Errata to MPC Course

Naman Kumar

March 25, 2024

Abstract

Basic notes on MPC.

Contents

1 Secure N-party Computation

It's probably worth reiterating some of the formalisms of these definitions with a bit more lucidity, just as a simple reference and as some sort of illumination.

1.1 Real World Instantiation

In the real world, N parties have inputs $\mathbf{x} = (x_1, \ldots, x_n)$, an agreed-upon function $f : (x_1, \ldots, x_n) \mapsto (y_1, \ldots, y_n)$ and they follow a protocol Π which gives them the set of *party outputs* $\textsc{out}_\Pi(\lambda, \mathbf{x}) := (y_1, \ldots, y_n)$. Assume that t of these parties are controlled by the adversary; define the *view* $\textsc{view}_{\Pi,\mathcal{A}}(\lambda, \mathbf{x})$ to be the set of all inputs of corrupted parties, along with messages exchanged that the adversary has sent, received, or eavesdropped-upon.

$\Longrightarrow$ **Definition 1.1** (Real World Distribution)**.** *The real-world output* $\textsc{real}_{\Pi,\mathcal{A}}(\lambda, \mathbf{x})$ *is defined as the tuple*

$$\textsc{real}_{\Pi,\mathcal{A}}(\lambda, \mathbf{x}) := (\textsc{out}_\Pi(\lambda, \mathbf{x}), \textsc{view}_{\Pi,\mathcal{A}}(\lambda, \mathbf{x})).$$

1.2 Ideal World Instantiation

In the ideal world, there is no protocol, only a functionality $\mathcal{F}$ which takes in inputs from each party, computes the output f, and returns $(f(x_1), \ldots, f(x_n))$ to the parties. The output, $\textsc{out}_F(\lambda, \mathbf{x})$ is the same – the output of the parties after the protocol execution – while instead of the *view* (which is, of course, not identical to that of the actual ideal-world 'protocol' execution; a protocol could be multi-round), there is the *view of a simulator* (a 'fake' adversary which works in the ideal world, but has access to the inputs of the real adversary), which is the output of the simulator on any given execution.

$\Longrightarrow$ **Definition 1.2** (Ideal World Distribution)**.** *The ideal-world output* $\textsc{ideal}_{\mathcal{F},\mathcal{S}}(\lambda, \mathbf{x})$ *is defined as the tuple*

$$\textsc{ideal}_{\mathcal{F},\mathcal{S}}(\lambda, \mathbf{x}) := (\textsc{out}_\mathcal{F}(\lambda, \mathbf{x}), \textsc{view}_{\mathcal{F},\mathcal{S}}(\lambda, \mathbf{x})).$$

Note that these distributions are *joint* distributions.

1.3 Secure N-Party Protocol

$\Longrightarrow$ **Definition 1.3** (t-Privacy of a Protocol)**.** *An N-Party protocol Π is considered t-private if for any PPT adversary that corrupts t of the parties, there exists a PPT simulator such that*

$$\{\textsc{ideal}_{\mathcal{F},\mathcal{S}}(\lambda, \mathbf{x})\} \equiv \{\textsc{real}_{\Pi,\mathcal{A}}(\lambda, \mathbf{x})\}.$$

Remark. There's two examples here of the non-privacy of a protocol. One of them is a function which takes no inputs and outputs a random $b \leftarrow \{0, 1\}$ to one of the parties, say P_0. The protocol in which P_1 samples a bit and sends it to P_0 is not secure since if P_1 is corrupted, the view of the adversary is different; it has the bit b in it. Similarly, if a functionality takes no input and outputs pq, $(p + q)$ to the parties, then, again, the protocol in which P_0 chooses p, q and sends $p + q$ is insecure since it learns the numbers p and q.

2 Oblivious Transfer

Oblivious Transfer is a simple MPC functionality parametrized by a selection of sender inputs and a receiver index.

> **Parameters:** The sender S has a selection of N input strings $(m_0, \ldots, m_{N-1})$, while the receiver R has an index $i \in [N]$.
> **Outputs:** S receives nothing while R receives m_i.

Figure 1: 1-out-of-N Oblivious Transfer.

2.1 Protocol for 1-out-of-2 OT

We now demonstrate a simple protocol for 1-out-of-2 OT [EGL85]. We begin with a CPA-secure encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, and define the notion of oblivious sampling.

$\Longrightarrow$ **Definition 2.1** (PKE with Obliviously Sampleable Encryption Key)**.** *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has obliviously sampleable public keys if there exist algorithms* Samp *and* pkSamp *such that*

- $\{\mathsf{Samp}(1^\lambda)\}$ *is computationally indistinguishable from* $\{\mathsf{pk} : (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)\}$.
- $\{(\mathsf{pk}, r) : r \xleftarrow{\$} \{0,1\}^\lambda, \mathsf{pk} \leftarrow \mathsf{Samp}(1^\lambda; r)\}$ *is computationally indistinguishable from* $\{(\mathsf{pk}, r) : (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}, r \leftarrow \mathsf{pkSim}(\mathsf{pk})\}$.

The protocol proceeds as follows.

> - Receiver runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $\mathsf{pk}' \leftarrow \mathsf{Samp}(1^\lambda)$ and sets $\mathsf{pk}_b = \mathsf{pk}$, and $\mathsf{pk}_{1-b} = \mathsf{pk}'$.
> - Receiver sends pk_0 and pk_1.
> - Sender encrypts $c_i = \mathsf{Enc}_{\mathsf{pk}_i}(m_i)$.
> - Sender sends c_0, c_1.
> - Receiver decrypts $m_b = \mathsf{Dec}_{\mathsf{sk}_b}(c_b)$.

Figure 2: 1-out-of-2 Oblivious Transfer from obliviously sampleable PKE.

2.2 Assumptions Underlying OT

Unfortunately, it is not known how oblivious transfer can be constructed from symmetric-key primitives. In particular, we can show that black-box constructions of oblivious transfer from OWFs is impossible.

$\Longrightarrow$ **Theorem 2.1** (Black-Box Separation of OT and OWFs)**.** *There is no construction of Oblivious Transfer that makes black-box use of One-Way Functions.*

Proof. We will show that Any Oblivious Transfer protocol implies a Key Exchange Protocol. The result then follows from [IR89], which separates key exchange from all one-way functions.

The KE protocol is very simple:

1. P_0 samples $r \xleftarrow{\$} \{0,1\}^\lambda$ and sends $(r, 0^\lambda)$ to $\mathcal{F}_{\mathsf{OT}}$.
2. P_1 sends 0 to $\mathcal{F}_{\mathsf{OT}}$.
3. P_1 receives r.

We claim that this is a secure key exchange. The proof proceeds by hybrid argument: first, the above protocol is indistinguishable to an outside observer from a hybrid in which P_1 inputs 1 instead of 0 by receiver security, and secondly it is indistinguishable from a hybrid in which P_0 inputs $(0^\lambda, 0^\lambda)$ instead by sender security. However, this second hybrid is (information-theoretically) independent of r, and it follows that the eavesdropper cannot gain any information about r, and we are done. $\square$

2.3 OT in the Correlated Randomness Model

We now consider the correlated randomness model, where at the beginning of the protocol both P_0 and P_1 are given random strings r_0 and r_1 by a trusted party such that they satisfy some correlation. In particular, we will show how to obtain OT from a set of *random OTs*.

$\Longrightarrow$ **Definition 2.2** (Random OTs)**.** *We say that two parties P_0 and P_1 possess random OT correlations if party P_0 (the sender) possesses $(\tilde{m}_0, \tilde{m}_1) \xleftarrow{\$} \{0,1\}^{2\lambda}$ and party P_1 (the receiver) possesses $(\vec{b}, (\tilde{m}_{b_i})_{i\in\lambda}) \in \{0,1\}^{2\lambda}$ where $\vec{b} \xleftarrow{\$} \{0,1\}^\lambda$.*

The above definition formalizes this idea of P_0 and P_1 having 'random OT correlations,' i.e. P_0 has random sender inputs and P_1 has random receiver inputs and outputs. We now see how this can be utilized to get OT.

Parameters: P_0 has input bits (m_0, m_1), while P_1 has input bit s.
Correlations: P_0 has random bits $(\tilde{m}_0, \tilde{m}_1)$, while P_1 has random bits $(b, \tilde{m}_b)$.

Protocol:

1. P_1 sends $b \oplus s$ to P_0.
2. P_0 replies with $(r_0, r_1) := (m_0 \oplus \tilde{m}_{b\oplus s}, m_1 \oplus \tilde{m}_{1\oplus b\oplus s})$.

Figure 3: OT from Random OT.

Correctness of the above protocol follows from simple considerations: if $s = 0$, then $b \oplus s = b$, and in this case P_1 can decrypt the string r_0 since it knows $\tilde{m}_b$. If $s = 1$, then $b \oplus s = b \oplus 1$, and in this case P_1 can decrypt r_1 since it knows $\tilde{m}_{b \oplus s \oplus 1} = \tilde{m}_b$.

3 Modifications to OT

We first construct two variants of OT that provide the additional property of *information-theoretic* security against the receiver.

3.1 Information-Theoretically Secure OT

- Sender samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n)$ and sends pk to receiver.
- Receiver samples $s_0, s_1 \leftarrow \{0,1\}$ and sets $c_b = \mathsf{Enc}_{\mathsf{pk}}(s_b)$ and $c_{1-b} \leftarrow \mathsf{Samp}$, and sends c_0, c_1 to sender.
- Sender sets $s_i = \mathsf{Dec}_{\mathsf{sk}}(c_i)$ and sends $x_i \oplus s_i$.
- Receiver computes
$$x_b = s_b \oplus (s_b \oplus x_b).$$

Figure 4: Variant of OT.

Informally, the security against sender is dependent on the (computational) indistinguishability of determining c_{1-b} from an honestly sampled encryption and on the security of the encryption scheme, while the security against the receiver is information-theoretic since it cannot determine s_1.

We see another variant of OT, secure under the DDH assumption.

- Receiver samples $r \leftarrow \mathbb{Z}_q$ and sets $(h_0, h_1) = (g_0^r, g_1^{r+b})$. It sends (h_0, h_1).
- Sender samples $a_0, b_0, a_1, b_1 \leftarrow \mathbb{Z}_q$ and sets
$$c_i = (g_0^{a_i} g_1^{b_i}, h_0^{a_i} h_1^{b_1} / (i \cdot g_1^{b_1}) x_i)$$
and sends (c_0, c_1).
- Receiver parses c_b as (c^1, c^2) and computes $x_b = c^2/(c^1)^r$.

Figure 5: Variant of OT secure assuming DDH.

3.2 Dual-Mode Cryptosystem

A dual-mode cryptosystem serves as a generic 'toggling' mechanism to achieve information-theoretic OT against the sender or the receiver. A strict definition of dual-mode cryptosystem is given in [PVW08], along with a generic technique that realizes the OT functionality.

Intuitively, a dual-mode cryptosystem is a hybrid cryptosystem that has two modes: a messy mode and a decryption mode. It has several features:

1. The cryptosystem is initialized with a trusted setup that produces a pair (crs, t) where crs is the common random string and t is the trapdoor.
2. Whether the system is in messy or decryption mode is decided during setup. Given a crs, the adversary cannot tell which mode the system is operating in.
3. There are two encryption branches. KeyGen takes the branch $\sigma \in \{0, 1\}$ as input, and produces a general pk and an sk that corresponds to the particular branch.
4. During encryption, a branch b is specified as input. b could be or could not be σ.
5. **Messy Mode:** In this mode if branch $b \neq \sigma$, an encryption is lossy; all information about the plaintext is lost (if encrypted on $b = \sigma$, then it is a normal encryption). Given t, it is possible to obtain the σ that corresponds to pk.
6. **Decryption Mode:** In this mode, both branches are decryptable; however, you need the trapdoor to find a key tuple $(\mathsf{pk}, \mathsf{sk}_0, \mathsf{sk}_1)$ which allows you to decrypt. Furthermore, $(\mathsf{pk}, \mathsf{sk}_\sigma)$ are statistically indistinguishable from $\mathsf{KeyGen}(\sigma)$.

We can now proceed with a formalism.

$\Longrightarrow$ **Definition 3.1** (Dual Mode Cryptosystem)**.** *A dual-mode cryptosystem with message space* $\{0, 1\}^\ell$ *consists of a tuple of probabilistic algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{FindMessy}, \mathsf{TrapKeyGen})$ *that have the following properties.*

- $\mathsf{Setup}(1^\lambda, \mu) \to (\mathsf{crs}, t)$. *$\mu$ refers to the mode.*
- $\mathsf{KeyGen}(\sigma) \to (\mathsf{pk}, \mathsf{sk})$. *If in messy mode,* sk *corresponds to the branch σ.*
- $\mathsf{Enc}(\mathsf{pk}, b, m) \to c$. *$b$ is the branch.*
- $\mathsf{Dec}(\mathsf{sk}, c) \to m$.
- $\mathsf{FindMessy}(t, \mathsf{pk}) \to b$. *Given the trapdoor (and a possibly malformed public key), the output is the messy branch of* pk.
- $\mathsf{TrapKeyGen}(t) \to (\mathsf{pk}, \mathsf{sk}_0, \mathsf{sk}_1)$. *Outputs a key tuple which allows you to encrypt/decrypt on both branches.*

3.2.1 OT From Dual-Mode Cryptosystem

We now consider the OT of [PVW08], which almost trivially follows from the definition of the DMC. The protocol can be set up in either mode. In messy mode, the receiver security is computational and the sender is statistical; in decryption mode, the security properties are reversed.

Security. We see briefly how this scheme provides security.

- In messy mode, y_σ is a correct encryption of m_σ, but $y_{1-\sigma}$ is statistically unrelated to $m_{1-\sigma}$. Thus the receiver gets no information about $m_{1-\sigma}$ at all. The receiver has computational security since pk computationally hides σ.

Parameters: S has input $(m_0, m_1) \in \{0,1\}^\ell$, while receiver R has input $\sigma \in \{0,1\}$. Suppose that the system is setup in mode μ.

Protocol:

1. Both S and R query $\mathcal{F}^{\mu}_{\mathsf{crs}}$ with the appropriate session ID to get the same crs (neither of them receive t).
2. R computes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\sigma)$ and sends pk to S.
3. S gets pk and computes $y_b \leftarrow \mathsf{Enc}(\mathsf{pk}, b, m_b)$ and sends it to R.
4. R decrypts $\mathsf{Dec}(\mathsf{sk}, y_\sigma)$ to obtain m_σ.

Figure 6: Two-Round OT from a Dual-Mode Cryptosystem.

- In decryption mode, the trapdoor allows the production of a public key which is statistically indistinguishable from a $\mathsf{KeyGen}(\sigma)$ output, so pk statistically hides σ. However, sender security is computational, since a 'true' decryption key exists (the one output by t).

3.2.2 Assumptions for Dual-Mode Cryptosystem

A dual-mode cryptosystem can be constructed based on **DDH**, **DCR**, **QR** and **LWE**. Information about the constructions is in [PVW08]. In particular, the construction based on **DDH** mimics the OT of Figure 5.

3.3 Extending the Usefulness of OT

We now look at two techniques which allow subtle (more useful) variants of OT.

3.3.1 Domain Extension

This technique allows us to obtain OT for ℓ-bit strings from OT for λ-bit strings. Note that λ is the security parameter – hence, it has to be a reasonable key length.

- Sender samples $k_0, k_1 \leftarrow \{0,1\}^\lambda$ and sends it to $\mathcal{F}_{\mathsf{OT}}$.
- Receiver sends b to $\mathcal{F}_{\mathsf{OT}}$ and receives k_b.
- Sender sends $c_i = \mathsf{Enc}_{k_i}(m_i)$ for each $i \in \{0,1\}$.
- Receiver decrypts $m_b = \mathsf{Dec}_{k_b}(c_b)$.

Figure 7: OT Domain Extension.

3.3.2 1-out-of-N OT from 1-out-of-2 OT

For simplicity, we can assume $N = 2^k$ for some k. Suppose that the receiver wants m_α for some $|\alpha| = k$.

- Sender samples k_i^b for $b \in \{0, 1\}$ and $i \in [k]$, and submits (k_i^0, k_i^1) to $\mathcal{F}_{\mathsf{OT}}$.
- Receiver sends α_i to the ith OT.
- Sender sets

  $$c_\beta = m_\beta \oplus \bigoplus_{i=1}^{k} F_{k_{\beta_i}}(\beta)$$

  and sends each c_β.
- Receiver decrypts $m_\alpha = c_\alpha \oplus \bigoplus_{i=1}^{k} F_{k_{\alpha_i}}(\alpha)$.

Figure 8: 1-out-of-N OT from 1-out-of-2 OT.

4 OT Extension

As we have seen, it is possible to perform OT of strings of long length using OT for strings of shorter length and a PRG. We will now answer the question of whether it's possible to perform a greater *number* of OTs using a fewer number of OTs.

4.1 IKNP OT Extension

In this section we will describe the IKNP OT Extension protocol from [IKNP03]. The protocol works by extending λ pairs of m-bit OT to m pairs of ℓ-bit OT. A full description is below. We note by $\mathbf{OT}_b^a$ an a-pairs b-bit OT functionality.

Parameters: The Sender holds m pairs of ℓ-bit strings $\{(m_{i,b})\}$. The receiver holds m selection bits $r = (r_1, \ldots, r_m)$. $H : [m] \times \{0,1\}^\lambda \to \{0,1\}^\ell$ is a random oracle.

Protocol:

- Sender selects $s \xleftarrow{\$} \{0,1\}^\lambda$ and sends it to $\mathbf{OT}_m^\lambda$ (as the receiver).
- Receiver selects a random matrix

$$T = \begin{pmatrix} T_1 & T_2 & \ldots & T_\lambda \end{pmatrix}_{m\times\lambda} = \begin{pmatrix} T^1 \\ T^2 \\ \vdots \\ T^m \end{pmatrix}_{m\times\lambda}$$

  and sends the inputs $\{(T_i, r \oplus T_i)\}_{i\in[\lambda]}$ to $\mathbf{OT}_m^\lambda$ (as the sender).
- Denote by Q the matrix received by the sender, which is

$$Q = \begin{pmatrix} Q_1 & Q_2 & \ldots & Q_\lambda \end{pmatrix}_{m\times\lambda} = \begin{pmatrix} Q^1 \\ Q^2 \\ \vdots \\ Q^m \end{pmatrix}_{m\times\lambda}$$

- Sender sends

$$(y_{j,0}, y_{j,1}) = (x_{j,0} \oplus H(j, Q^j), x_{j,1} \oplus H(j, Q^j \oplus s)).$$

- For each $1 \le j \le m$, receiver outputs $y_{j,r_j} \oplus H(j, T^j)$.

Figure 9: The IKNP OT Extension Protocol.

Correctness. We will now see why this protocol works. Consider $r_i = 0$ for some i. It follows that $(r_i)_{j\in[\lambda]} \oplus T^i = T^i$, and so $T^i = Q^i$ (regardless of whatever s was). It immediately follows that $H(j, T^j) = H(j, Q^j)$.

On the other hand, if $r_i = 1$, then $Q^i = (s \cdot r) \oplus T^j = s \oplus T^j$. Correctness follows similarly.

Discussion. The protocol provides perfect security against a semi-honest (even malicious) sender and statistical security against a semi-honest receiver. The protocol is instantiated with a random oracle, but can be instantiated with a weaker primitive called a *correlation-robust hash function.*

$\Longrightarrow$ **Definition 4.1** (Correlation Robustness)**.** *An efficiently computable function* $h : \{0,1\}^* \to \{0,1\}$ *is said to be correlation robust if for any polynomial* $q(\cdot)$ *and PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\epsilon(\cdot)$ *such that*

$$|\Pr[\mathcal{A}(t_1, \dots, t_m, h(t_1 \oplus s), \dots, h(t_m \oplus s)) = 1] - \Pr[\mathcal{A}(U_{(\lambda+1)m}) = 1]| \leq \epsilon(\lambda)$$

where $m \leq q(\lambda)$*. The probability is taken over random and independent choices of* $s, t_i \xleftarrow{\$} \{0,1\}^\lambda$ *for* $i \in [m]$*.*

The authors also modify the scheme to achieve full security against a malicious receiver through a cut-and-choose mechanism, in which a the receiver and the sender execute k instances of the (committed) protocol in parallel and the sender randomly checks whether a certain number of instances were correctly performed (and aborts otherwise).

Parameters. The above protocol is secure as long as $m = 2^{o(\lambda)}$. Note that if, say, $m = 2^\lambda$, then the security of the correlation-robust hash function breaks down since this allows a 'repeat', i.e. $H(j, Q^j \oplus s)$ will be called twice for some value of j. We will see the impossibility of $2^{\Omega(\lambda)}$-pairs OT extension in Minicrypt along with other impossibility results below. This concludes that the protocol is asymptotically optimal.

Improvements. [BCG+19] achieves n pairs using $\log n$ OTs and the (computational) LPN (Learning Parity with Noise) assumption. [Roy22] presents a practical improvement on the protocol.

4.2 Feasibility of OT Extension

The protocol of [IKNP03] is not the only OT Extension protocol. Before this, [Bea96] showed that k OTs can be extended to k^c OTs making a non-black box use of a one way function. The protocol of [IKNP03] can gain up to superpolynomial OTs with a black-box use of an OWF, but makes use of a Random Oracle.

Open Problem. Is there a protocol for (subexponential) OT extension that makes black-box use of a One Way Function without a random oracle?

[LZ13] study the feasibility of OT Extension, and prove the following results.

$\Longrightarrow$ **Theorem 4.1** (Information-Theoretic OT Extension)**.** *If there exists an OT extension protocol from* n *to* $n+1$ *(with security in the presence of static semi-honest adversaries), then there exist one-way functions.*

This proves that OT cannot be extended information-theoretically. Note that while the protocol of [IKNP03] does not make explicit use of an OWF, it does use a random oracle (which implies OT extension).

⟹ **Theorem 4.2** (Adaptively Secure OT Extension)**.** *If there exists an OT extension protocol from n to $n+1$ that is secure in the presence of adaptive semi-honest adversaries, then there exists an oblivious transfer protocol that is secure in the presence of static semi-honest adversaries.*

This proves that any adaptive OT extension protocol involves constructing statically secure OT extension from scratch. The problem of constructing adaptively secure OT extension was solved in [BPRS17], though their protocol uses public-key cryptography.

⟹ **Theorem 4.3** (Extending Logarithmic OTs)**.** *If there exists an OT extension protocol from $f(n) = O(\log n)$ to $f(n) + 1$ that is secure in the presence of static malicious adversaries, then there exists an OT protocol that is secure in the presence of static malicious adversaries.*

This proves that any OT extension protocol that have exponential extension must make use of an exponential number of OTs itself.

4.2.1 OT Extension Implies One-Way Functions

We now provide a sketch of the proof of Theorem 4.1, as given in [LZ13].

⟹ **Lemma 4.1** (OT Extension with Polynomial Stretch)**.** *Let $f : \mathbb{N} \to \mathbb{N}$ be any polynomially-bounded function, and let n be the security parameter. If there exists a protocol π that is an OT-extension from $f(n)$ to $f(n) + 1$ that is secure in the presence of adaptive, malicious adversaries, then for every polynomial $p(\cdot)$ there exists an OT-extension protocol from $f(n)$ to $p(n)$ that is secure in the presence of adaptive malicious adversaries.*

Proof. We begin by noting that any OT-extension protocol π can be converted into an extension protocol π' in two phases:

- **Ideal OT**: In this phase, the parties make $f(n)$ calls to an ideal OT.
- **Computation**: In this phase, the parties use the OT calls from the previous phase to get $f(n) + 1$ OTs.

The protocol to get $p(n)$ OTs is simple: first, use the OTs from the first round to get $f(n) + 1$ OTs, and then *reuse* them $p(n)$-many times to get $p(n)$ OTs (each invocation adds one additional OT). The formal proof proceeds by hybrid argument. □

We will now use this fact in order to generate two polynomial-time constructable probability ensembles that are *statistically* distinguishable but *computationally* indistinguishable. The existence of such ensembles is equivalent to a one-way function, shown by [Gol90]. In particular, Goldreich shows that the existence of pseudorandom generators is equivalent to the existence of such ensembles.

Proof Sketch of Theorem 4.1. Let π be an $n \to 2n+1$ OT-extension. Consider the random variables $X_0, X_1, X'_0, X'_1, \Sigma$, where

- $\Sigma \in \{0,1\}^{2n+1}$ is a uniformly distributed string (representing the receiver's input).
- $X_i, X'_i \in \{0,1\}^{2n+1}$ are uniformly distributed under the constraint that $X^i_{\Sigma^i} = X'^i_{\Sigma_i}$ (representing the possible sender inputs – if Σ^i is b, then X_b and X'_b agree on the ith bit, otherwise they are independent).

Let $\text{TRANS}^\pi(x_0, x_1, \sigma)$ be the transcript of π on the sender inputs x_0 and x_1 and on receiver input σ (the transcript does *not* contain inputs to the ideal OT functionality, however). Define the probability ensembles

$$\mathcal{E}^0_n = (X_0, X_1, \Sigma, \text{TRANS}^\pi(X_0, X_1, \Sigma))$$
$$\mathcal{E}^1_n = (X'_0, X'_1, \overline{\Sigma}, \text{TRANS}^\pi(X_0, X_1, \Sigma))$$

The key idea is this: these ensembles must be *computationally* indistinguishable since the receiver should be completely unable to distinguish whether X_i or X'_i was used given the transcript, since otherwise receiver security of the OT will be violated. Similarly, Σ should be indistinguishable from $\overline{\Sigma}$, since the sender security will be violated. Note that the first 3 elements are indistinguishable simply by construction. The non-trivial computational indistinguishability is that of the transcript.

Note, however, that the transcript must be *statistically* distinct. This is because the transcript *has* to contain some meaningful information about the input distributions – the only way this is impossible is if *all* information were communicated by the ideal OT that is executed as a subprotocol. But this cannot be the case, since only n instances of OT were carried out; some information about the other $n+1$ has to be contained in this transcript. This provides an intuitive reason why the two are statistically indistinguishable. □

The full proof is contained in Section 3 of [LZ13].

5 The GMW Protocol

We now consider the first generic protocol for multi-party computation of arbitrary circuits, introduced by Goldreich, Micali and Wigderson in [MGW87] and [Gol04]. At any given time, the protocol proceeds with both parties having shares of the value of the circuit wires. At the end, the parties have shares of the output wire labels, and they can broadcast them.

5.1 The Protocol

The protocol proceeds in stages. Suppose that P_0's input is $x \in \{0,1\}^n$ and P_1's input is $y \in \{0,1\}^m$. Furthermore, we consider each gate to have two input wires.

1. P_0 samples $r \stackrel{\$}{\leftarrow} \{0,1\}^n$ and sets its share of the input to be $x \oplus r$, and sends r to P_1. P_1 does the same with a random string $s \stackrel{\$}{\leftarrow} \{0,1\}^m$.
2. When encountering gate C with input wire labels w_i and w_j, P_0 holds shares s_i^0 and s_j^0 and P_1 holds shares s_i^1 and s_c^1 with

$$s_c^b \oplus s_c^{1-b} = w_c$$

for $c \in \{i, j\}$.
3. P_0 and P_1 perform a $\mathsf{GateEVAL}$ and obtain shares s_k^0 and s_k^1 for output wire w_k.
4. Once the parties obtain shares of all the output gates, they broadcast these shares and reconstruct them to get the values of the output wires.

Figure 10: The GMW Protocol for secure computation of arbitrary circuits.

We now formalize the $\mathsf{GateEVAL}$ subprotocol that allows the parties to generate output shares of a gate.

5.1.1 NOT Gate

A NOT gate only has one input, so assume that the parties hold s_i^0 and s_i^1 as input shares. Before the protocol, the parties agree that P_0 will flip its input bit to $\overline{s_0^i}$, which is a share of the output.

5.1.2 XOR Gate

Note that the output of an XOR gate can be rearranged as in the following:

$$\begin{aligned} w_k &= w_i \oplus w_j \\ &= (s_i^0 \oplus s_i^1) \oplus (s_j^0 \oplus s_j^1) \\ &= (s_i^0 \oplus s_j^0) \oplus (s_i^1 \oplus s_j^1) \end{aligned}$$

Hence, P_b can simply set $s_k^b := s_i^b \oplus s_j^b$.

5.1.3 AND Gate

The $\mathsf{GateEVAL}$s for the NOT and the XOR gate were non-interactive, and hence revealed no information about their shares to the other party. However, this is not the case with the AND gate. The evaluation of this gate proceeds as follows.

First, P_0 calculates the value of w_k for all possible inputs of P_1. Set $S(a,b)$ to be the value of w_k given $s_i^1 = a$ and $s_j^1 = b$. Then P_0 samples $r \xleftarrow{\$} \{0,1\}$ and sets $s_k^0 = r$. It sends

$$M = \begin{pmatrix} r \oplus S(0,0) \\ r \oplus S(0,1) \\ r \oplus S(1,0) \\ r \oplus S(1,1) \end{pmatrix}$$

as input to $\mathcal{F}_{\text{1-out-of-4-OT}}$ as the sender. P_1 sends $s = (s_i^1, s_j^1)$ as receiver input, and sets s_k^1 as the OT output.

Note that by the security properties of OT, P_0 receives no information about P_1's share, while P_1 obtains no information about P_0's share since the output has a random mask.

6 Garbled Circuits

A List of Assumptions

We provide here a lookup table for a list of common assumptions.

A.1 Assumptions Based on Discrete-Log

A.2 Assumptions based on Reciprocity

A.3 Assumptions based on Coding

A.4 Assumptions based on Lattices

References

[BCG+19] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent ot extension and more. Cryptology ePrint Archive, Paper 2019/448, 2019. https://eprint.iacr.org/2019/448.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 479–488, 1996.

[BPRS17] Megha Byali, Arpita Patra, Divya Ravi, and Pratik Sarkar. Fast and universally-composable oblivious transfer and commitment scheme with adaptive security. *Cryptology ePrint Archive*, 2017.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, jun 1985.

[Gol90] Oded Goldreich. A note on computational indistinguishability. *Information Processing Letters*, 34(6):277–281, 1990.

[Gol04] Oded Goldreich. *Foundations of Cryptography, Volume 2*. Cambridge university press Cambridge, 2004.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 145–161, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 44–61, 1989.

[LZ13] Yehuda Lindell and Hila Zarosim. On the feasibility of extending oblivious transfer. In *Theory of Cryptography Conference*, pages 519–538. Springer, 2013.

[MGW87] Silvio Micali, Oded Goldreich, and Avi Wigderson. How to play any mental game. In *Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC*, pages 218–229. ACM New York, NY, USA, 1987.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, pages 554–571, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[Roy22] Lawrence Roy. Softspokenot: Communication–computation tradeoffs in ot extension. Cryptology ePrint Archive, Paper 2022/192, 2022. https://eprint.iacr.org/2022/192.