \text{ if wire } k \text{ an output} \\ \mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant n+1 \text{ and } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{Out}(n,k)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{Out}(n,k)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

Finally, the channels

- $\mathsf{Wire}(k)$ for $k < K$

coming from the inductive protocol $\mathsf{Wires}(C, K)$ are designated as internal.

10.3 The Real Protocol

The real protocol consists of the $N + 2$ parties, plus an instance of the ideal 1-Out-Of-4 Oblivious Transfer (OT) functionality for each gate and each pair of parties $m, n \leqslant N + 1$. The code for each party is separated into three parts: in the initial phase, each party computes and distributes everyone's shares for each of its inputs. In the inductive phase, each party computes their share of each wire by induction on the ambient circuit. At last, in the final phase, parties send their shares of each output wire to one another and add them up to compute the result. We now describe the code for party n.

10.3.1 Initial Phase

A semi-honest party n leaks the value of each of its inputs, whereas an honest party only leaks the fact that an input has been received:

- $\begin{cases} \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i) \\ \quad \text{for } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } i < I_n \text{ if } n \text{ honest} \\ \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$

We next randomly generate shares for every party except party $N + 1$:

- $\mathsf{InShare\text{-}\$}(m,n,i) := x \leftarrow \mathsf{In}(n,i);\ \mathsf{samp}\ \mathsf{flip}$ for $m \leqslant N$ and $i < I_n$

- $\begin{cases} \mathsf{InShare\text{-}\$}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare\text{-}\$}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

To determine the share of party $N + 1$, we inductively compute the sum of all the shares we generated above:

- $\begin{cases} \mathsf{InShare\text{-}\$\text{-}}\Sigma(0,n,i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(0,n,i) \\ \quad \text{for } i < I_n \\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m+1,n,i) := x_\Sigma \leftarrow \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i);\ x_{m+1} \leftarrow \mathsf{InShare\text{-}\$}(m+1,n,i);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \text{ and } i < I_n \end{cases}$

- $\begin{cases} \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

We compute the share of party $N + 1$ by summing up the input i with the shares of all the other parties:

- $\mathsf{InShare\text{-}\$}(N+1,n,i) := x_\Sigma \leftarrow \mathsf{InShare\text{-}\$\text{-}}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ x_\Sigma \oplus x$ for $i < I_n$

- $\begin{cases} \textsf{InShare-\$}(N+1,n,i)^{\textsf{party}(n)}_{\textsf{adv}} := \textsf{read InShare-\$}(N+1,n,i) \\ \quad \text{for } i < I_n \text{ if } n \text{ semi-honest} \\ \textsf{InShare-\$}(N+1,n,i)^{\textsf{party}(n)}_{\textsf{adv}} := \textsf{read InShare-\$}(N+1,n,i)^{\textsf{party}(n)}_{\textsf{adv}} \\ \quad \text{for } i < I_n \text{ if } n \text{ honest} \end{cases}$

This concludes the computation of shares for inputs belonging to party n. We next send each computed share to the respective party:

- $\textsf{SendInShare}(m,n,i) := \textsf{read InShare-\$}(m,n,i)$ for $m \leqslant N+1$ and $i < I_n$
- $\begin{cases} \textsf{SendInShare}(m,n,i)^{\textsf{party}(n)}_{\textsf{adv}} := \textsf{read SendInShare}(m,n,i) \\ \quad \text{for } m \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \textsf{SendInShare}(m,n,i)^{\textsf{party}(n)}_{\textsf{adv}} := \textsf{read SendInShare}(m,n,i)^{\textsf{party}(n)}_{\textsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

If party n is semi-honest, each input share received for an input i belonging to party m is forwarded to the adversary:

- $\begin{cases} \textsf{RcvdInShare}(n,m,i)^{\textsf{party}(n)}_{\textsf{adv}} := \textsf{read SendInShare}(n,m,i) \\ \quad \text{for } m \leqslant N+1 \text{ and } i < I_m \text{ if } n \text{ semi-honest} \\ \textsf{RcvdInShare}(n,m,i)^{\textsf{party}(n)}_{\textsf{adv}} := \textsf{read RcvdInShare}(n,m,i)^{\textsf{party}(n)}_{\textsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ and } i < I_m \text{ if } n \text{ honest} \end{cases}$

Finally, the incoming input shares are recorded:

- $\textsf{InShare}(n,m,i) := \textsf{read SendInShare}(n,m,i)$ for $m \leqslant N+1$ and $i < I_m$
- $\begin{cases} \textsf{InShare}(n,m,i)^{\textsf{party}(n)}_{\textsf{adv}} := \textsf{read InShare}(n,m,i) \\ \quad \text{for } m \leqslant N+1 \text{ and } i < I_m \text{ if } n \text{ semi-honest} \\ \textsf{InShare}(n,m,i)^{\textsf{party}(n)}_{\textsf{adv}} := \textsf{read InShare}(n,m,i)^{\textsf{party}(n)}_{\textsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ and } i < I_m \text{ if } n \text{ honest} \end{cases}$

The channels

- $\textsf{InShare-\$}(m,n,i)$ for $m \leqslant N+1$ and $i < I_n$,
- $\textsf{InShare-\$-}\Sigma(m,n,i)$ for $m \leqslant N$ and $i < I_n$

are declared as internal.

10.3.2 Inductive Phase

In the case of an *input* gate, we use the corresponding input share from the initial phase. In the case of a *not* gate, parties $0,\ldots,N$ simply copy their share of the incoming wire, whereas party $N+1$ negates its share. If the gate is an *xor* gate, the resulting share is the sum of the shares of the incoming two wires. The case of an *and* gate is the most complex. The sum of everybody's shares must equal $\big(x_0 \oplus \ldots \oplus x_{N+1}\big) * \big(y_0 \oplus \ldots \oplus y_{N+1}\big)$, where x_n, y_n are the respective shares of party n on the incoming two wires. We have

$$\big(x_0 \oplus \ldots \oplus x_{N+1}\big) * \big(y_0 \oplus \ldots \oplus y_{N+1}\big) = \bigoplus_i \bigoplus_j x_i * y_j$$

As in the two-party case, parties n and m engage in an idealized 1-Out-Of-4 OT exchange to compute the quantity $(x_n * y_m) \oplus (x_m * y_n)$, again appropriately masked by a random Boolean.

We again set up our protocol so that each gate induces the same set of outputs, even though some of these channels may not be relevant to the specific gate in question. For wire K, the relevant non-adversarial outputs are among the following:

- $\mathsf{SendBit}(n, m, K)$ for storing the masking Boolean is relevant for an *and* gate when $n < m$,
- $\mathsf{RcvdBit}(n, m, K)$ for receiving the result of the OT exchange is relevant for an *and* gate when $n < m$,
- $\mathsf{Ctrb}(n, m, K)$ for storing the contribution of party m to the share of party n is relevant for an *and* gate,
- $\mathsf{Ctrb}\text{-}\Sigma(n, m, K)$ for summing up the contributions of parties $0, \ldots, m$ to the share of party n is relevant for an *and* gate,
- $\mathsf{Share}(n, K)$ for storing the share of party n is always relevant
- $\mathsf{OTMsg}_0(n, m, K)$, $\mathsf{OTMsg}_1(n, m, K)$, $\mathsf{OTMsg}_2(n, m, K)$, $\mathsf{OTMsg}_3(n, m, K)$ for the OT exchange from n to m are relevant for an *and* gate if $n < m$
- $\mathsf{OTChc}_0(n, m, K)$, $\mathsf{OTChc}_1(n, m, K)$ for the OT exchange from m to n are relevant for an *and* gate if $n > m$.

We define the inductive part of the real protocol a follows:

- $\mathsf{Circ}_n(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Circ}_n\big(C;\, \textit{input-gate}(p, i), K + 1\big)$ is the composition of $\mathsf{Circ}_n(C, K)$ with the following protocol. Our share is the input share as determined in the initial part of the protocol:
  - $\mathsf{Share}(n, K) \coloneqq \mathsf{read}\ \mathsf{InShare}(n, p, i)$
  - $\begin{cases} \mathsf{Share}(n, K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n, K) \\ \quad \text{if } n \text{ semi-honest} \\ \mathsf{Share}(n, K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n, K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{if } n \text{ honest} \end{cases}$

  As we said earlier, the 1-Out-Of-4 OT exchange with every other party is vacuous – party n does not function as a sender,

  - $\mathsf{OTMsg}_0(n, m, K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n, m, K)$ for $m \leqslant N + 1$
  - $\mathsf{OTMsg}_1(n, m, K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(n, m, K)$ for $m \leqslant N + 1$
  - $\mathsf{OTMsg}_2(n, m, K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(n, m, K)$ for $m \leqslant N + 1$
  - $\mathsf{OTMsg}_3(n, m, K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(n, m, K)$ for $m \leqslant N + 1$

  nor as a receiver:

  - $\mathsf{OTChc}_0(m, n, K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(m, n, K)$ for $m \leqslant N + 1$
  - $\mathsf{OTChc}_1(m, n, K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_1(m, n, K)$ for $m \leqslant N + 1$

  Since no OT exchange is taking place, no masking Boolean is needed:

  - $\mathsf{SendBit}(n, m, K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n, m, K)$ for $m \leqslant N + 1$
  - $\mathsf{SendBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $m \leqslant N + 1$

  There is nothing to receive from the OT functionality:

  - $\mathsf{RcvdBit}(n, m, K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n, m, K)$ for $m \leqslant N + 1$
  - $\mathsf{RcvdBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $m \leqslant N + 1$

  The individual contributions of each party to our share are likewise not needed:

  - $\mathsf{Ctrb}(n, m, K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n, m, K)$ for $m \leqslant N + 1$
  - $\mathsf{Ctrb}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $m \leqslant N + 1$

There is thus nothing to sum up:

- $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $m \leqslant N+1$

- $\mathsf{Circ}_n\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Circ}_n(C,K)$ with the following protocol. Our share is either the share on wire k (for parties $n \leqslant N$) or its negation (for party $N+1$):

  - $\begin{cases} \mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } n \leqslant N \\ \mathsf{Share}(N+1,K) := x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ \neg x_{N+1} \end{cases}$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{if } n \text{ honest} \end{cases}$

  The 1-Out-Of-4 OT exchange with every other party is again vacuous – party n does not function as a sender,

  - $\mathsf{OTMsg}_0(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$ for $m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)$ for $m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)$ for $m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)$ for $m \leqslant N+1$

  nor as a receiver:

  - $\mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_0(m,n,K)$ for $m \leqslant N+1$
  - $\mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_1(m,n,K)$ for $m \leqslant N+1$

  Since no OT exchange is taking place, no masking Boolean is needed:

  - $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $m \leqslant N+1$
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $m \leqslant N+1$

  There is nothing to receive from the OT functionality:

  - $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $m \leqslant N+1$

  The individual contributions of each party to our share are likewise not needed:

  - $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $m \leqslant N+1$

  There is thus nothing to sum up:

  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $m \leqslant N+1$

- $\mathsf{Circ}_n\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}_n(C,K)$ with the following protocol. Our share is the sum of shares on wires k and l:

  - $\mathsf{Share}(n,K) := x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n \oplus y_n$

- $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{if } n \text{ honest} \end{cases}$

The 1-Out-Of-4 OT exchange with every other party is once more vacuous – party n does not function as a sender,

- $\mathsf{OTMsg}_0(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$ for $m \leqslant N+1$
- $\mathsf{OTMsg}_1(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)$ for $m \leqslant N+1$
- $\mathsf{OTMsg}_2(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)$ for $m \leqslant N+1$
- $\mathsf{OTMsg}_3(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)$ for $m \leqslant N+1$

nor as a receiver:

- $\mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_0(m,n,K)$ for $m \leqslant N+1$
- $\mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_1(m,n,K)$ for $m \leqslant N+1$

Since no OT exchange is taking place, no masking Boolean is needed:

- $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $m \leqslant N+1$
- $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $m \leqslant N+1$

There is nothing to receive from the OT functionality:

- $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $m \leqslant N+1$
- $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $m \leqslant N+1$

The individual contributions of each party to our share are likewise not needed:

- $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $m \leqslant N+1$
- $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $m \leqslant N+1$

There is thus nothing to sum up:

- $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $m \leqslant N+1$

• $\mathsf{Circ}_n\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}_n(C,K)$ with the following protocol. First, for each party m with $n < m$ we generate the masking Boolean for the OT exchange where n is a sender and m is a receiver:

- $\begin{cases} \mathsf{SendBit}(n,m,K) := x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{samp\ flip} \\ \quad \text{for } m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

We now carry out the 1-Out-Of-4 OT exchanges where n is the sender:

- $$\begin{cases} \mathsf{OTMsg}_0(n,m,K) := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \\ \quad \text{for } m \leqslant N+1 \text{ if } n < m \\ \mathsf{OTMsg}_0(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K) \\ \quad \text{for } m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$$
- $$\begin{cases} \mathsf{OTMsg}_1(n,m,K) := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \\ \quad \text{for } m \leqslant N+1 \text{ if } n < m \\ \mathsf{OTMsg}_1(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K) \\ \quad \text{for } m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$$
- $$\begin{cases} \mathsf{OTMsg}_2(n,m,K) := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus y_n \\ \quad \text{for } m \leqslant N+1 \text{ if } n < m \\ \mathsf{OTMsg}_2(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K) \\ \quad \text{for } m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$$
- $$\begin{cases} \mathsf{OTMsg}_3(n,m,K) := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \oplus y_n \\ \quad \text{for } m \leqslant N+1 \text{ if } n < m \\ \mathsf{OTMsg}_3(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K) \\ \quad \text{for } m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$$

Next we carry out the 1-Out-Of-4 OT exchanges where n is the receiver:

- $$\begin{cases} \mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } m \leqslant N+1 \text{ if } m < n \\ \mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_0(m,n,K) \\ \quad \text{for } m \leqslant N+1 \text{ if } m \geqslant n \end{cases}$$
- $$\begin{cases} \mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{Share}(n,l) \\ \quad \text{for } m \leqslant N+1 \text{ if } m < n \\ \mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_1(m,n,K) \\ \quad \text{for } m \leqslant N+1 \text{ if } m \geqslant n \end{cases}$$

We now record the bits we received from the 1-Out-Of-4 OT exchanges:

- $\mathsf{RcvdBit}(n,m,K) := \mathsf{OTOut}(m,n,K)$ for $m \leqslant N+1$
- $$\begin{cases} \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

For each party m, we compute its contribution to our share as follows:

- $$\begin{cases} \mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}(n,n,K) := x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n * y_n \end{cases}$$
- $$\begin{cases} \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K) \\ \quad \text{for } m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

We now inductively sum up the individual contributions we generated above:

– $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,0,K) := \mathsf{read}\ \mathsf{Ctrb}(n,0,K) \\ \mathsf{Ctrb}\text{-}\Sigma(n,m+1,K) := b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,K);\ b \leftarrow \mathsf{Ctrb}(n,m+1,K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } m \leqslant N \end{cases}$

– $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K) \\ \quad \text{for } m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

At last, we declare our share to be the total sum of the contributions:

– $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,N+1,K)$

– $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{if } n \text{ honest} \end{cases}$

Finally, the channels

- $\mathsf{SendBit}(n,m,k)$ for $m \leqslant N+1$ and $k < K$,
- $\mathsf{RcvdBit}(n,m,k)$ for $m \leqslant N+1$ and $k < K$,
- $\mathsf{Ctrb}(n,m,k)$ for $m \leqslant N+1$ and $k < K$,
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,k)$ for $m \leqslant N+1$ and $k < K$

are declared as internal.

10.3.3 The Final Phase

For each output wire, we broadcast our share:

- $\begin{cases} \mathsf{SendOutShare}(m,n,k) := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } m \leqslant N+1 \text{ and } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(m,n,k) := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k) \\ \quad \text{for } m \leqslant N+1 \text{ and } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k) \\ \quad \text{for } m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

If party n is semi-honest, each output share received is forwarded to the adversary:

- $\begin{cases} \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(n,m,k) \\ \quad \text{for } m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

The incoming output shares are recorded:

- $\mathsf{OutShare}(n,m,k) := \mathsf{read}\ \mathsf{SendOutShare}(n,m,k)$ for $m \leqslant N+1$ and $k < K$

- $\begin{cases} \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,m,k) \\ \quad \text{for } m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

We now inductively sum up the output shares we recorded above:

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,0,k) := \mathsf{read}\ \mathsf{OutShare}(n,0,k) \\ \quad \text{for } k < K \\ \mathsf{OutShare}\text{-}\Sigma(n,m+1,k) := x_\Sigma \leftarrow \mathsf{OutShare}\text{-}\Sigma(n,m,k);\ x_{m+1} \leftarrow \mathsf{OutShare}(n,m+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m \leqslant N \text{ and } k < K \end{cases}$

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k) \\ \quad \text{for } m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

Finally, we declare the output to be the total sum of the output shares:

- $\mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)$ for $k < K$
- $\begin{cases} \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } k < K \text{ if } n \text{ honest} \end{cases}$

Finally, the channels

- $\mathsf{OutShare}(n,m,k)$ for $m \leqslant N+1$ and $k < K$,
- $\mathsf{OutShare}\text{-}\Sigma(n,-,-)$ for $m \leqslant N+1$ and $k < K$

are declared as internal.

10.3.4 1-Out-Of-4 Oblivious Transfer Functionality

For each wire $k < K$ and parties n, m we have a separate idealized 1-Out-Of-4 Oblivious Transfer functionality $\mathsf{1OutOf4OT}(n,m,k)$, where party n is the sender and party m is the receiver.
If the sender is semi-honest, the functionality leaks the value of all messages received from the sender:

- $\begin{cases} \mathsf{OTMsg}_0(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,k) \\ \quad \text{if } n \text{ semi-honest} \\ \mathsf{OTMsg}_0(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{OTMsg}_1(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,k) \\ \quad \text{if } n \text{ semi-honest} \\ \mathsf{OTMsg}_1(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{OTMsg}_2(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,k) \\ \quad \text{if } n \text{ semi-honest} \\ \mathsf{OTMsg}_2(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{if } n \text{ honest} \end{cases}$

- $$\begin{cases} \mathsf{OTMsg}_3(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,k) \\ \quad \text{if } n \text{ semi-honest} \\ \mathsf{OTMsg}_3(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{if } n \text{ honest} \end{cases}$$

Otherwise the functionality only lets the adversary know that a message from the sender has been received:

- $$\begin{cases} \mathsf{OTMsgRcvd}_0(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := m_0 \leftarrow \mathsf{OTMsg}_0(n,m,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_0(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{if } n \text{ semi-honest} \end{cases}$$
- $$\begin{cases} \mathsf{OTMsgRcvd}_1(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := m_1 \leftarrow \mathsf{OTMsg}_1(n,m,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_1(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{if } n \text{ semi-honest} \end{cases}$$
- $$\begin{cases} \mathsf{OTMsgRcvd}_2(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := m_2 \leftarrow \mathsf{OTMsg}_2(n,m,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_2(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{if } n \text{ semi-honest} \end{cases}$$
- $$\begin{cases} \mathsf{OTMsgRcvd}_3(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := m_3 \leftarrow \mathsf{OTMsg}_3(n,m,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_3(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{if } n \text{ semi-honest} \end{cases}$$

Analogously, if the receiver is semi-honest, the functionality leaks the value of all messages from the receiver:

- $$\begin{cases} \mathsf{OTChc}_0(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,k) \\ \quad \text{if } m \text{ semi-honest} \\ \mathsf{OTChc}_0(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{if } m \text{ honest} \end{cases}$$
- $$\begin{cases} \mathsf{OTChc}_1(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,k) \\ \quad \text{if } m \text{ semi-honest} \\ \mathsf{OTChc}_1(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{if } m \text{ honest} \end{cases}$$

Otherwise the functionality only lets the adversary know that a message from the receiver has been received:

- $$\begin{cases} \mathsf{OTChcRcvd}_0(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(n,m,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{if } m \text{ honest} \\ \mathsf{OTChcRcvd}_0(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{if } m \text{ semi-honest} \end{cases}$$
- $$\begin{cases} \mathsf{OTChcRcvd}_1(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := c_1 \leftarrow \mathsf{OTChc}_1(n,m,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{if } m \text{ honest} \\ \mathsf{OTChcRcvd}_1(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{if } m \text{ semi-honest} \end{cases}$$

The functionality then selects the appropriate message:

- $\mathsf{OTOut}(n,m,k) := m_0 \leftarrow \mathsf{OTMsg}_0(n,m,k);\ m_1 \leftarrow \mathsf{OTMsg}_1(n,m,k);\ m_2 \leftarrow \mathsf{OTMsg}_2(n,m,k);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(n,m,k);\ c_0 \leftarrow \mathsf{OTChc}_0(n,m,k);\ c_1 \leftarrow \mathsf{OTChc}_1(n,m,k);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then\ ret}\ m_3\ \mathsf{else\ ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then\ ret}\ m_1\ \mathsf{else\ ret}\ m_0\big)$

If the receiver is semi-honest, the functionality leaks the selected message to the adversary:

- $\begin{cases} \mathsf{OTOut}(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,k) \\ \quad \text{if } m \text{ semi-honest} \\ \mathsf{OTOut}(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,k)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{if } m \text{ honest} \end{cases}$

10.3.5 The Real Protocol

The complete code for party n arises as the composition of its initial, inductive, and final phases, followed by the hiding of the communication internal to party n - namely, the channels

- $\mathsf{InShare}(n,m,i)$ for $m \leqslant N+1$ and $i < I_m$,
- $\mathsf{Share}(n,k)$ for $k < K$.

The real protocol is a composition of the $N+2$ parties, plus an instance of the circuit-wide OT functionality for each pair of parties $n, m \leqslant N+1$,

- $\mathsf{1OutOf4OT}(n,m,k)$ for $k < K$,

all followed by the hiding of the internal communication among the two parties and the functionalities – namely the channels

- $\mathsf{SendInShare}(m,n,i)$ for $m,n \leqslant N+1$ and $i < I_n$,
- $\mathsf{OTMsg}_0(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{OTMsg}_1(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{OTMsg}_2(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{OTMsg}_3(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{OTChc}_0(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{OTChc}_1(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{OTOut}(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{SendOutShare}(m,n,k)$ for $m,n \leqslant N+1$ and $k < K$.

10.4 Real = Ideal + Simulator

Our goal is to keep simplifying the real protocol until it becomes clear how to extract out a suitable simulator. We first restructure the entire protocol as a composition of an initial part, an inductive part, and a final part, all followed by the hiding of the channels

- $\mathsf{InShare}(m,n,i)$ for $m,n \leqslant N+1$ and $i < I_n$,
- $\mathsf{Share}(n,k)$ for $n \leqslant N+1$ and $k < K$.

The initial part of the real protocol arises by composing together the respective initial parts for each party, and declaring their communication as internal. Specifically, we have the following protocol Init:

- $\begin{cases} \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i) \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$

- $\mathsf{InShare\text{-}\$}(m,n,i) := x \leftarrow \mathsf{In}(n,i);\ \mathsf{samp\ flip}$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$

- $\begin{cases} \mathsf{InShare\text{-}\$}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare\text{-}\$}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InShare\text{-}\$\text{-}}\Sigma(0,n,i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(0,n,i) \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m+1,n,i) := x_\Sigma \leftarrow \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i);\ x_{m+1} \leftarrow \mathsf{InShare\text{-}\$}(m+1,n,i);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \text{ and } n \leqslant N+1 \text{ and } i < I_n \end{cases}$

- $\begin{cases} \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\mathsf{InShare\text{-}\$}(N+1,n,i) := x_\Sigma \leftarrow \mathsf{InShare\text{-}\$\text{-}}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ x_\Sigma \oplus x$ for $n \leqslant N+1$ and $i < I_n$

- $\begin{cases} \mathsf{InShare\text{-}\$}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(N+1,n,i) \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare\text{-}\$}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\mathsf{SendInShare}(m,n,i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i)$ for $m,n \leqslant N+1$ and $i < I_n$

- $\begin{cases} \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(m,n,i) \\ \quad \text{for } m,n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m,n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(m,n,i) \\ \quad \text{for } m,n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} \\ \quad \text{for } m,n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$

- $\mathsf{InShare}(m,n,i) := \mathsf{read}\ \mathsf{SendInShare}(m,n,i)$ for $m,n \leqslant N+1$ and $i < I_n$

- $\begin{cases} \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(m,n,i) \\ \quad \text{for } m,n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} \\ \quad \text{for } m,n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$

This is followed by the hiding of the channels

- $\mathsf{InShare}\text{-}\$(m,n,i)$ for $m,n \leqslant N+1$ and $i < I_n$,
- $\mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i)$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$,
- $\mathsf{SendInShare}(m,n,i)$ for $m,n \leqslant N+1$ and $i < I_n$.

The inductive part of the real protocol arises by composing together the respective inductive parts for each party plus the circuit-wide OT functionalities, and declaring the communication with the OT functionalities as internal. Specifically, we have the following protocol $\mathsf{Circ}(C,K)$:

- $\mathsf{Circ}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Circ}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{InShare}(n,p,i)$ for $n \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_0 \leftarrow \mathsf{OTMsg}_0(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_1 \leftarrow \mathsf{OTMsg}_1(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_2 \leftarrow \mathsf{OTMsg}_2(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_3 \leftarrow \mathsf{OTMsg}_3(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_0(m,n,K)$ for $m,n \leqslant N+1$
- $\mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_1(m,n,K)$ for $m,n \leqslant N+1$
- $\begin{cases} \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_1(m,n,K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_1(m,n,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } m,n \leqslant N+1 \text{ if } m \text{ honest} \\ \mathsf{OTChcRcvd}_1(m,n,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(m,n,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } m,n \leqslant N+1 \text{ if } m \text{ semi-honest} \end{cases}$

- $\mathsf{OTOut}(n,m,K) \coloneqq m_0 \leftarrow \mathsf{OTMsg}_0(n,m,K);\ m_1 \leftarrow \mathsf{OTMsg}_1(n,m,K);\ m_2 \leftarrow \mathsf{OTMsg}_2(n,m,K);$ $m_3 \leftarrow \mathsf{OTMsg}_3(n,m,K);\ c_0 \leftarrow \mathsf{OTChc}_0(n,m,K);\ c_1 \leftarrow \mathsf{OTChc}_1(n,m,K);$ $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$ for $n, m \leqslant N+1$
- $\begin{cases} \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$

- $\mathsf{Circ}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n, m \leqslant N+1$
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n, m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } n \leqslant N \\ \mathsf{Share}(N+1,K) \coloneqq x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ \neg x_{N+1} \end{cases}$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)$ for $n, m \leqslant N+1$
  - $\begin{cases} \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_0 \leftarrow \mathsf{OTMsg}_0(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_1 \leftarrow \mathsf{OTMsg}_1(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_2 \leftarrow \mathsf{OTMsg}_2(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_3 \leftarrow \mathsf{OTMsg}_3(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_0(m,n,K)$ for $m,n \leqslant N+1$
- $\mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_1(m,n,K)$ for $m,n \leqslant N+1$
- $\begin{cases} \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_1(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \end{cases}$
- $\mathsf{OTOut}(n,m,K) := m_0 \leftarrow \mathsf{OTMsg}_0(n,m,K);\ m_1 \leftarrow \mathsf{OTMsg}_1(n,m,K);\ m_2 \leftarrow \mathsf{OTMsg}_2(n,m,K);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(n,m,K);\ c_0 \leftarrow \mathsf{OTChc}_0(n,m,K);\ c_1 \leftarrow \mathsf{OTChc}_1(n,m,K);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$ for $n,m \leqslant N+1$

- $\begin{cases} \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$

- $\mathsf{Circ}\big(C;\, \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n \oplus y_n$ for $n \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq m_0 \leftarrow \mathsf{OTMsg}_0(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}$

- $\begin{cases} \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_1 \leftarrow \mathsf{OTMsg}_1(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_2 \leftarrow \mathsf{OTMsg}_2(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_3 \leftarrow \mathsf{OTMsg}_3(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_0(m,n,K)$ for $m,n \leqslant N+1$
- $\mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_1(m,n,K)$ for $m,n \leqslant N+1$
- $\begin{cases} \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_1(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \end{cases}$
- $\mathsf{OTOut}(n,m,K) := m_0 \leftarrow \mathsf{OTMsg}_0(n,m,K);\ m_1 \leftarrow \mathsf{OTMsg}_1(n,m,K);\ m_2 \leftarrow \mathsf{OTMsg}_2(n,m,K);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(n,m,K);\ c_0 \leftarrow \mathsf{OTChc}_0(n,m,K);\ c_1 \leftarrow \mathsf{OTChc}_1(n,m,K);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$ for $n,m \leqslant N+1$
- $\begin{cases} \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$

- $\mathsf{Circ}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit}(n,m,K) := x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{samp}\ \mathsf{flip} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$

$$- \begin{cases} \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

- $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{OTOut}(n,m,K)$ for $n,m \leqslant N+1$

$$- \begin{cases} \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

$$- \begin{cases} \mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}(n,n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n * y_n \\ \quad \text{for } n \leqslant N+1 \end{cases}$$

$$- \begin{cases} \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

$$- \begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,0,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,0,K) \\ \quad \text{for } n \leqslant N+1 \\ \mathsf{Ctrb}\text{-}\Sigma(n,m+1,K) \coloneqq b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,K);\ b \leftarrow \mathsf{Ctrb}(n,m+1,K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \end{cases}$$

$$- \begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

- $\mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,N+1,K)$ for $n \leqslant N+1$

$$- \begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

$$- \begin{cases} \mathsf{OTMsg}_0(n,m,K) \coloneqq b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{OTMsg}_0(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$$

$$- \begin{cases} \mathsf{OTMsg}_1(n,m,K) \coloneqq b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{OTMsg}_1(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$$

$$
- \begin{cases} \mathsf{OTMsg}_2(n,m,K) := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{OTMsg}_2(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}
$$

$$
- \begin{cases} \mathsf{OTMsg}_3(n,m,K) := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{OTMsg}_3(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}
$$

$$
- \begin{cases} \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}
$$

$$
- \begin{cases} \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}
$$

$$
- \begin{cases} \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}
$$

$$
- \begin{cases} \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}
$$

$$
- \begin{cases} \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_0 \leftarrow \mathsf{OTMsg}_0(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}
$$

$$
- \begin{cases} \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_1 \leftarrow \mathsf{OTMsg}_1(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}
$$

$$
- \begin{cases} \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_2 \leftarrow \mathsf{OTMsg}_2(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}
$$

$$
- \begin{cases} \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_3 \leftarrow \mathsf{OTMsg}_3(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \end{cases}
$$

$$
- \begin{cases} \mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } m,n \leqslant N+1 \text{ if } m < n \\ \mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_0(m,n,K) \\ \quad \text{for } m,n \leqslant N+1 \text{ if } m \geqslant n \end{cases}
$$

– $\begin{cases} \mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{Share}(n,l) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_1(m,n,K) \\ \quad \text{for } m,n \leqslant N+1 \text{ if } m \geqslant n \end{cases}$

– $\begin{cases} \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$

– $\begin{cases} \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$

– $\begin{cases} \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \end{cases}$

– $\begin{cases} \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := c_1 \leftarrow \mathsf{OTChc}_1(n,m,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \end{cases}$

– $\mathsf{OTOut}(n,m,K) := m_0 \leftarrow \mathsf{OTMsg}_0(n,m,K);\ m_1 \leftarrow \mathsf{OTMsg}_1(n,m,K);\ m_2 \leftarrow \mathsf{OTMsg}_2(n,m,K);$ $m_3 \leftarrow \mathsf{OTMsg}_3(n,m,K);\ c_0 \leftarrow \mathsf{OTChc}_0(n,m,K);\ c_1 \leftarrow \mathsf{OTChc}_1(n,m,K);$ $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$ for $n,m \leqslant N+1$

– $\begin{cases} \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$

This is followed by the hiding of the channels

- $\mathsf{SendBit}(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{RcvdBit}(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{Ctrb}(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{OTMsg}_0(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{OTMsg}_1(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$
- $\mathsf{OTMsg}_2(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{OTMsg}_3(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{OTChc}_0(m,n,k)$ for $m,n \leqslant N+1$ and $k < K$,
- $\mathsf{OTChc}_1(m,n,k)$ for $m,n \leqslant N+1$ and $k < K$,
- $\mathsf{OTOut}(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$.

The final part of the real protocol arises by composing together the respective final parts for each party, and declaring their communication as internal. Specifically, we have the following protocol Fin:

- $\begin{cases} \mathsf{SendOutShare}(m,n,k) \coloneqq \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } m,n \leqslant N+1 \text{ and } k < K \text{ if wire } k \text{ an output} \\ \mathsf{SendOutShare}(m,n,k) \coloneqq \mathsf{read}\ \mathsf{SendOutShare}(m,n,k) \\ \quad \text{for } m,n \leqslant N+1 \text{ and } k < K \text{ if wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendOutShare}(m,n,k) \\ \quad \text{for } m,n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m,n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendOutShare}(n,m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

- $\mathsf{OutShare}(n,m,k) \coloneqq \mathsf{read}\ \mathsf{SendOutShare}(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$

- $\begin{cases} \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OutShare}(n,m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,0,k) \coloneqq \mathsf{ret}\ \mathsf{OutShare}(n,0,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \\ \mathsf{OutShare}\text{-}\Sigma(n,m+1,k) \coloneqq x_\Sigma \leftarrow \mathsf{OutShare}\text{-}\Sigma(n,m,k);\ x_{m+1} \leftarrow \mathsf{OutShare}(n,m+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \end{cases}$

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

- $\mathsf{Out}(n,k) \coloneqq \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)$ for $n \leqslant N+1$ and $k < K$

- $\begin{cases} \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

This is followed by the hiding of the channels

- $\mathsf{OutShare}(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{OutShare}\text{-}\Sigma(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{SendOutShare}(m,n,k)$ for $m,n \leqslant N+1$ and $k < K$.

10.4.1 Simplifying The Real Protocol: Initial Phase

If party n is semi-honest, then for any party m and input $i < I_n$ the channel

- $\mathsf{SendInShare}(m, n, i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(m, n, i)$

reads from the channel

- $\mathsf{SendInShare}(m, n, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m, n, i)$

so we can substitute:

- $\mathsf{SendInShare}(m, n, i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m, n, i)$

We thus get the following for channels $\mathsf{SendInShare}(-, -, -)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{SendInShare}(m, n, i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m, n, i) \\ \quad \text{for } m, n \leqslant N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{SendInShare}(m, n, i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(m, n, i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m, n \leqslant N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

If party n is semi-honest, then for any party m and input $i < I_m$ the channel

- $\mathsf{RcvdInShare}(n, m, i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(n, m, i)$

reads from the channel

- $\mathsf{SendInShare}(n, m, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(n, m, i)$

so we can substitute:

- $\mathsf{RcvdInShare}(n, m, i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(n, m, i)$

We thus get the following for channels $\mathsf{RcvdInShare}(-, -, -)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{RcvdInShare}(n, m, i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(n, m, i) \\ \quad \text{for } n, m \leqslant N + 1 \text{ and } i < I_m \text{ if } n \text{ semi-honest} \\ \mathsf{RcvdInShare}(n, m, i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(n, m, i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n, m \leqslant N + 1 \text{ and } i < I_m \text{ if } n \text{ honest} \end{cases}$

Finally, substituting the channels

- $\mathsf{SendInShare}(m, n, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m, n, i)$ for $m, n \leqslant N + 1$ and $i < I_n$

into the respective channels

- $\mathsf{InShare}(m, n, i) := \mathsf{read}\ \mathsf{SendInShare}(m, n, i)$ for $m, n \leqslant N + 1$ and $i < I_n$

yields the definition below:

- $\mathsf{InShare}(m, n, i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m, n, i)$ for $m, n \leqslant N + 1$ and $i < I_n$

At this point, the internal channels $\mathsf{SendInShare}(-, -, -)$ are unused and can be eliminated. The simplified version Init of the initial part of the real protocol is therefore as follows:

- $\begin{cases} \mathsf{In}(n, i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n, i) \\ \quad \text{for } n \leqslant N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{In}(n, i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n, i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$
- $\mathsf{InShare}\text{-}\$(m,n,i) := x \leftarrow \mathsf{In}(n,i);\ \mathsf{samp}\ \mathsf{flip}$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$
- $\begin{cases} \mathsf{InShare}\text{-}\$(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare}\text{-}\$(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{InShare}\text{-}\$\text{-}\Sigma(0,n,i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$(0,n,i) \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m+1,n,i) := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i);\ x_{m+1} \leftarrow \mathsf{InShare}\text{-}\$(m+1,n,i);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \text{ and } n \leqslant N+1 \text{ and } i < I_n \end{cases}$
- $\begin{cases} \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{InShare}\text{-}\$(N+1,n,i) := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ x_\Sigma \oplus x$ for $n \leqslant N+1$ and $i < I_n$
- $\begin{cases} \mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(N+1,n,i) \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(m,n,i) \\ \quad \text{for } m,n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m,n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(m,n,i) \\ \quad \text{for } m,n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} \\ \quad \text{for } m,n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$
- $\mathsf{InShare}(m,n,i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$(m,n,i)$ for $m,n \leqslant N+1$ and $i < I_n$
- $\begin{cases} \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(m,n,i) \\ \quad \text{for } m,n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} \\ \quad \text{for } m,n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$

This is followed by the hiding of the channels

- $\mathsf{InShare}\text{-}\$(m,n,i)$ for $m,n \leqslant N+1$ and $i < I_n$,
- $\mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i)$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$.

10.4.2 Simplifying The Real Protocol: Inductive Phase

Our next order of business is to eliminate all channels interacting with the OT functionalities. In the case of *input*-, *not*-, and *xor* gates, the OT channels are divergent and only appear in the corresponding leakage channels. The leakage channels themselves are therefore divergent. For instance, if party n is semi-honest, for any party m the channel

- $\mathsf{OTMsg}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n, m, K)$

reads from the divergent channel

- $\mathsf{OTMsg}_0(n, m, K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n, m, K)$

and so we may equivalently write the following:

- $\mathsf{OTMsg}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$

The simplified definition for channels $\mathsf{OTMsg}_0(-, -, K)^{\mathsf{ot}}_{\mathsf{adv}}$ is thus as follows:

- $\mathsf{OTMsg}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N + 1$

If party n is honest, for any party m the channel

- $\mathsf{OTMsgRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq m_0 \leftarrow \mathsf{OTMsg}_0(n, m, K);\ \mathsf{ret}\ \checkmark$

reads from the divergent channel

- $\mathsf{OTMsg}_0(n, m, K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n, m, K)$

and so we may equivalently write the following:

- $\mathsf{OTMsgRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$

The simplified definition for channels $\mathsf{OTMsgRcvd}_0(-, -, K)^{\mathsf{ot}}_{\mathsf{adv}}$ is thus as follows:

- $\mathsf{OTMsgRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N + 1$

We treat the receiver channels analogously. For instance, if party m is semi-honest, then for any party n the channel

- $\mathsf{OTChc}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(n, m, K)$

reads from the divergent channel

- $\mathsf{OTChc}_0(n, m, K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(n, m, K)$

and so we may equivalently write the following:

- $\mathsf{OTChc}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$

The simplified definition for channels $\mathsf{OTChc}_0(-, -, K)^{\mathsf{ot}}_{\mathsf{adv}}$ is thus as follows:

- $\mathsf{OTChc}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N + 1$

If party m is honest, for any party n the channel

- $\mathsf{OTChcRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq c_0 \leftarrow \mathsf{OTChc}_0(n, m, K);\ \mathsf{ret}\ \checkmark$

reads from the divergent channel

- $\mathsf{OTChc}_0(n, m, K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(n, m, K)$

and so we may equivalently write the following:

- $\mathsf{OTChcRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChcRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$

The simplified definition for channels $\mathsf{OTChcRcvd}_0(-, -, K)^{\mathsf{ot}}_{\mathsf{adv}}$ is thus as follows:

- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

Finally, for any parties n, m the channel

- $\mathsf{OTOut}(n,m,K) \coloneqq m_0 \leftarrow \mathsf{OTMsg}_0(n,m,K);\ m_1 \leftarrow \mathsf{OTMsg}_1(n,m,K);\ m_2 \leftarrow \mathsf{OTMsg}_2(n,m,K);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(n,m,K);\ c_0 \leftarrow \mathsf{OTChc}_0(n,m,K);\ c_1 \leftarrow \mathsf{OTChc}_1(n,m,K);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

reads from the divergent channel

- $\mathsf{OTMsg}_0(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$

and so we may equivalently write the following:

- $\mathsf{OTOut}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K)$

If party m is semi-honest, this in turn means that the channel

- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K)$

reads from a divergent channel, and so we may equivalently write the following:

- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$

The simplified definition for channels $\mathsf{OTOut}(-,-,K)^{\mathsf{ot}}_{\mathsf{adv}}$ is thus as follows:

- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

In the case of an *and* gate, we start by eliminating any mention of the OT channels from the leakage channels. For instance, if party n is semi-honest, for any party m with $n \geqslant m$ the channel

- $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$

reads from the divergent channel

- $\mathsf{OTMsg}_0(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$

and so we may equivalently write the following:

- $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$

On the other hand, if $n < m$ then the channel

- $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$

reads from the channel

- $\mathsf{OTMsg}_0(n,m,K) \coloneqq b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b$

so we may substitute:

- $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b$

Summarizing the above, we get the following for channels $\mathsf{OTMsg}_0(-,-,K)^{\mathsf{ot}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

Analogously for channels $\mathsf{OTMsg}_1(-,-,K)^{\mathsf{ot}}_{\mathsf{adv}}$, $\mathsf{OTMsg}_2(-,-,K)^{\mathsf{ot}}_{\mathsf{adv}}$, $\mathsf{OTMsg}_3(-,-,K)^{\mathsf{ot}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

If party n is honest, for any party m with $n \geqslant m$ the channel

- $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_0 \leftarrow \mathsf{OTMsg}_0(n,m,K);\ \mathsf{ret}\ \checkmark$

reads from the divergent channel

- $\mathsf{OTMsg}_0(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$

and so we may equivalently write the following:

- $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$

On the other hand, if $n < m$ then the channel

- $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := m_0 \leftarrow \mathsf{OTMsg}_0(n,m,K);\ \mathsf{ret}\ \checkmark$

reads from the channel

- $\mathsf{OTMsg}_0(n,m,K) := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b$

so we may substitute:

- $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark$

Summarizing the above, we get the following for channels $\mathsf{OTMsgRcvd}_0(-,-,K)^{\mathsf{ot}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

Analogously for channels $\mathsf{OTMsgRcvd}_1(-,-,K)^{\mathsf{ot}}_{\mathsf{adv}}$, $\mathsf{OTMsgRcvd}_2(-,-,K)^{\mathsf{ot}}_{\mathsf{adv}}$, $\mathsf{OTMsgRcvd}_3(-,-,K)^{\mathsf{ot}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

- $\begin{cases} \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

We treat the receiver channels similarly. For instance, if party m is semi-honest, then for any party n with $n \geqslant m$ the channel

- $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)$

reads from the divergent channel

- $\mathsf{OTChc}_0(n,m,K) := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)$

and so we may equivalently write the following:

- $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$

On the other hand, if $n < m$ then the channel

- $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)$

reads from the channel

- $\mathsf{OTChc}_0(n,m,K) := \mathsf{read}\ \mathsf{Share}(m,k)$

so we may substitute:

- $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k)$

Summarizing the above, we get the following for channels $\mathsf{OTChc}_0(-,-,K)^{\mathsf{ot}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$

Analogously for channels $\mathsf{OTChc}_1(-,-,K)^{\mathsf{ot}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,l) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$

If party m is honest, for any party n with $n \geqslant m$ the channel

- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(n,m,K);\ \mathsf{ret}\ \checkmark$

reads from the divergent channel

- $\mathsf{OTChc}_0(n,m,K) := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)$

and so we may equivalently write the following:

- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$

On the other hand, if $n < m$ then the channel

- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := c_0 \leftarrow \mathsf{OTChc}_0(n,m,K);\ \mathsf{ret}\ \checkmark$

reads from the channel

- $\mathsf{OTChc}_0(n, m, K) := \mathsf{read}\ \mathsf{Share}(m, k)$

so we may substitute:

- $\mathsf{OTChcRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := x_m \leftarrow \mathsf{Share}(m, k);\ \mathsf{ret}\ \checkmark$

Summarizing the above, we get the following for channels $\mathsf{OTChcRcvd}_0(-, -, K)^{\mathsf{ot}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OTChcRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := x_m \leftarrow \mathsf{Share}(m, k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n, m \leqslant N+1 \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n, m \leqslant N+1 \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$

Analogously for channels $\mathsf{OTChcRcvd}_1(-, -, K)^{\mathsf{ot}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OTChcRcvd}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := y_m \leftarrow \mathsf{Share}(m, l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n, m \leqslant N+1 \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n, m \leqslant N+1 \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$

At last, if party m is semi-honest, for any party n the channel

- $\mathsf{OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n, m, K)$

has the same definition as the channel

- $\mathsf{RcvdBit}(m, n, K) := \mathsf{read}\ \mathsf{OTOut}(n, m, K)$

so we may equivalently write the following:

- $\mathsf{OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(m, n, K)$

We thus get the following for channels $\mathsf{OTOut}(-, -, K)^{\mathsf{ot}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(m, n, K) \\ \quad \text{for } n, m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n, m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$

At this point, none of the leakage channels refer to any of the OT channels anymore. So outside of the OT channels themselves, the only place where we still refer to an OT channel is in the channel

- $\mathsf{RcvdBit}(n, m, K) := \mathsf{OTOut}(m, n, K)$

that stores the result of the OT exchange between parties m, n with m as a sender and n as a receiver. Substituting the channel

- $\mathsf{OTOut}(m, n, K) := m_0 \leftarrow \mathsf{OTMsg}_0(m, n, K);\ m_1 \leftarrow \mathsf{OTMsg}_1(m, n, K);\ m_2 \leftarrow \mathsf{OTMsg}_2(m, n, K);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(m, n, K);\ c_0 \leftarrow \mathsf{OTChc}_0(m, n, K);\ c_1 \leftarrow \mathsf{OTChc}_1(m, n, K);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

into the above thus yields the somewhat more verbose

- $\mathsf{RcvdBit}(n, m, K) := m_0 \leftarrow \mathsf{OTMsg}_0(m, n, K);\ m_1 \leftarrow \mathsf{OTMsg}_1(m, n, K);\ m_2 \leftarrow \mathsf{OTMsg}_2(m, n, K);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(m, n, K);\ c_0 \leftarrow \mathsf{OTChc}_0(m, n, K);\ c_1 \leftarrow \mathsf{OTChc}_1(m, n, K);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

with the advantage that it no longer mentions $\mathsf{OTOut}(m, n, K)$. If $m < n$ we may further substitute the channels

- $\mathsf{OTMsg}_0(m, n, K) := b \leftarrow \mathsf{SendBit}(m, n, K);\ x_m \leftarrow \mathsf{Share}(m, k);\ y_m \leftarrow \mathsf{Share}(m, l);\ \mathsf{ret}\ b$

- $\mathsf{OTMsg}_1(m,n,K) := b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);\ \mathsf{ret}\ b \oplus x_m$
- $\mathsf{OTMsg}_2(m,n,K) := b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);\ \mathsf{ret}\ b \oplus y_m$
- $\mathsf{OTMsg}_3(m,n,K) := b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);\ \mathsf{ret}\ b \oplus x_m \oplus y_m$

as well as the channels

- $\mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{Share}(n,k)$
- $\mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{Share}(n,l)$

to obtain the following:

- $\mathsf{RcvdBit}(n,m,K) := b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);\ x_n \leftarrow \mathsf{Share}(n,k);$
  $y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{if}\ x_n\ \mathsf{then}\ \big(\mathsf{if}\ y_n\ \mathsf{then}\ \mathsf{ret}\ b \oplus x_m \oplus y_m\ \mathsf{else}\ \mathsf{ret}\ b \oplus y_m\big)\ \mathsf{else}\ \big(\mathsf{if}\ y_n\ \mathsf{then}\ \mathsf{ret}\ b \oplus x_m\ \mathsf{else}\ \mathsf{ret}\ b\big)$

Here we no longer refer to any of the OT channels. We can express the above more concisely as follows:

- $\mathsf{RcvdBit}(n,m,K) := b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);$
  $x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$

Finally, if $m \geqslant n$ the channel

- $\mathsf{RcvdBit}(n,m,K) := m_0 \leftarrow \mathsf{OTMsg}_0(m,n,K);\ m_1 \leftarrow \mathsf{OTMsg}_1(m,n,K);\ m_2 \leftarrow \mathsf{OTMsg}_2(m,n,K);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(m,n,K);\ c_0 \leftarrow \mathsf{OTChc}_0(m,n,K);\ c_1 \leftarrow \mathsf{OTChc}_1(m,n,K);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$

reads from the divergent channel

- $\mathsf{OTMsg}_0(m,n,K) := \mathsf{read}\ \mathsf{OTMsg}_0(m,n,K)$

so we may equivalently write

- $\mathsf{RcvdBit}(n,m,K) := \mathsf{RcvdBit}(n,m,K)$

and further expand the above to

- $\mathsf{RcvdBit}(n,m,K) := b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);$
  $x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$

since the channel

- $\mathsf{SendBit}(m,n,K) := \mathsf{read}\ \mathsf{SendBit}(m,n,K)$

is likewise divergent. Summarizing, we get the following for channels $\mathsf{RcvdBit}(-,-,K)$:

- $\mathsf{RcvdBit}(n,m,K) := b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);$
  $x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n,m \leqslant N+1$

All in all, we can rewrite the inductive part $\mathsf{Circ}(C,K)$ of the real protocol as follows:

- $\mathsf{Circ}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Circ}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$

- $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{InShare}(n,p,i)$ for $n \leqslant N+1$
- $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{OTMsg}_0(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_1(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_2(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_3(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_0(m,n,K)$ for $m,n \leqslant N+1$
- $\mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_1(m,n,K)$ for $m,n \leqslant N+1$
- $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K) := \mathsf{read}\ \mathsf{OTOut}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

• $\mathsf{Circ}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol

- $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\begin{cases} \mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } n \leqslant N \\ \mathsf{Share}(N+1,K) \coloneqq x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ \neg x_{N+1} \end{cases}$

– $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

– $\mathsf{OTMsg}_0(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$ for $n,m \leqslant N+1$
– $\mathsf{OTMsg}_1(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)$ for $n,m \leqslant N+1$
– $\mathsf{OTMsg}_2(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)$ for $n,m \leqslant N+1$
– $\mathsf{OTMsg}_3(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)$ for $n,m \leqslant N+1$

– $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
– $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
– $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
– $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
– $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
– $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
– $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTChc}_0(m,n,K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(m,n,K)$ for $m,n \leqslant N+1$
– $\mathsf{OTChc}_1(m,n,K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_1(m,n,K)$ for $m,n \leqslant N+1$

– $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
– $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
– $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTOut}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K)$ for $n,m \leqslant N+1$
– $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Circ}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol

  – $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  – $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  – $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  – $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  – $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  – $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  – $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  – $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  – $\mathsf{Share}(n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n \oplus y_n$ for $n \leqslant N+1$

- $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{OTMsg}_0(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_1(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_2(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_3(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_0(m,n,K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(m,n,K)$ for $m,n \leqslant N+1$
- $\mathsf{OTChc}_1(m,n,K) \coloneqq \mathsf{read}\ \mathsf{OTChc}_1(m,n,K)$ for $m,n \leqslant N+1$
- $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Circ}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit}(n,m,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{samp\ flip} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
  - $\begin{cases} \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);$ $x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n<m \\ \mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m<n \\ \mathsf{Ctrb}(n,n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n * y_n \\ \quad \text{for } n \leqslant N+1 \end{cases}$
- $\begin{cases} \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,0,K) \coloneqq \mathsf{ret}\ \mathsf{Ctrb}(n,0,K) \\ \quad \text{for } n \leqslant N+1 \\ \mathsf{Ctrb}\text{-}\Sigma(n,m+1,K) \coloneqq b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,K);\ b \leftarrow \mathsf{Ctrb}(n,m+1,K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \end{cases}$
- $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,N+1,K)$ for $n \leqslant N+1$
- $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{OTMsg}_0(n,m,K) \coloneqq b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n<m \\ \mathsf{OTMsg}_0(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTMsg}_1(n,m,K) \coloneqq b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n<m \\ \mathsf{OTMsg}_1(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTMsg}_2(n,m,K) \coloneqq b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n<m \\ \mathsf{OTMsg}_2(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTMsg}_3(n,m,K) \coloneqq b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n<m \\ \mathsf{OTMsg}_3(n,m,K) \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n<m \\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

- $\begin{cases} \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } m,n \leqslant N+1 \text{ if } m < n \\ \mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_0(m,n,K) \\ \quad \text{for } m,n \leqslant N+1 \text{ if } m \geqslant n \end{cases}$
- $\begin{cases} \mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{Share}(n,l) \\ \quad \text{for } m,n \leqslant N+1 \text{ if } m < n \\ \mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_1(m,n,K) \\ \quad \text{for } m,n \leqslant N+1 \text{ if } m \geqslant n \end{cases}$
- $\begin{cases} \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,l) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$

- $\begin{cases} \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := x_m \leftarrow \mathsf{Share}(m,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := y_m \leftarrow \mathsf{Share}(m,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$
- $\mathsf{OTOut}(n,m,K) := m_0 \leftarrow \mathsf{OTMsg}_0(n,m,K);\ m_1 \leftarrow \mathsf{OTMsg}_1(n,m,K);\ m_2 \leftarrow \mathsf{OTMsg}_2(n,m,K);$ $m_3 \leftarrow \mathsf{OTMsg}_3(n,m,K);\ c_0 \leftarrow \mathsf{OTChc}_0(n,m,K);\ c_1 \leftarrow \mathsf{OTChc}_1(n,m,K);$ $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$ for $n,m \leqslant N+1$
- $\begin{cases} \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(m,n,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$

We now split the protocol $\mathsf{Circ}(C,K)$ into three parts. The first protocol $\mathsf{Shares}(C,K)$ performs the computation of shares and is defined as follows:

- $\mathsf{Shares}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{InShare}(n,p,i)$ for $n \leqslant N+1$
- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } n \leqslant N \\ \mathsf{Share}(N+1,K) := x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ \neg x_{N+1} \end{cases}$
- $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) := x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n \oplus y_n$ for $n \leqslant N+1$
- $\mathsf{Shares}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol

- $\begin{cases} \mathsf{SendBit}(n,m,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{samp\ flip} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read\ SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
- $\mathsf{RcvdBit}(n,m,K) \coloneqq b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);$ $x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n,m \leqslant N+1$
- $\begin{cases} \mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read\ SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read\ RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}(n,n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n * y_n \\ \quad \text{for } n \leqslant N+1 \end{cases}$
- $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,0,K) \coloneqq \mathsf{read\ Ctrb}(n,0,K) \\ \quad \text{for } n \leqslant N+1 \\ \mathsf{Ctrb}\text{-}\Sigma(n,m+1,K) \coloneqq b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,K);\ b \leftarrow \mathsf{Ctrb}(n,m+1,K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \end{cases}$
- $\mathsf{Share}(n,K) \coloneqq \mathsf{read\ Ctrb}\text{-}\Sigma(n,N+1,K)$ for $n \leqslant N+1$

The second protocol $\mathsf{Adv}(C,K)$ performs all leakages and is defined as follows:

- $\mathsf{Adv}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Adv}\big(C;\ \textit{input-gate}(p,i),K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read\ SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read\ RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read\ Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read\ Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read\ Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read\ Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read\ OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read\ OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read\ OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read\ OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read\ OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read\ OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read\ OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read\ OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read\ OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read\ OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read\ OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Adv}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Adv}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Adv}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol

  - $\begin{cases} \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$
  - $\begin{cases} \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$
  - $\begin{cases} \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

$$- \begin{cases} \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$$

$$- \begin{cases} \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$$

$$- \begin{cases} \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$$

$$- \begin{cases} \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$$

$$- \begin{cases} \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$$

$$- \begin{cases} \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$$

$$- \begin{cases} \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,l) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$$

$$- \begin{cases} \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := x_m \leftarrow \mathsf{Share}(m,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$$

$$- \begin{cases} \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := y_m \leftarrow \mathsf{Share}(m,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$$

$$- \begin{cases} \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(m,n,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$$

The third protocol $\mathsf{1OutOf4OT}(C,K)$ performs all Oblivious Transfer exchanges and is defined as follows:

- $\mathsf{1OutOf4OT}(\epsilon,0)$ is the protocol $\mathsf{0}$

- $\mathsf{1OutOf4OT}\big(C;\ \mathit{input\text{-}gate}(p,i), K+1\big)$ is the composition of $\mathsf{1OutOf4OT}(C,K)$ with the protocol
  - $\mathsf{OTMsg}_0(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_0(m,n,K)$ for $m,n \leqslant N+1$
  - $\mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_1(m,n,K)$ for $m,n \leqslant N+1$
  - $\mathsf{OTOut}(n,m,K) := \mathsf{read}\ \mathsf{OTOut}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{1OutOf4OT}\big(C;\ \mathit{not\text{-}gate}(k), K+1\big)$ is the composition of $\mathsf{1OutOf4OT}(C,K)$ with the protocol
  - $\mathsf{OTMsg}_0(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_0(m,n,K)$ for $m,n \leqslant N+1$
  - $\mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_1(m,n,K)$ for $m,n \leqslant N+1$
  - $\mathsf{OTOut}(n,m,K) := \mathsf{read}\ \mathsf{OTOut}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{1OutOf4OT}\big(C;\ \mathit{xor\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{1OutOf4OT}(C,K)$ with the protocol
  - $\mathsf{OTMsg}_0(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_0(m,n,K)$ for $m,n \leqslant N+1$
  - $\mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_1(m,n,K)$ for $m,n \leqslant N+1$
  - $\mathsf{OTOut}(n,m,K) := \mathsf{read}\ \mathsf{OTOut}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{1OutOf4OT}\big(C;\ \mathit{and\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{1OutOf4OT}(C,K)$ with the protocol
  - $\begin{cases} \mathsf{OTMsg}_0(n,m,K) := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{OTMsg}_0(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
  - $\begin{cases} \mathsf{OTMsg}_1(n,m,K) := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{OTMsg}_1(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
  - $\begin{cases} \mathsf{OTMsg}_2(n,m,K) := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{OTMsg}_2(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$

- $\begin{cases} \mathsf{OTMsg}_3(n,m,K) := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{OTMsg}_3(n,m,K) := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } m,n \leqslant N+1 \text{ if } m < n \\ \mathsf{OTChc}_0(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_0(m,n,K) \\ \quad \text{for } m,n \leqslant N+1 \text{ if } m \geqslant n \end{cases}$
- $\begin{cases} \mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{Share}(n,l) \\ \quad \text{for } m,n \leqslant N+1 \text{ if } m < n \\ \mathsf{OTChc}_1(m,n,K) := \mathsf{read}\ \mathsf{OTChc}_1(m,n,K) \\ \quad \text{for } m,n \leqslant N+1 \text{ if } m \geqslant n \end{cases}$
- $\mathsf{OTOut}(n,m,K) := m_0 \leftarrow \mathsf{OTMsg}_0(n,m,K);\ m_1 \leftarrow \mathsf{OTMsg}_1(n,m,K);\ m_2 \leftarrow \mathsf{OTMsg}_2(n,m,K);$
  $m_3 \leftarrow \mathsf{OTMsg}_3(n,m,K);\ c_0 \leftarrow \mathsf{OTChc}_0(n,m,K);\ c_1 \leftarrow \mathsf{OTChc}_1(n,m,K);$
  $\mathsf{if}\ c_0\ \mathsf{then}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_3\ \mathsf{else}\ \mathsf{ret}\ m_2\big)\ \mathsf{else}\ \big(\mathsf{if}\ c_1\ \mathsf{then}\ \mathsf{ret}\ m_1\ \mathsf{else}\ \mathsf{ret}\ m_0\big)$ for $n,m \leqslant N+1$

At this point, none of the channels defined by $\mathsf{1OutOf4OT}(C,K)$ are utilized anywhere outside of $\mathsf{1OutOf4OT}(C,K)$ and as such we may discard this protocol entirely. The inductive part of the real protocol therefore consists of the protocols $\mathsf{Shares}(C,K)$ and $\mathsf{Adv}(C,K)$, followed by the hiding of the channels

- $\mathsf{SendBit}(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{RcvdBit}(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{Ctrb}(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$.

10.4.3 Simplifying The Real Protocol: Final Phase

If party n is semi-honest and wire k is not an output, then for any party m the channel

- $\mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k)$

reads from the divergent channel

- $\mathsf{SendOutShare}(m,n,k) := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k)$

and thus we may equivalently write the following:

- $\mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}}$

If party n is semi-honest and wire k is an output, then for any party m the channel

- $\mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k)$

reads from the channel

- $\mathsf{SendOutShare}(m,n,k) := \mathsf{read}\ \mathsf{Share}(n,k)$

so we may substitute:

- $\mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,k)$

Summarizing the above, we get the following for channels $\mathsf{SendOutShare}(-,-,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } m,n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m,n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

If party n is semi-honest and wire k is not an output, then for any party m the channel

- $\mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(n,m,k)$

reads from the divergent channel

- $\mathsf{SendOutShare}(n,m,k) := \mathsf{read}\ \mathsf{SendOutShare}(n,m,k)$

and thus we may again write the following:

- $\mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}}$

If party n is semi-honest and wire k is an output, then for any party m the channel

- $\mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(n,m,k)$

reads from the channel

- $\mathsf{SendOutShare}(n,m,k) := \mathsf{read}\ \mathsf{Share}(m,k)$

so we may substitute:

- $\mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k)$

Summarizing the above, we get the following for channels $\mathsf{RcvdOutShare}(-,-,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(n,m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

If wire $k < K$ is not an output, then for any parties n, m the channel

- $\mathsf{OutShare}(n,m,k) := \mathsf{read}\ \mathsf{SendOutShare}(n,m,k)$

reads from the divergent channel

- $\mathsf{SendOutShare}(n,m,k) := \mathsf{read}\ \mathsf{SendOutShare}(n,m,k)$

and thus we may equivalently write the following:

- $\mathsf{OutShare}(n,m,k) := \mathsf{read}\ \mathsf{OutShare}(n,m,k)$

If wire $k < K$ is an output, then for any parties n, m the channel

- $\mathsf{OutShare}(n,m,k) := \mathsf{read}\ \mathsf{SendOutShare}(n,m,k)$

reads from the channel

- $\mathsf{SendOutShare}(n,m,k) := \mathsf{read}\ \mathsf{Share}(m,k)$

so we may substitute:

- $\mathsf{OutShare}(n,m,k) := \mathsf{read}\ \mathsf{Share}(m,k)$

Summarizing the above, we get the following for channels $\mathsf{OutShare}(-,-,-)$:

- $$\begin{cases} \mathsf{OutShare}(n,m,k) := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(n,m,k) := \mathsf{read}\ \mathsf{OutShare}(n,m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if wire } k \text{ not an output} \end{cases}$$

At this point, the internal channels $\mathsf{SendOutShare}(-,-,-)$ are unused and can be eliminated. The simplified version Fin of the final part of the real protocol is therefore as follows:

- $$\begin{cases} \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } m,n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m,n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$$

- $$\begin{cases} \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$$

- $$\begin{cases} \mathsf{OutShare}(n,m,k) := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if wire } k \text{ an output} \\ \mathsf{OutShare}(n,m,k) := \mathsf{read}\ \mathsf{OutShare}(n,m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if wire } k \text{ not an output} \end{cases}$$

- $$\begin{cases} \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$$

- $$\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,0,k) := \mathsf{read}\ \mathsf{OutShare}(n,0,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \\ \mathsf{OutShare}\text{-}\Sigma(n,m+1,k) := x_\Sigma \leftarrow \mathsf{OutShare}\text{-}\Sigma(n,m,k);\ x_{m+1} \leftarrow \mathsf{OutShare}(n,m+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \end{cases}$$

- $$\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$$

- $\mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)$ for $n \leqslant N+1$ and $k < K$

- $$\begin{cases} \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$$

This is followed by the hiding of the channels

- $\mathsf{OutShare}(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$,
- $\mathsf{OutShare}\text{-}\Sigma(n,m,k)$ for $n,m \leqslant N+1$ and $k < K$.

10.4.4 Timing of Shares I

Since the last party $N+1$ is by assumption honest, the simulator does not have access to its inputs. Therefore, any computation that depends on the value of the inputs belonging to party $N+1$ must be eliminated. In particular, all shares of party $N+1$ must be eliminated.

By design, summing up the respective shares $\bigoplus_{i \leqslant N+1} x_i$ of each party on a given wire yields the actual value x carried by the wire. If we got our hands on x, for example by inductively computing the circuit the same way the ideal functionality does, we could replace the share x_{N+1} of the last party by the sum $x \oplus \left(\bigoplus_{i \leqslant N} x_i\right)$ of x and the shares $x_0, \ldots, x_N$. To this end, we introduce new internal channels

- $\begin{cases} \mathsf{Share}\text{-}\Sigma(0,k) := \mathsf{read}\ \mathsf{Share}(0,k) \\ \quad \text{for } k < K \\ \mathsf{Share}\text{-}\Sigma(m+1,k) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,k);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \text{ and } k < K \end{cases}$

that inductively compute the the sum of shares for parties $0, \ldots, N$ on each wire $k < K$. For our strategy to work, however, we need to arrange the timing so that party $N+1$ computes its shares after everybody else.

To this end, we introduce new internal channels

- $\mathsf{InShare}\text{-}\checkmark(m,n,i) := \_ \leftarrow \mathsf{read}\ \mathsf{InShare}(m,n,i);\ \mathsf{ret}\ \checkmark$ for $m, n \leqslant N+1$ and $i < I_n$

for the timing of input shares in the initial part of the protocol and

- $\mathsf{Share}\text{-}\checkmark(n,k) := \_ \leftarrow \mathsf{read}\ \mathsf{Share}(n,k);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N+1$ and $k < K$

for the timing of shares in the inductive part of the protocol.

Since the primary job of the simulator is to construct the appropriate leakage, we start by eliminating any mention of the last party's shares from the leakage channels. Upon carefully examining the inductive part of the real protocol, we see that the only place where we leak information depending on the shares of party $N+1$ is when we leak the timing of the party's shares on behalf of the OT functionality in the case of an *and* gate. But even in this case, the *value* of the shares is immaterial - it is only the timing information that matters.

Specifically, take the protocol $\mathsf{Adv}(C,K)$ in the case of an *and* gate. For any party n with $n \leqslant N$ we can write the channels

- $\mathsf{OTChcRcvd}_0(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N$
- $\mathsf{OTChcRcvd}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := y_{N+1} \leftarrow \mathsf{Share}(N+1,l);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N$

equivalently as follows:

- $\mathsf{OTChcRcvd}_0(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(N+1,k)$ for $n \leqslant N$
- $\mathsf{OTChcRcvd}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(N+1,l)$ for $n \leqslant N$

We thus have the following for channels $\mathsf{OTChcRcvd}_0(-, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}}$ and $\mathsf{OTChcRcvd}_1(-, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OTChcRcvd}_0(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(N+1,k) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_0(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(N+1,l) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_1(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$

So the inductive part of the real protocol now has $\mathsf{Adv}(C,K)$ looking like so:

- $\mathsf{Adv}(\epsilon, 0)$ is the protocol $\mathsf{0}$

- $\mathsf{Adv}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{Adv}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Adv}\big(C;\ \mathit{xor\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol

  – $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

  – $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

  – $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Adv}\big(C;\ \mathit{and\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol

  – $\begin{cases} \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

  – $\begin{cases} \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

– $\begin{cases} \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

– $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

– $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

– $\begin{cases} \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

- $\begin{cases} \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,l) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := x_m \leftarrow \mathsf{Share}(m,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(N+1,k) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := y_m \leftarrow \mathsf{Share}(m,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_1(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(N+1,l) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_1(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$
- $\begin{cases} \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(m,n,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$

We now amend the shares of party $N+1$ with a gratuitous dependency on timing, which we will later convert into a dependency on the sum of shares of parties $0,\dots,N$. Specifically, in the presence of the channels $\mathsf{InShare}\text{-}\checkmark(-,-,-)$ and $\mathsf{Share}\text{-}\checkmark(-,-)$ we can express the protocol $\mathsf{Shares}(C,K)$ equivalently as follows:

- $\mathsf{Shares}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(p,i),K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{InShare}(n,p,i)$ for $n \leqslant N$
  - $\mathsf{Share}(N+1,K) := \_ \leftarrow \mathsf{InShare}\text{-}\checkmark(N+1,p,i);\ \mathsf{read}\ \mathsf{InShare}(N+1,p,i)$

- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{Share}(n,k)$ for $n \leqslant N$
  - $\mathsf{Share}(N+1,K) \coloneqq \_ \leftarrow \mathsf{Share}\text{-}\checkmark(N+1,k);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ \neg x_{N+1}$
- $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n \oplus y_n$ for $n \leqslant N$
  - $\mathsf{Share}(N+1,K) \coloneqq \_ \leftarrow \mathsf{Share}\text{-}\checkmark(N+1,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(N+1,l);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);$ $y_{N+1} \leftarrow \mathsf{Share}(N+1,l);\ \mathsf{ret}\ x_{N+1} \oplus y_{N+1}$
- $\mathsf{Shares}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit}(n,m,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{samp\ flip} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);$ $x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}(n,n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n * y_n \\ \quad \text{for } n \leqslant N+1 \end{cases}$
  - $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,0,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,0,K) \\ \quad \text{for } n \leqslant N+1 \\ \mathsf{Ctrb}\text{-}\Sigma(n,m+1,K) \coloneqq b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,K);\ b \leftarrow \mathsf{Ctrb}(n,m+1,K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \end{cases}$
  - $\mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,N+1,K)$ for $n \leqslant N$
  - $\mathsf{Share}(N+1,K) \coloneqq \_ \leftarrow \mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(N+1,N+1,K);\ \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(N+1,N+1,K)$

Clearly, we can make the timing of shares independent of their value; this was the point of introducing the timing channels in the first place. For input shares, the timing only depends on the timing of the corresponding input, so we introduce new internal channels

- $\mathsf{In}\text{-}\checkmark(n,i) \coloneqq \_ \leftarrow \mathsf{read}\ \mathsf{In}(n,i);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N+1$ and $i < I_n$

to keep track of whether an input has arrived. For a wire share, the timing depends on the timing of every input that recursively feeds into the wire. We can easily compute this in a new protocol $\mathsf{Wires}\text{-}\checkmark(C,K)$:

- $\mathsf{Wires}\text{-}\checkmark(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \mathit{input\text{-}gate}(p,i), K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{Wire}\text{-}\checkmark(K) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(p,i)$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \mathit{not\text{-}gate}(k), K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{Wire}\text{-}\checkmark(K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \mathsf{ret}\ \checkmark$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \mathit{xor\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{Wire}\text{-}\checkmark(K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(l);\ \mathsf{ret}\ \checkmark$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \mathit{and\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C, K)$ with the protocol
  - $\mathsf{Wire}\text{-}\checkmark(K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(l);\ \mathsf{ret}\ \checkmark$

Our goal now is to show that the timing channels can be equivalently characterized as follows:

- $\mathsf{InShare}\text{-}\checkmark(m,n,i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(n,i)$ for $m, n \leqslant N+1$ and $i < I_n$
- $\mathsf{Share}\text{-}\checkmark(n,k) := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$ for $n \leqslant N+1$ and $k < K$

To this end, we introduce new internal channels

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(m,n,i) := \_ \leftarrow \mathsf{read}\ \mathsf{InShare}\text{-}\$(m,n,i);\ \mathsf{ret}\ \checkmark$ for $m, n \leqslant N+1$ and $i < I_n$
- $\mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(m,n,i) := \_ \leftarrow \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i);\ \mathsf{ret}\ \checkmark$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$

to keep track of the timing information in the protocol Init and

- $\mathsf{SendBit}\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{SendBit}(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n, m \leqslant N+1$ and $k < K$
- $\mathsf{RcvdBit}\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{RcvdBit}(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n, m \leqslant N+1$ and $k < K$
- $\mathsf{Ctrb}\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{Ctrb}(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n, m \leqslant N+1$ and $k < K$
- $\mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n, m \leqslant N+1$ and $k < K$

to keep track of the timing information in the protocol $\mathsf{Shares}(C,K)$. Call the following protocol fragment $\mathsf{Init}\text{-}\checkmark$:

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(m,n,i) := \_ \leftarrow \mathsf{read}\ \mathsf{InShare}\text{-}\$(m,n,i);\ \mathsf{ret}\ \checkmark$ for $m, n \leqslant N+1$ and $i < I_n$
- $\mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(m,n,i) := \_ \leftarrow \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i);\ \mathsf{ret}\ \checkmark$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$
- $\mathsf{InShare}\text{-}\checkmark(m,n,i) := \_ \leftarrow \mathsf{read}\ \mathsf{InShare}(m,n,i);\ \mathsf{ret}\ \checkmark$ for $m, n \leqslant N+1$ and $i < I_n$

Call the following protocol fragment $\mathsf{Shares}\text{-}\checkmark(C,K)$:

- $\mathsf{SendBit}\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{SendBit}(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n, m \leqslant N+1$ and $k < K$
- $\mathsf{RcvdBit}\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{RcvdBit}(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n, m \leqslant N+1$ and $k < K$
- $\mathsf{Ctrb}\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{Ctrb}(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n, m \leqslant N+1$ and $k < K$
- $\mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n, m \leqslant N+1$ and $k < K$
- $\mathsf{Share}\text{-}\checkmark(n,k) := \_ \leftarrow \mathsf{read}\ \mathsf{Share}(n,k);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N+1$ and $k < K$

In the presence of the channels $\mathsf{InShare}\text{-}\checkmark(-,-,-)$ as well as the protocol $\mathsf{Shares}(C,K)$ we can define the protocol $\mathsf{Shares}\text{-}\checkmark(C,K)$ equivalently as follows:

- $\mathsf{Shares}\text{-}\checkmark(\epsilon, 0)$ is the protocol $\mathsf{0}$

- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}\text{-}\checkmark(n,K) := \mathsf{read}\ \mathsf{InShare}\text{-}\checkmark(n,p,i)$ for $n \leqslant N+1$
- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}\text{-}\checkmark(n,K) := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(n,k)$ for $n \leqslant N$
  - $\mathsf{Share}\text{-}\checkmark(N+1,K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(N+1,k);\ \mathsf{ret}\ \checkmark$
- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}\text{-}\checkmark(n,K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,l);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N+1$
- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C,K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit}\text{-}\checkmark(n,m,K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
  - $\mathsf{RcvdBit}\text{-}\checkmark(n,m,K) := \_ \leftarrow \mathsf{SendBit}\text{-}\checkmark(m,n,K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(m,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(m,l);$ $\_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,l);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Ctrb}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}\text{-}\checkmark(n,n,K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \end{cases}$
  - $\mathsf{Share}\text{-}\checkmark(n,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,N+1,K)$ for $n \leqslant N+1$

We also observe that in the presence of the protocol Init we can define the protocol $\mathsf{Init}\text{-}\checkmark$ equivalently as follows:

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(m,n,i) := \_ \leftarrow \mathsf{In}\text{-}\checkmark(n,i);\ \mathsf{ret}\ \checkmark$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$
- $\begin{cases} \mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(0,n,i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\checkmark(0,n,i) \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \\ \mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(m+1,n,i) := \_ \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(m,n,i);\ \_ \leftarrow \mathsf{InShare}\text{-}\$\text{-}\checkmark(m+1,n,i);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } m < N \text{ and } n \leqslant N+1 \text{ and } i < I_n \end{cases}$

- $\mathsf{InShare\text{-}\$\text{-}\checkmark}(N+1,n,i) := \_ \leftarrow \mathsf{InShare\text{-}\$\text{-}\Sigma\text{-}\checkmark}(N,n,i);\ \_ \leftarrow \mathsf{In\text{-}\checkmark}(n,i);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N+1$ and $i < I_n$
- $\mathsf{InShare\text{-}\checkmark}(m,n,i) := \mathsf{read}\ \mathsf{InShare\text{-}\$\text{-}\checkmark}(m,n,i)$ for $m,n \leqslant N+1$ and $i < I_n$

Furthermore, in the presence of the channels $\mathsf{In\text{-}\checkmark}(-,-)$ we can express the protocol $\mathsf{Init\text{-}\checkmark}$ equivalently as follows:

- $\mathsf{InShare\text{-}\$\text{-}\checkmark}(m,n,i) := \mathsf{read}\ \mathsf{In\text{-}\checkmark}(n,i)$ for $m,n \leqslant N+1$ and $i < I_n$
- $\mathsf{InShare\text{-}\$\text{-}\Sigma\text{-}\checkmark}(m,n,i) := \mathsf{read}\ \mathsf{In\text{-}\checkmark}(n,i)$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$
- $\mathsf{InShare\text{-}\checkmark}(m,n,i) := \mathsf{read}\ \mathsf{In\text{-}\checkmark}(n,i)$ for $m,n \leqslant N+1$ and $i < I_n$

Lastly, in the presence of the channels $\mathsf{InShare\text{-}\checkmark}(-,-,-)$ as well as the protocol $\mathsf{Wires\text{-}\checkmark}(C,K)$ we can express the protocol $\mathsf{Shares\text{-}\checkmark}(C,K)$ equivalently as the channels

- $\mathsf{Share\text{-}\checkmark}(n,k) := \mathsf{read}\ \mathsf{Wire\text{-}\checkmark}(k)$ for $n \leqslant N+1$ and $k < K$

together with the following protocol $\mathsf{Ctrbs\text{-}\checkmark}(C,K)$:

- $\mathsf{Ctrbs\text{-}\checkmark}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Ctrbs\text{-}\checkmark}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Ctrbs\text{-}\checkmark}(C,K)$ with the protocol
  - $\mathsf{SendBit\text{-}\checkmark}(n,m,K) := \mathsf{read}\ \mathsf{SendBit\text{-}\checkmark}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit\text{-}\checkmark}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit\text{-}\checkmark}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb\text{-}\checkmark}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb\text{-}\checkmark}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb\text{-}\Sigma\text{-}\checkmark}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb\text{-}\Sigma\text{-}\checkmark}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{Ctrbs\text{-}\checkmark}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Ctrbs\text{-}\checkmark}(C,K)$ with the protocol
  - $\mathsf{SendBit\text{-}\checkmark}(n,m,K) := \mathsf{read}\ \mathsf{SendBit\text{-}\checkmark}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit\text{-}\checkmark}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit\text{-}\checkmark}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb\text{-}\checkmark}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb\text{-}\checkmark}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb\text{-}\Sigma\text{-}\checkmark}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb\text{-}\Sigma\text{-}\checkmark}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{Ctrbs\text{-}\checkmark}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Ctrbs\text{-}\checkmark}(C,K)$ with the protocol
  - $\mathsf{SendBit\text{-}\checkmark}(n,m,K) := \mathsf{read}\ \mathsf{SendBit\text{-}\checkmark}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit\text{-}\checkmark}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit\text{-}\checkmark}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb\text{-}\checkmark}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb\text{-}\checkmark}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb\text{-}\Sigma\text{-}\checkmark}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb\text{-}\Sigma\text{-}\checkmark}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{Ctrbs\text{-}\checkmark}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Ctrbs\text{-}\checkmark}(C,K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit\text{-}\checkmark}(n,m,K) := \_ \leftarrow \mathsf{Wire\text{-}\checkmark}(k);\ \_ \leftarrow \mathsf{Wire\text{-}\checkmark}(l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit\text{-}\checkmark}(n,m,K) := \mathsf{read}\ \mathsf{SendBit\text{-}\checkmark}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
  - $\begin{cases} \mathsf{RcvdBit\text{-}\checkmark}(n,m,K) := \_ \leftarrow \mathsf{Wire\text{-}\checkmark}(k);\ \_ \leftarrow \mathsf{Wire\text{-}\checkmark}(l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \mathsf{RcvdBit\text{-}\checkmark}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit\text{-}\checkmark}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \geqslant n \end{cases}$
  - $\mathsf{Ctrb\text{-}\checkmark}(n,m,K) := \_ \leftarrow \mathsf{Wire\text{-}\checkmark}(k);\ \_ \leftarrow \mathsf{Wire\text{-}\checkmark}(l);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb\text{-}\Sigma\text{-}\checkmark}(n,m,K) := \_ \leftarrow \mathsf{Wire\text{-}\checkmark}(k);\ \_ \leftarrow \mathsf{Wire\text{-}\checkmark}(l);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$

We now revisit the case of an *and* gate in the protocol $\mathsf{Adv}(C,K)$. For any party n with $n \leqslant N$ we can write the channels

- $\mathsf{OTChcRcvd}_0(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(N+1, k)$
- $\mathsf{OTChcRcvd}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(N+1, l)$

equivalently as follows:

- $\mathsf{OTChcRcvd}_0(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$
- $\mathsf{OTChcRcvd}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(l)$

We thus have the following for channels $\mathsf{OTChcRcvd}_0(-, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}}$ and $\mathsf{OTChcRcvd}_1(-, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OTChcRcvd}_0(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_0(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(l) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_1(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$

So the inductive part of the real protocol now has $\mathsf{Adv}(C,K)$ looking like so:

- $\mathsf{Adv}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Adv}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$

- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Adv}\big(C;\ \mathit{not\text{-}gate}(k), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Adv}\big(C;\ \mathit{xor\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(m,n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(m,n,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $m,n \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Adv}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$
  - $\begin{cases} \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$
  - $\begin{cases} \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

- $$\begin{cases} \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$$

- $$\begin{cases} \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$$

- $$\begin{cases} \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$$

- $$\begin{cases} \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$$

- $$\begin{cases} \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$$

- $$\begin{cases} \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$$

- $$\begin{cases} \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,l) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$$

- $$\begin{cases} \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := x_m \leftarrow \mathsf{Share}(m,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$$

- $$\begin{cases} \mathsf{OTChcRcvd}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$$

- $$\begin{cases} \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := y_m \leftarrow \mathsf{Share}(m,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$$

- $$\begin{cases} \mathsf{OTChcRcvd}_1(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(l) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_1(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$$

- $\begin{cases} \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(m,n,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$

Finally, in the presence of the channels $\mathsf{InShare}\text{-}\checkmark(-,-,-)$, $\mathsf{Share}\text{-}\checkmark(-,-)$ and the protocols $\mathsf{Shares}\text{-}\checkmark$ and $\mathsf{Wires}\text{-}\checkmark$ we can express the protocol $\mathsf{Shares}(C,K)$ equivalently as follows:

- $\mathsf{Shares}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{InShare}(n,p,i)$ for $n \leqslant N$
  - $\mathsf{Share}(N+1,K) \coloneqq \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(K);\ \mathsf{read}\ \mathsf{InShare}(N+1,p,i)$
- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{Share}(n,k)$ for $n \leqslant N$
  - $\mathsf{Share}(N+1,K) \coloneqq \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ \neg x_{N+1}$
- $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n \oplus y_n$ for $n \leqslant N$
  - $\mathsf{Share}(N+1,K) \coloneqq \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ y_{N+1} \leftarrow \mathsf{Share}(N+1,l);\ \mathsf{ret}\ x_{N+1} \oplus y_{N+1}$
- $\mathsf{Shares}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit}(n,m,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{samp\ flip} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);$ $x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}(n,n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n * y_n \\ \quad \text{for } n \leqslant N+1 \end{cases}$

$$- \begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,0,K) := \mathsf{read}\ \mathsf{Ctrb}(n,0,K) \\ \quad \text{for } n \leqslant N+1 \\ \mathsf{Ctrb}\text{-}\Sigma(n,m+1,K) := b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,K);\ b \leftarrow \mathsf{Ctrb}(n,m+1,K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \end{cases}$$

- $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,N+1,K)$ for $n \leqslant N$
- $\mathsf{Share}(N+1,K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(K);\ \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(N+1,N+1,K)$

10.4.5 Timing of Shares II

We now revisit the real protocol in the form we had at the beginning of Section 10.4.4. We again start by introducing new internal channels

- $$\begin{cases} \mathsf{Share}\text{-}\Sigma(0,k) := \mathsf{read}\ \mathsf{Share}(0,k) \\ \quad \text{for } k < K \\ \mathsf{Share}\text{-}\Sigma(m+1,k) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,k);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \text{ and } k < K \end{cases}$$

that inductively compute the the sum of shares for parties $0,\dots,N$ on each wire $k < K$.

We first consider the protocol $\mathsf{Adv}(C,K)$ in the case of an *and* gate. For any party n with $n \leqslant N$ we want to write the channels

- $\mathsf{OTChcRcvd}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N$
- $\mathsf{OTChcRcvd}_1(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := y_{N+1} \leftarrow \mathsf{Share}(N+1,l);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N$

in the form below:

- $\mathsf{OTChcRcvd}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N$
- $\mathsf{OTChcRcvd}_1(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,l);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N$

We thus have the following for channels $\mathsf{OTChcRcvd}_0(-,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}}$ and $\mathsf{OTChcRcvd}_1(-,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}}$:

- $$\begin{cases} \mathsf{OTChcRcvd}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$$
- $$\begin{cases} \mathsf{OTChcRcvd}_1(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_1(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$$

So the inductive part of the real protocol now has $\mathsf{Adv}(C,K)$ looking like so:

- $\mathsf{Adv}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Adv}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Adv}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Adv}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{Adv}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

– $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

– $\begin{cases} \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n<m \\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n<m \\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n<m \\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n<m \\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n<m \\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n<m \\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n<m \\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n<m \\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n<m \\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,l) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := x_m \leftarrow \mathsf{Share}(m,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChcRcvd}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$

– $\begin{cases} \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := y_m \leftarrow \mathsf{Share}(m,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChcRcvd}_1(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_1(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$

– $\begin{cases} \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(m,n,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$

We now amend the shares of party $N+1$ with a gratuitous dependency on the sum of shares of parties $0,\dots,N$. Specifically, we modify the protocol $\mathsf{Shares}(C,K)$ as follows:

- $\mathsf{Shares}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{InShare}(n,p,i)$ for $n \leqslant N$
  - $\mathsf{Share}(N+1,K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ \mathsf{read}\ \mathsf{InShare}(N+1,p,i)$
- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{Share}(n,k)$
  - $\mathsf{Share}(N+1,K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ \neg x_{N+1}$
- $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol

- $\mathsf{SendBit}(n, m, K) := \mathsf{read}\ \mathsf{SendBit}(n, m, K)$ for $n, m \leqslant N+1$
- $\mathsf{RcvdBit}(n, m, K) := \mathsf{read}\ \mathsf{RcvdBit}(n, m, K)$ for $n, m \leqslant N+1$
- $\mathsf{Ctrb}(n, m, K) := \mathsf{read}\ \mathsf{Ctrb}(n, m, K)$ for $n, m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(n, m, K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n, m, K)$ for $n, m \leqslant N+1$
- $\mathsf{Share}(n, K) := x_n \leftarrow \mathsf{Share}(n, k);\ y_n \leftarrow \mathsf{Share}(n, l);\ \mathsf{ret}\ x_n \oplus y_n$ for $n \leqslant N$
- $\mathsf{Share}(N+1, K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1, k);$ $y_{N+1} \leftarrow \mathsf{Share}(N+1, l);\ \mathsf{ret}\ x_{N+1} \oplus y_{N+1}$

- $\mathsf{Shares}\big(C;\ \textit{and-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol

  - $\begin{cases} \mathsf{SendBit}(n, m, K) := x_n \leftarrow \mathsf{Share}(n, k);\ y_n \leftarrow \mathsf{Share}(n, l);\ \mathsf{samp\ flip} \\ \quad \text{for } n, m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n, m, K) := \mathsf{read}\ \mathsf{SendBit}(n, m, K) \\ \quad \text{for } n, m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
  - $\mathsf{RcvdBit}(n, m, K) := b \leftarrow \mathsf{SendBit}(m, n, K);\ x_m \leftarrow \mathsf{Share}(m, k);\ y_m \leftarrow \mathsf{Share}(m, l);$ $x_n \leftarrow \mathsf{Share}(n, k);\ y_n \leftarrow \mathsf{Share}(n, l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n, m \leqslant N+1$
  - $\begin{cases} \mathsf{Ctrb}(n, m, K) := \mathsf{read}\ \mathsf{SendBit}(n, m, K) \\ \quad \text{for } n, m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}(n, m, K) := \mathsf{read}\ \mathsf{RcvdBit}(n, m, K) \\ \quad \text{for } n, m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}(n, n, K) := x_n \leftarrow \mathsf{Share}(n, k);\ y_n \leftarrow \mathsf{Share}(n, l);\ \mathsf{ret}\ x_n * y_n \\ \quad \text{for } n \leqslant N+1 \end{cases}$
  - $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n, 0, K) := \mathsf{read}\ \mathsf{Ctrb}(n, 0, K) \\ \quad \text{for } n \leqslant N+1 \\ \mathsf{Ctrb}\text{-}\Sigma(n, m+1, K) := b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n, m, K);\ b \leftarrow \mathsf{Ctrb}(n, m+1, K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \end{cases}$
  - $\mathsf{Share}(n, K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n, N+1, K)$ for $n \leqslant N$
  - $\mathsf{Share}(N+1, K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, K);\ \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(N+1, N+1, K)$

It is not at all clear that the aforementioned amendments of $\mathsf{Adv}(C, K)$ and $\mathsf{Shares}(C, K)$ are sound. In the rest of this section we justify their soundness. We start by introducing the channels

- $\mathsf{InShare}\text{-}\checkmark(m, n, i) := \_ \leftarrow \mathsf{read}\ \mathsf{InShare}(m, n, i);\ \mathsf{ret}\ \checkmark$ for $m, n \leqslant N+1$ and $i < I_n$

for the timing of input shares in the initial part of the protocol and

- $\mathsf{Share}\text{-}\checkmark(n, k) := \_ \leftarrow \mathsf{read}\ \mathsf{Share}(n, k);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N+1$ and $k < K$

for the timing of shares in the inductive part of the protocol. In addition, we introduce the channels

- $\mathsf{Share}\text{-}\Sigma\text{-}\checkmark(m, k) := \_ \leftarrow \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(m, k);\ \mathsf{ret}\ \checkmark$ for $m \leqslant N$ and $k < K$

for the timing of the sum of shares.

Take the protocol $\mathsf{Adv}(C, K)$ in the case of an *and* gate. For any party n with $n \leqslant N$ we can write the channels

- $\mathsf{OTChcRcvd}_0(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N$
- $\mathsf{OTChcRcvd}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, l);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N$

equivalently as follows:

- $\mathsf{OTChcRcvd}_0(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N, k)$ for $n \leqslant N$

- $\mathsf{OTChcRcvd}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} :=$ read $\mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N, l)$ for $n \leqslant N$

We thus have the following for channels $\mathsf{OTChcRcvd}_0(-, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}}$ and $\mathsf{OTChcRcvd}_1(-, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OTChcRcvd}_0(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \text{read } \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N, k) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_0(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \text{read } \mathsf{OTChcRcvd}_0(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \text{read } \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N, l) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_1(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \text{read } \mathsf{OTChcRcvd}_1(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$

So the inductive part of the real protocol now has $\mathsf{Adv}(C, K)$ looking like so:

- $\mathsf{Adv}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Adv}\big(C;\, \textit{input-gate}(p, i), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} :=$ read $\mathsf{SendBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{RcvdBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} :=$ read $\mathsf{RcvdBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} :=$ read $\mathsf{Ctrb}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} :=$ read $\mathsf{Ctrb}\text{-}\Sigma(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n, K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \text{read } \mathsf{Share}(n, K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n, K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \text{read } \mathsf{Share}(n, K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} :=$ read $\mathsf{OTMsg}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} :=$ read $\mathsf{OTMsg}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} :=$ read $\mathsf{OTMsg}_2(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} :=$ read $\mathsf{OTMsg}_3(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} :=$ read $\mathsf{OTMsgRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} :=$ read $\mathsf{OTMsgRcvd}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_2(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} :=$ read $\mathsf{OTMsgRcvd}_2(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_3(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} :=$ read $\mathsf{OTMsgRcvd}_3(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTChc}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} :=$ read $\mathsf{OTChc}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTChc}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} :=$ read $\mathsf{OTChc}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} :=$ read $\mathsf{OTChcRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} :=$ read $\mathsf{OTChcRcvd}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} :=$ read $\mathsf{OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
- $\mathsf{Adv}\big(C;\, \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} :=$ read $\mathsf{SendBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{RcvdBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} :=$ read $\mathsf{RcvdBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} :=$ read $\mathsf{Ctrb}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} :=$ read $\mathsf{Ctrb}\text{-}\Sigma(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$

- $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

• $\mathsf{Adv}\big(C;\ \mathit{xor\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol

- $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Adv}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol

  - $$\begin{cases} \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

  - $$\begin{cases} \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

  - $$\begin{cases} \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

  - $$\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

  - $$\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

  - $$\begin{cases} \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$$

  - $$\begin{cases} \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$$

  - $$\begin{cases} \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$$

  - $$\begin{cases} \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$$

  - $$\begin{cases} \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$$

– $\begin{cases} \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,l) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := x_m \leftarrow \mathsf{Share}(m,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChcRcvd}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,k) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$

– $\begin{cases} \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := y_m \leftarrow \mathsf{Share}(m,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChcRcvd}_1(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,l) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_1(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$

– $\begin{cases} \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(m,n,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$

We now amend the shares of party $N+1$ further: in the presence of the channels $\mathsf{Share}\text{-}\Sigma\text{-}\checkmark(-,-)$ we can turn the gratuitous dependency on the sum of shares of parties $0,\dots,N$ into a dependency on the corresponding timing channel. Specifically, we can express the protocol $\mathsf{Shares}(C,K)$ equivalently as follows:

- $\mathsf{Shares}(\epsilon,0)$ is the protocol $\mathsf{0}$

- $\mathsf{Shares}\big(C;\ \mathit{input\text{-}gate}(p,i), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{InShare}(n,p,i)$ for $n \leqslant N$
  - $\mathsf{Share}(N+1,K) \coloneqq \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,K);\ \mathsf{read}\ \mathsf{InShare}(N+1,p,i)$
- $\mathsf{Shares}\big(C;\ \mathit{not\text{-}gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{Share}(n,k)$ for $n \leqslant N$
  - $\mathsf{Share}(N+1,K) \coloneqq \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ \neg x_{N+1}$
- $\mathsf{Shares}\big(C;\ \mathit{xor\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n \oplus y_n$ for $n \leqslant N$
  - $\mathsf{Share}(N+1,K) \coloneqq \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);$ $y_{N+1} \leftarrow \mathsf{Share}(N+1,l);\ \mathsf{ret}\ x_{N+1} \oplus y_{N+1}$
- $\mathsf{Shares}\big(C;\ \mathit{and\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit}(n,m,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{samp\ flip} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);$ $x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}(n,n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n * y_n \\ \quad \text{for } n \leqslant N+1 \end{cases}$
  - $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,0,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,0,K) \\ \quad \text{for } n \leqslant N+1 \\ \mathsf{Ctrb}\text{-}\Sigma(n,m+1,K) \coloneqq b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,K);\ b \leftarrow \mathsf{Ctrb}(n,m+1,K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \end{cases}$
  - $\mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,N+1,K)$ for $n \leqslant N$

– $\mathsf{Share}(N+1,K) := \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,K);\ \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(N+1,N+1,K)$

We now carry out largely the same steps as before to make the timing of shares independent of their value. To this end, we introduce new internal channels

- $\mathsf{In}\text{-}\checkmark(n,i) := \_ \leftarrow \mathsf{read}\ \mathsf{In}(n,i);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N+1$ and $i < I_n$

to keep track of whether an input has arrived, and a new protocol $\mathsf{Wires}\text{-}\checkmark(C,K)$ that keeps track of the timing of wire shares:

- $\mathsf{Wires}\text{-}\checkmark(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \textit{input-gate}(p,i),K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C,K)$ with the protocol
  – $\mathsf{Wire}\text{-}\checkmark(K) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(p,i)$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \textit{not-gate}(k),K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C,K)$ with the protocol
  – $\mathsf{Wire}\text{-}\checkmark(K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \mathsf{ret}\ \checkmark$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \textit{xor-gate}(k,l),K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C,K)$ with the protocol
  – $\mathsf{Wire}\text{-}\checkmark(K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(l);\ \mathsf{ret}\ \checkmark$
- $\mathsf{Wires}\text{-}\checkmark\big(C;\ \textit{and-gate}(k,l),K+1\big)$ is the composition of $\mathsf{Wires}\text{-}\checkmark(C,K)$ with the protocol
  – $\mathsf{Wire}\text{-}\checkmark(K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(l);\ \mathsf{ret}\ \checkmark$

Our goal is again to show that the timing channels can be equivalently characterized as follows:

- $\mathsf{InShare}\text{-}\checkmark(m,n,i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(n,i)$ for $m,n \leqslant N+1$ and $i < I_n$
- $\mathsf{Share}\text{-}\checkmark(n,k) := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$ for $n \leqslant N+1$ and $k < K$

To this end, we introduce new internal channels

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(m,n,i) := \_ \leftarrow \mathsf{read}\ \mathsf{InShare}\text{-}\$(m,n,i);\ \mathsf{ret}\ \checkmark$ for $m,n \leqslant N+1$ and $i < I_n$
- $\mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(m,n,i) := \_ \leftarrow \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i);\ \mathsf{ret}\ \checkmark$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$

to keep track of the timing information in the protocol Init and

- $\mathsf{SendBit}\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{SendBit}(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$ and $k < K$
- $\mathsf{RcvdBit}\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{RcvdBit}(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$ and $k < K$
- $\mathsf{Ctrb}\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{Ctrb}(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$ and $k < K$
- $\mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$ and $k < K$

to keep track of the timing information in the protocol $\mathsf{Shares}(C,K)$. Call the following protocol fragment $\mathsf{Init}\text{-}\checkmark$:

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(m,n,i) := \_ \leftarrow \mathsf{read}\ \mathsf{InShare}\text{-}\$(m,n,i);\ \mathsf{ret}\ \checkmark$ for $m,n \leqslant N+1$ and $i < I_n$
- $\mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(m,n,i) := \_ \leftarrow \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i);\ \mathsf{ret}\ \checkmark$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$
- $\mathsf{InShare}\text{-}\checkmark(m,n,i) := \_ \leftarrow \mathsf{read}\ \mathsf{InShare}(m,n,i);\ \mathsf{ret}\ \checkmark$ for $m,n \leqslant N+1$ and $i < I_n$

Call the following protocol fragment $\mathsf{Shares}\text{-}\checkmark(C,K)$:

- $\mathsf{SendBit}\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{SendBit}(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$ and $k < K$
- $\mathsf{RcvdBit}\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{RcvdBit}(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$ and $k < K$
- $\mathsf{Ctrb}\text{-}\checkmark(n,m,k) := \_ \leftarrow \mathsf{Ctrb}(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$ and $k < K$

- $\mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,k) \coloneqq \_ \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,k);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$ and $k < K$
- $\mathsf{Share}\text{-}\checkmark(n,k) \coloneqq \_ \leftarrow \mathsf{read\ Share}(n,k);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N+1$ and $k < K$

First we observe that in the presence of the channels $\mathsf{Share}\text{-}\Sigma(-,-)$ and $\mathsf{Share}\text{-}\checkmark(-,-)$ we can express the channels

- $\mathsf{Share}\text{-}\Sigma\text{-}\checkmark(m,k) \coloneqq \_ \leftarrow \mathsf{read\ Share}\text{-}\Sigma(m,k);\ \mathsf{ret}\ \checkmark$ for $m \leqslant N$ and $k < K$

equivalently as follows:

- $\begin{cases} \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(0,k) \coloneqq \mathsf{read\ Share}\text{-}\checkmark(0,k) \\ \quad \text{for } k < K \\ \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(m+1,k) \coloneqq \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(m,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(m+1,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } m < N \text{ and } k < K \end{cases}$

We further observe that in the presence of the channels $\mathsf{InShare}\text{-}\checkmark(-,-,-)$ and the protocol $\mathsf{Shares}(C,K)$ we can express the protocol $\mathsf{Shares}\text{-}\checkmark(C,K)$ equivalently as follows:

- $\mathsf{Shares}\text{-}\checkmark(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(n,m,K) \coloneqq \mathsf{read\ SendBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}\text{-}\checkmark(n,m,K) \coloneqq \mathsf{read\ RcvdBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\checkmark(n,m,K) \coloneqq \mathsf{read\ Ctrb}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K) \coloneqq \mathsf{read\ Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}\text{-}\checkmark(n,K) \coloneqq \mathsf{read\ InShare}\text{-}\checkmark(n,p,i)$ for $n \leqslant N$
  - $\mathsf{Share}\text{-}\checkmark(N+1,K) \coloneqq \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,K);\ \mathsf{read\ InShare}\text{-}\checkmark(N+1,p,i)$
- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(n,m,K) \coloneqq \mathsf{read\ SendBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}\text{-}\checkmark(n,m,K) \coloneqq \mathsf{read\ RcvdBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\checkmark(n,m,K) \coloneqq \mathsf{read\ Ctrb}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K) \coloneqq \mathsf{read\ Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}\text{-}\checkmark(n,K) \coloneqq \mathsf{read\ Share}\text{-}\checkmark(n,k)$ for $n \leqslant N$
  - $\mathsf{Share}\text{-}\checkmark(N+1,K) \coloneqq \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(N+1,k);\ \mathsf{ret}\ \checkmark$
- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(n,m,K) \coloneqq \mathsf{read\ SendBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}\text{-}\checkmark(n,m,K) \coloneqq \mathsf{read\ RcvdBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\checkmark(n,m,K) \coloneqq \mathsf{read\ Ctrb}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K) \coloneqq \mathsf{read\ Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}\text{-}\checkmark(n,K) \coloneqq \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,l);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N$
  - $\mathsf{Share}\text{-}\checkmark(N+1,K) \coloneqq \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(N+1,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(N+1,l);\ \mathsf{ret}\ \checkmark$
- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C,K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit}\text{-}\checkmark(n,m,K) \coloneqq \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}\text{-}\checkmark(n,m,K) \coloneqq \mathsf{read\ SendBit}\text{-}\checkmark(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$

- $\mathsf{RcvdBit}\text{-}\checkmark(n,m,K) := \_ \leftarrow \mathsf{SendBit}\text{-}\checkmark(m,n,K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(m,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(m,l);$ $\_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,l);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$
- $\begin{cases} \mathsf{Ctrb}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}\text{-}\checkmark(n,n,K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \end{cases}$
- $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,0,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\checkmark(n,0,K) \\ \quad \text{for } n \leqslant N+1 \\ \mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m+1,K) := \_ \leftarrow \mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K);\ \_ \leftarrow \mathsf{Ctrb}\text{-}\checkmark(n,m+1,K);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \end{cases}$
- $\mathsf{Share}\text{-}\checkmark(n,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,N+1,K)$ for $n \leqslant N$
- $\mathsf{Share}\text{-}\checkmark(N+1,K) := \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,K);\ \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(N+1,N+1,K)$

We also observe that in the presence of the protocol Init we can define the protocol $\mathsf{Init}\text{-}\checkmark$ equivalently as follows:

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(m,n,i) := \_ \leftarrow \mathsf{In}\text{-}\checkmark(n,i);\ \mathsf{ret}\ \checkmark$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$
- $\begin{cases} \mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(0,n,i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\checkmark(0,n,i) \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \\ \mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(m+1,n,i) := \_ \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(m,n,i);\ \_ \leftarrow \mathsf{InShare}\text{-}\$\text{-}\checkmark(m+1,n,i);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } m < N \text{ and } n \leqslant N+1 \text{ and } i < I_n \end{cases}$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(N+1,n,i) := \_ \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(N,n,i);\ \_ \leftarrow \mathsf{In}\text{-}\checkmark(n,i);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N+1$ and $i < I_n$
- $\mathsf{InShare}\text{-}\checkmark(m,n,i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\checkmark(m,n,i)$ for $m,n \leqslant N+1$ and $i < I_n$

Furthermore, in the presence of the channels $\mathsf{In}\text{-}\checkmark(-,-)$ we can express the protocol $\mathsf{Init}\text{-}\checkmark$ equivalently as follows:

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(m,n,i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(n,i)$ for $m,n \leqslant N+1$ and $i < I_n$
- $\mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(m,n,i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(n,i)$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$
- $\mathsf{InShare}\text{-}\checkmark(m,n,i) := \mathsf{read}\ \mathsf{In}\text{-}\checkmark(n,i)$ for $m,n \leqslant N+1$ and $i < I_n$

We can now merge the channels

- $\begin{cases} \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(0,k) := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(0,k) \\ \quad \text{for } k < K \\ \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(m+1,k) := \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(m,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(m+1,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } m < N \text{ and } k < K \end{cases}$

with the protocol $\mathsf{Shares}\text{-}\checkmark(C,K)$ to obtain the following new form of the protocol $\mathsf{Shares}\text{-}\checkmark(C,K)$:

- $\mathsf{Shares}\text{-}\checkmark(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$

- $\mathsf{Share}\text{-}\checkmark(n,K) := \mathsf{read}\ \mathsf{InShare}\text{-}\checkmark(n,p,i)$ for $n \leqslant N$
- $\begin{cases} \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(0,k) := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(0,k) \\ \quad \text{for } k < K \\ \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(m+1,k) := \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(m,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(m+1,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } m < N \text{ and } k < K \end{cases}$
- $\mathsf{Share}\text{-}\checkmark(N+1,K) := \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,K);\ \mathsf{read}\ \mathsf{InShare}\text{-}\checkmark(N+1,p,i)$

- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}\text{-}\checkmark(n,K) := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(n,k)$ for $n \leqslant N$
  - $\begin{cases} \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(0,k) := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(0,k) \\ \quad \text{for } k < K \\ \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(m+1,k) := \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(m,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(m+1,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } m < N \text{ and } k < K \end{cases}$
  - $\mathsf{Share}\text{-}\checkmark(N+1,K) := \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(N+1,k);\ \mathsf{ret}\ \checkmark$

- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C,K)$ with the protocol
  - $\mathsf{SendBit}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}\text{-}\checkmark(n,K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,l);\ \mathsf{ret}\ \checkmark$ for $n \leqslant N$
  - $\begin{cases} \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(0,k) := \mathsf{read}\ \mathsf{Share}\text{-}\checkmark(0,k) \\ \quad \text{for } k < K \\ \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(m+1,k) := \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(m,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(m+1,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } m < N \text{ and } k < K \end{cases}$
  - $\mathsf{Share}\text{-}\checkmark(N+1,K) := \_ \leftarrow \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(N+1,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(N+1,l);\ \mathsf{ret}\ \checkmark$

- $\mathsf{Shares}\text{-}\checkmark\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}\text{-}\checkmark(C,K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit}\text{-}\checkmark(n,m,K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
  - $\mathsf{RcvdBit}\text{-}\checkmark(n,m,K) := \_ \leftarrow \mathsf{SendBit}\text{-}\checkmark(m,n,K);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(m,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(m,l);$ $\_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,l);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Ctrb}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{SendBit}\text{-}\checkmark(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}\text{-}\checkmark(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}\text{-}\checkmark(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}\text{-}\checkmark(n,n,K) := \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,k);\ \_ \leftarrow \mathsf{Share}\text{-}\checkmark(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \end{cases}$

- $\begin{cases} \textsf{Ctrb-}\Sigma\textsf{-}\checkmark(n,0,K) \coloneqq \textsf{read Ctrb-}\checkmark(n,0,K) \\ \quad \text{for } n \leqslant N+1 \\ \textsf{Ctrb-}\Sigma\textsf{-}\checkmark(n,m+1,K) \coloneqq \_ \leftarrow \textsf{Ctrb-}\Sigma\textsf{-}\checkmark(n,m,K);\ \_ \leftarrow \textsf{Ctrb-}\checkmark(n,m+1,K);\ \textsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \end{cases}$
- $\textsf{Share-}\checkmark(n,K) \coloneqq \textsf{read Ctrb-}\Sigma\textsf{-}\checkmark(n,N+1,K)$ for $n \leqslant N$
- $\begin{cases} \textsf{Share-}\Sigma\textsf{-}\checkmark(0,k) \coloneqq \textsf{read Share-}\checkmark(0,k) \\ \quad \text{for } k < K \\ \textsf{Share-}\Sigma\textsf{-}\checkmark(m+1,k) \coloneqq \_ \leftarrow \textsf{Share-}\Sigma\textsf{-}\checkmark(m,k);\ \_ \leftarrow \textsf{Share-}\checkmark(m+1,k);\ \textsf{ret}\ \checkmark \\ \quad \text{for } m < N \text{ and } k < K \end{cases}$
- $\textsf{Share-}\checkmark(N+1,K) \coloneqq \_ \leftarrow \textsf{Share-}\Sigma\textsf{-}\checkmark(N,K);\ \textsf{read Ctrb-}\Sigma\textsf{-}\checkmark(N+1,N+1,K)$

In the presence of the channels $\textsf{InShare-}\checkmark(-,-,-)$ as well as the protocol $\textsf{Wires-}\checkmark(C,K)$ we can express the protocol $\textsf{Shares-}\checkmark(C,K)$ equivalently as the channels

- $\textsf{Share-}\checkmark(n,k) \coloneqq \textsf{read Wire-}\checkmark(k)$ for $n \leqslant N+1$ and $k < K$
- $\textsf{Share-}\Sigma\textsf{-}\checkmark(m,k) \coloneqq \textsf{read Wire-}\checkmark(k)$ for $m \leqslant N$ and $k < K$

together with the following protocol $\textsf{Ctrbs-}\checkmark(C,K)$:

- $\textsf{Ctrbs-}\checkmark(\epsilon,0)$ is the protocol $\textsf{0}$
- $\textsf{Ctrbs-}\checkmark\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\textsf{Ctrbs-}\checkmark(C,K)$ with the protocol
  - $\textsf{SendBit-}\checkmark(n,m,K) \coloneqq \textsf{read SendBit-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\textsf{RcvdBit-}\checkmark(n,m,K) \coloneqq \textsf{read RcvdBit-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\textsf{Ctrb-}\checkmark(n,m,K) \coloneqq \textsf{read Ctrb-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\textsf{Ctrb-}\Sigma\textsf{-}\checkmark(n,m,K) \coloneqq \textsf{read Ctrb-}\Sigma\textsf{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
- $\textsf{Ctrbs-}\checkmark\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\textsf{Ctrbs-}\checkmark(C,K)$ with the protocol
  - $\textsf{SendBit-}\checkmark(n,m,K) \coloneqq \textsf{read SendBit-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\textsf{RcvdBit-}\checkmark(n,m,K) \coloneqq \textsf{read RcvdBit-}\checkmark(n,m,K)$ for $m,n \leqslant N+1$
  - $\textsf{Ctrb-}\checkmark(n,m,K) \coloneqq \textsf{read Ctrb-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\textsf{Ctrb-}\Sigma\textsf{-}\checkmark(n,m,K) \coloneqq \textsf{read Ctrb-}\Sigma\textsf{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
- $\textsf{Ctrbs-}\checkmark\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\textsf{Ctrbs-}\checkmark(C,K)$ with the protocol
  - $\textsf{SendBit-}\checkmark(n,m,K) \coloneqq \textsf{read SendBit-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\textsf{RcvdBit-}\checkmark(n,m,K) \coloneqq \textsf{read RcvdBit-}\checkmark(n,m,K)$ for $m,n \leqslant N+1$
  - $\textsf{Ctrb-}\checkmark(n,m,K) \coloneqq \textsf{read Ctrb-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
  - $\textsf{Ctrb-}\Sigma\textsf{-}\checkmark(n,m,K) \coloneqq \textsf{read Ctrb-}\Sigma\textsf{-}\checkmark(n,m,K)$ for $n,m \leqslant N+1$
- $\textsf{Ctrbs-}\checkmark\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\textsf{Ctrbs-}\checkmark(C,K)$ with the protocol
  - $\begin{cases} \textsf{SendBit-}\checkmark(n,m,K) \coloneqq \_ \leftarrow \textsf{Wire-}\checkmark(k);\ \_ \leftarrow \textsf{Wire-}\checkmark(l);\ \textsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \textsf{SendBit-}\checkmark(n,m,K) \coloneqq \textsf{read SendBit-}\checkmark(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
  - $\begin{cases} \textsf{RcvdBit-}\checkmark(n,m,K) \coloneqq \_ \leftarrow \textsf{Wire-}\checkmark(k);\ \_ \leftarrow \textsf{Wire-}\checkmark(l);\ \textsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \textsf{RcvdBit-}\checkmark(n,m,K) \coloneqq \textsf{read RcvdBit-}\checkmark(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \geqslant n \end{cases}$

- $\mathsf{Ctrb}\text{-}\checkmark(n,m,K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(l);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma\text{-}\checkmark(n,m,K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(k);\ \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(l);\ \mathsf{ret}\ \checkmark$ for $n,m \leqslant N+1$

We now revisit the case of an *and* gate in the protocol $\mathsf{Adv}(C,K)$. For any party n with $n \leqslant N$ we can write the channels

- $\mathsf{OTChcRcvd}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,k)$
- $\mathsf{OTChcRcvd}_1(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma\text{-}\checkmark(N,l)$

equivalently as follows:

- $\mathsf{OTChcRcvd}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k)$
- $\mathsf{OTChcRcvd}_1(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(l)$

We thus have the following for channels $\mathsf{OTChcRcvd}_0(-,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}}$ and $\mathsf{OTChcRcvd}_1(-,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OTChcRcvd}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_1(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(l) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_1(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$

So the inductive part of the real protocol now has $\mathsf{Adv}(C,K)$ looking like so:

- $\mathsf{Adv}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Adv}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Adv}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{Adv}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

– $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Adv}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol

– $$\begin{cases} \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

– $$\begin{cases} \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

– $$\begin{cases} \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

– $$\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

– $$\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$$

– $$\begin{cases} \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$$

– $$\begin{cases} \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$$

– $\begin{cases} \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \oplus y_n \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,l) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m \text{ honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := x_m \leftarrow \mathsf{Share}(m,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChcRcvd}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}\text{-}\checkmark(k) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$

– $\begin{cases} \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := y_m \leftarrow \mathsf{Share}(m,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$

– $\begin{cases} \mathsf{OTChcRcvd}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ Wire\text{-}}\checkmark(l) \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_1(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ OTChcRcvd}_1(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$

– $\begin{cases} \mathsf{OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ RcvdBit}(m, n, K) \\ \quad \text{for } n, m \leqslant N+1 \text{ if } m \text{ semi-honest} \\ \mathsf{OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read\ OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n, m \leqslant N+1 \text{ if } m \text{ honest} \end{cases}$

Finally, in the presence of the channels $\mathsf{Share\text{-}}\Sigma\text{-}\checkmark(-,-)$ we can characterize the protocol $\mathsf{Shares}(C, K)$ equivalently as follows:

- $\mathsf{Shares}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(p, i), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(n, m, K) := \mathsf{read\ SendBit}(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{RcvdBit}(n, m, K) := \mathsf{read\ RcvdBit}(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}(n, m, K) := \mathsf{read\ Ctrb}(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb\text{-}}\Sigma(n, m, K) := \mathsf{read\ Ctrb\text{-}}\Sigma(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{Share}(n, K) := \mathsf{read\ InShare}(n, p, i)$ for $n \leqslant N$
  - $\mathsf{Share}(N+1, K) := \_ \leftarrow \mathsf{Wire\text{-}}\checkmark(K);\ \mathsf{read\ InShare}(N+1, p, i)$
- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(n, m, K) := \mathsf{read\ SendBit}(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{RcvdBit}(n, m, K) := \mathsf{read\ RcvdBit}(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}(n, m, K) := \mathsf{read\ Ctrb}(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb\text{-}}\Sigma(n, m, K) := \mathsf{read\ Ctrb\text{-}}\Sigma(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{Share}(n, K) := \mathsf{read\ Share}(n, k)$ for $n \leqslant N$
  - $\mathsf{Share}(N+1, K) := \_ \leftarrow \mathsf{Wire\text{-}}\checkmark(K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1, k);\ \mathsf{ret}\ \neg x_{N+1}$
- $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(n, m, K) := \mathsf{read\ SendBit}(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{RcvdBit}(n, m, K) := \mathsf{read\ RcvdBit}(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}(n, m, K) := \mathsf{read\ Ctrb}(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb\text{-}}\Sigma(n, m, K) := \mathsf{read\ Ctrb\text{-}}\Sigma(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{Share}(n, K) := x_n \leftarrow \mathsf{Share}(n, k);\ y_n \leftarrow \mathsf{Share}(n, l);\ \mathsf{ret}\ x_n \oplus y_n$ for $n \leqslant N$
  - $\mathsf{Share}(N+1, K) := \_ \leftarrow \mathsf{Wire\text{-}}\checkmark(K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1, k);\ y_{N+1} \leftarrow \mathsf{Share}(N+1, l);\ \mathsf{ret}\ x_{N+1} \oplus y_{N+1}$
- $\mathsf{Shares}\big(C;\ \textit{and-gate}(k, l), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit}(n, m, K) := x_n \leftarrow \mathsf{Share}(n, k);\ y_n \leftarrow \mathsf{Share}(n, l);\ \mathsf{samp\ flip} \\ \quad \text{for } n, m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n, m, K) := \mathsf{read\ SendBit}(n, m, K) \\ \quad \text{for } n, m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
  - $\mathsf{RcvdBit}(n, m, K) := b \leftarrow \mathsf{SendBit}(m, n, K);\ x_m \leftarrow \mathsf{Share}(m, k);\ y_m \leftarrow \mathsf{Share}(m, l);$ $x_n \leftarrow \mathsf{Share}(n, k);\ y_n \leftarrow \mathsf{Share}(n, l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n, m \leqslant N+1$

- $\begin{cases} \mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}(n,n,K) := x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n * y_n \\ \quad \text{for } n \leqslant N+1 \end{cases}$
- $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,0,K) := \mathsf{read}\ \mathsf{Ctrb}(n,0,K) \\ \quad \text{for } n \leqslant N+1 \\ \mathsf{Ctrb}\text{-}\Sigma(n,m+1,K) := b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,K);\ b \leftarrow \mathsf{Ctrb}(n,m+1,K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \end{cases}$
- $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,N+1,K)$ for $n \leqslant N$
- $\mathsf{Share}(N+1,K) := \_ \leftarrow \mathsf{Wire}\text{-}\checkmark(K);\ \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(N+1,N+1,K)$

At this point, the channels $\mathsf{Share}\text{-}\Sigma\text{-}\checkmark(-,-)$ are unused and we can eliminate them. The resulting real protocol is identical to the form of the real protocol we had at the end of Section 10.4.4, which justifies the amendments to $\mathsf{Adv}(C,K)$ and $\mathsf{Shares}(C,K)$ we made at the beginning of this section.

10.4.6 Sum Of Shares

We continue to simplify the real protocol after amending $\mathsf{Adv}(C,K)$ and $\mathsf{Shares}(C,K)$ as indicated in Section 10.4.5. Recall the channels

- $\begin{cases} \mathsf{Share}\text{-}\Sigma(0,k) := \mathsf{read}\ \mathsf{Share}(0,k) \\ \quad \text{for } k < K \\ \mathsf{Share}\text{-}\Sigma(m+1,k) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,k);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \text{ and } k < K \end{cases}$

that inductively compute the the sum of shares for parties $0,\ldots,N$ on each wire $k < K$. To also include the shares of party $N+1$, we add new internal channels as follows:

- $\mathsf{Share}\text{-}\Sigma(N+1,k) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$ for $k < K$

We now revisit the final part of the real protocol. If party n is semi-honest and wire k is not an output, then for any party m the channel

- $\mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,m,k)$

reads from the divergent channel

- $\mathsf{OutShare}(n,m,k) := \mathsf{read}\ \mathsf{OutShare}(n,m,k)$

and thus we may equivalently write the following:

- $\mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}}$

If party n is semi-honest and wire k is an output, then for any party m the channel

- $\mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,m,k)$

reads from the channel

- $\mathsf{OutShare}(n,m,k) := \mathsf{read}\ \mathsf{Share}(m,k)$

so we may substitute:

- $\mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k)$

Summarizing the above, we get the following for channels $\mathsf{OutShare}(-,-,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

If wire k is not an output, then we can rewrite the channels

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,0,k) := \mathsf{read}\ \mathsf{OutShare}(n,0,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \\ \mathsf{OutShare}\text{-}\Sigma(n,m+1,k) := x_\Sigma \leftarrow \mathsf{OutShare}\text{-}\Sigma(n,m,k);\ x_{m+1} \leftarrow \mathsf{OutShare}(n,m+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \end{cases}$

to the following:

- $\mathsf{OutShare}\text{-}\Sigma(n,m,k) := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)$ for $n,m \leqslant N+1$

If wire k is an output, then we can rewrite the channels

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,0,k) := \mathsf{read}\ \mathsf{OutShare}(n,0,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \\ \mathsf{OutShare}\text{-}\Sigma(n,m+1,k) := x_\Sigma \leftarrow \mathsf{OutShare}\text{-}\Sigma(n,m,k);\ x_{m+1} \leftarrow \mathsf{OutShare}(n,m+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \end{cases}$

to the following:

- $\mathsf{OutShare}\text{-}\Sigma(n,m,k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(m,k)$ for $n,m \leqslant N+1$

Summarizing the above, we get the following for channels $\mathsf{OutShare}\text{-}\Sigma(-,-,-)$:

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,m,k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n,m,k) := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } k \text{ not an output} \end{cases}$

If party n is semi-honest and wire k is not an output, then for any party m the channel

- $\mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)$

reads from the divergent channel

- $\mathsf{OutShare}\text{-}\Sigma(n,m,k) := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)$

and thus we may equivalently write the following:

- $\mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}}$

If party n is semi-honest and wire k is an output, then for any party m the channel

- $\mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)$

reads from the channel

- $\mathsf{OutShare}\text{-}\Sigma(n,m,k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(m,k)$

so we may substitute:

- $\mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(m,k)$

Summarizing the above, we get the following for channels $\mathsf{OutShare}\text{-}\Sigma(-,-,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

Finally, we can express the channels

- $\mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)$ for $n \leqslant N+1$

equivalently as follows:

- $\begin{cases} \mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(N+1,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } k \text{ an output} \\ \mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } k \text{ not an output} \end{cases}$

At this point, the internal channels $\mathsf{OutShare}(-,-,-)$ and $\mathsf{OutShare}\text{-}\Sigma(-,-,-)$ are unused and can be eliminated. The simplified version Fin of the final part of the real protocol is therefore as follows:

- $\begin{cases} \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } m,n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m,n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(m,k) \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n,m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(N+1,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } k \text{ an output} \\ \mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

We now show that for each wire, the respective shares of all parties add up to the value carried by the wire. At the same time we express the shares of party $N+1$ in a closed form, as the sum of shares of parties $0,\ldots,N$ plus the value on the wire. We proceed by an induction on circuits: in the presence of the protocol Init and the channels

- $\begin{cases} \mathsf{Share}\text{-}\Sigma(0,k) \coloneqq \mathsf{read}\ \mathsf{Share}(0,k) \\ \quad \text{for } k < K \\ \mathsf{Share}\text{-}\Sigma(m+1,k) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,k);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \text{ and } k < K \end{cases}$

we can express the channels

- $\mathsf{Share}\text{-}\Sigma(N+1,k) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$ for $k < K$

together with the protocol $\mathsf{Shares}(C,K)$ equivalently as the channels

- $\mathsf{Share}(N+1,k) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x$ for $k < K$

together with the following new form of the protocol $\mathsf{Shares}(C,K)$:

- $\mathsf{Shares}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{InShare}(n,p,i)$ for $n \leqslant N$
  - $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq \mathsf{read}\ \mathsf{In}(p,i)$
- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{Share}(n,k)$ for $n \leqslant N$
  - $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,k);\ \mathsf{ret}\ \neg x$
- $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n \oplus y_n$ for $n \leqslant N$
  - $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,k);\ y \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,l);\ \mathsf{ret}\ x \oplus y$
- $\mathsf{Shares}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit}(n,m,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{samp}\ \mathsf{flip} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$

- $\mathsf{RcvdBit}(n,m,K) := b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);$ $x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n,m \leqslant N+1$
- $\begin{cases} \mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n<m \\ \mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m<n \\ \mathsf{Ctrb}(n,n,K) := x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n * y_n \\ \quad \text{for } n \leqslant N+1 \end{cases}$
- $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,0,K) := \mathsf{read}\ \mathsf{Ctrb}(n,0,K) \\ \quad \text{for } n \leqslant N+1 \\ \mathsf{Ctrb}\text{-}\Sigma(n,m+1,K) := b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,K);\ b \leftarrow \mathsf{Ctrb}(n,m+1,K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \end{cases}$
- $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,N+1,K)$ for $n \leqslant N$
- $\mathsf{Share}\text{-}\Sigma(N+1,K) := x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,k);\ y \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,l);\ \mathsf{ret}\ x * y$

To see why this works, we consider each gate in turn.

- In the case of an *input* gate, we start by substituting the inductive form of the channel

  - $\mathsf{Share}(N+1,K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ \mathsf{read}\ \mathsf{InShare}(N+1,p,i)$

  into the closed form of the channel

  - $\mathsf{Share}\text{-}\Sigma(N+1,K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$

  which yields the following:

  - $\mathsf{Share}\text{-}\Sigma(N+1,K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x_{N+1} \leftarrow \mathsf{InShare}(N+1,p,i);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$

  By canceling out two applications of $x_\Sigma \oplus -$ we can reformulate the channel $\mathsf{Share}(N+1,K)$ as follows:

  - $\mathsf{Share}(N+1,K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x_{N+1} \leftarrow \mathsf{InShare}(N+1,p,i);\ \mathsf{ret}\ x_\Sigma \oplus (x_\Sigma \oplus x_{N+1})$

  The above can be expressed more concisely as

  - $\mathsf{Share}(N+1,K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x$

  and this is the desired closed form of the channel $\mathsf{Share}(N+1,K)$.

  We can now turn our attention to the sum of shares. In the presence of the channels

  - $\mathsf{Share}(m,K) := \mathsf{read}\ \mathsf{InShare}(m,p,i)$ for $m \leqslant N$

  and the channels

  - $\mathsf{InShare}(m,p,i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$(m,p,i)$ for $m \leqslant N$
  - $\begin{cases} \mathsf{InShare}\text{-}\$\text{-}\Sigma(0,p,i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$(0,p,i) \\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m+1,p,i) := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,p,i);\ x_{m+1} \leftarrow \mathsf{InShare}\text{-}\$(m+1,p,i);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m<N \end{cases}$

  from the protocol Init we can rewrite the inductive form of the channels

  - $\begin{cases} \mathsf{Share}\text{-}\Sigma(0,K) := \mathsf{read}\ \mathsf{Share}(0,K) \\ \mathsf{Share}\text{-}\Sigma(m+1,K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,K);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m<N \end{cases}$

to the following closed form:

- $\mathsf{Share}\text{-}\Sigma(m, K) \coloneqq \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m, p, i)$ for $m \leqslant N$

Substituting the channel $\mathsf{Share}\text{-}\Sigma(N, K)$ into the channel

- $\mathsf{Share}\text{-}\Sigma(N+1, K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, K);\ x_{N+1} \leftarrow \mathsf{InShare}(N+1, p, i);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$

thus yields the following:

- $\mathsf{Share}\text{-}\Sigma(N+1, K) \coloneqq x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N, p, i);\ x_{N+1} \leftarrow \mathsf{InShare}(N+1, p, i);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$

At this point, we rewrite the closed form of the channels

- $\mathsf{Share}\text{-}\Sigma(m, K) \coloneqq \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m, p, i)$ for $m \leqslant N$

back to their original inductive form:

- $\begin{cases} \mathsf{Share}\text{-}\Sigma(0, K) \coloneqq \mathsf{read}\ \mathsf{Share}(0, K) \\ \mathsf{Share}\text{-}\Sigma(m+1, K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m, K);\ x_{m+1} \leftarrow \mathsf{Share}(m+1, K);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \end{cases}$

In the presence of the channels

- $\mathsf{InShare}(N+1, p, i) \coloneqq \mathsf{read}\ \mathsf{InShare}\text{-}\$(N+1, p, i)$
- $\mathsf{InShare}\text{-}\$(N+1, p, i) \coloneqq x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N, p, i);\ x \leftarrow \mathsf{In}(p, i);\ \mathsf{ret}\ x_\Sigma \oplus x$

from the protocol Init we can further write the channel

- $\mathsf{Share}\text{-}\Sigma(N+1, K) \coloneqq x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N, p, i);\ x_{N+1} \leftarrow \mathsf{InShare}(N+1, p, i);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$

equivalently as follows:

- $\mathsf{Share}\text{-}\Sigma(N+1, K) \coloneqq x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N, p, i);\ x \leftarrow \mathsf{In}(p, i);\ \mathsf{ret}\ x_\Sigma \oplus (x_\Sigma \oplus x)$

We can cancel out the two applications of $x_\Sigma \oplus -$:

- $\mathsf{Share}\text{-}\Sigma(N+1, K) \coloneqq x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N, p, i);\ \mathsf{read}\ \mathsf{In}(p, i)$

This is almost what we want except for the extra dependency on the channel $\mathsf{InShare}\text{-}\$\text{-}\Sigma(N, p, i)$. To remove this dependency, we introduce new internal channels as follows:

- $\mathsf{In}\text{-}\checkmark(p, i) \coloneqq \_ \leftarrow \mathsf{read}\ \mathsf{In}(p, i);\ \mathsf{ret}\ \checkmark$
- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(m, p, i) \coloneqq \_ \leftarrow \mathsf{read}\ \mathsf{InShare}\text{-}\$(m, p, i);\ \mathsf{ret}\ \checkmark$ for $m \leqslant N$
- $\mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(m, p, i) \coloneqq \_ \leftarrow \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m, p, i);\ \mathsf{ret}\ \checkmark$ for $m \leqslant N$

In the presence of the channel $\mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(N, p, i)$ we can write the channel

- $\mathsf{Share}\text{-}\Sigma(N+1, K) \coloneqq x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N, p, i);\ \mathsf{read}\ \mathsf{In}(p, i)$

equivalently as follows:

- $\mathsf{Share}\text{-}\Sigma(N+1, K) \coloneqq \_ \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(N, p, i);\ \mathsf{read}\ \mathsf{In}(p, i)$

We can characterize the timing channels $\mathsf{InShare}\text{-}\$\text{-}\checkmark(-, p, i)$, $\mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(-, p, i)$ equivalently as follows:

- $\mathsf{InShare}\text{-}\$\text{-}\checkmark(m, p, i) \coloneqq \_ \leftarrow \mathsf{In}\text{-}\checkmark(p, i);\ \mathsf{ret}\ \checkmark$ for $m \leqslant N$
- $\begin{cases} \mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(0, p, i) \coloneqq \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\checkmark(0, p, i) \\ \mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(m+1, p, i) \coloneqq \_ \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma\text{-}\checkmark(m, p, i);\ \_ \leftarrow \mathsf{InShare}\text{-}\$\text{-}\checkmark(m+1, p, i);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } m < N \end{cases}$

Furthermore, we can write the above form of the timing channels equivalently as follows:

- $\mathsf{InShare\text{-}\$\text{-}\checkmark}(m,p,i) := \mathsf{read}\ \mathsf{In\text{-}\checkmark}(p,i)$ for $m \leqslant N$
- $\mathsf{InShare\text{-}\$\text{-}\Sigma\text{-}\checkmark}(m,p,i) := \mathsf{read}\ \mathsf{In\text{-}\checkmark}(p,i)$ for $m \leqslant N$

In the presence of the channel $\mathsf{InShare\text{-}\$\text{-}\Sigma\text{-}\checkmark}(N,p,i)$ we can thus write the channel

- $\mathsf{Share\text{-}\Sigma}(N+1,K) := \_ \leftarrow \mathsf{InShare\text{-}\$\text{-}\Sigma\text{-}\checkmark}(N,p,i);\ \mathsf{read}\ \mathsf{In}(p,i)$

equivalently as follows:

- $\mathsf{Share\text{-}\Sigma}(N+1,K) := \_ \leftarrow \mathsf{In\text{-}\checkmark}(p,i);\ \mathsf{read}\ \mathsf{In}(p,i)$

In the presence of the channel $\mathsf{In\text{-}\checkmark}(p,i)$ we can rewrite the above to the following:

- $\mathsf{Share\text{-}\Sigma}(N+1,K) := \mathsf{read}\ \mathsf{In}(p,i)$

But this is precisely the desired inductive form of the channel $\mathsf{Share\text{-}\Sigma}(N+1,K)$.
The channels $\mathsf{In\text{-}\checkmark}(p,i)$, $\mathsf{InShare\text{-}\$\text{-}\checkmark}(-,p,i)$, $\mathsf{InShare\text{-}\$\text{-}\Sigma\text{-}\checkmark}(-,p,i)$ are now unused and can be discarded.

- In the case of a *not* gate, we start by substituting the inductive form of the channel

  - $\mathsf{Share}(N+1,K) := x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ \neg x_{N+1}$

  into the closed form of the channel

  - $\mathsf{Share\text{-}\Sigma}(N+1,K) := x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$

  which yields the following:

  - $\mathsf{Share\text{-}\Sigma}(N+1,K) := x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ x_\Sigma \oplus (\neg x_{N+1})$

  By canceling out two applications of $x_\Sigma \oplus -$ we can reformulate the channel $\mathsf{Share}(N+1,K)$ as follows:

  - $\mathsf{Share}(N+1,K) := x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ x_\Sigma \oplus (x_\Sigma \oplus (\neg x_{N+1}))$

  The above can be expressed more concisely as

  - $\mathsf{Share}(N+1,K) := x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(N,K);\ x \leftarrow \mathsf{Share\text{-}\Sigma}(N+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x$

  and this is the desired closed form of the channel $\mathsf{Share}(N+1,K)$.
  We can now turn our attention to the sum of shares. In the presence of the channels

  - $\mathsf{Share}(m,K) := \mathsf{read}\ \mathsf{Share}(m,k)$ for $m \leqslant N$

  and the channels

  - $\begin{cases} \mathsf{Share\text{-}\Sigma}(0,k) := \mathsf{read}\ \mathsf{Share}(0,k) \\ \mathsf{Share\text{-}\Sigma}(m+1,k) := x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(m,k);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \end{cases}$

  we can rewrite the inductive form of the channels

  - $\begin{cases} \mathsf{Share\text{-}\Sigma}(0,K) := \mathsf{read}\ \mathsf{Share}(0,K) \\ \mathsf{Share\text{-}\Sigma}(m+1,K) := x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(m,K);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \end{cases}$

  to the following closed form:

  - $\mathsf{Share\text{-}\Sigma}(m,K) := \mathsf{read}\ \mathsf{Share\text{-}\Sigma}(m,k)$ for $m \leqslant N$

  Substituting the channel $\mathsf{Share\text{-}\Sigma}(N,K)$ into the channel

– $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ x_\Sigma \oplus (\neg x_{N+1})$

thus yields the following:

– $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ x_\Sigma \oplus (\neg x_{N+1})$

At this point, we rewrite the closed form of the channels

– $\mathsf{Share}\text{-}\Sigma(m,K) \coloneqq \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(m,k)$ for $m \leqslant N$

back to their original inductive form:

– $\begin{cases} \mathsf{Share}\text{-}\Sigma(0,K) \coloneqq \mathsf{read}\ \mathsf{Share}(0,K) \\ \mathsf{Share}\text{-}\Sigma(m+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,K);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \end{cases}$

The negation in the channel

– $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ x_\Sigma \oplus (\neg x_{N+1})$

can be brought to the top level:

– $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ \neg(x_\Sigma \oplus x_{N+1})$

But this is precisely what we get if we substitute the channel

– $\mathsf{Share}\text{-}\Sigma(N+1,k) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$ *(inductive hypothesis)*

into the desired inductive form of the channel

– $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,k);\ \mathsf{ret}\ \neg x$

so we are done.

- In the case of an *xor* gate, we start by substituting the inductive form of the channel

  – $\mathsf{Share}(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);$
  $y_{N+1} \leftarrow \mathsf{Share}(N+1,l);\ \mathsf{ret}\ x_{N+1} \oplus y_{N+1}$

  into the closed form of the channel

  – $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$

  which yields the following:

  – $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);$
  $y_{N+1} \leftarrow \mathsf{Share}(N+1,l);\ \mathsf{ret}\ x_\Sigma \oplus (x_{N+1} \oplus y_{N+1})$

  By canceling out two applications of $x_\Sigma \oplus -$ we can reformulate the channel $\mathsf{Share}(N+1,K)$ as follows:

  – $\mathsf{Share}(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);$
  $y_{N+1} \leftarrow \mathsf{Share}(N+1,l);\ \mathsf{ret}\ x_\Sigma \oplus \big(x_\Sigma \oplus (x_{N+1} \oplus y_{N+1})\big)$

  The above can be expressed more concisely as

  – $\mathsf{Share}(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x$

  and this is the desired closed form of the channel $\mathsf{Share}(N+1,K)$.

  We can now turn our attention to the sum of shares. In the presence of the channels

  – $\mathsf{Share}(m,K) \coloneqq x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);\ \mathsf{ret}\ x_m \oplus y_m$ for $m \leqslant N$

  and the channels

– $\begin{cases} \mathsf{Share}\text{-}\Sigma(0,k) \coloneqq \mathsf{read}\ \mathsf{Share}(0,k) \\ \mathsf{Share}\text{-}\Sigma(m+1,k) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,k);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \end{cases}$

– $\begin{cases} \mathsf{Share}\text{-}\Sigma(0,l) \coloneqq \mathsf{read}\ \mathsf{Share}(0,l) \\ \mathsf{Share}\text{-}\Sigma(m+1,l) \coloneqq y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,l);\ y_{m+1} \leftarrow \mathsf{Share}(m+1,l);\ \mathsf{ret}\ y_\Sigma \oplus y_{m+1} \\ \quad \text{for } m < N \end{cases}$

we can rewrite the inductive form of the channels

– $\begin{cases} \mathsf{Share}\text{-}\Sigma(0,K) \coloneqq \mathsf{read}\ \mathsf{Share}(0,K) \\ \mathsf{Share}\text{-}\Sigma(m+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,K);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \end{cases}$

to the following closed form:

– $\mathsf{Share}\text{-}\Sigma(m,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,l);\ \mathsf{ret}\ x_\Sigma \oplus y_\Sigma$ for $m \leqslant N$

In the base case, we substitute the channel

– $\mathsf{Share}(0,K) \coloneqq x_0 \leftarrow \mathsf{Share}(0,k);\ y_0 \leftarrow \mathsf{Share}(0,l);\ \mathsf{ret}\ x_0 \oplus y_0$

into the inductive form of the channel

– $\mathsf{Share}\text{-}\Sigma(0,K) \coloneqq \mathsf{Share}(0,K)$

which yields the following:

– $\mathsf{Share}\text{-}\Sigma(0,K) \coloneqq x_0 \leftarrow \mathsf{Share}(0,k);\ y_0 \leftarrow \mathsf{Share}(0,l);\ \mathsf{ret}\ x_0 \oplus y_0$

But this is precisely what we get if we substitute the channels

– $\mathsf{Share}\text{-}\Sigma(0,k) \coloneqq x_0 \leftarrow \mathsf{Share}(0,k);\ \mathsf{ret}\ x_0$
– $\mathsf{Share}\text{-}\Sigma(0,l) \coloneqq y_0 \leftarrow \mathsf{Share}(0,l);\ \mathsf{ret}\ y_0$

into the closed form of the channel

– $\mathsf{Share}\text{-}\Sigma(0,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(0,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(0,l);\ \mathsf{ret}\ x_\Sigma \oplus y_\Sigma$

Hence the inductive form and the closed form of the channel $\mathsf{Share}\text{-}\Sigma(0,K)$ are equivalent.
In the inductive case, we substitute the channels

– $\mathsf{Share}\text{-}\Sigma(m,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,l);\ \mathsf{ret}\ x_\Sigma \oplus y_\Sigma$ *(inductive hypothesis)*
– $\mathsf{Share}(m+1,K) \coloneqq x_{m+1} \leftarrow \mathsf{Share}(m+1,k);\ y_{m+1} \leftarrow \mathsf{Share}(m+1,l);\ \mathsf{ret}\ x_{m+1} \oplus y_{m+1}$

into the inductive form of the channel

– $\mathsf{Share}\text{-}\Sigma(m+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,K);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1}$

which yields the following:

– $\mathsf{Share}\text{-}\Sigma(m+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,l);$
$x_{m+1} \leftarrow \mathsf{Share}(m+1,k);\ y_{m+1} \leftarrow \mathsf{Share}(m+1,l);\ \mathsf{ret}\ (x_\Sigma \oplus y_\Sigma) \oplus (x_{m+1} \oplus y_{m+1})$

After a slight rearrangement we get the following:

– $\mathsf{Share}\text{-}\Sigma(m+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,k);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,k);$
$y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,l);\ y_{m+1} \leftarrow \mathsf{Share}(m+1,l);\ \mathsf{ret}\ (x_\Sigma \oplus x_{m+1}) \oplus (y_\Sigma \oplus y_{m+1})$

But this is precisely what we get if we substitute the channels

- $\mathsf{Share}\text{-}\Sigma(m+1,k) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,k);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1}$
- $\mathsf{Share}\text{-}\Sigma(m+1,l) \coloneqq y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,l);\ y_{m+1} \leftarrow \mathsf{Share}(m+1,l);\ \mathsf{ret}\ y_\Sigma \oplus y_{m+1}$

into the closed form of the channel

- $\mathsf{Share}\text{-}\Sigma(m+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m+1,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m+1,l);\ \mathsf{ret}\ x_\Sigma \oplus y_\Sigma$

Hence the inductive form and the closed form of the channel $\mathsf{Share}\text{-}\Sigma(m+1,K)$ are equivalent.

Substituting the channel

- $\mathsf{Share}\text{-}\Sigma(N,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,l);\ \mathsf{ret}\ x_\Sigma \oplus y_\Sigma$

into the channel

- $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);$
  $y_{N+1} \leftarrow \mathsf{Share}(N+1,l);\ \mathsf{ret}\ x_\Sigma \oplus (x_{N+1} \oplus y_{N+1})$

yields the following:

- $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,l);$
  $x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ y_{N+1} \leftarrow \mathsf{Share}(N+1,l);\ \mathsf{ret}\ (x_\Sigma \oplus y_\Sigma) \oplus (x_{N+1} \oplus y_{N+1})$

At this point, we rewrite the closed form of the channels

- $\mathsf{Share}\text{-}\Sigma(m,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,l);\ \mathsf{ret}\ x_\Sigma \oplus y_\Sigma$ for $m \leqslant N$

back to their original inductive form:

- $\begin{cases} \mathsf{Share}\text{-}\Sigma(0,K) \coloneqq \mathsf{read}\ \mathsf{Share}(0,K) \\ \mathsf{Share}\text{-}\Sigma(m+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,K);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \end{cases}$

After a slight rearrangement of the channel

- $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,l);$
  $x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ y_{N+1} \leftarrow \mathsf{Share}(N+1,l);\ \mathsf{ret}\ (x_\Sigma \oplus y_\Sigma) \oplus (x_{N+1} \oplus y_{N+1})$

we get the following:

- $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);$
  $y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,l);\ y_{N+1} \leftarrow \mathsf{Share}(N+1,l);\ \mathsf{ret}\ (x_\Sigma \oplus x_{N+1}) \oplus (y_\Sigma \oplus y_{N+1})$

But this is precisely what we get if we substitute the channels

- $\mathsf{Share}\text{-}\Sigma(N+1,k) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$ *(inductive hypothesis)*
- $\mathsf{Share}\text{-}\Sigma(N+1,l) \coloneqq y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,l);\ y_{N+1} \leftarrow \mathsf{Share}(N+1,l);\ \mathsf{ret}\ y_\Sigma \oplus y_{N+1}$ *(inductive hypothesis)*

into the desired inductive form of the channel

- $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,k);\ y \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,l);\ \mathsf{ret}\ x \oplus y$

so we are done.

- In the case of an *and* gate, we start by substituting the inductive form of the channel

  - $\mathsf{Share}(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(N+1,N+1,K)$

  into the closed form of the channel

  - $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$

which yields the following:

- $\mathsf{Share}\text{-}\Sigma(N+1,K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x_{N+1} \leftarrow \mathsf{Ctrb}\text{-}\Sigma(N+1,N+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$

By canceling out two applications of $x_\Sigma \oplus -$ we can reformulate the channel $\mathsf{Share}(N+1,K)$ as follows:

- $\mathsf{Share}(N+1,K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x_{N+1} \leftarrow \mathsf{Ctrb}\text{-}\Sigma(N+1,N+1,K);\ \mathsf{ret}\ x_\Sigma \oplus (x_\Sigma \oplus x_{N+1})$

The above can be expressed more concisely as

- $\mathsf{Share}(N+1,K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x$

and this is the desired closed form of the channel $\mathsf{Share}(N+1,K)$.

We can now turn our attention to the sum of shares. For each party n, the share $\mathsf{Share}(n,K)$ is a sum of the $N+2$ contributions $\mathsf{Ctrb}(n,0,K),\ldots,\mathsf{Ctrb}(n,N+1,K)$. So when performing the sum of shares, we are summing up the table of contributions $\{\mathsf{Ctrb}(i,j,K)\}_{i,j\leqslant N+1}$ by columns. To make this structure explicit, we introduce new internal channels

- $\mathsf{Col}(i,j) := \mathsf{Ctrb}\text{-}\Sigma(i,j,K)$ for $i,j \leqslant N+1$

that record the sum of the $j+1$ contributions $\mathsf{Ctrb}(i,0,K),\ldots,\mathsf{Ctrb}(i,j,K)$. Additionally, we introduce new internal channels

- $\begin{cases} \mathsf{Col}\text{-}\Sigma(0,j) := \mathsf{read}\ \mathsf{Col}(0,j) \\ \quad \text{for } j \leqslant N+1 \\ \mathsf{Col}\text{-}\Sigma(i+1,j) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,j);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,j);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i \leqslant N \text{ and } j \leqslant N+1 \end{cases}$

that record the sum of the $i+1$ columns $\mathsf{Col}(0,j),\ldots,\mathsf{Col}(i,j)$ of length $j+1$.

In the presence of the channels

- $\mathsf{Share}(m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(m,N+1,K)$ for $m \leqslant N$

and the channels

- $\mathsf{Col}(m,N+1) := \mathsf{Ctrb}\text{-}\Sigma(m,N+1,K)$ for $m \leqslant N$
- $\begin{cases} \mathsf{Col}\text{-}\Sigma(0,N+1) := \mathsf{read}\ \mathsf{Col}(0,N+1) \\ \mathsf{Col}\text{-}\Sigma(m+1,N+1) := x_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(m,N+1);\ x_{m+1} \leftarrow \mathsf{Col}(m+1,N+1);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \end{cases}$

we can rewrite the inductive form of the channels

- $\begin{cases} \mathsf{Share}\text{-}\Sigma(0,K) := \mathsf{read}\ \mathsf{Share}(0,K) \\ \mathsf{Share}\text{-}\Sigma(m+1,K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,K);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \end{cases}$

to the following closed form:

- $\mathsf{Share}\text{-}\Sigma(m,K) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(m,N+1)$ for $m \leqslant N$

Substituting the channel $\mathsf{Share}\text{-}\Sigma(N,K)$ into the channel

- $\mathsf{Share}\text{-}\Sigma(N+1,K) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,K);\ x_{N+1} \leftarrow \mathsf{Ctrb}\text{-}\Sigma(N+1,N+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$

thus yields the following:

- $\mathsf{Share}\text{-}\Sigma(N+1,K) := x_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(N,N+1);\ x_{N+1} \leftarrow \mathsf{Ctrb}\text{-}\Sigma(N+1,N+1,K);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$

At this point, we rewrite the closed form of the channels

- $\mathsf{Share}\text{-}\Sigma(m, K) \coloneqq \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(m, N+1)$ for $m \leqslant N$

back to their original inductive form:

- $\begin{cases} \mathsf{Share}\text{-}\Sigma(0, K) \coloneqq \mathsf{read}\ \mathsf{Share}(0, K) \\ \mathsf{Share}\text{-}\Sigma(m+1, K) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m, K);\ x_{m+1} \leftarrow \mathsf{Share}(m+1, K);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \end{cases}$

In the presence of the channels

- $\mathsf{Col}(N+1, N+1) \coloneqq \mathsf{Ctrb}\text{-}\Sigma(N+1, N+1, K)$
- $\mathsf{Col}\text{-}\Sigma(N+1, N+1) \coloneqq x_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(N, N+1);\ x_{N+1} \leftarrow \mathsf{Col}(N+1, N+1);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$

we can further write the channel

- $\mathsf{Share}\text{-}\Sigma(N+1, K) \coloneqq x_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(N, N+1);\ x_{N+1} \leftarrow \mathsf{Ctrb}\text{-}\Sigma(N+1, N+1, K);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$

equivalently as follows:

- $\mathsf{Share}\text{-}\Sigma(N+1, K) \coloneqq \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(N+1, N+1)$

In the presence of the channels

- $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(i, 0, K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(i, 0, K) \\ \quad \text{for } i \leqslant N+1 \\ \mathsf{Ctrb}\text{-}\Sigma(i, j+1, K) \coloneqq b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(i, j, K);\ b_{j+1} \leftarrow \mathsf{Ctrb}(i, j+1, K);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } i \leqslant N+1 \text{ and } j \leqslant N \end{cases}$

we can rewrite the closed form of the channels

- $\mathsf{Col}(i, j) \coloneqq \mathsf{Ctrb}\text{-}\Sigma(i, j, K)$ for $i, j \leqslant N+1$

to the following inductive form:

- $\begin{cases} \mathsf{Col}(i, 0) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(i, 0, K) \\ \quad \text{for } i \leqslant N+1 \\ \mathsf{Col}(i, j+1) \coloneqq b_\Sigma \leftarrow \mathsf{Col}(i, j);\ b_{j+1} \leftarrow \mathsf{Ctrb}(i, j+1, K);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } i \leqslant N+1 \text{ and } j \leqslant N \end{cases}$

Instead of summing up the table of contributions $\{\mathsf{Ctrb}(i, j, K)\}_{i,j \leqslant N+1}$ by columns, we can sum it up by rows. To this end, we introduce new internal channels

- $\begin{cases} \mathsf{Row}(0, j) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(0, j, K) \\ \quad \text{for } j \leqslant N+1 \\ \mathsf{Row}(i+1, j) \coloneqq b_\Sigma \leftarrow \mathsf{Row}(i, j);\ b_{i+1} \leftarrow \mathsf{Ctrb}(i+1, j, K);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i \leqslant N \text{ and } j \leqslant N+1 \end{cases}$

that record the sum of the $i+1$ contributions $\mathsf{Ctrb}(0, j, K), \ldots, \mathsf{Ctrb}(i, j, K)$. Additionally, we introduce new internal channels

- $\begin{cases} \mathsf{Row}\text{-}\Sigma(i, 0) \coloneqq \mathsf{read}\ \mathsf{Row}(i, 0) \\ \quad \text{for } i \leqslant N+1 \\ \mathsf{Row}\text{-}\Sigma(i, j+1) \coloneqq b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i, j);\ b_{j+1} \leftarrow \mathsf{Row}(i, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } i \leqslant N+1 \text{ and } j \leqslant N \end{cases}$

that record the sum of the $j+1$ rows $\mathsf{Row}(i, 0), \ldots, \mathsf{Row}(i, j)$ of length $i+1$.

Of course, summing up the contributions by columns is equivalent to summing them up by rows: in the presence of the channels

– $\begin{cases} \mathsf{Row}(0,j) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(0,j,K) \\ \quad \text{for } j \leqslant N+1 \\ \mathsf{Row}(i+1,j) \coloneqq b_\Sigma \leftarrow \mathsf{Row}(i,j);\ b_{i+1} \leftarrow \mathsf{Ctrb}(i+1,j,K);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i \leqslant N \text{ and } j \leqslant N+1 \end{cases}$

– $\begin{cases} \mathsf{Col}(i,0) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(i,0,K) \\ \quad \text{for } i \leqslant N+1 \\ \mathsf{Col}(i,j+1) \coloneqq b_\Sigma \leftarrow \mathsf{Col}(i,j);\ b_{j+1} \leftarrow \mathsf{Ctrb}(i,j+1,K);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } i \leqslant N+1 \text{ and } j \leqslant N \end{cases}$

we can express the protocol

– $\begin{cases} \mathsf{Col}\text{-}\Sigma(0,j) \coloneqq \mathsf{read}\ \mathsf{Col}(0,j) \\ \quad \text{for } j \leqslant N+1 \\ \mathsf{Col}\text{-}\Sigma(i+1,j) \coloneqq b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,j);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,j);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i \leqslant N \text{ and } j \leqslant N+1 \end{cases}$

– $\begin{cases} \mathsf{Row}\text{-}\Sigma(i,0) \coloneqq \mathsf{read}\ \mathsf{Row}(i,0) \\ \quad \text{for } i \leqslant N+1 \\ \mathsf{Row}\text{-}\Sigma(i,j+1) \coloneqq b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i,j);\ b_{j+1} \leftarrow \mathsf{Row}(i,j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } i \leqslant N+1 \text{ and } j \leqslant N \end{cases}$

where the channels $\mathsf{Col}\text{-}\Sigma(-,-)$ have an inductive form equivalently as the protocol

– $\mathsf{Col}\text{-}\Sigma(i,j) \coloneqq \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(i,j)$ for $i,j \leqslant N+1$

– $\begin{cases} \mathsf{Row}\text{-}\Sigma(i,0) \coloneqq \mathsf{read}\ \mathsf{Row}(i,0) \\ \quad \text{for } i \leqslant N+1 \\ \mathsf{Row}\text{-}\Sigma(i,j+1) \coloneqq b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i,j);\ b_{j+1} \leftarrow \mathsf{Row}(i,j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } i \leqslant N+1 \text{ and } j \leqslant N \end{cases}$

where the channels $\mathsf{Col}\text{-}\Sigma(-,-)$ have a closed form.

To show this, we proceed by induction on i. In the base case $i = 0$, we need to show that the channels

– $\mathsf{Col}\text{-}\Sigma(0,j) \coloneqq \mathsf{read}\ \mathsf{Col}(0,j)$ for $j \leqslant N+1$

– $\begin{cases} \mathsf{Row}\text{-}\Sigma(0,0) \coloneqq \mathsf{read}\ \mathsf{Row}(0,0) \\ \mathsf{Row}\text{-}\Sigma(0,j+1) \coloneqq b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(0,j);\ b_{j+1} \leftarrow \mathsf{Row}(0,j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } j \leqslant N \end{cases}$

can be expressed equivalently as the channels below:

– $\mathsf{Col}\text{-}\Sigma(0,j) \coloneqq \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(0,j)$ for $j \leqslant N+1$

– $\begin{cases} \mathsf{Row}\text{-}\Sigma(0,0) \coloneqq \mathsf{read}\ \mathsf{Row}(0,0) \\ \mathsf{Row}\text{-}\Sigma(0,j+1) \coloneqq b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(0,j);\ b_{j+1} \leftarrow \mathsf{Row}(0,j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } j \leqslant N \end{cases}$

We split the proof into two parts. In the first part of the base case $i = 0$, we show that the protocol

– $\mathsf{Col}\text{-}\Sigma(0,j) \coloneqq \mathsf{read}\ \mathsf{Col}(0,j)$ for $j \leqslant N+1$

– $\begin{cases} \mathsf{Row}\text{-}\Sigma(0,0) \coloneqq \mathsf{read}\ \mathsf{Row}(0,0) \\ \mathsf{Row}\text{-}\Sigma(0,j+1) \coloneqq b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(0,j);\ b_{j+1} \leftarrow \mathsf{Row}(0,j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } j \leqslant N \end{cases}$

where the channels $\mathsf{Row}\text{-}\Sigma(0,-)$ have an inductive form can be expressed equivalently as the protocol

- $\mathsf{Col}\text{-}\Sigma(0,j) := \mathsf{read}\ \mathsf{Col}(0,j)$ for $j \leqslant N+1$
- $\mathsf{Row}\text{-}\Sigma(0,j) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(0,j)$ for $j \leqslant N+1$

where the channels $\mathsf{Row}\text{-}\Sigma(0,-)$ have a closed form.

We proceed by induction on j. In the base case $j=0$, we need to show that the channels

- $\mathsf{Col}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Col}(0,0)$
- $\mathsf{Row}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Row}(0,0)$

can be expressed equivalently as the channels below:

- $\mathsf{Col}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Col}(0,0)$
- $\mathsf{Row}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(0,0)$

Substituting the channel

- $\mathsf{Row}(0,0) := \mathsf{read}\ \mathsf{Ctrb}(0,0,K)$

into the inductive form of the channel

- $\mathsf{Row}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Row}(0,0)$

yields the following:

- $\mathsf{Row}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Ctrb}(0,0,K)$

In the presence of the channel

- $\mathsf{Col}(0,0) := \mathsf{read}\ \mathsf{Ctrb}(0,0,K)$

we can express the channel $\mathsf{Row}\text{-}\Sigma(0,0)$ equivalently as follows:

- $\mathsf{Row}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Col}(0,0)$

In the presence of the channel

- $\mathsf{Col}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Col}(0,0)$

we can express the channel $\mathsf{Row}\text{-}\Sigma(0,0)$ equivalently as follows:

- $\mathsf{Row}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(0,0)$

This is precisely the desired closed form of the channel $\mathsf{Row}\text{-}\Sigma(0,0)$, which finishes the base case $j=0$.

In the inductive case $j+1$, we need to show that the channels

- $\mathsf{Col}\text{-}\Sigma(0,j+1) := \mathsf{read}\ \mathsf{Col}(0,j+1)$
- $\mathsf{Row}\text{-}\Sigma(0,j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(0,j);\ b_{j+1} \leftarrow \mathsf{Row}(0,j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1}$

can be expressed equivalently as the channels below:

- $\mathsf{Col}\text{-}\Sigma(0,j+1) := \mathsf{read}\ \mathsf{Col}(0,j+1)$
- $\mathsf{Row}\text{-}\Sigma(0,j+1) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(0,j+1)$

Substituting the channel

- $\mathsf{Row}\text{-}\Sigma(0,j) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(0,j)$ *(inductive hypothesis)*

into the inductive form of the channel

– $\mathsf{Row}\text{-}\Sigma(0, j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(0, j);\ b_{j+1} \leftarrow \mathsf{Row}(0, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1}$

yields the following:

– $\mathsf{Row}\text{-}\Sigma(0, j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(0, j);\ b_{j+1} \leftarrow \mathsf{Row}(0, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1}$

Further substituting the channels

– $\mathsf{Col}\text{-}\Sigma(0, j) := \mathsf{read}\ \mathsf{Col}(0, j)$
– $\mathsf{Row}(0, j+1) := \mathsf{read}\ \mathsf{Ctrb}(0, j+1, K)$

into the channel $\mathsf{Row}\text{-}\Sigma(0, j+1)$ yields the following:

– $\mathsf{Row}\text{-}\Sigma(0, j+1) := b_\Sigma \leftarrow \mathsf{Col}(0, j);\ b_{j+1} \leftarrow \mathsf{Ctrb}(0, j+1, K);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1}$

In the presence of the channel

– $\mathsf{Col}(0, j+1) := b_\Sigma \leftarrow \mathsf{Col}(0, j);\ b_{j+1} \leftarrow \mathsf{Ctrb}(0, j+1, K);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1}$

we can express the channel $\mathsf{Row}\text{-}\Sigma(0, j+1)$ more concisely as follows:

– $\mathsf{Row}\text{-}\Sigma(0, j+1) := \mathsf{read}\ \mathsf{Col}(0, j+1)$

In the presence of the channel

– $\mathsf{Col}\text{-}\Sigma(0, j+1) := \mathsf{read}\ \mathsf{Col}(0, j+1)$

we can further write the channel $\mathsf{Row}\text{-}\Sigma(0, j+1)$ as follows:

– $\mathsf{Row}\text{-}\Sigma(0, j+1) := \mathsf{Col}\text{-}\Sigma(0, j+1)$

This is precisely the desired closed form of the channel $\mathsf{Row}\text{-}\Sigma(0, j+1)$, which finishes the inductive case $j+1$. The first part of the base case $i = 0$ is now complete.

In the second part of the base case $i = 0$, we show that the protocol

– $\mathsf{Col}\text{-}\Sigma(0, j) := \mathsf{read}\ \mathsf{Col}(0, j)$ for $j \leqslant N+1$
– $\mathsf{Row}\text{-}\Sigma(0, j) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(0, j)$ for $j \leqslant N+1$

where the channels $\mathsf{Col}\text{-}\Sigma(0, -)$ have an inductive form while the channels $\mathsf{Row}\text{-}\Sigma(0, -)$ have a closed form can be expressed equivalently as the protocol

– $\mathsf{Col}\text{-}\Sigma(0, j) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(0, j)$ for $j \leqslant N+1$

– $\begin{cases} \mathsf{Row}\text{-}\Sigma(0, 0) := \mathsf{read}\ \mathsf{Row}(0, 0) \\ \mathsf{Row}\text{-}\Sigma(0, j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(0, j);\ b_{j+1} \leftarrow \mathsf{Row}(0, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } j \leqslant N \end{cases}$

where the channels $\mathsf{Col}\text{-}\Sigma(0, -)$ have a closed form while the channels $\mathsf{Row}\text{-}\Sigma(0, -)$ have an inductive form.

We proceed by induction on j. In the base case $j = 0$, we need to show that the channels below:

– $\mathsf{Col}\text{-}\Sigma(0, 0) := \mathsf{read}\ \mathsf{Col}(0, 0)$
– $\mathsf{Row}\text{-}\Sigma(0, 0) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(0, 0)$

can be expressed equivalently as the channels

– $\mathsf{Col}\text{-}\Sigma(0, 0) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(0, 0)$
– $\mathsf{Row}\text{-}\Sigma(0, 0) := \mathsf{read}\ \mathsf{Row}(0, 0)$

Substituting the channel

- $\mathsf{Row}(0,0) := \mathsf{read}\ \mathsf{Ctrb}(0,0,K)$

into the inductive form of the channel

- $\mathsf{Row}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Row}(0,0)$

yields the following:

- $\mathsf{Row}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Ctrb}(0,0,K)$

In the presence of the channel

- $\mathsf{Col}(0,0) := \mathsf{read}\ \mathsf{Ctrb}(0,0,K)$

we can express the channel $\mathsf{Row}\text{-}\Sigma(0,0)$ equivalently as follows:

- $\mathsf{Row}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Col}(0,0)$

Substituting the channel $\mathsf{Row}\text{-}\Sigma(0,0)$ into the closed form of the channel

- $\mathsf{Col}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(0,0)$

yields the following:

- $\mathsf{Col}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Col}(0,0)$

This is precisely the desired inductive form of the channel $\mathsf{Col}\text{-}\Sigma(0,0)$, in the presence of which we can write the channel

- $\mathsf{Row}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Col}(0,0)$

equivalently as follows:

- $\mathsf{Row}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(0,0)$

This is precisely the desired closed form of the channel $\mathsf{Row}\text{-}\Sigma(0,0)$, which finishes the base case $j=0$.

In the inductive case $j+1$, we need to show that the channels

- $\mathsf{Col}\text{-}\Sigma(0,j+1) := \mathsf{read}\ \mathsf{Col}(0,j+1)$
- $\mathsf{Row}\text{-}\Sigma(0,j+1) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(0,j+1)$

can be expressed equivalently as the channels below:

- $\mathsf{Col}\text{-}\Sigma(0,j+1) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(0,j+1)$
- $\mathsf{Row}\text{-}\Sigma(0,j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(0,j);\ b_{j+1} \leftarrow \mathsf{Row}(0,j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1}$

Substituting the channel

- $\mathsf{Row}\text{-}\Sigma(0,j) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(0,j)$ *(inductive hypothesis)*

into the inductive form of the channel

- $\mathsf{Row}\text{-}\Sigma(0,j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(0,j);\ b_{j+1} \leftarrow \mathsf{Row}(0,j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1}$

yields the following:

- $\mathsf{Row}\text{-}\Sigma(0,j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(0,j);\ b_{j+1} \leftarrow \mathsf{Row}(0,j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1}$

Further substituting the channels

- $\mathsf{Col}\text{-}\Sigma(0,j) := \mathsf{read}\ \mathsf{Col}(0,j)$ *(inductive hypothesis)*
- $\mathsf{Row}(0,j+1) := \mathsf{read}\ \mathsf{Ctrb}(0,j+1,K)$

into the channel $\mathsf{Row}\text{-}\Sigma(0, j+1)$ yields the following:

- $\mathsf{Row}\text{-}\Sigma(0, j+1) := b_\Sigma \leftarrow \mathsf{Col}(0, j);\ b_{j+1} \leftarrow \mathsf{Ctrb}(0, j+1, K);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1}$

In the presence of the channel

- $\mathsf{Col}(0, j+1) := b_\Sigma \leftarrow \mathsf{Col}(0, j);\ b_{j+1} \leftarrow \mathsf{Ctrb}(0, j+1, K);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1}$

we can express the channel $\mathsf{Row}\text{-}\Sigma(0, j+1)$ more concisely as follows:

- $\mathsf{Row}\text{-}\Sigma(0, j+1) := \mathsf{read}\ \mathsf{Col}(0, j+1)$

Substituting the channel $\mathsf{Row}\text{-}\Sigma(0, j+1)$ into the closed form of the channel

- $\mathsf{Col}\text{-}\Sigma(0, j+1) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(0, j+1)$

yields the following:

- $\mathsf{Col}\text{-}\Sigma(0, j+1) := \mathsf{read}\ \mathsf{Col}(0, j+1)$

But this is precisely the desired inductive form of the channel $\mathsf{Col}\text{-}\Sigma(0, j+1)$, in the presence of which we can write the channel

- $\mathsf{Row}\text{-}\Sigma(0, j+1) := \mathsf{read}\ \mathsf{Col}(0, j+1)$

equivalently as follows:

- $\mathsf{Row}\text{-}\Sigma(0, j+1) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(0, j+1)$

This is precisely the desired closed form of the channel $\mathsf{Row}\text{-}\Sigma(0, j+1)$, which finishes the inductive case $j+1$. The second part of the base case $i = 0$ is now complete.

In the inductive case $i+1$, we need to show that the channels

- $\mathsf{Col}\text{-}\Sigma(i+1, j) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i, j);\ b_{i+1} \leftarrow \mathsf{Col}(i+1, j);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$ for $j \leqslant N+1$
- $\begin{cases} \mathsf{Row}\text{-}\Sigma(i+1, 0) := \mathsf{read}\ \mathsf{Row}(i+1, 0) \\ \mathsf{Row}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i+1, j);\ b_{j+1} \leftarrow \mathsf{Row}(i+1, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } j \leqslant N \end{cases}$

can be expressed equivalently as the channels below:

- $\mathsf{Col}\text{-}\Sigma(i+1, j) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(i+1, j)$ for $j \leqslant N+1$
- $\begin{cases} \mathsf{Row}\text{-}\Sigma(i+1, 0) := \mathsf{read}\ \mathsf{Row}(i+1, 0) \\ \mathsf{Row}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i+1, j);\ b_{j+1} \leftarrow \mathsf{Row}(i+1, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } j \leqslant N \end{cases}$

We split the proof into two parts. In the first part of the inductive case $i+1$, we show that the protocol

- $\mathsf{Col}\text{-}\Sigma(i+1, j) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i, j);\ b_{i+1} \leftarrow \mathsf{Col}(i+1, j);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$ for $j \leqslant N+1$
- $\begin{cases} \mathsf{Row}\text{-}\Sigma(i+1, 0) := \mathsf{read}\ \mathsf{Row}(i+1, 0) \\ \mathsf{Row}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i+1, j);\ b_{j+1} \leftarrow \mathsf{Row}(i+1, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } j \leqslant N \end{cases}$

where the channels $\mathsf{Row}\text{-}\Sigma(i+1, -)$ have an inductive form can be expressed equivalently as the protocol

- $\mathsf{Col}\text{-}\Sigma(i+1, j) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i, j);\ b_{i+1} \leftarrow \mathsf{Col}(i+1, j);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$ for $j \leqslant N+1$
- $\mathsf{Row}\text{-}\Sigma(i+1, j) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(i+1, j)$ for $j \leqslant N+1$

where the channels $\mathsf{Row}\text{-}\Sigma(i+1,-)$ have a closed form.

We proceed by induction on j. In the base case $j = 0$, we need to show that the channels

- $\mathsf{Col}\text{-}\Sigma(i+1,0) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,0);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,0);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$
- $\mathsf{Row}\text{-}\Sigma(i+1,0) := \mathsf{read}\ \mathsf{Row}(i+1,0)$

can be expressed equivalently as the channels below:

- $\mathsf{Col}\text{-}\Sigma(i+1,0) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,0);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,0);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$
- $\mathsf{Row}\text{-}\Sigma(i+1,0) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(i+1,0)$

Substituting the channel

- $\mathsf{Row}(i+1,0) := b_\Sigma \leftarrow \mathsf{Row}(i,0);\ b_{i+1} \leftarrow \mathsf{Ctrb}(i+1,0,K);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

into the inductive form of the channel

- $\mathsf{Row}\text{-}\Sigma(i+1,0) := \mathsf{read}\ \mathsf{Row}(i+1,0)$

yields the following:

- $\mathsf{Row}\text{-}\Sigma(i+1,0) := b_\Sigma \leftarrow \mathsf{Row}(i,0);\ b_{i+1} \leftarrow \mathsf{Ctrb}(i+1,0,K);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

In the presence of the channels

- $\mathsf{Row}\text{-}\Sigma(i,0) := \mathsf{read}\ \mathsf{Row}(i,0)$
- $\mathsf{Col}(i+1,0) := \mathsf{read}\ \mathsf{Ctrb}(i+1,0,K)$

we can express the channel $\mathsf{Row}\text{-}\Sigma(i+1,0)$ equivalently as follows:

- $\mathsf{Row}\text{-}\Sigma(i+1,0) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i,0);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,0);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

In the presence of the channel

- $\mathsf{Col}\text{-}\Sigma(i,0) := \mathsf{Row}\text{-}\Sigma(i,0)$ *(inductive hypothesis)*

we can further write the channel $\mathsf{Row}\text{-}\Sigma(i+1,0)$ as follows:

- $\mathsf{Row}\text{-}\Sigma(i+1,0) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,0);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,0);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

Finally, in the presence of the channel

- $\mathsf{Col}\text{-}\Sigma(i+1,0) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,0);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,0);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

we can write the channel $\mathsf{Row}\text{-}\Sigma(i+1,0)$ equivalently as follows:

- $\mathsf{Row}\text{-}\Sigma(i+1,0) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(i+1,0)$

This is precisely the desired closed form of the channel $\mathsf{Row}\text{-}\Sigma(i+1,0)$, which finishes the base case $j = 0$.

In the inductive case $j+1$, we need to show that the channels

- $\mathsf{Col}\text{-}\Sigma(i+1,j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,j+1);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$
- $\mathsf{Row}\text{-}\Sigma(i+1,j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i+1,j);\ b_{j+1} \leftarrow \mathsf{Row}(i+1,j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1}$

can be expressed equivalently as the channels below:

- $\mathsf{Col}\text{-}\Sigma(i+1,j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,j+1);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$
- $\mathsf{Row}\text{-}\Sigma(i+1,j+1) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(i+1,j+1)$

Substituting the channel

– $\mathsf{Row}\text{-}\Sigma(i+1, j) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(i+1, j)$ *(inductive hypothesis)*

into the inductive form of the channel

– $\mathsf{Row}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i+1, j);\ b_{j+1} \leftarrow \mathsf{Row}(i+1, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1}$

yields the following:

– $\mathsf{Row}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i+1, j);\ b_{j+1} \leftarrow \mathsf{Row}(i+1, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1}$

Further substituting the channels

– $\mathsf{Col}\text{-}\Sigma(i+1, j) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i, j);\ b_\mathsf{C} \leftarrow \mathsf{Col}(i+1, j);\ \mathsf{ret}\ b_\Sigma \oplus b_\mathsf{C}$
– $\mathsf{Row}(i+1, j+1) := b_\mathsf{R} \leftarrow \mathsf{Row}(i, j+1);\ b \leftarrow \mathsf{Ctrb}(i+1, j+1, K);\ \mathsf{ret}\ b_\mathsf{R} \oplus b$

into the channel $\mathsf{Row}\text{-}\Sigma(i+1, j+1)$ yields the following:

– $\mathsf{Row}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i, j);\ b_\mathsf{C} \leftarrow \mathsf{Col}(i+1, j);$
$b_\mathsf{R} \leftarrow \mathsf{Row}(i, j+1);\ b \leftarrow \mathsf{Ctrb}(i+1, j+1, K);\ \mathsf{ret}\ (b_\Sigma \oplus b_\mathsf{C}) \oplus (b_\mathsf{R} \oplus b)$

After a slight rearrangement we get the following:

– $\mathsf{Row}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i, j);\ b_\mathsf{R} \leftarrow \mathsf{Row}(i, j+1);$
$b_\mathsf{C} \leftarrow \mathsf{Col}(i+1, j);\ b \leftarrow \mathsf{Ctrb}(i+1, j+1, K);\ \mathsf{ret}\ (b_\Sigma \oplus b_\mathsf{R}) \oplus (b_\mathsf{C} \oplus b)$

A final substitution of the channel

– $\mathsf{Col}\text{-}\Sigma(i, j) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(i, j)$ *(inductive hypothesis)*

into the channel $\mathsf{Row}\text{-}\Sigma(i+1, j+1)$ yields the following:

– $\mathsf{Row}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i, j);\ b_\mathsf{R} \leftarrow \mathsf{Row}(i, j+1);$
$b_\mathsf{C} \leftarrow \mathsf{Col}(i+1, j);\ b \leftarrow \mathsf{Ctrb}(i+1, j+1, K);\ \mathsf{ret}\ (b_\Sigma \oplus b_\mathsf{R}) \oplus (b_\mathsf{C} \oplus b)$

In the presence of the channels

– $\mathsf{Row}\text{-}\Sigma(i, j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i, j);\ b_\mathsf{R} \leftarrow \mathsf{Row}(i, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_\mathsf{R}$
– $\mathsf{Col}(i+1, j+1) := b_\mathsf{C} \leftarrow \mathsf{Col}(i+1, j);\ b \leftarrow \mathsf{Ctrb}(i+1, j+1, K);\ \mathsf{ret}\ b_\mathsf{C} \oplus b$

we can express the channel $\mathsf{Row}\text{-}\Sigma(i+1, j+1)$ more concisely as follows:

– $\mathsf{Row}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i, j+1);\ b_{i+1} \leftarrow \mathsf{Col}(i+1, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

In the presence of the channel

– $\mathsf{Col}\text{-}\Sigma(i, j+1) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(i, j+1)$ *(inductive hypothesis)*

we can further write the channel $\mathsf{Row}\text{-}\Sigma(i+1, j+1)$ equivalently as follows:

– $\mathsf{Row}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i, j+1);\ b_{i+1} \leftarrow \mathsf{Col}(i+1, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

Finally, in the presence of the channel

– $\mathsf{Col}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i, j+1);\ b_{i+1} \leftarrow \mathsf{Col}(i+1, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

we can write the channel $\mathsf{Row}\text{-}\Sigma(i+1, j+1)$ equivalently as follows:

– $\mathsf{Row}\text{-}\Sigma(i+1, j+1) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(i+1, j+1)$

This is precisely the desired closed form of the channel $\mathsf{Row}\text{-}\Sigma(i+1, j+1)$, which finishes the inductive case $j+1$. The first part of the inductive case $i+1$ is now complete.

In the second part of the inductive case $i+1$, we show that the protocol

- $\mathsf{Col}\text{-}\Sigma(i+1,j) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,j);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,j);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$ for $j \leqslant N+1$
- $\mathsf{Row}\text{-}\Sigma(i+1,j) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(i+1,j)$ for $j \leqslant N+1$

where the channels $\mathsf{Col}\text{-}\Sigma(i+1,-)$ have an inductive form while the channels $\mathsf{Row}\text{-}\Sigma(i+1,-)$ have a closed form can be expressed equivalently as the protocol

- $\mathsf{Col}\text{-}\Sigma(i+1,j) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(i+1,j)$ for $j \leqslant N+1$
- $\begin{cases} \mathsf{Row}\text{-}\Sigma(i+1,0) := \mathsf{read}\ \mathsf{Row}(i+1,0) \\ \mathsf{Row}\text{-}\Sigma(i+1,j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i+1,j);\ b_{j+1} \leftarrow \mathsf{Row}(i+1,j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } j \leqslant N \end{cases}$

where the channels $\mathsf{Col}\text{-}\Sigma(i+1,-)$ have a closed form while the channels $\mathsf{Row}\text{-}\Sigma(i+1,-)$ have an inductive form.

We proceed by induction on j. In the base case $j=0$, we need to show that the channels

- $\mathsf{Col}\text{-}\Sigma(i+1,0) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,0);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,0);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$
- $\mathsf{Row}\text{-}\Sigma(i+1,0) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(i+1,0)$

can be expressed equivalently as the channels below:

- $\mathsf{Col}\text{-}\Sigma(i+1,0) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(i+1,0)$
- $\mathsf{Row}\text{-}\Sigma(i+1,0) := \mathsf{read}\ \mathsf{Row}(i+1,0)$

Substituting the channel

- $\mathsf{Row}(i+1,0) := b_\Sigma \leftarrow \mathsf{Row}(i,0);\ b_{i+1} \leftarrow \mathsf{Ctrb}(i+1,0,K);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

into the inductive form of the channel

- $\mathsf{Row}\text{-}\Sigma(i+1,0) := \mathsf{read}\ \mathsf{Row}(i+1,0)$

yields the following:

- $\mathsf{Row}\text{-}\Sigma(i+1,0) := b_\Sigma \leftarrow \mathsf{Row}(i,0);\ b_{i+1} \leftarrow \mathsf{Ctrb}(i+1,0,K);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

In the presence of the channels

- $\mathsf{Row}\text{-}\Sigma(i,0) := \mathsf{read}\ \mathsf{Row}(i,0)$
- $\mathsf{Col}(i+1,0) := \mathsf{read}\ \mathsf{Ctrb}(i+1,0,K)$

we can express the channel $\mathsf{Row}\text{-}\Sigma(i+1,0)$ equivalently as follows:

- $\mathsf{Row}\text{-}\Sigma(i+1,0) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i,0);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,0);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

In the presence of the channel

- $\mathsf{Col}\text{-}\Sigma(i,0) := \mathsf{Row}\text{-}\Sigma(i,0)$ *(inductive hypothesis)*

we can further write the channel $\mathsf{Row}\text{-}\Sigma(i+1,0)$ as follows:

- $\mathsf{Row}\text{-}\Sigma(i+1,0) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,0);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,0);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

Substituting the channel $\mathsf{Row}\text{-}\Sigma(i+1,0)$ into the closed form of the channel

- $\mathsf{Col}\text{-}\Sigma(i+1,0) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(i+1,0)$

yields the following:

- $\mathsf{Col}\text{-}\Sigma(i+1,0) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,0);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,0);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

But this is precisely the desired inductive form of the channel $\mathsf{Col}\text{-}\Sigma(i+1,0)$, in the presence of which we can write the channel

- $\mathsf{Row}\text{-}\Sigma(i+1,0) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,0);\; b_{i+1} \leftarrow \mathsf{Col}(i+1,0);\; \mathsf{ret}\; b_\Sigma \oplus b_{i+1}$

equivalently as follows:

- $\mathsf{Row}\text{-}\Sigma(i+1,0) := \mathsf{read}\; \mathsf{Col}\text{-}\Sigma(i+1,0)$

This is precisely the desired closed form of the channel $\mathsf{Row}\text{-}\Sigma(i+1,0)$, which finishes the base case $j=0$.

In the inductive case $j+1$, we need to show that the channels

- $\mathsf{Col}\text{-}\Sigma(i+1,j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,j+1);\; b_{i+1} \leftarrow \mathsf{Col}(i+1,j+1);\; \mathsf{ret}\; b_\Sigma \oplus b_{i+1}$
- $\mathsf{Row}\text{-}\Sigma(i+1,j+1) := \mathsf{read}\; \mathsf{Col}\text{-}\Sigma(i+1,j+1)$

can be expressed equivalently as the channels below:

- $\mathsf{Col}\text{-}\Sigma(i+1,j+1) := \mathsf{read}\; \mathsf{Row}\text{-}\Sigma(i+1,j+1)$
- $\mathsf{Row}\text{-}\Sigma(i+1,j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i+1,j);\; b_{j+1} \leftarrow \mathsf{Row}(i+1,j+1);\; \mathsf{ret}\; b_\Sigma \oplus b_{j+1}$

Substituting the channel

- $\mathsf{Row}\text{-}\Sigma(i+1,j) := \mathsf{read}\; \mathsf{Col}\text{-}\Sigma(i+1,j)$ *(inductive hypothesis)*

into the inductive form of the channel

- $\mathsf{Row}\text{-}\Sigma(i+1,j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i+1,j);\; b_{j+1} \leftarrow \mathsf{Row}(i+1,j+1);\; \mathsf{ret}\; b_\Sigma \oplus b_{j+1}$

yields the following:

- $\mathsf{Row}\text{-}\Sigma(i+1,j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i+1,j);\; b_{j+1} \leftarrow \mathsf{Row}(i+1,j+1);\; \mathsf{ret}\; b_\Sigma \oplus b_{j+1}$

Further substituting the channels

- $\mathsf{Col}\text{-}\Sigma(i+1,j) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,j);\; b_{\mathsf{C}} \leftarrow \mathsf{Col}(i+1,j);\; \mathsf{ret}\; b_\Sigma \oplus b_{\mathsf{C}}$ *(inductive hypothesis)*
- $\mathsf{Row}(i+1,j+1) := b_{\mathsf{R}} \leftarrow \mathsf{Row}(i,j+1);\; b \leftarrow \mathsf{Ctrb}(i+1,j+1,K);\; \mathsf{ret}\; b_{\mathsf{R}} \oplus b$

into the channel $\mathsf{Row}\text{-}\Sigma(i+1,j+1)$ yields the following:

- $\mathsf{Row}\text{-}\Sigma(i+1,j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,j);\; b_{\mathsf{C}} \leftarrow \mathsf{Col}(i+1,j);$
  $b_{\mathsf{R}} \leftarrow \mathsf{Row}(i,j+1);\; b \leftarrow \mathsf{Ctrb}(i+1,j+1,K);\; \mathsf{ret}\; (b_\Sigma \oplus b_{\mathsf{C}}) \oplus (b_{\mathsf{R}} \oplus b)$

After a slight rearrangement we get the following:

- $\mathsf{Row}\text{-}\Sigma(i+1,j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,j);\; b_{\mathsf{R}} \leftarrow \mathsf{Row}(i,j+1);$
  $b_{\mathsf{C}} \leftarrow \mathsf{Col}(i+1,j);\; b \leftarrow \mathsf{Ctrb}(i+1,j+1,K);\; \mathsf{ret}\; (b_\Sigma \oplus b_{\mathsf{R}}) \oplus (b_{\mathsf{C}} \oplus b)$

A final substitution of the channel

- $\mathsf{Col}\text{-}\Sigma(i,j) := \mathsf{read}\; \mathsf{Row}\text{-}\Sigma(i,j)$ *(inductive hypothesis)*

into the channel $\mathsf{Row}\text{-}\Sigma(i+1,j+1)$ yields the following:

- $\mathsf{Row}\text{-}\Sigma(i+1,j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i,j);\; b_{\mathsf{R}} \leftarrow \mathsf{Row}(i,j+1);$
  $b_{\mathsf{C}} \leftarrow \mathsf{Col}(i+1,j);\; b \leftarrow \mathsf{Ctrb}(i+1,j+1,K);\; \mathsf{ret}\; (b_\Sigma \oplus b_{\mathsf{R}}) \oplus (b_{\mathsf{C}} \oplus b)$

In the presence of the channels

- $\mathsf{Row}\text{-}\Sigma(i,j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i,j);\; b_{\mathsf{R}} \leftarrow \mathsf{Row}(i,j+1);\; \mathsf{ret}\; b_\Sigma \oplus b_{\mathsf{R}}$
- $\mathsf{Col}(i+1,j+1) := b_{\mathsf{C}} \leftarrow \mathsf{Col}(i+1,j);\; b \leftarrow \mathsf{Ctrb}(i+1,j+1,K);\; \mathsf{ret}\; b_{\mathsf{C}} \oplus b$

we can express the channel $\mathsf{Row}\text{-}\Sigma(i+1, j+1)$ more concisely as follows:

- $\mathsf{Row}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i, j+1);\ b_{i+1} \leftarrow \mathsf{Col}(i+1, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

In the presence of the channel

- $\mathsf{Col}\text{-}\Sigma(i, j+1) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(i, j+1)$ *(inductive hypothesis)*

we can further write the channel $\mathsf{Row}\text{-}\Sigma(i+1, j+1)$ equivalently as follows:

- $\mathsf{Row}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i, j+1);\ b_{i+1} \leftarrow \mathsf{Col}(i+1, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

Substituting the channel $\mathsf{Row}\text{-}\Sigma(i+1, j+1)$ into the closed form of the channel

- $\mathsf{Col}\text{-}\Sigma(i+1, j+1) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(i+1, j+1)$

yields the following:

- $\mathsf{Col}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i, j+1);\ b_{i+1} \leftarrow \mathsf{Col}(i+1, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

But this is precisely the desired inductive form of the channel $\mathsf{Col}\text{-}\Sigma(i+1, j+1)$, in the presence of which we can write the channel

- $\mathsf{Row}\text{-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i, j+1);\ b_{i+1} \leftarrow \mathsf{Col}(i+1, j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

equivalently as follows:

- $\mathsf{Row}\text{-}\Sigma(i+1, j+1) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(i+1, j+1)$

This is precisely the desired closed form of the channel $\mathsf{Row}\text{-}\Sigma(i+1, j+1)$, which finishes the inductive case $j+1$. The second part of the inductive case $i+1$ is now complete.

We have thus shown that summing up the contributions by columns is equivalent to summing them up by rows. We will use this fact shortly.

We are ultimately interested in summing up $N+2$ columns of length $N+2$ (or equivalently, $N+2$ rows of length $N+2$). The summation of $n+1$ columns of length $n+1$ is therefore a special case that deserves its own name: we add new internal channels

- $\mathsf{Sqr}(n) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(n, n)$ for $n \leqslant N+1$

that record the sum of the $n+1$ columns $\mathsf{Col}(0, n), \ldots, \mathsf{Col}(n, n)$ of length $n+1$.

In the presence of the channels $\mathsf{Sqr}(-)$ we can write the channel

- $\mathsf{Share}\text{-}\Sigma(N+1, K) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(N+1, N+1)$

equivalently as follows:

- $\mathsf{Share}\text{-}\Sigma(N+1, K) := \mathsf{read}\ \mathsf{Sqr}(N+1)$

Our next goal is to show that in the presence of the channels

- $\begin{cases} \mathsf{Row}(0, j) := \mathsf{read}\ \mathsf{Ctrb}(0, j, K) \\ \quad \text{for } j \leqslant N+1 \\ \mathsf{Row}(i+1, j) := b_\Sigma \leftarrow \mathsf{Row}(i, j);\ b_{i+1} \leftarrow \mathsf{Ctrb}(i+1, j, K);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i \leqslant N \text{ and } j \leqslant N+1 \end{cases}$
- $\begin{cases} \mathsf{Col}(i, 0) := \mathsf{read}\ \mathsf{Ctrb}(i, 0, K) \\ \quad \text{for } i \leqslant N+1 \\ \mathsf{Col}(i, j+1) := b_\Sigma \leftarrow \mathsf{Col}(i, j);\ b_{j+1} \leftarrow \mathsf{Ctrb}(i, j+1, K);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } i \leqslant N+1 \text{ and } j \leqslant N \end{cases}$

as well as the channels

- $\begin{cases} \mathsf{Col}\text{-}\Sigma(0,j) := \mathsf{read}\ \mathsf{Col}(0,j) \\ \quad \text{for } j \leqslant N+1 \\ \mathsf{Col}\text{-}\Sigma(i+1,j) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,j);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,j);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i \leqslant N \text{ and } j \leqslant N+1 \end{cases}$
- $\begin{cases} \mathsf{Row}\text{-}\Sigma(i,0) := \mathsf{read}\ \mathsf{Row}(i,0) \\ \quad \text{for } i \leqslant N+1 \\ \mathsf{Row}\text{-}\Sigma(i,j+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(i,j);\ b_{j+1} \leftarrow \mathsf{Row}(i,j+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } i \leqslant N+1 \text{ and } j \leqslant N \end{cases}$

we can express the closed form of the channels

- $\mathsf{Sqr}(n) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(n,n)$ for $n \leqslant N+1$

equivalently by induction:

- $\begin{cases} \mathsf{Sqr}(0) := \mathsf{read}\ \mathsf{Col}(0,0) \\ \mathsf{Sqr}(n+1) := b_\mathsf{S} \leftarrow \mathsf{Sqr}(n);\ b_\mathsf{R} \leftarrow \mathsf{Row}(n,n+1);\ b_\mathsf{C} \leftarrow \mathsf{Col}(n+1,n+1);\ \mathsf{ret}\ (b_\mathsf{S} \oplus b_\mathsf{R}) \oplus b_\mathsf{C} \\ \quad \text{for } n \leqslant N \end{cases}$

In the base case, substituting the channel

- $\mathsf{Col}\text{-}\Sigma(0,0) := \mathsf{read}\ \mathsf{Col}(0,0)$

into the closed form of the channel

- $\mathsf{Sqr}(0) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(0,0)$

yields precisely the desired inductive form:

- $\mathsf{Sqr}(0) := \mathsf{read}\ \mathsf{Col}(0,0)$

In the inductive case, substituting the channel

- $\mathsf{Col}\text{-}\Sigma(n+1,n+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(n,n+1);\ b_\mathsf{C} \leftarrow \mathsf{Col}(n+1,n+1);\ \mathsf{ret}\ b_\Sigma \oplus b_\mathsf{C}$

into the closed form of the channel

- $\mathsf{Sqr}(n+1) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(n+1,n+1)$

yields the following:

- $\mathsf{Sqr}(n+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(n,n+1);\ b_\mathsf{C} \leftarrow \mathsf{Col}(n+1,n+1);\ \mathsf{ret}\ b_\Sigma \oplus b_\mathsf{C}$

We now appeal to our earlier observation that the inductive form of the channels

- $\begin{cases} \mathsf{Col}\text{-}\Sigma(0,j) := \mathsf{read}\ \mathsf{Col}(0,j) \\ \quad \text{for } j \leqslant N+1 \\ \mathsf{Col}\text{-}\Sigma(i+1,j) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i,j);\ b_{i+1} \leftarrow \mathsf{Col}(i+1,j);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i \leqslant N \text{ and } j \leqslant N+1 \end{cases}$

is equivalent to the following closed form:

- $\mathsf{Col}\text{-}\Sigma(i,j) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(i,j)$ for $i,j \leqslant N+1$

Substituting the channel

- $\mathsf{Col}\text{-}\Sigma(n,n+1) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(n,n+1)$

into the channel

- $\mathsf{Sqr}(n+1) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(n, n+1);\ b_\mathsf{C} \leftarrow \mathsf{Col}(n+1, n+1);\ \mathsf{ret}\ b_\Sigma \oplus b_\mathsf{C}$

yields the following:

- $\mathsf{Sqr}(n+1) := b_\Sigma \leftarrow \mathsf{Row}\text{-}\Sigma(n, n+1);\ b_\mathsf{C} \leftarrow \mathsf{Col}(n+1, n+1);\ \mathsf{ret}\ b_\Sigma \oplus b_\mathsf{C}$

Further substituting the channel

- $\mathsf{Row}\text{-}\Sigma(n, n+1) := b_\mathsf{S} \leftarrow \mathsf{Row}\text{-}\Sigma(n, n);\ b_\mathsf{R} \leftarrow \mathsf{Row}(n, n+1);\ \mathsf{ret}\ b_\mathsf{S} \oplus b_\mathsf{R}$

into the channel $\mathsf{Sqr}(n+1)$ yields the following:

- $\mathsf{Sqr}(n+1) := b_\mathsf{S} \leftarrow \mathsf{Row}\text{-}\Sigma(n, n);\ b_\mathsf{R} \leftarrow \mathsf{Row}(n, n+1);\ b_\mathsf{C} \leftarrow \mathsf{Col}(n+1, n+1);\ \mathsf{ret}\ (b_\mathsf{S} \oplus b_\mathsf{R}) \oplus b_\mathsf{C}$

In the presence of the channel

- $\mathsf{Col}\text{-}\Sigma(n, n) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(n, n)$

we can write the channel $\mathsf{Sqr}(n+1)$ equivalently as follows:

- $\mathsf{Sqr}(n+1) := b_\mathsf{S} \leftarrow \mathsf{Col}\text{-}\Sigma(n, n);\ b_\mathsf{R} \leftarrow \mathsf{Row}(n, n+1);\ b_\mathsf{C} \leftarrow \mathsf{Col}(n+1, n+1);\ \mathsf{ret}\ (b_\mathsf{S} \oplus b_\mathsf{R}) \oplus b_\mathsf{C}$

At this point, we rewrite the closed form of the channels

- $\mathsf{Col}\text{-}\Sigma(i, j) := \mathsf{read}\ \mathsf{Row}\text{-}\Sigma(i, j)$ for $i, j \leqslant N+1$

back to their original inductive form:

- $\begin{cases} \mathsf{Col}\text{-}\Sigma(0, j) := \mathsf{read}\ \mathsf{Col}(0, j) \\ \quad \text{for } j \leqslant N+1 \\ \mathsf{Col}\text{-}\Sigma(i+1, j) := b_\Sigma \leftarrow \mathsf{Col}\text{-}\Sigma(i, j);\ b_{i+1} \leftarrow \mathsf{Col}(i+1, j);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i \leqslant N \text{ and } j \leqslant N+1 \end{cases}$

In the presence of the channel

- $\mathsf{Sqr}(n) := \mathsf{read}\ \mathsf{Col}\text{-}\Sigma(n, n)$ *(inductive hypothesis)*

we can further write the channel $\mathsf{Sqr}(n+1)$ as follows:

- $\mathsf{Sqr}(n+1) := b_\mathsf{S} \leftarrow \mathsf{Sqr}(n);\ b_\mathsf{R} \leftarrow \mathsf{Row}(n, n+1);\ b_\mathsf{C} \leftarrow \mathsf{Col}(n+1, n+1);\ \mathsf{ret}\ (b_\mathsf{S} \oplus b_\mathsf{R}) \oplus b_\mathsf{C}$

This is precisely the desired inductive form of the channel $\mathsf{Sqr}(n+1)$.
The channels $\mathsf{Row}\text{-}\Sigma(-,-)$, $\mathsf{Col}\text{-}\Sigma(-,-)$ are now unused and can be discarded.
We now recall the channels below:

- $\begin{cases} \mathsf{Share}\text{-}\Sigma(0, k) := \mathsf{read}\ \mathsf{Share}(0, k) \\ \mathsf{Share}\text{-}\Sigma(m+1, k) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m, k);\ x_{m+1} \leftarrow \mathsf{Share}(m+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \end{cases}$
- $\begin{cases} \mathsf{Share}\text{-}\Sigma(0, l) := \mathsf{read}\ \mathsf{Share}(0, l) \\ \mathsf{Share}\text{-}\Sigma(m+1, l) := y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m, l);\ y_{m+1} \leftarrow \mathsf{Share}(m+1, l);\ \mathsf{ret}\ y_\Sigma \oplus y_{m+1} \\ \quad \text{for } m < N \end{cases}$

We also recall the following channels:

- $\mathsf{Share}\text{-}\Sigma(N+1, k) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x_{N+1} \leftarrow \mathsf{Share}(N+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x_{N+1}$ *(inductive hypothesis)*
- $\mathsf{Share}\text{-}\Sigma(N+1, l) := y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, l);\ y_{N+1} \leftarrow \mathsf{Share}(N+1, l);\ \mathsf{ret}\ y_\Sigma \oplus y_{N+1}$ *(inductive hypothesis)*

Altogether we thus have the following:

– $\begin{cases} \mathsf{Share}\text{-}\Sigma(0,k) := \mathsf{read}\ \mathsf{Share}(0,k) \\ \mathsf{Share}\text{-}\Sigma(m+1,k) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,k);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m \leqslant N \end{cases}$

– $\begin{cases} \mathsf{Share}\text{-}\Sigma(0,l) := \mathsf{read}\ \mathsf{Share}(0,l) \\ \mathsf{Share}\text{-}\Sigma(m+1,l) := y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,l);\ y_{m+1} \leftarrow \mathsf{Share}(m+1,l);\ \mathsf{ret}\ y_\Sigma \oplus y_{m+1} \\ \quad \text{for } m \leqslant N \end{cases}$

Our final goal is to show that in the presence of the channels

– $\begin{cases} \mathsf{Row}(0,j) := \mathsf{read}\ \mathsf{Ctrb}(0,j,K) \\ \quad \text{for } j \leqslant N+1 \\ \mathsf{Row}(i+1,j) := b_\Sigma \leftarrow \mathsf{Row}(i,j);\ b_{i+1} \leftarrow \mathsf{Ctrb}(i+1,j,K);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i \leqslant N \text{ and } j \leqslant N+1 \end{cases}$

– $\begin{cases} \mathsf{Col}(i,0) := \mathsf{read}\ \mathsf{Ctrb}(i,0,K) \\ \quad \text{for } i \leqslant N+1 \\ \mathsf{Col}(i,j+1) := b_\Sigma \leftarrow \mathsf{Col}(i,j);\ b_{j+1} \leftarrow \mathsf{Ctrb}(i,j+1,K);\ \mathsf{ret}\ b_\Sigma \oplus b_{j+1} \\ \quad \text{for } i \leqslant N+1 \text{ and } j \leqslant N \end{cases}$

as well as the channels

– $\begin{cases} \mathsf{SendBit}(n,m,K) := x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{samp\ flip} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$

– $\mathsf{RcvdBit}(n,m,K) := b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);$
$x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n,m \leqslant N+1$

– $\begin{cases} \mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}(n,n,K) := x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n * y_n \\ \quad \text{for } n \leqslant N+1 \end{cases}$

together with the channels

– $\begin{cases} \mathsf{Share}\text{-}\Sigma(0,k) := \mathsf{read}\ \mathsf{Share}(0,k) \\ \mathsf{Share}\text{-}\Sigma(m+1,k) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,k);\ x_{m+1} \leftarrow \mathsf{Share}(m+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m \leqslant N \end{cases}$

– $\begin{cases} \mathsf{Share}\text{-}\Sigma(0,l) := \mathsf{read}\ \mathsf{Share}(0,l) \\ \mathsf{Share}\text{-}\Sigma(m+1,l) := y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m,l);\ y_{m+1} \leftarrow \mathsf{Share}(m+1,l);\ \mathsf{ret}\ y_\Sigma \oplus y_{m+1} \\ \quad \text{for } m \leqslant N \end{cases}$

we can express the inductive form of the channels

– $\begin{cases} \mathsf{Sqr}(0) := \mathsf{read}\ \mathsf{Col}(0,0) \\ \mathsf{Sqr}(n+1) := b_\mathsf{S} \leftarrow \mathsf{Sqr}(n);\ b_\mathsf{R} \leftarrow \mathsf{Row}(n,n+1);\ b_\mathsf{C} \leftarrow \mathsf{Col}(n+1,n+1);\ \mathsf{ret}\ (b_\mathsf{S} \oplus b_\mathsf{R}) \oplus b_\mathsf{C} \\ \quad \text{for } n \leqslant N \end{cases}$

equivalently using the following closed form:

- $\mathsf{Sqr}(n) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n,l);\ \mathsf{ret}\ x_\Sigma * y_\Sigma$ for $n \leqslant N+1$

In the base case $n = 0$, substituting the channel

- $\mathsf{Col}(0,0) := \mathsf{read}\ \mathsf{Ctrb}(0,0,K)$

into the inductive form of the channel

- $\mathsf{Sqr}(0) := \mathsf{read}\ \mathsf{Col}(0,0)$

yields the following:

- $\mathsf{Sqr}(0) := \mathsf{read}\ \mathsf{Ctrb}(0,0,K)$

Further substituting the channel

- $\mathsf{Ctrb}(0,0,K) := x_0 \leftarrow \mathsf{Share}(0,k);\ y_0 \leftarrow \mathsf{Share}(0,l);\ \mathsf{ret}\ x_0 * y_0$

into the channel $\mathsf{Sqr}(0)$ yields the following:

- $\mathsf{Sqr}(0) := x_0 \leftarrow \mathsf{Share}(0,k);\ y_0 \leftarrow \mathsf{Share}(0,l);\ \mathsf{ret}\ x_0 * y_0$

But this is precisely what we get if we substitute the channels

- $\mathsf{Share}\text{-}\Sigma(0,k) := x_0 \leftarrow \mathsf{Share}(0,k);\ \mathsf{ret}\ x_0$
- $\mathsf{Share}\text{-}\Sigma(0,l) := y_0 \leftarrow \mathsf{Share}(0,l);\ \mathsf{ret}\ y_0$

into the closed form of the channel below:

- $\mathsf{Sqr}(0) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(0,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(0,l);\ \mathsf{ret}\ x_\Sigma * y_\Sigma$

Hence the inductive form and the closed form of the channel $\mathsf{Sqr}(0)$ are equivalent.

In the inductive case $n+1$, we substitute the channel

- $\mathsf{Col}(n+1,n+1) := b_\mathsf{C} \leftarrow \mathsf{Col}(n+1,n);\ b \leftarrow \mathsf{Ctrb}(n+1,n+1,K);\ \mathsf{ret}\ b_\mathsf{C} \oplus b$

into the channel

- $\mathsf{Sqr}(n+1) := b_\mathsf{S} \leftarrow \mathsf{Sqr}(n);\ b_\mathsf{R} \leftarrow \mathsf{Row}(n,n+1);\ b_\mathsf{C} \leftarrow \mathsf{Col}(n+1,n+1);\ \mathsf{ret}\ (b_\mathsf{S} \oplus b_\mathsf{R}) \oplus b_\mathsf{C}$

which yields the following:

- $\mathsf{Sqr}(n+1) := b_\mathsf{S} \leftarrow \mathsf{Sqr}(n);\ b_\mathsf{R} \leftarrow \mathsf{Row}(n,n+1);\ b_\mathsf{C} \leftarrow \mathsf{Col}(n+1,n);$
  $b \leftarrow \mathsf{Ctrb}(n+1,n+1,K);\ \mathsf{ret}\ (b_\mathsf{S} \oplus b_\mathsf{R}) \oplus (b_\mathsf{C} \oplus b)$

After a slight rearrangement we get the following:

- $\mathsf{Sqr}(n+1) := b_\mathsf{S} \leftarrow \mathsf{Sqr}(n);\ b_\mathsf{R} \leftarrow \mathsf{Row}(n,n+1);\ b_\mathsf{C} \leftarrow \mathsf{Col}(n+1,n);$
  $b \leftarrow \mathsf{Ctrb}(n+1,n+1,K);\ \mathsf{ret}\ b_\mathsf{S} \oplus (b_\mathsf{R} \oplus b_\mathsf{C}) \oplus b$

We now introduce new internal channels

- $\mathsf{DiagRefl}(i) := b \leftarrow \mathsf{Ctrb}(i,n+1,K);\ b_* \leftarrow \mathsf{Ctrb}(n+1,i,K);\ \mathsf{ret}\ b \oplus b_*$ for $i \leqslant n$
- $\mathsf{RowCol}(i) := b_\mathsf{R} \leftarrow \mathsf{Row}(i,n+1);\ b_\mathsf{C} \leftarrow \mathsf{Col}(n+1,i);\ \mathsf{ret}\ b_\mathsf{R} \oplus b_\mathsf{C}$ for $i \leqslant n$

that sum up a single contribution, respectively a row of contributions, with its reflection (a single contribution, respectively a column of contributions) along the bottom left-top right diagonal.

In the presence of the channel

- $\mathsf{RowCol}(n) := b_\mathsf{R} \leftarrow \mathsf{Row}(n,n+1);\ b_\mathsf{C} \leftarrow \mathsf{Col}(n+1,n);\ \mathsf{ret}\ b_\mathsf{R} \oplus b_\mathsf{C}$

we can express the channel $\mathsf{Sqr}(n+1)$ more concisely as follows:

– $\mathsf{Sqr}(n+1) := b_{\mathsf{S}} \leftarrow \mathsf{Sqr}(n);\ b_{\mathsf{RC}} \leftarrow \mathsf{RowCol}(n);\ b \leftarrow \mathsf{Ctrb}(n+1, n+1, K);\ \mathsf{ret}\ b_{\mathsf{S}} \oplus b_{\mathsf{RC}} \oplus b$

We now observe that we can express the closed form of the channels

– $\mathsf{RowCol}(i) := b_{\mathsf{R}} \leftarrow \mathsf{Row}(i, n+1);\ b_{\mathsf{C}} \leftarrow \mathsf{Col}(n+1, i);\ \mathsf{ret}\ b_{\mathsf{R}} \oplus b_{\mathsf{C}}$ for $i \leqslant n$

equivalently by induction:

– $\begin{cases} \mathsf{RowCol}(0) := \mathsf{read}\ \mathsf{DiagRefl}(0) \\ \mathsf{RowCol}(i+1) := b_\Sigma \leftarrow \mathsf{RowCol}(i);\ b_{i+1} \leftarrow \mathsf{DiagRefl}(i+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i < n \end{cases}$

In the base case $i = 0$, we substitute the channel

– $\mathsf{DiagRefl}(0) := b \leftarrow \mathsf{Ctrb}(0, n+1, K);\ b_* \leftarrow \mathsf{Ctrb}(n+1, 0, K);\ \mathsf{ret}\ b \oplus b_*$

into the inductive form of the channel

– $\mathsf{RowCol}(0) := \mathsf{read}\ \mathsf{DiagRefl}(0)$

which yields the following:

– $\mathsf{RowCol}(0) := b \leftarrow \mathsf{Ctrb}(0, n+1, K);\ b_* \leftarrow \mathsf{Ctrb}(n+1, 0, K);\ \mathsf{ret}\ b \oplus b_*$

But this is precisely what we get if we substitute the channels

– $\mathsf{Row}(0, n+1) := b \leftarrow \mathsf{Ctrb}(0, n+1, K);\ \mathsf{ret}\ b$
– $\mathsf{Col}(n+1, 0) := b_* \leftarrow \mathsf{Ctrb}(n+1, 0, K);\ \mathsf{ret}\ b_*$

into the closed form of the channel below:

– $\mathsf{RowCol}(0) := b_{\mathsf{R}} \leftarrow \mathsf{Row}(0, n+1);\ b_{\mathsf{C}} \leftarrow \mathsf{Col}(n+1, 0);\ \mathsf{ret}\ b_{\mathsf{R}} \oplus b_{\mathsf{C}}$

Hence the closed form and the inductive form of the channel $\mathsf{RowCol}(0)$ are equivalent.
In the inductive case $i+1$, we substitute the channels

– $\mathsf{RowCol}(i) := b_{\mathsf{R}} \leftarrow \mathsf{Row}(i, n+1);\ b_{\mathsf{C}} \leftarrow \mathsf{Col}(n+1, i);\ \mathsf{ret}\ b_{\mathsf{R}} \oplus b_{\mathsf{C}}$ *(inductive hypothesis)*
– $\mathsf{DiagRefl}(i+1) := b \leftarrow \mathsf{Ctrb}(i+1, n+1, K);\ b_* \leftarrow \mathsf{Ctrb}(n+1, i+1, K);\ \mathsf{ret}\ b \oplus b_*$

into the inductive form of the channel

– $\mathsf{RowCol}(i+1) := b_\Sigma \leftarrow \mathsf{RowCol}(i);\ b_{i+1} \leftarrow \mathsf{DiagRefl}(i+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

which yields the following:

– $\mathsf{RowCol}(i+1) := b_{\mathsf{R}} \leftarrow \mathsf{Row}(i, n+1);\ b_{\mathsf{C}} \leftarrow \mathsf{Col}(n+1, i);$
$b \leftarrow \mathsf{Ctrb}(i+1, n+1, K);\ b_* \leftarrow \mathsf{Ctrb}(n+1, i+1, K);\ \mathsf{ret}\ (b_{\mathsf{R}} \oplus b_{\mathsf{C}}) \oplus (b \oplus b_*)$

After a slight rearrangement we get the following:

– $\mathsf{RowCol}(i+1) := b_{\mathsf{R}} \leftarrow \mathsf{Row}(i, n+1);\ b \leftarrow \mathsf{Ctrb}(i+1, n+1, K);$
$b_{\mathsf{C}} \leftarrow \mathsf{Col}(n+1, i);\ b_* \leftarrow \mathsf{Ctrb}(n+1, i+1, K);\ \mathsf{ret}\ (b_{\mathsf{R}} \oplus b) \oplus (b_{\mathsf{C}} \oplus b_*)$

But this is precisely what we get if we substitute the channels

– $\mathsf{Row}(i+1, n+1) := b_{\mathsf{R}} \leftarrow \mathsf{Row}(i, n+1);\ b \leftarrow \mathsf{Ctrb}(i+1, n+1, K);\ \mathsf{ret}\ b_{\mathsf{R}} \oplus b$
– $\mathsf{Col}(n+1, i+1) := b_{\mathsf{C}} \leftarrow \mathsf{Col}(n+1, i);\ b_* \leftarrow \mathsf{Ctrb}(n+1, i+1, K);\ \mathsf{ret}\ b_{\mathsf{C}} \oplus b_*$

into the closed form of the channel below:

- $\mathsf{RowCol}(i+1) := b_{\mathsf{R}} \leftarrow \mathsf{Row}(i+1, n+1);\ b_{\mathsf{C}} \leftarrow \mathsf{Col}(n+1, i+1);\ \mathsf{ret}\ b_{\mathsf{R}} \oplus b_{\mathsf{C}}$

Hence the closed form and the inductive form of the channel $\mathsf{RowCol}(i+1)$ are equivalent.

We next observe that we can express the channels

- $\mathsf{DiagRefl}(i) := b \leftarrow \mathsf{Ctrb}(i, n+1, K);\ b_* \leftarrow \mathsf{Ctrb}(n+1, i, K);\ \mathsf{ret}\ b \oplus b_*$ for $i \leqslant n$

equivalently as follows:

- $\mathsf{DiagRefl}(i) := x_i \leftarrow \mathsf{Share}(i, k);\ y_i \leftarrow \mathsf{Share}(i, l);\ x_{n+1} \leftarrow \mathsf{Share}(n+1, k);$
  $y_{n+1} \leftarrow \mathsf{Share}(n+1, l);\ \mathsf{ret}\ (x_i * y_{n+1}) \oplus (x_{n+1} * y_i)$ for $i \leqslant n$

For any party $i \leqslant n$, substituting the channels

- $\mathsf{Ctrb}(i, n+1, K) := \mathsf{read}\ \mathsf{SendBit}(i, n+1, K)$
- $\mathsf{Ctrb}(n+1, i, K) := \mathsf{read}\ \mathsf{RcvdBit}(n+1, i, K)$

into the channel

- $\mathsf{DiagRefl}(i) := b \leftarrow \mathsf{Ctrb}(i, n+1, K);\ b_* \leftarrow \mathsf{Ctrb}(n+1, i, K);\ \mathsf{ret}\ b \oplus b_*$

yields the following:

- $\mathsf{DiagRefl}(i) := b \leftarrow \mathsf{SendBit}(i, n+1, K);\ b_* \leftarrow \mathsf{RcvdBit}(n+1, i, K);\ \mathsf{ret}\ b \oplus b_*$

Further substituting the channel

- $\mathsf{RcvdBit}(n+1, i, K) := b \leftarrow \mathsf{SendBit}(i, n+1, K);\ x_i \leftarrow \mathsf{Share}(i, k);\ y_i \leftarrow \mathsf{Share}(i, l);$
  $x_{n+1} \leftarrow \mathsf{Share}(n+1, k);\ y_{n+1} \leftarrow \mathsf{Share}(n+1, l);\ \mathsf{ret}\ b \oplus (x_i * y_{n+1}) \oplus (x_{n+1} * y_i)$

into the channel $\mathsf{DiagRefl}(i)$ yields the following:

- $\mathsf{DiagRefl}(i) := b \leftarrow \mathsf{SendBit}(i, n+1, K);\ x_i \leftarrow \mathsf{Share}(i, k);\ y_i \leftarrow \mathsf{Share}(i, l);$
  $x_{n+1} \leftarrow \mathsf{Share}(n+1, k);\ y_{n+1} \leftarrow \mathsf{Share}(n+1, l);\ \mathsf{ret}\ b \oplus \big(b \oplus (x_i * y_{n+1}) \oplus (x_{n+1} * y_i)\big)$

Canceling out the two applications of $b \oplus -$ yields the following:

- $\mathsf{DiagRefl}(i) := b \leftarrow \mathsf{SendBit}(i, n+1, K);\ x_i \leftarrow \mathsf{Share}(i, k);\ y_i \leftarrow \mathsf{Share}(i, l);$
  $x_{n+1} \leftarrow \mathsf{Share}(n+1, k);\ y_{n+1} \leftarrow \mathsf{Share}(n+1, l);\ \mathsf{ret}\ (x_i * y_{n+1}) \oplus (x_{n+1} * y_i)$

We can drop the dependency on the unused channel

- $\mathsf{SendBit}(i, n+1, K) := x_i \leftarrow \mathsf{Share}(i, k);\ y_i \leftarrow \mathsf{Share}(i, l);\ \mathsf{samp\ flip}$

because it only reads from the channels $\mathsf{Share}(i, k)$ and $\mathsf{Share}(i, l)$, which $\mathsf{DiagRefl}(i)$ reads from as well:

- $\mathsf{DiagRefl}(i) := x_i \leftarrow \mathsf{Share}(i, k);\ y_i \leftarrow \mathsf{Share}(i, l);\ x_{n+1} \leftarrow \mathsf{Share}(n+1, k);$
  $y_{n+1} \leftarrow \mathsf{Share}(n+1, l);\ \mathsf{ret}\ (x_i * y_{n+1}) \oplus (x_{n+1} * y_i)$

But this is precisely the desired form of $\mathsf{DiagRefl}(i)$.

We now observe that we can express the inductive form of the channels

- $\begin{cases} \mathsf{RowCol}(0) := \mathsf{read}\ \mathsf{DiagRefl}(0) \\ \mathsf{RowCol}(i+1) := b_\Sigma \leftarrow \mathsf{RowCol}(i);\ b_{i+1} \leftarrow \mathsf{DiagRefl}(i+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i < n \end{cases}$

equivalently using the following closed form:

- $\mathsf{RowCol}(i) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(i, k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(i, l);\ x_{n+1} \leftarrow \mathsf{Share}(n+1, k);$
  $y_{n+1} \leftarrow \mathsf{Share}(n+1, l);\ \mathsf{ret}\ (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma)$ for $i \leqslant n$

In the base case $i = 0$, we substitute the channel

– $\mathsf{DiagRefl}(0) := x_0 \leftarrow \mathsf{Share}(0, k);\ y_0 \leftarrow \mathsf{Share}(0, l);\ x_{n+1} \leftarrow \mathsf{Share}(n+1, k);$
$y_{n+1} \leftarrow \mathsf{Share}(n+1, l);\ \mathsf{ret}\ (x_0 * y_{n+1}) \oplus (x_{n+1} * y_0)$

into the inductive form of the channel

– $\mathsf{RowCol}(0) := \mathsf{read}\ \mathsf{DiagRefl}(0)$

which yields the following:

– $\mathsf{RowCol}(0) := x_0 \leftarrow \mathsf{Share}(0, k);\ y_0 \leftarrow \mathsf{Share}(0, l);\ x_{n+1} \leftarrow \mathsf{Share}(n+1, k);$
$y_{n+1} \leftarrow \mathsf{Share}(n+1, l);\ \mathsf{ret}\ (x_0 * y_{n+1}) \oplus (x_{n+1} * y_0)$

But this is precisely what we get if we substitute the channels

– $\mathsf{Share}\text{-}\Sigma(0, k) := x_0 \leftarrow \mathsf{Share}(0, k);\ \mathsf{ret}\ x_0$
– $\mathsf{Share}\text{-}\Sigma(0, l) := y_0 \leftarrow \mathsf{Share}(0, l);\ \mathsf{ret}\ y_0$

into the closed form of the channel below:

– $\mathsf{RowCol}(0) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(0, k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(0, l);\ x_{n+1} \leftarrow \mathsf{Share}(n+1, k);$
$y_{n+1} \leftarrow \mathsf{Share}(n+1, l);\ \mathsf{ret}\ (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma)$

Hence the inductive form and the closed form of the channel $\mathsf{RowCol}(0)$ are equivalent.

In the inductive case $i + 1$, we substitute the channels

– $\mathsf{RowCol}(i) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(i, k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(i, l);\ x_{n+1} \leftarrow \mathsf{Share}(n+1, k);$
$y_{n+1} \leftarrow \mathsf{Share}(n+1, l);\ \mathsf{ret}\ (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma)$ *(inductive hypothesis)*
– $\mathsf{DiagRefl}(i+1) := x_{i+1} \leftarrow \mathsf{Share}(i+1, k);\ y_{i+1} \leftarrow \mathsf{Share}(i+1, l);$
$x_{n+1} \leftarrow \mathsf{Share}(n+1, k);\ y_{n+1} \leftarrow \mathsf{Share}(n+1, l);\ \mathsf{ret}\ (x_{i+1} * y_{n+1}) \oplus (x_{n+1} * y_{i+1})$

into the inductive form of the channel

– $\mathsf{RowCol}(i+1) := b_\Sigma \leftarrow \mathsf{RowCol}(i);\ b_{i+1} \leftarrow \mathsf{DiagRefl}(i+1);\ \mathsf{ret}\ b_\Sigma \oplus b_{i+1}$

which yields the following:

– $\mathsf{RowCol}(i+1) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(i, k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(i, l);\ x_{n+1} \leftarrow \mathsf{Share}(n+1, k);$
$y_{n+1} \leftarrow \mathsf{Share}(n+1, l);\ x_{i+1} \leftarrow \mathsf{Share}(i+1, k);\ y_{i+1} \leftarrow \mathsf{Share}(i+1, l);$
$\mathsf{ret}\ (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma) \oplus (x_{i+1} * y_{n+1}) \oplus (x_{n+1} * y_{i+1})$

After a slight rearrangement we get the following:

– $\mathsf{RowCol}(i+1) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(i, k);\ x_{i+1} \leftarrow \mathsf{Share}(i+1, k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(i, l);$
$y_{i+1} \leftarrow \mathsf{Share}(i+1, l);\ x_{n+1} \leftarrow \mathsf{Share}(n+1, k);\ y_{n+1} \leftarrow \mathsf{Share}(n+1, l);$
$\mathsf{ret}\ (x_\Sigma * y_{n+1}) \oplus (x_{i+1} * y_{n+1}) \oplus (x_{n+1} * y_\Sigma) \oplus (x_{n+1} * y_{i+1})$

The above is equivalent to the following:

– $\mathsf{RowCol}(i+1) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(i, k);\ x_{i+1} \leftarrow \mathsf{Share}(i+1, k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(i, l);$
$y_{i+1} \leftarrow \mathsf{Share}(i+1, l);\ x_{n+1} \leftarrow \mathsf{Share}(n+1, k);\ y_{n+1} \leftarrow \mathsf{Share}(n+1, l);$
$\mathsf{ret}\ \big((x_\Sigma \oplus x_{i+1}) * y_{n+1}\big) \oplus \big(x_{n+1} * (y_\Sigma \oplus y_{i+1})\big)$

But this is precisely what we get if we substitute the channels

– $\mathsf{Share}\text{-}\Sigma(i+1, k) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(i, k);\ x_{i+1} \leftarrow \mathsf{Share}(i+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x_{i+1}$
– $\mathsf{Share}\text{-}\Sigma(i+1, l) := y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(i, l);\ y_{i+1} \leftarrow \mathsf{Share}(i+1, l);\ \mathsf{ret}\ y_\Sigma \oplus y_{i+1}$

into the closed form of the channel below:

- $\mathsf{RowCol}(i+1) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(i+1,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(i+1,l);$
  $x_{n+1} \leftarrow \mathsf{Share}(n+1,k);\ y_{n+1} \leftarrow \mathsf{Share}(n+1,l);\ \mathsf{ret}\ (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma)$

Hence the inductive form and the closed form of the channel $\mathsf{RowCol}(i+1)$ are equivalent.

The channels $\mathsf{DiagRefl}(-)$ are now unused and can be discarded.

We now substitute the channels

- $\mathsf{Sqr}(n) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n,l);\ \mathsf{ret}\ x_\Sigma * y_\Sigma$ *(inductive hypothesis)*
- $\mathsf{RowCol}(n) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n,l);\ x_{n+1} \leftarrow \mathsf{Share}(n+1,k);$
  $y_{n+1} \leftarrow \mathsf{Share}(n+1,l);\ \mathsf{ret}\ (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma)$
- $\mathsf{Ctrb}(n+1,n+1,K) \coloneqq x_{n+1} \leftarrow \mathsf{Share}(n+1,k);\ y_{n+1} \leftarrow \mathsf{Share}(n+1,l);\ \mathsf{ret}\ x_{n+1} * y_{n+1}$

into the channel

- $\mathsf{Sqr}(n+1) \coloneqq b_{\mathsf{S}} \leftarrow \mathsf{Sqr}(n);\ b_{\mathsf{RC}} \leftarrow \mathsf{RowCol}(n);\ b \leftarrow \mathsf{Ctrb}(n+1,n+1,K);\ \mathsf{ret}\ b_{\mathsf{S}} \oplus b_{\mathsf{RC}} \oplus b$

which yields the following:

- $\mathsf{Sqr}(n+1) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n,l);\ x_{n+1} \leftarrow \mathsf{Share}(n+1,k);$
  $y_{n+1} \leftarrow \mathsf{Share}(n+1,l);\ \mathsf{ret}\ (x_\Sigma * y_\Sigma) \oplus (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma) \oplus (x_{n+1} * y_{n+1})$

The channels $\mathsf{RowCol}(-)$ are now unused and can be discarded.

After a slight rearrangement of the channel $\mathsf{Sqr}(n+1)$ we get the following:

- $\mathsf{Sqr}(n+1) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n,k);\ x_{n+1} \leftarrow \mathsf{Share}(n+1,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n,l);$
  $y_{n+1} \leftarrow \mathsf{Share}(n+1,l);\ \mathsf{ret}\ (x_\Sigma * y_\Sigma) \oplus (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma) \oplus (x_{n+1} * y_{n+1})$

The above is equivalent to the following:

- $\mathsf{Sqr}(n+1) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n,k);\ x_{n+1} \leftarrow \mathsf{Share}(n+1,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n,l);$
  $y_{n+1} \leftarrow \mathsf{Share}(n+1,l);\ \mathsf{ret}\ (x_\Sigma \oplus x_{n+1}) * (y_\Sigma \oplus y_{n+1})$

But this is precisely what we get if we substitute the channels

- $\mathsf{Share}\text{-}\Sigma(n+1,k) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n,k);\ x_{n+1} \leftarrow \mathsf{Share}(n+1,k);\ \mathsf{ret}\ x_\Sigma \oplus x_{n+1}$
- $\mathsf{Share}\text{-}\Sigma(n+1,l) \coloneqq y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n,l);\ y_{n+1} \leftarrow \mathsf{Share}(n+1,l);\ \mathsf{ret}\ y_\Sigma \oplus y_{n+1}$

into the closed form of the channel below:

- $\mathsf{Sqr}(n+1) \coloneqq x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n+1,k);\ y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(n+1,l);\ \mathsf{ret}\ x_\Sigma * y_\Sigma$

Hence the inductive form and the closed form of the channel $\mathsf{Sqr}(n+1)$ are equivalent.

The channels $\mathsf{Row}(-,-)$, $\mathsf{Col}(-,-)$ are now unused and can be discarded.

Finally, we substitute the channel

- $\mathsf{Sqr}(N+1) \coloneqq x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,k);\ y \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,l);\ \mathsf{ret}\ x * y$

into the channel

- $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq \mathsf{read}\ \mathsf{Sqr}(N+1)$

which yields the following:

- $\mathsf{Share}\text{-}\Sigma(N+1,K) \coloneqq x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,k);\ y \leftarrow \mathsf{Share}\text{-}\Sigma(N+1,l);\ \mathsf{ret}\ x * y$

But this is precisely the desired inductive form of the channel $\mathsf{Share}\text{-}\Sigma(N+1,K)$.

The channels $\mathsf{Sqr}(-,-)$ are now unused and can be discarded.

We have now shown that summing up the respective shares $\bigoplus_{i \leqslant N+1} x_i$ of each party on a given wire yields the actual value x carried by the wire. Currently the computation is performed by the channels $\mathsf{Share}\text{-}\Sigma(N+1,-)$ but we can extract it out into a separate protocol $\mathsf{Wires}(C,K)$ as defined in the ideal functionality. Specifically, we introduce new internal channels

- $\mathsf{Wire}(k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(N+1,k)$ for $k < K$

and observe that together with the protocol $\mathsf{Shares}(C,K)$ we can express them equivalently as the channels

- $\mathsf{Share}\text{-}\Sigma(N+1,k) := \mathsf{read}\ \mathsf{Wire}(k)$ for $k < K$

together with the aforementioned protocol $\mathsf{Wires}(C,K)$ and the following new form of the protocol $\mathsf{Shares}(C,K)$:

- $\mathsf{Shares}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{InShare}(n,p,i)$ for $n \leqslant N$
- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{Share}(n,k)$ for $n \leqslant N$
- $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) := x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n \oplus y_n$ for $n \leqslant N$
- $\mathsf{Shares}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit}(n,m,K) := x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{samp\ flip} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
  - $\mathsf{RcvdBit}(n,m,K) := b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);$ $x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}(n,n,K) := x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n * y_n \\ \quad \text{for } n \leqslant N+1 \end{cases}$

$$
- \begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,0,K) := \mathsf{read}\ \mathsf{Ctrb}(n,0,K) \\ \quad \text{for } n \leqslant N+1 \\ \mathsf{Ctrb}\text{-}\Sigma(n,m+1,K) := b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,K);\ b \leftarrow \mathsf{Ctrb}(n,m+1,K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \end{cases}
$$

– $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,N+1,K)$ for $n \leqslant N$

10.4.7 Eliminating Shares of Party $N+1$

We start by eliminating the channels $\mathsf{InShare}\text{-}\$(N+1,-,-)$, $\mathsf{InShare}(N+1,-,-)$ from the initial part of the protocol. If party n is semi-honest, then for any input $i < I_n$ the channels

- $\mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(N+1,n,i)$
- $\mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(N+1,n,i)$

read from the channel

- $\mathsf{InShare}\text{-}\$(N+1,n,i) := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ x_\Sigma \oplus x$

so we can substitute:

- $\mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ x_\Sigma \oplus x$
- $\mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ x_\Sigma \oplus x$

We thus get the following for channels $\mathsf{InShare}\text{-}\$(N+1,-,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$ and $\mathsf{SendInShare}(N+1,-,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

The leakages

- $\mathsf{RcvdInShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdInShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $n \leqslant N+1$ and $i < I_n$

are vacuous since party $N+1$ is by assumption honest. Finally, substituting the channels

- $\mathsf{InShare}\text{-}\$(N+1,n,i) := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ x_\Sigma \oplus x$ for $n \leqslant N+1$ and $i < I_n$

into the channels

- $\mathsf{InShare}(N+1,n,i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$(N+1,n,i)$ for $n \leqslant N+1$ and $i < I_n$

yields the following:

- $\mathsf{InShare}(N+1,n,i) := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ x_\Sigma \oplus x$ for $n \leqslant N+1$ and $i < I_n$

At this point, the internal channels $\mathsf{InShare}\text{-}\$(N+1,-,-)$ are unused and can be eliminated.
The leakages

- $\mathsf{InShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $n \leqslant N+1$ and $i < I_n$

are again vacuous since party $N+1$ is by assumption honest. The top-level internal channels $\mathsf{InShare}(N+1,-,-)$ are now unused by the rest of the protocol and can also be eliminated. The resulting version Init of the initial part of the real protocol is therefore as follows:

- $\begin{cases} \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i) \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$

- $\mathsf{InShare}\text{-}\$(m,n,i) := x \leftarrow \mathsf{In}(n,i);\ \mathsf{samp}\ \mathsf{flip}$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$

- $\begin{cases} \mathsf{InShare}\text{-}\$(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare}\text{-}\$(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InShare}\text{-}\$\text{-}\Sigma(0,n,i) := \mathsf{read}\ \mathsf{InShare}\text{-}\$(0,n,i) \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m+1,n,i) := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i);\ x_{m+1} \leftarrow \mathsf{InShare}\text{-}\$(m+1,n,i);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \text{ and } n \leqslant N+1 \text{ and } i < I_n \end{cases}$

- $\begin{cases} \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \text{read } \mathsf{InShare}\text{-}\$(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \text{read } \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$
- $\mathsf{RcvdInShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \text{read } \mathsf{RcvdInShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $n \leqslant N+1$ and $i < I_n$
- $\mathsf{InShare}(m,n,i) := \text{read } \mathsf{InShare}\text{-}\$(m,n,i)$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$
- $\begin{cases} \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \text{read } \mathsf{InShare}(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \text{read } \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$
- $\mathsf{InShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \text{read } \mathsf{InShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $n \leqslant N+1$ and $i < I_n$

This is followed by the hiding of the channels

- $\mathsf{InShare}\text{-}\$(m,n,i)$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$,
- $\mathsf{InShare}\text{-}\$\text{-}\Sigma(m,n,i)$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$.

We now eliminate the channels $\mathsf{SendBit}(N+1,-,-)$, $\mathsf{RcvdBit}(N+1,-,-)$, $\mathsf{Ctrb}(N+1,-,-)$, $\mathsf{Ctrb}\text{-}\Sigma(N+1,-,-)$, $\mathsf{Share}(N+1,-)$ from the inductive part of the real protocol. To this end, we first revisit the protocol $\mathsf{Adv}(C,K)$. In the case of *input*-, *not*-, and *xor* gates, the leakage

- $\mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \text{read } \mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$

is vacuous since party $N+1$ is by assumption honest. In the case of an *and* gate, the leakages

- $\mathsf{SendBit}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \text{read } \mathsf{SendBit}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\mathsf{RcvdBit}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \text{read } \mathsf{RcvdBit}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\mathsf{Ctrb}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \text{read } \mathsf{Ctrb}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \text{read } \mathsf{Ctrb}\text{-}\Sigma(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \text{read } \mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$

are likewise vacuous since party $N+1$ is by assumption honest, and so are the leakages below:

- $\mathsf{OTMsg}_0(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \text{read } \mathsf{OTMsg}_0(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\mathsf{OTMsg}_1(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \text{read } \mathsf{OTMsg}_1(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\mathsf{OTMsg}_2(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \text{read } \mathsf{OTMsg}_2(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\mathsf{OTMsg}_2(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \text{read } \mathsf{OTMsg}_2(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

On the other hand, the leakages

- $\mathsf{OTMsgRcvd}_0(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \text{read } \mathsf{OTMsgRcvd}_0(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_1(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \text{read } \mathsf{OTMsgRcvd}_1(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_2(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \text{read } \mathsf{OTMsgRcvd}_2(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_3(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \text{read } \mathsf{OTMsgRcvd}_3(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

are vacuous because $N+1 \geqslant m$ for each party m, so party $N+1$ cannot ever function as a sender. The leakages

- $\mathsf{OTChc}_0(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n \leqslant N+1$
- $\mathsf{OTChc}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n \leqslant N+1$

are again vacuous because party $N+1$ is honest, and so are the leakages below:

- $\mathsf{OTOut}(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n \leqslant N+1$

The latest version of the protocol $\mathsf{Adv}(C, K)$ is therefore as follows:

- $\mathsf{Adv}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Adv}\big(C;\ \textit{input-gate}(p, i), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{RcvdBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n, K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n, K) \\ \quad \text{for } n \leqslant N \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n, K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n, K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{Share}(N+1, K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(N+1, K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_2(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_3(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTChc}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTChc}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
- $\mathsf{Adv}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Adv}(C, K)$ with the protocol
  - $\mathsf{SendBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{RcvdBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n, m, K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n, m \leqslant N+1$

- $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$
- $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Adv}\big(C;\, \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

• $\mathsf{Adv}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Adv}(C,K)$ with the protocol

- $\begin{cases} \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{SendBit}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\begin{cases} \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{RcvdBit}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\begin{cases} \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K) \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{Ctrb}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K) \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{Ctrb}\text{-}\Sigma(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$
- $\begin{cases} \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$
- $\mathsf{OTMsg}_0(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\begin{cases} \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\mathsf{OTMsg}_1(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus y_n \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\mathsf{OTMsg}_2(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \oplus y_n \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\mathsf{OTMsg}_3(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\mathsf{OTMsgRcvd}_0(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\mathsf{OTMsgRcvd}_1(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\mathsf{OTMsgRcvd}_2(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\mathsf{OTMsgRcvd}_3(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest or } n \geqslant m \end{cases}$

– $\mathsf{OTChc}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n \leqslant N+1$

– $\begin{cases} \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,l) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest or } n \geqslant m \end{cases}$

- $\mathsf{OTChc}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n \leqslant N+1$
- $\begin{cases} \mathsf{OTChcRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := x_m \leftarrow \mathsf{Share}(m, k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_0(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_0(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := y_m \leftarrow \mathsf{Share}(m, l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_1(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_1(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(N+1, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$
- $\begin{cases} \mathsf{OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(m, n, K) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest} \\ \mathsf{OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n, m, K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest} \end{cases}$
- $\mathsf{OTOut}(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n, N+1, K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n \leqslant N+1$

We now revisit the protocol $\mathsf{Shares}(C, K)$, the case of an *and* gate. For any party n the channel

- $\mathsf{RcvdBit}(n, N+1, K) := b \leftarrow \mathsf{SendBit}(N+1, n, K);\ x_{N+1} \leftarrow \mathsf{Share}(N+1, k);$ $y_{N+1} \leftarrow \mathsf{Share}(N+1, l);\ x_n \leftarrow \mathsf{Share}(n, k);\ y_n \leftarrow \mathsf{Share}(n, l);\ \mathsf{ret}\ b \oplus (x_{N+1} * y_n) \oplus (x_n * y_{N+1})$

reads from the divergent channel

- $\mathsf{SendBit}(N+1, n, K) := \mathsf{read}\ \mathsf{SendBit}(N+1, n, K)$

and so we may equivalently write the following:

- $\mathsf{RcvdBit}(n, N+1, K) := \mathsf{read}\ \mathsf{RcvdBit}(n, N+1, K)$

The simplified definition for channels $\mathsf{RcvdBit}(-, N+1, K)$ is thus as follows:

- $\mathsf{RcvdBit}(n, N+1, K) := \mathsf{read}\ \mathsf{RcvdBit}(n, N+1, K)$ for $n \leqslant N+1$

The latest version of the protocol $\mathsf{Shares}(C, K)$ is therefore as follows:

- $\mathsf{Shares}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(p, i), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(n, m, K) := \mathsf{read}\ \mathsf{SendBit}(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{RcvdBit}(n, m, K) := \mathsf{read}\ \mathsf{RcvdBit}(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}(n, m, K) := \mathsf{read}\ \mathsf{Ctrb}(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n, m, K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n, m, K)$ for $n, m \leqslant N+1$
  - $\mathsf{Share}(n, K) := \mathsf{read}\ \mathsf{InShare}(n, p, i)$ for $n \leqslant N$
- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol

- $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{Share}(n,k)$ for $n \leqslant N$

• $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol

- $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n,m \leqslant N+1$
- $\mathsf{Share}(n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n \oplus y_n$ for $n \leqslant N$

• $\mathsf{Shares}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C,K)$ with the protocol

- $\begin{cases} \mathsf{SendBit}(n,m,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{samp\ flip} \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
- $\mathsf{RcvdBit}(n,m,K) \coloneqq b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);$
$x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n \leqslant N+1$ and $m \leqslant N$
- $\mathsf{RcvdBit}(n,N+1,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,N+1,K)$ for $n \leqslant N+1$
- $\begin{cases} \mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n,m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}(n,n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n * y_n \\ \quad \text{for } n \leqslant N+1 \end{cases}$
- $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,0,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,0,K) \\ \quad \text{for } n \leqslant N+1 \\ \mathsf{Ctrb}\text{-}\Sigma(n,m+1,K) \coloneqq b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,K);\ b \leftarrow \mathsf{Ctrb}(n,m+1,K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \end{cases}$
- $\mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,N+1,K)$ for $n \leqslant N$

At this point, we can extract all computation carried out by party $N+1$ into a separate protocol $\mathsf{Ctrbs}_{N+1}(C,K)$:

• $\mathsf{Ctrbs}_{N+1}(\epsilon,0)$ is the protocol $\mathsf{0}$

• $\mathsf{Ctrb}_{N+1}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Ctrbs}_{N+1}(C,K)$ with the protocol

- $\mathsf{SendBit}(N+1,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(N+1,m,K)$ for $m \leqslant N+1$
- $\mathsf{RcvdBit}(N+1,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(N+1,m,K)$ for $m \leqslant N+1$
- $\mathsf{Ctrb}(N+1,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(N+1,m,K)$ for $m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(N+1,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(N+1,m,K)$ for $m \leqslant N+1$

• $\mathsf{Ctrbs}_{N+1}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Ctrbs}_{N+1}(C,K)$ with the protocol

- $\mathsf{SendBit}(N+1,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(N+1,m,K)$ for $m \leqslant N+1$

- $\mathsf{RcvdBit}(N+1, m, K) := \mathsf{read}\ \mathsf{RcvdBit}(N+1, m, K)$ for $m \leqslant N+1$
- $\mathsf{Ctrb}(N+1, m, K) := \mathsf{read}\ \mathsf{Ctrb}(N+1, m, K)$ for $m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(N+1, m, K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(N+1, m, K)$ for $m \leqslant N+1$

- $\mathsf{Ctrbs}_{N+1}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Ctrbs}_{N+1}(C, K)$ with the protocol
  - $\mathsf{SendBit}(N+1, m, K) := \mathsf{read}\ \mathsf{SendBit}(N+1, m, K)$ for $m \leqslant N+1$
  - $\mathsf{RcvdBit}(N+1, m, K) := \mathsf{read}\ \mathsf{RcvdBit}(N+1, m, K)$ for $m \leqslant N+1$
  - $\mathsf{Ctrb}(N+1, m, K) := \mathsf{read}\ \mathsf{Ctrb}(N+1, m, K)$ for $m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(N+1, m, K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(N+1, m, K)$ for $m \leqslant N+1$

- $\mathsf{Ctrb}_{N+1}\big(C;\ \textit{and-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Ctrbs}_{N+1}(C, K)$ with the protocol
  - $\mathsf{SendBit}(N+1, m, K) := \mathsf{read}\ \mathsf{SendBit}(N+1, m, K)$ for $m \leqslant N+1$
  - $\mathsf{RcvdBit}(N+1, m, K) := b \leftarrow \mathsf{SendBit}(m, N+1, K);\ x_m \leftarrow \mathsf{Share}(m, k);\ y_m \leftarrow \mathsf{Share}(m, l);$ $x_{N+1} \leftarrow \mathsf{Share}(N+1, k);\ y_{N+1} \leftarrow \mathsf{Share}(N+1, l);\ \mathsf{ret}\ b \oplus (x_m * y_{N+1}) \oplus (x_{N+1} * y_m)$ for $m \leqslant N$
  - $\mathsf{RcvdBit}(N+1, N+1, K) := \mathsf{read}\ \mathsf{RcvdBit}(N+1, N+1, K)$
  - $\begin{cases} \mathsf{Ctrb}(N+1, m, K) := \mathsf{read}\ \mathsf{RcvdBit}(N+1, m, K) \\ \quad \text{for } m \leqslant N \\ \mathsf{Ctrb}(N+1, N+1, K) := x_{N+1} \leftarrow \mathsf{Share}(N+1, k);\ y_{N+1} \leftarrow \mathsf{Share}(N+1, l);\ \mathsf{ret}\ x_{N+1} * y_{N+1} \end{cases}$
  - $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(N+1, 0, K) := \mathsf{read}\ \mathsf{Ctrb}(N+1, 0, K) \\ \mathsf{Ctrb}\text{-}\Sigma(N+1, m+1, K) := b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(N+1, m, K);\ b \leftarrow \mathsf{Ctrb}(N+1, m+1, K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } m \leqslant N \end{cases}$

After the extraction, the protocol $\mathsf{Shares}(C, K)$ is left looking as follows:

- $\mathsf{Shares}(\epsilon, 0)$ is the protocol $\mathsf{0}$
- $\mathsf{Shares}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(n, m, K) := \mathsf{read}\ \mathsf{SendBit}(n, m, K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{RcvdBit}(n, m, K) := \mathsf{read}\ \mathsf{RcvdBit}(n, m, K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{Ctrb}(n, m, K) := \mathsf{read}\ \mathsf{Ctrb}(n, m, K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n, m, K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n, m, K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{Share}(n, K) := \mathsf{read}\ \mathsf{InShare}(n, p, i)$ for $n \leqslant N$
- $\mathsf{Shares}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(n, m, K) := \mathsf{read}\ \mathsf{SendBit}(n, m, K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{RcvdBit}(n, m, K) := \mathsf{read}\ \mathsf{RcvdBit}(n, m, K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{Ctrb}(n, m, K) := \mathsf{read}\ \mathsf{Ctrb}(n, m, K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n, m, K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n, m, K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{Share}(n, K) := \mathsf{read}\ \mathsf{Share}(n, k)$ for $n \leqslant N$
- $\mathsf{Shares}\big(C;\ \textit{xor-gate}(k,l), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol
  - $\mathsf{SendBit}(n, m, K) := \mathsf{read}\ \mathsf{SendBit}(n, m, K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{RcvdBit}(n, m, K) := \mathsf{read}\ \mathsf{RcvdBit}(n, m, K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{Ctrb}(n, m, K) := \mathsf{read}\ \mathsf{Ctrb}(n, m, K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n, m, K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n, m, K)$ for $n \leqslant N$ and $m \leqslant N+1$

- $\mathsf{Share}(n, K) \coloneqq x_n \leftarrow \mathsf{Share}(n, k);\ y_n \leftarrow \mathsf{Share}(n, l);\ \mathsf{ret}\ x_n \oplus y_n$ for $n \leqslant N$

- $\mathsf{Shares}\big(C;\ \mathit{and\text{-}gate}(k, l), K+1\big)$ is the composition of $\mathsf{Shares}(C, K)$ with the protocol

  - $\begin{cases} \mathsf{SendBit}(n, m, K) \coloneqq x_n \leftarrow \mathsf{Share}(n, k);\ y_n \leftarrow \mathsf{Share}(n, l);\ \mathsf{samp\ flip} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n, m, K) \coloneqq \mathsf{read\ SendBit}(n, m, K) \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$

  - $\mathsf{RcvdBit}(n, m, K) \coloneqq b \leftarrow \mathsf{SendBit}(m, n, K);\ x_m \leftarrow \mathsf{Share}(m, k);\ y_m \leftarrow \mathsf{Share}(m, l);$ $x_n \leftarrow \mathsf{Share}(n, k);\ y_n \leftarrow \mathsf{Share}(n, l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n, m \leqslant N$

  - $\mathsf{RcvdBit}(n, N+1, K) \coloneqq \mathsf{read\ RcvdBit}(n, N+1, K)$ for $n \leqslant N$

  - $\begin{cases} \mathsf{Ctrb}(n, m, K) \coloneqq \mathsf{read\ SendBit}(n, m, K) \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}(n, m, K) \coloneqq \mathsf{read\ RcvdBit}(n, m, K) \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}(n, n, K) \coloneqq x_n \leftarrow \mathsf{Share}(n, k);\ y_n \leftarrow \mathsf{Share}(n, l);\ \mathsf{ret}\ x_n * y_n \\ \quad \text{for } n \leqslant N \end{cases}$

  - $\begin{cases} \mathsf{Ctrb\text{-}\Sigma}(n, 0, K) \coloneqq \mathsf{read\ Ctrb}(n, 0, K) \\ \quad \text{for } n \leqslant N \\ \mathsf{Ctrb\text{-}\Sigma}(n, m+1, K) \coloneqq b_\Sigma \leftarrow \mathsf{Ctrb\text{-}\Sigma}(n, m, K);\ b \leftarrow \mathsf{Ctrb}(n, m+1, K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } n, m \leqslant N \end{cases}$

  - $\mathsf{Share}(n, K) \coloneqq \mathsf{read\ Ctrb\text{-}\Sigma}(n, N+1, K)$ for $n \leqslant N$

None of the channels defined by $\mathsf{Ctrb}_{N+1}(C, K)$ are utilized anywhere outside of $\mathsf{Ctrb}_{N+1}(C, K)$ and as such we may discard this protocol fragment entirely. This in particular eliminates all references to the channels $\mathsf{Share}(N+1, -)$ from the inductive part of the protocol. To summarize, the inductive part of the real protocol now consists of the protocols $\mathsf{Shares}(C, K)$ and $\mathsf{Adv}(C, K)$, followed by the hiding of the channels

- $\mathsf{SendBit}(n, m, k)$ for $n \leqslant N$ and $m \leqslant N+1$ and $k < K$,

- $\mathsf{RcvdBit}(n, m, k)$ for $n \leqslant N$ and $m \leqslant N+1$ and $k < K$,

- $\mathsf{Ctrb}(n, m, k)$ for $n \leqslant N$ and $m \leqslant N+1$ and $k < K$,

- $\mathsf{Ctrb\text{-}\Sigma}(n, m, k)$ for $n \leqslant N$ and $m \leqslant N+1$ and $k < K$.

We recall that on the top level we also have the protocol $\mathsf{Wires}(C, K)$ and the channels

- $\begin{cases} \mathsf{Share\text{-}\Sigma}(0, k) \coloneqq \mathsf{read\ Share}(0, k) \\ \quad \text{for } k < K \\ \mathsf{Share\text{-}\Sigma}(m+1, k) \coloneqq x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(m, k);\ x_{m+1} \leftarrow \mathsf{Share}(m+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \text{ and } k < K \end{cases}$

together with the channels below:

- $\mathsf{Share\text{-}\Sigma}(N+1, k) \coloneqq \mathsf{read\ Wire}(k)$ for $k < K$

- $\mathsf{Share}(N+1, k) \coloneqq x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(N, k);\ x \leftarrow \mathsf{Share\text{-}\Sigma}(N+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x$ for $k < K$

We now eliminate all references to the shares of party $N+1$ from the final part of the protocol. The leakages

- $\mathsf{SendOutShare}(m, N+1, k)^{\mathsf{party}(N+1)}_{\mathsf{adv}} \coloneqq \mathsf{read\ SendOutShare}(m, N+1, k)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$

are vacuous since party $N+1$ is honest. If party n is semi-honest and wire k is an output, then the channels

- $\mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(N+1, k)$
- $\mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(N+1, k)$

read from the channel

- $\mathsf{Share}(N+1, k) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x$

so we may substitute:

- $\mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x$
- $\mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x$

We thus get the following for channels $\mathsf{RcvdOutShare}(-, N+1, -)^{\mathsf{party}(-)}_{\mathsf{adv}}$ and $\mathsf{OutShare}(-, N+1, -)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

The top-level internal channels $\mathsf{Share}(N+1, -)$ are now unused by the rest of the protocol and can be eliminated. The resulting version Fin of the final part of the real protocol is as follows:

- $\begin{cases} \mathsf{SendOutShare}(m, n, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n, k) \\ \quad \text{for } m \leqslant N+1 \text{ and } n \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{SendOutShare}(m, n, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m, n, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ and } n \leqslant N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\mathsf{SendOutShare}(m, N+1, k)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m, N+1, k)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\begin{cases} \mathsf{RcvdOutShare}(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m, k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m, k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(m, k) \\ \quad \text{for } n, m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n, m \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{Out}(n, k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(N+1, k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } k \text{ an output} \\ \mathsf{Out}(n, k) := \mathsf{read}\ \mathsf{Out}(n, k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{Out}(n, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n, k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{Out}(n, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

As a final step before the extraction of the simulator, we eliminate any reference to the channels $\mathsf{Share}\text{-}\Sigma(N+1, -)$ from the final part of the real protocol. If party n is semi-honest and wire k is an output, then we can substitute the channel

- $\mathsf{Share}\text{-}\Sigma(N+1, k) := \mathsf{read}\ \mathsf{Wire}(k)$

into the channels

- $\mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x$
- $\mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Share}\text{-}\Sigma(N+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x$

which yields the following:

- $\mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_\Sigma \oplus x$
- $\mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_\Sigma \oplus x$

We thus get the following for channels $\mathsf{RcvdOutShare}(-, N+1, -)^{\mathsf{party}(-)}_{\mathsf{adv}}$ and $\mathsf{OutShare}(-, N+1, -)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

If party n is semi-honest and wire k is an output, then the channel

- $\mathsf{OutShare}\text{-}\Sigma(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(N+1, k)$

reads from the channel

- $\mathsf{Share}\text{-}\Sigma(N+1,k) := \mathsf{read}\ \mathsf{Wire}(k)$

so we may substitute:

- $\mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}(k)$

We thus get the following for channels $\mathsf{OutShare}\text{-}\Sigma(-,N+1,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

Finally, if wire k is an output, then for any party n the channel

- $\mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(N+1,k)$

reads from the channel

- $\mathsf{Share}\text{-}\Sigma(N+1,k) := \mathsf{read}\ \mathsf{Wire}(k)$

so we may substitute:

- $\mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{Wire}(k)$

We thus get the following for channels $\mathsf{Out}(-,-)$:

- $\begin{cases} \mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } k \text{ an output} \\ \mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } k \text{ not an output} \end{cases}$

The top-level internal channels $\mathsf{Share}\text{-}\Sigma(N+1,-)$ are now unused by the rest of the protocol and can be eliminated. The resulting version Fin of the final part of the real protocol is as follows:

- $\begin{cases} \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } m \leqslant N+1 \text{ and } n \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ and } n \leqslant N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

- $\mathsf{SendOutShare}(m,N+1,k)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,N+1,k)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$

- $\begin{cases} \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } k \text{ an output} \\ \mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

10.4.8 Extracting The Simulator

We are now ready to extract the simulator. The internal protocol $\mathsf{Wires}(C,K)$ together with the output channels

- $\begin{cases} \mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } k \text{ an output} \\ \mathsf{Out}(n,k) := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } k \text{ not an output} \end{cases}$

will be factored out as coming from the ideal functionality. In particular, this leaves us with following version Fin of the final part of the soon-to-be simulator:

- $\begin{cases} \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } m \leqslant N+1 \text{ and } n \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ and } n \leqslant N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

- $\mathsf{SendOutShare}(m,N+1,k)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,N+1,k)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$

- $\begin{cases} \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Wire}(k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

In the remainder of the soon-to-be simulator, we must eliminate any references to the channels $\mathsf{In}(-,-)$, $\mathsf{Wire}(-)$, $\mathsf{Out}(-,-)$. We begin with the initial part of the real protocol. Recall the leakage

- $\begin{cases} \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i) \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$

from the ideal functionality. If party n is semi-honest, then for any input $i < I_n$ the channel

- $\mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)$

can be expressed equivalently as

- $\mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}}$

since we have the leakage below:

- $\mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)$

We thus get the following for channels $\mathsf{In}(-,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

If party n is honest, then for any input $i < I_n$ the channel

- $\mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ \checkmark$

can be expressed equivalently as

- $\mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}}$

since we have the leakage below:

- $\mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ \checkmark$

We thus get the following for channels $\mathsf{InRcvd}(-,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$

If party n is semi-honest, then for any party m and any input $i < I_n$ the channel

- $\mathsf{InShare\text{-}\$}(m,n,i) := x \leftarrow \mathsf{In}(n,i);\ \mathsf{samp\ flip}$

can be expressed equivalently as

- $\mathsf{InShare\text{-}\$}(m,n,i) := x \leftarrow \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{samp\ flip}$

since we have the leakage below:

- $\mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)$

If party n is honest, then for any party m and any input $i < I_n$ the channel

- $\mathsf{InShare\text{-}\$}(m,n,i) := x \leftarrow \mathsf{In}(n,i);\ \mathsf{samp\ flip}$

can be expressed equivalently as

- $\mathsf{InShare\text{-}\$}(m,n,i) := \_ \leftarrow \mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{samp\ flip}$

since we have the leakage below:

- $\mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}} := x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ \checkmark$

We thus get the following for channels $\mathsf{InShare}\text{-}\$(-,-,-)$:

- $\begin{cases} \mathsf{InShare}\text{-}\$(m,n,i) := x \leftarrow \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{samp\ flip} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare}\text{-}\$(m,n,i) := \_ \leftarrow \mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{samp\ flip} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

If party n is semi-honest, then for any input $i < I_n$ the channels

- $\mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ x_\Sigma \oplus x$
- $\mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i);\ \mathsf{ret}\ x_\Sigma \oplus x$

can be expressed equivalently as

- $\mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x$
- $\mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x$

since we have the leakage below:

- $\mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)$

We thus get the following for channels $\mathsf{InShare}\text{-}\$(N+1,-,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$ and $\mathsf{SendInShare}(N+1,-,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}\text{-}\$(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare}\text{-}\$\text{-}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

The final version Init of the initial part of the simulator is therefore as follows:

- $\begin{cases} \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \mathsf{InShare}\text{-}\$(m,n,i) := x \leftarrow \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{samp\ flip} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare}\text{-}\$(m,n,i) := \_ \leftarrow \mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{samp\ flip} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InShare\text{-}\$}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare\text{-}\$}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InShare\text{-}\$\text{-}}\Sigma(0,n,i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(0,n,i) \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m+1,n,i) := x_\Sigma \leftarrow \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i);\ x_{m+1} \leftarrow \mathsf{InShare\text{-}\$}(m+1,n,i);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \text{ and } n \leqslant N+1 \text{ and } i < I_n \end{cases}$

- $\begin{cases} \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InShare\text{-}\$}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare\text{-}\$\text{-}}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare\text{-}\$}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare\text{-}\$\text{-}}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$

- $\mathsf{RcvdInShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdInShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $n \leqslant N+1$ and $i < I_n$

- $\mathsf{InShare}(m,n,i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i)$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$

- $\begin{cases} \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$

- $\mathsf{InShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $n \leqslant N+1$ and $i < I_n$

This is followed by the hiding of the channels

- $\mathsf{InShare\text{-}\$}(m,n,i)$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$,

- $\mathsf{InShare\text{-}\$\text{-}\Sigma}(m,n,i)$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$.

We continue with the final part of the soon-to-be simulator. Recall the definition of the output channels

- $\begin{cases} \mathsf{Out}(n,k) := \mathsf{read\ Wire}(k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } k \text{ an output} \\ \mathsf{Out}(n,k) := \mathsf{read\ Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } k \text{ not an output} \end{cases}$

in the ideal functionality. If party n is semi-honest and wire k is an output, then the channels

- $\mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(N,k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_\Sigma \oplus x$
- $\mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(N,k);\ x \leftarrow \mathsf{Wire}(k);\ \mathsf{ret}\ x_\Sigma \oplus x$

can be expressed equivalently as

- $\mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(N,k);\ x \leftarrow \mathsf{Out}(n,k);\ \mathsf{ret}\ x_\Sigma \oplus x$
- $\mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(N,k);\ x \leftarrow \mathsf{Out}(n,k);\ \mathsf{ret}\ x_\Sigma \oplus x$

since we have the definition below:

- $\mathsf{Out}(n,k) := \mathsf{read\ Wire}(k)$

We thus get the following for channels $\mathsf{RcvdOutShare}(-,N+1,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$ and $\mathsf{OutShare}(-,N+1,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(N,k);\ x \leftarrow \mathsf{Out}(n,k);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read\ RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share\text{-}\Sigma}(N,k);\ x \leftarrow \mathsf{Out}(n,k);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read\ OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

If party n is semi-honest and wire k is an output, then the channel

- $\mathsf{OutShare\text{-}\Sigma}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read\ Wire}(k)$

can be expressed equivalently as

- $\mathsf{OutShare\text{-}\Sigma}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read\ Out}(n,k)$

since we have the definition below:

- $\mathsf{Out}(n,k) := \mathsf{read\ Wire}(k)$

We thus get the following for channels $\mathsf{OutShare\text{-}\Sigma}(-,N+1,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OutShare\text{-}\Sigma}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read\ Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare\text{-}\Sigma}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read\ OutShare\text{-}\Sigma}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

The latest version Fin of the final part of the soon-to-be simulator is therefore as follows:

- $\begin{cases} \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } m \leqslant N+1 \text{ and } n \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ and } n \leqslant N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\mathsf{SendOutShare}(m,N+1,k)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,N+1,k)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\begin{cases} \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x \leftarrow \mathsf{Out}(n,k);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x \leftarrow \mathsf{Out}(n,k);\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

We now recall the leakage

- $\begin{cases} \mathsf{Out}(n,k)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{Out}(n,k)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

from the ideal functionality. If party n is semi-honest and wire k is an output, then the channels

- $\mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Out}(n, k);\ \mathsf{ret}\ x_\Sigma \oplus x$
- $\mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Out}(n, k);\ \mathsf{ret}\ x_\Sigma \oplus x$

can be expressed equivalently as

- $\mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x$
- $\mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x$

since we have the leakage below:

- $\mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n, k)$

We thus get the following for channels $\mathsf{RcvdOutShare}(-, N+1, -)^{\mathsf{party}(-)}_{\mathsf{adv}}$ and $\mathsf{OutShare}(-, N+1, -)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

If party n is semi-honest and wire k is an output, then the channel

- $\mathsf{OutShare}\text{-}\Sigma(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n, k)$

can be expressed equivalently as

- $\mathsf{OutShare}\text{-}\Sigma(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}}$

since we have the leakage below:

- $\mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n, k)$

We thus get the following for channels $\mathsf{OutShare}\text{-}\Sigma(-, N+1, -)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

Finally, if party n is semi-honest, then for any wire k the channel

- $\mathsf{Out}(n, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n, k)$

can be expressed equivalently as

- $\mathsf{Out}(n, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}}$

since we have the leakage below:

- $\mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n, k)$

We thus get the following for channels $\mathsf{Out}(-,-)^{\mathsf{party}(-)}_{\mathsf{adv}}$:

- $\begin{cases} \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

The final version Fin of the final part of the simulator is therefore as follows:

- $\begin{cases} \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } m \leqslant N+1 \text{ and } n \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ and } n \leqslant N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

- $\mathsf{SendOutShare}(m,N+1,k)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,N+1,k)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$

- $\begin{cases} \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x \leftarrow \mathsf{Out}(n,k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ x \leftarrow \mathsf{Out}(n,k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n,N+1,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

10.4.9 The Simulator

The simulator consists of four parts. In the initial phase, we have the protocol Init:

- $\begin{cases} \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{In}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InRcvd}(n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$

- $\begin{cases} \mathsf{InShare\text{-}\$}(m,n,i) := x \leftarrow \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{samp\ flip} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare\text{-}\$}(m,n,i) := \_ \leftarrow \mathsf{InRcvd}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{samp\ flip} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InShare\text{-}\$}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare\text{-}\$}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InShare\text{-}\$\text{-}}\Sigma(0,n,i) := \mathsf{read}\ \mathsf{InShare\text{-}\$}(0,n,i) \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m+1,n,i) := x_\Sigma \leftarrow \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i);\ x_{m+1} \leftarrow \mathsf{InShare\text{-}\$}(m+1,n,i);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \text{ and } n \leqslant N+1 \text{ and } i < I_n \end{cases}$

- $\begin{cases} \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{InShare\text{-}\$}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{InShare\text{-}\$\text{-}}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{InShare\text{-}\$}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendInShare}(m,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq x_\Sigma \leftarrow \mathsf{InShare\text{-}\$\text{-}}\Sigma(N,n,i);\ x \leftarrow \mathsf{In}(n,i)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendInShare}(N+1,n,i)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{RcvdInShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$

- $\mathsf{RcvdInShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{RcvdInShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $n \leqslant N+1$ and $i < I_n$

- $\mathsf{InShare}(m,n,i) \coloneqq \mathsf{read}\ \mathsf{InShare\text{-}\$}(m,n,i)$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$

- $\begin{cases} \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{InShare}(m,n,i) \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{InShare}(m,n,i)^{\mathsf{party}(m)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N \text{ and } n \leqslant N+1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$

- $\mathsf{InShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{InShare}(N+1,n,i)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $n \leqslant N+1$ and $i < I_n$

This is followed by the hiding of the channels

- $\mathsf{InShare\text{-}\$}(m,n,i)$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$,
- $\mathsf{InShare\text{-}\$\text{-}}\Sigma(m,n,i)$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$.

In the inductive phase, we have the protocol $\mathsf{Circ}(C,K)$, obtained by merging the two protocols $\mathsf{Shares}(C,K)$ and $\mathsf{Adv}(C,K)$ from earlier:

- $\mathsf{Circ}(\epsilon,0)$ is the protocol $\mathsf{0}$
- $\mathsf{Circ}\big(C;\ \textit{input-gate}(p,i), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb\text{-}}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb\text{-}}\Sigma(n,m,K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{Ctrb\text{-}}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb\text{-}}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) \coloneqq \mathsf{read}\ \mathsf{InShare}(n,p,i)$ for $n \leqslant N$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$

- $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

• $\mathsf{Circ}\big(C;\ \textit{not-gate}(k), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol

- $\mathsf{SendBit}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n \leqslant N$ and $m \leqslant N+1$
- $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{RcvdBit}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n \leqslant N$ and $m \leqslant N+1$
- $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n \leqslant N$ and $m \leqslant N+1$
- $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n \leqslant N$ and $m \leqslant N+1$
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{Share}(n,k)$ for $n \leqslant N$
- $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$
- $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
- $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Circ}\big(C;\ \mathit{xor\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K) \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)$ for $n \leqslant N$ and $m \leqslant N+1$
  - $\mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{Share}(n,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n \oplus y_n$ for $n \leqslant N$
  - $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ if } n \text{ honest} \end{cases}$
  - $\mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$
  - $\mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$
  - $\mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n,m \leqslant N+1$

- $\mathsf{Circ}\big(C;\ \mathit{and\text{-}gate}(k,l), K+1\big)$ is the composition of $\mathsf{Circ}(C,K)$ with the protocol
  - $\begin{cases} \mathsf{SendBit}(n,m,K) \coloneqq x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{samp}\ \mathsf{flip} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n < m \\ \mathsf{SendBit}(n,m,K) \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \geqslant m \end{cases}$
  - $\begin{cases} \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \coloneqq \mathsf{read}\ \mathsf{SendBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$

- $\mathsf{SendBit}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendBit}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\mathsf{RcvdBit}(n,m,K) := b \leftarrow \mathsf{SendBit}(m,n,K);\ x_m \leftarrow \mathsf{Share}(m,k);\ y_m \leftarrow \mathsf{Share}(m,l);$ $x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n, m \leqslant N$
- $\mathsf{RcvdBit}(n,N+1,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,N+1,K)$ for $n \leqslant N$
- $\begin{cases} \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{RcvdBit}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\begin{cases} \mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{SendBit}(n,m,K) \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n < m \\ \mathsf{Ctrb}(n,m,K) := \mathsf{read}\ \mathsf{RcvdBit}(n,m,K) \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } m < n \\ \mathsf{Ctrb}(n,n,K) := x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ x_n * y_n \\ \quad \text{for } n \leqslant N \end{cases}$
- $\begin{cases} \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K) \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{Ctrb}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,0,K) := \mathsf{read}\ \mathsf{Ctrb}(n,0,K) \\ \quad \text{for } n \leqslant N \\ \mathsf{Ctrb}\text{-}\Sigma(n,m+1,K) := b_\Sigma \leftarrow \mathsf{Ctrb}\text{-}\Sigma(n,m,K);\ b \leftarrow \mathsf{Ctrb}(n,m+1,K);\ \mathsf{ret}\ b_\Sigma \oplus b \\ \quad \text{for } n, m \leqslant N \end{cases}$
- $\begin{cases} \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K) \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest} \\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,m,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{Ctrb}\text{-}\Sigma(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(N+1,m,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\mathsf{Share}(n,K) := \mathsf{read}\ \mathsf{Ctrb}\text{-}\Sigma(n,N+1,K)$ for $n \leqslant N$
- $\begin{cases} \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K) \\ \quad \text{for } n \leqslant N \text{ if } n \text{ semi-honest} \\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,K)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ if } n \text{ honest} \end{cases}$
- $\mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(N+1,K)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$
- $\begin{cases} \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$
- $\mathsf{OTMsg}_0(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_0(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\mathsf{OTMsg}_1(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_1(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus y_n \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\mathsf{OTMsg}_2(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_2(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ b \oplus x_n \oplus y_n \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest and } n < m \\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest or } n \geqslant m \end{cases}$

– $\mathsf{OTMsg}_3(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsg}_3(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\mathsf{OTMsgRcvd}_0(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_0(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\mathsf{OTMsgRcvd}_1(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_1(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\mathsf{OTMsgRcvd}_2(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_2(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := b \leftarrow \mathsf{SendBit}(n,m,K);\ x_n \leftarrow \mathsf{Share}(n,k);\ y_n \leftarrow \mathsf{Share}(n,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ honest and } n < m \\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N \text{ and } m \leqslant N+1 \text{ if } n \text{ semi-honest or } n \geqslant m \end{cases}$

– $\mathsf{OTMsgRcvd}_3(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTMsgRcvd}_3(N+1,m,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $m \leqslant N+1$

– $\begin{cases} \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest or } n \geqslant m \end{cases}$

– $\mathsf{OTChc}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n \leqslant N+1$

- $\begin{cases} \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,l) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest and } n < m \\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest or } n \geqslant m \end{cases}$
- $\mathsf{OTChc}_1(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChc}_1(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n \leqslant N+1$
- $\begin{cases} \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := x_m \leftarrow \mathsf{Share}(m,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_0(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,k);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_0(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := y_m \leftarrow \mathsf{Share}(m,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest and } n < m \\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest or } n \geqslant m \end{cases}$
- $\begin{cases} \mathsf{OTChcRcvd}_1(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := y_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N,l);\ \mathsf{ret}\ \checkmark \\ \quad \text{for } n \leqslant N \\ \mathsf{OTChcRcvd}_1(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTChcRcvd}_1(N+1,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} \end{cases}$
- $\begin{cases} \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdBit}(m,n,K) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ semi-honest} \\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,m,K)^{\mathsf{ot}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ if } m \text{ honest} \end{cases}$
- $\mathsf{OTOut}(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OTOut}(n,N+1,K)^{\mathsf{ot}}_{\mathsf{adv}}$ for $n \leqslant N+1$

This is followed by the hiding of the channels

- $\mathsf{SendBit}(n,m,k)$ for $n \leqslant N$ and $m \leqslant N+1$ and $k < K$,
- $\mathsf{RcvdBit}(n,m,k)$ for $n \leqslant N$ and $m \leqslant N+1$ and $k < K$,
- $\mathsf{Ctrb}(n,m,k)$ for $n \leqslant N$ and $m \leqslant N+1$ and $k < K$,
- $\mathsf{Ctrb}\text{-}\Sigma(n,m,k)$ for $n \leqslant N$ and $m \leqslant N+1$ and $k < K$.

In the final phase, we have the protocol Fin:

- $\begin{cases} \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(n,k) \\ \quad \text{for } m \leqslant N+1 \text{ and } n \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,n,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } m \leqslant N+1 \text{ and } n \leqslant N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\mathsf{SendOutShare}(m,N+1,k)^{\mathsf{party}(N+1)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{SendOutShare}(m,N+1,k)^{\mathsf{party}(N+1)}_{\mathsf{adv}}$ for $m \leqslant N+1$
- $\begin{cases} \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m,k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n,m,k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

- $\begin{cases} \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{RcvdOutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}(m, k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(N, k);\ x \leftarrow \mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}};\ \mathsf{ret}\ x_\Sigma \oplus x \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Share}\text{-}\Sigma(m, k) \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n, m, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } m \leqslant N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{OutShare}\text{-}\Sigma(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \mathsf{OutShare}\text{-}\Sigma(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{OutShare}\text{-}\Sigma(n, N+1, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$
- $\begin{cases} \mathsf{Out}(n, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \mathsf{Out}(n, k)^{\mathsf{party}(n)}_{\mathsf{adv}} := \mathsf{read}\ \mathsf{Out}(n, k)^{\mathsf{party}(n)}_{\mathsf{adv}} \\ \quad \text{for } n \leqslant N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

At last, we have the channels

- $\begin{cases} \mathsf{Share}\text{-}\Sigma(0, k) := \mathsf{read}\ \mathsf{Share}(0, k) \\ \quad \text{for } k < K \\ \mathsf{Share}\text{-}\Sigma(m+1, k) := x_\Sigma \leftarrow \mathsf{Share}\text{-}\Sigma(m, k);\ x_{m+1} \leftarrow \mathsf{Share}(m+1, k);\ \mathsf{ret}\ x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \text{ and } k < K \end{cases}$

that keep track of the sum of shares of parties $0, \ldots, N$.

The composition of the four parts is followed by the hiding of the channels

- $\mathsf{InShare}(m, n, i)$ for $m \leqslant N$ and $n \leqslant N+1$ and $i < I_n$,
- $\mathsf{Share}(n, k)$ for $n \leqslant N$ and $k < K$,
- $\mathsf{Share}\text{-}\Sigma(m, k)$ for $m \leqslant N$ and $k < K$.

Composing the ideal protocol with the simulator, and substituting away the ideal leakage

- $\mathsf{In}(n, i)^{\mathsf{id}}_{\mathsf{adv}}$ for $n \leqslant N+1$ and $i < I_n$,
- $\mathsf{InRcvd}(n, i)^{\mathsf{id}}_{\mathsf{adv}}$ for $n \leqslant N+1$ and $i < I_n$,
- $\mathsf{Out}(n, k)^{\mathsf{id}}_{\mathsf{adv}}$ for $n \leqslant N+1$ and $k < K$

as indicated in Section 10.4.8 yields precisely the version of the real protocol we had at the end of Section 10.4.7.