

IPDL Case Studies

Kristina Sojakova

Mihai Codescu

Joshua Gancher

June 4, 2024

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; Two-Party GMW Protocol in Section 9; and Multi-Party GMW Protocol in Section 10.

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

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

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

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

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

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

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

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

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

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

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

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

can be factored as the composition of the protocol

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

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

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

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

Applying the CPA assumption yields the composition of the protocol

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

and the (unchanged) protocol

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

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

Reversing the factorization process yields the protocol

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

as desired.

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

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

can be factored as the composition of the protocol

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

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

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

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

Applying the CPA assumption yields the composition of the protocol

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

and the (unchanged) protocol

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

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

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

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

The above can be factored as the composition of the protocol

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

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

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

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

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

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

and the (unchanged) protocol

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

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

Reversing the factorization process yields the protocol

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

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

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

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

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

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

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

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

can be factored as the composition of the protocol

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

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

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

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

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

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

and the (unchanged) protocol

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

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

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

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

The above can be factored as the composition of the protocol

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

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

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

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

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

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

and the (unchanged) protocol

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

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

Reversing the factorization process yields the protocol

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

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

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 $\text{In} : \text{msg}, \text{Key} : \text{key}, \text{Enc} : \text{ctxt}, \text{Dec} : \text{msg}$ the protocol

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

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

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

Applying this assumption q times, we get that the protocol

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

rewrites strictly to the protocol

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

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 $\{\text{In}(i) : \text{msg}\}_i, \text{Key} : \text{key}, \{\text{Enc}(i) : \text{ctxt}\}_i$ where $i := 1, \dots, q$, the protocol

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

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

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

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:

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

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:

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

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

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

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

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

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

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

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 $\text{LeakMsgRcvd}(-)_{\text{adv}}^{\text{id}}$, $\text{OkCtxt}(-)_{\text{net}}^{\text{adv}}$ are inputs to the simulator and the channels $\text{LeakCtxt}(-)_{\text{adv}}^{\text{net}}$, $\text{OkMsg}(-)_{\text{id}}^{\text{adv}}$ 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:

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

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

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

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

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

3.5 Real \approx Ideal + Simulator

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

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

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

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

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

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

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

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

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

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

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

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

As we observed earlier, the subprotocol

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

strictly rewrites to the following protocol snippet:

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

Our original protocol thus becomes:

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

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

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

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

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

Plugging this back into the original protocol yields the following:

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

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

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

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

4.1 The Assumptions

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

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

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

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

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 $\text{DDH}_3 : \text{msg} \times (\text{msg} \times \text{msg})$, the single-reaction no-input protocol

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

rewrites approximately to the protocol

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

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:

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

Here the channel 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:

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

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

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

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

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

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

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

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

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

Thus, we have the following code for Alice:

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

The code for Bob has the symmetric form:

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

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

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

and from Bob to Alice:

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

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

4.4 The Simulator

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

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

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

4.5 Real \approx Ideal + Simulator

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

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

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

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

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

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

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

- $\text{SecretKey}(\text{Alice}) := \text{samp unif}_{\text{key}}$

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

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

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

We can extract the key construction into a new internal channel SharedKey :

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

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

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

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

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

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

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

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

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

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

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

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

Unfolding the three samplings from the channel DDH_3 into new internal channels $\text{SecretKey}(\text{Alice})$, $\text{SecretKey}(\text{Bob})$, Key yields

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

The internal channel DDH_3 can now be substituted away:

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

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

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

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

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

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

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

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

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:

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

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

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

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:

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

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

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

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

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

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:

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

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

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

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

5.5 Real \approx Ideal + Simulator: One-Time Pad

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

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

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

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

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

where the channels Send and Recv are internal, followed by the hiding of the channels $\text{Key}(\text{Alice})$ and $\text{Key}(\text{Bob})$.

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

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

followed by the hiding of the channels $\text{Key}(\text{Alice})$ and $\text{Key}(\text{Bob})$.

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 $\text{OkKey}(\text{Alice})_{\text{id}}^{\text{adv}}$ and $\text{OkKey}(\text{Bob})_{\text{id}}^{\text{adv}}$. Thus, we can express the real protocol as the composition of the key exchange idealization, the key exchange simulator, and the protocol

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

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

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

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

where the channel `SharedKey` is internal. Canceling out k with its inverse in the channel `Out` yields

- `Out := _ ← OkCtxtnetadv; m ← In; _ ← OkKey(Alice)idadv; _ ← OkKey(Bob)idadv; k ← SharedKey; ret m`

Dropping the gratuitous dependency on the channel `SharedKey` yields

- `Out := _ ← OkCtxtnetadv; m ← In; _ ← OkKey(Alice)idadv; _ ← OkKey(Bob)idadv; ret m`

which simplifies to

- `Out := _ ← OkCtxtnetadv; _ ← OkKey(Alice)idadv; _ ← OkKey(Bob)idadv; read In`

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

- `SharedKey := samp unifmsg`
- `LeakCtxtadvnet := m ← In; _ ← OkKey(Alice)idadv; k ← SharedKey; ret m * k`
- `Out := _ ← OkCtxtnetadv; _ ← OkKey(Alice)idadv; _ ← OkKey(Bob)idadv; read In`

where the channel `SharedKey` is internal, followed by the hiding of the channels `OkKey(Alice)idadv` and `OkKey(Bob)idadv`. Since the channel `SharedKey` is now only used in the channel `LeakCtxtadvnet`, we can fold it in:

- `LeakCtxtadvnet := m ← In; _ ← OkKey(Alice)idadv; k ← unifmsg; ret m * k`
- `Out := _ ← OkCtxtnetadv; _ ← OkKey(Alice)idadv; _ ← OkKey(Bob)idadv; read In`

By assumption, the distribution `unifmsg` on messages is invariant under the operation of multiplication with a fixed message, so we can simplify the channel `LeakCtxtadvnet` as follows:

- `LeakCtxtadvnet := m ← In; _ ← OkKey(Alice)idadv; samp unifmsg`

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

- `LeakCtxtadvnet := m ← In; _ ← OkKey(Alice)idadv; samp unifmsg`
- `Out := _ ← OkCtxtnetadv; _ ← OkKey(Alice)idadv; _ ← OkKey(Bob)idadv; read In`

followed by the hiding of the channels `OkKey(Alice)idadv` and `OkKey(Bob)idadv`. 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 $\{0, \dots, p-1\}$, where p is a prime); a type `msg` of messages, also serving as public keys (elements of a cyclic group $G = \{g^0, \dots, g^{p-1}\}$); a uniform distribution `unifkey : 1 → key` on keys; a uniform distribution `unifmsg : 1 → msg` on messages; the generator `g : 1 → msg` of G ; the group multiplication function `mul : msg × msg → msg`, where we write $m * k$ in place of `mul(m, k)`; the group inverse function `inv : msg → msg`, where we write k^{-1} in place of `inv k`; and the exponentiation function `exp : msg × key → msg` that raises an element of G to the power of $k \in \{0, \dots, p-1\}$. We will write m^k in place of `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:

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

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

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

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

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

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 $\text{DDH}_3 : \text{msg} \times (\text{msg} \times \text{msg})$, the single-reaction no-input protocol

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

rewrites approximately to the protocol

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

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

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

rewrites approximately to the protocol

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

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

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

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

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

We subsequently add a gratuitous dependency on the channel $\text{SecretEphemeralKey}$ into the channel PublicKey :

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

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

- $\text{SecretKey} := \text{samp unif}_{\text{key}}$
- $\text{SecretEphemeralKey} := \text{samp unif}_{\text{key}}$
- $\text{DDH}_3 := k_B \leftarrow \text{SecretKey}; k_A \leftarrow \text{SecretEphemeralKey}; \text{ret } (g^{k_B}, (g^{k_A}, (g^{k_B})^{k_A}))$
- $\text{PublicKey} := (p_B, \text{ddh}_2) \leftarrow \text{DDH}_3; \text{ret } p_B$
- $\text{DDH}_2 := (p_B, \text{ddh}_2) \leftarrow \text{DDH}_3; \text{ret } \text{ddh}_2$

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

- $\text{DDH}_3 := k_B \leftarrow \text{unif}_{\text{key}}; k_A \leftarrow \text{unif}_{\text{key}}; \text{ret } (g^{k_B}, (g^{k_A}, (g^{k_B})^{k_A}))$
- $\text{PublicKey} := (p_B, \text{ddh}_2) \leftarrow \text{DDH}_3; \text{ret } p_B$
- $\text{DDH}_2 := (p_B, \text{ddh}_2) \leftarrow \text{DDH}_3; \text{ret } \text{ddh}_2$

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

- $\text{DDH}_3 := k_B \leftarrow \text{unif}_{\text{key}}; k_A \leftarrow \text{unif}_{\text{key}}; k_R \leftarrow \text{unif}_{\text{key}}; \text{ret } (g^{k_B}, (g^{k_A}, g^{k_R}))$
- $\text{PublicKey} := (p_B, \text{ddh}_2) \leftarrow \text{DDH}_3; \text{ret } p_B$
- $\text{DDH}_2 := (p_B, \text{ddh}_2) \leftarrow \text{DDH}_3; \text{ret } \text{ddh}_2$

Unfolding the three samplings from the channel DDH_3 into new internal channels SecretKey , $\text{SecretEphemeralKey}$, Key yields

- $\text{SecretKey} := \text{samp unif}_{\text{key}}$
- $\text{SecretEphemeralKey} := \text{samp unif}_{\text{key}}$
- $\text{Key} := \text{samp unif}_{\text{key}}$
- $\text{DDH}_3 := k_B \leftarrow \text{SecretKey}; k_A \leftarrow \text{SecretEphemeralKey}; k_R \leftarrow \text{Key}; \text{ret } (g^{k_B}, (g^{k_A}, g^{k_R}))$
- $\text{PublicKey} := (p_B, \text{ddh}_2) \leftarrow \text{DDH}_3; \text{ret } p_B$
- $\text{DDH}_2 := (p_B, \text{ddh}_2) \leftarrow \text{DDH}_3; \text{ret } \text{ddh}_2$

The internal channel DDH_3 can now be substituted away:

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

After dropping the gratuitous dependencies – on channels `SecretEphemeralKey`, `Key` in the channel `PublicKey`, and channel `SecretKey` in the channel `DDH2` – we end up with the following:

- `SecretKey := samp unifkey`
- `SecretEphemeralKey := samp unifkey`
- `Key := samp unifkey`
- `PublicKey := $k_B \leftarrow \text{SecretKey}$; ret g^{k_B}`
- `DDH2 := $k_A \leftarrow \text{SecretEphemeralKey}$; $k_R \leftarrow \text{Key}$; ret (g^{k_A}, g^{k_R})`

The channel `SecretKey` is now only used in the channel `PublicKey`, and the channels `SecretEphemeralKey` and `Key` are only used in the channel `DDH2`. Performing the appropriate foldings results in the protocol

- `PublicKey := $k_B \leftarrow \text{unif}_{\text{key}}$; ret g^{k_B}`
- `DDH2 := $k_A \leftarrow \text{unif}_{\text{key}}$; $k_R \leftarrow \text{unif}_{\text{key}}$; ret (g^{k_A}, g^{k_R})`

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

- `PublicKey := $k_B \leftarrow \text{unif}_{\text{key}}$; ret g^{k_B}`
- `DDH2(i) := $p_B \leftarrow \text{PublicKey}$; $k_A \leftarrow \text{unif}_{\text{key}}$; ret $(g^{k_A}, p_B^{k_A})$ for $0 \leq i < q$`

rewrites approximately to the protocol

- `PublicKey := $k_B \leftarrow \text{unif}_{\text{key}}$; ret g^{k_B}`
- `DDH2(i) := $k_A \leftarrow \text{unif}_{\text{key}}$; $k_R \leftarrow \text{unif}_{\text{key}}$; ret (g^{k_A}, g^{k_R}) for $0 \leq i < q$`

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:

- `LeakMsgRcvd(i)advid := $m \leftarrow \text{In}(i)$; ret \checkmark for $0 \leq i < q$`
- `Out(i) := $_ \leftarrow \text{OkMsg}(i)$ idadv; read $\text{In}(i)$ for $0 \leq i < q$`

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:

- `SecretKey := samp unifkey`

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:

- `PublicKey := $k_B \leftarrow \text{SecretKey}$; ret g^{k_B}`
- `LeakPublicKeyadvrec := read 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 $(g^{k_B})^{k_A}$:

- `SecretEphemeralKey(i) := samp unifkey for $0 \leq i < q$`
- `PublicEphemeralKey(i) := $k_A \leftarrow \text{SecretEphemeralKey}(i)$; ret g^{k_A}`

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

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

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

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 :

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

Thus, we have the following code for Alice:

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

The code for Bob is below:

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

Finally, we recall the code for the authenticated channel:

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

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

6.4 The Simulator

The channels $\text{OkCtxt}(-)_{\text{net}}^{\text{adv}}$, $\text{LeakMsgRcvd}(-)_{\text{adv}}^{\text{id}}$ are inputs to the simulator, whereas the channels $\text{OkMsg}(i)_{\text{id}}^{\text{adv}}$, $\text{LeakPublicKey}_{\text{adv}}^{\text{rec}}$, $\text{LeakCtxt}(-)_{\text{net}}^{\text{adv}}$ 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 :

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

6.5 Real \approx Ideal + Simulator

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

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

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

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

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

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

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

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

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

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

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

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

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

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

which simplifies to

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

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

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

The channel SecretKey is now only used in PublicKey , so we can fold it in:

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

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

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

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

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

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

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

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

As we observed earlier, the protocol snippet

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

rewrites approximately to

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

Our real-world protocol therefore approximately rewrites to the protocol

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

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

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

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

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

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

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

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

We now unfold the sampling from `PublicKey` into a new internal channel `SecretKey`:

- `SecretKey := samp unifkey`
- `PublicKey := $k_B \leftarrow \text{SecretKey}$; ret g^{k_B}`
- `LeakPublicKeyadvrec := read PublicKey`
- `SecretEphemeralKey(i) := samp unifkey for $0 \leq i < q$`
- `Key(i) := samp unifkey for $0 \leq i < q$`
- `PublicEphemeralKey(i) := $k_A \leftarrow \text{SecretEphemeralKey}(i)$; ret g^{k_A} for $0 \leq i < q$`
- `SessionKey(i) := $k_R \leftarrow \text{Key}(i)$; ret g^{k_R} for $0 \leq i < q$`
- `LeakCtxt(i)netadv := $m \leftarrow \text{In}(i)$; $k \leftarrow \text{SessionKey}(i)$; $p_A \leftarrow \text{PublicEphemeralKey}(i)$; ret $(p_A, m * k)$ for $0 \leq i < q$`
- `Out(i) := $_ \leftarrow \text{OkCtxt}(i)$ netadv; read $\text{In}(i)$ for $0 \leq i < q$`

The channel `Key(i)` is now only used in `SessionKey(i)`, so we can fold it in:

- `SecretKey := samp unifkey`
- `PublicKey := $k_B \leftarrow \text{SecretKey}$; ret g^{k_B}`
- `LeakPublicKeyadvrec := read PublicKey`
- `SecretEphemeralKey(i) := samp unifkey for $0 \leq i < q$`
- `PublicEphemeralKey(i) := $k_A \leftarrow \text{SecretEphemeralKey}(i)$; ret g^{k_A} for $0 \leq i < q$`
- `SessionKey(i) := $k_R \leftarrow \text{unif}_{\text{key}}$; ret g^{k_R} for $0 \leq i < q$`
- `LeakCtxt(i)netadv := $m \leftarrow \text{In}(i)$; $k \leftarrow \text{SessionKey}(i)$; $p_A \leftarrow \text{PublicEphemeralKey}(i)$; ret $(p_A, m * k)$ for $0 \leq i < q$`
- `Out(i) := $_ \leftarrow \text{OkCtxt}(i)$ netadv; read $\text{In}(i)$ for $0 \leq i < q$`

By assumption, sampling a random secret key k_R from `unifkey` and returning g^{k_R} is the same as sampling from `unifmsg` directly, so we can simplify the channel `SessionKey(i)` as follows:

- `SecretKey := samp unifkey`
- `PublicKey := $k_B \leftarrow \text{SecretKey}$; ret g^{k_B}`
- `LeakPublicKeyadvrec := read PublicKey`
- `SecretEphemeralKey(i) := samp unifkey for $0 \leq i < q$`
- `PublicEphemeralKey(i) := $k_A \leftarrow \text{SecretEphemeralKey}(i)$; ret g^{k_A} for $0 \leq i < q$`
- `SessionKey(i) := samp unifmsg for $0 \leq i < q$`
- `LeakCtxt(i)netadv := $m \leftarrow \text{In}(i)$; $k \leftarrow \text{SessionKey}(i)$; $p_A \leftarrow \text{PublicEphemeralKey}(i)$; ret $(p_A, m * k)$ for $0 \leq i < q$`
- `Out(i) := $_ \leftarrow \text{OkCtxt}(i)$ netadv; read $\text{In}(i)$ for $0 \leq i < q$`

We can now extract the encoding of the input message into a new internal channel `Enc(i)`:

- `SecretKey := samp unifkey`
- `PublicKey := $k_B \leftarrow \text{SecretKey}$; ret g^{k_B}`

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

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

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

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

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

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

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

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 $\text{flip} : 1 \rightarrow \text{Bool}$; a uniform distribution $\text{unif}_{\text{msg}} : 1 \rightarrow \text{msg}$ on messages; and a bitwise xor function $\oplus_{\text{msg}} : \text{msg} \times \text{msg} \rightarrow \text{msg}$ 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:

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

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

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

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:

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

In code:

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

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

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

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

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

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:

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

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

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

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

He subsequently sends his choice to the OT functionality:

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

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

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

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

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

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

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

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:

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

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:

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

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

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

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:

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

Thus, we have the following code for Alice:

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

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

The code for Bob has the following form:

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

Finally, we recall the code for the OT functionality:

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

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:

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

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

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

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

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

Substituting the channel ChcEnc into $\text{MsgEnc}(0)$ and $\text{MsgEnc}(1)$ yields

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

Substituting the channel SharedKey into Out yields

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

Further substituting the channels $\text{MsgEnc}(0)$ and $\text{MsgEnc}(1)$ into Out yields

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

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

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

We can now drop the gratuitous dependencies on channels $\text{Key}(0)$, $\text{Key}(1)$:

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

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

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

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

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

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

- $\text{Key}(0) := \text{samp unif}_{\text{msg}}$
- $\text{Key}(1) := \text{samp unif}_{\text{msg}}$
- $\text{Flip} := \text{samp flip}$
- $\text{Flip}_{\text{adv}}^{\text{rec}} := \text{read } \text{Flip}$
- $\text{OTMsgRcvd}(0)_{\text{adv}}^{\text{ot}} := k_0 \leftarrow \text{Key}(0); \text{ ret } \checkmark$
- $\text{OTMsgRcvd}(1)_{\text{adv}}^{\text{ot}} := k_1 \leftarrow \text{Key}(1); \text{ ret } \checkmark$
- $\text{OTChc}_{\text{adv}}^{\text{ot}} := \text{read } \text{Flip}$
- $\text{OTOut}_{\text{adv}}^{\text{ot}} := \text{read } \text{SharedKey}$
- $\text{SharedKey} := k_0 \leftarrow \text{Key}(0); k_1 \leftarrow \text{Key}(1); f \leftarrow \text{Flip}; \text{ if } f \text{ then ret } k_1 \text{ else ret } k_0$
- $\text{Chc}_{\text{adv}}^{\text{rec}} := \text{read } \text{Chc}$

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

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:

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

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

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

Rearranging the order of bindings inside KeyPair yields

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

Since sampling and branching are interchangeable, the three reaction snippets

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

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

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

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

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

Unfolding the samplings back thus yields the following protocol:

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

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

The internal channel KeyPair can now be substituted away:

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

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

The second key is now only referenced in the channels $\text{MsgEnc}(0)$ and $\text{MsgEnc}(1)$, 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 PrivMsg :

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

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

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

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

Rearranging the order of bindings inside PrivMsg yields

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

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

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

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

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

Unfolding the sampling back gives us:

- $\text{Key}(0) := \text{samp unif}_{\text{msg}}$
- $\text{Key}(1) := \text{samp unif}_{\text{msg}}$
- $\text{Flip} := \text{samp flip}$
- $\text{Flip}_{\text{adv}}^{\text{rec}} := \text{read Flip}$
- $\text{OTMsgRcvd}(0)_{\text{adv}}^{\text{ot}} := k_0 \leftarrow \text{Key}(0); \text{ret } \checkmark$
- $\text{OTMsgRcvd}(1)_{\text{adv}}^{\text{ot}} := k_1 \leftarrow \text{Key}(1); \text{ret } \checkmark$

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

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

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

7.5 The Simulator

The channel

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

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 $\text{Chc}_{\text{adv}}^{\text{id}}$ and $\text{Out}_{\text{adv}}^{\text{id}}$ that originally served as a line of communication for the adversary yields the final version of the real protocol, as desired.

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

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 $\text{flip} : 1 \rightarrow \text{Bool}$ and a Boolean sum function $\oplus : \text{Bool} \times \text{Bool} \rightarrow \text{Bool}$, 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:

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

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):

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

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:

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

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:

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

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:

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

To uniformly cover all cases, we assume channels $\text{AdvCommit}_{\text{party}(i)}^{\text{adv}}$ as inputs to the real protocol, for all $i \leq N + 1$ 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:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The output of each corrupted party i diverges:

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

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

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

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

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

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

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

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

8.4 Real = Ideal + Simulator

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

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

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 $\text{AllCommitted}(-)$ that inductively keep a global track of commitment:

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

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

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

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

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

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

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

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

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

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

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

- $\text{Commit}(i) := \text{samp flip}$
- $\text{Open}(i) := _ \leftarrow \text{AllCommitted}(N + 2); \text{ret } \checkmark$
- $\text{Out}(i) := \text{read SumOpened}(N + 2)$

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

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

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

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

- $\begin{cases} \text{SumOpened}(0) := \text{ret false} \\ \text{SumOpened}(j+1) := x_j \leftarrow \text{SumOpened}(j); o_j \leftarrow \text{Opened}(j); \text{ret } x_j \oplus o_j \text{ for } j \leq N+1 \end{cases}$
- $\begin{cases} \text{Out}(i) := \text{read SumOpened}(N+2) & \text{if } i \leq N+1 \text{ honest} \\ \text{Out}(i) := \text{read Out}(i) & \text{if } i \leq N+1 \text{ corrupt} \end{cases}$

We can keep a summary of the decisions to open the commits using new internal channels $\text{AllOpen}(-)$:

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

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

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

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

By substituting the channel $\text{SumOpened}(N+2)$ into $\text{Out}(i)$ for $i \leq N+1$ honest, the internal channels $\text{SumOpened}(-)$ become unused and we can eliminate them entirely:

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

We now note that since the channel

- $\text{Commit}(N+1) := \text{samp flip}$

introduces no dependencies, the dependency on $\text{Commit}(N+1)$ in the channel

- $\text{Committed}(N+1) := c_{N+1} \leftarrow \text{Commit}(N+1); \text{ret } \checkmark$

can be dropped, yielding the following:

- $\text{Committed}(N+1) := \text{ret } \checkmark$

In the presence of the above, we can rewrite the channel

- $\text{AllCommitted}(N+2) := _ \leftarrow \text{AllCommitted}(N+1); _ \leftarrow \text{Committed}(N+1); \text{ret } \checkmark$

equivalently as follows:

- $\text{AllCommitted}(N + 2) := \text{read AllCommitted}(N + 1)$

Summarizing, we now have the following protocol:

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

By substituting the channel $\text{AllCommitted}(N + 2)$ into $\text{Open}(i)$ for $i \leq N + 1$ honest, this internal channel becomes unused and we can eliminate it entirely:

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

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

The remaining channels $\text{AllCommitted}(-)$ can be simplified:

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

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

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

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

This is the final version of the real protocol. 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. 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. This suggests the following code for the simulator, where we note that in the channels $\text{SumCommit}(N + 2)$ and $\text{Opened}(N + 1)$ we use the channel LastCommit in lieu of $\text{Commit}(N + 1)$.

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

Plugging the simulator into the ideal protocol and substituting away the channels $\text{LeakFlip}(N)_{\text{adv}}^{\text{id}}$ and $\text{Ok}_{\text{id}}^{\text{adv}}$ that have now become internal yields the following:

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

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

We now note that for any honest party $i \leq N + 1$ we can express the channel

- $\text{Out}(i) := _ \leftarrow \text{AllOpen}(N + 2); x_{N+1} \leftarrow \text{SumCommit}(N + 1); \text{read Flip}$

equivalently as follows:

- $\text{Out}(i) := _ \leftarrow \text{AllOpen}(N + 2); \text{read SumCommit}(N + 2)$

To show this, we start by substituting the channel

- $\text{SumCommit}(N + 2) := x_{N+1} \leftarrow \text{SumCommit}(N + 1); c_{N+1} \leftarrow \text{LastCommit}; \text{ret } x_{N+1} \oplus c_{N+1}$

into the channel

- $\text{Out}(i) := _ \leftarrow \text{AllOpen}(N + 2); \text{read SumCommit}(N + 2)$

which yields the following:

- $\text{Out}(i) := _ \leftarrow \text{AllOpen}(N + 2); x_{N+1} \leftarrow \text{SumCommit}(N + 1); c_{N+1} \leftarrow \text{LastCommit}; \text{ret } x_{N+1} \oplus c_{N+1}$

We subsequently substitute the channel

- $\text{LastCommit} := x_{N+1} \leftarrow \text{SumCommit}(N + 1); f \leftarrow \text{Flip}; \text{ret } x_{N+1} \oplus f$

into the above, yielding

- $\text{Out}(i) := _ \leftarrow \text{AllOpen}(N + 2); x_{N+1} \leftarrow \text{SumCommit}(N + 1); f \leftarrow \text{Flip}; \text{ret } x_{N+1} \oplus (x_{N+1} \oplus f)$

By assumption, we can cancel out the Boolean sum:

- $\text{Out}(i) := _ \leftarrow \text{AllOpen}(N + 2); x_{N+1} \leftarrow \text{SumCommit}(N + 1); f \leftarrow \text{Flip}; \text{ret } f$

and this is equivalent to

- $\text{Out}(i) := _ \leftarrow \text{AllOpen}(N + 2); x_{N+1} \leftarrow \text{SumCommit}(N + 1); \text{read Flip}$

as desired. Thus, we have the following protocol:

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

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

The channel Flip now only occurs in the channel LastCommit, so we can fold it in:

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

- $\text{LeakOpened}(i)_{\text{adv}}^{\text{comm}} := \text{read Opened}(i)$ for $i \leq N + 1$
- $\begin{cases} \text{Out}(i) := _ \leftarrow \text{AllOpen}(N + 2); \text{read SumCommit}(N + 2) & \text{if } i \leq N + 1 \text{ honest} \\ \text{Out}(i) := \text{read Out}(i) & \text{if } i \leq N + 1 \text{ corrupt} \end{cases}$

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

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

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

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

The internal channel LastCommit can now be substituted away:

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

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

We now note that the channel

- $\text{Opened}(N + 1) := _ \leftarrow \text{Open}(N + 1); x_{N+1} \leftarrow \text{SumCommit}(N + 1); \text{read Commit}(N + 1)$

can be equivalently expressed as follows:

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

This is easy to see by substituting the channel

- $\text{Open}(N + 1) := x_{N+1} \leftarrow \text{SumCommit}(N + 1); \text{ret } \checkmark$

into both formulations, yielding

- $\text{Opened}(N + 1) := x_{N+1} \leftarrow \text{SumCommit}(N + 1); \text{read Commit}(N + 1)$

in both cases. The two formulations are therefore equivalent. Our protocol now looks as follows:

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

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

But this is precisely the final version of the real protocol.

8.5 The Simulator

We record here the simulator:

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

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

We represent Boolean circuits using the syntax below, where we assume Alice has $N \geq 0$ inputs labeled $\{0, \dots, N - 1\}$, and Bob has $M \geq 0$ inputs labeled $\{0, \dots, M - 1\}$. 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	$p := A, B$
Inputs	$i \in \mathbb{N}$
Wires	$k, l \in \mathbb{N}$
Circuits	$C ::= \epsilon \mid C; \text{input-gate}(p, i) \mid C; \text{not-gate}(k) \mid C; \text{xor-gate}(k, l) \mid C; \text{and-gate}(k, l)$

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

$$\begin{array}{c}
\frac{}{\epsilon \text{ circuit}(0)} \quad \frac{C \text{ circuit}(n) \quad i < N}{C; \text{input-gate}(A, i) \text{ circuit}(n+1)} \quad \frac{C \text{ circuit}(n) \quad i < M}{C; \text{input-gate}(B, i) \text{ circuit}(n+1)} \\
\\
\frac{C \text{ circuit}(n) \quad 0 \leq k < n}{C; \text{not-gate}(k) \text{ circuit}(n+1)} \quad \frac{C \text{ circuit}(n) \quad 0 \leq k < n \quad 0 \leq l < n}{C; \text{xor-gate}(k, l) \text{ circuit}(n+1)} \\
\\
\frac{C \text{ circuit}(n) \quad 0 \leq k < n \quad 0 \leq l < n}{C; \text{and-gate}(k, l) \text{ circuit}(n+1)}
\end{array}$$

We now fix an ambient Boolean circuit C with K wires $\{0, \dots, K-1\}$, 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:

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

Furthermore, Boolean multiplication distributes over Boolean sum:

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

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

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

Negating a Boolean equals summing it up with **true**:

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

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

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

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):

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

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:

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

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

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:

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

Finally, the channels $\text{Wire}(-)$ coming from the inductive protocol $\text{Wires}(C, K)$ 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:

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

She next randomly generates shares for each of her inputs:

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

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

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

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

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

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

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

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

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

The channels

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

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:

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

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

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

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

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

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

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

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

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

The channels

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

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:

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

We define Alice's inductive part as follows:

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

As we said earlier, the 1-Out-Of-4 OT exchange with Bob is vacuous:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Finally, the channels

- $\text{SendBit}(A, B, k)$ 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:

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

We define Bob's inductive part as follows:

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

As we said earlier, the 1-Out-Of-4 OT exchange with Alice is vacuous:

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

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

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

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

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

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

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

- $\text{RcvdBit}(\mathbf{B}, \mathbf{A}, K) := \text{read } \text{RcvdBit}(\mathbf{B}, \mathbf{A}, K)$
- $\text{Circ}_{\mathbf{B}}(C; \text{not-gate}(k), K + 1)$ the composition of $\text{Circ}_{\mathbf{B}}(C, K)$ with the following protocol. Bob's share is the negation of the share on wire k :
 - $\text{Share}(\mathbf{B}, K) := x_B \leftarrow \text{Share}(\mathbf{B}, k); \text{ret } \neg x_B$

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

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

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

- $\text{RcvdBit}(\mathbf{B}, \mathbf{A}, K) := \text{read } \text{RcvdBit}(\mathbf{B}, \mathbf{A}, K)$
- $\text{Circ}_{\mathbf{B}}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Circ}_{\mathbf{B}}(C, K)$ with the following protocol. Bob's share is the sum of shares on wires k and l :
 - $\text{Share}(\mathbf{B}, K) := x_B \leftarrow \text{Share}(\mathbf{B}, k); y_B \leftarrow \text{Share}(\mathbf{B}, l); \text{ret } x_B \oplus y_B$

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

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

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

- $\text{RcvdBit}(\mathbf{B}, \mathbf{A}, K) := \text{read } \text{RcvdBit}(\mathbf{B}, \mathbf{A}, K)$
- $\text{Circ}_{\mathbf{B}}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Circ}_{\mathbf{B}}(C, K)$ with the following protocol. First, Bob carries out the 1-Out-Of-4 OT exchange with Alice:
 - $\text{OTChc}_0(\mathbf{A}, \mathbf{B}, K) := \text{read } \text{Share}(\mathbf{B}, k)$
 - $\text{OTChc}_1(\mathbf{A}, \mathbf{B}, K) := \text{read } \text{Share}(\mathbf{B}, l)$

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

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

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

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

Finally, the channels

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

are declared as internal.

9.3.5 Alice: Final Phase

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

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

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

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

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

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

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

- $\text{Out}(\mathbf{A}, k) := x_A \leftarrow \text{OutShare}(\mathbf{A}, \mathbf{A}, k); x_B \leftarrow \text{OutShare}(\mathbf{A}, \mathbf{B}, k); \text{ret } x_A \oplus x_B \text{ for } k < K$
- $\text{Out}(\mathbf{A}, k)_{\text{adv}}^{\mathbf{A}} := \text{read Out}(\mathbf{A}, k) \text{ for } k < K$

The channels

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

are declared as internal.

9.3.6 Bob: Final Phase

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

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

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

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

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

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

The channels

- $\text{OutShare}(\mathbf{B}, \mathbf{A}, k)$ for $k < K$,
- $\text{OutShare}(\mathbf{B}, \mathbf{B}, k)$ 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 $1\text{OutOf4OT}(k)$, which we now describe. Since Alice is semi-honest, the functionality leaks the value of all messages received from Alice:

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

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

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

The functionality then selects the appropriate message:

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

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

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

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

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

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

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

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

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

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

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

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

This is followed by the hiding of the channels

- $\text{InShare-}\$(A, A, i)$ for $i < N$,
- $\text{InShare-}\$(A, B, i)$ for $i < M$,
- $\text{InShare-}\$(B, A, i)$ for $i < N$,
- $\text{InShare-}\$(B, B, i)$ for $i < M$,
- $\text{SendInShare}(B, A, i)$ for $i < N$,
- $\text{SendInShare}(A, B, i)$ 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 $\text{Circ}(C, K)$:

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

- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := c_1 \leftarrow \text{OTChc}_1(A, B, K); \text{ret } \checkmark$
- $\text{OTOut}(A, B, K) := m_0 \leftarrow \text{OTMsg}_0(A, B, K); m_1 \leftarrow \text{OTMsg}_1(A, B, K); m_2 \leftarrow \text{OTMsg}_2(A, B, K);$
 $m_3 \leftarrow \text{OTMsg}_3(A, B, K); c_0 \leftarrow \text{OTChc}_0(A, B, K); c_1 \leftarrow \text{OTChc}_1(A, B, K);$
if c_0 then (if c_1 then ret m_3 else ret m_2) else (if c_1 then ret m_1 else ret m_0)
- $\text{Circ}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Circ}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read InShare}(A, B, i)$
 - $\text{Share}(B, K) := \text{read InShare}(B, B, i)$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K) := \text{read OTMsg}_0(A, B, K)$
 - $\text{OTMsg}_1(A, B, K) := \text{read OTMsg}_1(A, B, K)$
 - $\text{OTMsg}_2(A, B, K) := \text{read OTMsg}_2(A, B, K)$
 - $\text{OTMsg}_3(A, B, K) := \text{read OTMsg}_3(A, B, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)$
 - $\text{OTChc}_0(A, B, K) := \text{read OTChc}_0(A, B, K)$
 - $\text{OTChc}_1(A, B, K) := \text{read OTChc}_1(A, B, K)$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := c_0 \leftarrow \text{OTChc}_0(A, B, K); \text{ret } \checkmark$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := c_1 \leftarrow \text{OTChc}_1(A, B, K); \text{ret } \checkmark$
 - $\text{OTOut}(A, B, K) := m_0 \leftarrow \text{OTMsg}_0(A, B, K); m_1 \leftarrow \text{OTMsg}_1(A, B, K); m_2 \leftarrow \text{OTMsg}_2(A, B, K);$
 $m_3 \leftarrow \text{OTMsg}_3(A, B, K); c_0 \leftarrow \text{OTChc}_0(A, B, K); c_1 \leftarrow \text{OTChc}_1(A, B, K);$
if c_0 then (if c_1 then ret m_3 else ret m_2) else (if c_1 then ret m_1 else ret m_0)
- $\text{Circ}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Circ}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read Share}(A, k)$
 - $\text{Share}(B, K) := x_B \leftarrow \text{Share}(B, k); \text{ret } \neg x_B$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K) := \text{read OTMsg}_0(A, B, K)$
 - $\text{OTMsg}_1(A, B, K) := \text{read OTMsg}_1(A, B, K)$
 - $\text{OTMsg}_2(A, B, K) := \text{read OTMsg}_2(A, B, K)$
 - $\text{OTMsg}_3(A, B, K) := \text{read OTMsg}_3(A, B, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)$

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

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

This is followed by the hiding of the channels

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

- $\left\{ \begin{array}{l} \text{SendOutShare}(B, A, k) := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{SendOutShare}(B, A, k) := \text{read SendOutShare}(B, A, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{SendOutShare}(A, B, k) := \text{read Share}(B, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{SendOutShare}(A, B, k) := \text{read SendOutShare}(A, B, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read SendOutShare}(B, A, k) \text{ for } k < K$
- $\text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read SendOutShare}(A, B, k) \text{ for } k < K$
- $\left\{ \begin{array}{l} \text{OutShare}(A, A, k) := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(A, A, k) := \text{read OutShare}(A, A, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$

- $\text{OutShare}(A, B, k) := \text{read SendOutShare}(A, B, k)$ for $k < K$
- $\text{OutShare}(B, A, k) := \text{read SendOutShare}(B, A, k)$ for $k < K$
- $\begin{cases} \text{OutShare}(B, B, k) := \text{read Share}(B, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(B, B, k) := \text{read OutShare}(B, B, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\text{OutShare}(A, A, k)_{\text{adv}}^A := \text{read OutShare}(A, A, k)$ for $k < K$
- $\text{OutShare}(A, B, k)_{\text{adv}}^A := \text{read OutShare}(A, B, k)$ for $k < K$
- $\text{Out}(A, k) := x_A \leftarrow \text{OutShare}(A, A, k); x_B \leftarrow \text{OutShare}(A, B, k); \text{ret } x_A \oplus x_B$ for $k < K$
- $\text{Out}(B, k) := x_A \leftarrow \text{OutShare}(B, A, k); x_B \leftarrow \text{OutShare}(B, B, k); \text{ret } x_A \oplus x_B$ for $k < K$
- $\text{Out}(A, k)_{\text{adv}}^A := \text{read Out}(A, k)$ for $k < K$

This is followed by the hiding of the channels

- $\text{OutShare}(A, A, k)$ for $k < K$,
- $\text{OutShare}(A, B, k)$ for $k < K$,
- $\text{OutShare}(B, A, k)$ for $k < K$,
- $\text{OutShare}(B, B, k)$ for $k < K$,
- $\text{SendOutShare}(B, A, k)$ for $k < K$,
- $\text{SendOutShare}(A, B, k)$ for $k < K$.

9.4.1 Simplifying The Real Protocol: Initial Phase

The internal channels $\text{SendInShare}(B, A, -)$, $\text{SendInShare}(A, B, -)$ can be substituted away, which yields the following simplified version Init of the initial part of the real protocol:

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

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

This is followed by the hiding of the channels

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

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

- $\text{OTMsg}_0(\mathbf{A}, \mathbf{B}, K)_{\text{adv}}^{\text{ot}} := \text{read } \text{OTMsg}_0(\mathbf{A}, \mathbf{B}, K)$

reads from the divergent channel

- $\text{OTMsg}_0(\mathbf{A}, \mathbf{B}, K) := \text{read } \text{OTMsg}_0(\mathbf{A}, \mathbf{B}, K)$

and so we may equivalently write the following:

- $\text{OTMsg}_0(\mathbf{A}, \mathbf{B}, K)_{\text{adv}}^{\text{ot}} := \text{read } \text{OTMsg}_0(\mathbf{A}, \mathbf{B}, K)_{\text{adv}}^{\text{ot}}$

Similarly, the channel

- $\text{OTChcRcvd}_0(\mathbf{A}, \mathbf{B}, K)_{\text{adv}}^{\text{ot}} := c_0 \leftarrow \text{OTChc}_0(\mathbf{A}, \mathbf{B}, K); \text{ ret } \checkmark$

reads from the divergent channel

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

and so we may equivalently write the following:

- $\text{OTChcRcvd}_0(\mathbf{A}, \mathbf{B}, K)_{\text{adv}}^{\text{ot}} := \text{read } \text{OTChcRcvd}_0(\mathbf{A}, \mathbf{B}, K)_{\text{adv}}^{\text{ot}}$

Finally, the channel

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

reads from the divergent channel

- $\text{OTMsg}_0(\mathbf{A}, \mathbf{B}, K) := \text{read } \text{OTMsg}_0(\mathbf{A}, \mathbf{B}, K)$

and so we may equivalently write the following:

- $\text{OTOut}(\mathbf{A}, \mathbf{B}, K) := \text{read } \text{OTOut}(\mathbf{A}, \mathbf{B}, K)$

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

- $\text{OTMsg}_0(\mathbf{A}, \mathbf{B}, K) := b_A \leftarrow \text{SendBit}(\mathbf{A}, \mathbf{B}, K); x_A \leftarrow \text{Share}(\mathbf{A}, k); y_A \leftarrow \text{Share}(\mathbf{A}, l); \text{ ret } b_A$

into the channel

- $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)$

yields the following:

- $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A$

Analogously, we have the following:

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

We treat the receiver channels similarly. For instance, substituting the channel

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

into the channel

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

yields the following:

- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := x_B \leftarrow \text{Share}(B, k); \text{ret } \checkmark$

Analogously, we have the following:

- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := y_B \leftarrow \text{Share}(B, l); \text{ret } \checkmark$

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

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

that stores the result of the OT exchange between Alice and Bob. Substituting the channel

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

into the above thus yields the somewhat more verbose

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

with the advantage that it no longer mentions $\text{OTOut}(A, B, K)$. We may further substitute the channels

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

as well as the channels

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

to obtain the following:

- $\text{RcvdBit}(B, A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{if } x_B \text{ then (if } y_B \text{ then ret } b_A \oplus x_A \oplus y_A \text{ else ret } b_A \oplus y_A) \text{ else (if } y_B \text{ then ret } b_A \oplus x_A \text{ else ret } b_A)$

Here we no longer refer to any of the OT channels. We can express the above more concisely as follows:

- $\text{RcvdBit}(B, A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$

Summarizing, we can rewrite the inductive part $\text{Circ}(C, K)$ of the real protocol as follows:

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

- $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTChc}_0(A, B, K) := \text{read OTChc}_0(A, B, K)$
- $\text{OTChc}_1(A, B, K) := \text{read OTChc}_1(A, B, K)$
- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTOut}(A, B, K) := \text{read OTOut}(A, B, K)$
- $\text{Circ}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Circ}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read Share}(A, k)$
 - $\text{Share}(B, K) := x_B \leftarrow \text{Share}(B, k); \text{ret } \neg x_B$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K) := \text{read OTMsg}_0(A, B, K)$
 - $\text{OTMsg}_1(A, B, K) := \text{read OTMsg}_1(A, B, K)$
 - $\text{OTMsg}_2(A, B, K) := \text{read OTMsg}_2(A, B, K)$
 - $\text{OTMsg}_3(A, B, K) := \text{read OTMsg}_3(A, B, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChc}_0(A, B, K) := \text{read OTChc}_0(A, B, K)$
 - $\text{OTChc}_1(A, B, K) := \text{read OTChc}_1(A, B, K)$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTOut}(A, B, K) := \text{read OTOut}(A, B, K)$
- $\text{Circ}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Circ}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } x_A \oplus y_A$
 - $\text{Share}(B, K) := x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } x_B \oplus y_B$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K) := \text{read OTMsg}_0(A, B, K)$
 - $\text{OTMsg}_1(A, B, K) := \text{read OTMsg}_1(A, B, K)$
 - $\text{OTMsg}_2(A, B, K) := \text{read OTMsg}_2(A, B, K)$

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

We now split the protocol $\text{Circ}(C, K)$ into three parts. The first protocol $\text{Shares}(C, K)$ performs the computation of shares and is defined as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$

- $\text{Share}(A, K) := \text{read InShare}(A, A, i)$
- $\text{Share}(B, K) := \text{read InShare}(B, A, i)$
- $\text{Shares}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read InShare}(A, B, i)$
 - $\text{Share}(B, K) := \text{read InShare}(B, B, i)$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read Share}(A, k)$
 - $\text{Share}(B, K) := x_B \leftarrow \text{Share}(B, k); \text{ret } \neg x_B$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } x_A \oplus y_A$
 - $\text{Share}(B, K) := x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } x_B \oplus y_B$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{samp flip}$
 - $\text{RcvdBit}(B, A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l);$
 $x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
 - $\text{Share}(A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } (x_A * y_A) \oplus b_A$
 - $\text{Share}(B, K) := b_B \leftarrow \text{RcvdBit}(B, A, K); x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_B \oplus (x_B * y_B)$

The second protocol $\text{Adv}(C, K)$ performs all leakages and is defined as follows:

- $\text{Adv}(\epsilon, 0)$ is the protocol 0
- $\text{Adv}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$

- $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus x_A$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus y_A$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus x_A \oplus y_A$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := x_B \leftarrow \text{Share}(B, k); \text{ret } \checkmark$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := y_B \leftarrow \text{Share}(B, l); \text{ret } \checkmark$

The third protocol $1\text{OutOf4OT}(C, K)$ performs all Oblivious Transfer exchanges and is defined as follows:

- $1\text{OutOf4OT}(\epsilon, 0)$ is the protocol 0
- $1\text{OutOf4OT}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $1\text{OutOf4OT}(C, K)$ with the protocol
 - $\text{OTMsg}_0(A, B, K) := \text{read OTMsg}_0(A, B, K)$
 - $\text{OTMsg}_1(A, B, K) := \text{read OTMsg}_1(A, B, K)$
 - $\text{OTMsg}_2(A, B, K) := \text{read OTMsg}_2(A, B, K)$

- $\text{OTMsg}_3(A, B, K) := \text{read OTMsg}_3(A, B, K)$
- $\text{OTChc}_0(A, B, K) := \text{read OTChc}_0(A, B, K)$
- $\text{OTChc}_1(A, B, K) := \text{read OTChc}_1(A, B, K)$
- $\text{OTOut}(A, B, K) := \text{read OTOut}(A, B, K)$
- $1\text{OutOf4OT}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $1\text{OutOf4OT}(C, K)$ with the protocol
 - $\text{OTMsg}_0(A, B, K) := \text{read OTMsg}_0(A, B, K)$
 - $\text{OTMsg}_1(A, B, K) := \text{read OTMsg}_1(A, B, K)$
 - $\text{OTMsg}_2(A, B, K) := \text{read OTMsg}_2(A, B, K)$
 - $\text{OTMsg}_3(A, B, K) := \text{read OTMsg}_3(A, B, K)$
 - $\text{OTChc}_0(A, B, K) := \text{read OTChc}_0(A, B, K)$
 - $\text{OTChc}_1(A, B, K) := \text{read OTChc}_1(A, B, K)$
 - $\text{OTOut}(A, B, K) := \text{read OTOut}(A, B, K)$
- $1\text{OutOf4OT}(C; \text{not-gate}(k), K + 1)$ is the composition of $1\text{OutOf4OT}(C, K)$ with the protocol
 - $\text{OTMsg}_0(A, B, K) := \text{read OTMsg}_0(A, B, K)$
 - $\text{OTMsg}_1(A, B, K) := \text{read OTMsg}_1(A, B, K)$
 - $\text{OTMsg}_2(A, B, K) := \text{read OTMsg}_2(A, B, K)$
 - $\text{OTMsg}_3(A, B, K) := \text{read OTMsg}_3(A, B, K)$
 - $\text{OTChc}_0(A, B, K) := \text{read OTChc}_0(A, B, K)$
 - $\text{OTChc}_1(A, B, K) := \text{read OTChc}_1(A, B, K)$
 - $\text{OTOut}(A, B, K) := \text{read OTOut}(A, B, K)$
- $1\text{OutOf4OT}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $1\text{OutOf4OT}(C, K)$ with the protocol
 - $\text{OTMsg}_0(A, B, K) := \text{read OTMsg}_0(A, B, K)$
 - $\text{OTMsg}_1(A, B, K) := \text{read OTMsg}_1(A, B, K)$
 - $\text{OTMsg}_2(A, B, K) := \text{read OTMsg}_2(A, B, K)$
 - $\text{OTMsg}_3(A, B, K) := \text{read OTMsg}_3(A, B, K)$
 - $\text{OTChc}_0(A, B, K) := \text{read OTChc}_0(A, B, K)$
 - $\text{OTChc}_1(A, B, K) := \text{read OTChc}_1(A, B, K)$
 - $\text{OTOut}(A, B, K) := \text{read OTOut}(A, B, K)$
- $1\text{OutOf4OT}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $1\text{OutOf4OT}(C, K)$ with the protocol
 - $\text{OTMsg}_0(A, B, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A$
 - $\text{OTMsg}_1(A, B, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus x_A$
 - $\text{OTMsg}_2(A, B, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus y_A$
 - $\text{OTMsg}_3(A, B, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus x_A \oplus y_A$
 - $\text{OTChc}_0(A, B, K) := \text{read Share}(B, k)$
 - $\text{OTChc}_1(A, B, K) := \text{read Share}(B, l)$
 - $\text{OTOut}(A, B, K) := m_0 \leftarrow \text{OTMsg}_0(A, B, K); m_1 \leftarrow \text{OTMsg}_1(A, B, K); m_2 \leftarrow \text{OTMsg}_2(A, B, K);$
 $m_3 \leftarrow \text{OTMsg}_3(A, B, K); c_0 \leftarrow \text{OTChc}_0(A, B, K); c_1 \leftarrow \text{OTChc}_1(A, B, K);$
 if c_0 then (if c_1 then ret m_3 else ret m_2) else (if c_1 then ret m_1 else ret m_0)

At this point, none of the channels defined by $1\text{OutOf4OT}(C, K)$ are utilized anywhere outside of $1\text{OutOf4OT}(C, K)$ and as such we may discard this protocol entirely. The inductive part of the real protocol therefore consists of the protocols $\text{Shares}(C, K)$ and $\text{Adv}(C, K)$, followed by the hiding of the channels

- $\text{SendBit}(A, B, k)$ for $k < K$,
- $\text{RcvdBit}(B, A, k)$ for $k < K$.

9.4.3 Simplifying The Real Protocol: Final Phase

If wire k is not an output, then the channels

- $\text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read SendOutShare}(B, A, k)$
- $\text{OutShare}(B, A, k) := \text{read SendOutShare}(B, A, k)$

read from the divergent channel

- $\text{SendOutShare}(B, A, k) := \text{read SendOutShare}(B, A, k)$

and thus we may equivalently write the following:

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

If wire k is an output, then the channels

- $\text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read SendOutShare}(B, A, k)$
- $\text{OutShare}(B, A, k) := \text{read SendOutShare}(B, A, k)$

read from the channel

- $\text{SendOutShare}(B, A, k) := \text{read Share}(A, k)$

so we may substitute:

- $\text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read Share}(A, k)$
- $\text{OutShare}(B, A, k) := \text{read Share}(A, k)$

Summarizing the above, we get the following for channels $\text{SendOutShare}(B, A, -)_{\text{adv}}^A$ and $\text{OutShare}(B, A, -)$:

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

If wire k is not an output, then the channels

- $\text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read SendOutShare}(A, B, k)$
- $\text{OutShare}(A, B, k) := \text{read SendOutShare}(A, B, k)$

read from the divergent channel

- $\text{SendOutShare}(A, B, k) := \text{read SendOutShare}(A, B, k)$

and thus we may equivalently write the following:

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

If wire k is an output, then the channels

- $\text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read SendOutShare}(A, B, k)$
- $\text{OutShare}(A, B, k) := \text{read SendOutShare}(A, B, k)$

read from the channel

- $\text{SendOutShare}(A, B, k) := \text{read Share}(B, k)$

so we may substitute:

- $\text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read Share}(B, k)$
- $\text{OutShare}(A, B, k) := \text{read Share}(B, k)$

Summarizing the above, we get the following for channels $\text{RcvdOutShare}(A, B, k)_{\text{adv}}^A$ and $\text{OutShare}(A, B, -)$:

- $\left\{ \begin{array}{l} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read Share}(B, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read RcvdOutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(A, B, k) := \text{read Share}(B, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(A, B, k) := \text{read OutShare}(A, B, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$

At this point, the internal channels $\text{SendOutShare}(B, A, -)$, $\text{SendOutShare}(A, B, -)$ are unused and can be eliminated. Our simplified version Fin of the final part of the real protocol is therefore as follows:

- $\left\{ \begin{array}{l} \text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read SendOutShare}(B, A, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read Share}(B, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read RcvdOutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(A, A, k) := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(A, A, k) := \text{read OutShare}(A, A, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(A, B, k) := \text{read Share}(B, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(A, B, k) := \text{read OutShare}(A, B, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(B, A, k) := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(B, A, k) := \text{read OutShare}(B, A, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$

- $\begin{cases} \text{OutShare}(B, B, k) := \text{read Share}(B, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(B, B, k) := \text{read OutShare}(B, B, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\text{OutShare}(A, A, k)_{\text{adv}}^A := \text{read OutShare}(A, A, k) \text{ for } k < K$
- $\text{OutShare}(A, B, k)_{\text{adv}}^A := \text{read OutShare}(A, B, k) \text{ for } k < K$
- $\text{Out}(A, k) := x_A \leftarrow \text{OutShare}(A, A, k); x_B \leftarrow \text{OutShare}(A, B, k); \text{ret } x_A \oplus x_B \text{ for } k < K$
- $\text{Out}(B, k) := x_A \leftarrow \text{OutShare}(B, A, k); x_B \leftarrow \text{OutShare}(B, B, k); \text{ret } x_A \oplus x_B \text{ for } k < K$
- $\text{Out}(A, k)_{\text{adv}}^A := \text{read Out}(A, k) \text{ for } k < K$

This is followed by the hiding of the channels

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

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

- $\text{InShare-}\checkmark(A, A, i) := _ \leftarrow \text{InShare}(A, A, i); \text{ret } \checkmark \text{ for } i < N$
- $\text{InShare-}\checkmark(A, B, i) := _ \leftarrow \text{InShare}(A, B, i); \text{ret } \checkmark \text{ for } i < M$
- $\text{InShare-}\checkmark(B, A, i) := _ \leftarrow \text{InShare}(B, A, i); \text{ret } \checkmark \text{ for } i < N$
- $\text{InShare-}\checkmark(B, B, i) := _ \leftarrow \text{InShare}(B, B, i); \text{ret } \checkmark \text{ for } i < M$

for the timing of input shares in the initial part of the protocol and

- $\text{Share-}\checkmark(A, k) := _ \leftarrow \text{Share}(A, k); \text{ret } \checkmark \text{ for } k < K$
- $\text{Share-}\checkmark(B, k) := _ \leftarrow \text{Share}(B, k); \text{ret } \checkmark \text{ for } k < K$

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 $\text{Adv}(C, K)$ in the case of an *and* gate. We can write the channels

- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := x_B \leftarrow \text{Share}(B, k); \text{ret } \checkmark$

- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := y_B \leftarrow \text{Share}(B, l); \text{ret } \checkmark$

equivalently as follows:

- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(B, k)$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(B, l)$

So the inductive part of the real protocol now has $\text{Adv}(C, K)$ looking like so:

- $\text{Adv}(\epsilon, 0)$ is the protocol 0
- $\text{Adv}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$

- $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus x_A$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus y_A$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus x_A \oplus y_A$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(B, k)$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(B, l)$

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 $0, \dots, N$. To this end, we introduce new internal channels

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

to keep track of the timing information in the protocol Init and

- $\text{SendBit-}\checkmark(A, B, k)$ for $k < K$
- $\text{RcvdBit-}\checkmark(A, B, k)$ for $k < K$

to keep track of the timing information in the protocol $\text{Shares}(C, K)$. Call the following protocol fragment $\text{Init-}\checkmark$:

- $\text{InShare-}\$-\checkmark(A, A, i)$ for $i < N$
- $\text{InShare-}\$-\checkmark(A, B, i)$ for $i < M$
- $\text{InShare-}\$-\checkmark(B, A, i)$ for $i < N$
- $\text{InShare-}\$-\checkmark(B, B, i)$ for $i < M$
- $\text{InShare-}\checkmark(A, A, i) := _ \leftarrow \text{InShare}(A, A, i); \text{ret } \checkmark$ for $i < N$
- $\text{InShare-}\checkmark(A, B, i) := _ \leftarrow \text{InShare}(A, B, i); \text{ret } \checkmark$ for $i < M$
- $\text{InShare-}\checkmark(B, A, i) := _ \leftarrow \text{InShare}(B, A, i); \text{ret } \checkmark$ for $i < N$
- $\text{InShare-}\checkmark(B, B, i) := _ \leftarrow \text{InShare}(B, B, i); \text{ret } \checkmark$ for $i < M$

Call the following protocol fragment $\text{Shares-}\checkmark(C, K)$:

- $\text{SendBit-}\checkmark(A, B, k)$ for $k < K$
- $\text{RcvdBit-}\checkmark(A, B, k)$ for $k < K$

- $\text{Share-}\checkmark(A, k) := _ \leftarrow \text{Share}(A, k); \text{ret } \checkmark \text{ for } k < K$
- $\text{Share-}\checkmark(B, k) := _ \leftarrow \text{Share}(B, k); \text{ret } \checkmark \text{ for } k < K$

In the presence of the channels $\text{InShare-}\checkmark(A, A, -)$, $\text{InShare-}\checkmark(A, B, -)$, $\text{InShare-}\checkmark(B, A, -)$, $\text{InShare-}\checkmark(B, B, -)$ and the protocol $\text{Shares-}\checkmark(C, K)$ we can express the protocol $\text{Shares}(C, K)$ equivalently as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read InShare}(A, A, i)$
 - $\text{Share}(B, K) := _ \leftarrow \text{InShare-}\checkmark(B, A, i); \text{read InShare}(B, A, i)$
- $\text{Shares}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read InShare}(A, B, i)$
 - $\text{Share}(B, K) := _ \leftarrow \text{InShare-}\checkmark(B, B, i); \text{read InShare}(B, B, i)$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read Share}(A, k)$
 - $\text{Share}(B, K) := _ \leftarrow \text{Share-}\checkmark(B, k); x_B \leftarrow \text{Share}(B, k); \text{ret } \neg x_B$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } x_A \oplus y_A$
 - $\text{Share}(B, K) := _ \leftarrow \text{Share-}\checkmark(B, k); _ \leftarrow \text{Share-}\checkmark(B, l); x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } x_B \oplus y_B$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{samp flip}$
 - $\text{RcvdBit}(B, A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l);$
 $x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
 - $\text{Share}(A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } (x_A * y_A) \oplus b_A$
 - $\text{Share}(B, K) := _ \leftarrow \text{RcvdBit-}\checkmark(A, B, K); _ \leftarrow \text{Share-}\checkmark(B, k); _ \leftarrow \text{Share-}\checkmark(B, l);$
 $b_B \leftarrow \text{RcvdBit}(B, A, K); x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_B \oplus (x_B * y_B)$

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

- $\text{In-}\checkmark(A, i) := _ \leftarrow \text{read In}(A, i); \text{ret } \checkmark \text{ for } i < N$
- $\text{In-}\checkmark(B, i) := _ \leftarrow \text{read In}(B, i); \text{ret } \checkmark \text{ for } i < M$

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 $\text{Wires-}\checkmark(C, K)$:

- $\text{Wires-}\checkmark(\epsilon, 0)$ is the protocol 0
- $\text{Wires-}\checkmark(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := \text{read In-}\checkmark(A, i)$
- $\text{Wires-}\checkmark(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := \text{read In-}\checkmark(B, i)$
- $\text{Wires-}\checkmark(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := _ \leftarrow \text{Wire-}\checkmark(k); \text{ret } \checkmark$
- $\text{Wires-}\checkmark(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark$
- $\text{Wires-}\checkmark(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark$

Our goal now is to show that the timing channels can be equivalently characterized as follows:

- $\text{InShare-}\checkmark(A, A, i) := \text{read In-}\checkmark(A, i)$ for $i < N$
- $\text{InShare-}\checkmark(A, B, i) := \text{read In-}\checkmark(B, i)$ for $i < M$
- $\text{InShare-}\checkmark(B, A, i) := \text{read In-}\checkmark(A, i)$ for $i < N$
- $\text{InShare-}\checkmark(B, B, i) := \text{read In-}\checkmark(B, i)$ for $i < M$
- $\text{Share-}\checkmark(A, k) := \text{read Wire-}\checkmark(k)$ for $k < K$
- $\text{Share-}\checkmark(B, k) := \text{read Wire-}\checkmark(k)$ for $k < K$

In the presence of the channels $\text{InShare-}\checkmark(A, A, -)$, $\text{InShare-}\checkmark(A, B, -)$, $\text{InShare-}\checkmark(B, A, -)$, $\text{InShare-}\checkmark(B, B, -)$ and the protocol $\text{Shares}(C, K)$ we can define the protocol $\text{Shares-}\checkmark(C, K)$ equivalently as follows:

- $\text{Shares-}\checkmark(\epsilon, 0)$ is the protocol 0
- $\text{Shares-}\checkmark(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
 - $\text{Share-}\checkmark(A, K) := \text{read InShare-}\checkmark(A, A, i)$
 - $\text{Share-}\checkmark(B, K) := \text{read InShare-}\checkmark(B, A, i)$
- $\text{Shares-}\checkmark(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
 - $\text{Share-}\checkmark(A, K) := \text{read InShare-}\checkmark(A, B, i)$
 - $\text{Share-}\checkmark(B, K) := \text{read InShare-}\checkmark(B, B, i)$
- $\text{Shares-}\checkmark(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
 - $\text{Share-}\checkmark(A, K) := \text{read Share-}\checkmark(A, k)$

- $\text{Share-}\checkmark(B, K) := _ \leftarrow \text{Share-}\checkmark(B, k); \text{ret } \checkmark$
- $\text{Shares-}\checkmark(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
 - $\text{Share-}\checkmark(A, K) := _ \leftarrow \text{Share-}\checkmark(A, k); _ \leftarrow \text{Share-}\checkmark(A, l); \text{ret } \checkmark$
 - $\text{Share-}\checkmark(B, K) := _ \leftarrow \text{Share-}\checkmark(B, k); _ \leftarrow \text{Share-}\checkmark(B, l); \text{ret } \checkmark$
- $\text{Shares-}\checkmark(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := _ \leftarrow \text{Share-}\checkmark(A, k); _ \leftarrow \text{Share-}\checkmark(A, l); \text{ret } \checkmark$
 - $\text{RcvdBit-}\checkmark(A, B, K) := _ \leftarrow \text{SendBit-}\checkmark(A, B, K); _ \leftarrow \text{Share-}\checkmark(A, k); _ \leftarrow \text{Share-}\checkmark(A, l);$
 $_ \leftarrow \text{Share-}\checkmark(B, k); _ \leftarrow \text{Share-}\checkmark(B, l); \text{ret } \checkmark$
 - $\text{Share-}\checkmark(A, K) := _ \leftarrow \text{SendBit-}\checkmark(A, B, K); _ \leftarrow \text{Share-}\checkmark(A, k); _ \leftarrow \text{Share-}\checkmark(A, l); \text{ret } \checkmark$
 - $\text{Share-}\checkmark(B, K) := _ \leftarrow \text{RcvdBit-}\checkmark(A, B, K); _ \leftarrow \text{Share-}\checkmark(B, k); _ \leftarrow \text{Share-}\checkmark(B, l); \text{ret } \checkmark$

We also observe that in the presence of the protocol Init we can define the protocol $\text{Init-}\checkmark$ equivalently as follows:

- $\text{InShare-}\$-\checkmark(A, A, i) := _ \leftarrow \text{In-}\checkmark(A, i); \text{ret } \checkmark \text{ for } i < N$
- $\text{InShare-}\$-\checkmark(A, B, i) := _ \leftarrow \text{In-}\checkmark(B, i); \text{ret } \checkmark \text{ for } i < M$
- $\text{InShare-}\$-\checkmark(B, A, i) := _ \leftarrow \text{In-}\checkmark(A, i); _ \leftarrow \text{InShare-}\$-\checkmark(A, A, i); \text{ret } \checkmark \text{ for } i < N$
- $\text{InShare-}\$-\checkmark(B, B, i) := _ \leftarrow \text{In-}\checkmark(B, i); _ \leftarrow \text{InShare-}\$-\checkmark(A, B, i); \text{ret } \checkmark \text{ for } i < M$
- $\text{InShare-}\checkmark(A, A, i) := \text{read InShare-}\$-\checkmark(A, A, i) \text{ for } i < N$
- $\text{InShare-}\checkmark(A, B, i) := \text{read InShare-}\$-\checkmark(A, B, i) \text{ for } i < M$
- $\text{InShare-}\checkmark(B, A, i) := \text{read InShare-}\$-\checkmark(B, A, i) \text{ for } i < N$
- $\text{InShare-}\checkmark(B, B, i) := \text{read InShare-}\$-\checkmark(B, B, i) \text{ for } i < M$

Furthermore, in the presence of the channels $\text{In-}\checkmark(A, -)$, $\text{In-}\checkmark(B, -)$ we can express the protocol $\text{Init-}\checkmark$ equivalently as follows:

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

Lastly, in the presence of the channels $\text{InShare-}\checkmark(A, A, -)$, $\text{InShare-}\checkmark(A, B, -)$, $\text{InShare-}\checkmark(B, A, -)$, $\text{InShare-}\checkmark(B, B, -)$ as well as the protocol $\text{Wires-}\checkmark(C, K)$ we can express the protocol $\text{Shares-}\checkmark(C, K)$ equivalently as the channels

- $\text{Share-}\checkmark(A, k) := \text{read Wire-}\checkmark(k) \text{ for } k < K$
- $\text{Share-}\checkmark(B, k) := \text{read Wire-}\checkmark(k) \text{ for } k < K$

together with the following protocol $\text{SendRcvdBits-}\checkmark(C, K)$:

- $\text{SendRcvdBits-}\checkmark(\epsilon, 0)$ is the protocol 0
- $\text{SendRcvdBits-}\checkmark(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{SendRcvdBits-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
- $\text{SendRcvdBits-}\checkmark(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{SendRcvdBits-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
- $\text{SendRcvdBits-}\checkmark(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{SendRcvdBits-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
- $\text{SendRcvdBits-}\checkmark(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{SendRcvdBits-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
- $\text{SendRcvdBits-}\checkmark(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{SendRcvdBits-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark$
 - $\text{RcvdBit-}\checkmark(A, B, K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark$

We now revisit the case of an *and* gate in the protocol $\text{Adv}(C, K)$. We can write the channels

- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(B, k)$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(B, l)$

equivalently as follows:

- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(k)$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(l)$

So the inductive part of the real protocol now has $\text{Adv}(C, K)$ looking like so:

- $\text{Adv}(\epsilon, 0)$ is the protocol 0
- $\text{Adv}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol

- $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
- $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
- $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus x_A$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus y_A$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus x_A \oplus y_A$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(k)$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(l)$

In the presence of the channels $\text{InShare-}\checkmark(A, A, -)$, $\text{InShare-}\checkmark(A, B, -)$, $\text{InShare-}\checkmark(B, A, -)$, $\text{InShare-}\checkmark(B, B, -)$ and the channels $\text{Share-}\checkmark(A, -)$, $\text{Share-}\checkmark(B, -)$ as well as the protocols $\text{Wires-}\checkmark$ and $\text{Shares-}\checkmark$ we can express the protocol $\text{Shares}(C, K)$ equivalently as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0

- $\text{Shares}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read InShare}(A, A, i)$
 - $\text{Share}(B, K) := _ \leftarrow \text{Wire-}\checkmark(K); \text{read InShare}(B, A, i)$
- $\text{Shares}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read InShare}(A, B, i)$
 - $\text{Share}(B, K) := _ \leftarrow \text{Wire-}\checkmark(K); \text{read InShare}(B, B, i)$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read Share}(A, k)$
 - $\text{Share}(B, K) := _ \leftarrow \text{Wire-}\checkmark(K); x_B \leftarrow \text{Share}(B, k); \text{ret } \neg x_B$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } x_A \oplus y_A$
 - $\text{Share}(B, K) := _ \leftarrow \text{Wire-}\checkmark(K); x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } x_B \oplus y_B$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{samp flip}$
 - $\text{RcvdBit}(B, A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l);$
 $x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
 - $\text{Share}(A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } (x_A * y_A) \oplus b_A$
 - $\text{Share}(B, K) := _ \leftarrow \text{Wire-}\checkmark(K); b_B \leftarrow \text{RcvdBit}(B, A, K);$
 $x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_B \oplus (x_B * y_B)$

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 $\text{Adv}(C, K)$ in the case of an *and* gate. We want to write the channels

- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := x_B \leftarrow \text{Share}(B, k); \text{ret } \checkmark$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := y_B \leftarrow \text{Share}(B, l); \text{ret } \checkmark$

in the form below:

- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := x_A \leftarrow \text{Share}(A, k); \text{ret } \checkmark$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := x_A \leftarrow \text{Share}(A, l); \text{ret } \checkmark$

So the inductive part of the real protocol now has $\text{Adv}(C, K)$ looking like so:

- $\text{Adv}(\epsilon, 0)$ is the protocol 0

- $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
- $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A$
- $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus x_A$
- $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus y_A$
- $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus x_A \oplus y_A$
- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := x_A \leftarrow \text{Share}(A, k); \text{ret } \checkmark$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := x_A \leftarrow \text{Share}(A, l); \text{ret } \checkmark$

We now amend Bob's shares with a gratuitous dependency on Alice's shares. Specifically, we modify the protocol $\text{Shares}(C, K)$ as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read InShare}(A, A, i)$
 - $\text{Share}(B, K) := x_A \leftarrow \text{Share}(A, K); \text{read InShare}(B, A, i)$
- $\text{Shares}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read InShare}(A, B, i)$
 - $\text{Share}(B, K) := x_A \leftarrow \text{Share}(A, K); \text{read InShare}(B, B, i)$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read Share}(A, k)$
 - $\text{Share}(B, K) := x_A \leftarrow \text{Share}(A, K); x_B \leftarrow \text{Share}(B, k); \text{ret } \neg x_B$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } x_A \oplus y_A$
 - $\text{Share}(B, K) := x_A \leftarrow \text{Share}(A, K); x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } x_B \oplus y_B$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{samp flip}$
 - $\text{RcvdBit}(B, A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l);$
 $x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
 - $\text{Share}(A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } (x_A * y_A) \oplus b_A$
 - $\text{Share}(B, K) := x_A \leftarrow \text{Share}(A, K); b_B \leftarrow \text{RcvdBit}(B, A, K);$
 $x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_B \oplus (x_B * y_B)$

It is not at all clear that the aforementioned amendments of $\text{Adv}(C, K)$ and $\text{Shares}(C, K)$ are sound. In the rest of this section we justify their soundness.

We start by introducing the channels

- $\text{InShare-}\checkmark(A, A, i) := _ \leftarrow \text{InShare}(A, A, i); \text{ ret } \checkmark \text{ for } i < N$
- $\text{InShare-}\checkmark(A, B, i) := _ \leftarrow \text{InShare}(A, B, i); \text{ ret } \checkmark \text{ for } i < M$
- $\text{InShare-}\checkmark(B, A, i) := _ \leftarrow \text{InShare}(B, A, i); \text{ ret } \checkmark \text{ for } i < N$
- $\text{InShare-}\checkmark(B, B, i) := _ \leftarrow \text{InShare}(B, B, i); \text{ ret } \checkmark \text{ for } i < M$

for the timing of input shares in the initial part of the protocol and

- $\text{Share-}\checkmark(A, k) := _ \leftarrow \text{Share}(A, k); \text{ ret } \checkmark \text{ for } k < K$
- $\text{Share-}\checkmark(B, k) := _ \leftarrow \text{Share}(B, k); \text{ ret } \checkmark \text{ for } k < K$

for the timing of shares in the inductive part of the protocol.

Take the protocol $\text{Adv}(C, K)$ in the case of an *and* gate. We can write the channels

- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := x_B \leftarrow \text{Share}(B, k); \text{ ret } \checkmark$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := y_B \leftarrow \text{Share}(B, l); \text{ ret } \checkmark$

equivalently as follows:

- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(A, k)$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(A, l)$

So the inductive part of the real protocol now has $\text{Adv}(C, K)$ looking like so:

- $\text{Adv}(\epsilon, 0)$ is the protocol 0
- $\text{Adv}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$

- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus x_A$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus y_A$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus x_A \oplus y_A$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(A, k)$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(A, l)$

We now amend Bob's shares further: in the presence of the channels $\text{Share-}\checkmark(A, -)$ we can turn the gratuitous dependency on Alice's shares into a dependency on the corresponding timing channel. Specifically, we can express the protocol $\text{Shares}(C, K)$ equivalently as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read InShare}(A, A, i)$
 - $\text{Share}(B, K) := _ \leftarrow \text{Share-}\checkmark(A, K); \text{read InShare}(B, A, i)$
- $\text{Shares}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol

- $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
- $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
- $\text{Share}(A, K) := \text{read InShare}(A, B, i)$
- $\text{Share}(B, K) := _ \leftarrow \text{Share-}\checkmark(A, K); \text{read InShare}(B, B, i)$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read Share}(A, k)$
 - $\text{Share}(B, K) := _ \leftarrow \text{Share-}\checkmark(A, K); x_B \leftarrow \text{Share}(B, k); \text{ret } \neg x_B$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } x_A \oplus y_A$
 - $\text{Share}(B, K) := _ \leftarrow \text{Share-}\checkmark(A, K); x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } x_B \oplus y_B$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{samp flip}$
 - $\text{RcvdBit}(B, A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l);$
 $x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
 - $\text{Share}(A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } (x_A * y_A) \oplus b_A$
 - $\text{Share}(B, K) := _ \leftarrow \text{Share-}\checkmark(A, K); b_B \leftarrow \text{RcvdBit}(B, A, K);$
 $x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_B \oplus (x_B * y_B)$

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

- $\text{In-}\checkmark(A, i) := _ \leftarrow \text{read In}(A, i); \text{ret } \checkmark \text{ for } i < N$
- $\text{In-}\checkmark(B, i) := _ \leftarrow \text{read In}(B, i); \text{ret } \checkmark \text{ for } i < M$

to keep track of whether an input has arrived, and a new protocol $\text{Wires-}\checkmark(C, K)$ that keeps track of the timing of wire shares:

- $\text{Wires-}\checkmark(\epsilon, 0)$ is the protocol 0
- $\text{Wires-}\checkmark(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := \text{read In-}\checkmark(A, i)$
- $\text{Wires-}\checkmark(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := \text{read In-}\checkmark(B, i)$
- $\text{Wires-}\checkmark(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := _ \leftarrow \text{Wire-}\checkmark(k); \text{ret } \checkmark$
- $\text{Wires-}\checkmark(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark$
- $\text{Wires-}\checkmark(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol

– $\text{Wire-}\checkmark(K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark$

Our goal is again to show that the timing channels can be equivalently characterized as follows:

- $\text{InShare-}\checkmark(A, A, i) := \text{read In-}\checkmark(A, i)$ for $i < N$
- $\text{InShare-}\checkmark(A, B, i) := \text{read In-}\checkmark(B, i)$ for $i < M$
- $\text{InShare-}\checkmark(B, A, i) := \text{read In-}\checkmark(A, i)$ for $i < N$
- $\text{InShare-}\checkmark(B, B, i) := \text{read In-}\checkmark(B, i)$ for $i < M$
- $\text{Share-}\checkmark(A, k) := \text{read Wire-}\checkmark(k)$ for $k < K$
- $\text{Share-}\checkmark(B, k) := \text{read Wire-}\checkmark(k)$ for $k < K$

To this end, we introduce new internal channels

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

to keep track of the timing information in the protocol Init and

- $\text{SendBit-}\checkmark(A, B, k)$ for $k < K$
- $\text{RcvdBit-}\checkmark(A, B, k)$ for $k < K$

to keep track of the timing information in the protocol $\text{Circ}(C, K)$. Call the following protocol fragment $\text{Init-}\checkmark$:

- $\text{InShare-}\$-\checkmark(A, A, i)$ for $i < N$
- $\text{InShare-}\$-\checkmark(A, B, i)$ for $i < M$
- $\text{InShare-}\$-\checkmark(B, A, i)$ for $i < N$
- $\text{InShare-}\$-\checkmark(B, B, i)$ for $i < M$
- $\text{InShare-}\checkmark(A, A, i) := _ \leftarrow \text{InShare}(A, A, i); \text{ret } \checkmark$ for $i < N$
- $\text{InShare-}\checkmark(A, B, i) := _ \leftarrow \text{InShare}(A, B, i); \text{ret } \checkmark$ for $i < M$
- $\text{InShare-}\checkmark(B, A, i) := _ \leftarrow \text{InShare}(B, A, i); \text{ret } \checkmark$ for $i < N$
- $\text{InShare-}\checkmark(B, B, i) := _ \leftarrow \text{InShare}(B, B, i); \text{ret } \checkmark$ for $i < M$

Call the following protocol fragment $\text{Shares-}\checkmark(C, K)$:

- $\text{SendBit-}\checkmark(A, B, k)$ for $k < K$
- $\text{RcvdBit-}\checkmark(A, B, k)$ for $k < K$
- $\text{Share-}\checkmark(A, k) := _ \leftarrow \text{Share}(A, k); \text{ret } \checkmark$ for $k < K$
- $\text{Share-}\checkmark(B, k) := _ \leftarrow \text{Share}(B, k); \text{ret } \checkmark$ for $k < K$

In the presence of the channels $\text{InShare-}\checkmark(A, A, -)$, $\text{InShare-}\checkmark(A, B, -)$, $\text{InShare-}\checkmark(B, A, -)$, $\text{InShare-}\checkmark(B, B, -)$ as well as the protocol $\text{Shares}(C, K)$ we can define the protocol $\text{Shares-}\checkmark(C, K)$ equivalently as follows:

- $\text{Shares-}\checkmark(\epsilon, 0)$ is the protocol 0

- $\text{Shares-}\checkmark(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
 - $\text{Share-}\checkmark(A, K) := \text{read InShare-}\checkmark(A, A, i)$
 - $\text{Share-}\checkmark(B, K) := _ \leftarrow \text{Share-}\checkmark(A, K); \text{read InShare-}\checkmark(B, A, i)$
- $\text{Shares-}\checkmark(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
 - $\text{Share-}\checkmark(A, K) := \text{read InShare-}\checkmark(A, B, i)$
 - $\text{Share-}\checkmark(B, K) := _ \leftarrow \text{Share-}\checkmark(A, K); \text{read InShare-}\checkmark(B, B, i)$
- $\text{Shares-}\checkmark(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
 - $\text{Share-}\checkmark(A, K) := \text{read Share-}\checkmark(A, k)$
 - $\text{Share-}\checkmark(B, K) := _ \leftarrow \text{Share-}\checkmark(A, K); _ \leftarrow \text{Share-}\checkmark(B, k); \text{ret } \checkmark$
- $\text{Shares-}\checkmark(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
 - $\text{Share-}\checkmark(A, K) := _ \leftarrow \text{Share-}\checkmark(A, k); _ \leftarrow \text{Share-}\checkmark(A, l); \text{ret } \checkmark$
 - $\text{Share-}\checkmark(B, K) := _ \leftarrow \text{Share-}\checkmark(A, K); _ \leftarrow \text{Share-}\checkmark(B, k); _ \leftarrow \text{Share-}\checkmark(B, l); \text{ret } \checkmark$
- $\text{Shares-}\checkmark(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := _ \leftarrow \text{Share-}\checkmark(A, k); _ \leftarrow \text{Share-}\checkmark(A, l); \text{ret } \checkmark$
 - $\text{RcvdBit-}\checkmark(A, B, K) := _ \leftarrow \text{SendBit-}\checkmark(A, B, K); _ \leftarrow \text{Share-}\checkmark(A, k); _ \leftarrow \text{Share-}\checkmark(A, l);$
 $_ \leftarrow \text{Share-}\checkmark(B, k); _ \leftarrow \text{Share-}\checkmark(B, l); \text{ret } \checkmark$
 - $\text{Share-}\checkmark(A, K) := _ \leftarrow \text{SendBit-}\checkmark(A, B, K); _ \leftarrow \text{Share-}\checkmark(A, k); _ \leftarrow \text{Share-}\checkmark(A, l); \text{ret } \checkmark$
 - $\text{Share-}\checkmark(B, K) := _ \leftarrow \text{Share-}\checkmark(A, K); _ \leftarrow \text{RcvdBit-}\checkmark(A, B, K);$
 $_ \leftarrow \text{Share-}\checkmark(B, k); _ \leftarrow \text{Share-}\checkmark(B, l); \text{ret } \checkmark$

We also observe that in the presence of the protocol Init we can define the protocol $\text{Init-}\checkmark$ equivalently as follows:

- $\text{InShare-}\$-\checkmark(A, A, i) := _ \leftarrow \text{In-}\checkmark(A, i); \text{ret } \checkmark$ for $i < N$
- $\text{InShare-}\$-\checkmark(A, B, i) := _ \leftarrow \text{In-}\checkmark(B, i); \text{ret } \checkmark$ for $i < M$
- $\text{InShare-}\$-\checkmark(B, A, i) := _ \leftarrow \text{In-}\checkmark(A, i); _ \leftarrow \text{InShare-}\$-\checkmark(A, A, i); \text{ret } \checkmark$ for $i < N$
- $\text{InShare-}\$-\checkmark(B, B, i) := _ \leftarrow \text{In-}\checkmark(B, i); _ \leftarrow \text{InShare-}\$-\checkmark(A, B, i); \text{ret } \checkmark$ for $i < M$
- $\text{InShare-}\checkmark(A, A, i) := \text{read InShare-}\$-\checkmark(A, A, i)$ for $i < N$
- $\text{InShare-}\checkmark(A, B, i) := \text{read InShare-}\$-\checkmark(A, B, i)$ for $i < M$
- $\text{InShare-}\checkmark(B, A, i) := \text{read InShare-}\$-\checkmark(B, A, i)$ for $i < N$
- $\text{InShare-}\checkmark(B, B, i) := \text{read InShare-}\$-\checkmark(B, B, i)$ for $i < M$

Furthermore, in the presence of the channels $\text{In-}\checkmark(A, -)$, $\text{In-}\checkmark(B, -)$ we can express the protocol $\text{Init-}\checkmark$ equivalently as follows:

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

In the presence of the channels $\text{InShare-}\checkmark(A, A, -)$, $\text{InShare-}\checkmark(A, B, -)$, $\text{InShare-}\checkmark(B, A, -)$, $\text{InShare-}\checkmark(B, B, -)$ as well as the protocol $\text{Wires-}\checkmark(C, K)$ we can define the protocol $\text{Shares-}\checkmark(C, K)$ equivalently as the channels

- $\text{Share-}\checkmark(A, k) := \text{read Wire-}\checkmark(k)$ for $k < K$
- $\text{Share-}\checkmark(B, k) := \text{read Wire-}\checkmark(k)$ for $k < K$

together with the following protocol $\text{SendRcvdBits-}\checkmark(C, K)$:

- $\text{SendRcvdBits-}\checkmark(\epsilon, 0)$ is the protocol 0
- $\text{SendRcvdBits-}\checkmark(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{SendRcvdBits-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
- $\text{SendRcvdBits-}\checkmark(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{SendRcvdBits-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
- $\text{SendRcvdBits-}\checkmark(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{SendRcvdBits-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
- $\text{SendRcvdBits-}\checkmark(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{SendRcvdBits-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := \text{read SendBit-}\checkmark(A, B, K)$
 - $\text{RcvdBit-}\checkmark(A, B, K) := \text{read RcvdBit-}\checkmark(A, B, K)$
- $\text{SendRcvdBits-}\checkmark(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{SendRcvdBits-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(A, B, K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark$
 - $\text{RcvdBit-}\checkmark(A, B, K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark$

We now revisit the case of an *and* gate in the protocol $\text{Adv}(C, K)$. We can write the channels

- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(A, k)$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(A, l)$

equivalently as follows:

- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(k)$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(l)$

So the inductive part of the real protocol now has $\text{Adv}(C, K)$ looking like so:

- $\text{Adv}(\epsilon, 0)$ is the protocol 0
- $\text{Adv}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)_{\text{adv}}^A$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}}$

- $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}}$
- $\text{Adv}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K)_{\text{adv}}^A := \text{read SendBit}(A, B, K)$
 - $\text{Share}(A, K)_{\text{adv}}^A := \text{read Share}(A, K)$
 - $\text{OTMsg}_0(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A$
 - $\text{OTMsg}_1(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus x_A$
 - $\text{OTMsg}_2(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus y_A$
 - $\text{OTMsg}_3(A, B, K)_{\text{adv}}^{\text{ot}} := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } b_A \oplus x_A \oplus y_A$
 - $\text{OTChcRcvd}_0(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(k)$
 - $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(l)$

Finally, in the presence of the channels $\text{Share-}\checkmark(A, -)$ we can characterize the protocol $\text{Shares}(C, K)$ equivalently as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read InShare}(A, A, i)$
 - $\text{Share}(B, K) := _ \leftarrow \text{Wire-}\checkmark(K); \text{read InShare}(B, A, i)$
- $\text{Shares}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read InShare}(A, B, i)$
 - $\text{Share}(B, K) := _ \leftarrow \text{Wire-}\checkmark(K); \text{read InShare}(B, B, i)$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read Share}(A, k)$
 - $\text{Share}(B, K) := _ \leftarrow \text{Wire-}\checkmark(K); x_B \leftarrow \text{Share}(B, k); \text{ret } \neg x_B$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } x_A \oplus y_A$
 - $\text{Share}(B, K) := _ \leftarrow \text{Wire-}\checkmark(K); x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } x_B \oplus y_B$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{samp flip}$

- $\text{RcvdBit}(B, A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l);$
 $x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
- $\text{Share}(A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } (x_A * y_A) \oplus b_A$
- $\text{Share}(B, K) := _ \leftarrow \text{Wire-}\checkmark(K); b_B \leftarrow \text{RcvdBit}(B, A, K);$
 $x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_B \oplus (x_B * y_B)$

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 $\text{Adv}(C, K)$ and $\text{Shares}(C, K)$ we made at the beginning of this section.

9.4.6 Sum Of Shares

We continue to simplify the real protocol after amending $\text{Adv}(C, K)$ and $\text{Shares}(C, K)$ as indicated in Section 9.4.5.

We start by adding new internal channels

- $\text{Share-}\Sigma(k) := x_A \leftarrow \text{Share-}\Sigma(A, k); x_B \leftarrow \text{Share}(B, k); \text{ret } x_A \oplus x_B \text{ for } k < K$

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

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

equivalently as follows:

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

We can express the channels

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

equivalently as follows:

- $\begin{cases} \text{Out}(A, k) := x_A \leftarrow \text{Share}(A, k); x_B \leftarrow \text{Share}(B, k); \text{ret } x_A \oplus x_B \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{Out}(A, k) := \text{read Out}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{Out}(B, k) := x_A \leftarrow \text{Share}(A, k); x_B \leftarrow \text{Share}(B, k); \text{ret } x_A \oplus x_B \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{Out}(B, k) := \text{read Out}(B, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

In presence of the channels $\text{Share-}\Sigma(-)$ we can express the above equivalently as follows:

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

At this point, the internal channels $\text{OutShare}(A, A, -)$, $\text{OutShare}(A, B, -)$, $\text{OutShare}(B, A, -)$, $\text{OutShare}(B, B, -)$ are unused and can be eliminated. The simplified version Fin of the final part of the real protocol is therefore as follows:

- $\begin{cases} \text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read SendOutShare}(B, A, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read Share}(B, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read RcvdOutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare}(A, A, k)_{\text{adv}}^A := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(A, A, k)_{\text{adv}}^A := \text{read OutShare}(A, A, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare}(A, B, k)_{\text{adv}}^A := \text{read Share}(B, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(A, B, k)_{\text{adv}}^A := \text{read OutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{Out}(A, k) := \text{read Share-}\Sigma(k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{Out}(A, k) := \text{read Out}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{Out}(B, k) := \text{read Share-}\Sigma(k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{Out}(B, k) := \text{read Out}(B, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\text{Out}(A, k)_{\text{adv}}^A := \text{read Out}(A, k) \text{ for } k < K$

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

- $\text{Share-}\Sigma(k) := x_A \leftarrow \text{Share}(A, k); x_B \leftarrow \text{Share}(B, k); \text{ret } x_A \oplus x_B \text{ for } k < K$

together with the protocol $\text{Shares}(C, K)$ equivalently as the channels

- $\text{Share}(\mathbf{B}, k) := x_A \leftarrow \text{Share}(\mathbf{A}, k); x \leftarrow \text{Share-}\Sigma(k); \text{ret } x_A \oplus x \text{ for } k < K$

together with the following new form of the protocol $\text{Shares}(C, K)$:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(\mathbf{A}, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(\mathbf{A}, \mathbf{B}, K) := \text{read SendBit}(\mathbf{A}, \mathbf{B}, K)$
 - $\text{RcvdBit}(\mathbf{B}, \mathbf{A}, K) := \text{read RcvdBit}(\mathbf{B}, \mathbf{A}, K)$
 - $\text{Share}(\mathbf{A}, K) := \text{read InShare}(\mathbf{A}, \mathbf{A}, i)$
 - $\text{Share-}\Sigma(K) := \text{In}(\mathbf{A}, i)$
- $\text{Shares}(C; \text{input-gate}(\mathbf{B}, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(\mathbf{A}, \mathbf{B}, K) := \text{read SendBit}(\mathbf{A}, \mathbf{B}, K)$
 - $\text{RcvdBit}(\mathbf{B}, \mathbf{A}, K) := \text{read RcvdBit}(\mathbf{B}, \mathbf{A}, K)$
 - $\text{Share}(\mathbf{A}, K) := \text{read InShare}(\mathbf{A}, \mathbf{B}, i)$
 - $\text{Share-}\Sigma(K) := \text{In}(\mathbf{B}, i)$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(\mathbf{A}, \mathbf{B}, K) := \text{read SendBit}(\mathbf{A}, \mathbf{B}, K)$
 - $\text{RcvdBit}(\mathbf{B}, \mathbf{A}, K) := \text{read RcvdBit}(\mathbf{B}, \mathbf{A}, K)$
 - $\text{Share}(\mathbf{A}, K) := \text{read Share}(\mathbf{A}, k)$
 - $\text{Share-}\Sigma(K) := x \leftarrow \text{Share-}\Sigma(k); \text{ret } \neg x$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(\mathbf{A}, \mathbf{B}, K) := \text{read SendBit}(\mathbf{A}, \mathbf{B}, K)$
 - $\text{RcvdBit}(\mathbf{B}, \mathbf{A}, K) := \text{read RcvdBit}(\mathbf{B}, \mathbf{A}, K)$
 - $\text{Share}(\mathbf{A}, K) := x_A \leftarrow \text{Share}(\mathbf{A}, k); y_A \leftarrow \text{Share}(\mathbf{A}, l); \text{ret } x_A \oplus y_A$
 - $\text{Share-}\Sigma(K) := x \leftarrow \text{Share-}\Sigma(k); y \leftarrow \text{Share-}\Sigma(l); \text{ret } x \oplus y$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(\mathbf{A}, \mathbf{B}, K) := x_A \leftarrow \text{Share}(\mathbf{A}, k); y_A \leftarrow \text{Share}(\mathbf{A}, l); \text{samp flip}$
 - $\text{RcvdBit}(\mathbf{B}, \mathbf{A}, K) := b_A \leftarrow \text{SendBit}(\mathbf{A}, \mathbf{B}, K); x_A \leftarrow \text{Share}(\mathbf{A}, k); y_A \leftarrow \text{Share}(\mathbf{A}, l);$
 $x_B \leftarrow \text{Share}(\mathbf{B}, k); y_B \leftarrow \text{Share}(\mathbf{B}, l); \text{ret } b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
 - $\text{Share}(\mathbf{A}, K) := b_A \leftarrow \text{SendBit}(\mathbf{A}, \mathbf{B}, K); x_A \leftarrow \text{Share}(\mathbf{A}, k); y_A \leftarrow \text{Share}(\mathbf{A}, l); \text{ret } (x_A * y_A) \oplus b_A$
 - $\text{Share-}\Sigma(K) := x \leftarrow \text{Share-}\Sigma(k); y \leftarrow \text{Share-}\Sigma(l); \text{ret } x * y$

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
 - $\text{Share}(\mathbf{B}, K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); \text{read InShare}(\mathbf{B}, \mathbf{A}, i)$

into the closed form of the channel

- $\text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{Share}(\mathbf{B}, K); \text{ret } x_A \oplus x_B$

which yields the following:

- $\text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{InShare}(\mathbf{B}, \mathbf{A}, i); \text{ret } x_A \oplus x_B$

By canceling out two applications of $x_A \oplus -$ we can reformulate the channel $\text{Share}(\mathbf{B}, K)$ as follows:

$$- \text{Share}(\mathbf{B}, K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{InShare}(\mathbf{B}, \mathbf{A}, i); \text{ret } x_A \oplus (x_A \oplus x_B)$$

The above can be expressed more concisely as

$$- \text{Share}(\mathbf{B}, K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x \leftarrow \text{Share-}\Sigma(K); \text{ret } x_A \oplus x$$

and this is the desired closed form of the channel $\text{Share}(\mathbf{B}, K)$.

We can now turn our attention to the sum of shares. In the presence of the channel

$$- \text{Share}(\mathbf{A}, K) := \text{read InShare}(\mathbf{A}, \mathbf{A}, i)$$

and the channel

$$- \text{InShare}(\mathbf{A}, \mathbf{A}, i) := \text{read InShare-}\$(\mathbf{A}, \mathbf{A}, i)$$

from the protocol Init we can write the channel

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{InShare}(\mathbf{B}, \mathbf{A}, i); \text{ret } x_A \oplus x_B$$

equivalently as follows:

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{InShare-}\$(\mathbf{A}, \mathbf{A}, i); x_B \leftarrow \text{InShare}(\mathbf{B}, \mathbf{A}, i); \text{ret } x_A \oplus x_B$$

In the presence of the channels

$$- \text{InShare}(\mathbf{B}, \mathbf{A}, i) := \text{read InShare-}\$(\mathbf{B}, \mathbf{A}, i)$$

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

from the protocol Init we can further write the channel $\text{Share-}\Sigma(K)$ as follows:

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{InShare-}\$(\mathbf{A}, \mathbf{A}, i); x \leftarrow \text{In}(\mathbf{A}, i); \text{ret } x_A \oplus (x_A \oplus x)$$

We can cancel out the two applications of $x_A \oplus -$:

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{InShare-}\$(\mathbf{A}, \mathbf{A}, i); \text{read In}(\mathbf{A}, i)$$

This is almost what we want except for the extra dependency on the channel $\text{InShare-}\$(\mathbf{A}, \mathbf{A}, i)$. But it is easy to see that this dependency can be dropped because the channel

$$- \text{InShare-}\$(\mathbf{A}, \mathbf{A}, i) := x \leftarrow \text{In}(\mathbf{A}, i); \text{samp flip}$$

only reads from the channel $\text{In}(\mathbf{A}, i)$, which $\text{Share-}\Sigma(K)$ reads from as well:

$$- \text{Share-}\Sigma(K) := \text{read In}(\mathbf{A}, i)$$

But this is precisely the desired inductive form of the channel $\text{Share-}\Sigma(K)$.

- In the case of an *input* gate for Bob's input, we start by substituting the inductive form of the channel

$$- \text{Share}(\mathbf{B}, K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); \text{read InShare}(\mathbf{B}, \mathbf{B}, i)$$

into the closed form of the channel

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{Share}(\mathbf{B}, K); \text{ret } x_A \oplus x_B$$

which yields the following:

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{InShare}(\mathbf{B}, \mathbf{B}, i); \text{ret } x_A \oplus x_B$$

By canceling out two applications of $x_A \oplus -$ we can reformulate the channel $\text{Share}(\mathbf{B}, K)$ as follows:

$$- \text{Share}(\mathbf{B}, K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{InShare}(\mathbf{B}, \mathbf{B}, i); \text{ret } x_A \oplus (x_A \oplus x_B)$$

The above can be expressed more concisely as

$$- \text{Share}(\mathbf{B}, K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x \leftarrow \text{Share-}\Sigma(K); \text{ret } x_A \oplus x$$

and this is the desired closed form of the channel $\text{Share}(\mathbf{B}, K)$.

We can now turn our attention to the sum of shares. In the presence of the channel

$$- \text{Share}(\mathbf{A}, K) := \text{read InShare}(\mathbf{A}, \mathbf{B}, i)$$

and the channel

$$- \text{InShare}(\mathbf{A}, \mathbf{B}, i) := \text{read InShare-}\$(\mathbf{A}, \mathbf{B}, i)$$

from the protocol Init we can write the channel

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{InShare}(\mathbf{B}, \mathbf{B}, i); \text{ret } x_A \oplus x_B$$

equivalently as follows:

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{InShare-}\$(\mathbf{A}, \mathbf{B}, i); x_B \leftarrow \text{InShare}(\mathbf{B}, \mathbf{B}, i); \text{ret } x_A \oplus x_B$$

In the presence of the channels

$$- \text{InShare}(\mathbf{B}, \mathbf{B}, i) := \text{read InShare-}\$(\mathbf{B}, \mathbf{B}, i)$$

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

from the protocol Init we can further write the channel $\text{Share-}\Sigma(K)$ as follows:

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{InShare-}\$(\mathbf{A}, \mathbf{B}, i); x \leftarrow \text{In}(\mathbf{B}, i); \text{ret } x_A \oplus (x_A \oplus x)$$

We can cancel out the two applications of $x_A \oplus -$:

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{InShare-}\$(\mathbf{A}, \mathbf{B}, i); \text{read In}(\mathbf{B}, i)$$

This is almost what we want except for the extra dependency on the channel $\text{InShare-}\$(\mathbf{A}, \mathbf{B}, i)$. But it is easy to see that this dependency can be dropped because the channel

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

only reads from the channel $\text{In}(\mathbf{B}, i)$, which $\text{Share-}\Sigma(K)$ reads from as well:

$$- \text{Share-}\Sigma(K) := \text{read In}(\mathbf{B}, i)$$

But this is precisely the desired inductive form of the channel $\text{Share-}\Sigma(K)$.

- In the case of a *not* gate, we start by substituting the inductive form of the channel

$$- \text{Share}(\mathbf{B}, K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{Share}(\mathbf{B}, k); \text{ret } \neg x_B$$

into the closed form of the channel

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{Share}(\mathbf{B}, K); \text{ret } x_A \oplus x_B$$

which yields the following:

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{Share}(\mathbf{B}, k); \text{ret } x_A \oplus (\neg x_B)$$

By canceling out two applications of $x_A \oplus -$ we can reformulate the channel $\text{Share}(\mathbf{B}, K)$ as follows:

$$- \text{Share}(\mathbf{B}, K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{Share}(\mathbf{B}, k); \text{ret } x_\Sigma \oplus (x_\Sigma \oplus (\neg x_B))$$

The above can be expressed more concisely as

$$- \text{Share}(\mathbf{B}, K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x \leftarrow \text{Share-}\Sigma(K); \text{ret } x_A \oplus x$$

and this is the desired closed form of the channel $\text{Share}(\mathbf{B}, K)$.

We can now turn our attention to the sum of shares. Substituting the channel

$$- \text{Share}(\mathbf{A}, K) := \text{read } \text{Share}(\mathbf{A}, k)$$

into the channel

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{Share}(\mathbf{B}, k); \text{ret } x_A \oplus (\neg x_B)$$

yields the following:

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(\mathbf{A}, k); x_B \leftarrow \text{Share}(\mathbf{B}, k); \text{ret } x_A \oplus (\neg x_B)$$

The negation can be brought to the top level:

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(\mathbf{A}, k); x_B \leftarrow \text{Share}(\mathbf{B}, k); \text{ret } \neg(x_A \oplus x_B)$$

But this is precisely what we get if we substitute the closed form of the channel

$$- \text{Share-}\Sigma(k) := x_A \leftarrow \text{Share}(\mathbf{A}, k); x_B \leftarrow \text{Share}(\mathbf{B}, k); \text{ret } x_A \oplus x_B$$

into the desired inductive form of the channel

$$- \text{Share-}\Sigma(K) := x \leftarrow \text{Share-}\Sigma(k); \text{ret } \neg x$$

so we are done.

- In the case of an *xor* gate, we start by substituting the inductive form of the channel

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

into the closed form of the channel

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{Share}(\mathbf{B}, K); \text{ret } x_A \oplus x_B$$

which yields the following:

$$- \text{Share-}\Sigma(\mathbf{B}, K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{Share}(\mathbf{B}, k); y_B \leftarrow \text{Share}(\mathbf{B}, l); \text{ret } x_A \oplus (x_B \oplus y_B)$$

By canceling out two applications of $x_A \oplus -$ we can reformulate the channel $\text{Share}(\mathbf{B}, K)$ as follows:

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

The above can be expressed more concisely as

$$- \text{Share}(\mathbf{B}, K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x \leftarrow \text{Share-}\Sigma(K); \text{ret } x_A \oplus x$$

and this is the desired closed form of the channel $\text{Share}(\mathbf{B}, K)$.

We can now turn our attention to the sum of shares. Substituting the channel

$$- \text{Share}(\mathbf{A}, K) := x_A \leftarrow \text{Share}(\mathbf{A}, k); y_A \leftarrow \text{Share}(\mathbf{A}, l); \text{ret } x_A \oplus y_A$$

into the channel

$$- \text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(\mathbf{A}, K); x_B \leftarrow \text{Share}(\mathbf{B}, k); y_B \leftarrow \text{Share}(\mathbf{B}, l); \text{ret } x_A \oplus (x_B \oplus y_B)$$

yields the following:

$$\begin{aligned} - \text{Share-}\Sigma(K) &:= x_A \leftarrow \text{Share}(\mathbf{A}, k); y_A \leftarrow \text{Share}(\mathbf{A}, l); \\ &x_B \leftarrow \text{Share}(\mathbf{B}, k); y_B \leftarrow \text{Share}(\mathbf{B}, l); \text{ret } (x_A \oplus y_A) \oplus (x_B \oplus y_B) \end{aligned}$$

After a slight rearrangement we get the following:

- $\text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(A, k); x_B \leftarrow \text{Share}(B, k);$
 $y_A \leftarrow \text{Share}(A, l); y_B \leftarrow \text{Share}(B, l); \text{ret } (x_A \oplus x_B) \oplus (y_A \oplus y_B)$

But this is precisely what we get if we substitute the channels

- $\text{Share-}\Sigma(k) := x_A \leftarrow \text{Share}(A, k); x_B \leftarrow \text{Share}(B, k); \text{ret } x_A \oplus x_B$
- $\text{Share-}\Sigma(l) := y_A \leftarrow \text{Share}(A, l); y_B \leftarrow \text{Share}(B, l); \text{ret } y_A \oplus y_B$

into the desired inductive form of the channel

- $\text{Share-}\Sigma(K) := x \leftarrow \text{Share-}\Sigma(k); y \leftarrow \text{Share-}\Sigma(l); \text{ret } x \oplus y$

so we are done.

- In the case of an *and* gate, we start by substituting the inductive form of the channel

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

into the closed form of the channel

- $\text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(A, K); x_B \leftarrow \text{Share}(B, K); \text{ret } x_A \oplus x_B$

which yields the following:

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

By canceling out two applications of $x_A \oplus$ – we can reformulate the channel $\text{Share}(B, K)$ as follows:

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

The above can be expressed more concisely as

- $\text{Share}(B, K) := x_A \leftarrow \text{Share-}\Sigma(A, K); x \leftarrow \text{Share-}\Sigma(K); \text{ret } x_A \oplus x$

and this is the desired closed form of the channel $\text{Share}(B, K)$.

We can now turn our attention to the sum of shares. Substituting the channels

- $\text{Share}(A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } (x_A * y_A) \oplus b_A$
- $\text{RcvdBit}(B, A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l);$
 $x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$

into the channel

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

yields the following:

- $\text{Share-}\Sigma(K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); x_B \leftarrow \text{Share}(B, k);$
 $y_B \leftarrow \text{Share}(B, l); \text{ret } ((x_A * y_A) \oplus b_A) \oplus ((b_A \oplus (x_A * y_B) \oplus (x_B * y_A)) \oplus (x_B * y_B))$

After a slight rearrangement we get the following:

- $\text{Share-}\Sigma(K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); x_B \leftarrow \text{Share}(B, k);$
 $y_B \leftarrow \text{Share}(B, l); \text{ret } (x_A * y_A) \oplus (b_A \oplus (b_A \oplus (x_A * y_B) \oplus (x_B * y_A))) \oplus (x_B * y_B)$

Canceling out the two applications of $b_A \oplus$ – yields the following:

- $\text{Share-}\Sigma(K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l);$
 $x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } (x_A * y_A) \oplus (x_A * y_B) \oplus (x_B * y_A) \oplus (x_B * y_B)$

We can drop the dependency on the unused channel

- $\text{SendBit}(A, B, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{samp flip}$

because it only reads from the channels $\text{Share}(A, k)$ and $\text{Share}(A, l)$, which $\text{Share-}\Sigma(K)$ reads from as well:

- $\text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); x_B \leftarrow \text{Share}(B, k);$
 $y_B \leftarrow \text{Share}(B, l); \text{ret } (x_A * y_A) \oplus (x_A * y_B) \oplus (x_B * y_A) \oplus (x_B * y_B)$

After a slight rearrangement we get the following:

- $\text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(A, k); x_B \leftarrow \text{Share}(B, k); y_A \leftarrow \text{Share}(A, l);$
 $y_B \leftarrow \text{Share}(B, l); \text{ret } (x_A * y_A) \oplus (x_A * y_B) \oplus (x_B * y_A) \oplus (x_B * y_B)$

The above is equivalent to the following:

- $\text{Share-}\Sigma(K) := x_A \leftarrow \text{Share}(A, k); x_B \leftarrow \text{Share}(B, k);$
 $y_A \leftarrow \text{Share}(A, l); y_B \leftarrow \text{Share}(B, l); \text{ret } (x_A \oplus x_B) * (y_A \oplus y_B)$

But this is precisely what we get if we substitute the channels

- $\text{Share-}\Sigma(k) := x_A \leftarrow \text{Share}(A, k); x_B \leftarrow \text{Share}(B, k); \text{ret } x_A \oplus x_B$
- $\text{Share-}\Sigma(l) := y_A \leftarrow \text{Share}(A, l); y_B \leftarrow \text{Share}(B, l); \text{ret } y_A \oplus y_B$

into the desired inductive form of the channel

- $\text{Share-}\Sigma(k) := x \leftarrow \text{Share}(k); y \leftarrow \text{Share}(l); \text{ret } x * y$

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 $\text{Share-}\Sigma(-)$ but we can extract it out into a separate protocol $\text{Wires}(C, K)$ as defined in the ideal functionality. Specifically, we introduce new internal channels

- $\text{Wire}(k) := \text{read Share-}\Sigma(k)$ for $k < K$

and observe that together with the protocol $\text{Shares}(C, K)$ we can express them equivalently as the channels

- $\text{Share-}\Sigma(k) := \text{read Wire}(k)$ for $k < K$

together with the aforementioned protocol $\text{Wires}(C, K)$ and the following new form of the protocol $\text{Shares}(C, K)$:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read InShare}(A, A, i)$
- $\text{Shares}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := \text{read InShare}(A, B, i)$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol

- $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
- $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
- $\text{Share}(A, K) := \text{read Share}(A, k)$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
 - $\text{Share}(A, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } x_A \oplus y_A$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{samp flip}$
 - $\text{RcvdBit}(B, A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l);$
 $x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ret } b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$
 - $\text{Share}(A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } (x_A * y_A) \oplus b_A$

9.4.7 Eliminating Bob's Shares

We begin by eliminating the channels $\text{InShare-}\$(B, A, -)$, $\text{InShare-}\$(B, B, -)$, $\text{InShare}(B, A, -)$, $\text{InShare}(B, B, -)$ from the initial part of the protocol. Substituting the channels

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

into the channels

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

yields the following:

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

Substituting the channels

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

into the channels

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

yields the following:

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

At this point, the internal channels $\text{InShare-}\$(B, A, -)$, $\text{InShare-}\$(B, B, -)$ are unused and can be eliminated.

The top-level internal channels $\text{InShare}(B, A, -)$, $\text{InShare}(B, B, -)$ 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:

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

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

This is followed by the hiding of the channels

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

We now eliminate the channels $\text{RcvdBit}(B, A, -)$, $\text{Share}(B, -)$ from the inductive part of the real protocol. We can extract the computation of the channels $\text{RcvdBit}(B, A, -)$ into a separate protocol $\text{RcvdBits}(C, K)$:

- $\text{RcvdBits}(\epsilon, 0)$ is the protocol 0
- $\text{RcvdBits}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{RcvdBits}(C, K)$ with the protocol
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
- $\text{RcvdBits}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{RcvdBits}(C, K)$ with the protocol
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
- $\text{RcvdBits}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{RcvdBits}(C, K)$ with the protocol
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
- $\text{RcvdBits}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{RcvdBits}(C, K)$ with the protocol
 - $\text{RcvdBit}(B, A, K) := \text{read RcvdBit}(B, A, K)$
- $\text{RcvdBits}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{RcvdBits}(C, K)$ with the protocol
 - $\text{RcvdBit}(B, A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l);$
 $x_B \leftarrow \text{Share}(B, k); y_B \leftarrow \text{Share}(B, l); \text{ ret } b_A \oplus (x_A * y_B) \oplus (x_B * y_A)$

After the extraction, the protocol $\text{Shares}(C, K)$ is left looking as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(A, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read SendBit}(A, B, K)$
 - $\text{Share}(A, K) := \text{read InShare}(A, A, i)$
- $\text{Shares}(C; \text{input-gate}(B, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol

- $\text{SendBit}(A, B, K) := \text{read } \text{SendBit}(A, B, K)$
- $\text{Share}(A, K) := \text{read } \text{InShare}(A, B, i)$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read } \text{SendBit}(A, B, K)$
 - $\text{Share}(A, K) := \text{read } \text{Share}(A, k)$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := \text{read } \text{SendBit}(A, B, K)$
 - $\text{Share}(A, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } x_A \oplus y_A$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(A, B, K) := x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{samp flip}$
 - $\text{Share}(A, K) := b_A \leftarrow \text{SendBit}(A, B, K); x_A \leftarrow \text{Share}(A, k); y_A \leftarrow \text{Share}(A, l); \text{ret } (x_A * y_A) \oplus b_A$

None of the channels defined by $\text{RcvdBits}(C, K)$ are utilized anywhere outside of $\text{RcvdBits}(C, K)$ and as such we may discard this protocol fragment entirely. This in particular eliminates all references to the channels $\text{Share}(B, -)$ from the inductive part of the protocol. To summarize, the inductive part of the real protocol now consists of the protocols $\text{Shares}(C, K)$ and $\text{Adv}(C, K)$, followed by the hiding of the channels

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

We recall that on the top level we also have the protocol $\text{Wires}(C, K)$ and the channels below:

- $\text{Share-}\Sigma(k) := \text{read } \text{Wire}(k)$ for $k < K$
- $\text{Share}(B, k) := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Share-}\Sigma(k); \text{ret } x_A \oplus x$ 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

- $\text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read } \text{Share}(B, k)$
- $\text{OutShare}(A, B, k)_{\text{adv}}^A := \text{read } \text{Share}(B, k)$

read from the channel

- $\text{Share}(B, k) := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Share-}\Sigma(k); \text{ret } x_A \oplus x$

so we may substitute:

- $\text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Share-}\Sigma(k); \text{ret } x_A \oplus x$
- $\text{OutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Share-}\Sigma(k); \text{ret } x_A \oplus x$

We thus get the following for channels $\text{RcvdOutShare}(A, B, -)_{\text{adv}}^A$ and $\text{OutShare}(A, B, -)_{\text{adv}}^A$:

- $\begin{cases} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Share-}\Sigma(k); \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read } \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Share-}\Sigma(k); \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(A, B, k)_{\text{adv}}^A := \text{read } \text{OutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

The top-level internal channels $\text{Share}(\mathbf{B}, -)$ 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:

- $\left\{ \begin{array}{l} \text{SendOutShare}(\mathbf{B}, \mathbf{A}, k)_{\text{adv}}^{\mathbf{A}} := \text{read Share}(\mathbf{A}, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{SendOutShare}(\mathbf{B}, \mathbf{A}, k)_{\text{adv}}^{\mathbf{A}} := \text{read SendOutShare}(\mathbf{B}, \mathbf{A}, k)_{\text{adv}}^{\mathbf{A}} \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(\mathbf{A}, \mathbf{B}, k)_{\text{adv}}^{\mathbf{A}} := x_A \leftarrow \text{Share}(\mathbf{A}, k); x \leftarrow \text{Share-}\Sigma(k); \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(\mathbf{A}, \mathbf{B}, k)_{\text{adv}}^{\mathbf{A}} := \text{read RcvdOutShare}(\mathbf{A}, \mathbf{B}, k)_{\text{adv}}^{\mathbf{A}} \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(\mathbf{A}, \mathbf{A}, k)_{\text{adv}}^{\mathbf{A}} := \text{read Share}(\mathbf{A}, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(\mathbf{A}, \mathbf{A}, k)_{\text{adv}}^{\mathbf{A}} := \text{read OutShare}(\mathbf{A}, \mathbf{A}, k)_{\text{adv}}^{\mathbf{A}} \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(\mathbf{A}, \mathbf{B}, k)_{\text{adv}}^{\mathbf{A}} := x_A \leftarrow \text{Share}(\mathbf{A}, k); x \leftarrow \text{Share-}\Sigma(k); \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(\mathbf{A}, \mathbf{B}, k)_{\text{adv}}^{\mathbf{A}} := \text{read OutShare}(\mathbf{A}, \mathbf{B}, k)_{\text{adv}}^{\mathbf{A}} \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(\mathbf{A}, k) := \text{read Share-}\Sigma(k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(\mathbf{A}, k) := \text{read Out}(\mathbf{A}, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(\mathbf{B}, k) := \text{read Share-}\Sigma(k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(\mathbf{B}, k) := \text{read Out}(\mathbf{B}, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\text{Out}(\mathbf{A}, k)_{\text{adv}}^{\mathbf{A}} := \text{read Out}(\mathbf{A}, k) \text{ for } k < K$

As a final step before the extraction of the simulator, we eliminate any reference to the channels $\text{Share-}\Sigma(-)$ from the final part of the real protocol. If wire k is an output, then we can substitute the channel

- $\text{Share-}\Sigma(k) := \text{read Wire}(k)$

into the channels

- $\text{RcvdOutShare}(\mathbf{A}, \mathbf{B}, k)_{\text{adv}}^{\mathbf{A}} := x_A \leftarrow \text{Share}(\mathbf{A}, k); x \leftarrow \text{Share-}\Sigma(k); \text{ret } x_A \oplus x$
- $\text{OutShare}(\mathbf{A}, \mathbf{B}, k)_{\text{adv}}^{\mathbf{A}} := x_A \leftarrow \text{Share}(\mathbf{A}, k); x \leftarrow \text{Share-}\Sigma(k); \text{ret } x_A \oplus x$

which yields the following:

- $\text{RcvdOutShare}(\mathbf{A}, \mathbf{B}, k)_{\text{adv}}^{\mathbf{A}} := x_A \leftarrow \text{Share}(\mathbf{A}, k); x \leftarrow \text{Wire}(k); \text{ret } x_A \oplus x$
- $\text{OutShare}(\mathbf{A}, \mathbf{B}, k)_{\text{adv}}^{\mathbf{A}} := x_A \leftarrow \text{Share}(\mathbf{A}, k); x \leftarrow \text{Wire}(k); \text{ret } x_A \oplus x$

We thus get the following for channels $\text{RcvdOutShare}(\mathbf{A}, \mathbf{B}, -)_{\text{adv}}^{\mathbf{A}}$ and $\text{OutShare}(\mathbf{A}, \mathbf{B}, -)_{\text{adv}}^{\mathbf{A}}$:

- $\left\{ \begin{array}{l} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Wire}(k); \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read RcvdOutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Wire}(k); \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(A, B, k)_{\text{adv}}^A := \text{read OutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$

If wire k is an output, then the channels

- $\text{Out}(A, k) := \text{read Share-}\Sigma(k)$
- $\text{Out}(B, k) := \text{read Share-}\Sigma(k)$

read from the channel

- $\text{Share-}\Sigma(k) := \text{read Wire}(k)$

so we may substitute:

- $\text{Out}(A, k) := \text{read Wire}(k)$
- $\text{Out}(B, k) := \text{read Wire}(k)$

We thus get the following for channels $\text{Out}(A, -)$ and $\text{Out}(B, -)$:

- $\left\{ \begin{array}{l} \text{Out}(A, k) := \text{read Wire}(k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(A, k) := \text{read Out}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(B, k) := \text{read Wire}(k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(B, k) := \text{read Out}(B, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$

The top-level internal channels $\text{Share-}\Sigma(-)$ 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:

- $\left\{ \begin{array}{l} \text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read SendOutShare}(B, A, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Wire}(k); \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read RcvdOutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(A, A, k)_{\text{adv}}^A := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(A, A, k)_{\text{adv}}^A := \text{read OutShare}(A, A, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$

- $\left\{ \begin{array}{l} \text{OutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Wire}(k); \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(A, B, k)_{\text{adv}}^A := \text{read OutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(A, k) := \text{read Wire}(k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{Out}(A, k) := \text{read Out}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(B, k) := \text{read Wire}(k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{Out}(B, k) := \text{read Out}(B, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\text{Out}(A, k)_{\text{adv}}^A := \text{read Out}(A, k) \text{ for } k < K$

9.4.8 Extracting The Simulator

We are now ready to extract the simulator. The internal protocol $\text{Wires}(C, K)$ together with the output channels

- $\left\{ \begin{array}{l} \text{Out}(A, k) := \text{read Wire}(k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{Out}(A, k) := \text{read Out}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(B, k) := \text{read Wire}(k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{Out}(B, k) := \text{read Out}(B, k) \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$

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:

- $\left\{ \begin{array}{l} \text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read SendOutShare}(B, A, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Wire}(k); \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read RcvdOutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(A, A, k)_{\text{adv}}^A := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(A, A, k)_{\text{adv}}^A := \text{read OutShare}(A, A, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$

- $\begin{cases} \text{OutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Wire}(k); \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(A, B, k)_{\text{adv}}^A := \text{read OutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\text{Out}(A, k)_{\text{adv}}^A := \text{read Out}(A, k) \text{ for } k < K$

In the remainder of the soon-to-be simulator, we must eliminate any references to the channels $\text{In}(A, -)$, $\text{Wire}(-)$, $\text{Out}(A, -)$, $\text{Out}(B, -)$. We begin with the initial part of the real protocol. Recall the leakage

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

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:

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

This is followed by the hiding of the channels

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

We continue with the final part of the soon-to-be simulator. Recall the definition of the output channels

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

in the ideal functionality. If wire k is an output, then the channels

- $\text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Wire}(k); \text{ret } x_A \oplus x$
- $\text{OutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Wire}(k); \text{ret } x_A \oplus x$

can be expressed equivalently as

- $\text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k); \text{ret } x_A \oplus x$
- $\text{OutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k); \text{ret } x_A \oplus x$

since we have the definition below:

- $\text{Out}(A, k) := \text{read Wire}(k)$

We thus get the following for channels $\text{RcvdOutShare}(A, B, -)_{\text{adv}}^A$ and $\text{OutShare}(A, B, -)_{\text{adv}}^A$:

- $\begin{cases} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k); \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read RcvdOutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k); \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(A, B, k)_{\text{adv}}^A := \text{read OutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

The latest version Fin of the final part of the soon-to-be simulator is therefore as follows:

- $\begin{cases} \text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read SendOutShare}(B, A, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k); \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read RcvdOutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare}(A, A, k)_{\text{adv}}^A := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(A, A, k)_{\text{adv}}^A := \text{read OutShare}(A, A, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k); \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(A, B, k)_{\text{adv}}^A := \text{read OutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\text{Out}(A, k)_{\text{adv}}^A := \text{read Out}(A, k) \text{ for } k < K$

We now recall the leakage

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

from the ideal functionality. If wire k is an output, then the channels

- $\text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k); \text{ret } x_A \oplus x$
- $\text{OutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k); \text{ret } x_A \oplus x$

can be expressed equivalently as

- $\text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k)_{\text{adv}}^{\text{id}}; \text{ret } x_A \oplus x$
- $\text{OutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k)_{\text{adv}}^{\text{id}}; \text{ret } x_A \oplus x$

We thus get the following for channels $\text{RcvdOutShare}(A, B, -)_{\text{adv}}^A$ and $\text{OutShare}(A, B, -)_{\text{adv}}^A$:

- $\begin{cases} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k)_{\text{adv}}^{\text{id}}; \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read RcvdOutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k)_{\text{adv}}^{\text{id}}; \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(A, B, k)_{\text{adv}}^A := \text{read OutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$

Finally, the channels

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

can be expressed equivalently as

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

The final version Fin of the final part of the simulator is therefore as follows:

- $\begin{cases} \text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read SendOutShare}(B, A, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k)_{\text{adv}}^{\text{id}}; \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read RcvdOutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare}(A, A, k)_{\text{adv}}^A := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(A, A, k)_{\text{adv}}^A := \text{read OutShare}(A, A, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k)_{\text{adv}}^{\text{id}}; \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(A, B, k)_{\text{adv}}^A := \text{read OutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\text{Out}(A, k)_{\text{adv}}^A := \text{read Out}(A, k)_{\text{adv}}^{\text{id}} \text{ for } k < K$

9.4.9 The Simulator

The simulator consists of three parts. In the initial phase, we have the protocol Init:

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

This is followed by the hiding of the channels

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

In the inductive phase, we have the protocol $\text{Circ}(C, K)$, obtained by merging the two protocols $\text{Shares}(C, K)$ and $\text{Adv}(C, K)$ from earlier:

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

- Share(A, K)_{adv}^A := read Share(A, K)
- OTMsg₀(A, B, K)_{adv}^{ot} := read OTMsg₀(A, B, K)_{adv}^{ot}
- OTMsg₁(A, B, K)_{adv}^{ot} := read OTMsg₁(A, B, K)_{adv}^{ot}
- OTMsg₂(A, B, K)_{adv}^{ot} := read OTMsg₂(A, B, K)_{adv}^{ot}
- OTMsg₃(A, B, K)_{adv}^{ot} := read OTMsg₃(A, B, K)_{adv}^{ot}
- OTChcRcvd₀(A, B, K)_{adv}^{ot} := read OTChcRcvd₀(A, B, K)_{adv}^{ot}
- OTChcRcvd₁(A, B, K)_{adv}^{ot} := read OTChcRcvd₁(A, B, K)_{adv}^{ot}

- Circ(C; not-gate(k), K + 1) is the composition of Circ(C, K) with the protocol
 - SendBit(A, B, K) := read SendBit(A, B, K)
 - SendBit(A, B, K)_{adv}^A := read SendBit(A, B, K)_{adv}^A
 - Share(A, K) := read Share(A, k)
 - Share(A, K)_{adv}^A := read Share(A, K)
 - OTMsg₀(A, B, K)_{adv}^{ot} := read OTMsg₀(A, B, K)_{adv}^{ot}
 - OTMsg₁(A, B, K)_{adv}^{ot} := read OTMsg₁(A, B, K)_{adv}^{ot}
 - OTMsg₂(A, B, K)_{adv}^{ot} := read OTMsg₂(A, B, K)_{adv}^{ot}
 - OTMsg₃(A, B, K)_{adv}^{ot} := read OTMsg₃(A, B, K)_{adv}^{ot}
 - OTChcRcvd₀(A, B, K)_{adv}^{ot} := read OTChcRcvd₀(A, B, K)_{adv}^{ot}
 - OTChcRcvd₁(A, B, K)_{adv}^{ot} := read OTChcRcvd₁(A, B, K)_{adv}^{ot}

- Circ(C; xor-gate(k, l), K + 1) is the composition of Circ(C, K) with the protocol
 - SendBit(A, B, K) := read SendBit(A, B, K)
 - SendBit(A, B, K)_{adv}^A := read SendBit(A, B, K)_{adv}^A
 - Share(A, K) := x_A ← Share(A, k); y_A ← Share(A, l); ret x_A ⊕ y_A
 - Share(A, K)_{adv}^A := read Share(A, K)
 - OTMsg₀(A, B, K)_{adv}^{ot} := read OTMsg₀(A, B, K)_{adv}^{ot}
 - OTMsg₁(A, B, K)_{adv}^{ot} := read OTMsg₁(A, B, K)_{adv}^{ot}
 - OTMsg₂(A, B, K)_{adv}^{ot} := read OTMsg₂(A, B, K)_{adv}^{ot}
 - OTMsg₃(A, B, K)_{adv}^{ot} := read OTMsg₃(A, B, K)_{adv}^{ot}
 - OTChcRcvd₀(A, B, K)_{adv}^{ot} := read OTChcRcvd₀(A, B, K)_{adv}^{ot}
 - OTChcRcvd₁(A, B, K)_{adv}^{ot} := read OTChcRcvd₁(A, B, K)_{adv}^{ot}

- Circ(C; and-gate(k, l), K + 1) is the composition of Circ(C, K) with the protocol
 - SendBit(A, B, K) := x_A ← Share(A, k); y_A ← Share(A, l); samp flip
 - SendBit(A, B, K)_{adv}^A := read SendBit(A, B, K)
 - Share(A, K) := b_A ← SendBit(A, B, K); x_A ← Share(A, k); y_A ← Share(A, l); ret (x_A * y_A) ⊕ b_A
 - Share(A, K)_{adv}^A := read Share(A, K)
 - OTMsg₀(A, B, K)_{adv}^{ot} := b_A ← SendBit(A, B, K); x_A ← Share(A, k); y_A ← Share(A, l); ret b_A
 - OTMsg₁(A, B, K)_{adv}^{ot} := b_A ← SendBit(A, B, K); x_A ← Share(A, k); y_A ← Share(A, l); ret b_A ⊕ x_A
 - OTMsg₂(A, B, K)_{adv}^{ot} := b_A ← SendBit(A, B, K); x_A ← Share(A, k); y_A ← Share(A, l); ret b_A ⊕ y_A
 - OTMsg₃(A, B, K)_{adv}^{ot} := b_A ← SendBit(A, B, K); x_A ← Share(A, k); y_A ← Share(A, l); ret b_A ⊕ x_A ⊕ y_A
 - OTChcRcvd₀(A, B, K)_{adv}^{ot} := x_A ← Share(A, k); ret ✓

– $\text{OTChcRcvd}_1(A, B, K)_{\text{adv}}^{\text{ot}} := x_A \leftarrow \text{Share}(A, l); \text{ret } \checkmark$

This is followed by the hiding of the channels

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

In the final phase, we have the protocol Fin:

- $\left\{ \begin{array}{l} \text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{SendOutShare}(B, A, k)_{\text{adv}}^A := \text{read SendOutShare}(B, A, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k)_{\text{adv}}^{\text{id}}; \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(A, B, k)_{\text{adv}}^A := \text{read RcvdOutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(A, A, k)_{\text{adv}}^A := \text{read Share}(A, k) \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(A, A, k)_{\text{adv}}^A := \text{read OutShare}(A, A, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(A, B, k)_{\text{adv}}^A := x_A \leftarrow \text{Share}(A, k); x \leftarrow \text{Out}(A, k)_{\text{adv}}^{\text{id}}; \text{ret } x_A \oplus x \\ \text{for } k < K \text{ if wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(A, B, k)_{\text{adv}}^A := \text{read OutShare}(A, B, k)_{\text{adv}}^A \\ \text{for } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\text{Out}(A, k)_{\text{adv}}^A := \text{read Out}(A, k)_{\text{adv}}^{\text{id}}$ for $k < K$

The composition of the three parts is followed by the hiding of the channels

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

Composing the ideal protocol with the simulator, and substituting away the ideal leakage

- $\text{In}(A, i)_{\text{adv}}^{\text{id}}$ for $i < N$,
- $\text{InRcvd}(B, i)_{\text{adv}}^{\text{id}}$ for $i < M$,
- $\text{Out}(A, k)_{\text{adv}}^{\text{id}}$ for $k < K$,
- $\text{Out}(B, k)_{\text{adv}}^{\text{id}}$ 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 $0, \dots, N + 1$ 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 $n := 0, \dots, N + 1$ 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 $\text{flip} : 1 \rightarrow \text{Bool}$; a Boolean sum function $\oplus : \text{Bool} \times \text{Bool} \rightarrow \text{Bool}$, where we write $x \oplus y$ in place of $\oplus(x, y)$; a Boolean multiplication function $*$: $\text{Bool} \times \text{Bool} \rightarrow \text{Bool}$, where we write $x * y$ in place of $*$ (x, y); and a Boolean negation function $\neg : \text{Bool} \rightarrow \text{Bool}$, where we write $\neg x$ in place of $\neg x$.

We represent Boolean circuits using the syntax below, where we assume that party $p := 0, \dots, N + 1$ has $I_p \geq 0$ inputs labeled $\{0, \dots, I_p - 1\}$. 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.

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

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

$$\begin{array}{c}
\frac{}{\epsilon \text{ circuit}(0)} \quad \frac{C \text{ circuit}(n) \quad 0 \leq p \leq N + 1 \quad i < I_p}{C; \text{input-gate}(p, i) \text{ circuit}(n + 1)} \quad \frac{C \text{ circuit}(n) \quad k < n}{C; \text{not-gate}(k) \text{ circuit}(n + 1)} \\
\frac{C \text{ circuit}(n) \quad k < n \quad 0 \leq l < n}{C; \text{xor-gate}(k, l) \text{ circuit}(n + 1)} \quad \frac{C \text{ circuit}(n) \quad k < n \quad 0 \leq l < n}{C; \text{and-gate}(k, l) \text{ circuit}(n + 1)}
\end{array}$$

We now fix an ambient Boolean circuit C with K wires $\{0, \dots, K - 1\}$, 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:

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

Furthermore, Boolean multiplication distributes over Boolean sum:

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

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

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

Negating a Boolean equals summing it up with **true**:

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

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

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

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):

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

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 :

- $\begin{cases} \text{In}(n, i)_{\text{adv}}^{\text{id}} := \text{read In}(n, i) \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{In}(n, i)_{\text{adv}}^{\text{id}} := \text{read In}(n, i)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{InRcvd}(n, i)_{\text{adv}}^{\text{id}} := x \leftarrow \text{In}(n, i); \text{ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \text{InRcvd}(n, i)_{\text{adv}}^{\text{id}} := \text{read InRcvd}(n, i)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$

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

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

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:

- $\begin{cases} \text{Out}(n, k) := \text{read Wire}(k) \\ \text{for } n \leq n + 1 \text{ and } k < K \text{ if wire } k \text{ an output} \\ \text{Out}(n, k) := \text{read Out}(n, k) \\ \text{for } n \leq n + 1 \text{ and } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{Out}(n, k)_{\text{adv}}^{\text{id}} := \text{read Out}(n, k) \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{Out}(n, k)_{\text{adv}}^{\text{id}} := \text{read Out}(n, k)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

Finally, the channels

- $\text{Wire}(k)$ for $k < K$

coming from the inductive protocol $\text{Wires}(C, K)$ 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 $m, n \leq N + 1$. 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:

- $$\begin{cases} \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{In}(n, i) \\ \text{for } i < I_n \text{ if } n \text{ semi-honest} \\ \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } i < I_n \text{ if } n \text{ honest} \end{cases}$$
- $$\begin{cases} \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := x \leftarrow \text{In}(n, i); \text{ ret } \checkmark \\ \text{for } i < I_n \text{ if } n \text{ honest} \\ \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$$

We next randomly generate shares for every party except party $N + 1$:

- $\text{InShare-}\$(m, n, i) := x \leftarrow \text{In}(n, i); \text{ samp flip for } m \leq N \text{ and } i < I_n$
- $$\begin{cases} \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{InShare-}\$(m, n, i) \\ \text{for } m \leq N \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$$

To determine the share of party $N + 1$, we inductively compute the sum of all the shares we generated above:

- $$\begin{cases} \text{InShare-}\$-\Sigma(0, n, i) := \text{read } \text{InShare-}\$(0, n, i) \\ \text{for } i < I_n \\ \text{InShare-}\$-\Sigma(m + 1, n, i) := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(m, n, i); x_{m+1} \leftarrow \text{InShare-}\$(m + 1, n, i); \text{ ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \text{ and } i < I_n \end{cases}$$
- $$\begin{cases} \text{InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{InShare-}\$-\Sigma(m, n, i) \\ \text{for } m \leq N \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$$

We compute the share of party $N + 1$ by summing up the input i with the shares of all the other parties:

- $\text{InShare-}\$(N + 1, n, i) := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i); \text{ ret } x_\Sigma \oplus x \text{ for } i < I_n$

- $$\begin{cases} \text{InShare-}\$(N+1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(N+1, n, i) \\ \text{for } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(N+1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(N+1, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } i < I_n \text{ if } n \text{ honest} \end{cases}$$

This concludes the computation of shares for inputs belonging to party n . We next send each computed share to the respective party:

- $$\begin{cases} \text{SendInShare}(m, n, i) := \text{read InShare-}\$(m, n, i) \text{ for } m \leq N+1 \text{ and } i < I_n \\ \text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(m, n, i) \\ \text{for } m \leq N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$$

If party n is semi-honest, each input share received for an input i belonging to party m is forwarded to the adversary:

- $$\begin{cases} \text{RcvdInShare}(n, m, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(n, m, i) \\ \text{for } m \leq N+1 \text{ and } i < I_m \text{ if } n \text{ semi-honest} \\ \text{RcvdInShare}(n, m, i)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdInShare}(n, m, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N+1 \text{ and } i < I_m \text{ if } n \text{ honest} \end{cases}$$

Finally, the incoming input shares are recorded:

- $$\begin{cases} \text{InShare}(n, m, i) := \text{read SendInShare}(n, m, i) \text{ for } m \leq N+1 \text{ and } i < I_m \\ \text{InShare}(n, m, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare}(n, m, i) \\ \text{for } m \leq N+1 \text{ and } i < I_m \text{ if } n \text{ semi-honest} \\ \text{InShare}(n, m, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare}(n, m, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N+1 \text{ and } i < I_m \text{ if } n \text{ honest} \end{cases}$$

The channels

- $\text{InShare-}\$(m, n, i)$ for $m \leq N+1$ and $i < I_n$,
- $\text{InShare-}\$-\Sigma(m, n, i)$ for $m \leq N$ 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 $0, \dots, N$ 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 $(x_0 \oplus \dots \oplus x_{N+1}) * (y_0 \oplus \dots \oplus y_{N+1})$, where x_n, y_n are the respective shares of party n on the incoming two wires. We have

$$(x_0 \oplus \dots \oplus x_{N+1}) * (y_0 \oplus \dots \oplus y_{N+1}) = \bigoplus_i \bigoplus_j x_i * y_j$$

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:

- $\text{SendBit}(n, m, K)$ for storing the masking Boolean is relevant for an *and* gate when $n < m$,
- $\text{RcvdBit}(n, m, K)$ for receiving the result of the OT exchange is relevant for an *and* gate when $n < m$,
- $\text{Ctrb}(n, m, K)$ for storing the contribution of party m to the share of party n is relevant for an *and* gate,
- $\text{Ctrb-}\Sigma(n, m, K)$ for summing up the contributions of parties $0, \dots, m$ to the share of party n is relevant for an *and* gate,
- $\text{Share}(n, K)$ for storing the share of party n is always relevant
- $\text{OTMsg}_0(n, m, K)$, $\text{OTMsg}_1(n, m, K)$, $\text{OTMsg}_2(n, m, K)$, $\text{OTMsg}_3(n, m, K)$ for the OT exchange from n to m are relevant for an *and* gate if $n < m$
- $\text{OTChc}_0(n, m, K)$, $\text{OTChc}_1(n, m, K)$ 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 as follows:

- $\text{Circ}_n(\epsilon, 0)$ is the protocol 0
- $\text{Circ}_n(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Circ}_n(C, K)$ with the following protocol. Our share is the input share as determined in the initial part of the protocol:

$$\begin{aligned}
 & - \text{Share}(n, K) := \text{read InShare}(n, p, i) \\
 & - \begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{if } n \text{ honest} \end{cases}
 \end{aligned}$$

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,

- $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K)$ for $m \leq N + 1$
- $\text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K)$ for $m \leq N + 1$
- $\text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K)$ for $m \leq N + 1$
- $\text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K)$ for $m \leq N + 1$

nor as a receiver:

- $\text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K)$ for $m \leq N + 1$
- $\text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K)$ for $m \leq N + 1$

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

- $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $m \leq N + 1$
- $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $m \leq N + 1$

There is nothing to receive from the OT functionality:

- $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $m \leq N + 1$
- $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $m \leq N + 1$

The individual contributions of each party to our share are likewise not needed:

- $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $m \leq N + 1$
- $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $m \leq N + 1$

There is thus nothing to sum up:

- $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $m \leq N + 1$
- $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $m \leq N + 1$
- $\text{Circ}_n(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Circ}_n(C, K)$ with the following protocol. Our share is either the share on wire k (for parties $n \leq N$) or its negation (for party $N + 1$):

$$\begin{aligned}
 & - \begin{cases} \text{Share}(n, K) := \text{read Share}(n, k) \\ \text{for } n \leq N \\ \text{Share}(N + 1, K) := x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ret } \neg x_{N+1} \end{cases} \\
 & - \begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{if } n \text{ honest} \end{cases}
 \end{aligned}$$

The 1-Out-Of-4 OT exchange with every other party is again vacuous – party n does not function as a sender,

- $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K)$ for $m \leq N + 1$
- $\text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K)$ for $m \leq N + 1$
- $\text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K)$ for $m \leq N + 1$
- $\text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K)$ for $m \leq N + 1$

nor as a receiver:

- $\text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K)$ for $m \leq N + 1$
- $\text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K)$ for $m \leq N + 1$

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

- $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $m \leq N + 1$
- $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $m \leq N + 1$

There is nothing to receive from the OT functionality:

- $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $m \leq N + 1$
- $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $m \leq N + 1$

The individual contributions of each party to our share are likewise not needed:

- $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $m \leq N + 1$
- $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $m \leq N + 1$

There is thus nothing to sum up:

- $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $m \leq N + 1$
- $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $m \leq N + 1$
- $\text{Circ}_n(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Circ}_n(C, K)$ with the following protocol. Our share is the sum of shares on wires k and l :
 - $\text{Share}(n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n \oplus y_n$

$$- \begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \quad \text{if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{if } n \text{ honest} \end{cases}$$

The 1-Out-Of-4 OT exchange with every other party is once more vacuous – party n does not function as a sender,

- $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K)$ for $m \leq N + 1$
- $\text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K)$ for $m \leq N + 1$
- $\text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K)$ for $m \leq N + 1$
- $\text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K)$ for $m \leq N + 1$

nor as a receiver:

- $\text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K)$ for $m \leq N + 1$
- $\text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K)$ for $m \leq N + 1$

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

- $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $m \leq N + 1$
- $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $m \leq N + 1$

There is nothing to receive from the OT functionality:

- $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $m \leq N + 1$
- $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $m \leq N + 1$

The individual contributions of each party to our share are likewise not needed:

- $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $m \leq N + 1$
- $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $m \leq N + 1$

There is thus nothing to sum up:

- $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $m \leq N + 1$
- $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $m \leq N + 1$

- $\text{Circ}_n(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Circ}_n(C, K)$ 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:

$$- \begin{cases} \text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ samp flip} \\ \quad \text{for } m \leq N + 1 \text{ if } n < m \\ \text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \\ \quad \text{for } m \leq N + 1 \text{ if } n \geq m \end{cases}$$

$$- \begin{cases} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K) \\ \quad \text{for } m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$$

We now carry out the 1-Out-Of-4 OT exchanges where n is the sender:

$$\begin{aligned}
& \left\{ \begin{array}{l} \text{OTMsg}_0(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \\ \text{for } m \leq N + 1 \text{ if } n < m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K) \\ \text{for } m \leq N + 1 \text{ if } n \geq m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTMsg}_1(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \\ \text{for } m \leq N + 1 \text{ if } n < m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K) \\ \text{for } m \leq N + 1 \text{ if } n \geq m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTMsg}_2(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus y_n \\ \text{for } m \leq N + 1 \text{ if } n < m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K) \\ \text{for } m \leq N + 1 \text{ if } n \geq m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTMsg}_3(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \oplus y_n \\ \text{for } m \leq N + 1 \text{ if } n < m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K) \\ \text{for } m \leq N + 1 \text{ if } n \geq m \end{array} \right.
\end{aligned}$$

Next we carry out the 1-Out-Of-4 OT exchanges where n is the receiver:

$$\begin{aligned}
& \left\{ \begin{array}{l} \text{OTChc}_0(m, n, K) := \text{read Share}(n, k) \\ \text{for } m \leq N + 1 \text{ if } m < n \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K) \\ \text{for } m \leq N + 1 \text{ if } m \geq n \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTChc}_1(m, n, K) := \text{read Share}(n, l) \\ \text{for } m \leq N + 1 \text{ if } m < n \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K) \\ \text{for } m \leq N + 1 \text{ if } m \geq n \end{array} \right.
\end{aligned}$$

We now record the bits we received from the 1-Out-Of-4 OT exchanges:

$$\begin{aligned}
& \text{RcvdBit}(n, m, K) := \text{OTOut}(m, n, K) \text{ for } m \leq N + 1 \\
& \left\{ \begin{array}{l} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K) \\ \text{for } m \leq N + 1 \text{ if } n \text{ semi-honest} \end{array} \right. \\
& \left\{ \begin{array}{l} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.
\end{aligned}$$

For each party m , we compute its contribution to our share as follows:

$$\begin{aligned}
& \left\{ \begin{array}{l} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } m \leq N + 1 \text{ if } n < m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \text{for } m \leq N + 1 \text{ if } m < n \end{array} \right. \\
& \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n * y_n \\
& \left\{ \begin{array}{l} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K) \\ \text{for } m \leq N + 1 \text{ if } n \text{ semi-honest} \end{array} \right. \\
& \left\{ \begin{array}{l} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.
\end{aligned}$$

We now inductively sum up the individual contributions we generated above:

$$\begin{aligned}
& - \begin{cases} \text{Ctrb-}\Sigma(n, 0, K) := \text{read Ctrb}(n, 0, K) \\ \text{Ctrb-}\Sigma(n, m + 1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(n, m, K); b \leftarrow \text{Ctrb}(n, m + 1, K); \text{ret } b_\Sigma \oplus b \\ \text{for } m \leq N \end{cases} \\
& - \begin{cases} \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K) \\ \text{for } m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N + 1 \text{ if } n \text{ honest} \end{cases}
\end{aligned}$$

At last, we declare our share to be the total sum of the contributions:

$$\begin{aligned}
& - \text{Share}(n, K) := \text{read Ctrb-}\Sigma(n, N + 1, K) \\
& - \begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{if } n \text{ honest} \end{cases}
\end{aligned}$$

Finally, the channels

- $\text{SendBit}(n, m, k)$ for $m \leq N + 1$ and $k < K$,
- $\text{RcvdBit}(n, m, k)$ for $m \leq N + 1$ and $k < K$,
- $\text{Ctrb}(n, m, k)$ for $m \leq N + 1$ and $k < K$,
- $\text{Ctrb-}\Sigma(n, m, k)$ for $m \leq N + 1$ and $k < K$

are declared as internal.

10.3.3 The Final Phase

For each output wire, we broadcast our share:

$$\begin{aligned}
& \bullet \begin{cases} \text{SendOutShare}(m, n, k) := \text{read Share}(n, k) \\ \text{for } m \leq N + 1 \text{ and } k < K \text{ if wire } k \text{ an output} \\ \text{SendOutShare}(m, n, k) := \text{read SendOutShare}(m, n, k) \\ \text{for } m \leq N + 1 \text{ and } k < K \text{ if wire } k \text{ not an output} \end{cases} \\
& \bullet \begin{cases} \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k) \\ \text{for } m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}
\end{aligned}$$

If party n is semi-honest, each output share received is forwarded to the adversary:

$$\bullet \begin{cases} \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(n, m, k) \\ \text{for } m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$$

The incoming output shares are recorded:

- $\text{OutShare}(n, m, k) := \text{read SendOutShare}(n, m, k)$ for $m \leq N + 1$ and $k < K$

- $\begin{cases} \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k) \\ \text{for } m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

We now inductively sum up the output shares we recorded above:

- $\begin{cases} \text{OutShare-}\Sigma(n, 0, k) := \text{read OutShare}(n, 0, k) \\ \text{for } k < K \\ \text{OutShare-}\Sigma(n, m + 1, k) := x_{\Sigma} \leftarrow \text{OutShare-}\Sigma(n, m, k); x_{m+1} \leftarrow \text{OutShare}(n, m + 1, k); \text{ret } x_{\Sigma} \oplus x_{m+1} \\ \text{for } m \leq N \text{ and } k < K \end{cases}$
- $\begin{cases} \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k) \\ \text{for } m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

Finally, we declare the output to be the total sum of the output shares:

- $\text{Out}(n, k) := \text{read OutShare-}\Sigma(n, N + 1, k) \text{ for } k < K$
- $\begin{cases} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k) \\ \text{for } k < K \text{ if } n \text{ semi-honest} \\ \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } k < K \text{ if } n \text{ honest} \end{cases}$

Finally, the channels

- $\text{OutShare}(n, m, k) \text{ for } m \leq N + 1 \text{ and } k < K,$
- $\text{OutShare-}\Sigma(n, -, -) \text{ for } m \leq N + 1 \text{ and } k < K$

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 $1\text{OutOf4OT}(n, m, k)$, 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:

- $\begin{cases} \text{OTMsg}_0(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, k) \\ \text{if } n \text{ semi-honest} \\ \text{OTMsg}_0(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, k)_{\text{adv}}^{\text{ot}} \\ \text{if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{OTMsg}_1(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, k) \\ \text{if } n \text{ semi-honest} \\ \text{OTMsg}_1(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, k)_{\text{adv}}^{\text{ot}} \\ \text{if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{OTMsg}_2(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, k) \\ \text{if } n \text{ semi-honest} \\ \text{OTMsg}_2(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, k)_{\text{adv}}^{\text{ot}} \\ \text{if } n \text{ honest} \end{cases}$

$$\bullet \begin{cases} \text{OTMsg}_3(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, k) \\ \quad \text{if } n \text{ semi-honest} \\ \text{OTMsg}_3(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, k)_{\text{adv}}^{\text{ot}} \\ \quad \text{if } n \text{ honest} \end{cases}$$

Otherwise the functionality only lets the adversary know that a message from the sender has been received:

$$\bullet \begin{cases} \text{OTMsgRcvd}_0(n, m, k)_{\text{adv}}^{\text{ot}} := m_0 \leftarrow \text{OTMsg}_0(n, m, k); \text{ ret } \checkmark \\ \quad \text{if } n \text{ honest} \\ \text{OTMsgRcvd}_0(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, k)_{\text{adv}}^{\text{ot}} \\ \quad \text{if } n \text{ semi-honest} \end{cases}$$

$$\bullet \begin{cases} \text{OTMsgRcvd}_1(n, m, k)_{\text{adv}}^{\text{ot}} := m_1 \leftarrow \text{OTMsg}_1(n, m, k); \text{ ret } \checkmark \\ \quad \text{if } n \text{ honest} \\ \text{OTMsgRcvd}_1(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, k)_{\text{adv}}^{\text{ot}} \\ \quad \text{if } n \text{ semi-honest} \end{cases}$$

$$\bullet \begin{cases} \text{OTMsgRcvd}_2(n, m, k)_{\text{adv}}^{\text{ot}} := m_2 \leftarrow \text{OTMsg}_2(n, m, k); \text{ ret } \checkmark \\ \quad \text{if } n \text{ honest} \\ \text{OTMsgRcvd}_2(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, k)_{\text{adv}}^{\text{ot}} \\ \quad \text{if } n \text{ semi-honest} \end{cases}$$

$$\bullet \begin{cases} \text{OTMsgRcvd}_3(n, m, k)_{\text{adv}}^{\text{ot}} := m_3 \leftarrow \text{OTMsg}_3(n, m, k); \text{ ret } \checkmark \\ \quad \text{if } n \text{ honest} \\ \text{OTMsgRcvd}_3(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, k)_{\text{adv}}^{\text{ot}} \\ \quad \text{if } n \text{ semi-honest} \end{cases}$$

Analogously, if the receiver is semi-honest, the functionality leaks the value of all messages from the receiver:

$$\bullet \begin{cases} \text{OTChc}_0(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, k) \\ \quad \text{if } m \text{ semi-honest} \\ \text{OTChc}_0(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, k)_{\text{adv}}^{\text{ot}} \\ \quad \text{if } m \text{ honest} \end{cases}$$

$$\bullet \begin{cases} \text{OTChc}_1(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, k) \\ \quad \text{if } m \text{ semi-honest} \\ \text{OTChc}_1(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, k)_{\text{adv}}^{\text{ot}} \\ \quad \text{if } m \text{ honest} \end{cases}$$

Otherwise the functionality only lets the adversary know that a message from the receiver has been received:

$$\bullet \begin{cases} \text{OTChcRcvd}_0(n, m, k)_{\text{adv}}^{\text{ot}} := c_0 \leftarrow \text{OTChc}_0(n, m, k); \text{ ret } \checkmark \\ \quad \text{if } m \text{ honest} \\ \text{OTChcRcvd}_0(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, k)_{\text{adv}}^{\text{ot}} \\ \quad \text{if } m \text{ semi-honest} \end{cases}$$

$$\bullet \begin{cases} \text{OTChcRcvd}_1(n, m, k)_{\text{adv}}^{\text{ot}} := c_1 \leftarrow \text{OTChc}_1(n, m, k); \text{ ret } \checkmark \\ \quad \text{if } m \text{ honest} \\ \text{OTChcRcvd}_1(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, k)_{\text{adv}}^{\text{ot}} \\ \quad \text{if } m \text{ semi-honest} \end{cases}$$

The functionality then selects the appropriate message:

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

If the receiver is semi-honest, the functionality leaks the selected message to the adversary:

- $\begin{cases} \text{OTOut}(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, k) \\ \text{if } m \text{ semi-honest} \\ \text{OTOut}(n, m, k)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, k)_{\text{adv}}^{\text{ot}} \\ \text{if } m \text{ honest} \end{cases}$

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

- $\text{InShare}(n, m, i)$ for $m \leq N + 1$ and $i < I_m$,
- $\text{Share}(n, k)$ 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 $n, m \leq N + 1$,

- $\text{1OutOf4OT}(n, m, k)$ for $k < K$,

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

- $\text{SendInShare}(m, n, i)$ for $m, n \leq N + 1$ and $i < I_n$,
- $\text{OTMsg}_0(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{OTMsg}_1(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{OTMsg}_2(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{OTMsg}_3(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{OTChc}_0(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{OTChc}_1(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{OTOut}(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{SendOutShare}(m, n, k)$ for $m, n \leq N + 1$ 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

- $\text{InShare}(m, n, i)$ for $m, n \leq N + 1$ and $i < I_n$,
- $\text{Share}(n, k)$ for $n \leq N + 1$ 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`:

- $\begin{cases} \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read In}(n, i) \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read In}(n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := x \leftarrow \text{In}(n, i); \text{ ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$
- $\text{InShare-}\$(m, n, i) := x \leftarrow \text{In}(n, i); \text{ samp flip for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n$
- $\begin{cases} \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i) \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{InShare-}\$-\Sigma(0, n, i) := \text{read InShare-}\$(0, n, i) \\ \text{for } n \leq N + 1 \text{ and } i < I_n \\ \text{InShare-}\$-\Sigma(m + 1, n, i) := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(m, n, i); x_{m+1} \leftarrow \text{InShare-}\$(m + 1, n, i); \text{ ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \text{ and } n \leq N + 1 \text{ and } i < I_n \end{cases}$
- $\begin{cases} \text{InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$-\Sigma(m, n, i) \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\text{InShare-}\$(N + 1, n, i) := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i); \text{ ret } x_\Sigma \oplus x \text{ for } n \leq N + 1 \text{ and } i < I_n$
- $\begin{cases} \text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(N + 1, n, i) \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\text{SendInShare}(m, n, i) := \text{read InShare-}\$(m, n, i) \text{ for } m, n \leq N + 1 \text{ and } i < I_n$
- $\begin{cases} \text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(m, n, i) \\ \text{for } m, n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m, n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read SendInShare}(m, n, i) \\ \text{for } m, n \leq N + 1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \text{RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} \\ \text{for } m, n \leq N + 1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$
- $\text{InShare}(m, n, i) := \text{read SendInShare}(m, n, i) \text{ for } m, n \leq N + 1 \text{ and } i < I_n$

$$\bullet \begin{cases} \text{InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read InShare}(m, n, i) \\ \quad \text{for } m, n \leq N + 1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \text{InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} \\ \quad \text{for } m, n \leq N + 1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$$

This is followed by the hiding of the channels

- $\text{InShare-}\$(m, n, i)$ for $m, n \leq N + 1$ and $i < I_n$,
- $\text{InShare-}\$(m, n, i)$ for $m \leq N$ and $n \leq N + 1$ and $i < I_n$,
- $\text{SendInShare}(m, n, i)$ for $m, n \leq N + 1$ 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 $\text{Circ}(C, K)$:

- $\text{Circ}(\epsilon, 0)$ is the protocol 0
- $\text{Circ}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Circ}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := \text{read InShare}(n, p, i)$ for $n \leq N + 1$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \quad \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K)$ for $n, m \leq N + 1$
 - $\begin{cases} \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K) \\ \quad \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \quad \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\begin{cases} \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K) \\ \quad \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \quad \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := m_0 \leftarrow \text{OTMsg}_0(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \\ \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := m_1 \leftarrow \text{OTMsg}_1(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \\ \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := m_2 \leftarrow \text{OTMsg}_2(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \\ \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := m_3 \leftarrow \text{OTMsg}_3(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \\ \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K) \text{ for } m, n \leq N + 1$
- $\text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K) \text{ for } m, n \leq N + 1$
- $\begin{cases} \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := c_0 \leftarrow \text{OTChc}_0(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \\ \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_1(m, n, K)_{\text{adv}}^{\text{ot}} := c_0 \leftarrow \text{OTChc}_1(m, n, K); \text{ ret } \checkmark \\ \text{for } m, n \leq N + 1 \text{ if } m \text{ honest} \\ \text{OTChcRcvd}_1(m, n, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(m, n, K)_{\text{adv}}^{\text{ot}} \\ \text{for } m, n \leq N + 1 \text{ if } m \text{ semi-honest} \end{cases}$

- $\text{OTOut}(n, m, K) := m_0 \leftarrow \text{OTMsg}_0(n, m, K); m_1 \leftarrow \text{OTMsg}_1(n, m, K); m_2 \leftarrow \text{OTMsg}_2(n, m, K);$
 $m_3 \leftarrow \text{OTMsg}_3(n, m, K); c_0 \leftarrow \text{OTChc}_0(n, m, K); c_1 \leftarrow \text{OTChc}_1(n, m, K);$
 if c_0 then (if c_1 then ret m_3 else ret m_2) else (if c_1 then ret m_1 else ret m_0) for $n, m \leq N + 1$
- $\begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}$
- $\text{Circ}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Circ}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\begin{cases} \text{Share}(n, K) := \text{read Share}(n, k) \\ \text{for } n \leq N \\ \text{Share}(N + 1, K) := x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ret } \neg x_{N+1} \end{cases}$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K)$ for $n, m \leq N + 1$
 - $\begin{cases} \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\begin{cases} \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\begin{cases} \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$

- $\begin{cases} \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := m_0 \leftarrow \text{OTMsg}_0(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \\ \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := m_1 \leftarrow \text{OTMsg}_1(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \\ \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := m_2 \leftarrow \text{OTMsg}_2(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \\ \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := m_3 \leftarrow \text{OTMsg}_3(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \\ \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K) \text{ for } m, n \leq N + 1$
- $\text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K) \text{ for } m, n \leq N + 1$
- $\begin{cases} \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}$
- $\begin{cases} \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := c_0 \leftarrow \text{OTChc}_0(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \\ \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := c_0 \leftarrow \text{OTChc}_1(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \\ \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \end{cases}$
- $\text{OTOut}(n, m, K) := m_0 \leftarrow \text{OTMsg}_0(n, m, K); m_1 \leftarrow \text{OTMsg}_1(n, m, K); m_2 \leftarrow \text{OTMsg}_2(n, m, K);$
 $m_3 \leftarrow \text{OTMsg}_3(n, m, K); c_0 \leftarrow \text{OTChc}_0(n, m, K); c_1 \leftarrow \text{OTChc}_1(n, m, K);$
 $\text{if } c_0 \text{ then (if } c_1 \text{ then ret } m_3 \text{ else ret } m_2) \text{ else (if } c_1 \text{ then ret } m_1 \text{ else ret } m_0) \text{ for } n, m \leq N + 1$

$$- \begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}$$

- $\text{Circ}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Circ}(C, K)$ with the protocol

$$\begin{aligned}
& - \text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \text{ for } n, m \leq N + 1 \\
& - \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N + 1 \\
& - \text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K) \text{ for } n, m \leq N + 1 \\
& - \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N + 1 \\
& - \text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K) \text{ for } n, m \leq N + 1 \\
& - \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N + 1 \\
& - \text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K) \text{ for } n, m \leq N + 1 \\
& - \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N + 1 \\
& - \text{Share}(n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n \oplus y_n \text{ for } n \leq N + 1 \\
& - \begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases} \\
& - \text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K) \text{ for } n, m \leq N + 1 \\
& - \text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K) \text{ for } n, m \leq N + 1 \\
& - \text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K) \text{ for } n, m \leq N + 1 \\
& - \text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K) \text{ for } n, m \leq N + 1 \\
& - \begin{cases} \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases} \\
& - \begin{cases} \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases} \\
& - \begin{cases} \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases} \\
& - \begin{cases} \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases} \\
& - \begin{cases} \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := m_0 \leftarrow \text{OTMsg}_0(n, m, K); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \\ \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}
\end{aligned}$$

- $\begin{cases} \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := m_1 \leftarrow \text{OTMsg}_1(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \\ \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := m_2 \leftarrow \text{OTMsg}_2(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \\ \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := m_3 \leftarrow \text{OTMsg}_3(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \\ \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K) \text{ for } m, n \leq N + 1$
- $\text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K) \text{ for } m, n \leq N + 1$
- $\begin{cases} \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}$
- $\begin{cases} \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := c_0 \leftarrow \text{OTChc}_0(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \\ \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := c_0 \leftarrow \text{OTChc}_1(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \\ \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \end{cases}$
- $\text{OTOut}(n, m, K) := m_0 \leftarrow \text{OTMsg}_0(n, m, K); m_1 \leftarrow \text{OTMsg}_1(n, m, K); m_2 \leftarrow \text{OTMsg}_2(n, m, K);$
 $m_3 \leftarrow \text{OTMsg}_3(n, m, K); c_0 \leftarrow \text{OTChc}_0(n, m, K); c_1 \leftarrow \text{OTChc}_1(n, m, K);$
if c_0 then (if c_1 then ret m_3 else ret m_2) else (if c_1 then ret m_1 else ret m_0) for $n, m \leq N + 1$
- $\begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}$

- $\text{Circ}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Circ}(C, K)$ with the protocol

- $\begin{cases} \text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ samp flip} \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{cases}$

- $\left\{ \begin{array}{l} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.$
- $\text{RcvdBit}(n, m, K) := \text{OTOut}(n, m, K) \text{ for } n, m \leq N + 1$
- $\left\{ \begin{array}{l} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n * y_n \\ \text{for } n \leq N + 1 \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Ctrb-}\Sigma(n, 0, K) := \text{read Ctrb}(n, 0, K) \\ \text{for } n \leq N + 1 \\ \text{Ctrb-}\Sigma(n, m + 1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(n, m, K); b \leftarrow \text{Ctrb}(n, m + 1, K); \text{ret } b_\Sigma \oplus b \\ \text{for } n \leq N + 1 \text{ and } m \leq N \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.$
- $\text{Share}(n, K) := \text{read Ctrb-}\Sigma(n, N + 1, K) \text{ for } n \leq N + 1$
- $\left\{ \begin{array}{l} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OTMsg}_0(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OTMsg}_1(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{array} \right.$

$\text{OTMsg}_2(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus y_n$
 for $n, m \leq N + 1$ if $n < m$
 $\text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K)$
 for $n, m \leq N + 1$ if $n \geq m$
 $\text{OTMsg}_3(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \oplus y_n$
 for $n, m \leq N + 1$ if $n < m$
 $\text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K)$
 for $n, m \leq N + 1$ if $n \geq m$
 $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)$
 for $n, m \leq N + 1$ if n semi-honest
 $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n honest
 $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)$
 for $n, m \leq N + 1$ if n semi-honest
 $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n honest
 $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)$
 for $n, m \leq N + 1$ if n semi-honest
 $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n honest
 $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)$
 for $n, m \leq N + 1$ if n semi-honest
 $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n honest
 $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := m_0 \leftarrow \text{OTMsg}_0(n, m, K); \text{ret } \checkmark$
 for $n, m \leq N + 1$ if n honest
 $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n semi-honest
 $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := m_1 \leftarrow \text{OTMsg}_1(n, m, K); \text{ret } \checkmark$
 for $n, m \leq N + 1$ if n honest
 $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n semi-honest
 $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := m_2 \leftarrow \text{OTMsg}_2(n, m, K); \text{ret } \checkmark$
 for $n, m \leq N + 1$ if n honest
 $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n semi-honest
 $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := m_3 \leftarrow \text{OTMsg}_3(n, m, K); \text{ret } \checkmark$
 for $n, m \leq N + 1$ if n honest
 $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n semi-honest
 $\text{OTChc}_0(m, n, K) := \text{read Share}(n, k)$
 for $m, n \leq N + 1$ if $m < n$
 $\text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K)$
 for $m, n \leq N + 1$ if $m \geq n$

$$\begin{aligned}
& - \begin{cases} \text{OTChc}_1(m, n, K) := \text{read Share}(n, l) \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K) \\ \text{for } m, n \leq N + 1 \text{ if } m \geq n \end{cases} \\
& - \begin{cases} \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases} \\
& - \begin{cases} \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases} \\
& - \begin{cases} \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := c_0 \leftarrow \text{OTChc}_0(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \\ \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \end{cases} \\
& - \begin{cases} \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := c_1 \leftarrow \text{OTChc}_1(n, m, K); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \\ \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \end{cases} \\
& - \text{OTOut}(n, m, K) := m_0 \leftarrow \text{OTMsg}_0(n, m, K); m_1 \leftarrow \text{OTMsg}_1(n, m, K); m_2 \leftarrow \text{OTMsg}_2(n, m, K); \\
& \quad m_3 \leftarrow \text{OTMsg}_3(n, m, K); c_0 \leftarrow \text{OTChc}_0(n, m, K); c_1 \leftarrow \text{OTChc}_1(n, m, K); \\
& \quad \text{if } c_0 \text{ then (if } c_1 \text{ then ret } m_3 \text{ else ret } m_2) \text{ else (if } c_1 \text{ then ret } m_1 \text{ else ret } m_0) \text{ for } n, m \leq N + 1 \\
& - \begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}
\end{aligned}$$

This is followed by the hiding of the channels

- $\text{SendBit}(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{RcvdBit}(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{Ctrb}(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{Ctrb-}\Sigma(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{OTMsg}_0(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{OTMsg}_1(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{OTMsg}_2(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{OTMsg}_3(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{OTChc}_0(m, n, k)$ for $m, n \leq N + 1$ and $k < K$,
- $\text{OTChc}_1(m, n, k)$ for $m, n \leq N + 1$ and $k < K$,
- $\text{OTOut}(n, m, k)$ for $n, m \leq N + 1$ 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:

- $\left\{ \begin{array}{l} \text{SendOutShare}(m, n, k) := \text{read Share}(n, k) \\ \quad \text{for } m, n \leq N + 1 \text{ and } k < K \text{ if wire } k \text{ an output} \\ \text{SendOutShare}(m, n, k) := \text{read SendOutShare}(m, n, k) \\ \quad \text{for } m, n \leq N + 1 \text{ and } k < K \text{ if wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k) \\ \quad \text{for } m, n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } m, n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(n, m, k) \\ \quad \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{array} \right.$
- $\text{OutShare}(n, m, k) := \text{read SendOutShare}(n, m, k) \text{ for } n, m \leq N + 1 \text{ and } k < K$
- $\left\{ \begin{array}{l} \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k) \\ \quad \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, 0, k) := \text{ret OutShare}(n, 0, k) \\ \quad \text{for } n \leq N + 1 \text{ and } k < K \\ \text{OutShare-}\Sigma(n, m + 1, k) := x_{\Sigma} \leftarrow \text{OutShare-}\Sigma(n, m, k); x_{m+1} \leftarrow \text{OutShare}(n, m + 1, k); \text{ret } x_{\Sigma} \oplus x_{m+1} \\ \quad \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k) \\ \quad \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{array} \right.$
- $\text{Out}(n, k) := \text{read OutShare-}\Sigma(n, N + 1, k) \text{ for } n \leq N + 1 \text{ and } k < K$
- $\left\{ \begin{array}{l} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k) \\ \quad \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{array} \right.$

This is followed by the hiding of the channels

- $\text{OutShare}(n, m, k) \text{ for } n, m \leq N + 1 \text{ and } k < K,$
- $\text{OutShare-}\Sigma(n, m, k) \text{ for } n, m \leq N + 1 \text{ and } k < K,$
- $\text{SendOutShare}(m, n, k) \text{ for } m, n \leq N + 1 \text{ and } k < K.$

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

- $\text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(m, n, i)$

reads from the channel

- $\text{SendInShare}(m, n, i) := \text{read InShare-}\(m, n, i)

so we can substitute:

- $\text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\(m, n, i)

We thus get the following for channels $\text{SendInShare}(-, -, -)_{\text{adv}}^{\text{party}(-)}$:

- $$\begin{cases} \text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i) \\ \text{for } m, n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m, n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$$

If party n is semi-honest, then for any party m and input $i < I_m$ the channel

- $\text{RcvdInShare}(n, m, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(n, m, i)$

reads from the channel

- $\text{SendInShare}(n, m, i) := \text{read InShare-}\(n, m, i)

so we can substitute:

- $\text{RcvdInShare}(n, m, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\(n, m, i)

We thus get the following for channels $\text{RcvdInShare}(-, -, -)_{\text{adv}}^{\text{party}(-)}$:

- $$\begin{cases} \text{RcvdInShare}(n, m, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(n, m, i) \\ \text{for } n, m \leq N + 1 \text{ and } i < I_m \text{ if } n \text{ semi-honest} \\ \text{RcvdInShare}(n, m, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(n, m, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ and } i < I_m \text{ if } n \text{ honest} \end{cases}$$

Finally, substituting the channels

- $\text{SendInShare}(m, n, i) := \text{read InShare-}\(m, n, i) for $m, n \leq N + 1$ and $i < I_n$

into the respective channels

- $\text{InShare}(m, n, i) := \text{read SendInShare}(m, n, i)$ for $m, n \leq N + 1$ and $i < I_n$

yields the definition below:

- $\text{InShare}(m, n, i) := \text{read InShare-}\(m, n, i) for $m, n \leq N + 1$ and $i < I_n$

At this point, the internal channels $\text{SendInShare}(-, -, -)$ are unused and can be eliminated. The simplified version Init of the initial part of the real protocol is therefore as follows:

- $$\begin{cases} \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read In}(n, i) \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read In}(n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$$

- $$\begin{cases} \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := x \leftarrow \text{In}(n, i); \text{ ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$$
- $$\text{InShare-}\$(m, n, i) := x \leftarrow \text{In}(n, i); \text{ samp flip for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n$$
- $$\begin{cases} \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i) \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$$
- $$\begin{cases} \text{InShare-}\$-\Sigma(0, n, i) := \text{read InShare-}\$(0, n, i) \\ \text{for } n \leq N + 1 \text{ and } i < I_n \\ \text{InShare-}\$-\Sigma(m + 1, n, i) := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(m, n, i); x_{m+1} \leftarrow \text{InShare-}\$(m + 1, n, i); \text{ ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \text{ and } n \leq N + 1 \text{ and } i < I_n \end{cases}$$
- $$\begin{cases} \text{InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$-\Sigma(m, n, i) \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$$
- $$\text{InShare-}\$(N + 1, n, i) := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i); \text{ ret } x_\Sigma \oplus x \text{ for } n \leq N + 1 \text{ and } i < I_n$$
- $$\begin{cases} \text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(N + 1, n, i) \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$$
- $$\begin{cases} \text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i) \\ \text{for } m, n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m, n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$$
- $$\begin{cases} \text{RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read InShare-}\$(m, n, i) \\ \text{for } m, n \leq N + 1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \text{RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} \\ \text{for } m, n \leq N + 1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$$
- $$\text{InShare}(m, n, i) := \text{read InShare-}\$(m, n, i) \text{ for } m, n \leq N + 1 \text{ and } i < I_n$$
- $$\begin{cases} \text{InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read InShare}(m, n, i) \\ \text{for } m, n \leq N + 1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \text{InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} \\ \text{for } m, n \leq N + 1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$$

This is followed by the hiding of the channels

- $\text{InShare-}\$(m, n, i)$ for $m, n \leq N + 1$ and $i < I_n$,
- $\text{InShare-}\$-\Sigma(m, n, i)$ for $m \leq N$ and $n \leq N + 1$ 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

- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)$

reads from the divergent channel

- $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K)$

and so we may equivalently write the following:

- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$

The simplified definition for channels $\text{OTMsg}_0(-, -, K)_{\text{adv}}^{\text{ot}}$ is thus as follows:

- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$

If party n is honest, for any party m the channel

- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := m_0 \leftarrow \text{OTMsg}_0(n, m, K); \text{ ret } \checkmark$

reads from the divergent channel

- $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K)$

and so we may equivalently write the following:

- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$

The simplified definition for channels $\text{OTMsgRcvd}_0(-, -, K)_{\text{adv}}^{\text{ot}}$ is thus as follows:

- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$

We treat the receiver channels analogously. For instance, if party m is semi-honest, then for any party n the channel

- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)$

reads from the divergent channel

- $\text{OTChc}_0(n, m, K) := \text{read OTChc}_0(n, m, K)$

and so we may equivalently write the following:

- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$

The simplified definition for channels $\text{OTChc}_0(-, -, K)_{\text{adv}}^{\text{ot}}$ is thus as follows:

- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$

If party m is honest, for any party n the channel

- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := c_0 \leftarrow \text{OTChc}_0(n, m, K); \text{ ret } \checkmark$

reads from the divergent channel

- $\text{OTChc}_0(n, m, K) := \text{read OTChc}_0(n, m, K)$

and so we may equivalently write the following:

- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$

The simplified definition for channels $\text{OTChcRcvd}_0(-, -, K)_{\text{adv}}^{\text{ot}}$ is thus as follows:

- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$

Finally, for any parties n, m the channel

- $\text{OTOut}(n, m, K) := m_0 \leftarrow \text{OTMsg}_0(n, m, K); m_1 \leftarrow \text{OTMsg}_1(n, m, K); m_2 \leftarrow \text{OTMsg}_2(n, m, K);$
 $m_3 \leftarrow \text{OTMsg}_3(n, m, K); c_0 \leftarrow \text{OTChc}_0(n, m, K); c_1 \leftarrow \text{OTChc}_1(n, m, K);$
 if c_0 then (if c_1 then ret m_3 else ret m_2) else (if c_1 then ret m_1 else ret m_0)

reads from the divergent channel

- $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K)$

and so we may equivalently write the following:

- $\text{OTOut}(n, m, K) := \text{read OTOut}(n, m, K)$

If party m is semi-honest, this in turn means that the channel

- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)$

reads from a divergent channel, and so we may equivalently write the following:

- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$

The simplified definition for channels $\text{OTOut}(-, -, K)_{\text{adv}}^{\text{ot}}$ is thus as follows:

- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$

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 $n \geq m$ the channel

- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)$

reads from the divergent channel

- $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K)$

and so we may equivalently write the following:

- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$

On the other hand, if $n < m$ then the channel

- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)$

reads from the channel

- $\text{OTMsg}_0(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b$

so we may substitute:

- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b$

Summarizing the above, we get the following for channels $\text{OTMsg}_0(-, -, K)_{\text{adv}}^{\text{ot}}$:

- $$\begin{cases} \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$$

Analogously for channels $\text{OTMsg}_1(-, -, K)_{\text{adv}}^{\text{ot}}$, $\text{OTMsg}_2(-, -, K)_{\text{adv}}^{\text{ot}}$, $\text{OTMsg}_3(-, -, K)_{\text{adv}}^{\text{ot}}$:

- $\begin{cases} \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$
- $\begin{cases} \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus y_n \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$
- $\begin{cases} \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \oplus y_n \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$

If party n is honest, for any party m with $n \geq m$ the channel

- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := m_0 \leftarrow \text{OTMsg}_0(n, m, K); \text{ret } \checkmark$

reads from the divergent channel

- $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K)$

and so we may equivalently write the following:

- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$

On the other hand, if $n < m$ then the channel

- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := m_0 \leftarrow \text{OTMsg}_0(n, m, K); \text{ret } \checkmark$

reads from the channel

- $\text{OTMsg}_0(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b$

so we may substitute:

- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark$

Summarizing the above, we get the following for channels $\text{OTMsgRcvd}_0(-, -, K)_{\text{adv}}^{\text{ot}}$:

- $\begin{cases} \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases}$

Analogously for channels $\text{OTMsgRcvd}_1(-, -, K)_{\text{adv}}^{\text{ot}}$, $\text{OTMsgRcvd}_2(-, -, K)_{\text{adv}}^{\text{ot}}$, $\text{OTMsgRcvd}_3(-, -, K)_{\text{adv}}^{\text{ot}}$:

- $\begin{cases} \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases}$
- $\begin{cases} \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases}$

- $$\begin{cases} \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases}$$

We treat the receiver channels similarly. For instance, if party m is semi-honest, then for any party n with $n \geq m$ the channel

- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)$

reads from the divergent channel

- $\text{OTChc}_0(n, m, K) := \text{read OTChc}_0(n, m, K)$

and so we may equivalently write the following:

- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$

On the other hand, if $n < m$ then the channel

- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)$

reads from the channel

- $\text{OTChc}_0(n, m, K) := \text{read Share}(m, k)$

so we may substitute:

- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, k)$

Summarizing the above, we get the following for channels $\text{OTChc}_0(-, -, K)_{\text{adv}}^{\text{ot}}$:

- $$\begin{cases} \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, k) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest and } n < m \\ \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest or } n \geq m \end{cases}$$

Analogously for channels $\text{OTChc}_1(-, -, K)_{\text{adv}}^{\text{ot}}$:

- $$\begin{cases} \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, l) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest and } n < m \\ \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest or } n \geq m \end{cases}$$

If party m is honest, for any party n with $n \geq m$ the channel

- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := c_0 \leftarrow \text{OTChc}_0(n, m, K); \text{ret } \checkmark$

reads from the divergent channel

- $\text{OTChc}_0(n, m, K) := \text{read OTChc}_0(n, m, K)$

and so we may equivalently write the following:

- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$

On the other hand, if $n < m$ then the channel

- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := c_0 \leftarrow \text{OTChc}_0(n, m, K); \text{ret } \checkmark$

reads from the channel

- $\text{OTChc}_0(n, m, K) := \text{read Share}(m, k)$

so we may substitute:

- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := x_m \leftarrow \text{Share}(m, k); \text{ret } \checkmark$

Summarizing the above, we get the following for channels $\text{OTChcRcvd}_0(-, -, K)_{\text{adv}}^{\text{ot}}$:

- $\begin{cases} \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := x_m \leftarrow \text{Share}(m, k); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest or } n \geq m \end{cases}$

Analogously for channels $\text{OTChcRcvd}_1(-, -, K)_{\text{adv}}^{\text{ot}}$:

- $\begin{cases} \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := y_m \leftarrow \text{Share}(m, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest or } n \geq m \end{cases}$

At last, if party m is semi-honest, for any party n the channel

- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)$

has the same definition as the channel

- $\text{RcvdBit}(m, n, K) := \text{read OTOut}(n, m, K)$

so we may equivalently write the following:

- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read RcvdBit}(m, n, K)$

We thus get the following for channels $\text{OTOut}(-, -, K)_{\text{adv}}^{\text{ot}}$:

- $\begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read RcvdBit}(m, n, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}$

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

- $\text{RcvdBit}(n, m, K) := \text{OTOut}(m, n, K)$

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

- $\text{OTOut}(m, n, K) := m_0 \leftarrow \text{OTMsg}_0(m, n, K); m_1 \leftarrow \text{OTMsg}_1(m, n, K); m_2 \leftarrow \text{OTMsg}_2(m, n, K);$
 $m_3 \leftarrow \text{OTMsg}_3(m, n, K); c_0 \leftarrow \text{OTChc}_0(m, n, K); c_1 \leftarrow \text{OTChc}_1(m, n, K);$
 if c_0 then (if c_1 then ret m_3 else ret m_2) else (if c_1 then ret m_1 else ret m_0)

into the above thus yields the somewhat more verbose

- $\text{RcvdBit}(n, m, K) := m_0 \leftarrow \text{OTMsg}_0(m, n, K); m_1 \leftarrow \text{OTMsg}_1(m, n, K); m_2 \leftarrow \text{OTMsg}_2(m, n, K);$
 $m_3 \leftarrow \text{OTMsg}_3(m, n, K); c_0 \leftarrow \text{OTChc}_0(m, n, K); c_1 \leftarrow \text{OTChc}_1(m, n, K);$
 if c_0 then (if c_1 then ret m_3 else ret m_2) else (if c_1 then ret m_1 else ret m_0)

with the advantage that it no longer mentions $\text{OTOut}(m, n, K)$. If $m < n$ we may further substitute the channels

- $\text{OTMsg}_0(m, n, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l); \text{ret } b$

- $\text{OTMsg}_1(m, n, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l); \text{ret } b \oplus x_m$
- $\text{OTMsg}_2(m, n, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l); \text{ret } b \oplus y_m$
- $\text{OTMsg}_3(m, n, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l); \text{ret } b \oplus x_m \oplus y_m$

as well as the channels

- $\text{OTChc}_0(m, n, K) := \text{read Share}(n, k)$
- $\text{OTChc}_1(m, n, K) := \text{read Share}(n, l)$

to obtain the following:

- $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{if } x_n \text{ then (if } y_n \text{ then ret } b \oplus x_m \oplus y_m \text{ else ret } b \oplus y_m) \text{ else (if } y_n \text{ then ret } b \oplus x_m \text{ else ret } b)$

Here we no longer refer to any of the OT channels. We can express the above more concisely as follows:

- $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus (x_m * y_n) \oplus (x_n * y_m)$

Finally, if $m \geq n$ the channel

- $\text{RcvdBit}(n, m, K) := m_0 \leftarrow \text{OTMsg}_0(m, n, K); m_1 \leftarrow \text{OTMsg}_1(m, n, K); m_2 \leftarrow \text{OTMsg}_2(m, n, K); m_3 \leftarrow \text{OTMsg}_3(m, n, K); c_0 \leftarrow \text{OTChc}_0(m, n, K); c_1 \leftarrow \text{OTChc}_1(m, n, K); \text{if } c_0 \text{ then (if } c_1 \text{ then ret } m_3 \text{ else ret } m_2) \text{ else (if } c_1 \text{ then ret } m_1 \text{ else ret } m_0)$

reads from the divergent channel

- $\text{OTMsg}_0(m, n, K) := \text{read OTMsg}_0(m, n, K)$

so we may equivalently write

- $\text{RcvdBit}(n, m, K) := \text{RcvdBit}(n, m, K)$

and further expand the above to

- $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus (x_m * y_n) \oplus (x_n * y_m)$

since the channel

- $\text{SendBit}(m, n, K) := \text{read SendBit}(m, n, K)$

is likewise divergent. Summarizing, we get the following for channels $\text{RcvdBit}(-, -, K)$:

- $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus (x_m * y_n) \oplus (x_n * y_m) \text{ for } n, m \leq N + 1$

All in all, we can rewrite the inductive part $\text{Circ}(C, K)$ of the real protocol as follows:

- $\text{Circ}(\epsilon, 0)$ is the protocol 0
- $\text{Circ}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Circ}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \text{ for } n, m \leq N + 1$
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K) \text{ for } n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K) \text{ for } n, m \leq N + 1$

- $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
- $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
- $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
- $\text{Share}(n, K) := \text{read InShare}(n, p, i)$ for $n \leq N + 1$
- $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K)$ for $n, m \leq N + 1$
- $\text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K)$ for $n, m \leq N + 1$
- $\text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K)$ for $n, m \leq N + 1$
- $\text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K)$ for $n, m \leq N + 1$
- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K)$ for $m, n \leq N + 1$
- $\text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K)$ for $m, n \leq N + 1$
- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K) := \text{read OTOut}(n, m, K)$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Circ}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Circ}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$

- $\begin{cases} \text{Share}(n, K) := \text{read Share}(n, k) \\ \text{for } n \leq N \\ \text{Share}(N+1, K) := x_{N+1} \leftarrow \text{Share}(N+1, k); \text{ret } \neg x_{N+1} \end{cases}$
- $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N+1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ if } n \text{ honest} \end{cases}$
- $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K) \text{ for } n, m \leq N+1$
- $\text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K) \text{ for } n, m \leq N+1$
- $\text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K) \text{ for } n, m \leq N+1$
- $\text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K) \text{ for } n, m \leq N+1$
- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
- $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
- $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
- $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
- $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
- $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
- $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
- $\text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K) \text{ for } m, n \leq N+1$
- $\text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K) \text{ for } m, n \leq N+1$
- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
- $\text{OTOut}(n, m, K) := \text{read OTOut}(n, m, K) \text{ for } n, m \leq N+1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
- $\text{Circ}(C; \text{xor-gate}(k, l), K+1)$ is the composition of $\text{Circ}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \text{ for } n, m \leq N+1$
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N+1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K) \text{ for } n, m \leq N+1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N+1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K) \text{ for } n, m \leq N+1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N+1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K) \text{ for } n, m \leq N+1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N+1$
 - $\text{Share}(n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n \oplus y_n \text{ for } n \leq N+1$

- $\left\{ \begin{array}{l} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.$
- $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K) \text{ for } n, m \leq N + 1$
- $\text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K) \text{ for } n, m \leq N + 1$
- $\text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K) \text{ for } n, m \leq N + 1$
- $\text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K) \text{ for } n, m \leq N + 1$
- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
- $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
- $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
- $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
- $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
- $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
- $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
- $\text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K) \text{ for } m, n \leq N + 1$
- $\text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K) \text{ for } m, n \leq N + 1$
- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
- $\text{OTOut}(n, m, K) := \text{read OTOut}(n, m, K) \text{ for } n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$

- $\text{Circ}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Circ}(C, K)$ with the protocol

- $\left\{ \begin{array}{l} \text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{samp flip} \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{array} \right.$
- $\left\{ \begin{array}{l} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.$
- $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l);$
 $x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus (x_m * y_n) \oplus (x_n * y_m) \text{ for } n, m \leq N + 1$
- $\left\{ \begin{array}{l} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.$

$$\begin{aligned}
& \left\{ \begin{array}{l} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \quad \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \quad \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n * y_n \\ \quad \text{for } n \leq N + 1 \end{array} \right. \\
& - \left\{ \begin{array}{l} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K) \\ \quad \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{array} \right. \\
& - \left\{ \begin{array}{l} \text{Ctrb-}\Sigma(n, 0, K) := \text{ret Ctrb}(n, 0, K) \\ \quad \text{for } n \leq N + 1 \\ \text{Ctrb-}\Sigma(n, m + 1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(n, m, K); b \leftarrow \text{Ctrb}(n, m + 1, K); \text{ret } b_\Sigma \oplus b \\ \quad \text{for } n \leq N + 1 \text{ and } m \leq N \end{array} \right. \\
& - \left\{ \begin{array}{l} \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K) \\ \quad \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{array} \right. \\
& - \text{Share}(n, K) := \text{read Ctrb-}\Sigma(n, N + 1, K) \text{ for } n \leq N + 1 \\
& - \left\{ \begin{array}{l} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \quad \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{array} \right. \\
& - \left\{ \begin{array}{l} \text{OTMsg}_0(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \\ \quad \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K) \\ \quad \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{array} \right. \\
& - \left\{ \begin{array}{l} \text{OTMsg}_1(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \\ \quad \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K) \\ \quad \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{array} \right. \\
& - \left\{ \begin{array}{l} \text{OTMsg}_2(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus y_n \\ \quad \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K) \\ \quad \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{array} \right. \\
& - \left\{ \begin{array}{l} \text{OTMsg}_3(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \oplus y_n \\ \quad \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K) \\ \quad \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{array} \right. \\
& - \left\{ \begin{array}{l} \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \\ \quad \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \quad \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{array} \right.
\end{aligned}$$

$\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n$
 for $n, m \leq N + 1$ if n semi-honest and $n < m$
 $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n honest or $n \geq m$
 $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus y_n$
 for $n, m \leq N + 1$ if n semi-honest and $n < m$
 $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n honest or $n \geq m$
 $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \oplus y_n$
 for $n, m \leq N + 1$ if n semi-honest and $n < m$
 $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n honest or $n \geq m$
 $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark$
 for $n, m \leq N + 1$ if n honest and $n < m$
 $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n semi-honest or $n \geq m$
 $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark$
 for $n, m \leq N + 1$ if n honest and $n < m$
 $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n semi-honest or $n \geq m$
 $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark$
 for $n, m \leq N + 1$ if n honest and $n < m$
 $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n semi-honest or $n \geq m$
 $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark$
 for $n, m \leq N + 1$ if n honest and $n < m$
 $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n semi-honest or $n \geq m$
 $\text{OTChc}_0(m, n, K) := \text{read Share}(n, k)$
 for $m, n \leq N + 1$ if $m < n$
 $\text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K)$
 for $m, n \leq N + 1$ if $m \geq n$
 $\text{OTChc}_1(m, n, K) := \text{read Share}(n, l)$
 for $m, n \leq N + 1$ if $m < n$
 $\text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K)$
 for $m, n \leq N + 1$ if $m \geq n$
 $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, k)$
 for $n, m \leq N + 1$ if m semi-honest and $n < m$
 $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if m honest or $n \geq m$
 $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, l)$
 for $n, m \leq N + 1$ if m semi-honest and $n < m$
 $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if m honest or $n \geq m$

$$\begin{aligned}
& - \begin{cases} \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := x_m \leftarrow \text{Share}(m, k); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest or } n \geq m \end{cases} \\
& - \begin{cases} \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := y_m \leftarrow \text{Share}(m, l); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest or } n \geq m \end{cases} \\
& - \text{OTOut}(n, m, K) := m_0 \leftarrow \text{OTMsg}_0(n, m, K); m_1 \leftarrow \text{OTMsg}_1(n, m, K); m_2 \leftarrow \text{OTMsg}_2(n, m, K); \\
& \quad m_3 \leftarrow \text{OTMsg}_3(n, m, K); c_0 \leftarrow \text{OTChc}_0(n, m, K); c_1 \leftarrow \text{OTChc}_1(n, m, K); \\
& \quad \text{if } c_0 \text{ then (if } c_1 \text{ then ret } m_3 \text{ else ret } m_2) \text{ else (if } c_1 \text{ then ret } m_1 \text{ else ret } m_0) \text{ for } n, m \leq N + 1 \\
& - \begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read RcvdBit}(m, n, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}
\end{aligned}$$

We now split the protocol $\text{Circ}(C, K)$ into three parts. The first protocol $\text{Shares}(C, K)$ performs the computation of shares and is defined as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := \text{read InShare}(n, p, i)$ for $n \leq N + 1$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\begin{cases} \text{Share}(n, K) := \text{read Share}(n, k) \\ \text{for } n \leq N \\ \text{Share}(N + 1, K) := x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ ret } \neg x_{N+1} \end{cases}$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ ret } x_n \oplus y_n$ for $n \leq N + 1$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol

- $\left\{ \begin{array}{l} \text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ samp flip} \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{array} \right.$
- $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l);$
 $x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ ret } b \oplus (x_m * y_n) \oplus (x_n * y_m) \text{ for } n, m \leq N + 1$
- $\left\{ \begin{array}{l} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ ret } x_n * y_n \\ \text{for } n \leq N + 1 \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Ctrb-}\Sigma(n, 0, K) := \text{read Ctrb}(n, 0, K) \\ \text{for } n \leq N + 1 \\ \text{Ctrb-}\Sigma(n, m + 1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(n, m, K); b \leftarrow \text{Ctrb}(n, m + 1, K); \text{ ret } b_\Sigma \oplus b \\ \text{for } n \leq N + 1 \text{ and } m \leq N \end{array} \right.$
- $\text{Share}(n, K) := \text{read Ctrb-}\Sigma(n, N + 1, K) \text{ for } n \leq N + 1$

The second protocol $\text{Adv}(C, K)$ performs all leakages and is defined as follows:

- $\text{Adv}(\epsilon, 0)$ is the protocol 0
- $\text{Adv}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N + 1$
 - $\left\{ \begin{array}{l} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.$
 - $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$

- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Adv}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Adv}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$

- $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$

- $\text{Adv}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol

- $\begin{cases} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \end{cases}$
- $\begin{cases} \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$
- $\begin{cases} \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \end{cases}$
- $\begin{cases} \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$
- $\begin{cases} \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus y_n \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \end{cases}$
- $\begin{cases} \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$

$$\begin{aligned}
- & \begin{cases} \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \oplus y_n \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, k) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest and } n < m \\ \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, l) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest and } n < m \\ \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := x_m \leftarrow \text{Share}(m, k); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := y_m \leftarrow \text{Share}(m, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read RcvdBit}(m, n, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}
\end{aligned}$$

The third protocol $1\text{OutOf4OT}(C, K)$ performs all Oblivious Transfer exchanges and is defined as follows:

- $1\text{OutOf4OT}(\epsilon, 0)$ is the protocol 0

- $1\text{OutOf4OT}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $1\text{OutOf4OT}(C, K)$ with the protocol
 - $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K)$ for $m, n \leq N + 1$
 - $\text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K)$ for $m, n \leq N + 1$
 - $\text{OTOut}(n, m, K) := \text{read OTOut}(n, m, K)$ for $n, m \leq N + 1$
- $1\text{OutOf4OT}(C; \text{not-gate}(k), K + 1)$ is the composition of $1\text{OutOf4OT}(C, K)$ with the protocol
 - $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K)$ for $m, n \leq N + 1$
 - $\text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K)$ for $m, n \leq N + 1$
 - $\text{OTOut}(n, m, K) := \text{read OTOut}(n, m, K)$ for $n, m \leq N + 1$
- $1\text{OutOf4OT}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $1\text{OutOf4OT}(C, K)$ with the protocol
 - $\text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K)$ for $n, m \leq N + 1$
 - $\text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K)$ for $m, n \leq N + 1$
 - $\text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K)$ for $m, n \leq N + 1$
 - $\text{OTOut}(n, m, K) := \text{read OTOut}(n, m, K)$ for $n, m \leq N + 1$
- $1\text{OutOf4OT}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $1\text{OutOf4OT}(C, K)$ with the protocol
 - $\begin{cases} \text{OTMsg}_0(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \\ \text{for } n, m \leq N + 1 \text{ if } n < m \end{cases}$
 - $\begin{cases} \text{OTMsg}_0(n, m, K) := \text{read OTMsg}_0(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{cases}$
 - $\begin{cases} \text{OTMsg}_1(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \\ \text{for } n, m \leq N + 1 \text{ if } n < m \end{cases}$
 - $\begin{cases} \text{OTMsg}_1(n, m, K) := \text{read OTMsg}_1(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{cases}$
 - $\begin{cases} \text{OTMsg}_2(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus y_n \\ \text{for } n, m \leq N + 1 \text{ if } n < m \end{cases}$
 - $\begin{cases} \text{OTMsg}_2(n, m, K) := \text{read OTMsg}_2(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{cases}$

$$\begin{aligned}
& - \begin{cases} \text{OTMsg}_3(n, m, K) := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \oplus y_n \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{OTMsg}_3(n, m, K) := \text{read OTMsg}_3(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{cases} \\
& - \begin{cases} \text{OTChc}_0(m, n, K) := \text{read Share}(n, k) \\ \text{for } m, n \leq N + 1 \text{ if } m < n \\ \text{OTChc}_0(m, n, K) := \text{read OTChc}_0(m, n, K) \\ \text{for } m, n \leq N + 1 \text{ if } m \geq n \end{cases} \\
& - \begin{cases} \text{OTChc}_1(m, n, K) := \text{read Share}(n, l) \\ \text{for } m, n \leq N + 1 \text{ if } m < n \\ \text{OTChc}_1(m, n, K) := \text{read OTChc}_1(m, n, K) \\ \text{for } m, n \leq N + 1 \text{ if } m \geq n \end{cases} \\
& - \text{OTOut}(n, m, K) := m_0 \leftarrow \text{OTMsg}_0(n, m, K); m_1 \leftarrow \text{OTMsg}_1(n, m, K); m_2 \leftarrow \text{OTMsg}_2(n, m, K); \\
& \quad m_3 \leftarrow \text{OTMsg}_3(n, m, K); c_0 \leftarrow \text{OTChc}_0(n, m, K); c_1 \leftarrow \text{OTChc}_1(n, m, K); \\
& \quad \text{if } c_0 \text{ then (if } c_1 \text{ then ret } m_3 \text{ else ret } m_2) \text{ else (if } c_1 \text{ then ret } m_1 \text{ else ret } m_0) \text{ for } n, m \leq N + 1
\end{aligned}$$

At this point, none of the channels defined by $\text{1OutOf4OT}(C, K)$ are utilized anywhere outside of $\text{1OutOf4OT}(C, K)$ and as such we may discard this protocol entirely. The inductive part of the real protocol therefore consists of the protocols $\text{Shares}(C, K)$ and $\text{Adv}(C, K)$, followed by the hiding of the channels

- $\text{SendBit}(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{RcvdBit}(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{Ctrb}(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{Ctrb-}\Sigma(n, m, k)$ for $n, m \leq N + 1$ 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

- $\text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k)$

reads from the divergent channel

- $\text{SendOutShare}(m, n, k) := \text{read SendOutShare}(m, n, k)$

and thus we may equivalently write the following:

- $\text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)}$

If party n is semi-honest and wire k is an output, then for any party m the channel

- $\text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k)$

reads from the channel

- $\text{SendOutShare}(m, n, k) := \text{read Share}(n, k)$

so we may substitute:

- $\text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, k)$

Summarizing the above, we get the following for channels $\text{SendOutShare}(-, -, -)_{\text{adv}}^{\text{party}(-)}$:

- $\begin{cases} \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, k) \\ \text{for } m, n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m, n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

If party n is semi-honest and wire k is not an output, then for any party m the channel

- $\text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(n, m, k)$

reads from the divergent channel

- $\text{SendOutShare}(n, m, k) := \text{read SendOutShare}(n, m, k)$

and thus we may again write the following:

- $\text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)}$

If party n is semi-honest and wire k is an output, then for any party m the channel

- $\text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(n, m, k)$

reads from the channel

- $\text{SendOutShare}(n, m, k) := \text{read Share}(m, k)$

so we may substitute:

- $\text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k)$

Summarizing the above, we get the following for channels $\text{RcvdOutShare}(-, -, -)_{\text{adv}}^{\text{party}(-)}$:

- $\begin{cases} \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(n, m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

If wire $k < K$ is not an output, then for any parties n, m the channel

- $\text{OutShare}(n, m, k) := \text{read SendOutShare}(n, m, k)$

reads from the divergent channel

- $\text{SendOutShare}(n, m, k) := \text{read SendOutShare}(n, m, k)$

and thus we may equivalently write the following:

- $\text{OutShare}(n, m, k) := \text{read OutShare}(n, m, k)$

If wire $k < K$ is an output, then for any parties n, m the channel

- $\text{OutShare}(n, m, k) := \text{read SendOutShare}(n, m, k)$

reads from the channel

- $\text{SendOutShare}(n, m, k) := \text{read Share}(m, k)$

so we may substitute:

- $\text{OutShare}(n, m, k) := \text{read Share}(m, k)$

Summarizing the above, we get the following for channels $\text{OutShare}(-, -, -)$:

- $\begin{cases} \text{OutShare}(n, m, k) := \text{read Share}(m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(n, m, k) := \text{read OutShare}(n, m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if wire } k \text{ not an output} \end{cases}$

At this point, the internal channels $\text{SendOutShare}(-, -, -)$ are unused and can be eliminated. The simplified version Fin of the final part of the real protocol is therefore as follows:

- $\begin{cases} \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, k) \\ \text{for } m, n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m, n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare}(n, m, k) := \text{read Share}(m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if wire } k \text{ an output} \\ \text{OutShare}(n, m, k) := \text{read OutShare}(n, m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{OutShare-}\Sigma(n, 0, k) := \text{read OutShare}(n, 0, k) \\ \text{for } n \leq N + 1 \text{ and } k < K \\ \text{OutShare-}\Sigma(n, m + 1, k) := x_{\Sigma} \leftarrow \text{OutShare-}\Sigma(n, m, k); x_{m+1} \leftarrow \text{OutShare}(n, m + 1, k); \text{ret } x_{\Sigma} \oplus x_{m+1} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \end{cases}$
- $\begin{cases} \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$
- $\text{Out}(n, k) := \text{read OutShare-}\Sigma(n, N + 1, k) \text{ for } n \leq N + 1 \text{ and } k < K$
- $\begin{cases} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k) \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

This is followed by the hiding of the channels

- $\text{OutShare}(n, m, k)$ for $n, m \leq N + 1$ and $k < K$,
- $\text{OutShare-}\Sigma(n, m, k)$ for $n, m \leq N + 1$ 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 $\bigoplus_{i \leq N+1} x_i$ 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 $x \oplus (\bigoplus_{i \leq N} x_i)$ of x and the shares x_0, \dots, x_N . To this end, we introduce new internal channels

$$\bullet \begin{cases} \text{Share-}\Sigma(0, k) := \text{read Share}(0, k) \\ \text{for } k < K \\ \text{Share-}\Sigma(m + 1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); x_{m+1} \leftarrow \text{Share}(m + 1, k); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \text{ and } k < K \end{cases}$$

that inductively compute the the sum of shares for parties $0, \dots, N$ on each wire $k < K$. 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

$$\bullet \text{InShare-}\checkmark(m, n, i) := _ \leftarrow \text{read InShare}(m, n, i); \text{ret } \checkmark \text{ for } m, n \leq N + 1 \text{ and } i < I_n$$

for the timing of input shares in the initial part of the protocol and

$$\bullet \text{Share-}\checkmark(n, k) := _ \leftarrow \text{read Share}(n, k); \text{ret } \checkmark \text{ for } n \leq N + 1 \text{ and } k < K$$

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 $\text{Adv}(C, K)$ in the case of an *and* gate. For any party n with $n \leq N$ we can write the channels

$$\begin{aligned} \bullet \text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} &:= x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ret } \checkmark \text{ for } n \leq N \\ \bullet \text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} &:= y_{N+1} \leftarrow \text{Share}(N + 1, l); \text{ret } \checkmark \text{ for } n \leq N \end{aligned}$$

equivalently as follows:

$$\begin{aligned} \bullet \text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} &:= \text{read Share-}\checkmark(N + 1, k) \text{ for } n \leq N \\ \bullet \text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} &:= \text{read Share-}\checkmark(N + 1, l) \text{ for } n \leq N \end{aligned}$$

We thus have the following for channels $\text{OTChcRcvd}_0(-, N + 1, K)_{\text{adv}}^{\text{ot}}$ and $\text{OTChcRcvd}_1(-, N + 1, K)_{\text{adv}}^{\text{ot}}$:

$$\bullet \begin{cases} \text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(N + 1, k) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \\ \text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(N + 1, l) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{cases}$$

So the inductive part of the real protocol now has $\text{Adv}(C, K)$ looking like so:

$$\bullet \text{Adv}(\epsilon, 0) \text{ is the protocol } 0$$

- $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Adv}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Adv}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\begin{cases} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\begin{cases} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$

$$\begin{aligned}
- & \begin{cases} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases} \\
- & \begin{cases} \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases} \\
- & \begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases} \\
- & \begin{cases} \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus y_n \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \oplus y_n \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases}
\end{aligned}$$

$$\begin{aligned}
- & \begin{cases} \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, k) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest and } n < m \\ \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, l) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest and } n < m \\ \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest or } n \geq m \end{cases} \\
- & \begin{cases} \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := x_m \leftarrow \text{Share}(m, k); \text{ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ semi-honest or } n \geq m \\ \text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(N + 1, k) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{cases} \\
- & \begin{cases} \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := y_m \leftarrow \text{Share}(m, l); \text{ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ semi-honest or } n \geq m \\ \text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(N + 1, l) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{cases} \\
- & \begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read RcvdBit}(m, n, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}
\end{aligned}$$

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 $0, \dots, N$. To this end, we introduce new internal channels

- $\text{InShare-}\$-\checkmark(m, n, i) := _ \leftarrow \text{read InShare-}\$(m, n, i); \text{ret } \checkmark$ for $m, n \leq N + 1$ and $i < I_n$
- $\text{InShare-}\$-\Sigma-\checkmark(m, n, i) := _ \leftarrow \text{read InShare-}\$-\Sigma(m, n, i); \text{ret } \checkmark$ for $m \leq N$ and $n \leq N + 1$ and $i < I_n$

to keep track of the timing information in the protocol Init and

- $\text{SendBit-}\checkmark(n, m, k) := _ \leftarrow \text{SendBit}(n, m, k); \text{ret } \checkmark$ for $n, m \leq N + 1$ and $k < K$
- $\text{RcvdBit-}\checkmark(n, m, k) := _ \leftarrow \text{RcvdBit}(n, m, k); \text{ret } \checkmark$ for $n, m \leq N + 1$ and $k < K$
- $\text{Ctrb-}\checkmark(n, m, k) := _ \leftarrow \text{Ctrb}(n, m, k); \text{ret } \checkmark$ for $n, m \leq N + 1$ and $k < K$
- $\text{Ctrb-}\Sigma-\checkmark(n, m, k) := _ \leftarrow \text{Ctrb-}\Sigma(n, m, k); \text{ret } \checkmark$ for $n, m \leq N + 1$ and $k < K$

to keep track of the timing information in the protocol $\text{Shares}(C, K)$. Call the following protocol fragment $\text{Init-}\checkmark$:

- $\text{InShare-}\checkmark(m, n, i) := _ \leftarrow \text{read InShare-}\$(m, n, i); \text{ ret } \checkmark \text{ for } m, n \leq N + 1 \text{ and } i < I_n$
- $\text{InShare-}\Sigma\checkmark(m, n, i) := _ \leftarrow \text{read InShare-}\Sigma\$(m, n, i); \text{ ret } \checkmark \text{ for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n$
- $\text{InShare-}\checkmark(m, n, i) := _ \leftarrow \text{read InShare}(m, n, i); \text{ ret } \checkmark \text{ for } m, n \leq N + 1 \text{ and } i < I_n$

Call the following protocol fragment $\text{Shares-}\checkmark(C, K)$:

- $\text{SendBit-}\checkmark(n, m, k) := _ \leftarrow \text{SendBit}(n, m, k); \text{ ret } \checkmark \text{ for } n, m \leq N + 1 \text{ and } k < K$
- $\text{RcvdBit-}\checkmark(n, m, k) := _ \leftarrow \text{RcvdBit}(n, m, k); \text{ ret } \checkmark \text{ for } n, m \leq N + 1 \text{ and } k < K$
- $\text{Ctrb-}\checkmark(n, m, k) := _ \leftarrow \text{Ctrb}(n, m, k); \text{ ret } \checkmark \text{ for } n, m \leq N + 1 \text{ and } k < K$
- $\text{Ctrb-}\Sigma\checkmark(n, m, k) := _ \leftarrow \text{Ctrb-}\Sigma(n, m, k); \text{ ret } \checkmark \text{ for } n, m \leq N + 1 \text{ and } k < K$
- $\text{Share-}\checkmark(n, k) := _ \leftarrow \text{read Share}(n, k); \text{ ret } \checkmark \text{ for } n \leq N + 1 \text{ and } k < K$

In the presence of the channels $\text{InShare}(-, -, -)$ as well as the protocol $\text{Shares-}\checkmark(C, K)$ we can express the protocol $\text{Shares}(C, K)$ equivalently as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \text{ for } n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K) \text{ for } n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K) \text{ for } n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K) \text{ for } n, m \leq N + 1$
 - $\text{Share}(n, K) := \text{read InShare}(n, p, i) \text{ for } n \leq N$
 - $\text{Share}(N + 1, K) := _ \leftarrow \text{InShare-}\checkmark(N + 1, p, i); \text{ read InShare}(N + 1, p, i)$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \text{ for } n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K) \text{ for } n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K) \text{ for } n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K) \text{ for } n, m \leq N + 1$
 - $\text{Share}(n, K) := \text{read Share}(n, k) \text{ for } n \leq N$
 - $\text{Share}(N + 1, K) := _ \leftarrow \text{Share-}\checkmark(N + 1, k); x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ ret } \neg x_{N+1}$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \text{ for } n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K) \text{ for } n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K) \text{ for } n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K) \text{ for } n, m \leq N + 1$
 - $\text{Share}(n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ ret } x_n \oplus y_n \text{ for } n \leq N$
 - $\text{Share}(N + 1, K) := _ \leftarrow \text{Share-}\checkmark(N + 1, k); _ \leftarrow \text{Share-}\checkmark(N + 1, l); x_{N+1} \leftarrow \text{Share}(N + 1, k); y_{N+1} \leftarrow \text{Share}(N + 1, l); \text{ ret } x_{N+1} \oplus y_{N+1}$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol

- $\begin{cases} \text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ samp flip} \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{cases}$
- $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l);$
 $x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ ret } b \oplus (x_m * y_n) \oplus (x_n * y_m) \text{ for } n, m \leq N + 1$
- $\begin{cases} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ ret } x_n * y_n \\ \text{for } n \leq N + 1 \end{cases}$
- $\begin{cases} \text{Ctrb-}\Sigma(n, 0, K) := \text{read Ctrb}(n, 0, K) \\ \text{for } n \leq N + 1 \\ \text{Ctrb-}\Sigma(n, m + 1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(n, m, K); b \leftarrow \text{Ctrb}(n, m + 1, K); \text{ ret } b_\Sigma \oplus b \\ \text{for } n \leq N + 1 \text{ and } m \leq N \end{cases}$
- $\text{Share}(n, K) := \text{read Ctrb-}\Sigma(n, N + 1, K) \text{ for } n \leq N$
- $\text{Share}(N + 1, K) := _ \leftarrow \text{Ctrb-}\Sigma\text{-}\checkmark(N + 1, N + 1, K); \text{ read Ctrb-}\Sigma(N + 1, N + 1, K)$

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

- $\text{In-}\checkmark(n, i) := _ \leftarrow \text{read In}(n, i); \text{ ret } \checkmark \text{ for } n \leq N + 1 \text{ and } i < I_n$

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 $\text{Wires-}\checkmark(C, K)$:

- $\text{Wires-}\checkmark(\epsilon, 0)$ is the protocol 0
- $\text{Wires-}\checkmark(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := \text{read In-}\checkmark(p, i)$
- $\text{Wires-}\checkmark(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := _ \leftarrow \text{Wire-}\checkmark(k); \text{ ret } \checkmark$
- $\text{Wires-}\checkmark(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ ret } \checkmark$
- $\text{Wires-}\checkmark(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ ret } \checkmark$

Our goal now is to show that the timing channels can be equivalently characterized as follows:

- $\text{InShare-}\checkmark(m, n, i) := \text{read In-}\checkmark(n, i) \text{ for } m, n \leq N + 1 \text{ and } i < I_n$
- $\text{Share-}\checkmark(n, k) := \text{read Wire-}\checkmark(k) \text{ for } n \leq N + 1 \text{ and } k < K$

In the presence of the channels $\text{InShare-}\checkmark(-, -, -)$ as well as the protocol $\text{Shares}(C, K)$ we can define the protocol $\text{Shares-}\checkmark(C, K)$ equivalently as follows:

- $\text{Shares-}\checkmark(\epsilon, 0)$ is the protocol 0
- $\text{Shares-}\checkmark(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share-}\checkmark(n, K) := \text{read InShare-}\checkmark(n, p, i)$ for $n \leq N + 1$
- $\text{Shares-}\checkmark(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share-}\checkmark(n, K) := \text{read Share-}\checkmark(n, k)$ for $n \leq N$
 - $\text{Share-}\checkmark(N + 1, K) := _ \leftarrow \text{Share-}\checkmark(N + 1, k); \text{ret } \checkmark$
- $\text{Shares-}\checkmark(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share-}\checkmark(n, K) := _ \leftarrow \text{Share-}\checkmark(n, k); _ \leftarrow \text{Share-}\checkmark(n, l); \text{ret } \checkmark$ for $n \leq N + 1$
- $\text{Shares-}\checkmark(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\left\{ \begin{array}{l} \text{SendBit-}\checkmark(n, m, K) := _ \leftarrow \text{Share-}\checkmark(n, k); _ \leftarrow \text{Share-}\checkmark(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n < m \end{array} \right.$
 - $\left\{ \begin{array}{l} \text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{array} \right.$
 - $\text{RcvdBit-}\checkmark(n, m, K) := _ \leftarrow \text{SendBit-}\checkmark(m, n, K); _ \leftarrow \text{Share-}\checkmark(m, k); _ \leftarrow \text{Share-}\checkmark(m, l);$
 $_ \leftarrow \text{Share-}\checkmark(n, k); _ \leftarrow \text{Share-}\checkmark(n, l); \text{ret } \checkmark$ for $n, m \leq N + 1$
 - $\left\{ \begin{array}{l} \text{Ctrb-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{Ctrb-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{Ctrb-}\checkmark(n, n, K) := _ \leftarrow \text{Share-}\checkmark(n, k); _ \leftarrow \text{Share-}\checkmark(n, l); \text{ret } \checkmark \\ \text{for } n \leq N + 1 \end{array} \right.$
 - $\left\{ \begin{array}{l} \text{Ctrb-}\Sigma\text{-}\checkmark(n, 0, K) := \text{read Ctrb-}\checkmark(n, 0, K) \\ \text{for } n \leq N + 1 \\ \text{Ctrb-}\Sigma\text{-}\checkmark(n, m + 1, K) := _ \leftarrow \text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K); _ \leftarrow \text{Ctrb-}\checkmark(n, m + 1, K); \text{ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } m \leq N \end{array} \right.$
 - $\text{Share-}\checkmark(n, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, N + 1, K)$ for $n \leq N + 1$

We also observe that in the presence of the protocol Init we can define the protocol $\text{Init-}\checkmark$ equivalently as follows:

- $\text{InShare-}\$-\checkmark(m, n, i) := _ \leftarrow \text{In-}\checkmark(n, i); \text{ ret } \checkmark$ for $m \leq N$ and $n \leq N + 1$ and $i < I_n$
- $\begin{cases} \text{InShare-}\$-\Sigma-\checkmark(0, n, i) := \text{read InShare-}\$-\checkmark(0, n, i) \\ \text{for } n \leq N + 1 \text{ and } i < I_n \\ \text{InShare-}\$-\Sigma-\checkmark(m + 1, n, i) := _ \leftarrow \text{InShare-}\$-\Sigma-\checkmark(m, n, i); _ \leftarrow \text{InShare-}\$-\checkmark(m + 1, n, i); \text{ ret } \checkmark \\ \text{for } m < N \text{ and } n \leq N + 1 \text{ and } i < I_n \end{cases}$
- $\text{InShare-}\$-\checkmark(N + 1, n, i) := _ \leftarrow \text{InShare-}\$-\Sigma-\checkmark(N, n, i); _ \leftarrow \text{In-}\checkmark(n, i); \text{ ret } \checkmark$ for $n \leq N + 1$ and $i < I_n$
- $\text{InShare-}\checkmark(m, n, i) := \text{read InShare-}\$-\checkmark(m, n, i)$ for $m, n \leq N + 1$ and $i < I_n$

Furthermore, in the presence of the channels $\text{In-}\checkmark(-, -)$ we can express the protocol $\text{Init-}\checkmark$ equivalently as follows:

- $\text{InShare-}\$-\checkmark(m, n, i) := \text{read In-}\checkmark(n, i)$ for $m, n \leq N + 1$ and $i < I_n$
- $\text{InShare-}\$-\Sigma-\checkmark(m, n, i) := \text{read In-}\checkmark(n, i)$ for $m \leq N$ and $n \leq N + 1$ and $i < I_n$
- $\text{InShare-}\checkmark(m, n, i) := \text{read In-}\checkmark(n, i)$ for $m, n \leq N + 1$ and $i < I_n$

Lastly, in the presence of the channels $\text{InShare-}\checkmark(-, -, -)$ as well as the protocol $\text{Wires-}\checkmark(C, K)$ we can express the protocol $\text{Shares-}\checkmark(C, K)$ equivalently as the channels

- $\text{Share-}\checkmark(n, k) := \text{read Wire-}\checkmark(k)$ for $n \leq N + 1$ and $k < K$

together with the following protocol $\text{Ctrbs-}\checkmark(C, K)$:

- $\text{Ctrbs-}\checkmark(\epsilon, 0)$ is the protocol 0
- $\text{Ctrbs-}\checkmark(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Ctrbs-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma-\checkmark(n, m, K) := \text{read Ctrb-}\Sigma-\checkmark(n, m, K)$ for $n, m \leq N + 1$
- $\text{Ctrbs-}\checkmark(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Ctrbs-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma-\checkmark(n, m, K) := \text{read Ctrb-}\Sigma-\checkmark(n, m, K)$ for $n, m \leq N + 1$
- $\text{Ctrbs-}\checkmark(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Ctrbs-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma-\checkmark(n, m, K) := \text{read Ctrb-}\Sigma-\checkmark(n, m, K)$ for $n, m \leq N + 1$
- $\text{Ctrbs-}\checkmark(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Ctrbs-}\checkmark(C, K)$ with the protocol
 - $\begin{cases} \text{SendBit-}\checkmark(n, m, K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{cases}$

- $\begin{cases} \text{RcvdBit-}\checkmark(n, m, K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \geq n \end{cases}$
- $\text{Ctrb-}\checkmark(n, m, K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark \text{ for } n, m \leq N + 1$
- $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark \text{ for } n, m \leq N + 1$

We now revisit the case of an *and* gate in the protocol $\text{Adv}(C, K)$. For any party n with $n \leq N$ we can write the channels

- $\text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(N + 1, k)$
- $\text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\checkmark(N + 1, l)$

equivalently as follows:

- $\text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(k)$
- $\text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(l)$

We thus have the following for channels $\text{OTChcRcvd}_0(-, N + 1, K)_{\text{adv}}^{\text{ot}}$ and $\text{OTChcRcvd}_1(-, N + 1, K)_{\text{adv}}^{\text{ot}}$:

- $\begin{cases} \text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(k) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(l) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{cases}$

So the inductive part of the real protocol now has $\text{Adv}(C, K)$ looking like so:

- $\text{Adv}(\epsilon, 0)$ is the protocol 0
- $\text{Adv}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N + 1$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$
 - $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N + 1$

- $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Adv}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Adv}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(m, n, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(m, n, K)_{\text{adv}}^{\text{party}(n)}$ for $m, n \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$

- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$

• $\text{Adv}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol

- $\begin{cases} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \end{cases}$
- $\begin{cases} \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$

$\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n$
 for $n, m \leq N + 1$ if n semi-honest and $n < m$
 $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n honest or $n \geq m$

$\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus y_n$
 for $n, m \leq N + 1$ if n semi-honest and $n < m$
 $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n honest or $n \geq m$

$\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \oplus y_n$
 for $n, m \leq N + 1$ if n semi-honest and $n < m$
 $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n honest or $n \geq m$

$\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark$
 for $n, m \leq N + 1$ if n honest and $n < m$
 $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n semi-honest or $n \geq m$

$\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark$
 for $n, m \leq N + 1$ if n honest and $n < m$
 $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n semi-honest or $n \geq m$

$\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark$
 for $n, m \leq N + 1$ if n honest and $n < m$
 $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n semi-honest or $n \geq m$

$\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark$
 for $n, m \leq N + 1$ if n honest and $n < m$
 $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if n semi-honest or $n \geq m$

$\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, k)$
 for $n, m \leq N + 1$ if m semi-honest and $n < m$
 $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if m honest or $n \geq m$

$\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, l)$
 for $n, m \leq N + 1$ if m semi-honest and $n < m$
 $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n, m \leq N + 1$ if m honest or $n \geq m$

$\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := x_m \leftarrow \text{Share}(m, k); \text{ret } \checkmark$
 for $n \leq N + 1$ and $m \leq N$ if m honest and $n < m$
 $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$
 for $n \leq N + 1$ and $m \leq N$ if m semi-honest or $n \geq m$

$\text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(k)$
 for $n \leq N$
 $\text{OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}}$

$$\begin{aligned}
& - \begin{cases} \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := y_m \leftarrow \text{Share}(m, l); \text{ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ semi-honest or } n \geq m \end{cases} \\
& - \begin{cases} \text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(l) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{cases} \\
& - \begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read RcvdBit}(m, n, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}
\end{aligned}$$

Finally, in the presence of the channels $\text{InShare-}\checkmark(-, -, -)$, $\text{Share-}\checkmark(-, -)$ and the protocols $\text{Shares-}\checkmark$ and $\text{Wires-}\checkmark$ we can express the protocol $\text{Shares}(C, K)$ equivalently as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := \text{read InShare}(n, p, i)$ for $n \leq N$
 - $\text{Share}(N + 1, K) := _ \leftarrow \text{Wire-}\checkmark(K); \text{read InShare}(N + 1, p, i)$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := \text{read Share}(n, k)$ for $n \leq N$
 - $\text{Share}(N + 1, K) := _ \leftarrow \text{Wire-}\checkmark(K); x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ret } \neg x_{N+1}$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n \oplus y_n$ for $n \leq N$
 - $\text{Share}(N + 1, K) := _ \leftarrow \text{Wire-}\checkmark(K); x_{N+1} \leftarrow \text{Share}(N + 1, k); y_{N+1} \leftarrow \text{Share}(N + 1, l); \text{ret } x_{N+1} \oplus y_{N+1}$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{samp flip}$ for $n, m \leq N + 1$ if $n < m$
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$ if $n \geq m$

- $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l);$
 $x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus (x_m * y_n) \oplus (x_n * y_m) \text{ for } n, m \leq N + 1$
- $\begin{cases} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n * y_n \\ \text{for } n \leq N + 1 \end{cases}$
- $\begin{cases} \text{Ctrb-}\Sigma(n, 0, K) := \text{read Ctrb}(n, 0, K) \\ \text{for } n \leq N + 1 \\ \text{Ctrb-}\Sigma(n, m + 1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(n, m, K); b \leftarrow \text{Ctrb}(n, m + 1, K); \text{ret } b_\Sigma \oplus b \\ \text{for } n \leq N + 1 \text{ and } m \leq N \end{cases}$
- $\text{Share}(n, K) := \text{read Ctrb-}\Sigma(n, N + 1, K) \text{ for } n \leq N$
- $\text{Share}(N + 1, K) := _ \leftarrow \text{Wire-}\checkmark(K); \text{read Ctrb-}\Sigma(N + 1, N + 1, K)$

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

- $\begin{cases} \text{Share-}\Sigma(0, k) := \text{read Share}(0, k) \\ \text{for } k < K \\ \text{Share-}\Sigma(m + 1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); x_{m+1} \leftarrow \text{Share}(m + 1, k); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \text{ and } k < K \end{cases}$

that inductively compute the the sum of shares for parties $0, \dots, N$ on each wire $k < K$.

We first consider the protocol $\text{Adv}(C, K)$ in the case of an *and* gate. For any party n with $n \leq N$ we want to write the channels

- $\text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ret } \checkmark \text{ for } n \leq N$
- $\text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := y_{N+1} \leftarrow \text{Share}(N + 1, l); \text{ret } \checkmark \text{ for } n \leq N$

in the form below:

- $\text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); \text{ret } \checkmark \text{ for } n \leq N$
- $\text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := y_\Sigma \leftarrow \text{Share-}\Sigma(N, l); \text{ret } \checkmark \text{ for } n \leq N$

We thus have the following for channels $\text{OTChcRcvd}_0(-, N + 1, K)_{\text{adv}}^{\text{ot}}$ and $\text{OTChcRcvd}_1(-, N + 1, K)_{\text{adv}}^{\text{ot}}$:

- $\begin{cases} \text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); \text{ret } \checkmark \\ \text{for } n \leq N \\ \text{OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := y_\Sigma \leftarrow \text{Share-}\Sigma(N, l); \text{ret } \checkmark \\ \text{for } n \leq N \\ \text{OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{cases}$

So the inductive part of the real protocol now has $\text{Adv}(C, K)$ looking like so:

- $\text{Adv}(\epsilon, 0)$ is the protocol 0
- $\text{Adv}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol

- [illegible]

- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Adv}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Adv}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\begin{cases} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\begin{cases} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\begin{cases} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$

[illegible]

$$\begin{aligned}
& - \begin{cases} \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, k) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest and } n < m \\ \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest or } n \geq m \end{cases} \\
& - \begin{cases} \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, l) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest and } n < m \\ \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest or } n \geq m \end{cases} \\
& - \begin{cases} \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := x_m \leftarrow \text{Share}(m, k); \text{ ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ semi-honest or } n \geq m \end{cases} \\
& - \begin{cases} \text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); \text{ ret } \checkmark \\ \text{for } n \leq N \\ \text{OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{cases} \\
& - \begin{cases} \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := y_m \leftarrow \text{Share}(m, l); \text{ ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ semi-honest or } n \geq m \end{cases} \\
& - \begin{cases} \text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := y_\Sigma \leftarrow \text{Share-}\Sigma(N, l); \text{ ret } \checkmark \\ \text{for } n \leq N \\ \text{OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{cases} \\
& - \begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read RcvdBit}(m, n, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}
\end{aligned}$$

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 $\text{Shares}(C, K)$ as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := \text{read InShare}(n, p, i)$ for $n \leq N$
 - $\text{Share}(N + 1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); \text{ read InShare}(N + 1, p, i)$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$

- $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
- $\text{Share}(n, K) := \text{read Share}(n, k)$
- $\text{Share}(N + 1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ret } \neg x_{N+1}$
- $\text{Shares}(C; \text{ xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n \oplus y_n$ for $n \leq N$
 - $\text{Share}(N + 1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Share}(N + 1, k); y_{N+1} \leftarrow \text{Share}(N + 1, l); \text{ret } x_{N+1} \oplus y_{N+1}$
- $\text{Shares}(C; \text{ and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\begin{cases} \text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{samp flip} \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{cases}$
 - $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n, m \leq N + 1$
 - $\begin{cases} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n * y_n \\ \text{for } n \leq N + 1 \end{cases}$
 - $\begin{cases} \text{Ctrb-}\Sigma(n, 0, K) := \text{read Ctrb}(n, 0, K) \\ \text{for } n \leq N + 1 \\ \text{Ctrb-}\Sigma(n, m + 1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(n, m, K); b \leftarrow \text{Ctrb}(n, m + 1, K); \text{ret } b_\Sigma \oplus b \\ \text{for } n \leq N + 1 \text{ and } m \leq N \end{cases}$
 - $\text{Share}(n, K) := \text{read Ctrb-}\Sigma(n, N + 1, K)$ for $n \leq N$
 - $\text{Share}(N + 1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); \text{read Ctrb-}\Sigma(N + 1, N + 1, K)$

It is not at all clear that the aforementioned amendments of $\text{Adv}(C, K)$ and $\text{Shares}(C, K)$ are sound. In the rest of this section we justify their soundness. We start by introducing the channels

- $\text{InShare-}\checkmark(m, n, i) := _ \leftarrow \text{read InShare}(m, n, i); \text{ret } \checkmark$ for $m, n \leq N + 1$ and $i < I_n$

for the timing of input shares in the initial part of the protocol and

- $\text{Share-}\checkmark(n, k) := _ \leftarrow \text{read Share}(n, k); \text{ret } \checkmark$ for $n \leq N + 1$ and $k < K$

for the timing of shares in the inductive part of the protocol. In addition, we introduce the channels

- $\text{Share-}\Sigma\text{-}\checkmark(m, k) := _ \leftarrow \text{read Share-}\Sigma(m, k); \text{ret } \checkmark$ for $m \leq N$ and $k < K$

for the timing of the sum of shares.

Take the protocol $\text{Adv}(C, K)$ in the case of an *and* gate. For any party n with $n \leq N$ we can write the channels

- $\text{OTChcRcvd}_0(n, N+1, K)_{\text{adv}}^{\text{ot}} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); \text{ ret } \checkmark \text{ for } n \leq N$
- $\text{OTChcRcvd}_1(n, N+1, K)_{\text{adv}}^{\text{ot}} := y_\Sigma \leftarrow \text{Share-}\Sigma(N, l); \text{ ret } \checkmark \text{ for } n \leq N$

equivalently as follows:

- $\text{OTChcRcvd}_0(n, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\Sigma\text{-}\checkmark(N, k) \text{ for } n \leq N$
- $\text{OTChcRcvd}_1(n, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\Sigma\text{-}\checkmark(N, l) \text{ for } n \leq N$

We thus have the following for channels $\text{OTChcRcvd}_0(-, N+1, K)_{\text{adv}}^{\text{ot}}$ and $\text{OTChcRcvd}_1(-, N+1, K)_{\text{adv}}^{\text{ot}}$:

- $\begin{cases} \text{OTChcRcvd}_0(n, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\Sigma\text{-}\checkmark(N, k) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_0(N+1, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(N+1, N+1, K)_{\text{adv}}^{\text{ot}} \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_1(n, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\Sigma\text{-}\checkmark(N, l) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_1(N+1, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(N+1, N+1, K)_{\text{adv}}^{\text{ot}} \end{cases}$

So the inductive part of the real protocol now has $\text{Adv}(C, K)$ looking like so:

- $\text{Adv}(\epsilon, 0)$ is the protocol 0
- $\text{Adv}(C; \text{input-gate}(p, i), K+1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N+1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N+1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N+1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \text{ for } n, m \leq N+1$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N+1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ if } n \text{ honest} \end{cases}$
 - $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
 - $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
 - $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
 - $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
 - $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
 - $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
 - $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
 - $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
 - $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
 - $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
 - $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
 - $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
 - $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \text{ for } n, m \leq N+1$
- $\text{Adv}(C; \text{not-gate}(k), K+1)$ is the composition of $\text{Adv}(C, K)$ with the protocol

- SendBit(n, m, K) $_{adv}^{party(n)} := \text{read SendBit}(n, m, K)_{adv}^{party(n)}$ for $n, m \leq N + 1$
 - RcvdBit(n, m, K) $_{adv}^{party(n)} := \text{read RcvdBit}(n, m, K)_{adv}^{party(n)}$ for $n, m \leq N + 1$
 - Ctrb(n, m, K) $_{adv}^{party(n)} := \text{read Ctrb}(n, m, K)_{adv}^{party(n)}$ for $n, m \leq N + 1$
 - Ctrb- Σ (n, m, K) $_{adv}^{party(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{adv}^{party(n)}$ for $n, m \leq N + 1$
 - $\begin{cases} \text{Share}(n, K)_{adv}^{party(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{adv}^{party(n)} := \text{read Share}(n, K)_{adv}^{party(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - OTMsg $_0$ (n, m, K) $_{adv}^{ot} := \text{read OTMsg}_0(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTMsg $_1$ (n, m, K) $_{adv}^{ot} := \text{read OTMsg}_1(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTMsg $_2$ (n, m, K) $_{adv}^{ot} := \text{read OTMsg}_2(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTMsg $_3$ (n, m, K) $_{adv}^{ot} := \text{read OTMsg}_3(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTMsgRcvd $_0$ (n, m, K) $_{adv}^{ot} := \text{read OTMsgRcvd}_0(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTMsgRcvd $_1$ (n, m, K) $_{adv}^{ot} := \text{read OTMsgRcvd}_1(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTMsgRcvd $_2$ (n, m, K) $_{adv}^{ot} := \text{read OTMsgRcvd}_2(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTMsgRcvd $_3$ (n, m, K) $_{adv}^{ot} := \text{read OTMsgRcvd}_3(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTChc $_0$ (n, m, K) $_{adv}^{ot} := \text{read OTChc}_0(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTChc $_1$ (n, m, K) $_{adv}^{ot} := \text{read OTChc}_1(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTChcRcvd $_0$ (n, m, K) $_{adv}^{ot} := \text{read OTChcRcvd}_0(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTChcRcvd $_1$ (n, m, K) $_{adv}^{ot} := \text{read OTChcRcvd}_1(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTOut(n, m, K) $_{adv}^{ot} := \text{read OTOut}(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - Adv($C; xor\text{-gate}(k, l), K + 1$) is the composition of Adv(C, K) with the protocol
 - SendBit(n, m, K) $_{adv}^{party(n)} := \text{read SendBit}(n, m, K)_{adv}^{party(n)}$ for $n, m \leq N + 1$
 - RcvdBit(n, m, K) $_{adv}^{party(n)} := \text{read RcvdBit}(n, m, K)_{adv}^{party(n)}$ for $n, m \leq N + 1$
 - Ctrb(n, m, K) $_{adv}^{party(n)} := \text{read Ctrb}(n, m, K)_{adv}^{party(n)}$ for $n, m \leq N + 1$
 - Ctrb- Σ (n, m, K) $_{adv}^{party(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{adv}^{party(n)}$ for $n, m \leq N + 1$
 - $\begin{cases} \text{Share}(n, K)_{adv}^{party(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{adv}^{party(n)} := \text{read Share}(n, K)_{adv}^{party(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - OTMsg $_0$ (n, m, K) $_{adv}^{ot} := \text{read OTMsg}_0(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTMsg $_1$ (n, m, K) $_{adv}^{ot} := \text{read OTMsg}_1(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTMsg $_2$ (n, m, K) $_{adv}^{ot} := \text{read OTMsg}_2(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTMsg $_3$ (n, m, K) $_{adv}^{ot} := \text{read OTMsg}_3(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTMsgRcvd $_0$ (n, m, K) $_{adv}^{ot} := \text{read OTMsgRcvd}_0(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTMsgRcvd $_1$ (n, m, K) $_{adv}^{ot} := \text{read OTMsgRcvd}_1(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTMsgRcvd $_2$ (n, m, K) $_{adv}^{ot} := \text{read OTMsgRcvd}_2(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$
 - OTMsgRcvd $_3$ (n, m, K) $_{adv}^{ot} := \text{read OTMsgRcvd}_3(n, m, K)_{adv}^{ot}$ for $n, m \leq N + 1$

- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$

- $\text{Adv}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol

- $\begin{cases} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$
- $\begin{cases} \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$
- $\begin{cases} \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus y_n \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$

$$\begin{aligned}
& \left\{ \begin{array}{l} \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \oplus y_n \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, k) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest and } n < m \\ \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest or } n \geq m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, l) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest and } n < m \\ \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest or } n \geq m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := x_m \leftarrow \text{Share}(m, k); \text{ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ semi-honest or } n \geq m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\Sigma\text{-}\checkmark(N, k) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := y_m \leftarrow \text{Share}(m, l); \text{ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ semi-honest or } n \geq m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\Sigma\text{-}\checkmark(N, l) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{array} \right.
\end{aligned}$$

$$- \begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read RcvdBit}(m, n, K) \\ \quad \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \\ \quad \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}$$

We now amend the shares of party $N + 1$ further: in the presence of the channels $\text{Share-}\Sigma\text{-}\checkmark(-, -)$ 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 $\text{Shares}(C, K)$ equivalently as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := \text{read InShare}(n, p, i)$ for $n \leq N$
 - $\text{Share}(N + 1, K) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(N, K); \text{read InShare}(N + 1, p, i)$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := \text{read Share}(n, k)$ for $n \leq N$
 - $\text{Share}(N + 1, K) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(N, K); x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ret } \neg x_{N+1}$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n \oplus y_n$ for $n \leq N$
 - $\text{Share}(N + 1, K) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(N, K); x_{N+1} \leftarrow \text{Share}(N + 1, k); y_{N+1} \leftarrow \text{Share}(N + 1, l); \text{ret } x_{N+1} \oplus y_{N+1}$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{samp flip}$ for $n, m \leq N + 1$ if $n < m$
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$ if $n \geq m$
 - $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n, m \leq N + 1$

- $\begin{cases} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n * y_n \\ \text{for } n \leq N + 1 \end{cases}$
- $\begin{cases} \text{Ctrb-}\Sigma(n, 0, K) := \text{read Ctrb}(n, 0, K) \\ \text{for } n \leq N + 1 \\ \text{Ctrb-}\Sigma(n, m + 1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(n, m, K); b \leftarrow \text{Ctrb}(n, m + 1, K); \text{ret } b_\Sigma \oplus b \\ \text{for } n \leq N + 1 \text{ and } m \leq N \end{cases}$
- $\text{Share}(n, K) := \text{read Ctrb-}\Sigma(n, N + 1, K) \text{ for } n \leq N$
- $\text{Share}(N + 1, K) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(N, K); \text{read Ctrb-}\Sigma(N + 1, N + 1, K)$

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

- $\text{In-}\checkmark(n, i) := _ \leftarrow \text{read In}(n, i); \text{ret } \checkmark \text{ for } n \leq N + 1 \text{ and } i < I_n$

to keep track of whether an input has arrived, and a new protocol $\text{Wires-}\checkmark(C, K)$ that keeps track of the timing of wire shares:

- $\text{Wires-}\checkmark(\epsilon, 0)$ is the protocol 0
- $\text{Wires-}\checkmark(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := \text{read In-}\checkmark(p, i)$
- $\text{Wires-}\checkmark(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := _ \leftarrow \text{Wire-}\checkmark(k); \text{ret } \checkmark$
- $\text{Wires-}\checkmark(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark$
- $\text{Wires-}\checkmark(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Wires-}\checkmark(C, K)$ with the protocol
 - $\text{Wire-}\checkmark(K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark$

Our goal is again to show that the timing channels can be equivalently characterized as follows:

- $\text{InShare-}\checkmark(m, n, i) := \text{read In-}\checkmark(n, i) \text{ for } m, n \leq N + 1 \text{ and } i < I_n$
- $\text{Share-}\checkmark(n, k) := \text{read Wire-}\checkmark(k) \text{ for } n \leq N + 1 \text{ and } k < K$

To this end, we introduce new internal channels

- $\text{InShare-}\$ \text{-}\checkmark(m, n, i) := _ \leftarrow \text{read InShare-}\$(m, n, i); \text{ret } \checkmark \text{ for } m, n \leq N + 1 \text{ and } i < I_n$
- $\text{InShare-}\$ \text{-}\Sigma\text{-}\checkmark(m, n, i) := _ \leftarrow \text{read InShare-}\$ \text{-}\Sigma(m, n, i); \text{ret } \checkmark \text{ for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n$

to keep track of the timing information in the protocol Init and

- $\text{SendBit-}\checkmark(n, m, k) := _ \leftarrow \text{SendBit}(n, m, k); \text{ret } \checkmark \text{ for } n, m \leq N + 1 \text{ and } k < K$
- $\text{RcvdBit-}\checkmark(n, m, k) := _ \leftarrow \text{RcvdBit}(n, m, k); \text{ret } \checkmark \text{ for } n, m \leq N + 1 \text{ and } k < K$
- $\text{Ctrb-}\checkmark(n, m, k) := _ \leftarrow \text{Ctrb}(n, m, k); \text{ret } \checkmark \text{ for } n, m \leq N + 1 \text{ and } k < K$

- $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, k) := _ \leftarrow \text{Ctrb-}\Sigma(n, m, k)$; ret \checkmark for $n, m \leq N + 1$ and $k < K$

to keep track of the timing information in the protocol $\text{Shares}(C, K)$. Call the following protocol fragment $\text{Init-}\checkmark$:

- $\text{InShare-}\$-\checkmark(m, n, i) := _ \leftarrow \text{read InShare-}\(m, n, i) ; ret \checkmark for $m, n \leq N + 1$ and $i < I_n$
- $\text{InShare-}\$-\Sigma\text{-}\checkmark(m, n, i) := _ \leftarrow \text{read InShare-}\$-\Sigma(m, n, i)$; ret \checkmark for $m \leq N$ and $n \leq N + 1$ and $i < I_n$
- $\text{InShare-}\checkmark(m, n, i) := _ \leftarrow \text{read InShare}(m, n, i)$; ret \checkmark for $m, n \leq N + 1$ and $i < I_n$

Call the following protocol fragment $\text{Shares-}\checkmark(C, K)$:

- $\text{SendBit-}\checkmark(n, m, k) := _ \leftarrow \text{SendBit}(n, m, k)$; ret \checkmark for $n, m \leq N + 1$ and $k < K$
- $\text{RcvdBit-}\checkmark(n, m, k) := _ \leftarrow \text{RcvdBit}(n, m, k)$; ret \checkmark for $n, m \leq N + 1$ and $k < K$
- $\text{Ctrb-}\checkmark(n, m, k) := _ \leftarrow \text{Ctrb}(n, m, k)$; ret \checkmark for $n, m \leq N + 1$ and $k < K$
- $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, k) := _ \leftarrow \text{Ctrb-}\Sigma(n, m, k)$; ret \checkmark for $n, m \leq N + 1$ and $k < K$
- $\text{Share-}\checkmark(n, k) := _ \leftarrow \text{read Share}(n, k)$; ret \checkmark for $n \leq N + 1$ and $k < K$

First we observe that in the presence of the channels $\text{Share-}\Sigma(-, -)$ and $\text{Share-}\checkmark(-, -)$ we can express the channels

- $\text{Share-}\Sigma\text{-}\checkmark(m, k) := _ \leftarrow \text{read Share-}\Sigma(m, k)$; ret \checkmark for $m \leq N$ and $k < K$

equivalently as follows:

- $$\begin{cases} \text{Share-}\Sigma\text{-}\checkmark(0, k) := \text{read Share-}\checkmark(0, k) \\ \text{for } k < K \\ \text{Share-}\Sigma\text{-}\checkmark(m + 1, k) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(m, k); _ \leftarrow \text{Share-}\checkmark(m + 1, k); \text{ret } \checkmark \\ \text{for } m < N \text{ and } k < K \end{cases}$$

We further observe that in the presence of the channels $\text{InShare-}\checkmark(-, -, -)$ and the protocol $\text{Shares}(C, K)$ we can express the protocol $\text{Shares-}\checkmark(C, K)$ equivalently as follows:

- $\text{Shares-}\checkmark(\epsilon, 0)$ is the protocol 0
- $\text{Shares-}\checkmark(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share-}\checkmark(n, K) := \text{read InShare-}\checkmark(n, p, i)$ for $n \leq N$
 - $\text{Share-}\checkmark(N + 1, K) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(N, K); \text{read InShare-}\checkmark(N + 1, p, i)$
- $\text{Shares-}\checkmark(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share-}\checkmark(n, K) := \text{read Share-}\checkmark(n, k)$ for $n \leq N$
 - $\text{Share-}\checkmark(N + 1, K) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(N, K); _ \leftarrow \text{Share-}\checkmark(N + 1, k); \text{ret } \checkmark$
- $\text{Shares-}\checkmark(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol

- $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
- $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
- $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
- $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
- $\text{Share-}\checkmark(n, K) := _ \leftarrow \text{Share-}\checkmark(n, k); _ \leftarrow \text{Share-}\checkmark(n, l); \text{ret } \checkmark$ for $n \leq N$
- $\text{Share-}\checkmark(N + 1, K) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(N, K); _ \leftarrow \text{Share-}\checkmark(N + 1, k); _ \leftarrow \text{Share-}\checkmark(N + 1, l); \text{ret } \checkmark$

- $\text{Shares-}\checkmark(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol

- $\left\{ \begin{array}{l} \text{SendBit-}\checkmark(n, m, K) := _ \leftarrow \text{Share-}\checkmark(n, k); _ \leftarrow \text{Share-}\checkmark(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{array} \right.$
- $\text{RcvdBit-}\checkmark(n, m, K) := _ \leftarrow \text{SendBit-}\checkmark(m, n, K); _ \leftarrow \text{Share-}\checkmark(m, k); _ \leftarrow \text{Share-}\checkmark(m, l);$
 $_ \leftarrow \text{Share-}\checkmark(n, k); _ \leftarrow \text{Share-}\checkmark(n, l); \text{ret } \checkmark$ for $n, m \leq N + 1$
- $\left\{ \begin{array}{l} \text{Ctrb-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{Ctrb-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{Ctrb-}\checkmark(n, n, K) := _ \leftarrow \text{Share-}\checkmark(n, k); _ \leftarrow \text{Share-}\checkmark(n, l); \text{ret } \checkmark \\ \text{for } n \leq N + 1 \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Ctrb-}\Sigma\text{-}\checkmark(n, 0, K) := \text{read Ctrb-}\checkmark(n, 0, K) \\ \text{for } n \leq N + 1 \\ \text{Ctrb-}\Sigma\text{-}\checkmark(n, m + 1, K) := _ \leftarrow \text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K); _ \leftarrow \text{Ctrb-}\checkmark(n, m + 1, K); \text{ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } m \leq N \end{array} \right.$
- $\text{Share-}\checkmark(n, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, N + 1, K)$ for $n \leq N$
- $\text{Share-}\checkmark(N + 1, K) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(N, K); \text{read Ctrb-}\Sigma\text{-}\checkmark(N + 1, N + 1, K)$

We also observe that in the presence of the protocol Init we can define the protocol $\text{Init-}\checkmark$ equivalently as follows:

- $\text{InShare-}\$-\checkmark(m, n, i) := _ \leftarrow \text{In-}\checkmark(n, i); \text{ret } \checkmark$ for $m \leq N$ and $n \leq N + 1$ and $i < I_n$
- $\left\{ \begin{array}{l} \text{InShare-}\$-\Sigma\text{-}\checkmark(0, n, i) := \text{read InShare-}\$-\checkmark(0, n, i) \\ \text{for } n \leq N + 1 \text{ and } i < I_n \\ \text{InShare-}\$-\Sigma\text{-}\checkmark(m + 1, n, i) := _ \leftarrow \text{InShare-}\$-\Sigma\text{-}\checkmark(m, n, i); _ \leftarrow \text{InShare-}\$-\checkmark(m + 1, n, i); \text{ret } \checkmark \\ \text{for } m < N \text{ and } n \leq N + 1 \text{ and } i < I_n \end{array} \right.$
- $\text{InShare-}\$-\checkmark(N + 1, n, i) := _ \leftarrow \text{InShare-}\$-\Sigma\text{-}\checkmark(N, n, i); _ \leftarrow \text{In-}\checkmark(n, i); \text{ret } \checkmark$ for $n \leq N + 1$ and $i < I_n$
- $\text{InShare-}\checkmark(m, n, i) := \text{read InShare-}\$-\checkmark(m, n, i)$ for $m, n \leq N + 1$ and $i < I_n$

Furthermore, in the presence of the channels $\text{In-}\checkmark(-, -)$ we can express the protocol $\text{Init-}\checkmark$ equivalently as follows:

- $\text{InShare-}\$-\checkmark(m, n, i) := \text{read In-}\checkmark(n, i)$ for $m, n \leq N + 1$ and $i < I_n$
- $\text{InShare-}\$-\Sigma\text{-}\checkmark(m, n, i) := \text{read In-}\checkmark(n, i)$ for $m \leq N$ and $n \leq N + 1$ and $i < I_n$
- $\text{InShare-}\checkmark(m, n, i) := \text{read In-}\checkmark(n, i)$ for $m, n \leq N + 1$ and $i < I_n$

We can now merge the channels

- $\begin{cases} \text{Share-}\Sigma\text{-}\checkmark(0, k) := \text{read Share-}\checkmark(0, k) \\ \text{for } k < K \\ \text{Share-}\Sigma\text{-}\checkmark(m+1, k) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(m, k); _ \leftarrow \text{Share-}\checkmark(m+1, k); \text{ret } \checkmark \\ \text{for } m < N \text{ and } k < K \end{cases}$

with the protocol $\text{Shares-}\checkmark(C, K)$ to obtain the following new form of the protocol $\text{Shares-}\checkmark(C, K)$:

- $\text{Shares-}\checkmark(\epsilon, 0)$ is the protocol 0
- $\text{Shares-}\checkmark(C; \text{input-gate}(p, i), K+1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N+1$
 - $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $n, m \leq N+1$
 - $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N+1$
 - $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, m, K)$ for $n, m \leq N+1$
 - $\text{Share-}\checkmark(n, K) := \text{read InShare-}\checkmark(n, p, i)$ for $n \leq N$
 - $\begin{cases} \text{Share-}\Sigma\text{-}\checkmark(0, k) := \text{read Share-}\checkmark(0, k) \\ \text{for } k < K \\ \text{Share-}\Sigma\text{-}\checkmark(m+1, k) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(m, k); _ \leftarrow \text{Share-}\checkmark(m+1, k); \text{ret } \checkmark \\ \text{for } m < N \text{ and } k < K \end{cases}$
 - $\text{Share-}\checkmark(N+1, K) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(N, K); \text{read InShare-}\checkmark(N+1, p, i)$
- $\text{Shares-}\checkmark(C; \text{not-gate}(k), K+1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N+1$
 - $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $n, m \leq N+1$
 - $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N+1$
 - $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, m, K)$ for $n, m \leq N+1$
 - $\text{Share-}\checkmark(n, K) := \text{read Share-}\checkmark(n, k)$ for $n \leq N$
 - $\begin{cases} \text{Share-}\Sigma\text{-}\checkmark(0, k) := \text{read Share-}\checkmark(0, k) \\ \text{for } k < K \\ \text{Share-}\Sigma\text{-}\checkmark(m+1, k) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(m, k); _ \leftarrow \text{Share-}\checkmark(m+1, k); \text{ret } \checkmark \\ \text{for } m < N \text{ and } k < K \end{cases}$
 - $\text{Share-}\checkmark(N+1, K) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(N, K); _ \leftarrow \text{Share-}\checkmark(N+1, k); \text{ret } \checkmark$
- $\text{Shares-}\checkmark(C; \text{xor-gate}(k, l), K+1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N+1$
 - $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $n, m \leq N+1$
 - $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N+1$
 - $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, m, K)$ for $n, m \leq N+1$
 - $\text{Share-}\checkmark(n, K) := _ \leftarrow \text{Share-}\checkmark(n, k); _ \leftarrow \text{Share-}\checkmark(n, l); \text{ret } \checkmark$ for $n \leq N$
 - $\begin{cases} \text{Share-}\Sigma\text{-}\checkmark(0, k) := \text{read Share-}\checkmark(0, k) \\ \text{for } k < K \\ \text{Share-}\Sigma\text{-}\checkmark(m+1, k) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(m, k); _ \leftarrow \text{Share-}\checkmark(m+1, k); \text{ret } \checkmark \\ \text{for } m < N \text{ and } k < K \end{cases}$
 - $\text{Share-}\checkmark(N+1, K) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(N, K); _ \leftarrow \text{Share-}\checkmark(N+1, k); _ \leftarrow \text{Share-}\checkmark(N+1, l); \text{ret } \checkmark$

- $\text{Shares-}\checkmark(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares-}\checkmark(C, K)$ with the protocol

$$\begin{aligned}
& \left\{ \begin{array}{l} \text{SendBit-}\checkmark(n, m, K) := _ \leftarrow \text{Share-}\checkmark(n, k); _ \leftarrow \text{Share-}\checkmark(n, l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n < m \end{array} \right. \\
& \left\{ \begin{array}{l} \text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{array} \right. \\
& \text{RcvdBit-}\checkmark(n, m, K) := _ \leftarrow \text{SendBit-}\checkmark(m, n, K); _ \leftarrow \text{Share-}\checkmark(m, k); _ \leftarrow \text{Share-}\checkmark(m, l); \\
& _ \leftarrow \text{Share-}\checkmark(n, k); _ \leftarrow \text{Share-}\checkmark(n, l); \text{ret } \checkmark \text{ for } n, m \leq N + 1 \\
& \left\{ \begin{array}{l} \text{Ctrb-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{Ctrb-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{Ctrb-}\checkmark(n, n, K) := _ \leftarrow \text{Share-}\checkmark(n, k); _ \leftarrow \text{Share-}\checkmark(n, l); \text{ret } \checkmark \\ \text{for } n \leq N + 1 \end{array} \right. \\
& \left\{ \begin{array}{l} \text{Ctrb-}\Sigma\text{-}\checkmark(n, 0, K) := \text{read Ctrb-}\checkmark(n, 0, K) \\ \text{for } n \leq N + 1 \\ \text{Ctrb-}\Sigma\text{-}\checkmark(n, m + 1, K) := _ \leftarrow \text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K); _ \leftarrow \text{Ctrb-}\checkmark(n, m + 1, K); \text{ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } m \leq N \end{array} \right. \\
& \text{Share-}\checkmark(n, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, N + 1, K) \text{ for } n \leq N \\
& \left\{ \begin{array}{l} \text{Share-}\Sigma\text{-}\checkmark(0, k) := \text{read Share-}\checkmark(0, k) \\ \text{for } k < K \\ \text{Share-}\Sigma\text{-}\checkmark(m + 1, k) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(m, k); _ \leftarrow \text{Share-}\checkmark(m + 1, k); \text{ret } \checkmark \\ \text{for } m < N \text{ and } k < K \end{array} \right. \\
& \text{Share-}\checkmark(N + 1, K) := _ \leftarrow \text{Share-}\Sigma\text{-}\checkmark(N, K); \text{read Ctrb-}\Sigma\text{-}\checkmark(N + 1, N + 1, K)
\end{aligned}$$

In the presence of the channels $\text{InShare-}\checkmark(-, -, -)$ as well as the protocol $\text{Wires-}\checkmark(C, K)$ we can express the protocol $\text{Shares-}\checkmark(C, K)$ equivalently as the channels

- $\text{Share-}\checkmark(n, k) := \text{read Wire-}\checkmark(k)$ for $n \leq N + 1$ and $k < K$
- $\text{Share-}\Sigma\text{-}\checkmark(m, k) := \text{read Wire-}\checkmark(k)$ for $m \leq N$ and $k < K$

together with the following protocol $\text{Ctrbs-}\checkmark(C, K)$:

- $\text{Ctrbs-}\checkmark(\epsilon, 0)$ is the protocol 0
- $\text{Ctrbs-}\checkmark(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Ctrbs-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
- $\text{Ctrbs-}\checkmark(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Ctrbs-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $m, n \leq N + 1$
 - $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, m, K)$ for $n, m \leq N + 1$

- $\text{Ctrbs-}\checkmark(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Ctrbs-}\checkmark(C, K)$ with the protocol
 - $\text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K)$ for $m, n \leq N + 1$
 - $\text{Ctrb-}\checkmark(n, m, K) := \text{read Ctrb-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K) := \text{read Ctrb-}\Sigma\text{-}\checkmark(n, m, K)$ for $n, m \leq N + 1$
- $\text{Ctrbs-}\checkmark(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Ctrbs-}\checkmark(C, K)$ with the protocol
 - $\begin{cases} \text{SendBit-}\checkmark(n, m, K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{SendBit-}\checkmark(n, m, K) := \text{read SendBit-}\checkmark(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{cases}$
 - $\begin{cases} \text{RcvdBit-}\checkmark(n, m, K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{RcvdBit-}\checkmark(n, m, K) := \text{read RcvdBit-}\checkmark(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \geq n \end{cases}$
 - $\text{Ctrb-}\checkmark(n, m, K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma\text{-}\checkmark(n, m, K) := _ \leftarrow \text{Wire-}\checkmark(k); _ \leftarrow \text{Wire-}\checkmark(l); \text{ret } \checkmark$ for $n, m \leq N + 1$

We now revisit the case of an *and* gate in the protocol $\text{Adv}(C, K)$. For any party n with $n \leq N$ we can write the channels

- $\text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\Sigma\text{-}\checkmark(N, k)$
- $\text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Share-}\Sigma\text{-}\checkmark(N, l)$

equivalently as follows:

- $\text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(k)$
- $\text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(l)$

We thus have the following for channels $\text{OTChcRcvd}_0(-, N + 1, K)_{\text{adv}}^{\text{ot}}$ and $\text{OTChcRcvd}_1(-, N + 1, K)_{\text{adv}}^{\text{ot}}$:

- $\begin{cases} \text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(k) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(l) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{cases}$

So the inductive part of the real protocol now has $\text{Adv}(C, K)$ looking like so:

- $\text{Adv}(\epsilon, 0)$ is the protocol 0
- $\text{Adv}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$

- [illegible]

- $\text{Adv}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol

- $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
- $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
- $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
- $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
- $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$

- $\text{Adv}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol

- $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
- $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
- $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
- $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
- $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$

- $\text{Adv}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol

- $\begin{cases} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$

[illegible]

$$\begin{aligned}
& - \begin{cases} \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, l) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest and } n < m \\ \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest or } n \geq m \end{cases} \\
& - \begin{cases} \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := x_m \leftarrow \text{Share}(m, k); \text{ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ semi-honest or } n \geq m \\ \text{OTChcRcvd}_0(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(k) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{cases} \\
& - \begin{cases} \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := y_m \leftarrow \text{Share}(m, l); \text{ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ if } m \text{ semi-honest or } n \geq m \\ \text{OTChcRcvd}_1(n, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read Wire-}\checkmark(l) \\ \text{for } n \leq N \\ \text{OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(N + 1, N + 1, K)_{\text{adv}}^{\text{ot}} \end{cases} \\
& - \begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read RcvdBit}(m, n, K) \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ semi-honest} \\ \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n, m \leq N + 1 \text{ if } m \text{ honest} \end{cases}
\end{aligned}$$

Finally, in the presence of the channels $\text{Share-}\Sigma\checkmark(-, -)$ we can characterize the protocol $\text{Shares}(C, K)$ equivalently as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := \text{read InShare}(n, p, i)$ for $n \leq N$
 - $\text{Share}(N + 1, K) := _ \leftarrow \text{Wire-}\checkmark(K); \text{read InShare}(N + 1, p, i)$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := \text{read Share}(n, k)$ for $n \leq N$
 - $\text{Share}(N + 1, K) := _ \leftarrow \text{Wire-}\checkmark(K); x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ret } \neg x_{N+1}$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol

- $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
- $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
- $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
- $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
- $\text{Share}(n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n \oplus y_n$ for $n \leq N$
- $\text{Share}(N + 1, K) := _ \leftarrow \text{Wire-}\checkmark(K); x_{N+1} \leftarrow \text{Share}(N + 1, k); y_{N+1} \leftarrow \text{Share}(N + 1, l); \text{ret } x_{N+1} \oplus y_{N+1}$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\left\{ \begin{array}{l} \text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{samp flip} \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{array} \right.$
 - $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l);$
 $x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n, m \leq N + 1$
 - $\left\{ \begin{array}{l} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n * y_n \\ \text{for } n \leq N + 1 \end{array} \right.$
 - $\left\{ \begin{array}{l} \text{Ctrb-}\Sigma(n, 0, K) := \text{read Ctrb}(n, 0, K) \\ \text{for } n \leq N + 1 \\ \text{Ctrb-}\Sigma(n, m + 1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(n, m, K); b \leftarrow \text{Ctrb}(n, m + 1, K); \text{ret } b_\Sigma \oplus b \\ \text{for } n \leq N + 1 \text{ and } m \leq N \end{array} \right.$
 - $\text{Share}(n, K) := \text{read Ctrb-}\Sigma(n, N + 1, K)$ for $n \leq N$
 - $\text{Share}(N + 1, K) := _ \leftarrow \text{Wire-}\checkmark(K); \text{read Ctrb-}\Sigma(N + 1, N + 1, K)$

At this point, the channels $\text{Share-}\Sigma\checkmark(-, -)$ 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 $\text{Adv}(C, K)$ and $\text{Shares}(C, K)$ we made at the beginning of this section.

10.4.6 Sum Of Shares

We continue to simplify the real protocol after amending $\text{Adv}(C, K)$ and $\text{Shares}(C, K)$ as indicated in Section 10.4.5. Recall the channels

- $\left\{ \begin{array}{l} \text{Share-}\Sigma(0, k) := \text{read Share}(0, k) \\ \text{for } k < K \\ \text{Share-}\Sigma(m + 1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); x_{m+1} \leftarrow \text{Share}(m + 1, k); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \text{ and } k < K \end{array} \right.$

that inductively compute the the sum of shares for parties $0, \dots, N$ on each wire $k < K$. To also include the shares of party $N + 1$, we add new internal channels as follows:

- $\text{Share-}\Sigma(N + 1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ret } x_\Sigma \oplus x_{N+1}$ 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

- $\text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k)$

reads from the divergent channel

- $\text{OutShare}(n, m, k) := \text{read OutShare}(n, m, k)$

and thus we may equivalently write the following:

- $\text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)}$

If party n is semi-honest and wire k is an output, then for any party m the channel

- $\text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k)$

reads from the channel

- $\text{OutShare}(n, m, k) := \text{read Share}(m, k)$

so we may substitute:

- $\text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k)$

Summarizing the above, we get the following for channels $\text{OutShare}(-, -, -)_{\text{adv}}^{\text{party}(-)}$:

- $$\begin{cases} \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$$

If wire k is not an output, then we can rewrite the channels

- $$\begin{cases} \text{OutShare-}\Sigma(n, 0, k) := \text{read OutShare}(n, 0, k) \\ \text{for } n \leq N + 1 \text{ and } k < K \\ \text{OutShare-}\Sigma(n, m + 1, k) := x_{\Sigma} \leftarrow \text{OutShare-}\Sigma(n, m, k); x_{m+1} \leftarrow \text{OutShare}(n, m + 1, k); \text{ret } x_{\Sigma} \oplus x_{m+1} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \end{cases}$$

to the following:

- $\text{OutShare-}\Sigma(n, m, k) := \text{read OutShare-}\Sigma(n, m, k) \text{ for } n, m \leq N + 1$

If wire k is an output, then we can rewrite the channels

- $$\begin{cases} \text{OutShare-}\Sigma(n, 0, k) := \text{read OutShare}(n, 0, k) \\ \text{for } n \leq N + 1 \text{ and } k < K \\ \text{OutShare-}\Sigma(n, m + 1, k) := x_{\Sigma} \leftarrow \text{OutShare-}\Sigma(n, m, k); x_{m+1} \leftarrow \text{OutShare}(n, m + 1, k); \text{ret } x_{\Sigma} \oplus x_{m+1} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \end{cases}$$

to the following:

- $\text{OutShare-}\Sigma(n, m, k) := \text{read Share-}\Sigma(m, k) \text{ for } n, m \leq N + 1$

Summarizing the above, we get the following for channels $\text{OutShare-}\Sigma(-, -, -)$:

- $$\begin{cases} \text{OutShare-}\Sigma(n, m, k) := \text{read Share-}\Sigma(m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } k \text{ an output} \\ \text{OutShare-}\Sigma(n, m, k) := \text{read OutShare-}\Sigma(n, m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } k \text{ not an output} \end{cases}$$

If party n is semi-honest and wire k is not an output, then for any party m the channel

- $\text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k)$

reads from the divergent channel

- $\text{OutShare-}\Sigma(n, m, k) := \text{read OutShare-}\Sigma(n, m, k)$

and thus we may equivalently write the following:

- $\text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)}$

If party n is semi-honest and wire k is an output, then for any party m the channel

- $\text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k)$

reads from the channel

- $\text{OutShare-}\Sigma(n, m, k) := \text{read Share-}\Sigma(m, k)$

so we may substitute:

- $\text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share-}\Sigma(m, k)$

Summarizing the above, we get the following for channels $\text{OutShare-}\Sigma(-, -, -)_{\text{adv}}^{\text{party}(-)}$:

- $$\begin{cases} \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share-}\Sigma(m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$$

Finally, we can express the channels

- $\text{Out}(n, k) := \text{read OutShare-}\Sigma(n, N + 1, k) \text{ for } n \leq N + 1$

equivalently as follows:

- $$\begin{cases} \text{Out}(n, k) := \text{read Share-}\Sigma(N + 1, k) \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } k \text{ an output} \\ \text{Out}(n, k) := \text{read Out}(n, k) \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } k \text{ not an output} \end{cases}$$

At this point, the internal channels $\text{OutShare}(-, -, -)$ and $\text{OutShare-}\Sigma(-, -, -)$ are unused and can be eliminated. The simplified version Fin of the final part of the real protocol is therefore as follows:

- $$\begin{cases} \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, k) \\ \text{for } m, n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m, n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$$
- $$\begin{cases} \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$$
- $$\begin{cases} \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$$

- $\begin{cases} \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share-}\Sigma(m, k) \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{Out}(n, k) := \text{read Share-}\Sigma(N + 1, k) \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } k \text{ an output} \\ \text{Out}(n, k) := \text{read Out}(n, k) \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k) \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

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 $0, \dots, N$ plus the value on the wire. We proceed by an induction on circuits: in the presence of the protocol `Init` and the channels

- $\begin{cases} \text{Share-}\Sigma(0, k) := \text{read Share}(0, k) \\ \text{for } k < K \\ \text{Share-}\Sigma(m + 1, k) := x_{\Sigma} \leftarrow \text{Share-}\Sigma(m, k); x_{m+1} \leftarrow \text{Share}(m + 1, k); \text{ret } x_{\Sigma} \oplus x_{m+1} \\ \text{for } m < N \text{ and } k < K \end{cases}$

we can express the channels

- $\text{Share-}\Sigma(N + 1, k) := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ret } x_{\Sigma} \oplus x_{N+1} \text{ for } k < K$

together with the protocol `Shares(C, K)` equivalently as the channels

- $\text{Share}(N + 1, k) := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Share-}\Sigma(N + 1, k); \text{ret } x_{\Sigma} \oplus x \text{ for } k < K$

together with the following new form of the protocol `Shares(C, K)`:

- `Shares($\epsilon, 0$)` is the protocol 0
- `Shares($C; \text{input-gate}(p, i), K + 1$)` is the composition of `Shares(C, K)` with the protocol
 - `SendBit(n, m, K)` := read `SendBit(n, m, K)` for $n, m \leq N + 1$
 - `RcvdBit(n, m, K)` := read `RcvdBit(n, m, K)` for $n, m \leq N + 1$
 - `Ctrb(n, m, K)` := read `Ctrb(n, m, K)` for $n, m \leq N + 1$
 - `Ctrb- Σ (n, m, K)` := read `Ctrb- Σ (n, m, K)` for $n, m \leq N + 1$
 - `Share(n, K)` := read `InShare(n, p, i)` for $n \leq N$
 - `Share- Σ ($N + 1, K$)` := read `In(p, i)`
- `Shares($C; \text{not-gate}(k), K + 1$)` is the composition of `Shares(C, K)` with the protocol
 - `SendBit(n, m, K)` := read `SendBit(n, m, K)` for $n, m \leq N + 1$
 - `RcvdBit(n, m, K)` := read `RcvdBit(n, m, K)` for $n, m \leq N + 1$
 - `Ctrb(n, m, K)` := read `Ctrb(n, m, K)` for $n, m \leq N + 1$
 - `Ctrb- Σ (n, m, K)` := read `Ctrb- Σ (n, m, K)` for $n, m \leq N + 1$
 - `Share(n, K)` := read `Share(n, k)` for $n \leq N$

- $\text{Share-}\Sigma(N+1, K) := x \leftarrow \text{Share-}\Sigma(N+1, k); \text{ret } \neg x$
- $\text{Shares}(C; \text{xor-gate}(k, l), K+1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N+1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N+1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N+1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N+1$
 - $\text{Share}(n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n \oplus y_n$ for $n \leq N$
 - $\text{Share-}\Sigma(N+1, K) := x \leftarrow \text{Share-}\Sigma(N+1, k); y \leftarrow \text{Share-}\Sigma(N+1, l); \text{ret } x \oplus y$
- $\text{Shares}(C; \text{and-gate}(k, l), K+1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\begin{cases} \text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{samp flip} \\ \text{for } n, m \leq N+1 \text{ if } n < m \\ \text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N+1 \text{ if } n \geq m \end{cases}$
 - $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l);$
 $x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n, m \leq N+1$
 - $\begin{cases} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N+1 \text{ if } n < m \\ \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N+1 \text{ if } m < n \\ \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n * y_n \\ \text{for } n \leq N+1 \end{cases}$
 - $\begin{cases} \text{Ctrb-}\Sigma(n, 0, K) := \text{read Ctrb}(n, 0, K) \\ \text{for } n \leq N+1 \\ \text{Ctrb-}\Sigma(n, m+1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(n, m, K); b \leftarrow \text{Ctrb}(n, m+1, K); \text{ret } b_\Sigma \oplus b \\ \text{for } n \leq N+1 \text{ and } m \leq N \end{cases}$
 - $\text{Share}(n, K) := \text{read Ctrb-}\Sigma(n, N+1, K)$ for $n \leq N$
 - $\text{Share-}\Sigma(N+1, K) := x \leftarrow \text{Share-}\Sigma(N+1, k); y \leftarrow \text{Share-}\Sigma(N+1, l); \text{ret } x * y$

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
 - $\text{Share}(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); \text{read InShare}(N+1, p, i)$

into the closed form of the channel

- $\text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Share}(N+1, K); \text{ret } x_\Sigma \oplus x_{N+1}$

which yields the following:

- $\text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{InShare}(N+1, p, i); \text{ret } x_\Sigma \oplus x_{N+1}$

By canceling out two applications of $x_\Sigma \oplus$ – we can reformulate the channel $\text{Share}(N+1, K)$ as follows:

- $\text{Share}(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{InShare}(N+1, p, i); \text{ret } x_\Sigma \oplus (x_\Sigma \oplus x_{N+1})$

The above can be expressed more concisely as

- $\text{Share}(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x \leftarrow \text{Share-}\Sigma(N+1, K); \text{ret } x_\Sigma \oplus x$

and this is the desired closed form of the channel $\text{Share}(N + 1, K)$.

We can now turn our attention to the sum of shares. In the presence of the channels

- $\text{Share}(m, K) := \text{read } \text{InShare}(m, p, i)$ for $m \leq N$

and the channels

- $\text{InShare}(m, p, i) := \text{read } \text{InShare-}\(m, p, i) for $m \leq N$
- $\begin{cases} \text{InShare-}\$(0, p, i) := \text{read } \text{InShare-}\$(0, p, i) \\ \text{InShare-}\$(m + 1, p, i) := x_\Sigma \leftarrow \text{InShare-}\$(m, p, i); x_{m+1} \leftarrow \text{InShare-}\$(m + 1, p, i); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \end{cases}$

from the protocol Init we can rewrite the inductive form of the channels

- $\begin{cases} \text{Share-}\Sigma(0, K) := \text{read } \text{Share}(0, K) \\ \text{Share-}\Sigma(m + 1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, K); x_{m+1} \leftarrow \text{Share}(m + 1, K); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \end{cases}$

to the following closed form:

- $\text{Share-}\Sigma(m, K) := \text{read } \text{InShare-}\(m, p, i) for $m \leq N$

Substituting the channel $\text{Share-}\Sigma(N, K)$ into the channel

- $\text{Share-}\Sigma(N + 1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{InShare}(N + 1, p, i); \text{ret } x_\Sigma \oplus x_{N+1}$

thus yields the following:

- $\text{Share-}\Sigma(N + 1, K) := x_\Sigma \leftarrow \text{InShare-}\$(N, p, i); x_{N+1} \leftarrow \text{InShare}(N + 1, p, i); \text{ret } x_\Sigma \oplus x_{N+1}$

At this point, we rewrite the closed form of the channels

- $\text{Share-}\Sigma(m, K) := \text{read } \text{InShare-}\(m, p, i) for $m \leq N$

back to their original inductive form:

- $\begin{cases} \text{Share-}\Sigma(0, K) := \text{read } \text{Share}(0, K) \\ \text{Share-}\Sigma(m + 1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, K); x_{m+1} \leftarrow \text{Share}(m + 1, K); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \end{cases}$

In the presence of the channels

- $\text{InShare}(N + 1, p, i) := \text{read } \text{InShare-}\$(N + 1, p, i)$
- $\text{InShare-}\$(N + 1, p, i) := x_\Sigma \leftarrow \text{InShare-}\$(N, p, i); x \leftarrow \text{In}(p, i); \text{ret } x_\Sigma \oplus x$

from the protocol Init we can further write the channel

- $\text{Share-}\Sigma(N + 1, K) := x_\Sigma \leftarrow \text{InShare-}\$(N, p, i); x_{N+1} \leftarrow \text{InShare}(N + 1, p, i); \text{ret } x_\Sigma \oplus x_{N+1}$

equivalently as follows:

- $\text{Share-}\Sigma(N + 1, K) := x_\Sigma \leftarrow \text{InShare-}\$(N, p, i); x \leftarrow \text{In}(p, i); \text{ret } x_\Sigma \oplus (x_\Sigma \oplus x)$

We can cancel out the two applications of $x_\Sigma \oplus -$:

- $\text{Share-}\Sigma(N + 1, K) := x_\Sigma \leftarrow \text{InShare-}\$(N, p, i); \text{read } \text{In}(p, i)$

This is almost what we want except for the extra dependency on the channel $\text{InShare-}\$(N, p, i)$. To remove this dependency, we introduce new internal channels as follows:

- $\text{In-}\checkmark(p, i) := _ \leftarrow \text{read } \text{In}(p, i); \text{ ret } \checkmark$
- $\text{InShare-}\$-\checkmark(m, p, i) := _ \leftarrow \text{read } \text{InShare-}\$(m, p, i); \text{ ret } \checkmark \text{ for } m \leq N$
- $\text{InShare-}\$-\Sigma-\checkmark(m, p, i) := _ \leftarrow \text{read } \text{InShare-}\$-\Sigma(m, p, i); \text{ ret } \checkmark \text{ for } m \leq N$

In the presence of the channel $\text{InShare-}\$-\Sigma-\checkmark(N, p, i)$ we can write the channel

- $\text{Share-}\Sigma(N + 1, K) := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(N, p, i); \text{ read } \text{In}(p, i)$

equivalently as follows:

- $\text{Share-}\Sigma(N + 1, K) := _ \leftarrow \text{InShare-}\$-\Sigma-\checkmark(N, p, i); \text{ read } \text{In}(p, i)$

We can characterize the timing channels $\text{InShare-}\$-\checkmark(-, p, i)$, $\text{InShare-}\$-\Sigma-\checkmark(-, p, i)$ equivalently as follows:

- $\text{InShare-}\$-\checkmark(m, p, i) := _ \leftarrow \text{In-}\checkmark(p, i); \text{ ret } \checkmark \text{ for } m \leq N$
- $\begin{cases} \text{InShare-}\$-\Sigma-\checkmark(0, p, i) := \text{read } \text{InShare-}\$-\checkmark(0, p, i) \\ \text{InShare-}\$-\Sigma-\checkmark(m + 1, p, i) := _ \leftarrow \text{InShare-}\$-\Sigma-\checkmark(m, p, i); _ \leftarrow \text{InShare-}\$-\checkmark(m + 1, p, i); \text{ ret } \checkmark \\ \text{for } m < N \end{cases}$

Furthermore, we can write the above form of the timing channels equivalently as follows:

- $\text{InShare-}\$-\checkmark(m, p, i) := \text{read } \text{In-}\checkmark(p, i) \text{ for } m \leq N$
- $\text{InShare-}\$-\Sigma-\checkmark(m, p, i) := \text{read } \text{In-}\checkmark(p, i) \text{ for } m \leq N$

In the presence of the channel $\text{InShare-}\$-\Sigma-\checkmark(N, p, i)$ we can thus write the channel

- $\text{Share-}\Sigma(N + 1, K) := _ \leftarrow \text{InShare-}\$-\Sigma-\checkmark(N, p, i); \text{ read } \text{In}(p, i)$

equivalently as follows:

- $\text{Share-}\Sigma(N + 1, K) := _ \leftarrow \text{In-}\checkmark(p, i); \text{ read } \text{In}(p, i)$

In the presence of the channel $\text{In-}\checkmark(p, i)$ we can rewrite the above to the following:

- $\text{Share-}\Sigma(N + 1, K) := \text{read } \text{In}(p, i)$

But this is precisely the desired inductive form of the channel $\text{Share-}\Sigma(N + 1, K)$.

The channels $\text{In-}\checkmark(p, i)$, $\text{InShare-}\$-\checkmark(-, p, i)$, $\text{InShare-}\$-\Sigma-\checkmark(-, p, i)$ are now unused and can be discarded.

- In the case of a *not* gate, we start by substituting the inductive form of the channel

- $\text{Share}(N + 1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ ret } \neg x_{N+1}$

into the closed form of the channel

- $\text{Share-}\Sigma(N + 1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Share}(N + 1, K); \text{ ret } x_\Sigma \oplus x_{N+1}$

which yields the following:

- $\text{Share-}\Sigma(N + 1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ ret } x_\Sigma \oplus (\neg x_{N+1})$

By canceling out two applications of $x_\Sigma \oplus -$ we can reformulate the channel $\text{Share}(N + 1, K)$ as follows:

- $\text{Share}(N + 1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ ret } x_\Sigma \oplus (x_\Sigma \oplus (\neg x_{N+1}))$

The above can be expressed more concisely as

- $\text{Share}(N + 1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x \leftarrow \text{Share-}\Sigma(N + 1, K); \text{ ret } x_\Sigma \oplus x$

and this is the desired closed form of the channel $\text{Share}(N + 1, K)$.

We can now turn our attention to the sum of shares. In the presence of the channels

- $\text{Share}(m, K) := \text{read Share}(m, k)$ for $m \leq N$

and the channels

$$- \begin{cases} \text{Share-}\Sigma(0, k) := \text{read Share}(0, k) \\ \text{Share-}\Sigma(m+1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); x_{m+1} \leftarrow \text{Share}(m+1, k); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \end{cases}$$

we can rewrite the inductive form of the channels

$$- \begin{cases} \text{Share-}\Sigma(0, K) := \text{read Share}(0, K) \\ \text{Share-}\Sigma(m+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, K); x_{m+1} \leftarrow \text{Share}(m+1, K); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \end{cases}$$

to the following closed form:

- $\text{Share-}\Sigma(m, K) := \text{read Share-}\Sigma(m, k)$ for $m \leq N$

Substituting the channel $\text{Share-}\Sigma(N, K)$ into the channel

- $\text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Share}(N+1, k); \text{ret } x_\Sigma \oplus (\neg x_{N+1})$

thus yields the following:

- $\text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x_{N+1} \leftarrow \text{Share}(N+1, k); \text{ret } x_\Sigma \oplus (\neg x_{N+1})$

At this point, we rewrite the closed form of the channels

- $\text{Share-}\Sigma(m, K) := \text{read Share-}\Sigma(m, k)$ for $m \leq N$

back to their original inductive form:

$$- \begin{cases} \text{Share-}\Sigma(0, K) := \text{read Share}(0, K) \\ \text{Share-}\Sigma(m+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, K); x_{m+1} \leftarrow \text{Share}(m+1, K); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \end{cases}$$

The negation in the channel

- $\text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x_{N+1} \leftarrow \text{Share}(N+1, k); \text{ret } x_\Sigma \oplus (\neg x_{N+1})$

can be brought to the top level:

- $\text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x_{N+1} \leftarrow \text{Share}(N+1, k); \text{ret } \neg(x_\Sigma \oplus x_{N+1})$

But this is precisely what we get if we substitute the channel

- $\text{Share-}\Sigma(N+1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x_{N+1} \leftarrow \text{Share}(N+1, k); \text{ret } x_\Sigma \oplus x_{N+1}$ (*inductive hypothesis*)

into the desired inductive form of the channel

- $\text{Share-}\Sigma(N+1, K) := x \leftarrow \text{Share-}\Sigma(N+1, k); \text{ret } \neg x$

so we are done.

- In the case of an *xor* gate, we start by substituting the inductive form of the channel

- $\text{Share}(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Share}(N+1, k);$
 $y_{N+1} \leftarrow \text{Share}(N+1, l); \text{ret } x_{N+1} \oplus y_{N+1}$

into the closed form of the channel

- $\text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Share}(N+1, K); \text{ret } x_\Sigma \oplus x_{N+1}$

which yields the following:

- $\text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Share}(N+1, k);$
 $y_{N+1} \leftarrow \text{Share}(N+1, l); \text{ret } x_\Sigma \oplus (x_{N+1} \oplus y_{N+1})$

By canceling out two applications of $x_\Sigma \oplus$ – we can reformulate the channel $\text{Share}(N+1, K)$ as follows:

- $\text{Share}(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Share}(N+1, k);$
 $y_{N+1} \leftarrow \text{Share}(N+1, l); \text{ret } x_\Sigma \oplus (x_\Sigma \oplus (x_{N+1} \oplus y_{N+1}))$

The above can be expressed more concisely as

- $\text{Share}(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x \leftarrow \text{Share-}\Sigma(N+1, K); \text{ret } x_\Sigma \oplus x$

and this is the desired closed form of the channel $\text{Share}(N+1, K)$.

We can now turn our attention to the sum of shares. In the presence of the channels

- $\text{Share}(m, K) := x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l); \text{ret } x_m \oplus y_m \text{ for } m \leq N$

and the channels

- $\begin{cases} \text{Share-}\Sigma(0, k) := \text{read Share}(0, k) \\ \text{Share-}\Sigma(m+1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); x_{m+1} \leftarrow \text{Share}(m+1, k); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \end{cases}$
- $\begin{cases} \text{Share-}\Sigma(0, l) := \text{read Share}(0, l) \\ \text{Share-}\Sigma(m+1, l) := y_\Sigma \leftarrow \text{Share-}\Sigma(m, l); y_{m+1} \leftarrow \text{Share}(m+1, l); \text{ret } y_\Sigma \oplus y_{m+1} \\ \text{for } m < N \end{cases}$

we can rewrite the inductive form of the channels

- $\begin{cases} \text{Share-}\Sigma(0, K) := \text{read Share}(0, K) \\ \text{Share-}\Sigma(m+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, K); x_{m+1} \leftarrow \text{Share}(m+1, K); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \end{cases}$

to the following closed form:

- $\text{Share-}\Sigma(m, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); y_\Sigma \leftarrow \text{Share-}\Sigma(m, l); \text{ret } x_\Sigma \oplus y_\Sigma \text{ for } m \leq N$

In the base case, we substitute the channel

- $\text{Share}(0, K) := x_0 \leftarrow \text{Share}(0, k); y_0 \leftarrow \text{Share}(0, l); \text{ret } x_0 \oplus y_0$

into the inductive form of the channel

- $\text{Share-}\Sigma(0, K) := \text{Share}(0, K)$

which yields the following:

- $\text{Share-}\Sigma(0, K) := x_0 \leftarrow \text{Share}(0, k); y_0 \leftarrow \text{Share}(0, l); \text{ret } x_0 \oplus y_0$

But this is precisely what we get if we substitute the channels

- $\text{Share-}\Sigma(0, k) := x_0 \leftarrow \text{Share}(0, k); \text{ret } x_0$
- $\text{Share-}\Sigma(0, l) := y_0 \leftarrow \text{Share}(0, l); \text{ret } y_0$

into the closed form of the channel

- $\text{Share-}\Sigma(0, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(0, k); y_\Sigma \leftarrow \text{Share-}\Sigma(0, l); \text{ret } x_\Sigma \oplus y_\Sigma$

Hence the inductive form and the closed form of the channel $\text{Share-}\Sigma(0, K)$ are equivalent.

In the inductive case, we substitute the channels

- $\text{Share-}\Sigma(m, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); y_\Sigma \leftarrow \text{Share-}\Sigma(m, l); \text{ret } x_\Sigma \oplus y_\Sigma$ (*inductive hypothesis*)
- $\text{Share}(m+1, K) := x_{m+1} \leftarrow \text{Share}(m+1, k); y_{m+1} \leftarrow \text{Share}(m+1, l); \text{ret } x_{m+1} \oplus y_{m+1}$

into the inductive form of the channel

- $\text{Share-}\Sigma(m+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, K); x_{m+1} \leftarrow \text{Share}(m+1, K); \text{ret } x_\Sigma \oplus x_{m+1}$

which yields the following:

- $\text{Share-}\Sigma(m+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); y_\Sigma \leftarrow \text{Share-}\Sigma(m, l);$
 $x_{m+1} \leftarrow \text{Share}(m+1, k); y_{m+1} \leftarrow \text{Share}(m+1, l); \text{ret } (x_\Sigma \oplus y_\Sigma) \oplus (x_{m+1} \oplus y_{m+1})$

After a slight rearrangement we get the following:

- $\text{Share-}\Sigma(m+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); x_{m+1} \leftarrow \text{Share}(m+1, k);$
 $y_\Sigma \leftarrow \text{Share-}\Sigma(m, l); y_{m+1} \leftarrow \text{Share}(m+1, l); \text{ret } (x_\Sigma \oplus x_{m+1}) \oplus (y_\Sigma \oplus y_{m+1})$

But this is precisely what we get if we substitute the channels

- $\text{Share-}\Sigma(m+1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); x_{m+1} \leftarrow \text{Share}(m+1, k); \text{ret } x_\Sigma \oplus x_{m+1}$
- $\text{Share-}\Sigma(m+1, l) := y_\Sigma \leftarrow \text{Share-}\Sigma(m, l); y_{m+1} \leftarrow \text{Share}(m+1, l); \text{ret } y_\Sigma \oplus y_{m+1}$

into the closed form of the channel

- $\text{Share-}\Sigma(m+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m+1, k); y_\Sigma \leftarrow \text{Share-}\Sigma(m+1, l); \text{ret } x_\Sigma \oplus y_\Sigma$

Hence the inductive form and the closed form of the channel $\text{Share-}\Sigma(m+1, K)$ are equivalent.

Substituting the channel

- $\text{Share-}\Sigma(N, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); y_\Sigma \leftarrow \text{Share-}\Sigma(N, l); \text{ret } x_\Sigma \oplus y_\Sigma$

into the channel

- $\text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Share}(N+1, k);$
 $y_{N+1} \leftarrow \text{Share}(N+1, l); \text{ret } x_\Sigma \oplus (x_{N+1} \oplus y_{N+1})$

yields the following:

- $\text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); y_\Sigma \leftarrow \text{Share-}\Sigma(N, l);$
 $x_{N+1} \leftarrow \text{Share}(N+1, k); y_{N+1} \leftarrow \text{Share}(N+1, l); \text{ret } (x_\Sigma \oplus y_\Sigma) \oplus (x_{N+1} \oplus y_{N+1})$

At this point, we rewrite the closed form of the channels

- $\text{Share-}\Sigma(m, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); y_\Sigma \leftarrow \text{Share-}\Sigma(m, l); \text{ret } x_\Sigma \oplus y_\Sigma$ for $m \leq N$

back to their original inductive form:

- $\begin{cases} \text{Share-}\Sigma(0, K) := \text{read Share}(0, K) \\ \text{Share-}\Sigma(m+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, K); x_{m+1} \leftarrow \text{Share}(m+1, K); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \end{cases}$

After a slight rearrangement of the channel

- $\text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); y_\Sigma \leftarrow \text{Share-}\Sigma(N, l);$
 $x_{N+1} \leftarrow \text{Share}(N+1, k); y_{N+1} \leftarrow \text{Share}(N+1, l); \text{ret } (x_\Sigma \oplus y_\Sigma) \oplus (x_{N+1} \oplus y_{N+1})$

we get the following:

- $\text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x_{N+1} \leftarrow \text{Share}(N+1, k);$
 $y_\Sigma \leftarrow \text{Share-}\Sigma(N, l); y_{N+1} \leftarrow \text{Share}(N+1, l); \text{ret } (x_\Sigma \oplus x_{N+1}) \oplus (y_\Sigma \oplus y_{N+1})$

But this is precisely what we get if we substitute the channels

- $\text{Share-}\Sigma(N+1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x_{N+1} \leftarrow \text{Share}(N+1, k); \text{ret } x_\Sigma \oplus x_{N+1}$ (*inductive hypothesis*)
- $\text{Share-}\Sigma(N+1, l) := y_\Sigma \leftarrow \text{Share-}\Sigma(N, l); y_{N+1} \leftarrow \text{Share}(N+1, l); \text{ret } y_\Sigma \oplus y_{N+1}$ (*inductive hypothesis*)

into the desired inductive form of the channel

- $\text{Share-}\Sigma(N+1, K) := x \leftarrow \text{Share-}\Sigma(N+1, k); y \leftarrow \text{Share-}\Sigma(N+1, l); \text{ret } x \oplus y$

so we are done.

- In the case of an *and* gate, we start by substituting the inductive form of the channel

- $\text{Share}(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); \text{read Ctrb-}\Sigma(N+1, N+1, K)$

into the closed form of the channel

- $\text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Share}(N+1, K); \text{ret } x_\Sigma \oplus x_{N+1}$

which yields the following:

- $\text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Ctrb-}\Sigma(N+1, N+1, K); \text{ret } x_\Sigma \oplus x_{N+1}$

By canceling out two applications of $x_\Sigma \oplus -$ we can reformulate the channel $\text{Share}(N+1, K)$ as follows:

- $\text{Share}(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Ctrb-}\Sigma(N+1, N+1, K); \text{ret } x_\Sigma \oplus (x_\Sigma \oplus x_{N+1})$

The above can be expressed more concisely as

- $\text{Share}(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x \leftarrow \text{Share-}\Sigma(N+1, K); \text{ret } x_\Sigma \oplus x$

and this is the desired closed form of the channel $\text{Share}(N+1, K)$.

We can now turn our attention to the sum of shares. For each party n , the share $\text{Share}(n, K)$ is a sum of the $N+2$ contributions $\text{Ctrb}(n, 0, K), \dots, \text{Ctrb}(n, N+1, K)$. So when performing the sum of shares, we are summing up the table of contributions $\{\text{Ctrb}(i, j, K)\}_{i,j \leq N+1}$ by columns. To make this structure explicit, we introduce new internal channels

- $\text{Col}(i, j) := \text{Ctrb-}\Sigma(i, j, K)$ for $i, j \leq N+1$

that record the sum of the $j+1$ contributions $\text{Ctrb}(i, 0, K), \dots, \text{Ctrb}(i, j, K)$. Additionally, we introduce new internal channels

- $$\begin{cases} \text{Col-}\Sigma(0, j) := \text{read Col}(0, j) \\ \text{for } j \leq N+1 \\ \text{Col-}\Sigma(i+1, j) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); b_{i+1} \leftarrow \text{Col}(i+1, j); \text{ret } b_\Sigma \oplus b_{i+1} \\ \text{for } i \leq N \text{ and } j \leq N+1 \end{cases}$$

that record the sum of the $i+1$ columns $\text{Col}(0, j), \dots, \text{Col}(i, j)$ of length $j+1$.

In the presence of the channels

- $\text{Share}(m, K) := \text{read Ctrb-}\Sigma(m, N+1, K)$ for $m \leq N$

and the channels

- $\text{Col}(m, N+1) := \text{Ctrb-}\Sigma(m, N+1, K)$ for $m \leq N$

$$- \begin{cases} \text{Col-}\Sigma(0, N+1) := \text{read Col}(0, N+1) \\ \text{Col-}\Sigma(m+1, N+1) := x_\Sigma \leftarrow \text{Col-}\Sigma(m, N+1); x_{m+1} \leftarrow \text{Col}(m+1, N+1); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \end{cases}$$

we can rewrite the inductive form of the channels

$$- \begin{cases} \text{Share-}\Sigma(0, K) := \text{read Share}(0, K) \\ \text{Share-}\Sigma(m+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, K); x_{m+1} \leftarrow \text{Share}(m+1, K); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \end{cases}$$

to the following closed form:

$$- \text{Share-}\Sigma(m, K) := \text{read Col-}\Sigma(m, N+1) \text{ for } m \leq N$$

Substituting the channel $\text{Share-}\Sigma(N, K)$ into the channel

$$- \text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, K); x_{N+1} \leftarrow \text{Ctrb-}\Sigma(N+1, N+1, K); \text{ret } x_\Sigma \oplus x_{N+1}$$

thus yields the following:

$$- \text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Col-}\Sigma(N, N+1); x_{N+1} \leftarrow \text{Ctrb-}\Sigma(N+1, N+1, K); \text{ret } x_\Sigma \oplus x_{N+1}$$

At this point, we rewrite the closed form of the channels

$$- \text{Share-}\Sigma(m, K) := \text{read Col-}\Sigma(m, N+1) \text{ for } m \leq N$$

back to their original inductive form:

$$- \begin{cases} \text{Share-}\Sigma(0, K) := \text{read Share}(0, K) \\ \text{Share-}\Sigma(m+1, K) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, K); x_{m+1} \leftarrow \text{Share}(m+1, K); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \end{cases}$$

In the presence of the channels

$$\begin{aligned} & - \text{Col}(N+1, N+1) := \text{Ctrb-}\Sigma(N+1, N+1, K) \\ & - \text{Col-}\Sigma(N+1, N+1) := x_\Sigma \leftarrow \text{Col-}\Sigma(N, N+1); x_{N+1} \leftarrow \text{Col}(N+1, N+1); \text{ret } x_\Sigma \oplus x_{N+1} \end{aligned}$$

we can further write the channel

$$- \text{Share-}\Sigma(N+1, K) := x_\Sigma \leftarrow \text{Col-}\Sigma(N, N+1); x_{N+1} \leftarrow \text{Ctrb-}\Sigma(N+1, N+1, K); \text{ret } x_\Sigma \oplus x_{N+1}$$

equivalently as follows:

$$- \text{Share-}\Sigma(N+1, K) := \text{read Col-}\Sigma(N+1, N+1)$$

In the presence of the channels

$$- \begin{cases} \text{Ctrb-}\Sigma(i, 0, K) := \text{read Ctrb}(i, 0, K) \\ \text{for } i \leq N+1 \\ \text{Ctrb-}\Sigma(i, j+1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(i, j, K); b_{j+1} \leftarrow \text{Ctrb}(i, j+1, K); \text{ret } b_\Sigma \oplus b_{j+1} \\ \text{for } i \leq N+1 \text{ and } j \leq N \end{cases}$$

we can rewrite the closed form of the channels

$$- \text{Col}(i, j) := \text{Ctrb-}\Sigma(i, j, K) \text{ for } i, j \leq N+1$$

to the following inductive form:

$$- \begin{cases} \text{Col}(i, 0) := \text{read Ctrb}(i, 0, K) \\ \quad \text{for } i \leq N + 1 \\ \text{Col}(i, j + 1) := b_\Sigma \leftarrow \text{Col}(i, j); b_{j+1} \leftarrow \text{Ctrb}(i, j + 1, K); \text{ret } b_\Sigma \oplus b_{j+1} \\ \quad \text{for } i \leq N + 1 \text{ and } j \leq N \end{cases}$$

Instead of summing up the table of contributions $\{\text{Ctrb}(i, j, K)\}_{i, j \leq N+1}$ by columns, we can sum it up by rows. To this end, we introduce new internal channels

$$- \begin{cases} \text{Row}(0, j) := \text{read Ctrb}(0, j, K) \\ \quad \text{for } j \leq N + 1 \\ \text{Row}(i + 1, j) := b_\Sigma \leftarrow \text{Row}(i, j); b_{i+1} \leftarrow \text{Ctrb}(i + 1, j, K); \text{ret } b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i \leq N \text{ and } j \leq N + 1 \end{cases}$$

that record the sum of the $i + 1$ contributions $\text{Ctrb}(0, j, K), \dots, \text{Ctrb}(i, j, K)$. Additionally, we introduce new internal channels

$$- \begin{cases} \text{Row-}\Sigma(i, 0) := \text{read Row}(i, 0) \\ \quad \text{for } i \leq N + 1 \\ \text{Row-}\Sigma(i, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i, j); b_{j+1} \leftarrow \text{Row}(i, j + 1); \text{ret } b_\Sigma \oplus b_{j+1} \\ \quad \text{for } i \leq N + 1 \text{ and } j \leq N \end{cases}$$

that record the sum of the $j + 1$ rows $\text{Row}(i, 0), \dots, \text{Row}(i, j)$ 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

$$\begin{aligned} & - \begin{cases} \text{Row}(0, j) := \text{read Ctrb}(0, j, K) \\ \quad \text{for } j \leq N + 1 \\ \text{Row}(i + 1, j) := b_\Sigma \leftarrow \text{Row}(i, j); b_{i+1} \leftarrow \text{Ctrb}(i + 1, j, K); \text{ret } b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i \leq N \text{ and } j \leq N + 1 \end{cases} \\ & - \begin{cases} \text{Col}(i, 0) := \text{read Ctrb}(i, 0, K) \\ \quad \text{for } i \leq N + 1 \\ \text{Col}(i, j + 1) := b_\Sigma \leftarrow \text{Col}(i, j); b_{j+1} \leftarrow \text{Ctrb}(i, j + 1, K); \text{ret } b_\Sigma \oplus b_{j+1} \\ \quad \text{for } i \leq N + 1 \text{ and } j \leq N \end{cases} \end{aligned}$$

we can express the protocol

$$\begin{aligned} & - \begin{cases} \text{Col-}\Sigma(0, j) := \text{read Col}(0, j) \\ \quad \text{for } j \leq N + 1 \\ \text{Col-}\Sigma(i + 1, j) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); b_{i+1} \leftarrow \text{Col}(i + 1, j); \text{ret } b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i \leq N \text{ and } j \leq N + 1 \end{cases} \\ & - \begin{cases} \text{Row-}\Sigma(i, 0) := \text{read Row}(i, 0) \\ \quad \text{for } i \leq N + 1 \\ \text{Row-}\Sigma(i, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i, j); b_{j+1} \leftarrow \text{Row}(i, j + 1); \text{ret } b_\Sigma \oplus b_{j+1} \\ \quad \text{for } i \leq N + 1 \text{ and } j \leq N \end{cases} \end{aligned}$$

where the channels $\text{Col-}\Sigma(-, -)$ have an inductive form equivalently as the protocol

$$\begin{aligned} & - \text{Col-}\Sigma(i, j) := \text{read Row-}\Sigma(i, j) \text{ for } i, j \leq N + 1 \\ & - \begin{cases} \text{Row-}\Sigma(i, 0) := \text{read Row}(i, 0) \\ \quad \text{for } i \leq N + 1 \\ \text{Row-}\Sigma(i, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i, j); b_{j+1} \leftarrow \text{Row}(i, j + 1); \text{ret } b_\Sigma \oplus b_{j+1} \\ \quad \text{for } i \leq N + 1 \text{ and } j \leq N \end{cases} \end{aligned}$$

where the channels $\text{Col-}\Sigma(-, -)$ 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

- $\text{Col-}\Sigma(0, j) := \text{read Col}(0, j)$ for $j \leq N + 1$
- $\begin{cases} \text{Row-}\Sigma(0, 0) := \text{read Row}(0, 0) \\ \text{Row-}\Sigma(0, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(0, j); b_{j+1} \leftarrow \text{Row}(0, j + 1); \text{ret } b_\Sigma \oplus b_{j+1} \\ \text{for } j \leq N \end{cases}$

can be expressed equivalently as the channels below:

- $\text{Col-}\Sigma(0, j) := \text{read Row-}\Sigma(0, j)$ for $j \leq N + 1$
- $\begin{cases} \text{Row-}\Sigma(0, 0) := \text{read Row}(0, 0) \\ \text{Row-}\Sigma(0, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(0, j); b_{j+1} \leftarrow \text{Row}(0, j + 1); \text{ret } b_\Sigma \oplus b_{j+1} \\ \text{for } j \leq N \end{cases}$

We split the proof into two parts. In the first part of the base case $i = 0$, we show that the protocol

- $\text{Col-}\Sigma(0, j) := \text{read Col}(0, j)$ for $j \leq N + 1$
- $\begin{cases} \text{Row-}\Sigma(0, 0) := \text{read Row}(0, 0) \\ \text{Row-}\Sigma(0, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(0, j); b_{j+1} \leftarrow \text{Row}(0, j + 1); \text{ret } b_\Sigma \oplus b_{j+1} \\ \text{for } j \leq N \end{cases}$

where the channels $\text{Row-}\Sigma(0, -)$ have an inductive form can be expressed equivalently as the protocol

- $\text{Col-}\Sigma(0, j) := \text{read Col}(0, j)$ for $j \leq N + 1$
- $\text{Row-}\Sigma(0, j) := \text{read Col-}\Sigma(0, j)$ for $j \leq N + 1$

where the channels $\text{Row-}\Sigma(0, -)$ have a closed form.

We proceed by induction on j . In the base case $j = 0$, we need to show that the channels

- $\text{Col-}\Sigma(0, 0) := \text{read Col}(0, 0)$
- $\text{Row-}\Sigma(0, 0) := \text{read Row}(0, 0)$

can be expressed equivalently as the channels below:

- $\text{Col-}\Sigma(0, 0) := \text{read Col}(0, 0)$
- $\text{Row-}\Sigma(0, 0) := \text{read Col-}\Sigma(0, 0)$

Substituting the channel

- $\text{Row}(0, 0) := \text{read Ctrb}(0, 0, K)$

into the inductive form of the channel

- $\text{Row-}\Sigma(0, 0) := \text{read Row}(0, 0)$

yields the following:

- $\text{Row-}\Sigma(0, 0) := \text{read Ctrb}(0, 0, K)$

In the presence of the channel

- $\text{Col}(0, 0) := \text{read Ctrb}(0, 0, K)$

we can express the channel $\text{Row-}\Sigma(0, 0)$ equivalently as follows:

- $\text{Row-}\Sigma(0, 0) := \text{read Col}(0, 0)$

In the presence of the channel

- $\text{Col-}\Sigma(0, 0) := \text{read Col}(0, 0)$

we can express the channel $\text{Row-}\Sigma(0, 0)$ equivalently as follows:

- $\text{Row-}\Sigma(0, 0) := \text{read Col-}\Sigma(0, 0)$

This is precisely the desired closed form of the channel $\text{Row-}\Sigma(0, 0)$, which finishes the base case $j = 0$.

In the inductive case $j + 1$, we need to show that the channels

- $\text{Col-}\Sigma(0, j + 1) := \text{read Col}(0, j + 1)$
- $\text{Row-}\Sigma(0, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(0, j); b_{j+1} \leftarrow \text{Row}(0, j + 1); \text{ret } b_\Sigma \oplus b_{j+1}$

can be expressed equivalently as the channels below:

- $\text{Col-}\Sigma(0, j + 1) := \text{read Col}(0, j + 1)$
- $\text{Row-}\Sigma(0, j + 1) := \text{read Col-}\Sigma(0, j + 1)$

Substituting the channel

- $\text{Row-}\Sigma(0, j) := \text{read Col-}\Sigma(0, j)$ (*inductive hypothesis*)

into the inductive form of the channel

- $\text{Row-}\Sigma(0, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(0, j); b_{j+1} \leftarrow \text{Row}(0, j + 1); \text{ret } b_\Sigma \oplus b_{j+1}$

yields the following:

- $\text{Row-}\Sigma(0, j + 1) := b_\Sigma \leftarrow \text{Col-}\Sigma(0, j); b_{j+1} \leftarrow \text{Row}(0, j + 1); \text{ret } b_\Sigma \oplus b_{j+1}$

Further substituting the channels

- $\text{Col-}\Sigma(0, j) := \text{read Col}(0, j)$
- $\text{Row}(0, j + 1) := \text{read Ctrb}(0, j + 1, K)$

into the channel $\text{Row-}\Sigma(0, j + 1)$ yields the following:

- $\text{Row-}\Sigma(0, j + 1) := b_\Sigma \leftarrow \text{Col}(0, j); b_{j+1} \leftarrow \text{Ctrb}(0, j + 1, K); \text{ret } b_\Sigma \oplus b_{j+1}$

In the presence of the channel

- $\text{Col}(0, j + 1) := b_\Sigma \leftarrow \text{Col}(0, j); b_{j+1} \leftarrow \text{Ctrb}(0, j + 1, K); \text{ret } b_\Sigma \oplus b_{j+1}$

we can express the channel $\text{Row-}\Sigma(0, j + 1)$ more concisely as follows:

- $\text{Row-}\Sigma(0, j + 1) := \text{read Col}(0, j + 1)$

In the presence of the channel

- $\text{Col-}\Sigma(0, j + 1) := \text{read Col}(0, j + 1)$

we can further write the channel $\text{Row-}\Sigma(0, j + 1)$ as follows:

- $\text{Row-}\Sigma(0, j + 1) := \text{Col-}\Sigma(0, j + 1)$

This is precisely the desired closed form of the channel $\text{Row-}\Sigma(0, j + 1)$, 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

- $\text{Col-}\Sigma(0, j) := \text{read Col}(0, j)$ for $j \leq N + 1$
- $\text{Row-}\Sigma(0, j) := \text{read Col-}\Sigma(0, j)$ for $j \leq N + 1$

where the channels $\text{Col-}\Sigma(0, -)$ have an inductive form while the channels $\text{Row-}\Sigma(0, -)$ have a closed form can be expressed equivalently as the protocol

- $\text{Col-}\Sigma(0, j) := \text{read Row-}\Sigma(0, j)$ for $j \leq N + 1$
- $\begin{cases} \text{Row-}\Sigma(0, 0) := \text{read Row}(0, 0) \\ \text{Row-}\Sigma(0, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(0, j); b_{j+1} \leftarrow \text{Row}(0, j + 1); \text{ret } b_\Sigma \oplus b_{j+1} \\ \text{for } j \leq N \end{cases}$

where the channels $\text{Col-}\Sigma(0, -)$ have a closed form while the channels $\text{Row-}\Sigma(0, -)$ have an inductive form.

We proceed by induction on j . In the base case $j = 0$, we need to show that the channels below:

- $\text{Col-}\Sigma(0, 0) := \text{read Col}(0, 0)$
- $\text{Row-}\Sigma(0, 0) := \text{read Col-}\Sigma(0, 0)$

can be expressed equivalently as the channels

- $\text{Col-}\Sigma(0, 0) := \text{read Row-}\Sigma(0, 0)$
- $\text{Row-}\Sigma(0, 0) := \text{read Row}(0, 0)$

Substituting the channel

- $\text{Row}(0, 0) := \text{read Ctrb}(0, 0, K)$

into the inductive form of the channel

- $\text{Row-}\Sigma(0, 0) := \text{read Row}(0, 0)$

yields the following:

- $\text{Row-}\Sigma(0, 0) := \text{read Ctrb}(0, 0, K)$

In the presence of the channel

- $\text{Col}(0, 0) := \text{read Ctrb}(0, 0, K)$

we can express the channel $\text{Row-}\Sigma(0, 0)$ equivalently as follows:

- $\text{Row-}\Sigma(0, 0) := \text{read Col}(0, 0)$

Substituting the channel $\text{Row-}\Sigma(0, 0)$ into the closed form of the channel

- $\text{Col-}\Sigma(0, 0) := \text{read Row-}\Sigma(0, 0)$

yields the following:

- $\text{Col-}\Sigma(0, 0) := \text{read Col}(0, 0)$

This is precisely the desired inductive form of the channel $\text{Col-}\Sigma(0, 0)$, in the presence of which we can write the channel

- $\text{Row-}\Sigma(0, 0) := \text{read Col}(0, 0)$

equivalently as follows:

- $\text{Row-}\Sigma(0, 0) := \text{read Col-}\Sigma(0, 0)$

This is precisely the desired closed form of the channel $\text{Row-}\Sigma(0, 0)$, which finishes the base case $j = 0$.

In the inductive case $j + 1$, we need to show that the channels

- $\text{Col-}\Sigma(0, j + 1) := \text{read Col}(0, j + 1)$
- $\text{Row-}\Sigma(0, j + 1) := \text{read Col-}\Sigma(0, j + 1)$

can be expressed equivalently as the channels below:

- $\text{Col-}\Sigma(0, j + 1) := \text{read Row-}\Sigma(0, j + 1)$
- $\text{Row-}\Sigma(0, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(0, j); b_{j+1} \leftarrow \text{Row}(0, j + 1); \text{ret } b_\Sigma \oplus b_{j+1}$

Substituting the channel

- $\text{Row-}\Sigma(0, j) := \text{read Col-}\Sigma(0, j)$ (*inductive hypothesis*)

into the inductive form of the channel

- $\text{Row-}\Sigma(0, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(0, j); b_{j+1} \leftarrow \text{Row}(0, j + 1); \text{ret } b_\Sigma \oplus b_{j+1}$

yields the following:

- $\text{Row-}\Sigma(0, j + 1) := b_\Sigma \leftarrow \text{Col-}\Sigma(0, j); b_{j+1} \leftarrow \text{Row}(0, j + 1); \text{ret } b_\Sigma \oplus b_{j+1}$

Further substituting the channels

- $\text{Col-}\Sigma(0, j) := \text{read Col}(0, j)$ (*inductive hypothesis*)
- $\text{Row}(0, j + 1) := \text{read Ctrb}(0, j + 1, K)$

into the channel $\text{Row-}\Sigma(0, j + 1)$ yields the following:

- $\text{Row-}\Sigma(0, j + 1) := b_\Sigma \leftarrow \text{Col}(0, j); b_{j+1} \leftarrow \text{Ctrb}(0, j + 1, K); \text{ret } b_\Sigma \oplus b_{j+1}$

In the presence of the channel

- $\text{Col}(0, j + 1) := b_\Sigma \leftarrow \text{Col}(0, j); b_{j+1} \leftarrow \text{Ctrb}(0, j + 1, K); \text{ret } b_\Sigma \oplus b_{j+1}$

we can express the channel $\text{Row-}\Sigma(0, j + 1)$ more concisely as follows:

- $\text{Row-}\Sigma(0, j + 1) := \text{read Col}(0, j + 1)$

Substituting the channel $\text{Row-}\Sigma(0, j + 1)$ into the closed form of the channel

- $\text{Col-}\Sigma(0, j + 1) := \text{read Row-}\Sigma(0, j + 1)$

yields the following:

- $\text{Col-}\Sigma(0, j + 1) := \text{read Col}(0, j + 1)$

But this is precisely the desired inductive form of the channel $\text{Col-}\Sigma(0, j + 1)$, in the presence of which we can write the channel

- $\text{Row-}\Sigma(0, j + 1) := \text{read Col}(0, j + 1)$

equivalently as follows:

- $\text{Row-}\Sigma(0, j + 1) := \text{read Col-}\Sigma(0, j + 1)$

This is precisely the desired closed form of the channel $\text{Row-}\Sigma(0, j + 1)$, 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

- $\text{Col-}\Sigma(i+1, j) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); b_{i+1} \leftarrow \text{Col}(i+1, j); \text{ret } b_\Sigma \oplus b_{i+1} \text{ for } j \leq N+1$
- $\begin{cases} \text{Row-}\Sigma(i+1, 0) := \text{read Row}(i+1, 0) \\ \text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i+1, j); b_{j+1} \leftarrow \text{Row}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{j+1} \\ \text{for } j \leq N \end{cases}$

can be expressed equivalently as the channels below:

- $\text{Col-}\Sigma(i+1, j) := \text{read Row-}\Sigma(i+1, j) \text{ for } j \leq N+1$
- $\begin{cases} \text{Row-}\Sigma(i+1, 0) := \text{read Row}(i+1, 0) \\ \text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i+1, j); b_{j+1} \leftarrow \text{Row}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{j+1} \\ \text{for } j \leq N \end{cases}$

We split the proof into two parts. In the first part of the inductive case $i+1$, we show that the protocol

- $\text{Col-}\Sigma(i+1, j) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); b_{i+1} \leftarrow \text{Col}(i+1, j); \text{ret } b_\Sigma \oplus b_{i+1} \text{ for } j \leq N+1$
- $\begin{cases} \text{Row-}\Sigma(i+1, 0) := \text{read Row}(i+1, 0) \\ \text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i+1, j); b_{j+1} \leftarrow \text{Row}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{j+1} \\ \text{for } j \leq N \end{cases}$

where the channels $\text{Row-}\Sigma(i+1, -)$ have an inductive form can be expressed equivalently as the protocol

- $\text{Col-}\Sigma(i+1, j) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); b_{i+1} \leftarrow \text{Col}(i+1, j); \text{ret } b_\Sigma \oplus b_{i+1} \text{ for } j \leq N+1$
- $\text{Row-}\Sigma(i+1, j) := \text{read Col-}\Sigma(i+1, j) \text{ for } j \leq N+1$

where the channels $\text{Row-}\Sigma(i+1, -)$ have a closed form.

We proceed by induction on j . In the base case $j=0$, we need to show that the channels

- $\text{Col-}\Sigma(i+1, 0) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, 0); b_{i+1} \leftarrow \text{Col}(i+1, 0); \text{ret } b_\Sigma \oplus b_{i+1}$
- $\text{Row-}\Sigma(i+1, 0) := \text{read Row}(i+1, 0)$

can be expressed equivalently as the channels below:

- $\text{Col-}\Sigma(i+1, 0) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, 0); b_{i+1} \leftarrow \text{Col}(i+1, 0); \text{ret } b_\Sigma \oplus b_{i+1}$
- $\text{Row-}\Sigma(i+1, 0) := \text{read Col-}\Sigma(i+1, 0)$

Substituting the channel

- $\text{Row}(i+1, 0) := b_\Sigma \leftarrow \text{Row}(i, 0); b_{i+1} \leftarrow \text{Ctrb}(i+1, 0, K); \text{ret } b_\Sigma \oplus b_{i+1}$

into the inductive form of the channel

- $\text{Row-}\Sigma(i+1, 0) := \text{read Row}(i+1, 0)$

yields the following:

- $\text{Row-}\Sigma(i+1, 0) := b_\Sigma \leftarrow \text{Row}(i, 0); b_{i+1} \leftarrow \text{Ctrb}(i+1, 0, K); \text{ret } b_\Sigma \oplus b_{i+1}$

In the presence of the channels

- $\text{Row-}\Sigma(i, 0) := \text{read Row}(i, 0)$
- $\text{Col}(i+1, 0) := \text{read Ctrb}(i+1, 0, K)$

we can express the channel $\text{Row-}\Sigma(i+1, 0)$ equivalently as follows:

- $\text{Row-}\Sigma(i+1, 0) := b_\Sigma \leftarrow \text{Row-}\Sigma(i, 0); b_{i+1} \leftarrow \text{Col}(i+1, 0); \text{ret } b_\Sigma \oplus b_{i+1}$

In the presence of the channel

- $\text{Col-}\Sigma(i, 0) := \text{Row-}\Sigma(i, 0)$ (*inductive hypothesis*)

we can further write the channel $\text{Row-}\Sigma(i + 1, 0)$ as follows:

- $\text{Row-}\Sigma(i + 1, 0) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, 0); b_{i+1} \leftarrow \text{Col}(i + 1, 0); \text{ret } b_\Sigma \oplus b_{i+1}$

Finally, in the presence of the channel

- $\text{Col-}\Sigma(i + 1, 0) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, 0); b_{i+1} \leftarrow \text{Col}(i + 1, 0); \text{ret } b_\Sigma \oplus b_{i+1}$

we can write the channel $\text{Row-}\Sigma(i + 1, 0)$ equivalently as follows:

- $\text{Row-}\Sigma(i + 1, 0) := \text{read Col-}\Sigma(i + 1, 0)$

This is precisely the desired closed form of the channel $\text{Row-}\Sigma(i + 1, 0)$, which finishes the base case $j = 0$.

In the inductive case $j + 1$, we need to show that the channels

- $\text{Col-}\Sigma(i + 1, j + 1) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j + 1); b_{i+1} \leftarrow \text{Col}(i + 1, j + 1); \text{ret } b_\Sigma \oplus b_{i+1}$
- $\text{Row-}\Sigma(i + 1, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i + 1, j); b_{j+1} \leftarrow \text{Row}(i + 1, j + 1); \text{ret } b_\Sigma \oplus b_{j+1}$

can be expressed equivalently as the channels below:

- $\text{Col-}\Sigma(i + 1, j + 1) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j + 1); b_{i+1} \leftarrow \text{Col}(i + 1, j + 1); \text{ret } b_\Sigma \oplus b_{i+1}$
- $\text{Row-}\Sigma(i + 1, j + 1) := \text{read Col-}\Sigma(i + 1, j + 1)$

Substituting the channel

- $\text{Row-}\Sigma(i + 1, j) := \text{read Col-}\Sigma(i + 1, j)$ (*inductive hypothesis*)

into the inductive form of the channel

- $\text{Row-}\Sigma(i + 1, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i + 1, j); b_{j+1} \leftarrow \text{Row}(i + 1, j + 1); \text{ret } b_\Sigma \oplus b_{j+1}$

yields the following:

- $\text{Row-}\Sigma(i + 1, j + 1) := b_\Sigma \leftarrow \text{Col-}\Sigma(i + 1, j); b_{j+1} \leftarrow \text{Row}(i + 1, j + 1); \text{ret } b_\Sigma \oplus b_{j+1}$

Further substituting the channels

- $\text{Col-}\Sigma(i + 1, j) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); b_C \leftarrow \text{Col}(i + 1, j); \text{ret } b_\Sigma \oplus b_C$
- $\text{Row}(i + 1, j + 1) := b_R \leftarrow \text{Row}(i, j + 1); b \leftarrow \text{Ctrb}(i + 1, j + 1, K); \text{ret } b_R \oplus b$

into the channel $\text{Row-}\Sigma(i + 1, j + 1)$ yields the following:

- $\text{Row-}\Sigma(i + 1, j + 1) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); b_C \leftarrow \text{Col}(i + 1, j);$
 $b_R \leftarrow \text{Row}(i, j + 1); b \leftarrow \text{Ctrb}(i + 1, j + 1, K); \text{ret } (b_\Sigma \oplus b_C) \oplus (b_R \oplus b)$

After a slight rearrangement we get the following:

- $\text{Row-}\Sigma(i + 1, j + 1) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); b_R \leftarrow \text{Row}(i, j + 1);$
 $b_C \leftarrow \text{Col}(i + 1, j); b \leftarrow \text{Ctrb}(i + 1, j + 1, K); \text{ret } (b_\Sigma \oplus b_R) \oplus (b_C \oplus b)$

A final substitution of the channel

- $\text{Col-}\Sigma(i, j) := \text{read Row-}\Sigma(i, j)$ (*inductive hypothesis*)

into the channel $\text{Row-}\Sigma(i + 1, j + 1)$ yields the following:

- $\text{Row-}\Sigma(i + 1, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i, j); b_R \leftarrow \text{Row}(i, j + 1);$
 $b_C \leftarrow \text{Col}(i + 1, j); b \leftarrow \text{Ctrb}(i + 1, j + 1, K); \text{ret } (b_\Sigma \oplus b_R) \oplus (b_C \oplus b)$

In the presence of the channels

- $\text{Row-}\Sigma(i, j+1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i, j); b_R \leftarrow \text{Row}(i, j+1); \text{ret } b_\Sigma \oplus b_R$
- $\text{Col}(i+1, j+1) := b_C \leftarrow \text{Col}(i+1, j); b \leftarrow \text{Ctrb}(i+1, j+1, K); \text{ret } b_C \oplus b$

we can express the channel $\text{Row-}\Sigma(i+1, j+1)$ more concisely as follows:

- $\text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i, j+1); b_{i+1} \leftarrow \text{Col}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{i+1}$

In the presence of the channel

- $\text{Col-}\Sigma(i, j+1) := \text{read Row-}\Sigma(i, j+1)$ (*inductive hypothesis*)

we can further write the channel $\text{Row-}\Sigma(i+1, j+1)$ equivalently as follows:

- $\text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j+1); b_{i+1} \leftarrow \text{Col}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{i+1}$

Finally, in the presence of the channel

- $\text{Col-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j+1); b_{i+1} \leftarrow \text{Col}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{i+1}$

we can write the channel $\text{Row-}\Sigma(i+1, j+1)$ equivalently as follows:

- $\text{Row-}\Sigma(i+1, j+1) := \text{read Col-}\Sigma(i+1, j+1)$

This is precisely the desired closed form of the channel $\text{Row-}\Sigma(i+1, j+1)$, 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

- $\text{Col-}\Sigma(i+1, j) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); b_{i+1} \leftarrow \text{Col}(i+1, j); \text{ret } b_\Sigma \oplus b_{i+1}$ for $j \leq N+1$
- $\text{Row-}\Sigma(i+1, j) := \text{read Col-}\Sigma(i+1, j)$ for $j \leq N+1$

where the channels $\text{Col-}\Sigma(i+1, -)$ have an inductive form while the channels $\text{Row-}\Sigma(i+1, -)$ have a closed form can be expressed equivalently as the protocol

- $\text{Col-}\Sigma(i+1, j) := \text{read Row-}\Sigma(i+1, j)$ for $j \leq N+1$
- $\begin{cases} \text{Row-}\Sigma(i+1, 0) := \text{read Row}(i+1, 0) \\ \text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i+1, j); b_{j+1} \leftarrow \text{Row}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{j+1} \\ \text{for } j \leq N \end{cases}$

where the channels $\text{Col-}\Sigma(i+1, -)$ have a closed form while the channels $\text{Row-}\Sigma(i+1, -)$ have an inductive form.

We proceed by induction on j . In the base case $j=0$, we need to show that the channels

- $\text{Col-}\Sigma(i+1, 0) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, 0); b_{i+1} \leftarrow \text{Col}(i+1, 0); \text{ret } b_\Sigma \oplus b_{i+1}$
- $\text{Row-}\Sigma(i+1, 0) := \text{read Col-}\Sigma(i+1, 0)$

can be expressed equivalently as the channels below:

- $\text{Col-}\Sigma(i+1, 0) := \text{read Row-}\Sigma(i+1, 0)$
- $\text{Row-}\Sigma(i+1, 0) := \text{read Row}(i+1, 0)$

Substituting the channel

- $\text{Row}(i+1, 0) := b_\Sigma \leftarrow \text{Row}(i, 0); b_{i+1} \leftarrow \text{Ctrb}(i+1, 0, K); \text{ret } b_\Sigma \oplus b_{i+1}$

into the inductive form of the channel

- $\text{Row-}\Sigma(i+1, 0) := \text{read Row}(i+1, 0)$

yields the following:

- $\text{Row-}\Sigma(i+1, 0) := b_\Sigma \leftarrow \text{Row}(i, 0); b_{i+1} \leftarrow \text{Ctrb}(i+1, 0, K); \text{ret } b_\Sigma \oplus b_{i+1}$

In the presence of the channels

- $\text{Row-}\Sigma(i, 0) := \text{read Row}(i, 0)$
- $\text{Col}(i+1, 0) := \text{read Ctrb}(i+1, 0, K)$

we can express the channel $\text{Row-}\Sigma(i+1, 0)$ equivalently as follows:

- $\text{Row-}\Sigma(i+1, 0) := b_\Sigma \leftarrow \text{Row-}\Sigma(i, 0); b_{i+1} \leftarrow \text{Col}(i+1, 0); \text{ret } b_\Sigma \oplus b_{i+1}$

In the presence of the channel

- $\text{Col-}\Sigma(i, 0) := \text{Row-}\Sigma(i, 0)$ (*inductive hypothesis*)

we can further write the channel $\text{Row-}\Sigma(i+1, 0)$ as follows:

- $\text{Row-}\Sigma(i+1, 0) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, 0); b_{i+1} \leftarrow \text{Col}(i+1, 0); \text{ret } b_\Sigma \oplus b_{i+1}$

Substituting the channel $\text{Row-}\Sigma(i+1, 0)$ into the closed form of the channel

- $\text{Col-}\Sigma(i+1, 0) := \text{read Row-}\Sigma(i+1, 0)$

yields the following:

- $\text{Col-}\Sigma(i+1, 0) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, 0); b_{i+1} \leftarrow \text{Col}(i+1, 0); \text{ret } b_\Sigma \oplus b_{i+1}$

But this is precisely the desired inductive form of the channel $\text{Col-}\Sigma(i+1, 0)$, in the presence of which we can write the channel

- $\text{Row-}\Sigma(i+1, 0) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, 0); b_{i+1} \leftarrow \text{Col}(i+1, 0); \text{ret } b_\Sigma \oplus b_{i+1}$

equivalently as follows:

- $\text{Row-}\Sigma(i+1, 0) := \text{read Col-}\Sigma(i+1, 0)$

This is precisely the desired closed form of the channel $\text{Row-}\Sigma(i+1, 0)$, which finishes the base case $j = 0$.

In the inductive case $j+1$, we need to show that the channels

- $\text{Col-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j+1); b_{i+1} \leftarrow \text{Col}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{i+1}$
- $\text{Row-}\Sigma(i+1, j+1) := \text{read Col-}\Sigma(i+1, j+1)$

can be expressed equivalently as the channels below:

- $\text{Col-}\Sigma(i+1, j+1) := \text{read Row-}\Sigma(i+1, j+1)$
- $\text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i+1, j); b_{j+1} \leftarrow \text{Row}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{j+1}$

Substituting the channel

- $\text{Row-}\Sigma(i+1, j) := \text{read Col-}\Sigma(i+1, j)$ (*inductive hypothesis*)

into the inductive form of the channel

- $\text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i+1, j); b_{j+1} \leftarrow \text{Row}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{j+1}$

yields the following:

- $\text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Col-}\Sigma(i+1, j); b_{j+1} \leftarrow \text{Row}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{j+1}$

Further substituting the channels

- $\text{Col-}\Sigma(i+1, j) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); b_C \leftarrow \text{Col}(i+1, j); \text{ret } b_\Sigma \oplus b_C$ (*inductive hypothesis*)
- $\text{Row}(i+1, j+1) := b_R \leftarrow \text{Row}(i, j+1); b \leftarrow \text{Ctrb}(i+1, j+1, K); \text{ret } b_R \oplus b$

into the channel $\text{Row-}\Sigma(i+1, j+1)$ yields the following:

- $\text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); b_C \leftarrow \text{Col}(i+1, j);$
 $b_R \leftarrow \text{Row}(i, j+1); b \leftarrow \text{Ctrb}(i+1, j+1, K); \text{ret } (b_\Sigma \oplus b_C) \oplus (b_R \oplus b)$

After a slight rearrangement we get the following:

- $\text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); b_R \leftarrow \text{Row}(i, j+1);$
 $b_C \leftarrow \text{Col}(i+1, j); b \leftarrow \text{Ctrb}(i+1, j+1, K); \text{ret } (b_\Sigma \oplus b_R) \oplus (b_C \oplus b)$

A final substitution of the channel

- $\text{Col-}\Sigma(i, j) := \text{read Row-}\Sigma(i, j)$ (*inductive hypothesis*)

into the channel $\text{Row-}\Sigma(i+1, j+1)$ yields the following:

- $\text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i, j); b_R \leftarrow \text{Row}(i, j+1);$
 $b_C \leftarrow \text{Col}(i+1, j); b \leftarrow \text{Ctrb}(i+1, j+1, K); \text{ret } (b_\Sigma \oplus b_R) \oplus (b_C \oplus b)$

In the presence of the channels

- $\text{Row-}\Sigma(i, j+1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i, j); b_R \leftarrow \text{Row}(i, j+1); \text{ret } b_\Sigma \oplus b_R$
- $\text{Col}(i+1, j+1) := b_C \leftarrow \text{Col}(i+1, j); b \leftarrow \text{Ctrb}(i+1, j+1, K); \text{ret } b_C \oplus b$

we can express the channel $\text{Row-}\Sigma(i+1, j+1)$ more concisely as follows:

- $\text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i, j+1); b_{i+1} \leftarrow \text{Col}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{i+1}$

In the presence of the channel

- $\text{Col-}\Sigma(i, j+1) := \text{read Row-}\Sigma(i, j+1)$ (*inductive hypothesis*)

we can further write the channel $\text{Row-}\Sigma(i+1, j+1)$ equivalently as follows:

- $\text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j+1); b_{i+1} \leftarrow \text{Col}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{i+1}$

Substituting the channel $\text{Row-}\Sigma(i+1, j+1)$ into the closed form of the channel

- $\text{Col-}\Sigma(i+1, j+1) := \text{read Row-}\Sigma(i+1, j+1)$

yields the following:

- $\text{Col-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j+1); b_{i+1} \leftarrow \text{Col}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{i+1}$

But this is precisely the desired inductive form of the channel $\text{Col-}\Sigma(i+1, j+1)$, in the presence of which we can write the channel

- $\text{Row-}\Sigma(i+1, j+1) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j+1); b_{i+1} \leftarrow \text{Col}(i+1, j+1); \text{ret } b_\Sigma \oplus b_{i+1}$

equivalently as follows:

- $\text{Row-}\Sigma(i+1, j+1) := \text{read Col-}\Sigma(i+1, j+1)$

This is precisely the desired closed form of the channel $\text{Row-}\Sigma(i+1, j+1)$, 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

- $\text{Sqr}(n) := \text{read Col-}\Sigma(n, n)$ for $n \leq N + 1$

that record the sum of the $n + 1$ columns $\text{Col}(0, n), \dots, \text{Col}(n, n)$ of length $n + 1$.

In the presence of the channels $\text{Sqr}(-)$ we can write the channel

- $\text{Share-}\Sigma(N + 1, K) := \text{read Col-}\Sigma(N + 1, N + 1)$

equivalently as follows:

- $\text{Share-}\Sigma(N + 1, K) := \text{read Sqr}(N + 1)$

Our next goal is to show that in the presence of the channels

$$\begin{aligned}
 & \left\{ \begin{array}{l} \text{Row}(0, j) := \text{read Ctrb}(0, j, K) \\ \text{for } j \leq N + 1 \end{array} \right. \\
 & \left\{ \begin{array}{l} \text{Row}(i + 1, j) := b_\Sigma \leftarrow \text{Row}(i, j); \ b_{i+1} \leftarrow \text{Ctrb}(i + 1, j, K); \text{ret } b_\Sigma \oplus b_{i+1} \\ \text{for } i \leq N \text{ and } j \leq N + 1 \end{array} \right. \\
 & \left\{ \begin{array}{l} \text{Col}(i, 0) := \text{read Ctrb}(i, 0, K) \\ \text{for } i \leq N + 1 \end{array} \right. \\
 & \left\{ \begin{array}{l} \text{Col}(i, j + 1) := b_\Sigma \leftarrow \text{Col}(i, j); \ b_{j+1} \leftarrow \text{Ctrb}(i, j + 1, K); \text{ret } b_\Sigma \oplus b_{j+1} \\ \text{for } i \leq N + 1 \text{ and } j \leq N \end{array} \right.
 \end{aligned}$$

as well as the channels

$$\begin{aligned}
 & \left\{ \begin{array}{l} \text{Col-}\Sigma(0, j) := \text{read Col}(0, j) \\ \text{for } j \leq N + 1 \end{array} \right. \\
 & \left\{ \begin{array}{l} \text{Col-}\Sigma(i + 1, j) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); \ b_{i+1} \leftarrow \text{Col}(i + 1, j); \text{ret } b_\Sigma \oplus b_{i+1} \\ \text{for } i \leq N \text{ and } j \leq N + 1 \end{array} \right. \\
 & \left\{ \begin{array}{l} \text{Row-}\Sigma(i, 0) := \text{read Row}(i, 0) \\ \text{for } i \leq N + 1 \end{array} \right. \\
 & \left\{ \begin{array}{l} \text{Row-}\Sigma(i, j + 1) := b_\Sigma \leftarrow \text{Row-}\Sigma(i, j); \ b_{j+1} \leftarrow \text{Row}(i, j + 1); \text{ret } b_\Sigma \oplus b_{j+1} \\ \text{for } i \leq N + 1 \text{ and } j \leq N \end{array} \right.
 \end{aligned}$$

we can express the closed form of the channels

- $\text{Sqr}(n) := \text{read Col-}\Sigma(n, n)$ for $n \leq N + 1$

equivalently by induction:

$$\begin{aligned}
 & \left\{ \begin{array}{l} \text{Sqr}(0) := \text{read Col}(0, 0) \\ \text{Sqr}(n + 1) := b_S \leftarrow \text{Sqr}(n); \ b_R \leftarrow \text{Row}(n, n + 1); \ b_C \leftarrow \text{Col}(n + 1, n + 1); \text{ret } (b_S \oplus b_R) \oplus b_C \\ \text{for } n \leq N \end{array} \right.
 \end{aligned}$$

In the base case, substituting the channel

- $\text{Col-}\Sigma(0, 0) := \text{read Col}(0, 0)$

into the closed form of the channel

- $\text{Sqr}(0) := \text{read Col-}\Sigma(0, 0)$

yields precisely the desired inductive form:

- $\text{Sqr}(0) := \text{read Col}(0, 0)$

In the inductive case, substituting the channel

– $\text{Col-}\Sigma(n+1, n+1) := b_\Sigma \leftarrow \text{Col-}\Sigma(n, n+1); b_C \leftarrow \text{Col}(n+1, n+1); \text{ret } b_\Sigma \oplus b_C$

into the closed form of the channel

– $\text{Sqr}(n+1) := \text{read Col-}\Sigma(n+1, n+1)$

yields the following:

– $\text{Sqr}(n+1) := b_\Sigma \leftarrow \text{Col-}\Sigma(n, n+1); b_C \leftarrow \text{Col}(n+1, n+1); \text{ret } b_\Sigma \oplus b_C$

We now appeal to our earlier observation that the inductive form of the channels

–
$$\begin{cases} \text{Col-}\Sigma(0, j) := \text{read Col}(0, j) \\ \quad \text{for } j \leq N+1 \\ \text{Col-}\Sigma(i+1, j) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); b_{i+1} \leftarrow \text{Col}(i+1, j); \text{ret } b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i \leq N \text{ and } j \leq N+1 \end{cases}$$

is equivalent to the following closed form:

– $\text{Col-}\Sigma(i, j) := \text{read Row-}\Sigma(i, j) \text{ for } i, j \leq N+1$

Substituting the channel

– $\text{Col-}\Sigma(n, n+1) := \text{read Row-}\Sigma(n, n+1)$

into the channel

– $\text{Sqr}(n+1) := b_\Sigma \leftarrow \text{Col-}\Sigma(n, n+1); b_C \leftarrow \text{Col}(n+1, n+1); \text{ret } b_\Sigma \oplus b_C$

yields the following:

– $\text{Sqr}(n+1) := b_\Sigma \leftarrow \text{Row-}\Sigma(n, n+1); b_C \leftarrow \text{Col}(n+1, n+1); \text{ret } b_\Sigma \oplus b_C$

Further substituting the channel

– $\text{Row-}\Sigma(n, n+1) := b_S \leftarrow \text{Row-}\Sigma(n, n); b_R \leftarrow \text{Row}(n, n+1); \text{ret } b_S \oplus b_R$

into the channel $\text{Sqr}(n+1)$ yields the following:

– $\text{Sqr}(n+1) := b_S \leftarrow \text{Row-}\Sigma(n, n); b_R \leftarrow \text{Row}(n, n+1); b_C \leftarrow \text{Col}(n+1, n+1); \text{ret } (b_S \oplus b_R) \oplus b_C$

In the presence of the channel

– $\text{Col-}\Sigma(n, n) := \text{read Row-}\Sigma(n, n)$

we can write the channel $\text{Sqr}(n+1)$ equivalently as follows:

– $\text{Sqr}(n+1) := b_S \leftarrow \text{Col-}\Sigma(n, n); b_R \leftarrow \text{Row}(n, n+1); b_C \leftarrow \text{Col}(n+1, n+1); \text{ret } (b_S \oplus b_R) \oplus b_C$

At this point, we rewrite the closed form of the channels

– $\text{Col-}\Sigma(i, j) := \text{read Row-}\Sigma(i, j) \text{ for } i, j \leq N+1$

back to their original inductive form:

–
$$\begin{cases} \text{Col-}\Sigma(0, j) := \text{read Col}(0, j) \\ \quad \text{for } j \leq N+1 \\ \text{Col-}\Sigma(i+1, j) := b_\Sigma \leftarrow \text{Col-}\Sigma(i, j); b_{i+1} \leftarrow \text{Col}(i+1, j); \text{ret } b_\Sigma \oplus b_{i+1} \\ \quad \text{for } i \leq N \text{ and } j \leq N+1 \end{cases}$$

In the presence of the channel

- $\text{Sqr}(n) := \text{read Col-}\Sigma(n, n)$ (*inductive hypothesis*)

we can further write the channel $\text{Sqr}(n + 1)$ as follows:

- $\text{Sqr}(n + 1) := b_S \leftarrow \text{Sqr}(n); b_R \leftarrow \text{Row}(n, n + 1); b_C \leftarrow \text{Col}(n + 1, n + 1); \text{ret } (b_S \oplus b_R) \oplus b_C$

This is precisely the desired inductive form of the channel $\text{Sqr}(n + 1)$.

The channels $\text{Row-}\Sigma(-, -)$, $\text{Col-}\Sigma(-, -)$ are now unused and can be discarded.

We now recall the channels below:

- $\begin{cases} \text{Share-}\Sigma(0, k) := \text{read Share}(0, k) \\ \text{Share-}\Sigma(m + 1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); x_{m+1} \leftarrow \text{Share}(m + 1, k); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \end{cases}$
- $\begin{cases} \text{Share-}\Sigma(0, l) := \text{read Share}(0, l) \\ \text{Share-}\Sigma(m + 1, l) := y_\Sigma \leftarrow \text{Share-}\Sigma(m, l); y_{m+1} \leftarrow \text{Share}(m + 1, l); \text{ret } y_\Sigma \oplus y_{m+1} \\ \text{for } m < N \end{cases}$

We also recall the following channels:

- $\text{Share-}\Sigma(N + 1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x_{N+1} \leftarrow \text{Share}(N + 1, k); \text{ret } x_\Sigma \oplus x_{N+1}$ (*inductive hypothesis*)
- $\text{Share-}\Sigma(N + 1, l) := y_\Sigma \leftarrow \text{Share-}\Sigma(N, l); y_{N+1} \leftarrow \text{Share}(N + 1, l); \text{ret } y_\Sigma \oplus y_{N+1}$ (*inductive hypothesis*)

Altogether we thus have the following:

- $\begin{cases} \text{Share-}\Sigma(0, k) := \text{read Share}(0, k) \\ \text{Share-}\Sigma(m + 1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); x_{m+1} \leftarrow \text{Share}(m + 1, k); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m \leq N \end{cases}$
- $\begin{cases} \text{Share-}\Sigma(0, l) := \text{read Share}(0, l) \\ \text{Share-}\Sigma(m + 1, l) := y_\Sigma \leftarrow \text{Share-}\Sigma(m, l); y_{m+1} \leftarrow \text{Share}(m + 1, l); \text{ret } y_\Sigma \oplus y_{m+1} \\ \text{for } m \leq N \end{cases}$

Our final goal is to show that in the presence of the channels

- $\begin{cases} \text{Row}(0, j) := \text{read Ctrb}(0, j, K) \\ \text{for } j \leq N + 1 \\ \text{Row}(i + 1, j) := b_\Sigma \leftarrow \text{Row}(i, j); b_{i+1} \leftarrow \text{Ctrb}(i + 1, j, K); \text{ret } b_\Sigma \oplus b_{i+1} \\ \text{for } i \leq N \text{ and } j \leq N + 1 \end{cases}$
- $\begin{cases} \text{Col}(i, 0) := \text{read Ctrb}(i, 0, K) \\ \text{for } i \leq N + 1 \\ \text{Col}(i, j + 1) := b_\Sigma \leftarrow \text{Col}(i, j); b_{j+1} \leftarrow \text{Ctrb}(i, j + 1, K); \text{ret } b_\Sigma \oplus b_{j+1} \\ \text{for } i \leq N + 1 \text{ and } j \leq N \end{cases}$

as well as the channels

- $\begin{cases} \text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{samp flip} \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{cases}$
- $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus (x_m * y_n) \oplus (x_n * y_m) \text{ for } n, m \leq N + 1$

$$- \left\{ \begin{array}{l} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \quad \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \quad \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n * y_n \\ \quad \text{for } n \leq N + 1 \end{array} \right.$$

together with the channels

$$- \left\{ \begin{array}{l} \text{Share-}\Sigma(0, k) := \text{read Share}(0, k) \\ \text{Share-}\Sigma(m + 1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); x_{m+1} \leftarrow \text{Share}(m + 1, k); \text{ret } x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m \leq N \\ \text{Share-}\Sigma(0, l) := \text{read Share}(0, l) \\ \text{Share-}\Sigma(m + 1, l) := y_\Sigma \leftarrow \text{Share-}\Sigma(m, l); y_{m+1} \leftarrow \text{Share}(m + 1, l); \text{ret } y_\Sigma \oplus y_{m+1} \\ \quad \text{for } m \leq N \end{array} \right.$$

we can express the inductive form of the channels

$$- \left\{ \begin{array}{l} \text{Sqr}(0) := \text{read Col}(0, 0) \\ \text{Sqr}(n + 1) := b_S \leftarrow \text{Sqr}(n); b_R \leftarrow \text{Row}(n, n + 1); b_C \leftarrow \text{Col}(n + 1, n + 1); \text{ret } (b_S \oplus b_R) \oplus b_C \\ \quad \text{for } n \leq N \end{array} \right.$$

equivalently using the following closed form:

$$- \text{Sqr}(n) := x_\Sigma \leftarrow \text{Share-}\Sigma(n, k); y_\Sigma \leftarrow \text{Share-}\Sigma(n, l); \text{ret } x_\Sigma * y_\Sigma \text{ for } n \leq N + 1$$

In the base case $n = 0$, substituting the channel

$$- \text{Col}(0, 0) := \text{read Ctrb}(0, 0, K)$$

into the inductive form of the channel

$$- \text{Sqr}(0) := \text{read Col}(0, 0)$$

yields the following:

$$- \text{Sqr}(0) := \text{read Ctrb}(0, 0, K)$$

Further substituting the channel

$$- \text{Ctrb}(0, 0, K) := x_0 \leftarrow \text{Share}(0, k); y_0 \leftarrow \text{Share}(0, l); \text{ret } x_0 * y_0$$

into the channel $\text{Sqr}(0)$ yields the following:

$$- \text{Sqr}(0) := x_0 \leftarrow \text{Share}(0, k); y_0 \leftarrow \text{Share}(0, l); \text{ret } x_0 * y_0$$

But this is precisely what we get if we substitute the channels

$$- \text{Share-}\Sigma(0, k) := x_0 \leftarrow \text{Share}(0, k); \text{ret } x_0$$

$$- \text{Share-}\Sigma(0, l) := y_0 \leftarrow \text{Share}(0, l); \text{ret } y_0$$

into the closed form of the channel below:

$$- \text{Sqr}(0) := x_\Sigma \leftarrow \text{Share-}\Sigma(0, k); y_\Sigma \leftarrow \text{Share-}\Sigma(0, l); \text{ret } x_\Sigma * y_\Sigma$$

Hence the inductive form and the closed form of the channel $\text{Sqr}(0)$ are equivalent.

In the inductive case $n + 1$, we substitute the channel

- $\text{Col}(n+1, n+1) := b_C \leftarrow \text{Col}(n+1, n); b \leftarrow \text{Ctrb}(n+1, n+1, K); \text{ret } b_C \oplus b$

into the channel

- $\text{Sqr}(n+1) := b_S \leftarrow \text{Sqr}(n); b_R \leftarrow \text{Row}(n, n+1); b_C \leftarrow \text{Col}(n+1, n+1); \text{ret } (b_S \oplus b_R) \oplus b_C$

which yields the following:

- $\text{Sqr}(n+1) := b_S \leftarrow \text{Sqr}(n); b_R \leftarrow \text{Row}(n, n+1); b_C \leftarrow \text{Col}(n+1, n);$
 $b \leftarrow \text{Ctrb}(n+1, n+1, K); \text{ret } (b_S \oplus b_R) \oplus (b_C \oplus b)$

After a slight rearrangement we get the following:

- $\text{Sqr}(n+1) := b_S \leftarrow \text{Sqr}(n); b_R \leftarrow \text{Row}(n, n+1); b_C \leftarrow \text{Col}(n+1, n);$
 $b \leftarrow \text{Ctrb}(n+1, n+1, K); \text{ret } b_S \oplus (b_R \oplus b_C) \oplus b$

We now introduce new internal channels

- $\text{DiagRefl}(i) := b \leftarrow \text{Ctrb}(i, n+1, K); b_* \leftarrow \text{Ctrb}(n+1, i, K); \text{ret } b \oplus b_*$ for $i \leq n$
- $\text{RowCol}(i) := b_R \leftarrow \text{Row}(i, n+1); b_C \leftarrow \text{Col}(n+1, i); \text{ret } b_R \oplus b_C$ for $i \leq n$

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

- $\text{RowCol}(n) := b_R \leftarrow \text{Row}(n, n+1); b_C \leftarrow \text{Col}(n+1, n); \text{ret } b_R \oplus b_C$

we can express the channel $\text{Sqr}(n+1)$ more concisely as follows:

- $\text{Sqr}(n+1) := b_S \leftarrow \text{Sqr}(n); b_{RC} \leftarrow \text{RowCol}(n); b \leftarrow \text{Ctrb}(n+1, n+1, K); \text{ret } b_S \oplus b_{RC} \oplus b$

We now observe that we can express the closed form of the channels

- $\text{RowCol}(i) := b_R \leftarrow \text{Row}(i, n+1); b_C \leftarrow \text{Col}(n+1, i); \text{ret } b_R \oplus b_C$ for $i \leq n$

equivalently by induction:

- $$\begin{cases} \text{RowCol}(0) := \text{read DiagRefl}(0) \\ \text{RowCol}(i+1) := b_\Sigma \leftarrow \text{RowCol}(i); b_{i+1} \leftarrow \text{DiagRefl}(i+1); \text{ret } b_\Sigma \oplus b_{i+1} \\ \text{for } i < n \end{cases}$$

In the base case $i = 0$, we substitute the channel

- $\text{DiagRefl}(0) := b \leftarrow \text{Ctrb}(0, n+1, K); b_* \leftarrow \text{Ctrb}(n+1, 0, K); \text{ret } b \oplus b_*$

into the inductive form of the channel

- $\text{RowCol}(0) := \text{read DiagRefl}(0)$

which yields the following:

- $\text{RowCol}(0) := b \leftarrow \text{Ctrb}(0, n+1, K); b_* \leftarrow \text{Ctrb}(n+1, 0, K); \text{ret } b \oplus b_*$

But this is precisely what we get if we substitute the channels

- $\text{Row}(0, n+1) := b \leftarrow \text{Ctrb}(0, n+1, K); \text{ret } b$
- $\text{Col}(n+1, 0) := b_* \leftarrow \text{Ctrb}(n+1, 0, K); \text{ret } b_*$

into the closed form of the channel below:

- $\text{RowCol}(0) := b_R \leftarrow \text{Row}(0, n+1); b_C \leftarrow \text{Col}(n+1, 0); \text{ret } b_R \oplus b_C$

Hence the closed form and the inductive form of the channel $\text{RowCol}(0)$ are equivalent.

In the inductive case $i + 1$, we substitute the channels

- $\text{RowCol}(i) := b_R \leftarrow \text{Row}(i, n + 1); b_C \leftarrow \text{Col}(n + 1, i); \text{ret } b_R \oplus b_C$ (*inductive hypothesis*)
- $\text{DiagRefl}(i + 1) := b \leftarrow \text{Ctrb}(i + 1, n + 1, K); b_* \leftarrow \text{Ctrb}(n + 1, i + 1, K); \text{ret } b \oplus b_*$

into the inductive form of the channel

- $\text{RowCol}(i + 1) := b_\Sigma \leftarrow \text{RowCol}(i); b_{i+1} \leftarrow \text{DiagRefl}(i + 1); \text{ret } b_\Sigma \oplus b_{i+1}$

which yields the following:

- $\text{RowCol}(i + 1) := b_R \leftarrow \text{Row}(i, n + 1); b_C \leftarrow \text{Col}(n + 1, i);$
 $b \leftarrow \text{Ctrb}(i + 1, n + 1, K); b_* \leftarrow \text{Ctrb}(n + 1, i + 1, K); \text{ret } (b_R \oplus b_C) \oplus (b \oplus b_*)$

After a slight rearrangement we get the following:

- $\text{RowCol}(i + 1) := b_R \leftarrow \text{Row}(i, n + 1); b \leftarrow \text{Ctrb}(i + 1, n + 1, K);$
 $b_C \leftarrow \text{Col}(n + 1, i); b_* \leftarrow \text{Ctrb}(n + 1, i + 1, K); \text{ret } (b_R \oplus b) \oplus (b_C \oplus b_*)$

But this is precisely what we get if we substitute the channels

- $\text{Row}(i + 1, n + 1) := b_R \leftarrow \text{Row}(i, n + 1); b \leftarrow \text{Ctrb}(i + 1, n + 1, K); \text{ret } b_R \oplus b$
- $\text{Col}(n + 1, i + 1) := b_C \leftarrow \text{Col}(n + 1, i); b_* \leftarrow \text{Ctrb}(n + 1, i + 1, K); \text{ret } b_C \oplus b_*$

into the closed form of the channel below:

- $\text{RowCol}(i + 1) := b_R \leftarrow \text{Row}(i + 1, n + 1); b_C \leftarrow \text{Col}(n + 1, i + 1); \text{ret } b_R \oplus b_C$

Hence the closed form and the inductive form of the channel $\text{RowCol}(i + 1)$ are equivalent.

We next observe that we can express the channels

- $\text{DiagRefl}(i) := b \leftarrow \text{Ctrb}(i, n + 1, K); b_* \leftarrow \text{Ctrb}(n + 1, i, K); \text{ret } b \oplus b_*$ for $i \leq n$

equivalently as follows:

- $\text{DiagRefl}(i) := x_i \leftarrow \text{Share}(i, k); y_i \leftarrow \text{Share}(i, l); x_{n+1} \leftarrow \text{Share}(n + 1, k);$
 $y_{n+1} \leftarrow \text{Share}(n + 1, l); \text{ret } (x_i * y_{n+1}) \oplus (x_{n+1} * y_i)$ for $i \leq n$

For any party $i \leq n$, substituting the channels

- $\text{Ctrb}(i, n + 1, K) := \text{read SendBit}(i, n + 1, K)$
- $\text{Ctrb}(n + 1, i, K) := \text{read RcvdBit}(n + 1, i, K)$

into the channel

- $\text{DiagRefl}(i) := b \leftarrow \text{Ctrb}(i, n + 1, K); b_* \leftarrow \text{Ctrb}(n + 1, i, K); \text{ret } b \oplus b_*$

yields the following:

- $\text{DiagRefl}(i) := b \leftarrow \text{SendBit}(i, n + 1, K); b_* \leftarrow \text{RcvdBit}(n + 1, i, K); \text{ret } b \oplus b_*$

Further substituting the channel

- $\text{RcvdBit}(n + 1, i, K) := b \leftarrow \text{SendBit}(i, n + 1, K); x_i \leftarrow \text{Share}(i, k); y_i \leftarrow \text{Share}(i, l);$
 $x_{n+1} \leftarrow \text{Share}(n + 1, k); y_{n+1} \leftarrow \text{Share}(n + 1, l); \text{ret } b \oplus (x_i * y_{n+1}) \oplus (x_{n+1} * y_i)$

into the channel $\text{DiagRefl}(i)$ yields the following:

- $\text{DiagRefl}(i) := b \leftarrow \text{SendBit}(i, n + 1, K); x_i \leftarrow \text{Share}(i, k); y_i \leftarrow \text{Share}(i, l);$
 $x_{n+1} \leftarrow \text{Share}(n + 1, k); y_{n+1} \leftarrow \text{Share}(n + 1, l); \text{ret } b \oplus (b \oplus (x_i * y_{n+1}) \oplus (x_{n+1} * y_i))$

Canceling out the two applications of $b \oplus -$ yields the following:

- $\text{DiagRefl}(i) := b \leftarrow \text{SendBit}(i, n+1, K); x_i \leftarrow \text{Share}(i, k); y_i \leftarrow \text{Share}(i, l);$
 $x_{n+1} \leftarrow \text{Share}(n+1, k); y_{n+1} \leftarrow \text{Share}(n+1, l); \text{ret } (x_i * y_{n+1}) \oplus (x_{n+1} * y_i)$

We can drop the dependency on the unused channel

- $\text{SendBit}(i, n+1, K) := x_i \leftarrow \text{Share}(i, k); y_i \leftarrow \text{Share}(i, l); \text{samp flip}$

because it only reads from the channels $\text{Share}(i, k)$ and $\text{Share}(i, l)$, which $\text{DiagRefl}(i)$ reads from as well:

- $\text{DiagRefl}(i) := x_i \leftarrow \text{Share}(i, k); y_i \leftarrow \text{Share}(i, l); x_{n+1} \leftarrow \text{Share}(n+1, k);$
 $y_{n+1} \leftarrow \text{Share}(n+1, l); \text{ret } (x_i * y_{n+1}) \oplus (x_{n+1} * y_i)$

But this is precisely the desired form of $\text{DiagRefl}(i)$.

We now observe that we can express the inductive form of the channels

- $\begin{cases} \text{RowCol}(0) := \text{read DiagRefl}(0) \\ \text{RowCol}(i+1) := b_\Sigma \leftarrow \text{RowCol}(i); b_{i+1} \leftarrow \text{DiagRefl}(i+1); \text{ret } b_\Sigma \oplus b_{i+1} \\ \text{for } i < n \end{cases}$

equivalently using the following closed form:

- $\text{RowCol}(i) := x_\Sigma \leftarrow \text{Share-}\Sigma(i, k); y_\Sigma \leftarrow \text{Share-}\Sigma(i, l); x_{n+1} \leftarrow \text{Share}(n+1, k);$
 $y_{n+1} \leftarrow \text{Share}(n+1, l); \text{ret } (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma) \text{ for } i \leq n$

In the base case $i = 0$, we substitute the channel

- $\text{DiagRefl}(0) := x_0 \leftarrow \text{Share}(0, k); y_0 \leftarrow \text{Share}(0, l); x_{n+1} \leftarrow \text{Share}(n+1, k);$
 $y_{n+1} \leftarrow \text{Share}(n+1, l); \text{ret } (x_0 * y_{n+1}) \oplus (x_{n+1} * y_0)$

into the inductive form of the channel

- $\text{RowCol}(0) := \text{read DiagRefl}(0)$

which yields the following:

- $\text{RowCol}(0) := x_0 \leftarrow \text{Share}(0, k); y_0 \leftarrow \text{Share}(0, l); x_{n+1} \leftarrow \text{Share}(n+1, k);$
 $y_{n+1} \leftarrow \text{Share}(n+1, l); \text{ret } (x_0 * y_{n+1}) \oplus (x_{n+1} * y_0)$

But this is precisely what we get if we substitute the channels

- $\text{Share-}\Sigma(0, k) := x_0 \leftarrow \text{Share}(0, k); \text{ret } x_0$
- $\text{Share-}\Sigma(0, l) := y_0 \leftarrow \text{Share}(0, l); \text{ret } y_0$

into the closed form of the channel below:

- $\text{RowCol}(0) := x_\Sigma \leftarrow \text{Share-}\Sigma(0, k); y_\Sigma \leftarrow \text{Share-}\Sigma(0, l); x_{n+1} \leftarrow \text{Share}(n+1, k);$
 $y_{n+1} \leftarrow \text{Share}(n+1, l); \text{ret } (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma)$

Hence the inductive form and the closed form of the channel $\text{RowCol}(0)$ are equivalent.

In the inductive case $i+1$, we substitute the channels

- $\text{RowCol}(i) := x_\Sigma \leftarrow \text{Share-}\Sigma(i, k); y_\Sigma \leftarrow \text{Share-}\Sigma(i, l); x_{n+1} \leftarrow \text{Share}(n+1, k);$
 $y_{n+1} \leftarrow \text{Share}(n+1, l); \text{ret } (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma) \text{ (inductive hypothesis)}$
- $\text{DiagRefl}(i+1) := x_{i+1} \leftarrow \text{Share}(i+1, k); y_{i+1} \leftarrow \text{Share}(i+1, l);$
 $x_{n+1} \leftarrow \text{Share}(n+1, k); y_{n+1} \leftarrow \text{Share}(n+1, l); \text{ret } (x_{i+1} * y_{n+1}) \oplus (x_{n+1} * y_{i+1})$

into the inductive form of the channel

– $\text{RowCol}(i + 1) := b_\Sigma \leftarrow \text{RowCol}(i); b_{i+1} \leftarrow \text{DiagRefl}(i + 1); \text{ret } b_\Sigma \oplus b_{i+1}$

which yields the following:

– $\text{RowCol}(i + 1) := x_\Sigma \leftarrow \text{Share-}\Sigma(i, k); y_\Sigma \leftarrow \text{Share-}\Sigma(i, l); x_{n+1} \leftarrow \text{Share}(n + 1, k);$
 $y_{n+1} \leftarrow \text{Share}(n + 1, l); x_{i+1} \leftarrow \text{Share}(i + 1, k); y_{i+1} \leftarrow \text{Share}(i + 1, l);$
 $\text{ret } (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma) \oplus (x_{i+1} * y_{n+1}) \oplus (x_{n+1} * y_{i+1})$

After a slight rearrangement we get the following:

– $\text{RowCol}(i + 1) := x_\Sigma \leftarrow \text{Share-}\Sigma(i, k); x_{i+1} \leftarrow \text{Share}(i + 1, k); y_\Sigma \leftarrow \text{Share-}\Sigma(i, l);$
 $y_{i+1} \leftarrow \text{Share}(i + 1, l); x_{n+1} \leftarrow \text{Share}(n + 1, k); y_{n+1} \leftarrow \text{Share}(n + 1, l);$
 $\text{ret } (x_\Sigma * y_{n+1}) \oplus (x_{i+1} * y_{n+1}) \oplus (x_{n+1} * y_\Sigma) \oplus (x_{n+1} * y_{i+1})$

The above is equivalent to the following:

– $\text{RowCol}(i + 1) := x_\Sigma \leftarrow \text{Share-}\Sigma(i, k); x_{i+1} \leftarrow \text{Share}(i + 1, k); y_\Sigma \leftarrow \text{Share-}\Sigma(i, l);$
 $y_{i+1} \leftarrow \text{Share}(i + 1, l); x_{n+1} \leftarrow \text{Share}(n + 1, k); y_{n+1} \leftarrow \text{Share}(n + 1, l);$
 $\text{ret } ((x_\Sigma \oplus x_{i+1}) * y_{n+1}) \oplus (x_{n+1} * (y_\Sigma \oplus y_{i+1}))$

But this is precisely what we get if we substitute the channels

– $\text{Share-}\Sigma(i + 1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(i, k); x_{i+1} \leftarrow \text{Share}(i + 1, k); \text{ret } x_\Sigma \oplus x_{i+1}$
– $\text{Share-}\Sigma(i + 1, l) := y_\Sigma \leftarrow \text{Share-}\Sigma(i, l); y_{i+1} \leftarrow \text{Share}(i + 1, l); \text{ret } y_\Sigma \oplus y_{i+1}$

into the closed form of the channel below:

– $\text{RowCol}(i + 1) := x_\Sigma \leftarrow \text{Share-}\Sigma(i + 1, k); y_\Sigma \leftarrow \text{Share-}\Sigma(i + 1, l);$
 $x_{n+1} \leftarrow \text{Share}(n + 1, k); y_{n+1} \leftarrow \text{Share}(n + 1, l); \text{ret } (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma)$

Hence the inductive form and the closed form of the channel $\text{RowCol}(i + 1)$ are equivalent.

The channels $\text{DiagRefl}(-)$ are now unused and can be discarded.

We now substitute the channels

– $\text{Sqr}(n) := x_\Sigma \leftarrow \text{Share-}\Sigma(n, k); y_\Sigma \leftarrow \text{Share-}\Sigma(n, l); \text{ret } x_\Sigma * y_\Sigma$ (*inductive hypothesis*)
– $\text{RowCol}(n) := x_\Sigma \leftarrow \text{Share-}\Sigma(n, k); y_\Sigma \leftarrow \text{Share-}\Sigma(n, l); x_{n+1} \leftarrow \text{Share}(n + 1, k);$
 $y_{n+1} \leftarrow \text{Share}(n + 1, l); \text{ret } (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma)$
– $\text{Ctrb}(n + 1, n + 1, K) := x_{n+1} \leftarrow \text{Share}(n + 1, k); y_{n+1} \leftarrow \text{Share}(n + 1, l); \text{ret } x_{n+1} * y_{n+1}$

into the channel

– $\text{Sqr}(n + 1) := b_S \leftarrow \text{Sqr}(n); b_{RC} \leftarrow \text{RowCol}(n); b \leftarrow \text{Ctrb}(n + 1, n + 1, K); \text{ret } b_S \oplus b_{RC} \oplus b$

which yields the following:

– $\text{Sqr}(n + 1) := x_\Sigma \leftarrow \text{Share-}\Sigma(n, k); y_\Sigma \leftarrow \text{Share-}\Sigma(n, l); x_{n+1} \leftarrow \text{Share}(n + 1, k);$
 $y_{n+1} \leftarrow \text{Share}(n + 1, l); \text{ret } (x_\Sigma * y_\Sigma) \oplus (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma) \oplus (x_{n+1} * y_{n+1})$

The channels $\text{RowCol}(-)$ are now unused and can be discarded.

After a slight rearrangement of the channel $\text{Sqr}(n + 1)$ we get the following:

– $\text{Sqr}(n + 1) := x_\Sigma \leftarrow \text{Share-}\Sigma(n, k); x_{n+1} \leftarrow \text{Share}(n + 1, k); y_\Sigma \leftarrow \text{Share-}\Sigma(n, l);$
 $y_{n+1} \leftarrow \text{Share}(n + 1, l); \text{ret } (x_\Sigma * y_\Sigma) \oplus (x_\Sigma * y_{n+1}) \oplus (x_{n+1} * y_\Sigma) \oplus (x_{n+1} * y_{n+1})$

The above is equivalent to the following:

– $\text{Sqr}(n + 1) := x_\Sigma \leftarrow \text{Share-}\Sigma(n, k); x_{n+1} \leftarrow \text{Share}(n + 1, k); y_\Sigma \leftarrow \text{Share-}\Sigma(n, l);$
 $y_{n+1} \leftarrow \text{Share}(n + 1, l); \text{ret } (x_\Sigma \oplus x_{n+1}) * (y_\Sigma \oplus y_{n+1})$

But this is precisely what we get if we substitute the channels

- $\text{Share-}\Sigma(n+1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(n, k); x_{n+1} \leftarrow \text{Share}(n+1, k); \text{ret } x_\Sigma \oplus x_{n+1}$
- $\text{Share-}\Sigma(n+1, l) := y_\Sigma \leftarrow \text{Share-}\Sigma(n, l); y_{n+1} \leftarrow \text{Share}(n+1, l); \text{ret } y_\Sigma \oplus y_{n+1}$

into the closed form of the channel below:

- $\text{Sqr}(n+1) := x_\Sigma \leftarrow \text{Share-}\Sigma(n+1, k); y_\Sigma \leftarrow \text{Share-}\Sigma(n+1, l); \text{ret } x_\Sigma * y_\Sigma$

Hence the inductive form and the closed form of the channel $\text{Sqr}(n+1)$ are equivalent.

The channels $\text{Row}(-, -)$, $\text{Col}(-, -)$ are now unused and can be discarded.

Finally, we substitute the channel

- $\text{Sqr}(N+1) := x \leftarrow \text{Share-}\Sigma(N+1, k); y \leftarrow \text{Share-}\Sigma(N+1, l); \text{ret } x * y$

into the channel

- $\text{Share-}\Sigma(N+1, K) := \text{read Sqr}(N+1)$

which yields the following:

- $\text{Share-}\Sigma(N+1, K) := x \leftarrow \text{Share-}\Sigma(N+1, k); y \leftarrow \text{Share-}\Sigma(N+1, l); \text{ret } x * y$

But this is precisely the desired inductive form of the channel $\text{Share-}\Sigma(N+1, K)$.

The channels $\text{Sqr}(-, -)$ are now unused and can be discarded.

We have now shown that summing up the respective shares $\bigoplus_{i \leq N+1} x_i$ of each party on a given wire yields the actual value x carried by the wire. Currently the computation is performed by the channels $\text{Share-}\Sigma(N+1, -)$ but we can extract it out into a separate protocol $\text{Wires}(C, K)$ as defined in the ideal functionality. Specifically, we introduce new internal channels

- $\text{Wire}(k) := \text{read Share-}\Sigma(N+1, k)$ for $k < K$

and observe that together with the protocol $\text{Shares}(C, K)$ we can express them equivalently as the channels

- $\text{Share-}\Sigma(N+1, k) := \text{read Wire}(k)$ for $k < K$

together with the aforementioned protocol $\text{Wires}(C, K)$ and the following new form of the protocol $\text{Shares}(C, K)$:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(p, i), K+1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N+1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N+1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N+1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N+1$
 - $\text{Share}(n, K) := \text{read InShare}(n, p, i)$ for $n \leq N$
- $\text{Shares}(C; \text{not-gate}(k), K+1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N+1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N+1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N+1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N+1$
 - $\text{Share}(n, K) := \text{read Share}(n, k)$ for $n \leq N$
- $\text{Shares}(C; \text{xor-gate}(k, l), K+1)$ is the composition of $\text{Shares}(C, K)$ with the protocol

- $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
- $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
- $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
- $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
- $\text{Share}(n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n \oplus y_n$ for $n \leq N$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\begin{cases} \text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{samp flip} \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{cases}$
 - $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l);$
 $x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n, m \leq N + 1$
 - $\begin{cases} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n * y_n \\ \text{for } n \leq N + 1 \end{cases}$
 - $\begin{cases} \text{Ctrb-}\Sigma(n, 0, K) := \text{read Ctrb}(n, 0, K) \\ \text{for } n \leq N + 1 \\ \text{Ctrb-}\Sigma(n, m + 1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(n, m, K); b \leftarrow \text{Ctrb}(n, m + 1, K); \text{ret } b_\Sigma \oplus b \\ \text{for } n \leq N + 1 \text{ and } m \leq N \end{cases}$
 - $\text{Share}(n, K) := \text{read Ctrb-}\Sigma(n, N + 1, K)$ for $n \leq N$

10.4.7 Eliminating Shares of Party $N + 1$

We start by eliminating the channels $\text{InShare-}\$(N + 1, -, -)$, $\text{InShare}(N + 1, -, -)$ from the initial part of the protocol. If party n is semi-honest, then for any input $i < I_n$ the channels

- $\text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(N + 1, n, i)$
- $\text{SendInShare}(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(N + 1, n, i)$

read from the channel

- $\text{InShare-}\$(N + 1, n, i) := x_\Sigma \leftarrow \text{InShare-}\Sigma(N, n, i); x \leftarrow \text{In}(n, i); \text{ret } x_\Sigma \oplus x$

so we can substitute:

- $\text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{InShare-}\Sigma(N, n, i); x \leftarrow \text{In}(n, i); \text{ret } x_\Sigma \oplus x$
- $\text{SendInShare}(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{InShare-}\Sigma(N, n, i); x \leftarrow \text{In}(n, i); \text{ret } x_\Sigma \oplus x$

We thus get the following for channels $\text{InShare-}\$(N + 1, -, -)_{\text{adv}}^{\text{party}(-)}$ and $\text{SendInShare}(N + 1, -, -)_{\text{adv}}^{\text{party}(-)}$:

- $\begin{cases} \text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{InShare-}\Sigma(N, n, i); x \leftarrow \text{In}(n, i); \text{ret } x_\Sigma \oplus x \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $$\begin{cases} \text{SendInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{InShare-}\$(N, n, i); x \leftarrow \text{In}(n, i); \text{ret } x_\Sigma \oplus x \\ \text{for } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{SendInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$$

The leakages

- $$\text{RcvdInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(N+1)} := \text{read RcvdInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(N+1)} \text{ for } n \leq N+1 \text{ and } i < I_n$$

are vacuous since party $N+1$ is by assumption honest. Finally, substituting the channels

- $$\text{InShare-}\$(N+1, n, i) := x_\Sigma \leftarrow \text{InShare-}\$(N, n, i); x \leftarrow \text{In}(n, i); \text{ret } x_\Sigma \oplus x \text{ for } n \leq N+1 \text{ and } i < I_n$$

into the channels

- $$\text{InShare}(N+1, n, i) := \text{read InShare-}\$(N+1, n, i) \text{ for } n \leq N+1 \text{ and } i < I_n$$

yields the following:

- $$\text{InShare}(N+1, n, i) := x_\Sigma \leftarrow \text{InShare-}\$(N, n, i); x \leftarrow \text{In}(n, i); \text{ret } x_\Sigma \oplus x \text{ for } n \leq N+1 \text{ and } i < I_n$$

At this point, the internal channels $\text{InShare-}\$(N+1, -, -)$ are unused and can be eliminated.

The leakages

- $$\text{InShare}(N+1, n, i)_{\text{adv}}^{\text{party}(N+1)} := \text{read InShare}(N+1, n, i)_{\text{adv}}^{\text{party}(N+1)} \text{ for } n \leq N+1 \text{ and } i < I_n$$

are again vacuous since party $N+1$ is by assumption honest. The top-level internal channels $\text{InShare}(N+1, -, -)$ 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:

- $$\begin{cases} \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read In}(n, i) \\ \text{for } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read In}(n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$$
- $$\begin{cases} \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := x \leftarrow \text{In}(n, i); \text{ret } \checkmark \\ \text{for } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$$
- $$\text{InShare-}\$(m, n, i) := x \leftarrow \text{In}(n, i); \text{samp flip for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n$$
- $$\begin{cases} \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i) \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$$
- $$\begin{cases} \text{InShare-}\$(0, n, i) := \text{read InShare-}\$(0, n, i) \\ \text{for } n \leq N+1 \text{ and } i < I_n \\ \text{InShare-}\$(m+1, n, i) := x_\Sigma \leftarrow \text{InShare-}\$(m, n, i); x_{m+1} \leftarrow \text{InShare-}\$(m+1, n, i); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \text{ and } n \leq N+1 \text{ and } i < I_n \end{cases}$$
- $$\begin{cases} \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$$

- $\left\{ \begin{array}{l} \text{InShare-}\$(N+1, n, i)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i); \text{ret } x_{\Sigma} \oplus x \\ \text{for } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{InShare-}\$(N+1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(N+1, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i) \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{SendInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i); \text{ret } x_{\Sigma} \oplus x \\ \text{for } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{SendInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read InShare-}\$(m, n, i) \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{array} \right.$
- $\text{RcvdInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(N+1)} := \text{read RcvdInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(N+1)}$ for $n \leq N+1$ and $i < I_n$
- $\text{InShare}(m, n, i) := \text{read InShare-}\(m, n, i) for $m \leq N$ and $n \leq N+1$ and $i < I_n$
- $\left\{ \begin{array}{l} \text{InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read InShare}(m, n, i) \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{array} \right.$
- $\text{InShare}(N+1, n, i)_{\text{adv}}^{\text{party}(N+1)} := \text{read InShare}(N+1, n, i)_{\text{adv}}^{\text{party}(N+1)}$ for $n \leq N+1$ and $i < I_n$

This is followed by the hiding of the channels

- $\text{InShare-}\$(m, n, i)$ for $m \leq N$ and $n \leq N+1$ and $i < I_n$,
- $\text{InShare-}\$-\Sigma(m, n, i)$ for $m \leq N$ and $n \leq N+1$ and $i < I_n$.

We now eliminate the channels $\text{SendBit}(N+1, -, -)$, $\text{RcvdBit}(N+1, -, -)$, $\text{Ctrb}(N+1, -, -)$, $\text{Ctrb-}\Sigma(N+1, -, -)$, $\text{Share}(N+1, -)$ from the inductive part of the real protocol. To this end, we first revisit the protocol $\text{Adv}(C, K)$. In the case of *input*-, *not*-, and *xor* gates, the leakage

- $\text{Share}(N+1, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Share}(N+1, K)_{\text{adv}}^{\text{party}(N+1)}$

is vacuous since party $N+1$ is by assumption honest. In the case of an *and* gate, the leakages

- $\text{SendBit}(N+1, m, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read SendBit}(N+1, m, K)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N+1$
- $\text{RcvdBit}(N+1, m, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read RcvdBit}(N+1, m, K)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N+1$
- $\text{Ctrb}(N+1, m, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Ctrb}(N+1, m, K)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N+1$
- $\text{Ctrb-}\Sigma(N+1, m, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Ctrb-}\Sigma(N+1, m, K)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N+1$

- $\text{Share}(N+1, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Share}(N+1, K)_{\text{adv}}^{\text{party}(N+1)}$

are likewise vacuous since party $N+1$ is by assumption honest, and so are the leakages below:

- $\text{OTMsg}_0(N+1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(N+1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N+1$
- $\text{OTMsg}_1(N+1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(N+1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N+1$
- $\text{OTMsg}_2(N+1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(N+1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N+1$
- $\text{OTMsg}_2(N+1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(N+1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N+1$

On the other hand, the leakages

- $\text{OTMsgRcvd}_0(N+1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(N+1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N+1$
- $\text{OTMsgRcvd}_1(N+1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(N+1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N+1$
- $\text{OTMsgRcvd}_2(N+1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(N+1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N+1$
- $\text{OTMsgRcvd}_3(N+1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(N+1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N+1$

are vacuous because $N+1 \geq m$ for each party m , so party $N+1$ cannot ever function as a sender. The leakages

- $\text{OTChc}_0(n, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, N+1, K)_{\text{adv}}^{\text{ot}}$ for $n \leq N+1$
- $\text{OTChc}_1(n, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, N+1, K)_{\text{adv}}^{\text{ot}}$ for $n \leq N+1$

are again vacuous because party $N+1$ is honest, and so are the leakages below:

- $\text{OTOOut}(n, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTOOut}(n, N+1, K)_{\text{adv}}^{\text{ot}}$ for $n \leq N+1$

The latest version of the protocol $\text{Adv}(C, K)$ is therefore as follows:

- $\text{Adv}(\epsilon, 0)$ is the protocol 0
- $\text{Adv}(C; \text{input-gate}(p, i), K+1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N+1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N+1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N+1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N+1$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ if } n \text{ honest} \end{cases}$
 - $\text{Share}(N+1, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Share}(N+1, K)_{\text{adv}}^{\text{party}(N+1)}$
 - $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N+1$
 - $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N+1$
 - $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N+1$
 - $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N+1$
 - $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N+1$
 - $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N+1$
 - $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N+1$

- $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Adv}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ if } n \text{ honest} \end{cases}$
 - $\text{Share}(N + 1, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Share}(N + 1, K)_{\text{adv}}^{\text{party}(N+1)}$
 - $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Adv}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ if } n \text{ honest} \end{cases}$

- $\text{Share}(N + 1, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Share}(N + 1, K)_{\text{adv}}^{\text{party}(N+1)}$
- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Adv}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Adv}(C, K)$ with the protocol
 - $\left\{ \begin{array}{l} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K) \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest} \end{array} \right.$
 - $\left\{ \begin{array}{l} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.$
 - $\text{SendBit}(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read SendBit}(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N + 1$
 - $\left\{ \begin{array}{l} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K) \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest} \end{array} \right.$
 - $\left\{ \begin{array}{l} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.$
 - $\text{RcvdBit}(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read RcvdBit}(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N + 1$
 - $\left\{ \begin{array}{l} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K) \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest} \end{array} \right.$
 - $\left\{ \begin{array}{l} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.$
 - $\text{Ctrb}(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Ctrb}(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N + 1$
 - $\left\{ \begin{array}{l} \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K) \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest} \end{array} \right.$
 - $\left\{ \begin{array}{l} \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest} \end{array} \right.$
 - $\text{Ctrb-}\Sigma(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Ctrb-}\Sigma(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N + 1$
 - $\left\{ \begin{array}{l} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N \text{ if } n \text{ semi-honest} \end{array} \right.$
 - $\left\{ \begin{array}{l} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ if } n \text{ honest} \end{array} \right.$

[illegible]

- $\text{OTMsgRcvd}_3(N+1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(N+1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N+1$
- $\begin{cases} \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, k) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ semi-honest and } n < m \\ \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ honest or } n \geq m \end{cases}$
- $\text{OTChc}_0(n, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, N+1, K)_{\text{adv}}^{\text{ot}}$ for $n \leq N+1$
- $\begin{cases} \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, l) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ semi-honest and } n < m \\ \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ honest or } n \geq m \end{cases}$
- $\text{OTChc}_1(n, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, N+1, K)_{\text{adv}}^{\text{ot}}$ for $n \leq N+1$
- $\begin{cases} \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := x_m \leftarrow \text{Share}(m, k); \text{ ret } \checkmark \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ semi-honest or } n \geq m \\ \text{OTChcRcvd}_0(n, N+1, K)_{\text{adv}}^{\text{ot}} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); \text{ ret } \checkmark \\ \text{for } n \leq N \\ \text{OTChcRcvd}_0(N+1, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(N+1, N+1, K)_{\text{adv}}^{\text{ot}} \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := y_m \leftarrow \text{Share}(m, l); \text{ ret } \checkmark \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ honest and } n < m \\ \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ semi-honest or } n \geq m \\ \text{OTChcRcvd}_1(n, N+1, K)_{\text{adv}}^{\text{ot}} := y_\Sigma \leftarrow \text{Share-}\Sigma(N, l); \text{ ret } \checkmark \\ \text{for } n \leq N \\ \text{OTChcRcvd}_1(N+1, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(N+1, N+1, K)_{\text{adv}}^{\text{ot}} \end{cases}$
- $\begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read RcvdBit}(m, n, K) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ semi-honest} \\ \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ honest} \end{cases}$
- $\text{OTOut}(n, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, N+1, K)_{\text{adv}}^{\text{ot}}$ for $n \leq N+1$

We now revisit the protocol $\text{Shares}(C, K)$, the case of an *and* gate. For any party n the channel

- $\text{RcvdBit}(n, N+1, K) := b \leftarrow \text{SendBit}(N+1, n, K); x_{N+1} \leftarrow \text{Share}(N+1, k);$
 $y_{N+1} \leftarrow \text{Share}(N+1, l); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ ret } b \oplus (x_{N+1} * y_n) \oplus (x_n * y_{N+1})$

reads from the divergent channel

- $\text{SendBit}(N+1, n, K) := \text{read SendBit}(N+1, n, K)$

and so we may equivalently write the following:

- $\text{RcvdBit}(n, N+1, K) := \text{read RcvdBit}(n, N+1, K)$

The simplified definition for channels $\text{RcvdBit}(-, N+1, K)$ is thus as follows:

- $\text{RcvdBit}(n, N+1, K) := \text{read RcvdBit}(n, N+1, K)$ for $n \leq N+1$

The latest version of the protocol $\text{Shares}(C, K)$ is therefore as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := \text{read InShare}(n, p, i)$ for $n \leq N$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := \text{read Share}(n, k)$ for $n \leq N$
- $\text{Shares}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n \oplus y_n$ for $n \leq N$
- $\text{Shares}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\left\{ \begin{array}{l} \text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{samp flip} \\ \text{for } n, m \leq N + 1 \text{ if } n < m \end{array} \right.$
 - $\left\{ \begin{array}{l} \text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n \geq m \end{array} \right.$
 - $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l);$
 $x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n \leq N + 1$ and $m \leq N$
 - $\text{RcvdBit}(n, N + 1, K) := \text{read RcvdBit}(n, N + 1, K)$ for $n \leq N + 1$
 - $\left\{ \begin{array}{l} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } n < m \\ \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \text{for } n, m \leq N + 1 \text{ if } m < n \\ \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n * y_n \\ \text{for } n \leq N + 1 \end{array} \right.$
 - $\left\{ \begin{array}{l} \text{Ctrb-}\Sigma(n, 0, K) := \text{read Ctrb}(n, 0, K) \\ \text{for } n \leq N + 1 \\ \text{Ctrb-}\Sigma(n, m + 1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(n, m, K); b \leftarrow \text{Ctrb}(n, m + 1, K); \text{ret } b_\Sigma \oplus b \\ \text{for } n \leq N + 1 \text{ and } m \leq N \end{array} \right.$
 - $\text{Share}(n, K) := \text{read Ctrb-}\Sigma(n, N + 1, K)$ for $n \leq N$

At this point, we can extract all computation carried out by party $N + 1$ into a separate protocol $\text{Ctrbs}_{N+1}(C, K)$:

- $\text{Ctrbs}_{N+1}(\epsilon, 0)$ is the protocol 0
- $\text{Ctrbs}_{N+1}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Ctrbs}_{N+1}(C, K)$ with the protocol
 - $\text{SendBit}(N + 1, m, K) := \text{read SendBit}(N + 1, m, K)$ for $m \leq N + 1$
 - $\text{RcvdBit}(N + 1, m, K) := \text{read RcvdBit}(N + 1, m, K)$ for $m \leq N + 1$
 - $\text{Ctrb}(N + 1, m, K) := \text{read Ctrb}(N + 1, m, K)$ for $m \leq N + 1$
 - $\text{Ctrb-}\Sigma(N + 1, m, K) := \text{read Ctrb-}\Sigma(N + 1, m, K)$ for $m \leq N + 1$
- $\text{Ctrbs}_{N+1}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Ctrbs}_{N+1}(C, K)$ with the protocol
 - $\text{SendBit}(N + 1, m, K) := \text{read SendBit}(N + 1, m, K)$ for $m \leq N + 1$
 - $\text{RcvdBit}(N + 1, m, K) := \text{read RcvdBit}(N + 1, m, K)$ for $m \leq N + 1$
 - $\text{Ctrb}(N + 1, m, K) := \text{read Ctrb}(N + 1, m, K)$ for $m \leq N + 1$
 - $\text{Ctrb-}\Sigma(N + 1, m, K) := \text{read Ctrb-}\Sigma(N + 1, m, K)$ for $m \leq N + 1$
- $\text{Ctrbs}_{N+1}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Ctrbs}_{N+1}(C, K)$ with the protocol
 - $\text{SendBit}(N + 1, m, K) := \text{read SendBit}(N + 1, m, K)$ for $m \leq N + 1$
 - $\text{RcvdBit}(N + 1, m, K) := \text{read RcvdBit}(N + 1, m, K)$ for $m \leq N + 1$
 - $\text{Ctrb}(N + 1, m, K) := \text{read Ctrb}(N + 1, m, K)$ for $m \leq N + 1$
 - $\text{Ctrb-}\Sigma(N + 1, m, K) := \text{read Ctrb-}\Sigma(N + 1, m, K)$ for $m \leq N + 1$
- $\text{Ctrbs}_{N+1}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Ctrbs}_{N+1}(C, K)$ with the protocol
 - $\text{SendBit}(N + 1, m, K) := \text{read SendBit}(N + 1, m, K)$ for $m \leq N + 1$
 - $\text{RcvdBit}(N + 1, m, K) := b \leftarrow \text{SendBit}(m, N + 1, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l);$
 $x_{N+1} \leftarrow \text{Share}(N + 1, k); y_{N+1} \leftarrow \text{Share}(N + 1, l); \text{ret } b \oplus (x_m * y_{N+1}) \oplus (x_{N+1} * y_m)$ for $m \leq N$
 - $\text{RcvdBit}(N + 1, N + 1, K) := \text{read RcvdBit}(N + 1, N + 1, K)$
 - $\begin{cases} \text{Ctrb}(N + 1, m, K) := \text{read RcvdBit}(N + 1, m, K) \\ \text{for } m \leq N \\ \text{Ctrb}(N + 1, N + 1, K) := x_{N+1} \leftarrow \text{Share}(N + 1, k); y_{N+1} \leftarrow \text{Share}(N + 1, l); \text{ret } x_{N+1} * y_{N+1} \end{cases}$
 - $\begin{cases} \text{Ctrb-}\Sigma(N + 1, 0, K) := \text{read Ctrb}(N + 1, 0, K) \\ \text{Ctrb-}\Sigma(N + 1, m + 1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(N + 1, m, K); b \leftarrow \text{Ctrb}(N + 1, m + 1, K); \text{ret } b_\Sigma \oplus b \\ \text{for } m \leq N \end{cases}$

After the extraction, the protocol $\text{Shares}(C, K)$ is left looking as follows:

- $\text{Shares}(\epsilon, 0)$ is the protocol 0
- $\text{Shares}(C; \text{input-gate}(p, i), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{Share}(n, K) := \text{read InShare}(n, p, i)$ for $n \leq N$
- $\text{Shares}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol

- $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
- $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
- $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
- $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
- $\text{Share}(n, K) := \text{read Share}(n, k)$ for $n \leq N$
- $\text{Shares}(C; \text{ xor-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{Share}(n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n \oplus y_n$ for $n \leq N$
- $\text{Shares}(C; \text{ and-gate}(k, l), K + 1)$ is the composition of $\text{Shares}(C, K)$ with the protocol
 - $\left\{ \begin{array}{l} \text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{samp flip} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n < m \\ \text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \geq m \end{array} \right.$
 - $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l);$
 $x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n, m \leq N$
 - $\text{RcvdBit}(n, N + 1, K) := \text{read RcvdBit}(n, N + 1, K)$ for $n \leq N$
 - $\left\{ \begin{array}{l} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n < m \\ \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } m < n \\ \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n * y_n \\ \text{for } n \leq N \end{array} \right.$
 - $\left\{ \begin{array}{l} \text{Ctrb-}\Sigma(n, 0, K) := \text{read Ctrb}(n, 0, K) \\ \text{for } n \leq N \\ \text{Ctrb-}\Sigma(n, m + 1, K) := b_\Sigma \leftarrow \text{Ctrb-}\Sigma(n, m, K); b \leftarrow \text{Ctrb}(n, m + 1, K); \text{ret } b_\Sigma \oplus b \\ \text{for } n, m \leq N \end{array} \right.$
 - $\text{Share}(n, K) := \text{read Ctrb-}\Sigma(n, N + 1, K)$ for $n \leq N$

None of the channels defined by $\text{Ctrbs}_{N+1}(C, K)$ are utilized anywhere outside of $\text{Ctrbs}_{N+1}(C, K)$ and as such we may discard this protocol fragment entirely. This in particular eliminates all references to the channels $\text{Share}(N + 1, -)$ from the inductive part of the protocol. To summarize, the inductive part of the real protocol now consists of the protocols $\text{Shares}(C, K)$ and $\text{Adv}(C, K)$, followed by the hiding of the channels

- $\text{SendBit}(n, m, k)$ for $n \leq N$ and $m \leq N + 1$ and $k < K$,
- $\text{RcvdBit}(n, m, k)$ for $n \leq N$ and $m \leq N + 1$ and $k < K$,
- $\text{Ctrb}(n, m, k)$ for $n \leq N$ and $m \leq N + 1$ and $k < K$,
- $\text{Ctrb-}\Sigma(n, m, k)$ for $n \leq N$ and $m \leq N + 1$ and $k < K$.

We recall that on the top level we also have the protocol $\text{Wires}(C, K)$ and the channels

- $\begin{cases} \text{Share-}\Sigma(0, k) := \text{read Share}(0, k) \\ \text{for } k < K \\ \text{Share-}\Sigma(m+1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); x_{m+1} \leftarrow \text{Share}(m+1, k); \text{ret } x_\Sigma \oplus x_{m+1} \\ \text{for } m < N \text{ and } k < K \end{cases}$

together with the channels below:

- $\text{Share-}\Sigma(N+1, k) := \text{read Wire}(k) \text{ for } k < K$
- $\text{Share}(N+1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Share-}\Sigma(N+1, k); \text{ret } x_\Sigma \oplus x \text{ for } k < K$

We now eliminate all references to the shares of party $N+1$ from the final part of the protocol. The leakages

- $\text{SendOutShare}(m, N+1, k)_{\text{adv}}^{\text{party}(N+1)} := \text{read SendOutShare}(m, N+1, k)_{\text{adv}}^{\text{party}(N+1)} \text{ for } m \leq N+1$

are vacuous since party $N+1$ is honest. If party n is semi-honest and wire k is an output, then the channels

- $\text{RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(N+1, k)$
- $\text{OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(N+1, k)$

read from the channel

- $\text{Share}(N+1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Share-}\Sigma(N+1, k); \text{ret } x_\Sigma \oplus x$

so we may substitute:

- $\text{RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Share-}\Sigma(N+1, k); \text{ret } x_\Sigma \oplus x$
- $\text{OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Share-}\Sigma(N+1, k); \text{ret } x_\Sigma \oplus x$

We thus get the following for channels $\text{RcvdOutShare}(-, N+1, -)_{\text{adv}}^{\text{party}(-)}$ and $\text{OutShare}(-, N+1, -)_{\text{adv}}^{\text{party}(-)}$:

- $\begin{cases} \text{RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Share-}\Sigma(N+1, k); \text{ret } x_\Sigma \oplus x \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Share-}\Sigma(N+1, k); \text{ret } x_\Sigma \oplus x \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$

The top-level internal channels $\text{Share}(N+1, -)$ 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:

- $\begin{cases} \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, k) \\ \text{for } m \leq N+1 \text{ and } n \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N+1 \text{ and } n \leq N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\text{SendOutShare}(m, N+1, k)_{\text{adv}}^{\text{party}(N+1)} := \text{read SendOutShare}(m, N+1, k)_{\text{adv}}^{\text{party}(N+1)} \text{ for } m \leq N+1$
- $\begin{cases} \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$

- $\left\{ \begin{array}{l} \text{RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Share-}\Sigma(N+1, k); \text{ret } x_{\Sigma} \oplus x \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Share-}\Sigma(N+1, k); \text{ret } x_{\Sigma} \oplus x \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share-}\Sigma(m, k) \\ \text{for } n, m \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n, m \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(n, k) := \text{read Share-}\Sigma(N+1, k) \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(n, k) := \text{read Out}(n, k) \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k) \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{array} \right.$

As a final step before the extraction of the simulator, we eliminate any reference to the channels $\text{Share-}\Sigma(N+1, -)$ 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

- $\text{Share-}\Sigma(N+1, k) := \text{read Wire}(k)$

into the channels

- $\text{RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Share-}\Sigma(N+1, k); \text{ret } x_{\Sigma} \oplus x$
- $\text{OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Share-}\Sigma(N+1, k); \text{ret } x_{\Sigma} \oplus x$

which yields the following:

- $\text{RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Wire}(k); \text{ret } x_{\Sigma} \oplus x$
- $\text{OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Wire}(k); \text{ret } x_{\Sigma} \oplus x$

We thus get the following for channels $\text{RcvdOutShare}(-, N+1, -)_{\text{adv}}^{\text{party}(-)}$ and $\text{OutShare}(-, N+1, -)_{\text{adv}}^{\text{party}(-)}$:

$$\begin{aligned}
& \bullet \left\{ \begin{array}{l} \text{RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Wire}(k); \text{ret } x_{\Sigma} \oplus x \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \end{array} \right. \\
& \bullet \left\{ \begin{array}{l} \text{RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{array} \right. \\
& \bullet \left\{ \begin{array}{l} \text{OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Wire}(k); \text{ret } x_{\Sigma} \oplus x \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \end{array} \right. \\
& \bullet \left\{ \begin{array}{l} \text{OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.
\end{aligned}$$

If party n is semi-honest and wire k is an output, then the channel

$$\bullet \text{OutShare-}\Sigma(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share-}\Sigma(N+1, k)$$

reads from the channel

$$\bullet \text{Share-}\Sigma(N+1, k) := \text{read Wire}(k)$$

so we may substitute:

$$\bullet \text{OutShare-}\Sigma(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read Wire}(k)$$

We thus get the following for channels $\text{OutShare-}\Sigma(-, N+1, -)_{\text{adv}}^{\text{party}(-)}$:

$$\bullet \left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read Wire}(k) \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare-}\Sigma(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, N+1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.$$

Finally, if wire k is an output, then for any party n the channel

$$\bullet \text{Out}(n, k) := \text{read Share-}\Sigma(N+1, k)$$

reads from the channel

$$\bullet \text{Share-}\Sigma(N+1, k) := \text{read Wire}(k)$$

so we may substitute:

$$\bullet \text{Out}(n, k) := \text{read Wire}(k)$$

We thus get the following for channels $\text{Out}(-, -)$:

$$\bullet \left\{ \begin{array}{l} \text{Out}(n, k) := \text{read Wire}(k) \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } k \text{ an output} \\ \text{Out}(n, k) := \text{read Out}(n, k) \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } k \text{ not an output} \end{array} \right.$$

The top-level internal channels $\text{Share-}\Sigma(N+1, -)$ 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:

$$\bullet \left\{ \begin{array}{l} \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, k) \\ \text{for } m \leq N+1 \text{ and } n \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N+1 \text{ and } n \leq N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{array} \right.$$

- $\text{SendOutShare}(m, N+1, k)_{\text{adv}}^{\text{party}(N+1)} := \text{read SendOutShare}(m, N+1, k)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N+1$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Wire}(k); \text{ret } x_\Sigma \oplus x \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Wire}(k); \text{ret } x_\Sigma \oplus x \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share-}\Sigma(m, k) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read Wire}(k) \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, N+1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(n, k) := \text{read Wire}(k) \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(n, k) := \text{read Out}(n, k) \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k) \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{array} \right.$

10.4.8 Extracting The Simulator

We are now ready to extract the simulator. The internal protocol $\text{Wires}(C, K)$ together with the output channels

- $\begin{cases} \text{Out}(n, k) := \text{read Wire}(k) \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } k \text{ an output} \\ \text{Out}(n, k) := \text{read Out}(n, k) \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } k \text{ not an output} \end{cases}$

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:

- $\begin{cases} \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, k) \\ \text{for } m \leq N + 1 \text{ and } n \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N + 1 \text{ and } n \leq N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\text{SendOutShare}(m, N + 1, k)_{\text{adv}}^{\text{party}(N+1)} := \text{read SendOutShare}(m, N + 1, k)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N + 1$
- $\begin{cases} \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k) \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Wire}(k); \text{ret } x_{\Sigma} \oplus x \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k) \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Wire}(k); \text{ret } x_{\Sigma} \oplus x \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share-}\Sigma(m, k) \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{OutShare-}\Sigma(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read Wire}(k) \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare-}\Sigma(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$
- $\begin{cases} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k) \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$

In the remainder of the soon-to-be simulator, we must eliminate any references to the channels $\text{In}(-, -)$, $\text{Wire}(-)$, $\text{Out}(-, -)$. We begin with the initial part of the real protocol. Recall the leakage

$$\begin{aligned} & \bullet \begin{cases} \text{In}(n, i)_{\text{adv}}^{\text{id}} := \text{read } \text{In}(n, i) \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{In}(n, i)_{\text{adv}}^{\text{id}} := \text{read } \text{In}(n, i)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases} \\ & \bullet \begin{cases} \text{InRcvd}(n, i)_{\text{adv}}^{\text{id}} := x \leftarrow \text{In}(n, i); \text{ ret } \checkmark \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \text{InRcvd}(n, i)_{\text{adv}}^{\text{id}} := \text{read } \text{InRcvd}(n, i)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{cases} \end{aligned}$$

from the ideal functionality. If party n is semi-honest, then for any input $i < I_n$ the channel

$$\bullet \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{In}(n, i)$$

can be expressed equivalently as

$$\bullet \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{In}(n, i)_{\text{adv}}^{\text{id}}$$

since we have the leakage below:

$$\bullet \text{In}(n, i)_{\text{adv}}^{\text{id}} := \text{read } \text{In}(n, i)$$

We thus get the following for channels $\text{In}(-, -)_{\text{adv}}^{\text{party}(-)}$:

$$\bullet \begin{cases} \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{In}(n, i)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$$

If party n is honest, then for any input $i < I_n$ the channel

$$\bullet \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := x \leftarrow \text{In}(n, i); \text{ ret } \checkmark$$

can be expressed equivalently as

$$\bullet \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{InRcvd}(n, i)_{\text{adv}}^{\text{id}}$$

since we have the leakage below:

$$\bullet \text{InRcvd}(n, i)_{\text{adv}}^{\text{id}} := x \leftarrow \text{In}(n, i); \text{ ret } \checkmark$$

We thus get the following for channels $\text{InRcvd}(-, -)_{\text{adv}}^{\text{party}(-)}$:

$$\bullet \begin{cases} \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{InRcvd}(n, i)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{cases}$$

If party n is semi-honest, then for any party m and any input $i < I_n$ the channel

$$\bullet \text{InShare-}\$(m, n, i) := x \leftarrow \text{In}(n, i); \text{ samp flip}$$

can be expressed equivalently as

$$\bullet \text{InShare-}\$(m, n, i) := x \leftarrow \text{In}(n, i)_{\text{adv}}^{\text{id}}; \text{ samp flip}$$

since we have the leakage below:

- $\text{In}(n, i)_{\text{adv}}^{\text{id}} := \text{read } \text{In}(n, i)$

If party n is honest, then for any party m and any input $i < I_n$ the channel

- $\text{InShare-}\$(m, n, i) := x \leftarrow \text{In}(n, i); \text{ samp flip}$

can be expressed equivalently as

- $\text{InShare-}\$(m, n, i) := _ \leftarrow \text{InRcvd}(n, i)_{\text{adv}}^{\text{id}}; \text{ samp flip}$

since we have the leakage below:

- $\text{InRcvd}(n, i)_{\text{adv}}^{\text{id}} := x \leftarrow \text{In}(n, i); \text{ ret } \checkmark$

We thus get the following for channels $\text{InShare-}\$(-, -, -)$:

- $\begin{cases} \text{InShare-}\$(m, n, i) := x \leftarrow \text{In}(n, i)_{\text{adv}}^{\text{id}}; \text{ samp flip} \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(m, n, i) := _ \leftarrow \text{InRcvd}(n, i)_{\text{adv}}^{\text{id}}; \text{ samp flip} \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

If party n is semi-honest, then for any input $i < I_n$ the channels

- $\text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i); \text{ ret } x_\Sigma \oplus x$
- $\text{SendInShare}(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i); \text{ ret } x_\Sigma \oplus x$

can be expressed equivalently as

- $\text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i)_{\text{adv}}^{\text{id}}; \text{ ret } x_\Sigma \oplus x$
- $\text{SendInShare}(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i)_{\text{adv}}^{\text{id}}; \text{ ret } x_\Sigma \oplus x$

since we have the leakage below:

- $\text{In}(n, i)_{\text{adv}}^{\text{id}} := \text{read } \text{In}(n, i)$

We thus get the following for channels $\text{InShare-}\$(N + 1, -, -)_{\text{adv}}^{\text{party}(-)}$ and $\text{SendInShare}(N + 1, -, -)_{\text{adv}}^{\text{party}(-)}$:

- $\begin{cases} \text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i)_{\text{adv}}^{\text{id}}; \text{ ret } x_\Sigma \oplus x \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{SendInShare}(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i)_{\text{adv}}^{\text{id}}; \text{ ret } x_\Sigma \oplus x \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{SendInShare}(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{SendInShare}(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

The final version Init of the initial part of the simulator is therefore as follows:

- $\begin{cases} \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{In}(n, i)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read } \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$

- $$\left\{ \begin{array}{l} \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InRcvd}(n, i)_{\text{adv}}^{\text{id}} \\ \quad \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \\ \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{array} \right.$$
- $$\left\{ \begin{array}{l} \text{InShare-}\$(m, n, i) := x \leftarrow \text{In}(n, i)_{\text{adv}}^{\text{id}}; \text{ samp flip} \\ \quad \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(m, n, i) := _ \leftarrow \text{InRcvd}(n, i)_{\text{adv}}^{\text{id}}; \text{ samp flip} \\ \quad \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{array} \right.$$
- $$\left\{ \begin{array}{l} \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i) \\ \quad \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{array} \right.$$
- $$\left\{ \begin{array}{l} \text{InShare-}\$-\Sigma(0, n, i) := \text{read InShare-}\$(0, n, i) \\ \quad \text{for } n \leq N + 1 \text{ and } i < I_n \\ \text{InShare-}\$-\Sigma(m + 1, n, i) := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(m, n, i); x_{m+1} \leftarrow \text{InShare-}\$(m + 1, n, i); \text{ ret } x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \text{ and } n \leq N + 1 \text{ and } i < I_n \end{array} \right.$$
- $$\left\{ \begin{array}{l} \text{InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$-\Sigma(m, n, i) \\ \quad \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{array} \right.$$
- $$\left\{ \begin{array}{l} \text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i)_{\text{adv}}^{\text{id}}; \text{ ret } x_\Sigma \oplus x \\ \quad \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{array} \right.$$
- $$\left\{ \begin{array}{l} \text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i) \\ \quad \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{array} \right.$$
- $$\left\{ \begin{array}{l} \text{SendInShare}(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i)_{\text{adv}}^{\text{id}}; \text{ ret } x_\Sigma \oplus x \\ \quad \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{SendInShare}(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(N + 1, n, i)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{array} \right.$$
- $$\left\{ \begin{array}{l} \text{RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read InShare-}\$(m, n, i) \\ \quad \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \text{RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} \\ \quad \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{array} \right.$$
- $$\text{RcvdInShare}(N + 1, n, i)_{\text{adv}}^{\text{party}(N+1)} := \text{read RcvdInShare}(N + 1, n, i)_{\text{adv}}^{\text{party}(N+1)} \text{ for } n \leq N + 1 \text{ and } i < I_n$$
- $$\text{InShare}(m, n, i) := \text{read InShare-}\$(m, n, i) \text{ for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n$$

- $$\begin{cases} \text{InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read InShare}(m, n, i) \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \text{InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$$
- $\text{InShare}(N + 1, n, i)_{\text{adv}}^{\text{party}(N+1)} := \text{read InShare}(N + 1, n, i)_{\text{adv}}^{\text{party}(N+1)}$ for $n \leq N + 1$ and $i < I_n$

This is followed by the hiding of the channels

- $\text{InShare-}\$(m, n, i)$ for $m \leq N$ and $n \leq N + 1$ and $i < I_n$,
- $\text{InShare-}\$-\Sigma(m, n, i)$ for $m \leq N$ and $n \leq N + 1$ and $i < I_n$.

We continue with the final part of the soon-to-be simulator. Recall the definition of the output channels

- $$\begin{cases} \text{Out}(n, k) := \text{read Wire}(k) \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } k \text{ an output} \\ \text{Out}(n, k) := \text{read Out}(n, k) \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } k \text{ not an output} \end{cases}$$

in the ideal functionality. If party n is semi-honest and wire k is an output, then the channels

- $\text{RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Wire}(k); \text{ret } x_\Sigma \oplus x$
- $\text{OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Wire}(k); \text{ret } x_\Sigma \oplus x$

can be expressed equivalently as

- $\text{RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k); \text{ret } x_\Sigma \oplus x$
- $\text{OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k); \text{ret } x_\Sigma \oplus x$

since we have the definition below:

- $\text{Out}(n, k) := \text{read Wire}(k)$

We thus get the following for channels $\text{RcvdOutShare}(-, N + 1, -)_{\text{adv}}^{\text{party}(-)}$ and $\text{OutShare}(-, N + 1, -)_{\text{adv}}^{\text{party}(-)}$:

- $$\begin{cases} \text{RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k); \text{ret } x_\Sigma \oplus x \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$$
- $$\begin{cases} \text{OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k); \text{ret } x_\Sigma \oplus x \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$$

If party n is semi-honest and wire k is an output, then the channel

- $\text{OutShare-}\Sigma(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read Wire}(k)$

can be expressed equivalently as

- $\text{OutShare-}\Sigma(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)$

since we have the definition below:

- $\text{Out}(n, k) := \text{read Wire}(k)$

We thus get the following for channels $\text{OutShare-}\Sigma(-, N+1, -)^{\text{party}(-)}_{\text{adv}}$:

- $$\begin{cases} \text{OutShare-}\Sigma(n, N+1, k)^{\text{party}(n)}_{\text{adv}} := \text{read Out}(n, k) \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare-}\Sigma(n, N+1, k)^{\text{party}(n)}_{\text{adv}} := \text{read OutShare-}\Sigma(n, N+1, k)^{\text{party}(n)}_{\text{adv}} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$$

The latest version Fin of the final part of the soon-to-be simulator is therefore as follows:

- $$\begin{cases} \text{SendOutShare}(m, n, k)^{\text{party}(n)}_{\text{adv}} := \text{read Share}(n, k) \\ \text{for } m \leq N+1 \text{ and } n \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{SendOutShare}(m, n, k)^{\text{party}(n)}_{\text{adv}} := \text{read SendOutShare}(m, n, k)^{\text{party}(n)}_{\text{adv}} \\ \text{for } m \leq N+1 \text{ and } n \leq N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$$
- $$\text{SendOutShare}(m, N+1, k)^{\text{party}(N+1)}_{\text{adv}} := \text{read SendOutShare}(m, N+1, k)^{\text{party}(N+1)}_{\text{adv}} \text{ for } m \leq N+1$$
- $$\begin{cases} \text{RcvdOutShare}(n, m, k)^{\text{party}(n)}_{\text{adv}} := \text{read Share}(m, k) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{RcvdOutShare}(n, m, k)^{\text{party}(n)}_{\text{adv}} := \text{read RcvdOutShare}(n, m, k)^{\text{party}(n)}_{\text{adv}} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$$
- $$\begin{cases} \text{RcvdOutShare}(n, N+1, k)^{\text{party}(n)}_{\text{adv}} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k); \text{ret } x_\Sigma \oplus x \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{RcvdOutShare}(n, N+1, k)^{\text{party}(n)}_{\text{adv}} := \text{read RcvdOutShare}(n, N+1, k)^{\text{party}(n)}_{\text{adv}} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$$
- $$\begin{cases} \text{OutShare}(n, m, k)^{\text{party}(n)}_{\text{adv}} := \text{read Share}(m, k) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare}(n, m, k)^{\text{party}(n)}_{\text{adv}} := \text{read OutShare}(n, m, k)^{\text{party}(n)}_{\text{adv}} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$$
- $$\begin{cases} \text{OutShare}(n, N+1, k)^{\text{party}(n)}_{\text{adv}} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k); \text{ret } x_\Sigma \oplus x \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare}(n, N+1, k)^{\text{party}(n)}_{\text{adv}} := \text{read OutShare}(n, N+1, k)^{\text{party}(n)}_{\text{adv}} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$$
- $$\begin{cases} \text{OutShare-}\Sigma(n, m, k)^{\text{party}(n)}_{\text{adv}} := \text{read Share-}\Sigma(m, k) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare-}\Sigma(n, m, k)^{\text{party}(n)}_{\text{adv}} := \text{read OutShare-}\Sigma(n, m, k)^{\text{party}(n)}_{\text{adv}} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$$
- $$\begin{cases} \text{OutShare-}\Sigma(n, N+1, k)^{\text{party}(n)}_{\text{adv}} := \text{read Out}(n, k) \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare-}\Sigma(n, N+1, k)^{\text{party}(n)}_{\text{adv}} := \text{read OutShare-}\Sigma(n, N+1, k)^{\text{party}(n)}_{\text{adv}} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$$

$$\bullet \begin{cases} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k) \\ \quad \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$$

We now recall the leakage

$$\bullet \begin{cases} \text{Out}(n, k)_{\text{adv}}^{\text{id}} := \text{read Out}(n, k) \\ \quad \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{Out}(n, k)_{\text{adv}}^{\text{id}} := \text{read Out}(n, k)_{\text{adv}}^{\text{id}} \\ \quad \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$$

from the ideal functionality. If party n is semi-honest and wire k is an output, then the channels

$$\bullet \text{RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k); \text{ret } x_{\Sigma} \oplus x$$

$$\bullet \text{OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k); \text{ret } x_{\Sigma} \oplus x$$

can be expressed equivalently as

$$\bullet \text{RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k)_{\text{adv}}^{\text{id}}; \text{ret } x_{\Sigma} \oplus x$$

$$\bullet \text{OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k)_{\text{adv}}^{\text{id}}; \text{ret } x_{\Sigma} \oplus x$$

since we have the leakage below:

$$\bullet \text{Out}(n, k)_{\text{adv}}^{\text{id}} := \text{read Out}(n, k)$$

We thus get the following for channels $\text{RcvdOutShare}(-, N + 1, -)_{\text{adv}}^{\text{party}(-)}$ and $\text{OutShare}(-, N + 1, -)_{\text{adv}}^{\text{party}(-)}$:

$$\bullet \begin{cases} \text{RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k)_{\text{adv}}^{\text{id}}; \text{ret } x_{\Sigma} \oplus x \\ \quad \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$$

$$\bullet \begin{cases} \text{OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k)_{\text{adv}}^{\text{id}}; \text{ret } x_{\Sigma} \oplus x \\ \quad \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} \\ \quad \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$$

If party n is semi-honest and wire k is an output, then the channel

$$\bullet \text{OutShare-}\Sigma(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)$$

can be expressed equivalently as

$$\bullet \text{OutShare-}\Sigma(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{id}}$$

since we have the leakage below:

$$\bullet \text{Out}(n, k)_{\text{adv}}^{\text{id}} := \text{read Out}(n, k)$$

We thus get the following for channels $\text{OutShare-}\Sigma(-, N + 1, -)_{\text{adv}}^{\text{party}(-)}$:

- $$\begin{cases} \text{OutShare-}\Sigma(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare-}\Sigma(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, N+1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$$

Finally, if party n is semi-honest, then for any wire k the channel

- $\text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)$

can be expressed equivalently as

- $\text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{id}}$

since we have the leakage below:

- $\text{Out}(n, k)_{\text{adv}}^{\text{id}} := \text{read Out}(n, k)$

We thus get the following for channels $\text{Out}(-, -)_{\text{adv}}^{\text{party}(-)}$:

- $$\begin{cases} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest} \\ \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest} \end{cases}$$

The final version Fin of the final part of the simulator is therefore as follows:

- $$\begin{cases} \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, k) \\ \text{for } m \leq N+1 \text{ and } n \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N+1 \text{ and } n \leq N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$$
- $\text{SendOutShare}(m, N+1, k)_{\text{adv}}^{\text{party}(N+1)} := \text{read SendOutShare}(m, N+1, k)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N+1$
- $$\begin{cases} \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$$
- $$\begin{cases} \text{RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k)_{\text{adv}}^{\text{id}}; \text{ret } x_{\Sigma} \oplus x \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \\ \text{RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{cases}$$
- $$\begin{cases} \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$$
- $$\begin{cases} \text{OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k)_{\text{adv}}^{\text{id}}; \text{ret } x_{\Sigma} \oplus x \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \\ \text{OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, N+1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{cases}$$

- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share-}\Sigma(m, k) \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{array} \right.$

10.4.9 The Simulator

The simulator consists of four parts. In the initial phase, we have the protocol Init:

- $\left\{ \begin{array}{l} \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read In}(n, i)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{In}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read In}(n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InRcvd}(n, i)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InRcvd}(n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{InShare-}\$(m, n, i) := x \leftarrow \text{In}(n, i)_{\text{adv}}^{\text{id}}; \text{ samp flip} \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{InShare-}\$(m, n, i) := _ \leftarrow \text{InRcvd}(n, i)_{\text{adv}}^{\text{id}}; \text{ samp flip} \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i) \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{InShare-}\$-\Sigma(0, n, i) := \text{read InShare-}\$(0, n, i) \\ \text{for } n \leq N + 1 \text{ and } i < I_n \end{array} \right.$
- $\left\{ \begin{array}{l} \text{InShare-}\$-\Sigma(m + 1, n, i) := x_{\Sigma} \leftarrow \text{InShare-}\$-\Sigma(m, n, i); x_{m+1} \leftarrow \text{InShare-}\$(m + 1, n, i); \text{ ret } x_{\Sigma} \oplus x_{m+1} \\ \text{for } m < N \text{ and } n \leq N + 1 \text{ and } i < I_n \end{array} \right.$
- $\left\{ \begin{array}{l} \text{InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$-\Sigma(m, n, i) \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$-\Sigma(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N \text{ and } n \leq N + 1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{array} \right.$

- $\begin{cases} \text{InShare-}\$(N+1, n, i)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i)_{\text{adv}}^{\text{id}}; \text{ret } x_\Sigma \oplus x \\ \text{for } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{InShare-}\$(N+1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(N+1, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read InShare-}\$(m, n, i) \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(m, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{SendInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(n)} := x_\Sigma \leftarrow \text{InShare-}\$-\Sigma(N, n, i); x \leftarrow \text{In}(n, i)_{\text{adv}}^{\text{id}}; \text{ret } x_\Sigma \oplus x \\ \text{for } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ semi-honest} \\ \text{SendInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(n)} := \text{read SendInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N+1 \text{ and } i < I_n \text{ if } n \text{ honest} \end{cases}$
- $\begin{cases} \text{RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read InShare-}\$(m, n, i) \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \text{RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read RcvdInShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$
- $\text{RcvdInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(N+1)} := \text{read RcvdInShare}(N+1, n, i)_{\text{adv}}^{\text{party}(N+1)}$ for $n \leq N+1$ and $i < I_n$
- $\text{InShare}(m, n, i) := \text{read InShare-}\(m, n, i) for $m \leq N$ and $n \leq N+1$ and $i < I_n$
- $\begin{cases} \text{InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read InShare}(m, n, i) \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } m \text{ semi-honest} \\ \text{InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} := \text{read InShare}(m, n, i)_{\text{adv}}^{\text{party}(m)} \\ \text{for } m \leq N \text{ and } n \leq N+1 \text{ and } i < I_n \text{ if } m \text{ honest} \end{cases}$
- $\text{InShare}(N+1, n, i)_{\text{adv}}^{\text{party}(N+1)} := \text{read InShare}(N+1, n, i)_{\text{adv}}^{\text{party}(N+1)}$ for $n \leq N+1$ and $i < I_n$

This is followed by the hiding of the channels

- $\text{InShare-}\$(m, n, i)$ for $m \leq N$ and $n \leq N+1$ and $i < I_n$,
- $\text{InShare-}\$-\Sigma(m, n, i)$ for $m \leq N$ and $n \leq N+1$ and $i < I_n$.

In the inductive phase, we have the protocol $\text{Circ}(C, K)$, obtained by merging the two protocols $\text{Shares}(C, K)$ and $\text{Adv}(C, K)$ from earlier:

- $\text{Circ}(\epsilon, 0)$ is the protocol 0
- $\text{Circ}(C; \text{input-gate}(p, i), K+1)$ is the composition of $\text{Circ}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n \leq N$ and $m \leq N+1$
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N+1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n \leq N$ and $m \leq N+1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N+1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n \leq N$ and $m \leq N+1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N+1$

- $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
- $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
- $\text{Share}(n, K) := \text{read InShare}(n, p, i)$ for $n \leq N$
- $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ if } n \text{ honest} \end{cases}$
- $\text{Share}(N + 1, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Share}(N + 1, K)_{\text{adv}}^{\text{party}(N+1)}$
- $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Circ}(C; \text{not-gate}(k), K + 1)$ is the composition of $\text{Circ}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := \text{read Share}(n, k)$ for $n \leq N$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ if } n \text{ honest} \end{cases}$
 - $\text{Share}(N + 1, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Share}(N + 1, K)_{\text{adv}}^{\text{party}(N+1)}$
 - $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$

- $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Circ}(C; \text{xor-gate}(k, l), K + 1)$ is the composition of $\text{Circ}(C, K)$ with the protocol
 - $\text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := \text{read RcvdBit}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb}(n, m, K) := \text{read Ctrb}(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K) := \text{read Ctrb-}\Sigma(n, m, K)$ for $n \leq N$ and $m \leq N + 1$
 - $\text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)}$ for $n, m \leq N + 1$
 - $\text{Share}(n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n \oplus y_n$ for $n \leq N$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N \text{ if } n \text{ semi-honest} \end{cases}$
 - $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ if } n \text{ honest} \end{cases}$
 - $\text{Share}(N + 1, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Share}(N + 1, K)_{\text{adv}}^{\text{party}(N+1)}$
 - $\text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
 - $\text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$

- $\text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}}$ for $n, m \leq N + 1$
- $\text{Circ}(C; \text{and-gate}(k, l), K + 1)$ is the composition of $\text{Circ}(C, K)$ with the protocol
 - $\begin{cases} \text{SendBit}(n, m, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{samp flip} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n < m \\ \text{SendBit}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \geq m \end{cases}$
 - $\begin{cases} \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K) \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read SendBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\text{SendBit}(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read SendBit}(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N + 1$
 - $\text{RcvdBit}(n, m, K) := b \leftarrow \text{SendBit}(m, n, K); x_m \leftarrow \text{Share}(m, k); y_m \leftarrow \text{Share}(m, l);$
 $x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus (x_m * y_n) \oplus (x_n * y_m)$ for $n, m \leq N$
 - $\text{RcvdBit}(n, N + 1, K) := \text{read RcvdBit}(n, N + 1, K)$ for $n \leq N$
 - $\begin{cases} \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K) \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdBit}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\text{RcvdBit}(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read RcvdBit}(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N + 1$
 - $\begin{cases} \text{Ctrb}(n, m, K) := \text{read SendBit}(n, m, K) \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n < m \\ \text{Ctrb}(n, m, K) := \text{read RcvdBit}(n, m, K) \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } m < n \\ \text{Ctrb}(n, n, K) := x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } x_n * y_n \\ \text{for } n \leq N \end{cases}$
 - $\begin{cases} \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K) \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb}(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\text{Ctrb}(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Ctrb}(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N + 1$
 - $\begin{cases} \text{Ctrb-}\Sigma(n, 0, K) := \text{read Ctrb}(n, 0, K) \\ \text{for } n \leq N \\ \text{Ctrb-}\Sigma(n, m + 1, K) := b_{\Sigma} \leftarrow \text{Ctrb-}\Sigma(n, m, K); b \leftarrow \text{Ctrb}(n, m + 1, K); \text{ret } b_{\Sigma} \oplus b \\ \text{for } n, m \leq N \end{cases}$
 - $\begin{cases} \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K) \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest} \\ \text{Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} := \text{read Ctrb-}\Sigma(n, m, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest} \end{cases}$
 - $\text{Ctrb-}\Sigma(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Ctrb-}\Sigma(N + 1, m, K)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N + 1$

- $\text{Share}(n, K) := \text{read Ctrb-}\Sigma(n, N + 1, K)$ for $n \leq N$
- $\begin{cases} \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K) \\ \text{for } n \leq N \text{ if } n \text{ semi-honest} \\ \text{Share}(n, K)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, K)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N \text{ if } n \text{ honest} \end{cases}$
- $\text{Share}(N + 1, K)_{\text{adv}}^{\text{party}(N+1)} := \text{read Share}(N + 1, K)_{\text{adv}}^{\text{party}(N+1)}$
- $\begin{cases} \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$
- $\text{OTMsg}_0(N + 1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_0(N + 1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N + 1$
- $\begin{cases} \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$
- $\text{OTMsg}_1(N + 1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_1(N + 1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N + 1$
- $\begin{cases} \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus y_n \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$
- $\text{OTMsg}_2(N + 1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_2(N + 1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N + 1$
- $\begin{cases} \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } b \oplus x_n \oplus y_n \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest and } n < m \\ \text{OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest or } n \geq m \end{cases}$
- $\text{OTMsg}_3(N + 1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsg}_3(N + 1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N + 1$
- $\begin{cases} \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases}$
- $\text{OTMsgRcvd}_0(N + 1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_0(N + 1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N + 1$
- $\begin{cases} \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases}$
- $\text{OTMsgRcvd}_1(N + 1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_1(N + 1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N + 1$
- $\begin{cases} \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ honest and } n < m \\ \text{OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N \text{ and } m \leq N + 1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases}$

- $\text{OTMsgRcvd}_2(N+1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_2(N+1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N+1$
- $\begin{cases} \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := b \leftarrow \text{SendBit}(n, m, K); x_n \leftarrow \text{Share}(n, k); y_n \leftarrow \text{Share}(n, l); \text{ret } \checkmark \\ \text{for } n \leq N \text{ and } m \leq N+1 \text{ if } n \text{ honest and } n < m \end{cases}$
- $\begin{cases} \text{OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N \text{ and } m \leq N+1 \text{ if } n \text{ semi-honest or } n \geq m \end{cases}$
- $\text{OTMsgRcvd}_3(N+1, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTMsgRcvd}_3(N+1, m, K)_{\text{adv}}^{\text{ot}}$ for $m \leq N+1$
- $\begin{cases} \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, k) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ semi-honest and } n < m \end{cases}$
- $\begin{cases} \text{OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ honest or } n \geq m \end{cases}$
- $\text{OTChc}_0(n, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_0(n, N+1, K)_{\text{adv}}^{\text{ot}}$ for $n \leq N+1$
- $\begin{cases} \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read Share}(m, l) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ semi-honest and } n < m \end{cases}$
- $\begin{cases} \text{OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ honest or } n \geq m \end{cases}$
- $\text{OTChc}_1(n, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChc}_1(n, N+1, K)_{\text{adv}}^{\text{ot}}$ for $n \leq N+1$
- $\begin{cases} \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := x_m \leftarrow \text{Share}(m, k); \text{ret } \checkmark \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ honest and } n < m \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ semi-honest or } n \geq m \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_0(n, N+1, K)_{\text{adv}}^{\text{ot}} := x_\Sigma \leftarrow \text{Share-}\Sigma(N, k); \text{ret } \checkmark \\ \text{for } n \leq N \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_0(N+1, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_0(N+1, N+1, K)_{\text{adv}}^{\text{ot}} \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := y_m \leftarrow \text{Share}(m, l); \text{ret } \checkmark \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ honest and } n < m \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ semi-honest or } n \geq m \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_1(n, N+1, K)_{\text{adv}}^{\text{ot}} := y_\Sigma \leftarrow \text{Share-}\Sigma(N, l); \text{ret } \checkmark \\ \text{for } n \leq N \end{cases}$
- $\begin{cases} \text{OTChcRcvd}_1(N+1, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTChcRcvd}_1(N+1, N+1, K)_{\text{adv}}^{\text{ot}} \end{cases}$
- $\begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read RcvdBit}(m, n, K) \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ semi-honest} \end{cases}$
- $\begin{cases} \text{OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, m, K)_{\text{adv}}^{\text{ot}} \\ \text{for } n \leq N+1 \text{ and } m \leq N \text{ if } m \text{ honest} \end{cases}$
- $\text{OTOut}(n, N+1, K)_{\text{adv}}^{\text{ot}} := \text{read OTOut}(n, N+1, K)_{\text{adv}}^{\text{ot}}$ for $n \leq N+1$

This is followed by the hiding of the channels

- $\text{SendBit}(n, m, k)$ for $n \leq N$ and $m \leq N+1$ and $k < K$,
- $\text{RcvdBit}(n, m, k)$ for $n \leq N$ and $m \leq N+1$ and $k < K$,
- $\text{Ctrb}(n, m, k)$ for $n \leq N$ and $m \leq N+1$ and $k < K$,
- $\text{Ctrb-}\Sigma(n, m, k)$ for $n \leq N$ and $m \leq N+1$ and $k < K$.

In the final phase, we have the protocol Fin:

- $\left\{ \begin{array}{l} \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(n, k) \\ \text{for } m \leq N + 1 \text{ and } n \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} := \text{read SendOutShare}(m, n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } m \leq N + 1 \text{ and } n \leq N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{array} \right.$
- $\text{SendOutShare}(m, N + 1, k)_{\text{adv}}^{\text{party}(N+1)} := \text{read SendOutShare}(m, N + 1, k)_{\text{adv}}^{\text{party}(N+1)}$ for $m \leq N + 1$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k) \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k)_{\text{adv}}^{\text{id}}; \text{ret } x_{\Sigma} \oplus x \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and wire } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read RcvdOutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or wire } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share}(m, k) \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := x_{\Sigma} \leftarrow \text{Share-}\Sigma(N, k); x \leftarrow \text{Out}(n, k)_{\text{adv}}^{\text{id}}; \text{ret } x_{\Sigma} \oplus x \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare}(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read Share-}\Sigma(m, k) \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, m, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } m \leq N \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest and } k \text{ an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{OutShare-}\Sigma(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} := \text{read OutShare-}\Sigma(n, N + 1, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest or } k \text{ not an output} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{id}} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ semi-honest} \end{array} \right.$
- $\left\{ \begin{array}{l} \text{Out}(n, k)_{\text{adv}}^{\text{party}(n)} := \text{read Out}(n, k)_{\text{adv}}^{\text{party}(n)} \\ \text{for } n \leq N + 1 \text{ and } k < K \text{ if } n \text{ honest} \end{array} \right.$

At last, we have the channels

$$\bullet \begin{cases} \text{Share-}\Sigma(0, k) := \text{read Share}(0, k) \\ \quad \text{for } k < K \\ \text{Share-}\Sigma(m+1, k) := x_\Sigma \leftarrow \text{Share-}\Sigma(m, k); x_{m+1} \leftarrow \text{Share}(m+1, k); \text{ret } x_\Sigma \oplus x_{m+1} \\ \quad \text{for } m < N \text{ and } k < K \end{cases}$$

that keep track of the sum of shares of parties $0, \dots, N$.

The composition of the four parts is followed by the hiding of the channels

- $\text{InShare}(m, n, i)$ for $m \leq N$ and $n \leq N+1$ and $i < I_n$,
- $\text{Share}(n, k)$ for $n \leq N$ and $k < K$,
- $\text{Share-}\Sigma(m, k)$ for $m \leq N$ and $k < K$.

Composing the ideal protocol with the simulator, and substituting away the ideal leakage

- $\text{In}(n, i)_{\text{adv}}^{\text{id}}$ for $n \leq N+1$ and $i < I_n$,
- $\text{InRcvd}(n, i)_{\text{adv}}^{\text{id}}$ for $n \leq N+1$ and $i < I_n$,
- $\text{Out}(n, k)_{\text{adv}}^{\text{id}}$ for $n \leq N+1$ 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.