CS 355: TOPICS IN CRYPTOGRAPHY

DAVID WU

Abstract. Preliminary notes based on course material from Professor Boneh's Topics in Cryptography course (CS 355) in Spring, 2014. There are probably typos. **Last updated: June 19, 2014**

Contents

1. 4/1: Pseudorandom Generators

1.1. **Pseudorandom Generators.**

We begin with a discussion on pseudorandom generators (PRG).

Definition 1.1. Let $\lambda \in \mathbb{Z}^+$ be a security parameter. A PRG is an efficiently computable function $G_\lambda : \{0,1\}^{s(\lambda)} \to \{0,1\}^{n(\lambda)}$ where $n(\lambda) > s(\lambda)$. We refer to $s(\lambda)$ as the seed length and $n(\lambda)$ to be the output length. Note that when we say "efficiently computable", we mean computable in polynomial-time (possibly using randomness) with respect to the security parameter λ. We will also use PPT (probabilistic polynomial time) to describe "efficient" algorithms.

For notational convenience, we will drop the dependence on the security parameter λ in our exposition. An "old" (and insufficient) definition of security for a PRG specify that a PRG is secure if it passes a series of statistical tests. For instance, one such statistical test might to examine the absolute difference between the number of 0s and the number of 1s and compare the value to the expected difference ($\approx \sqrt{n}$). However, this definition is too weak to be suitable for cryptographic purposes. We give an improved definition by Yao that defines security relative to all possible statistical tests.

Definition 1.2. A PRG G is secure if no efficient statistical test can distinguish the output of the PRG from the output of a truly random element from the output space, that is, the following two distributions are indistinguishable:

$$\left[s \xleftarrow{\$} \{0,1\}^s : G(s)\right] \overset{c}{\approx} \left[r \xleftarrow{\$} \{0,1\}^n\right],$$

where we write $\mathcal{D}_0 \overset{c}{\approx} \mathcal{D}_1$ if two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are computationally indistinguishable (no efficient algorithm can distinguish $\mathcal{D}_0$ from $\mathcal{D}_1$ with non-negligible probability).

To be more concrete, we give a game based definition for the security of a PRG. For an algorithm $\mathcal{A}$, define the following two games:

- World 0 (pseudorandom): $W_0 := \Pr\left[s \xleftarrow{\$} \{0,1\}^s, r \leftarrow G(s) : \mathcal{A}(r) = 1\right]$.
- World 1 (truly random): $W_1 := \Pr\left[r \xleftarrow{\$} \{0,1\}^n : \mathcal{A}(r) = 1\right]$.

In terms of these games, the PRG advantage $\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G} = |W_0 - W_1|$ is the absolute difference between W_0 and W_1. Note that the PRG advantage is also a function in the security parameter λ. With this, we can define the notion of a secure PRG, and show that the existence of a secure PRG necessarily implies $\mathsf{P} \neq \mathsf{NP}$.

Definition 1.3. A PRG G is secure if for all efficient algorithms $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G}$ is negligible in the security parameter λ. A function $f(\lambda)$ is negligible in the parameter λ if $f = \lambda^{-\omega(1)}$.

Theorem 1.4. *If secure* PRGs *(as defined in Definition 1.3) exist, then* $\mathsf{P} \neq \mathsf{NP}$.

Proof. We proceed by contradiction. Suppose $\mathsf{P} = \mathsf{NP}$, and that there exists a secure PRG $G : \{0,1\}^s \to \{0,1\}^n$ with $n > s$. We can construct an adversary that can distinguish the output of G from a truly random string with advantage of at least $1 - \frac{1}{2^{n-s}} \geq \frac{1}{2}$. In the PRG game, the adversary is given a string $y = \{0,1\}^n$ and it must decide whether it is in World 0 or World 1. The problem of deciding whether there exists $x \in \{0,1\}^s$ such that $G(x) = y$ is an NP problem (with witness x). Since $\mathsf{P} = \mathsf{NP}$, there exists an efficient algorithm $\mathcal{B}$ that on input y, decides whether there exists an x such that $G(x) = y$. We use $\mathcal{B}$ to construct a PRG adversary $\mathcal{A}$ that wins the PRG game with advantage at least $\frac{1}{2}$. Adversary $\mathcal{A}$ works as follows:

PRG Adversary $\mathcal{A}$
On input $y \in \{0,1\}^n$, run $\mathcal{B}$ on y.
If $\mathcal{B}(y) = 0$ (there is no preimage $x \in \{0,1\}^s$ such that $G(x) = y$), output 1. Otherwise output 0.

Consider the advantage of this adversary. In World 0, the adversary is given a string in the image of G, so algorithm $\mathcal{B}$ always outputs 1. Thus, $\mathcal{A}$ always outputs 0 in World 0, so $W_0 = 0$. In World 1, the adversary is given a truly random string $y \in \{0,1\}^n$. With probability at most $\frac{2^s}{2^n} = \frac{1}{2^{n-s}}$, y is in the image of G. Thus, $\mathcal{B}(y) = 1$ with probability at most $\frac{1}{2^{n-s}}$. Thus, $\mathcal{A}$ outputs 0 with probability at most $\frac{1}{2^{n-s}}$. Thus, $W_1 \geq 1 - \frac{1}{2^{n-s}}$. Finally,

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G} = |W_0 - W_1| = 1 - \frac{1}{2^{n-s}} \geq \frac{1}{2},$$

which is certainly non-negligible. Thus, G is not secure, and we have our desired contradiction. □

1.2. **The Blum-Micali Construction.** The previous theorem shows that in order to construct PRGs, we will have to rely on additional hardness assumptions. For now, we will assume the existence of a one-bit secure PRG, that is, a PRG $G : \{0,1\}^s \to \{0,1\}^{s+1}$. In the subsequent lecture, we will show that if one-way permutations exist, then one-bit PRGs exist. Given a one-bit PRG, we show how to construct a many-bit PRG via the Blum-Micali construction [BM82]. We describe this construction in abstract terms. Suppose we have a PRG $G : S \to R \times S$. In the case of a one-bit secure PRG, $S = \{0,1\}^s$ and $R = \{0,1\}$. Our goal is to construct a PRG $G_{\text{BM}} : S \to R^n \times S$ for any $n > 0$. The Blum-Micali construction proceeds as follows:

Blum-Micali PRG Construction
(1) The generator is given as input a seed $s_0 \in S$.
(2) For $r = 1, \dots, n$, compute $(s_i, r_i) \leftarrow G(s_{i-1})$.
(3) Output $(r_1, \dots, r_n, s_n) \in R^n \times S$.

Example 1.5 (Blum-Blum-Shub (BBS))**.** As an example, of a PRG, we describe the Blum-Blum-Shub (BBS) PRG [BBS86]. Let N be an RSA modulus. Then, assuming factoring is hard, we can construct a secure one-bit PRG $G : \mathbb{Z}_N \to \mathbb{Z}_N \times \{0,1\}$.

$$G_N(s) = \left[s^2 \mod N, \text{lsb}(s)\right].$$

Here, $\text{lsb}(\cdot)$ denotes the least-significant bit of the input. As a technicality, we note that the two primes p, q in the RSA modulus should be congruent to 3 (mod 4). This is necessary to guarantee that each quadratic residue has a square root that is itself a quadratic residue.

If we apply the Blum-Micali construction to the BBS PRG, then we obtain a secure PRG for arbitrary lengths. We now prove that the Blum-Micali construction yields a secure PRG.

Theorem 1.6. *If $G : S \to R \times S$ is a secure PRG, then $G_{\mathsf{BM}} : S \to R^n \times S$ formed using the Blum-Micali construction is a secure PRG. Specifically, for all PRG adversaries $\mathcal{A}$ for G_{BM}, there exists a PRG adversary $\mathcal{B}$ for G such that*

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A}, G_{\text{BM}}} \le n \cdot \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B}, G}.$$

Proof. We defer the proof of this statement to next lecture. □

2. 4/3: Pseudorandom Generators

2.1. **The Blum-Micali Construction.** In the previous lecture, we presented the Blum-Micali construction [BM82]. In this lecture, we complete its proof of security. First, we introduce the notion of computational indistinguishability.

Definition 2.1. Let $(\Omega, P_0), (\Omega, P_1)$ be two distributions over Ω. For an algorithm $\mathcal{A}$, define the distinguishing advantage of $\mathcal{A}$ with respect to P_0, P_1 as

$$\mathsf{Adv}_{\mathcal{A},(P_0,P_1)} := \left|\Pr\left[x \leftarrow P_0 : \mathcal{A}(x) = 1\right] - \Pr\left[x \leftarrow P_1 : \mathcal{A}(x) = 1\right]\right|.$$

Then, P_0 and P_1 are computationally indistinguishable if for all efficient algorithms $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A},(P_0,P_1)}$ is negligible. We denote this by $P_0 \overset{c}{\approx} P_1$.

We can also define a notion of statistical indistinguishability. First, we can define the statistical distance between two distributions

$$\Delta(P_0; P_1) = \frac{1}{2} \sum_{x \in \Omega} \left|P_0(x) - P_1(x)\right|.$$

Definition 2.2. Two distributions P_0, P_1 are statistically indistinguishable (denoted $P_0 \overset{s}{\approx} P_1$) if $\Delta(P_0; P_1)$ is negligible.

Theorem 2.3. *For all algorithms $\mathcal{A}$ (including computationally unbounded ones),*

$$\mathsf{Adv}_{\mathcal{A},(P_0,P_1)} \le \Delta(P_0; P_1).$$

Proof. For simplicity, we prove a weaker version of the theorem, that is, for all *deterministic* algorithms $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A},(P_0,P_1)} \leq 2 \cdot \Delta(P_0; P_1)$. To see this, let $S_\mathcal{A} = \{x \in \Omega : \mathcal{A}(x) = 1\}$. Then,

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A},(P_0,P_1)} = |P_0(S_\mathcal{A}) - P_1(S_\mathcal{A})| = \left|\sum_{x\in S_\mathcal{A}} P_0(x) - P_1(x)\right| &\leq \sum_{x \in S_\mathcal{A}} |P_0(x) - P_1(x)| \\ &\leq \sum_{x\in\Omega} |P_0(x) - P_1(x)| \\ &\leq 2 \cdot \Delta(P_0; P_1).\end{aligned}$$

With a more careful analysis, we can generalize this result to apply to randomized algorithms and eliminate the factor of two in the bound. □

Example 2.4. Take integers m, n with $m > n$. We consider the statistical difference between the uniform distribution over the integers between 0 and $m-1$ and the uniform distribution over the integers between 0 and $n-1$:

$$\Delta(m;n) := \Delta\left(r_0 \xleftarrow{\$} \{0,\ldots,m-1\}; r_1 \xleftarrow{R} \{0,\ldots,n-1\}\right) = \frac{1}{2}\left[\sum_{x=0}^{n-1}\left(\frac{1}{n} - \frac{1}{m}\right) + \sum_{x=n}^{m-1}\frac{1}{m}\right] = \frac{m-n}{m}.$$

We describe an application of this result. Consider a secure PRG $G : S \to \{0,\ldots,2^\ell - 1\}$. We can construct a secure PRG $G' : S \to \{0,\ldots,n-1\}$ for $n < 2^\ell$: simply let $G'(s) = G(s) \pmod n$. To show that this is secure, we prove the following theorem.

Theorem 2.5. *Let $G : S \to \{0,\ldots,2^\ell - 1\}$ be a secure* PRG*, and let $G' : S \to \{0,\ldots,n-1\}$ for $n < 2^\ell$ where $G'(s) = G(s) \pmod n$. Then, for all efficient adversaries $\mathcal{A}$ against G', there exists an efficient adversary $\mathcal{B}$ against G such that*

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G'} \leq \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B},G} + \frac{n}{2^\ell}.$$

Proof. Let $m = 2^\ell$ and set $k = \lfloor \frac{m}{n} \rfloor$. Then, given an adversary $\mathcal{A}$ for G', we construct an adversary $\mathcal{B}$ as follows. On input $x \in \{0,\ldots,m-1\}$, output $\mathcal{A}(x \pmod n)$. Then, in World 0, we have

$$\begin{aligned}p_0 := \Pr\left[s \xleftarrow{\$} S, r \xleftarrow{\$} G'(s) : \mathcal{A}(r) = 1\right] &= \Pr\left[s \xleftarrow{\$} S, r \leftarrow G(s) \bmod n : \mathcal{A}(r) = 1\right] \\ &= \Pr\left[s \xleftarrow{\$} S, r \leftarrow G(s) : \mathcal{B}(r) = 1\right],\end{aligned}$$

and in World 1, we have

$$\begin{aligned}p_1 := \Pr\left[r \xleftarrow{\$} \{0,\ldots,n-1\} : \mathcal{A}(r) = 1\right] &= \Pr\left[r \xleftarrow{\$} \{0,\ldots,kn-1\} : \mathcal{A}(r \bmod n) = 1\right] \\ &= \Pr\left[r \xleftarrow{\$} \{0,\ldots,n-1\} : \mathcal{B}(r) = 1\right].\end{aligned}$$

To bound the absolute difference between the two probabilities, we define a new distribution

$$p^\star := \Pr\left[r \xleftarrow{\$} \{0,\ldots,m-1\} : \mathcal{B}(r) = 1\right].$$

Then,

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G'} = |p_0 - p_1| = |p_0 - p^\star + p^\star - p_1| \leq |p_0 - p^\star| + |p^\star - p_1|.$$

We can bound $|p_1 - p^\star|$ using the statistical distance between a uniform distribution over $\{0,\ldots,n-1\}$ and a uniform distribution over $\{0,\ldots,m-1\}$. Using the result from Example 2.4, this becomes

$$|p_1 - p^\star| \leq \frac{m - kn}{m} \leq \frac{n}{m}.$$

Moreover, we note that $|p_0 - p^\star| \leq \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B},G}$, and so

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G'} \leq \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B},G} + \frac{n}{m} = \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B},G} + \frac{n}{2^\ell}.$$

□

With this preparation, we revisit the proof of security of the the Blum-Micali generator introduced in Section 1.2.

Theorem 2.6. *If G is a secure* PRG *and n is bounded, then G_{BM} is a secure* PRG *where G_{BM} is the* PRG *constructed by n iterations of the Blum-Micali construction. Specifically, we show that for all efficient adversaries $\mathcal{A}$ for G_{BM}, there exists an adversary $\mathcal{B}$ for G such that*

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G_{\mathrm{BM}}} \le n \cdot \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B},G}.$$

Proof. We use a hybrid argument. Let $\mathcal{A}$ be an adversary for G_{BM}. We define $n+1$ distributions P_j on $R^n \times S$:

$$P_j := \left[\begin{array}{l} r_1,\dots,r_j \xleftarrow{\$} R,\ s_j \xleftarrow{\$} S \\ \text{for } i = j+1,\dots,n,\ (s_i, r_i) \leftarrow G(s_{i-1}) \\ \text{output } (r_1,\dots,r_n,s_n) \end{array}\right].$$

In words, the first j components of an output of distribution P_j are truly random and the remaining components are pseudorandom. We show that if G is secure, then

$$P_0 \overset{c}{\approx} P_1 \overset{c}{\approx} \cdots \overset{c}{\approx} P_n,$$

which will imply that $P_0 \overset{c}{\approx} P_n$. Let $p_j := \Pr\left[z \xleftarrow{\$} P_j : \mathcal{A}(z) = 1\right]$. Observe that in P_0, all of the components are pseudorandom. This is precisely the distribution of outputs adversary $\mathcal{A}$ sees in World 0, so $W_0 = p_0$. Analogously, in P_n, all of the components are chosen uniformly at random, which is identical to $\mathcal{A}$'s view in World 1, and so $W_1 = p_n$. Thus, we now have that $|p_0 - p_n| = \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G_{\mathrm{BM}}}$.

Claim 2.7. For $j = 0,\dots,n-1$, there exists $\mathcal{B}_j$ such that $|p_j - p_{j+1}| = \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B}_j,G}$.

Proof. We describe algorithm $\mathcal{B}_j$. Since $\mathcal{B}_j$ is an adversary for G, it is given, as input, $(r^\star, s^\star) \in R \times S$.

Distinguishing Algorithm $\mathcal{B}_j$

(1) Generate $\hat{r}_1,\dots,\hat{r}_j \xleftarrow{\$} R$ and $s_{j+1} \xleftarrow{\$} S$.
(2) For $i = j+2,\dots,n$, $(r_i, s_i) \leftarrow G(s_{i-1})$.
(3) Output $\mathcal{A}\left(\underbrace{\hat{r}_1,\dots,\hat{r}_j, r^\star, r_{j+2},\dots,r_n,s_n}_{z}\right)$.

We consider what happens in the two worlds. In World 0, $(r^\star, s^\star) \leftarrow G(s)$, so z is a uniform random sample from P_j. In World 1, $(r^\star, s^\star) \xleftarrow{\$} R \times S$, and so z is a uniform random sample from P_{j+1}. By definition then,

$$|p_j - p_{j+1}| = \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B}_j,G}.$$

□

Using the claim, we have that

$$\begin{aligned} \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G_{\mathrm{BM}}} &= |p_0 - p_n| = |p_0 - p_1 + p_1 - p_2 + \cdots + p_{n-1} - p_n| \\ &\le |p_0 - p_1| + |p_1 - p_2| + \cdots + |p_{n-1} - p_n| \\ &\le \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B}_0,G} + \cdots + \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B}_{n-1},G}. \\ &= \sum_{j=0}^{n-1} \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B}_j,G} \end{aligned}$$

Since the advantage is always nonnegative, then there is some $0 \le j < n$ such that

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B}_j,G} \ge \frac{1}{n} \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G_{\mathrm{BM}}}.$$

Rearranging terms, we have that for every adversary $\mathcal{A}$ for G_{BM}, there is a corresponding adversary $\mathcal{B}$ for G such that $\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G_{\mathsf{BM}}} \le n \cdot \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B},G}$. Thus, if we assume that G is a secure PRG, then $\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B},G_{\mathsf{BM}}}$ is negligible for all efficient algorithms $\mathcal{B}$. Assuming n is polynomial (in the security parameter), then G_{BM} is correspondingly a secure PRG. □

3. 4/8: Pseudorandomness

3.1. The Goldreich-Goldwasser-Micali (GGM) Construction. Recall from our previous lecture that two distributions P_0 and P_1 are computationally indistinguishable ($P_0 \overset{c}{\approx} P_1$) if no efficient algorithm can distinguish them, and P_0 and P_1 are statistically indistinguishable ($P_0 \overset{s}{\approx} P_1$) if the statistical difference between the two distributions is negligible. Then, a PRG $G : S \to R$ is secure if and only if $\left\{s \xleftarrow{\$} S : G(s)\right\} \overset{c}{\approx} \left\{r \xleftarrow{\$} R : r\right\}$. Using the Blum-Micali construction, we demonstrated how to take a PRG $G : S \to R \times S$ to obtain a generator $G_{\mathsf{BM}} : S \to R^n \times S$. An alternative construction for building a PRG with a greater expansion factor is the Goldreich-Goldwasser-Micali (GGM) construction [GGM86]. In the GGM construction, we transform a PRG $G : S \to S \times S$ into a PRG $G^{(d)}_{\mathrm{GGM}} : S \to S^{(2^d)}$. The GGM construction works as follows:

GGM PRG Construction
(1) Let $s_{0,0} \in S$ be the seed.
(2) For $i = 1, \ldots, d$, and $k = 0, \ldots, 2^{i-1}$, let $(s_{i,2k}, s_{i,2k+1}) \leftarrow G(s_{i-1,k})$.
(3) Output $\left(s_{d,0}, \ldots, s_{d,2^d-1}\right)$.

Theorem 3.1. *For any constant d, if G is a secure PRG, then $G^{(d)}_{\mathrm{GGM}}$ is a secure PRG, that is, for every PRG adversary $\mathcal{A}$ against $G^{(d)}_{\mathrm{GGM}}$, there exists a PRG adversary $\mathcal{B}$ against G such that*

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G^{(d)}_{\mathrm{GGM}}} \le 2^d \cdot \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B},G}.$$

Proof. We proceed using a hybrid argument. Since G is a secure PRG, we have that $(s_{1,0}, s_{1,1}) \overset{c}{\approx} (\hat{s}_{1,0}, \hat{s}_{1,1})$ where $(\hat{s}_{1,0}, \hat{s}_{1,1}) \xleftarrow{\$} S \times S$. Using an inductive argument, assume that $\left(s_{i,0}, \ldots, s_{i,2^i-1}\right) \overset{c}{\approx} \left(\hat{s}_{i,0}, \ldots, \hat{s}_{i,2^i-1}\right)$ where $\left(\hat{s}_{i,0}, \ldots, \hat{s}_{i,2^i-1}\right) \xleftarrow{\$} S^{2^i-1}$. Then, invoking the security of G (when applied to each of $\hat{s}_{i,0}, \ldots, \hat{s}_{i,2^i-1}$, we conclude that

$$\left(s_{i+1,0}, \ldots, s_{i+1,2^{i+1}-1}\right) \overset{c}{\approx} \left(\hat{s}_{i+1,0}, \ldots, \hat{s}_{i+1,2^{i+1}-1}\right),$$

which concludes the claim. ☐

3.2. Derandomization via a PRG. One of the open problems in complexity theory is whether $\mathsf{BPP} \subseteq \mathsf{P}$ (in words, this is the question of whether polynomial-time randomized algorithms can be derandomized). We show how we might use a PRG to derandomize algorithms. Let $\mathcal{A}(x, r)$ be an efficient randomized algorithm for some problem. Here, $x \in X$ is the input (from an input space X) and r is the random bits (drawn from some space R) used by $\mathcal{A}$. Suppose moreover that $\mathcal{A}$ succeeds with probability at least $\frac{1}{3}$ (we will denote this by saying $\mathcal{A}(x, r) = \mathsf{success}$, even though $\mathcal{A}(x, r)$ may not actually output "$\mathsf{success}$"). Specifically, we have that

$$\forall x \in X : \Pr\left[r \xleftarrow{\$} R : \mathcal{A}(x, r) = \mathsf{success}\right] \ge \frac{1}{3}.$$

Using a PRG $G : S \to R$, we can construct a deterministic algorithm $\mathcal{B}$ that solves the same problem. Specifically, we construct $\mathcal{B}$ as follows:

Derandomized Version of $\mathcal{A}$
(1) For all $s \in S$, if $\mathcal{A}(x, G(s)) = \mathsf{success}$, output $G(s)$.
(2) Otherwise, output fail.

Claim 3.2. If G is a secure PRG, then for all $x \in X$, $\mathcal{B}(x) \ne \mathsf{fail}$.

Proof. Suppose there is an $x \in X$ such that $\mathcal{B}(x) = \mathsf{fail}$. We use this fact to construct a distinguisher $\mathcal{D}_x$ for G:

$$\mathcal{D}_x(r \in R) = \begin{cases} 1 & \mathcal{A}(x, r) = \mathsf{success} \\ 0 & \text{otherwise.} \end{cases}$$

Consider the PRG advantage of $\mathcal{D}_x$. In World 0, $\mathcal{D}_x$ is given the output $G(s)$ (for some uniformly sampled seed) of the PRG, and by construction of x, $\mathcal{A}(x, r) = \mathcal{A}(x, G(s))$ will output fail. Thus, $\mathcal{D}_x(r \in R)$ outputs 1 with probability

0 in World 0. In World 1, $\mathcal{D}_x$ is given a truly random element of R, so $\mathcal{A}(x, r)$ outputs 1 with probability at least $\frac{1}{3}$. The PRG advantage of $\mathcal{D}_x$ is then

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{D}_x,G} \geq \left|0 - \frac{1}{3}\right| = \frac{1}{3}.$$

$\square$

Observe moreover that the runtime of algorithm $\mathcal{B}$ is $|S| \times \mathsf{RunTime}(\mathcal{A})$, and so if $|S|$ was polynomial in the input size of the problem, then we can derandomize all polynomial-time randomized algorithms. However, we do not have any construction of a secure PRG (and as mentioned previously, doing so would imply $\mathsf{P} = \mathsf{NP}$). In complexity theory, we have ways of building PRGs secure against weak adversaries (e.g., algorithms in L, that is, algorithms requiring logarithmic space). These results in turn imply that L can be derandomized.

3.3. Goldreich-Levin Theorem.

In this section, we demonstrate a classic result showing how to construct PRGs from one-way permutations (OWP). First, recall the definitions of one-way permutations and one-way functions:

Definition 3.3. A function $f : X \to X$ is a one-way permutation (OWP) if f is efficiently computable and for all efficient algorithms $\mathcal{A}$, $\Pr\left[x \xleftarrow{\$} X : \mathcal{A}(f(x)) = x\right] < \mathsf{negl}$.

Definition 3.4. A function $f : X \to Y$ is a one-way function (OWF) if f is efficiently computable and for all efficient algorithms $\mathcal{A}$, $\Pr\left[x \xleftarrow{\$} X : f\left(\mathcal{A}(f(x))\right) = f(x)\right] < \mathsf{negl}$.

Example 3.5. Let p be a prime. Then, the following function f is (believed to be) a one-way permutation over $\{1, \ldots, p-1\}$, where $f(x) = g^x \pmod p$. This is effectively the statement of the hardness of computing discrete logarithms in $\mathbb{Z}_p^*$.

Definition 3.6. Let $f : X \to Y$ be a OWF. Then, $h : X \to R$ is a hard-core (predicate) for f if

$$\left\{x \xleftarrow{\$} X : (f(x), h(x))\right\} \overset{c}{\approx} \left\{x \xleftarrow{\$} X, r \xleftarrow{\$} R : (f(x), r)\right\}.$$

Intuitively, a hard-core for f if it is the case that even given $f(x)$, any efficient adversary cannot distinguish $h(x)$ from random. Very often, we will speak of hard-core bits; this is the case where $R = \{0, 1\}$. We give a few examples below.

Example 3.7. Let $g \in \mathbb{G}$ be a generator of a group of prime order q. Let $f : \mathbb{Z}_q^* \to \mathbb{G}$ be the function $f(x) = g^x$. If f is a OWF, then the function $h(x) = \mathsf{lsb}(x)$ is a hard-core bit of f, where $\mathsf{lsb}(x)$ denotes the least significant bit of x.

Proof. Suppose there exists an algorithm $\mathcal{A}$ that on input g^x computes $\mathsf{lsb}(x)$. For simplicity, suppose that $\mathcal{A}$ always works. Then, using $\mathcal{A}$, we construct an algorithm $\mathcal{B}$ that inverts f (that is, solves discrete log in $\mathbb{G}$). Let $b_n b_{n-1} \cdots b_1$ denote the binary representation of x. Then, algorithm $\mathcal{B}$ proceeds as follows:

```
Algorithm to Compute Discrete Log in $\mathbb{G}$
(1) Set $z_1 \leftarrow g^x$.
(2) For each $i = 1, \ldots, n$:
    (a) Output $b_i = \mathcal{A}(z_i)$. Then, compute $z_{i+1}$ as
        $$z_{i+1} \leftarrow \begin{cases} z_i^{(q+1)/2} & b_i = 0 \\ \left(\frac{z_i}{g}\right)^{(q+1)/2} & b_i = 1. \end{cases}$$
```

We claim that the above algorithm computes $x = b_n b_{n-1} \cdots b_1$. For notational convenience, define $x_i = b_n b_{n-1} \cdots b_i$. It suffices to show that $z_i = g^{x_i}$. Correctness then follows since $\mathcal{A}(z_i)$ yields $\mathsf{lsb}(x_i) = b_i$. We proceed by induction on i. The base case $i = 1$ follows by definition of z_1. Supposing that $z_i = g^{x_i}$, consider z_{i+1}. If $b_i = 0$, then

$$z_{i+1} = z_i^{(q+1)/2} = \left(g^{x_i}\right)^{(q+1)/2} = \left(g^{x_i/2}\right)^{q+1} = g^{x_i/2},$$

using the fact that g is of order q in the last equality. Since $b_i = 0$, x_i is even, and thus divisible by 2. Moreover, $x_i/2 = b_n b_{n-1} \cdots b_{i+1} = x_{i+1}$. Thus, $z_{i+1} = g^{x_{i+1}}$, as desired. The case where $b_i = 1$ is handled similarly:

$$z_{i+1} = \left(\frac{z_i}{g}\right)^{(q+1)/2} = \left(g^{(x_i-1)/2}\right)^{q+1} = g^{(x_i-1)/2}.$$

In this case, $b_i = 1$, so x_i is odd, and thus $x_i - 1$ is divisible by 2. Finally, $(x_i - 1)/2 = b_n b_{n-1} \cdots b_{i+1} = x_{i+1}$, so $z_{i+1} = g^{x_{i+1}}$. By induction on i, we conclude that for all $i = 1, \ldots, n$, $z_i = g^{x_i}$, so the claim follows. Thus, we have demonstrated that if there exists an efficient algorithm $\mathcal{A}$ that always outputs $\mathsf{lsb}(x)$ on input g^x, there exists an efficient algorithm that computes x given $f(x) = g^x$. Using a boosting argument and showing a random self-reduction for computing the least significant bit, a similar argument holds when algorithm $\mathcal{A}$ outputs $\mathsf{lsb}(x)$ with probability $\frac{1}{2} + \varepsilon$. Thus, we conclude that if f is a OWF, then, then $\mathsf{lsb}(x)$ is a hard-core bit for f. □

Example 3.8. Let p be a prime and let $\mathbb{G} = \mathbb{Z}_p^*$ with generator g. Consider a variant of the most significant bit function defined as

$$\mathsf{msb}(x) = \begin{cases} 0 & x < \frac{p-1}{2} \\ 1 & \text{otherwise.} \end{cases}$$

Take the function $f(x) = g^x \in \mathbb{Z}_p^*$. If f is a OWP, then, $\mathsf{msb}(x)$ is hard-core for f. Note that we require this particular definition for the most significant bit because if p is not a power of two, there will be some bias in the most significant bit.

Proof. We proceed by contradiction. Suppose there is an algorithm $\mathcal{A}$ that on input g^x outputs $\mathsf{msb}(x)$. As before, we assume that $\mathcal{A}$ always works. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that inverts f. As before, write the binary representation of x as $x = b_n b_{n-1} \cdots b_1$. We begin by describing how to compute the least significant bit $\mathsf{lsb}(x)$ of x. Given g^x, consider the quantity $(g^x)^{\frac{p-1}{2}}$. If x is even, then

$$(g^x)^{\frac{p-1}{2}} = \left(g^{\frac{x}{2}}\right)^{p-1} = 1 \in \mathbb{Z}_p^*,$$

since $\left|\mathbb{Z}_p^*\right| = p - 1$. Conversely, if x is odd, $(g^x)^{\frac{p-1}{2}} \neq 1 \in \mathbb{Z}_p^*$. Thus, we can efficiently compute $\mathsf{lsb}(x)$ given g^x. Our algorithm $\mathcal{B}$ to invert f then works as follows:

Algorithm to Compute Discrete Log in $\mathbb{Z}_p^*$

```
(1) Set $z_1 \leftarrow g^x$.
(2) For each $i = 1, \ldots, n$:
    (a) Write $z_i = g^{\hat{x}}$. Compute $b_i = \mathsf{lsb}(\hat{x})$ using $z_i$ by first computing $t \leftarrow z_i^{\frac{p-1}{2}} \in \mathbb{Z}_p^*$. If $t = 1$, then
        $\mathsf{lsb}(\hat{x}) = 0$, otherwise, $\mathsf{lsb}(\hat{x}) = 1$.
    (b) If $b_i = 1$, then set $\hat{z}_i \leftarrow \frac{z_i}{g}$. Let $y_0$ and $y_1$ be the two square roots of $\hat{z}_i$. If $\mathcal{A}(y_0) = 0$, then set
        $z_{i+1} \leftarrow y_0$. Otherwise, $z_{i+1} \leftarrow y_1$.
```

We claim that this algorithm computes $x = b_n b_{n-1} \cdots b_1$. For notational convenience, we again define $x_i = b_n b_{n-1} \cdots b_i$. It suffices to show that $z_i = g^{x_i}$, since at each iteration, we output the least significant bit of x_i. We proceed by induction on i. The base case $i = 1$ holds by construction. Now, supposing that $z_i = g^{x_i}$, consider z_{i+1}. By construction,

$$\hat{z}_i = g^{b_n b_{n-1} \cdots b_{i+1} 0} = (g^{x_{i+1}})^2 = \left(g^{\frac{p-1}{2} + x_{i+1}}\right)^2,$$

where we again have used the fact that $\left|\mathbb{Z}_p^*\right| = p - 1$. Thus, the two square roots of $\hat{z}_i$ in $\mathbb{Z}_p^*$ are $y_0 = g^{x_{i+1}}$ and $y_1 = g^{\frac{p-1}{2} + x_{i+1}}$,. By definition of the msb function, we have $\mathsf{msb}(y_0) = 0$ while $\mathsf{msb}(y_1) = 1$. Therefore, we can distinguish between the two square roots of $\hat{z}_i$. By setting z_{i+1} to the root whose most significant bit is 0, we have $z_{i+1} = g^{x_{i+1}}$, as desired. By induction on i, we have that $z_i = g^{x_i}$, and the claim follows. Thus, if there exists an efficient algorithm $\mathcal{A}$ that always outputs $\mathsf{msb}(x)$ on input g^x , there exists an efficient algorithm that computes x given $f(x) = g^x$. Using a boosting argument, we can extend this argument to the case where $\mathcal{A}$ only succeeds with probability $\frac{1}{2} + \varepsilon$. Thus, if f is a OWP, then $\mathsf{msb}(x)$ is hard-core for f in $\mathbb{Z}_p^*$. □

Before continuing to the next example, we note a critical difference in the two examples above. In the case where $\mathbb{G}$ is a *prime* order group with generator g, the $\mathsf{lsb}(x)$ is a hard-core for $f(x) = g^x$. However, if $\mathbb{G} = \mathbb{Z}_p^*$ (which is not prime order), we saw that there was an efficient algorithm for computing $\mathsf{lsb}(x)$ from g^x. In the latter scenario then, the least significant bit is *not* hard-core.

Example 3.9. Let $N = pq$ be an RSA modulus and consider the function $f : \mathbb{Z}_N \to \mathbb{Z}_n$ where $f(x) = x^e$. Then, if f is a OWP, the function $h(x) = \mathsf{lsb}(x)$ is a hard-core bit for f.

We now show how to construct PRGs from OWFs. We begin by showing that if we begin a OWF with a corresponding hard-core bit, we trivially obtain a PRG. We then show that any OWF can be transformed into an OWF with a hard-core bit (Goldreich-Levin Theorem).

Lemma 3.10. *Let $f : X \to X$ be a OWP and suppose that $h : X \to R$ is hard-core for f. Then, $G : X \to X \times R$ where $G(s) = (f(s), h(s))$ is a secure PRG.*

Proof. By definition of h being a hard-core for f and the fact that f is a permutation,

$$\left\{s \xleftarrow{\$} X : (f(s), h(s))\right\} \overset{c}{\approx} \left\{s \xleftarrow{\$} X, r \xleftarrow{\$} R : (f(s), r)\right\} \overset{s}{\approx} \left\{s \xleftarrow{\$} X, r \xleftarrow{\$} R : (s, r)\right\}.$$

□

Theorem 3.11 (Goldreich-Levin [GL89])**.** *Let $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ be a OWP. Define the augmented function $g : \mathbb{Z}_2^{2n} \to \mathbb{Z}_2^{2n}$ where $g(x, r) = (f(x), r)$. Then,*

$$B(x, r) = \langle x, r\rangle = \sum_{i=1}^{n} x_i r_i \in \mathbb{Z}_2$$

is a hard-core for g. Note that this bit is also referred to as the Goldreich-Levin bit.

Proof. See subsequent lecture (Theorem 4.1). □

4. 4/10: Pseudorandomness and Commitment Schemes

4.1. Goldreich-Levin Theorem.

Recall from the previous lecture that if $f : X \to Y$ is a OWP, then $h : X \to \{0, 1\}$ is a hard-core bit for f if

$$\left\{x \xleftarrow{\$} X : (f(x), h(x))\right\} \overset{c}{\approx} \left\{x \xleftarrow{\$} X, b \xleftarrow{\$} \{0, 1\} : (f(x), b)\right\}.$$

Also, recall the Goldreich-Levin hard-core bit. Given a OWP $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$, we can define a new OWP $g : \mathbb{Z}_2^{2n} \to \mathbb{Z}_2^{2n}$ where $g(x, r) = (f(x), r)$. Then, we claim that the function

$$h((x, r)) = \langle x, r\rangle = \sum_{i=0}^{n-1} x_i r_i \in \mathbb{Z}_2$$

is a hard-core bit for g. We now prove the Goldreich-Levin theorem as stated in the previous lecture.

Theorem 4.1 (Goldreich-Levin [GL89])**.** *Let $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ be a OWP. Define the augmented function $g : \mathbb{Z}_2^{2n} \to \mathbb{Z}_2^{2n}$ where $g(x, r) = (f(x), r)$. Then,*

$$B(x, r) = \langle x, r\rangle = \sum_{i=1}^{n} x_i r_i \in \mathbb{Z}_2$$

is a hard-core for g.

Proof. We proceed via contradiction. Suppose that $B(x, r)$ is not hard-core for g. Then, there exists a distinguisher $\mathcal{D}$ such that

$$\Pr\left[x, r \xleftarrow{\$} \mathbb{Z}_2^n : \mathcal{D}(g(x, r)) = B(x, r)\right] \geq \frac{1}{2} + \varepsilon$$

for some non-negligible ε. Consider the set

$$S = \left\{x \in \mathbb{Z}_2^n \mid \Pr\left[r \xleftarrow{\$} \mathbb{Z}_2^n : \mathcal{D}(f(x), r) = B(x, r)\right] \geq \frac{1}{2} + \frac{\varepsilon}{2}\right\}.$$

Using an averaging argument, we note that $|S| > \frac{\varepsilon}{2}\left|\mathbb{Z}_2^n\right|$. Otherwise,

$$\Pr\left[x, r \xleftarrow{\$} \mathbb{Z}_2^n : \mathcal{D}(g(x, r)) = B(x, r)\right] \leq \frac{\varepsilon}{2} \cdot 1 + \left(1 - \frac{\varepsilon}{2}\right) \cdot \left(\frac{1}{2} + \frac{\varepsilon}{2}\right) < \frac{1}{2} + \varepsilon.$$

Now, we show that using the distinguisher $\mathcal{D}$, we can construct an algorithm $\mathcal{A}$ that inverts $f(x)$ with high probability for all $x \in S$. This is sufficient to derive a contradiction to the one-wayness of f since

$$\Pr\left[x \xleftarrow{\$} \mathbb{Z}_2^n : \mathcal{A}(f(x)) = x\right] \leq \Pr\left[\mathcal{A}\left(f(x)\right) = x \mid x \in S\right] \Pr\left[x \xleftarrow{\$} \mathbb{Z}_2^n : x \in S\right] = \frac{\varepsilon}{2} \cdot \Pr\left[\mathcal{A}\left(f(x)\right) = x \mid x \in S\right],$$

which is non-negligible.

To develop some intuition for how this might be done, consider first the case where the distinguishing advantage of $\mathcal{D}$ on inputs $x \in S$ is exactly 1. Then, for all $x \in S$,

$$\Pr\left[r \xleftarrow{\$} \mathbb{Z}_2^n : \mathcal{D}(f(x), r) = B(x, r)\right] = 1.$$

It is then trivial to invert a given $x \in S$. To obtain the i^{th} component x_i of x, simply invoke the distinguisher on input $(f(x), e_i)$ where $e_i \in \mathbb{Z}_2^n$ is the i^{th} canonical basis vector ($e_{ij} = 1$ if $i = j$ and 0 otherwise). Observe that

$$B(x, r) = B(x, e_i) = \langle x, e_i \rangle = x_i \in \mathbb{Z}_2.$$

Since algorithm $\mathcal{D}$ on input $(f(x), e_i)$ will produce $B(x, e_i) = x_i$ with probability 1, this enables us to recover the i^{th} component of x. Repeating this process for $i = 1, \ldots, n$ allows us to recover x from $f(x)$ with probability 1. To summarize, algorithm $\mathcal{A}$ works as follows in this scenario:

Algorithm $\mathcal{A}$ when Distinguishing Advantage is 1

```
(1) For $i = 1, \ldots, n$, output the value of $\mathcal{D}(f(x), e_i)$ as $x_i$.
```

Of course, it is not appropriate to assume that the distinguishing advantage of $\mathcal{D}$ is exactly 1. Suppose instead that the distinguishing advantage was $\frac{3}{4} + \varepsilon$. In this scenario, there is again a simple algorithm that inverts $f(x)$ for $x \in S$. It is worth noting that the algorithm described previously (where $\mathcal{D}$ had distinguishing advantage 1) will not work in this case, since $\mathcal{D}(x, e_i)$ does not necessarily produce the correct response. Instead, we use the fact that the inner product is linear in each operation. Specifically, we note that

$$\langle x, r \rangle + \langle x, r + e_i \rangle = \langle x, r + r + e_i \rangle = \langle x, e_i \rangle = x_i \in \mathbb{Z}_2,$$

where $r \xleftarrow{\$} \mathbb{Z}_2^n$. For $r \xleftarrow{\$} \mathbb{Z}_2^n$, consider the probability that

$$x_i = \mathcal{D}(f(x), r) + \mathcal{D}(f(x), r + e_i). \tag{4.1}$$

We have that $\mathcal{D}(f(x), r)$ computes $\langle x, r \rangle$ with probability $\frac{3}{4} + \varepsilon$, and so $\mathcal{D}\left(f(x), r\right)$ fails with probability $\frac{1}{4} - \varepsilon$, and similarly for $\mathcal{D}(f(x), r + e_i)$. By a union bound, the probability that at least one of $\mathcal{D}(f(x), r)$ and $\mathcal{D}(f(x), r + e_i)$ fails is at most $2\left(\frac{1}{4} - \varepsilon\right) = \frac{1}{2} - 2\varepsilon$. The probability that we obtain the correct bit x_i using (4.1) is at least $\frac{1}{2} + 2\varepsilon$. Using this fact, we can construct an algorithm to recover the i^{th} bit of x. Algorithm $\mathcal{A}$ firsts chooses $r \xleftarrow{\$} \mathbb{Z}_2^n$ and evaluates (4.1). With probability $\frac{1}{2} + 2\varepsilon$, this will yield the correct value x_i. Since ε is non-negligible, this probability is bounded away from $\frac{1}{2}$ by a polynomial factor. Thus, $\mathcal{A}$ repeats this step polynomially many times (using an independently chosen value $r \in \mathbb{Z}_2^n$ each time) and takes the majority response to achieve a constant success probability. Algorithm $\mathcal{A}$ repeats this process for each component of x, and in doing so, recovers x with constant success probability. To summarize, algorithm $\mathcal{A}$ works as follows in this scenario:

Algorithm $\mathcal{A}$ when Distinguishing Advantage at Least $\frac{3}{4} + \varepsilon$

```
(1) Choose $r_1, \ldots, r_m \xleftarrow{\$} \mathbb{Z}_2^n$ where $m = \text{poly}(n)$.
(2) For $i = 1, \ldots, n$,
    (a) For $j = 1, \ldots, m$, compute $y_j = \mathcal{D}(f(x), r_j) + \mathcal{D}(f(x), r_j + e_i) \in \mathbb{Z}_2$.
    (b) Output $x_i$ as the majority value $\{y_j\}_{j=1}^m$.
```

Finally, we consider the general case where the distinguishing advantage of $\mathcal{D}$ is $\frac{1}{2} + \varepsilon$. □

As an aside, we present a coding-theoretic view of the Goldreich-Levin Theorem. We first describe the Hadamard code. The Hadamard code maps elements of $\mathbb{Z}_2^k$ (alternatively, bitstrings of length k) to elements of $\mathbb{Z}_2^{2^k}$ (alternatively, bitstrings of length 2^k). Let $\mathcal{H} : \mathbb{Z}_2^k \to \mathbb{Z}_2^{2^k}$ denote the Hadamard encoding function defined as

$$\mathcal{H}(x) = (\langle x, y \rangle)_{y \in \mathbb{Z}_2^k}.$$

In words, the Hadamard encoding of an element of $\mathbb{Z}_2^k$ is constructed by taking the vector of inner products between x and the 2^k elements of $\mathbb{Z}_2^k$. As an example, the Hadamard encoding of $x \in \mathbb{Z}_2^3$ is the vector

$$(\langle x, (0,0,0) \rangle, \langle x, (0,0,1) \rangle, \ldots, \langle x, (1,1,1) \rangle) \in \mathbb{Z}_2^{2^3}.$$

By construction of the Hadamard code, the embedding of two distinct values in $\mathbb{Z}_2^k$ will always have Hamming distance 2^{k-1}. We now return to our discussion of the Goldreich-Levin algorithm. As described above, the Goldreich-Levin algorithm operates over the set

$$S = \left\{ x \in \mathbb{Z}_2^n \mid \Pr\left[r \xleftarrow{\$} \mathbb{Z}_2^n : \mathcal{D}(f(x), r) = \langle x, r \rangle \right] \geq \frac{1}{2} + \frac{\varepsilon}{2} \right\}.$$

Take an element $x \in S$. Using algorithm $\mathcal{D}$, we have access to a *noisy* version of the vector $(\langle x, r \rangle)_{r \in \mathbb{Z}_2^n} = \tilde{\mathcal{H}}(x)$. Thus, the Goldreich-Levin algorithm can be viewed as a local list-decoding algorithm for the Hadamard code, that is, an algorithm that recovers x when given local access to a noisy version of a Hadamard codeword (some value $\mathcal{H}(x)$).

We can also generalize (without proof) the Goldreich-Levin construction as follows.

Theorem 4.2 (Naslund [Näs96]). *Let q be an n-bit prime and let $f : \mathbb{Z}_q \to \{0,1\}^m$ be an injective* OWF*. Define $g(x, a, b) = (f(x), a, b)$ where $a, b \in \mathbb{Z}_q$. Then, for all $1 \leq i \leq n - \log_2 n$, the function*

$$B_i(x, a, b) = (ax + b \bmod q)|_i$$

is hard-core for g. Here, $ax + b \pmod{q}|_i$ denotes the i^{th} bit of $ax + b \pmod{q}$.

Note that in the above theorem, we restrict $i \leq n - \log_2 n$ because there is a bias in the most significant bits of $ax + b$ (since q is *not* a power of 2). As was the case in the proof of the Goldreich-Levin theorem, if there was an efficient algorithm that computed $(ax + b \bmod q)|_i$ with non-negligible advantage, then there exists an efficient algorithm that finds preimages of x with non-negligible advantage.

Corollary 4.3. *Let $g \in \mathbb{Z}_p^*$ be a generator of a subgroup $G \leq \mathbb{Z}_q^*$ of prime order q. Assuming discrete log is hard in G, then, for all $1 \leq i \leq n - \log_2 n$, $x|_i$ is a hard-core bit for $f(x) = g^x \pmod{p}$.*

Proof. Suppose there exists an algorithm $\mathcal{A}_i$ where

$$\Pr\left[x \leftarrow \{1, \ldots, q\} : \mathcal{A}\left(g^x \bmod p\right) = x|_i \right] \geq \frac{1}{2} + \varepsilon.$$

Using $\mathcal{A}_i$, we build an algorithm $\mathcal{B}$ that solves discrete log in G. The input to algorithm $\mathcal{B}$ is an element $y \in G$. Define the function $h_y(a, b) = (ax + b \bmod q)|_i$ where $g^x = y$. Observe that $g^{ax+b} = y^a g^b$. Then,

$$h_y(a, b) = (ax + b \bmod q)|_i = \mathrm{Dlog}_g\left(y^a g^b\right)\big|_i.$$

Now,

$$\Pr_{a, b \leftarrow \mathbb{Z}_q}\left[\mathcal{A}_i\left(y^a \cdot g^b\right) = h_y(a, b)|_i = (ax + b \bmod q)|_i \right] > \frac{1}{2} + \varepsilon.$$

Using $\mathcal{A}_i$, we can construct an algorithm that given g^x, computes $(ax + b \bmod q)|_i$ with non-negligible advantage. Using Nashlund's algorithm, we can recover $x = \mathrm{Dlog}_g(y)$. Thus, assuming the hardness of discrete log, $x|_i$ is hard-core for $f(x) = g^x \pmod{p}$. □

4.2. **Commitment Schemes.** We now begin a discussion on commitment schemes. At a high level, a commitment scheme is a two-party protocol between a sender S and a receiver R. In the first phase of the protocol, the sender S *commits* to some value m. The sender then sends its commitment to the receiver. At a later time, the sender can *open* the commitment by providing m and some auxiliary information to the receiver, which the receiver may use to verify that the value it received is indeed the value the sender committed to during the first phase. For a commitment scheme to be useful, if must thus satisfy the following two properties (which we will formally define later):

- **Hiding**: Given the commitment $\mathsf{Commit}(m)$ of a message m, the receiver should not learn anything about the value of m.

- **Binding**: Given a commitment $\mathsf{Commit}(m)$ of a message m, the sender should only be able to Open the commitment to message m.

Now, we give game-based definitions for what it means for a commitment scheme to be a hiding commitment. As usual in indistinguishability definitions, let b denote the world bit.

Commitment Hiding Game

(1) The adversary chooses two messages m_0 and m_1 and sends them to the challenger.
(2) The challenger responds with $\mathsf{Commit}(m_b)$.
(3) The adversary outputs $b' \in \{0, 1\}$.

We define the adversary's advantage $\mathsf{Adv}_{\mathcal{A}}^{H}$ in the commitment hiding game to be

$$\mathsf{Adv}_{\mathcal{A}}^{(H)} = \left|\Pr\left[b' = 1 \mid b = 0\right] - \Pr\left[b' = 1 \mid b = 1\right]\right|.$$

Definition 4.4. A commitment scheme is perfectly hiding (PH) if for all adversaries $\mathcal{A}$ (including computationally unbounded ones), $\mathsf{Adv}_{\mathcal{A}}^{H}$ is negligible. A commitment scheme is computationally binding (CH) if for all efficient adversaries $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{H}$ is negligible.

Analogously, we define a security game to specify what we mean when we say a commitment scheme is binding.

Commitment Binding Game

(1) The adversary sends a commitment c to the challenger.
(2) The adversary then opens c as m_0 and m_1 where $m_0 \neq m_1$.
(3) The challenger outputs $b = 1$ if the adversary is able to open the commitment as m_0 and m_1 with $m_0 \neq m_1$. Otherwise, the challenger outputs $b = 0$.

We define the adversary's advantage in the commitment binding game to be

$$\mathsf{Adv}_{\mathcal{A}}^{B} = \Pr\left[b = 1\right].$$

Definition 4.5. A commitment scheme is perfectly binding (PB) if for all adversaries $\mathcal{A}$ (even computationally unbounded ones), $\mathsf{Adv}_{\mathcal{A}}^{B}$ is negligible. A commitment scheme is computationally binding (CB) if for all efficient adversaries $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{B}$ is negligible.

Theorem 4.6. *It is impossible to have commitment schemes that are both perfectly hiding and perfectly binding.*

Proof. We note that the two conditions are mutually conflicting. Let $c = \mathsf{Commit}(x)$ be a commitment to the value x. To open the commitment, the sender sends the tuple (x, t) to the receiver, where t denotes the auxiliary information the receiver uses to verify the commitment. For the commitment scheme to be perfectly binding, it must be the case that

$$\forall x' \neq x, t : (x', t') \text{ is not a valid way of opening the commitment } c.$$

Otherwise, a computationally unbounded adversary can win the commitment binding game with advantage 1 by exhaustively searching over all values (x, t) to find two valid ways $(x, t), (x', t')$ with $x' \neq x$ to open the commitment. But this means that the commitment scheme cannot be perfectly hiding. To see this, observe that given a commitment c, there is only one value of x that can be used to open the commitment. As a result, a computationally unbounded adversary will be able to win the commitment hiding game with advantage 1 using an exhaustive search over values of t:

Adversary for Commitment Hiding Game

(1) The adversary sends two messages $m_0 \neq m_1$ to the challenger.
(2) The challenger replies with a commitment c on one of the messages m_b.
(3) For all values of t, the adversary tries to open the commitment with (m_0, t). If the adversary is able to open the commitment using (m_0, t) for some t, it outputs 0. Otherwise it outputs 1.

If the commitment scheme is perfectly binding, the adversary's advantage in the above game is 1. If $b = 0$, then, there is some value of t such that (m_0, t) opens the commitment, so the adversary will output 0 with probability 1. If $b = 1$, then there is no value of t such that (m_0, t) opens the commitment (otherwise, the scheme is not perfectly binding), so the adversary will output 1 with probability 1. Its advantage in the commitment hiding game is thus 1. □

While it is impossible to have commitment schemes that are both perfectly hiding and perfectly binding, it is possible to have all the other configurations. In particular, perfectly hiding and computationally binding commitments will be very useful in constructing zero-knowledge proofs.

4.2.1. *Computational Hiding / Perfect Binding Commitment Scheme from* OWPs. We begin with a construction of a CH/PB commitment scheme from OWPs. Let $f : X \to X$ be a OWP and let $B : X \to \{0, 1\}$ be a hard-core bit for f. We define a single-bit commitment scheme:

Commitment Scheme from OWPs

- Commit: To commit to a bit $b \in \{0, 1\}$, the sender chooses $x \overset{\$}{\leftarrow} X$ and outputs $c = (f(x), b \oplus B(x))$.
- Open: To open a commitment $c = (c_1, c_2)$, the sender sends $b' \in \{0, 1\}$ and $x' \in X$ to the receiver. The receiver accepts if and only if $c_1 = f(x')$ and $c_2 = b' \oplus B(x')$.

Theorem 4.7. *The above commitment scheme is perfectly binding and computationally hiding.*

Proof. To show that the commitment is perfectly binding, we argue that there is only one way each commitment can be opened. Suppose the adversary was able to open a commitment $c = (c_1, c_2)$ as $(x, b), (x', b')$ where $b \neq b'$. Then, the challenger will only accept if $c_1 = f(x) = f(x')$, and since f is a permutation, we have $x = x'$. It must further be the case that

$$b \oplus B(x) = c_2 = b' \oplus B(x').$$

Since $x = x'$, $B(x) = B(x')$, and so $b = b'$, which is a contradiction. Thus, each commitment can only be opened one way, so the scheme is perfectly binding. Computational hiding follows from the fact that $B(x)$ is hard-core for f. More precisely, for $b, b' \in \{0, 1\}$, we have that

$$\left\{x \overset{\$}{\leftarrow} X : (f(x), b \oplus B(x))\right\} \overset{c}{\approx} \left\{x \overset{\$}{\leftarrow} X, r \overset{\$}{\leftarrow} \{0,1\} : (f(x), b \oplus r)\right\} \overset{c}{\approx} \left\{x \overset{\$}{\leftarrow} X : (f(x), b' \oplus B(x))\right\}.$$

□

In practice, observe that we can replace f by a collision-resistant hash function (CRHF). This will yield a CH/CB commitment scheme. To see that the scheme is computationally binding, observe that an adversary that is able to open the commitment two different ways must be able to produce $x \neq x'$ such that $f(x) = f(x')$, where f is a collision-resistant hash function. We appeal to the definition of collision resistance to conclude that the commitment scheme is computationally binding.

We note also that this commitment scheme is malleable. Specifically, given a commitment on a bit b, and another bit $b' \in \{0, 1\}$, it is easy to construct a commitment for $b \oplus b'$. Thus, this commitment scheme is additively homomorphic. It is a much more difficult problem to develop fully homomorphic commitment schemes as well as non-malleable commitment schemes.

One might observe that the above commitment scheme looks similar to an encryption scheme. Specifically, if there was a trapdoor for f that allowed anyone who knew the trapdoor to compute x given $f(x)$, then they can "decrypt" $b \oplus B(x)$ to recover b. However, substituting an encryption scheme does *not* yield a secure commitment scheme. To see this, take E to be a semantically secure symmetric-key encryption scheme with key-space $\mathcal{K}$.

Candidate Commitment Scheme from Symmetric Key Encryption

- Commit: To commit to a message m, the sender chooses $k \overset{\$}{\leftarrow} \mathcal{K}$ and output the commitment $c \leftarrow E(k, m)$.
- Open: To open a commitment c, the sender sends m' along with a key k'. The receiver verifies that $c = E(k', m')$.

Since E is semantically secure, this scheme is computationally hiding as before. However, the scheme is not necessarily binding. For instance, if E in this case was the one-time pad, the encryption scheme is certainly not binding! In particular, given a commitment c, the adversary can open the commitment to any value m it chooses by simply choosing the key as $k = c \oplus m$. Even if we replace E by a PRP (for instance, AES), the resulting scheme is still not secure. The adversary is still able to open the commitment c to a message m as some m' by choosing a random key $k' \xleftarrow{\$} \mathcal{K}$ and setting $m' \leftarrow D(k', c)$, where $D = E^{-1}$. By construction, (m', k') is a valid opening of the commitment c and with non-negligible probability, $m' \neq m$. While it may no longer be the case that the adversary can open the commitment to an arbitrary message of its choosing, it is able to open the commitment as a different message, which suffices to show that the commitment scheme is not binding. Thus, one must be very careful when constructing candidate commitment schemes.

4.2.2. *Computational Hiding / Perfect Binding Commitment Schemes from* PRGs. The previous commitment scheme we considered required OWPs. Here, we present another CH/PB single-bit commitment scheme based on PRGs (as demonstrated previously, it suffices to have OWFs in order to construct PRGs).

Let G be a secure PRG with seed space $\{0,1\}^n$ and output space $\{0,1\}^{3n}$. Our commitment scheme operates as follows:

Commitment Scheme from PRGs

- Commit: Suppose the sender wants to commit to a bit b. Then, the sender and receiver engage in the following protocol:
  1. The receiver chooses $r \xleftarrow{\$} \{0,1\}^{3n}$ and sends r to the sender.
  2. The sender chooses a seed $s \xleftarrow{\$} \{0,1\}^n$ and computes
  $$c = \begin{cases} G(s) & b = 0 \\ G(s) \oplus r & b = 1. \end{cases} \tag{4.2}$$
  The sender then sends c to the receiver.
- Open: To open a commitment c, the sender sends the bit b' and the seed s' used to construct the commitment. To verify, the receiver checks (4.2).

Theorem 4.8. *The above commitment scheme is perfectly binding and computationally hiding.*

Proof. We first that this scheme is computationally hiding. First, we note that if $b = 0$, $c = G(s)$ and if $b = 1$, $c = G(s) \oplus r$. The view of the receiver is the tuple (c, r). Now, we have,

$$\left\{s \xleftarrow{\$} \{0,1\}^n : (G(s), r)\right\} \overset{c}{\approx} \left\{R \xleftarrow{\$} \{0,1\}^{3n} : (R, r)\right\} \overset{c}{\approx} \left\{s \xleftarrow{\$} \{0,1\}^n : (G(s) \oplus r, r)\right\},$$

where the two equivalences hold by the security of the PRG. Next, we show that the scheme is perfectly binding. In order to win the binding game, the adversary needs to find $s_0, s_1 \in \{0,1\}^n$ such that $G(s_0) = G(s_1) \oplus r$, or equivalently, $G(s_0) \oplus G(s_1) = r$. Now, observe that $r \xleftarrow{\$} \{0,1\}^{3n}$ and moreover, that there are at most 2^{2n} possible values for $G(s_0) \oplus G(s_1)$. Thus, the probability that the receiver chose a value r such that there exists $s_0, s_1 \in \{0,1\}^n$ where $G(s_0) \oplus G(s_1) = r$ is at most $\frac{2^{2n}}{2^{3n}} = \frac{1}{2^n}$. Thus, the advantage of any adversary in winning the commitment binding game is upper bounded:

$$\mathsf{Adv}_{\mathcal{A}}^{B} \leq \frac{2^{2n}}{2^{3n}} = \frac{1}{2^n},$$

which is negligible function in n. □

5. 4/17: Commitment Schemes

5.1. **Perfectly Hiding Commitment Schemes.** We continue our previous discussion of commitment schemes. In the previous lecture, we described commitment schemes that were perfectly binding and computationally hiding. In this lecture, we present commitment schemes that are perfectly hiding and computationally binding. There are several possible constructions from different primitives:

- From OWPs $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$.
- From collision-resistant hash functions (CRHFs): $f : \{0,1\}^{2n} \to \{0,1\}^n$.

- From discrete log (Pederson commitments [Ped91]). Let $\mathbb{G}$ be a group of prime order p with generators g, h (chosen randomly from $\mathbb{G}$ during setup).

Commitment Scheme from Discrete Log (Pederson)

- Commit: To commit to a message $m \in \mathbb{Z}_p^*$, choose $r \xleftarrow{\$} \mathbb{Z}_p^*$ and send $c \leftarrow g^m h^r \in \mathbb{G}$.
- Open: To open a commitment c, send the message m' and randomness r' used to construct the commitment. The receiver verifies that $c = g^{m'} h^{r'}$.

- From RSA (similar to Guillou-Quisquater zero-knowledge protocol [GQ88]). Let $N = pq$ be an RSA modulus and let e be a prime number that satisfies $\gcd(e, \varphi(n)) = 1$. Finally, take $g \in \mathbb{Z}_N^*$.

Commitment Scheme from RSA

- Commit: To commit to a message $m \in \{0, \dots, e-1\}$, choose $r \xleftarrow{\$} \mathbb{Z}_N^*$ and send $c = g^m r^e \pmod N$.
- Open: To open a commitment c, send the message m' and randomness r' used to construct the commitment. The receiver verifies that $c = g^{m'} (r')^e \pmod N$.

Theorem 5.1. *The Pederson commitment scheme (defined above) is perfectly hiding and computationally binding (under the assumption that the discrete log is hard).*

Proof. We show first that the scheme is perfectly hiding. Let $c = g^m h^r$ be a commitment for a message m. To show that the scheme is perfectly hiding, we show that for all messages m', there exist a unique r' such that $c = g^{m'} h^{r'}$. Since $h \in \mathbb{G}$ and g is a generator of $\mathbb{G}$, it follows that there exist some $\alpha \in \mathbb{Z}_p^*$ such that $h = g^\alpha$. The relation that we must satisfy is then given by

$$g^{m+\alpha r} = g^m h^r = c = g^{m'} h^{r'} = g^{m'+\alpha r'}.$$

Since $\mathbb{G}$ is a group of order p, we have $m + \alpha r = m' + \alpha r' \in \mathbb{Z}_p^*$, so setting $r' = \alpha^{-1}(m - m' + \alpha r) \in \mathbb{Z}_p^*$ satisfies the necessary relation. Thus, for every message m', there exist a unique r' such that $g^{m'} h^{r'} = c$. Thus, the h^r term in $c = g^m h^r$ perfectly hides all information about m, so we conclude that the scheme is perfectly hiding.

Next, we show that the scheme is computationally binding under the assumption that discrete log is hard in $\mathbb{G}$. Suppose there was an adversary $\mathcal{A}$ that is able to open a commitment c two different ways. Then, the adversary must be able to produce (with non-negligible advantage) messages $m \neq m'$ with associated randomness r, r' such that $g^m h^r = g^{m'} h^{r'}$. If we write $h = g^\alpha$, this means that $\mathcal{A}$ is able to produce values m, m', r, r', such that the following holds

$$m + \alpha r = m' + \alpha r' \pmod p, \tag{5.1}$$

from which we may solve for $\alpha = (m - m')(r' - r)^{-1} \in \mathbb{Z}_p^*$. Note that $r' - r \neq 0$ since otherwise, $m = m'$ which is a contradiction. Using $\mathcal{A}$, we can construct an adversary $\mathcal{B}$ that solves the discrete log problem in G. Algorithm $\mathcal{B}$ takes as input a discrete log challenge (g, g^α). Then $\mathcal{B}$ proceeds as follows:

Algorithm $\mathcal{B}$ to Solve Discrete Log in $\mathbb{G}$

(1) Invoke $\mathcal{A}$ on the Pederson commitment scheme over group $\mathbb{G}$ with generators $g, h = g^\alpha$. Algorithm $\mathcal{A}$ produces output $m, m', r, r' \in \mathbb{Z}_p^*$.
(2) Output $\alpha' = (m - m')(r' - r)^{-1}$.

First, observe that the inputs to $\mathcal{A}$ are distributed properly (in the same manner as an actual instantiation of Pederson's scheme). By assumption then, $\mathcal{A}$ will produce m, m', r, r' that satisfy relation (5.1) with non-negligible advantage ε. But if m, m', r, r' satisfy (5.1), $\alpha = \alpha'$, so $\mathcal{B}$ succeeds with advantage ε, as desired. □

Theorem 5.2. *The RSA-based commitment scheme (defined above) is perfectly hiding and computationally binding.*

Proof. First, we show that the scheme is perfectly hiding. Let $c = g^m r^e$ be a commitment to a message m. Then, for every $m' \in \{0, \dots, e-1\}$, there exists a unique $r' \in \mathbb{Z}_N^*$ such that $c = g^{m'} (r')^e$, namely the value of r' that satisfies $(r')^e = g^{m-m'} r^e$. Existence and uniqueness of r' follows from the fact that the function $f(x) = x^e \pmod N$ is a permutation over $\mathbb{Z}_N^*$ (since $\gcd(e, \varphi(n)) = 1$).

Next, we show that the scheme is computationally binding under the RSA assumption (the hardness of computing e^{th} roots in $\mathbb{Z}_N^*$). Suppose there was an algorithm $\mathcal{A}$ that can win the commitment binding game with non-negligible advantage. We construct an algorithm $\mathcal{B}$ that can compute e^{th} roots in $\mathbb{Z}_N^*$. Algorithm $\mathcal{B}$ takes N, e, g as input and will output $g^{1/e}$ with non-negligible advantage.

Algorithm $\mathcal{B}$ to Compute e^{th} Roots in $\mathbb{Z}_N^*$

(1) Invoke $\mathcal{A}$ on the commitment scheme instantiated with parameters N, e, g. Algorithm $\mathcal{A}$ outputs $m, m' \in \{0, \ldots, e-1\}$ and $r, r' \in \mathbb{Z}_N^*$.
(2) Check that $m \neq m'$ and $g^m r^e = g^{m'} (r')^e$. If this relation is not satisfied, then abort.
(3) Set $a = r' r^{-1}$ and $b = m - m'$. We claim that $\gcd(a, b) = 1$, so there exists $s, t \in \mathbb{Z}$ such that $as + et = 1$ over the integers. Output $a^s g^t$.

We show that if algorithm $\mathcal{A}$ has advantage ε in the commitment binding game, then algorithm $\mathcal{B}$ outputs $g^{1/e}$ with the same advantage ε. First, observe that the parameters on which we invoke algorithm $\mathcal{A}$ are properly distributed, so with probability ε, $\mathcal{A}$ will produce $m \neq m'$ and r, r' that satisfy the relation $g^m r^e = g^{m'} (r')^e$ in the first step of algorithm $\mathcal{B}$. We now show that if m, m', r, r' satisfy this relation, then the value of $a^s g^t$ computed in the third step of $\mathcal{B}$ is precisely the e^{th} root of g. By construction of a and b we have that

$$g^b = g^{m-m'} = \left(\frac{r'}{r}\right)^e = a^e.$$

Next, note that since $m \neq m'$ and $m, m' \in \{0, \ldots, e-1\}$, it must be the case that $0 < |m - m'| < e$. Thus, $0 < |b| < e$, and since e is prime, b and e must be relatively prime. Using the Euclidean algorithm, we can efficiently compute $s, t \in \mathbb{Z}$ such that $bs + et = 1$. Then,

$$\left(a^s g^t\right)^e = a^{se} g^{et} = g^{bs} g^{et} = g,$$

using the above relation $g^b = a^e$. Thus, $a^s e^t$ is the e^{th} root of g. Thus, if $\mathcal{A}$ outputs a valid m, m', r, r' with advantage ε, then $\mathcal{B}$ will output the e^{th} root of g with the same advantage ε. Thus, assuming the hardness of computing e^{th} roots in $\mathbb{Z}_N^*$, the commitment scheme is computationally binding. $\square$

Observe that the Pederson and RSA-based commitment schemes described above are additively homomorphic. In the case of Pederson commitments, given commitments $\mathsf{Commit}(m_1) = g^{m_1} h^{r_1}$ and $\mathsf{Commit}(m_2) = g^{m_2} h^{r_2}$. We can construct a commitment to $m_1 + m_2$ by computing

$$\mathsf{Commit}\left(m_1 + m_2\right) = \left(g^{m_1} h^{r_1}\right)\left(g^{m_2} h^{r_2}\right) h^{r'} = g^{m_1+m_2} h^{r''},$$

where $r' \xleftarrow{\$} \mathbb{Z}_p^*$ is freshly generated randomness, and where $r'' = r_1 + r_2 + r'$. Note that this is distributed identically to a fresh commitment to $m_1 + m_2$. To open the commitment, the sender provides $m_1 + m_2$ and r''. A similar homomorphic property holds for the RSA-based commitments.

5.2. Universal Hash Functions and the Leftover Hash Lemma.

In this section, we show how to construct perfectly-hiding, computationally-binding commitment schemes from CRHFs. Let $H : \{0,1\}^T \to \{0,1\}^t$ be a CRHF. First, consider the following candidate scheme:

Candidate Construction of Perfectly Hiding Commitments from CRHFs

- Commit: To commit to a message $m \in \{0,1\}^T$, choose $r \xleftarrow{\$} \{0,1\}^T$. Send the commitment $(H(r), m \oplus r)$.
- Open: To open a commitment $c = (c_1, c_2)$, send the message m' and the randomness r' used to construct the commitment. The receiver verifies that $c_1 = H(r')$ and $c_2 = m' \oplus r'$.

Unfortunately, this scheme is not necessarily perfectly hiding. For instance, $H(x)$ might leak the first bit of x. In this case, given $(H(r), m \oplus r)$, the adversary would be able to determine the first bit of r, and correspondingly, the first bit of m. More generally, the problem we have is that given $H(r)$, r is no longer necessarily uniformly random over $\{0,1\}^T$. To address this problem, we will need to define the notion of an extractor. Specifically, we need a way to extract entropy from r in order to produce a uniformly random string.

Definition 5.3. Let $\mathcal{H}$ be a family of $X \to Y$ hash functions. For $\varepsilon \in [0,1]$, $\mathcal{H}$ is an ε-universal hash family (ε-UHF) if for all $x \neq x' \in X$,

$$\Pr\left[h \xleftarrow{\$} \mathcal{H} : h(x) = h(x')\right] < \varepsilon.$$

Fact 5.4. *If $\mathcal{H}$ is a ε-UHF from $X \to Y$, then it must be the case $\varepsilon \geq \frac{1}{|Y|} - \frac{1}{|X|}$.*

In contrast to collision-resistant hash functions, it is possible to construct universal hash functions. The distinction in this case is the fact that collision resistance is defined with respect to a specific hash function, while universality is defined with respect to a family of hash functions. As a result, it is possible to construct universal hash functions information theoretically. We give a construction of a universal hash family below.

Take p to be a prime and let $X = \mathbb{F}_p = Y$. For $t \in \mathbb{F}_p$, define

$$h_t\left(m_0, \ldots, m_{n-1}\right) = \sum_{i=0}^{n-1} m_i t^i.$$

Then define $\mathcal{H} = \{h_t\}\mid_{t \in \mathbb{F}_p}$.

Claim 5.5. $\mathcal{H}$ is $\left(\frac{n-1}{p}\right)$-UHF.

Proof. We use the key observation that a degree $n-1$ polynomial over a finite field can have at most $n-1$ roots. Take $m \neq m'$ and suppose that $h_t(m) = h_t(m')$. Then, it must be the case that

$$h_t(m) - h_t(m') = \sum_{i=0}^{n-1} \left(m_i - m_i'\right) t^i = 0.$$

Since $m_i \neq m_i'$, $h_t(m) - h_t(m')$ is not identically 0. Since a degree $n-1$ polynomial has at most $n-1$ roots,

$$\Pr\left[h \xleftarrow{\$} \mathcal{H} : h(m) = h(m')\right] = \Pr\left[t \xleftarrow{\$} \mathbb{F}_p : \sum_{i=0}^{n-1} \left(m_i - m_i'\right) t^i = 0\right] \leq \frac{n-1}{p}.$$

□

5.2.1. *Measures of Randomness.* We digress briefly to describe some methods of measuring the randomness of a distribution. Suppose R is an random variable over a set Ω where $|\Omega| = n$. Let U denote the uniform distribution on Ω.

Definition 5.6. R is δ-uniform if

$$\delta = \Delta(R; U) = \frac{1}{2} \sum_{x \in \Omega} \left|\Pr(R = x) - \frac{1}{n}\right|.$$

Definition 5.7. The guessing probability $\gamma(R)$ is given by

$$\gamma(R) = \max_{x \in \Omega} \left\{\Pr\left(R = x\right)\right\}.$$

The min-entropy $H_\infty(R)$ is given by

$$H_\infty(R) = -\log \gamma(R) = -\log \max_{x \in \Omega} \Pr\left(R = x\right).$$

Definition 5.8. The collision probability $\kappa(R)$ is given by

$$\kappa(R) = \sum_{x \in \Omega} \Pr\left(R = x\right)^2.$$

The Rényi entropy $H_2(R)$ is given by

$$H_2(R) = -\log \kappa(R) = -\log \sum_{x \in \Omega} \Pr\left(R = x\right)^2.$$

Another way we can view the guessing and collision probabilities is to define $p_R \in [0,1]^{|\Omega|}$ to be the vector of probabilities for the values of R. Then, the guessing probability is the ℓ_∞-norm of p_R, that is, $\gamma(R) = \|p_R\|_\infty$ and the collision probability is the ℓ_2-norm of p_R, that is, $\kappa(R) = \|p_R\|_2^2$.

Fact 5.9. *For the uniform distribution U on Ω with $|\Omega| = n$, $\gamma(U) = \kappa(U) = \frac{1}{n}$.*

Proof. By definition, the counting probability $\gamma(U) = \frac{1}{n}$. For the collision probability, we have

$$\kappa(U) = \sum_{x\in\Omega} \Pr(x)^2 = \sum_{x\in\Omega} \left(\frac{1}{n}\right)^2 = \frac{1}{n}.$$

□

Fact 5.10. *If R is δ-uniform on Ω where $n = |\Omega|$, then*

$$\frac{1}{n} + \frac{4\delta^2}{n} \le \kappa(R) \le \gamma(R) \le \frac{1}{n} + \delta.$$

Proof. Since R is δ-uniform, we have

$$\sum_{x\in\Omega} \left|\Pr(R = x) - \frac{1}{n}\right| = 2\delta.$$

First, we show that $\gamma(R) \le \frac{1}{n} + \delta$. Suppose otherwise and let $x^\star = \max_x \{\Pr(R = x)\}$. Using the triangle inequality, we have

$$\sum_{x\in\Omega} \left|\Pr(R = x) - \frac{1}{n}\right| \ge \left|\Pr(R = x^\star) - \frac{1}{n}\right| + \left|\sum_{x\neq x^\star} \Pr(R = x) - \frac{1}{n}\right| > \delta + \left|\sum_{x\neq x^\star}\left(\Pr(R = x) - \frac{1}{n}\right)\right|.$$

Next, we have

$$\sum_{x\neq x^\star} \Pr(R = x) = 1 - \Pr(R = x^\star) < 1 - \frac{1}{n} - \delta$$

Thus,

$$\sum_{x\in\Omega} \left|\Pr(R = x) - \frac{1}{n}\right| > \delta + \left|\sum_{x\neq x^\star}\left(\Pr(R = x) - \frac{1}{n}\right)\right| > \delta + \left|1 - \frac{1}{n} - \delta - \left(1 - \frac{1}{n}\right)\right| = 2\delta,$$

which is a contradiction. Next, to show that $\kappa(R) \le \gamma(R)$, simply observe that

$$\kappa(R) = \sum_{x\in\Omega} \Pr\left(R = x\right)^2 \le \sum_{x\in\Omega} \left[\left(\max_{x\in\Omega} \Pr(R = x)\right) \cdot \Pr\left(R = x\right)\right] = \max_{x\in\Omega} \Pr(R = x) = \gamma(R).$$

The final inequality follows by Jensen's inequality. Let $\delta_x = \Pr(R = x) - \frac{1}{n}$. Then,

$$\kappa(R) = \sum_{x\in\Omega} \Pr(R = x)^2 = \sum_{x\in\Omega} \left(\frac{1}{n} + \delta_x\right)^2 = \sum_{x\in\Omega} \left(\frac{1}{n^2} + \frac{2}{n}\delta_x + \delta_x^2\right) = \frac{1}{n} + \sum_{x\in\Omega} |\delta_x|^2,$$

where we have used the fact that $\sum_x \delta_x = \sum_x \Pr\left(R = x\right) - 1 = 0$. Using Jensen's inequality, we have that

$$\frac{1}{n}\sum_{x\in\Omega} |\delta_x|^2 \ge \left(\frac{1}{n}\sum_{x\in\Omega} |\delta_x|\right)^2 = \frac{4\delta^2}{n^2},$$

since $\sum_x |\delta_x| = 2\delta$ is precisely the expression for twice the statistical distance. Thus, we conclude that

$$\kappa(R) = \frac{1}{n} + \sum_{x\in\Omega} |\delta_x|^2 \ge \frac{1}{n} + \frac{4\delta^2}{n},$$

which proves the claim:

$$\frac{1}{n} + \frac{4\delta^2}{n} \le \kappa(R) \le \gamma(R) \le \frac{1}{n} + \delta.$$

□

Lemma 5.11 (Leftover Hash Lemma [ILL89]). *Let $\mathcal{H} = \{h : X \to Y\}$ be an ε-universal hash family. Let R be a random variable over X. Choose $h \xleftarrow{\$} \mathcal{H}$ independently of R. Then, $(h, h(R))$ is δ-uniform over $\mathcal{H} \times Y$ where*

$$\delta \le \frac{1}{2}\sqrt{|Y| \cdot (\kappa(R) + \varepsilon) - 1}.$$

Example 5.12. Suppose R is uniform on $S \subseteq \Omega$ and $|S| = 2^{300}$. Then, $\kappa(R) = \frac{1}{|S|} = 2^{-300}$. Let $\mathcal{H}$ be a 2^{-100}-UHF from $\Omega \to Y$, with $|Y| = 2^{100}$. Then, $(h, h(R))$ is δ-uniform for

$$\delta \leq \frac{1}{2}\sqrt{2^{100}\left(2^{-300} + 2^{-100}\right) - 1} = 2^{-101}.$$

With this preparation, we are ready to describe the perfectly hiding commitment scheme based on collision resistant hash functions. Let $H : \{0,1\}^T \to \{0,1\}^t$ be a CRHF, and let $\mathcal{H}$ be a universal hash family from $\{0,1\}^T \to \{0,1\}^T$. Our commitment scheme is then defined as follows:

Perfectly Hiding Commitment Scheme based on CRHFs

- Commit : To commit to a message $m \in \{0,1\}^T$, choose $r \xleftarrow{\$} \{0,1\}^T$ and $h \xleftarrow{\$} \mathcal{H}$. The sender then sends $(h, H(r), h(r) \oplus m)$.
- Open: To open a commitment $c = (c_1, c_2, c_3)$, the sender sends the message m', randomness r', and hash function h' used to construct the commitment. The receiver then verifies that $c_2 = H(r')$ and that $c_3 = h'(r') \oplus m'$.

Choose $r \xleftarrow{\$} \{0,1\}^T$ and $h \xleftarrow{\$} \mathcal{H}$. To commit, send $h, c_1 \leftarrow H(r), c_2 \leftarrow h(r) \oplus b$. To open, send m and b. By the Leftover Hash Lemma, this scheme is perfectly hiding. The scheme is computational binding since H is a CRHF.

6. 4/22: Oblivious Transfer

6.1. **Oblivious Transfer.** In this lecture, we begin our discussion of oblivious transfer. Suppose we have two parties, a sender $\mathcal{S}$ and a receiver $\mathcal{R}$. In *1-out-of-n* OT, the sender holds a database $\{m_1, \ldots, m_n\}$ of n iterms and the receiver holds an index $i \in \{1, \ldots, n\}$. In what follows, we will often write $\mathcal{S} \leftarrow \{m_1, \ldots, m_n\}$ and $\mathcal{R} \leftarrow i$ to indicate that $\mathcal{S}$ holds $\{m_1, \ldots, m_n\}$ and $\mathcal{R}$ holds i. An oblivious transfer (OT) protocol allows the receiver to learn the message m_i without the sender learning anything. We will formalize the precise security notion shortly. As an aside, we note that oblivious transfer may be viewed as strong private information retrieval (PIR). The distinction is that in standard PIR, the receiver might learn additional information about the database beyond just m_i. Note that the sender still learns nothing in the PIR setting.

In the standard notion of OT, a fresh instantiation of the protocol is necessary each time the receiver requests a message m_i. We relax this constraint in a generalization of OT, called adaptive OT. In adaptive OT, there is a setup phase where the sender first sends to the receiver some function of the messages $m_1, \ldots, m_n$ (for instance, these might be encryptions of the messages). Once this setup phase is complete, the receiver can efficiently query for messages for indices $i_1, \ldots, i_k$. In general, querying for k indices in an adaptive OT scheme will be more efficient than performing k separate instantiations of an *1-out-of-n* OT protocol. Currently, using the best known constructions of adaptive OT, it is possible to process each query in $O(\lambda \log n)$ time, where λ is the security parameter.

We now precisely define the security properties for OT protocols. In this presentation, we will consider the *half-simulation* security model. In this model, the security definition for the receiver is based on an indistinguishability game while security for the sender is based on a simulation definition. We contrast this with the *full-simulation* security model where both sender and receiver security are based upon a simulation-based definition. Specifically, we require the following. We present the high-level definitions for standard *1-out-of-n* OT protocols.

- **Recipient Privacy**: At the end of the OT protocol, the sender does not learn any information about the recipient's index i.
- **Sender Privacy**: At the end of the OT protocol, the recipient learns only m_i and nothing else about the database (beyond what is explicitly leaked by m_i).

As mentioned previously, we adopt an indistinguishability-based definition for recipient privacy. Intuitively, our definition will capture the notion that the sender's view (this includes the sender's database as well as any messages sent between the two parties) is independent of the receiver's choice of index.

Definition 6.1. Let $\mathsf{View}_{\mathcal{S}}\left(i, (m_1, \ldots, m_n)\right)$ be a random variable for the sender's view of the protocol where $\mathcal{S} \leftarrow (m_1, \ldots, m_n)$ and $\mathcal{R} \leftarrow i$. Then, recipient privacy holds if for all indices $i \neq j \in \{1, \ldots, n\}$ and for all messages $m_1, \ldots, m_n \in \mathcal{M}$ in the message space $\mathcal{M}$, it holds that

$$\mathsf{View}_{\mathcal{S}}\left(i, (m_1, \ldots, m_n)\right) \overset{c}{\approx} \mathsf{View}_{\mathcal{S}}\left(j, (m_1, \ldots, m_n)\right).$$

Next, we define the notion of security for the sender. Here, our definition will capture the idea that at the end of the protocol, the recipient will learn m_i and nothing more. To formally specify this notion, we introduce a simulation-based security definition. First, we define a trusted party T, that on input i, returns the value m_i. In the simulation-based definition then, we require that the output of any efficient adversary interacting with the sender in the *real* protocol can be efficiently simulated by an efficient algorithm that only interacts with the trusted party (the *ideal* instantiation of the protocol where we have a trusted party). If this definition holds, we conclude that any information any efficiently adversary learns by its interactions with the sender S must be computable from m_i alone, in which case, we may conclude that the adversary does not learn anything else about the contents of the sender's database beyond what is explicitly leaked by m_i.

Definition 6.2. For all messages $m_1, \ldots, m_n \in \mathcal{M}$ and for all efficient adversaries $\mathcal{A}$ interacting with the sender, there exists an efficient simulator $\mathcal{S}$ interacting only with a trusted party $\mathcal{T}$ such that the output of $\mathcal{A}$ is computationally indistinguishable from the output of Sim.

Fact 6.3. *Oblivious transfer (1-out-of-2) implies key exchange secure against eavesdropping.*

By virtue of the above fact, we cannot have key exchange mechanisms based on AES or PRPs. Our constructions will thus rely on stronger assumptions from algebra.

6.2. **Blind Signatures.** For our first construction of OT, we will need a blind signature scheme. A blind signature scheme is specified by a protocol between two parties, the sender (or signer) $\mathcal{S}$ and the recipient $\mathcal{R}$. In a blind signature scheme, the recipient is able to request signatures σ_m on arbitrary messages $m \in \mathcal{M}$ in the message space $\mathcal{M}$, without revealing to the sender which signature he is requesting. We specify the scheme using the following algorithms:

Blind Signature Schemes

- $\mathsf{KeyGen}(\lambda)$: key generation algorithm that takes in a security parameter λ and outputs a public/private key-pair $(\mathsf{pk}, \mathsf{sk})$. The secret key sk is given to the sender.
- $\mathsf{BlindSign}$: an algorithm (protocol) between the sender and the recipient. The recipient provides as input a message m and the sender holds the secret signing key. At the end of the protocol, the recipient learns a signature σ_m on m and the sender should learn nothing.
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: a verification algorithm that outputs 1 if σ is a valid signature for m and 0 otherwise.

Next, we define our notions of security for a blind signature scheme. The first property we need is an adaptation of the unforgeability requirement for normal signature schemes. Recall that a signature scheme is existentially unforgeable under a chosen message attack if all efficient adversaries with access to a signing oracle is unable (except perhaps with negligible probability) to produce a signature on a message it did not explicitly query. Unfortunately, this is not the proper notion of unforgeability for a blind signature scheme, simply because the challenger (or sender) does not learn which messages the adversary is requesting signatures on. As a result, the challenger cannot determine whether the adversary was able to forge a signature on a new message at the conclusion of the game. Instead, we modify our definition to capture an appropriate notion of unforgeability for a blind signature scheme, namely that an adversary that has seen q valid signatures on messages of its choosing cannot produce a new signature on any message.

Definition 6.4. A blind signature scheme is unforgeable if for all efficient adversaries $\mathcal{A}$ that makes q invocations of $\mathsf{BlindSign}$, the probability that $\mathcal{A}$ outputs $q+1$ valid message-signature pairs (m, σ_m) is negligible.

The second notion we will require is the notion of privacy for the recipient. Just as in OT, we required that the sender learned no information about the receiver's index, in a blind signature scheme, we require that the sender learns nothing about the recipient's request. Thus, we introduce an analogous game-based definition that captures recipient privacy:

Recipient Privacy Game for Blind Signature Schemes

(1) The adversary chooses a public/private key pair $(\mathsf{pk}, \mathsf{sk})$ for the signature scheme and sends pk to the recipient, along with two messages $m_0, m_1 \in \mathcal{M}$.
(2) The challenger requests the signature (by invoking $\mathsf{BlindSign}$) on m_b for $b \in \{0, 1\}$.

(3) The adversary outputs b'. The adversary wins if $b = b'$.

Definition 6.5. A blind signature scheme is recipient private if for all efficient adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the recipient privacy game outlined above is negligible.

Before describing our first construction of a blind signature scheme, we also define the notion of a unique signature scheme.

Definition 6.6. A unique signature scheme is a signature scheme where for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\lambda)$ and for all messages $m \in \mathcal{M}$ in the message space $\mathcal{M}$, there exists a *unique* signature σ such that $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$.

6.2.1. *Blind Signatures from* RSA. We describe our first construction of a blind signature scheme based on the RSA problem [Cha82]. The protocol is specified by the following algorithms:

Blind Signature Scheme based on RSA

- $\mathsf{KeyGen}(\lambda)$: Choose primes p, q and set $N \leftarrow pq$. Choose e such that $\gcd(e, \varphi(N)) = 1$. Let $d \leftarrow e^{-1} \pmod{\varphi(n)}$. Let $H : \mathcal{M} \to \mathbb{Z}_n$ be a hash function from the message space $\mathcal{M}$ to $\mathbb{Z}_N$. The public key is $\mathsf{pk} \leftarrow (N, e)$ and the secret key is $\mathsf{sk} \leftarrow d$.
- $\mathsf{BlindSign}(\mathsf{sk}, m)$: To sign a message m, the sender and recipient engage in the following protocol:
  (1) The recipient chooses $r \xleftarrow{\$} \mathbb{Z}_N^*$ and sends $y \leftarrow H(m) \cdot r^e \in \mathbb{Z}_N$.
  (2) The sender replies with $z \leftarrow y^{\mathsf{sk}} = y^d \in \mathbb{Z}_n$.
  (3) The recipient computes the signature $\sigma \leftarrow zr^{-1} \in \mathbb{Z}_n$.
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: To verify the signature, use the public key $\mathsf{pk} = (N, e)$ and verify that $\sigma^e = H(m)$.

First, we verify that signature scheme is correct. Let σ be the result of invoking $\mathsf{BlindSign}(\mathsf{sk}, m)$. Then, we have that

$$\sigma = zr^{-1} = y^d r^{-1} = H(m)^d r^{de} r^{-1} = H(m)^d,$$

since $de = 1 \pmod{\varphi(n)}$. Then, in the verification algorithm, we have $\sigma^e = H(m)^{de} = H(m)$, as required. Next, we consider the security of this scheme. Finally, we observe that the RSA signature scheme described here is an example of a unique signature scheme. Specifically, there is only one signature, namely $\sigma \leftarrow H(m)^d$, for each message m.

We can show that this scheme is sender-secure in the random oracle model. However, we are unable to show that the scheme is recipient-secure. Observe in the recipient-security game, the adversary chooses the public key pk, and specifically, if the adversary chooses e such that $e \mid \varphi(n)$. It is an open problem whether there are efficient algorithms for this. Need to include a ZKP in the first message that e does not divide $\varphi(N)$.

6.2.2. *Blind Signatures from Pairings.* In this section, we present the Boneh-Lynn-Shacham (BLS) signature scheme based upon pairings [BLS01]. The advantage of this scheme over the RSA-based blind signature scheme is that we do not include a zero-knowledge proof of a statement like $\gcd(e, \varphi(N)) = 1$. Before presenting the BLS signature scheme, we first begin with a discussion of bilinear maps (pairings).

Definition 6.7. Let $\mathbb{G}$ be a group of prime order p with generator g. Then, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a symmetric bilinear map (pairing) if the following conditions hold

- e is efficiently computable.
- **Non-degeneracy**: $e(g, g) \neq 1$ where 1 is the identity element in $\mathbb{G}$.
- **Bilinearity**: for all $a, b \in \mathbb{Z}_p$,$e(g^a, g^b) = e(g, g)^{ab}$.

All known constructions of bilinear maps operate over elliptic curve groups; in these case, the target group $\mathbb{G}_T$ is actually a finite field. We will defer a discussion of actual construction of such bilinear maps to a subsequent part of the course. For most applications, it suffices to work with our abstract definition of bilinear maps. We show now how to use these bilinear maps to construct a signature scheme over a group $\mathbb{G}$ with a symmetric bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

BLS Signature Scheme

- $\mathsf{KeyGen}(\lambda)$: Choose a random exponent $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and let $H : \mathcal{M} \to \mathbb{G}$ be a hash function from the message space $\mathcal{M}$ to $\mathbb{G}$. Then, output the public key $\mathsf{pk} \leftarrow (g^\alpha, H)$ and the secret key $\mathsf{sk} \leftarrow \alpha$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: To sign a message m, compute $\sigma \leftarrow H(m)^{\mathsf{sk}} = H(m)^\alpha$.
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: To verify a signature σ on a message m, we verify that the following condition holds:
$$e(g^\alpha, H(m)) = e(g, \sigma).$$

Correctness of the BLS signature scheme follows from the bilinearity property:

$$e(g,\sigma) = e(g, H(m)^\alpha) = e(g, H(m))^\alpha = e(g^\alpha, H(m)).$$

Like the RSA signature scheme described earlier, the BLS signature scheme is also a unique signature scheme, that is, the only valid signature on message m is $H(m)^\alpha$.

Theorem 6.8. *The BLS signature scheme (described above) is existentially unforgeable in the random oracle model (where we model H as a random oracle) and the computational Diffie-Hellman assumption holds in $\mathbb{G}$.*

Now, we note that the BLS signature scheme can easily be modified to give a blind encryption scheme. Specifically, we replace the signing algorithm $\mathsf{Sign}(\mathsf{sk}, m)$ with a corresponding blind signing algorithm $\mathsf{BlindSign}(\mathsf{sk}, m)$. The blind signature algorithm is now a interactive protocol between the sender and the receiver. To obtain a signature on message m, the receiver engages in the following protocol with the sender:

$\mathsf{BlindSign}(\mathsf{sk}, m)$ for BLS Signatures

(1) The receiver chooses $r \xleftarrow{\$} \mathbb{Z}_p$. Then, the receiver computes and sends $y \leftarrow H(m) \cdot g^r \in \mathbb{G}$.
(2) The sender replies with $z \leftarrow y^\alpha = H(m)^\alpha \cdot (g^\alpha)^r$.
(3) The receiver computes $\sigma \leftarrow \frac{z}{(g^\alpha)^r}$.

It is easy to see that the scheme is correct. In particular, observe that the resulting signature the receiver obtains satisfies

$$\sigma = \frac{z}{(g^\alpha)^r} = \frac{H(m)^\alpha \cdot (g^\alpha)^r}{(g^\alpha)^r} = H(m)^\alpha,$$

which we have shown earlier to be a valid signature for α. The benefit of BLS is that we do not need to verify the public key (no need for ZKP).

6.3. **1-out-of-n Adaptive Oblivious Transfer Construction.** We show now how to build adaptive 1-out-of-n OT from unique, blind signature schemes [CNS07]. Consider a unique, blind-signature scheme with message space $\mathcal{M} = \{1, \ldots, n\}$ and signature space $\mathcal{S}$. Let $H : \mathcal{M} \times \mathcal{S} \to \{0,1\}^\ell$ be a hash function (that we model as a random oracle). Recall previously that in an adaptive OT scheme, there is an initial setup phase followed by a query phase, where the receiver is able to request arbitrarily many messages of his choice. Suppose the sender has messages $m_1, \ldots, m_n$ and the receiver has an index i. The setup phase of the protocol then proceeds as follows:

Setup Phase for Adaptive OT Protocol

(1) The sender generates a public-private key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\lambda)$ for the blind signature scheme.
(2) For $i = 1, \ldots, n$, the sender computes $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, i)$ and $c_i \leftarrow H(i, \sigma_i) \oplus m_i$. The sender sends $\mathsf{pk}, c_1, \ldots, c_n$ to the recipient.

Query Phase for Adaptive OT Protocol

(1) The receiver requests a blind signature on index i to obtain σ_i.
(2) The receiver computes $m_i \leftarrow c_i \oplus H(i, \sigma_i)$.

Theorem 6.9. *The OT scheme described above based on blind signatures is half-simulation secure. Note that the scheme is adaptively secure, but we do not prove that here.*

Proof. To show that the scheme is half-simulation secure, we show that the scheme is both sender-secure and recipient-secure. Recall from our definitions that we use an indistinguishability argument for recipient security and a simulation-based argument for sender security.

Claim 6.10. Under the assumption that the underlying blind signature scheme provides recipient-privacy, the adaptive OT scheme is recipient-secure. In particular, the sender cannot distinguish between a request to an index i versus a query to an index j.

Proof. This follows directly from the recipient-privacy property of the blind signature scheme. More specifically, to query for an index i, the only interaction between the recipient and sender is the recipient's request for a signature on index i. Recipient privacy of the underlying blind signature scheme means that the sender cannot distinguish between a signature request for index i from one for index j. □

Claim 6.11. Under the assumption that the underlying blind signature scheme is unforgeable, the adaptive OT scheme is sender-secure. Specifically, for all efficient adversaries $\mathcal{A}$ interacting with the sender, there exists an efficient simulator $\mathcal{S}$ interacting only with a trusted party such that the output of $\mathcal{A}$ is computationally indistinguishable from the output of $\mathcal{S}$.

Proof. Let $\mathcal{A}$ be an adversary for the sender-security game. We use $\mathcal{A}$ to construct a simulator $\mathcal{S}$ that only interacts with the trusted party $\mathcal{T}$. The simulator's operation is described as follows:

Simulator $\mathcal{S}$ in the Ideal Environment

(1) Generate a public/private key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\lambda)$ for the blind signature scheme. Choose random strings $c_1, \ldots, c_\ell \xleftarrow{\$} \{0,1\}^\ell$. Give the public key pk and the random strings $c_1, \ldots, c_\ell$ to $\mathcal{A}$.

(2) There are two interactions that we simulate for $\mathcal{A}$:
- **Random Oracle Query**: When $\mathcal{A}$ makes a random oracle query on (j, σ), the simulator checks whether σ is a valid signature on j (by invoking $\mathsf{Verify}(\mathsf{sk}, j, \sigma)$). If so, then $\mathcal{S}$ requests m_j from the trusted party $\mathcal{T}$ (the ideal functionality). Then, $\mathcal{S}$ programs the random oracle to return $m_j \oplus c_j$. Otherwise, $\mathcal{S}$ programs the random oracle to return a random string $r \xleftarrow{\$} \{0,1\}^\ell$.
- **Blind Signature Request**: At some point in the protocol, $\mathcal{A}$ can engage in the transfer protocol, that is, $\mathcal{A}$ requests a blind signature from the sender. The simulator $\mathcal{S}$ plays the role as the honest sender in the $\mathsf{BlindSign}$ protocol. Since $\mathcal{S}$ knows sk, it can simulate this perfectly.

(3) At some point, $\mathcal{A}$ will output some string s. At this point, $\mathcal{S}$ halts and outputs s.

We argue that $\mathcal{S}$ perfectly simulates the behavior of the real sender to $\mathcal{A}$, from which we may conclude that any information that $\mathcal{A}$ can learn by interacting with the sender in the real protocol can be simulated by a simulator that only has access to a trusted party. To see this, note first that the public key pk is generated exactly as in the real scheme. Similarly, the $c_1, \ldots, c_\ell$ that $\mathcal{A}$ receives is distributed identically in both scenarios: in the real environment, they are one-time pad encryptions, and in the simulated environment, they are truly random strings. Next, in the blind signature query, $\mathcal{S}$ plays the role as the sender precisely as in the real setup. What remains to be examined are the random oracle queries. In the real environment, the outputs of the random oracle at points (i, σ_i) where σ_i denotes a signature on i should be given by $c_i \oplus m_i$. Everywhere else, the output must be random. By construction, at points not of the form (i, σ_i), the simulated values are properly distributed. Next, we argue that any efficient adversary $\mathcal{A}$ can only produce at most one value of the form (i, σ_i) where σ_i is the *unique* signature on i. Observe first that the adversary is only allowed to make a single blind signature request (issuing the blind signature request amounts to engaging in the transfer protocol). Thus, if the adversary is able to make two valid random oracle queries (i, σ_i) and (j, σ_j) with $i \neq j$, then adversary $\mathcal{A}$ can be used to win the signature unforgeability game for the underlying signature scheme. As long as the underlying blind signature scheme is secure, this can only happen with negligible probability, so we conclude that $\mathcal{A}$ can make at most one valid query (i, σ_i) where σ_i is a valid signature on i. But in this case, we return the correct value $H(i, \sigma_i) = m_i \oplus c_i$, precisely as in the real environment. We conclude that $\mathcal{S}$ perfectly simulates the real environment to adversary $\mathcal{A}$, and thus the output of $\mathcal{S}$ in the ideal environment is statistically indistinguishable from the output of $\mathcal{A}$ in the real environment. □

Since the OT protocol satisfies both recipient-privacy (under an indistinguishability definition), as well as sender-privacy (under a simulation definition), we conclude that the scheme is half-simulation secure. □

7. 4/24: Oblivious Transfer and Private Information Retrieval

7.1. **Oblivious Transfer.** In the previous lecture, we described an oblivious transfer protocol based on blind signatures that was secure in the random oracle model. We consider another construction (for *1-out-of-2* oblivious transfer) due to Bellare and Micali [BM89], which is also secure under the CDH assumption in the random oracle model. Let $\mathbb{G}$ be a group of prime order p with generator g. Let H be a hash function (modeled as a random oracle) $\mathbb{G} \to \{0,1\}^{\ell}$. As usual, let m_0 and m_1 be the sender's messages and let $b \in \{0,1\}$ be the receiver's query.

OT Construction from CDH in the Random Oracle Model (Bellare-Micali)

(1) The sender chooses $c \xleftarrow{\$} \mathbb{G}$ and sends c to the receiver.
(2) The receiver chooses a random key $k \xleftarrow{\$} \mathbb{Z}_p$ and computes two ElGamal public keys $y_b \leftarrow g^k$ and $y_{1-b} \leftarrow \frac{c}{g^k}$, and sends y_0, y_1 to the sender.
(3) If $y_0 y_1 = c$, the sender chooses $r_0, r_1 \leftarrow \mathbb{Z}_p$ and computes ElGamal ciphertexts $c_0 \leftarrow (g^{r_0}, H(y_0^{r_0}) \oplus m_0)$ and $c_1 \leftarrow (g^{r_1}, H(y_1^{r_1}) \oplus m_1)$. The sender sends c_0, c_1 to the receiver, if $y_0 y_1 \neq c$, the sender aborts.
(4) The recipient decrypts $c_b = (v_0, v_1)$ using knowledge of k and outputs $m_b = H(v_0^k) \oplus v_1$.

By construction, observe that the receiver only knows the discrete log of either y_b or y_{1-b}, but not both. In this case, we have information theoretic security for the recipient, since the roles of y_0 and y_1 are interchangeable. Security for the sender follows from the CDH assumption in the random oracle model. We leave this as an exercise.

An alternative construction of OT that does not require random oracles is due to Naor and Pinkas [NP01]. They provide a construction of *1-out-of-2* OT that is secure under the DDH assumption without random oracles. In this protocol, security for the sender will be information theoretic while security for the recipient will be computational (based on the DDH assumption). Let $\mathbb{G}$ be a group of prime order p with generator g. As before, the sender holds messages m_0 and m_1 and the receiver holds a bit $b \in \{0,1\}$, denoting which message it wants to query. The OT protocol is as follows:

OT Construction from DDH (Naor-Pinkas)

(1) The receiver chooses $s, t \xleftarrow{\$} \mathbb{Z}_p$ and computes $x \leftarrow g^s$, $y \leftarrow g^t$, and $z_b \leftarrow g^{st}$. Finally, the receiver chooses $z_{1-b} \xleftarrow{\$} \mathbb{G}$. The receiver sends (x, y, z_0, z_1) to the sender.
(2) The sender verifies that $z_0 \neq z_1$. Then, the sender chooses $u_0, v_0, u_1, v_1 \xleftarrow{\$} \mathbb{Z}_p$ and computes $w_b \leftarrow x^{u_b} g^{v_b}$ and $c_b \leftarrow z_b^{u_b} y^{v_b} m_b$ for $b \in \{0,1\}$. The sender sends (w_0, c_0) and (w_1, c_1) to the receiver.
(3) The receiver takes (w_b, c_b) and outputs $(w_b^t)^{-1} c_b$.

First, we show that the scheme above is correct, that is, if $z_b = g^{st}$, then $w_b^{-t} c_b = m_b$. Expanding out the relations, we have,

$$w_b^{-t} c_b = (x^{u_b} g^{v_b})^{-t} (z_b^{u_b} y^{v_b} m_b) = g^{-(stu_b + v_b t)} g^{stu_b + v_b t} m_b = m_b.$$

Next, we argue security. In this scenario, sender security is information theoretic.

For improved performance, we can develop OT schemes where the bulk of the work is performed offline during a preprocessing phase. For instance, in the offline step, we precompute a series of OT correlations. In each correlated pair, the sender has a random (y_0, y_1) and the recipient have (k, y_k) where $k \xleftarrow{\$} \{0,1\}$. Note that k is unknown to the sender. Then, to perform OT, we use the following online protocol. Here, the sender has bits x_0 and x_1 while the receiver has a bit $b \in \{0,1\}$. In the online component of the protocol, the sender and receiver will make use of the precomputed correlated pair. Specifically, the sender has the pair (y_0, y_1) while the receiver has (k, y_k). The OT protocol proceeds as follows.

Online Portion of OT Protocol with Offline Preprocessing

(1) The recipient sends $c \leftarrow b \oplus k$ to the sender.
(2) The sender computes and sends $z_0 \leftarrow y_c \oplus x_0$ and $z_1 \leftarrow y_{1-c} \oplus x_1$.
(3) The recipient computes $x_b \leftarrow z_b \oplus y_k$.

To see that this works, consider the case where $b = 0$. Then,

$$z_0 \oplus y_k = (y_c \oplus x_0) \oplus y_k = y_k \oplus x_0 \oplus y_k = x_0,$$

and correspondingly in the case where $b = 1$:

$$z_1 \oplus y_k = y_{1-c} \oplus x_1 \oplus y_k = y_k \oplus x_1 \oplus y_k = x_1.$$

Recipient privacy follows since $b \oplus k$ is a one-time-pad encryption of b. Sender privacy follows from the fact that the recipient only knows one of y_0 and y_1, and the other is hidden again via a one-time-pad argument.

7.2. **Oblivious Polynomial Evaluation.** There are many generalizations of OT. One such generalization is oblivious polynomial evaluation. Consider a polynomial $f \in \mathbb{F}_p[x]$ over the finite field $\mathbb{F}_p$. In an oblivious polynomial evaluation protocol, the sender has a polynomial f and the recipient has an element $y \in \mathbb{F}_p$. At the end of the protocol, the recipient learns $f(y)$ and the sender does not learn anything (these security notions may be formalized in precisely the same manner as those for oblivious transfer). Existing oblivious polynomial evaluation protocols run in $O(\deg f)$ time. Note also that the existence of a protocol for oblivious polynomial evaluation immediately implies a protocol for *1-out-of-n* OT. Specifically, in a *1-out-of-n* protocol, the sender holds a database $a_1, \ldots, a_n \in \mathbb{F}_p$ and the recipient holds an index $i \in \{1, \ldots, n\}$. The protocol is given below:

1-out-of-n **Oblivious Transfer from Oblivious Polynomial Evaluation**

(1) The sender chooses a random polynomial $f \in \mathbb{F}_p[x]$ of degree n such that for all $i \in \{1, \ldots, n\}$, $f(i) = a_i$.
(2) The recipient engages in an oblivious polynomial evaluation protocol with the sender to learn $f(i) = a_i$.

Security of the above protocol follows directly from the security of the oblivious polynomial evaluation protocol. We note that we choose a random polynomial of degree n (as opposed to $n-1$) to ensure that if the recipient queries at a point $i \notin \{1, \ldots, n\}$, then the value is uniform over $\mathbb{F}_p[x]$.

7.2.1. *Private Equality Testing.* Another application of an oblivious polynomial evaluation protocol is in the construction of private equality testing protocols. Such private equality testing protocols can in turn be used in password authentication protocols. For instance, if a client desires to authenticate to a server via a password, the server can execute the private equality testing protocol to learn whether the client provided the correct password or not. At the end of the protocol, the server only learns whether the password was correct or not, and nothing else about the client's input.

We describe our equality testing protocol. In this setting, the sender has a value $a \in \mathbb{F}_p$ and the recipient has a value $b \in \mathbb{F}_p$. At the end of the protocol, the receiver should learn whether $a = b$, and nothing else about a. The sender should not learn anything. As usual in these protocols, the sender does not learn anything. We construct the protocol as follows:

Equality Testing Protocol from Oblivious Polynomial Evaluation

(1) The sender chooses $r \xleftarrow{\$} \mathbb{F}_p$ and sets $f(x) = r(x - a)$.
(2) The recipient engages in an oblivious polynomial evaluation protocol to learn $f(b)$. If $a = b$, then $f(b) = 0$. Otherwise, $f(b) = r(b - a)$, which is uniformly random to the recipient (since he does not know r and $r \xleftarrow{\$} \mathbb{F}_p$)

Security of the above equality testing protocol follows directly from the security of the oblivious polynomial evaluation scheme. We remark that in general, the same polynomial $f(x)$ from step 1 cannot be used for multiple instantiations of the protocol. Otherwise, learning multiple values of $f(x)$ will allow the receiver to learn additional information about a.

7.2.2. *Private Set Membership Testing.* We consider another application of oblivious polynomial evaluation: testing for set membership. In this scenario, the sender holds a set $S = \{a_1, \ldots, a_n\} \subseteq \mathbb{F}_p$ and the recipient holds an element $b \in \mathbb{F}_p$. The recipient would like to test whether $b \in S$ without revealing any information about b to the sender. Privacy for the sender means that the recipient only learns whether $b \in S$ and nothing else about S.

Private Set Membership Testing from Oblivious Polynomial Evaluation

(1) The sender chooses $r \xleftarrow{\$} \mathbb{F}_p$ and computes the polynomial $f(x) = r \prod_{i=1}^{n} (x - a_i)$.

(2) The receiver engages in an oblivious polynomial evaluation protocol with the sender to learn $f(b)$. If $f(b) = 0$, then $b \in S$. Otherwise, $b \notin S$.

Security of the protocol again follows from the security of the underlying oblivious polynomial evaluation protocol. As we noted above, the polynomial $f(x)$ constructed in step 1 cannot be reused for multiple queries. Generally, the security properties apply only for a single execution of the protocol.

7.2.3. *Oblivious* PRFs. We conclude by describe an alternative method for private set intersection, this time using oblivious PRFs. Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a PRF with key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$. Then, F is an oblivious PRF if there is an oblivious evaluation protocol for F. As usual, the oblivious evaluation protocol is a two-party protocol between a sender and a receiver. The sender holds a key $k \in \mathcal{K}$ and the recipient holds a value $x \in \mathcal{X}$. At the end of the oblivious evaluation protocol, the the receiver learns $F(k, x)$ and the sender learns nothing. One example of a PRF that admits an efficiently-computable oblivious evaluation function is the Naor-Reingold PRF [NR97].

The Naor-Reingold PRF is an algebraic PRF that derives hardness from the DDH assumption. The construction is as follows. Let $\mathbb{G}$ be a group of prime order p with generator g. Then, the Naor-Reingold PRF F_{NR} is a function from $\left(\mathbb{Z}_p^{n+1}\right) \times \{0,1\}^n \to \mathbb{G}$ where

$$F_{\mathsf{NR}}\left((x_0, \ldots, x_n), (b_1, \ldots, b_n)\right) = g^{x_0 \prod_{i : b_i = 1} x_i}.$$

Here, $\mathbb{Z}_p^{n+1}$ is taken to be the key space and $\{0,1\}^n$ is the input space of the PRF. It turns out that if DDH is hard in $\mathbb{G}$, then the Naor-Reingold PRF is secure.

Theorem 7.1. *If the* DDH *assumption holds in* $\mathbb{G}$*, then* F_{NR} *is a secure* PRF.

As noted previously, the Naor-Reingold PRF has a natural oblivious evaluation function. In this setting, the sender holds the PRF key $(x_0, \ldots, x_n) \in \mathbb{Z}_p^n$ and the receiver holds an input $(b_1, \ldots, b_n) \in \{0,1\}^n$. To learn $F(b_1, \ldots, b_n)$, the receiver engages in n *1-out-of-2* OTs to obtain either a_i (if $b_i = 0$) or $a_i x_i$ (if $b_i = 1$), where $a_i \xleftarrow{\$} \mathbb{Z}_p$. At the end of the protocol, the sender gives $g^{x_0 / \prod_i a_i}$ to the recipient, from which the recipient can compute $F_{\mathsf{NR}}(b_1, \ldots, b_n)\, g^{x_0 \prod_{x_i = 1} x_i}$. We describe the protocol in greater detail below. In the protocol below, the sender holds the secret key $(x_0, \ldots, x_n) \in \mathbb{Z}_p^n$ and the receiver has a point $(b_1, \ldots, b_n) \in \{0,1\}^n$.

Protocol for Oblivious Evaluation of Naor-Reingold PRF

(1) For each $i = 1, \ldots, n$:
   (a) The sender chooses $a_i \xleftarrow{\$} \mathbb{Z}_p$.
   (b) The sender and receiver engage in a *1-out-of-2* OT. The sender's inputs are $m_0 = a_i$ and $m_1 = a_i x_i$. The receiver's input is b_i. At the end of the OT protocol, the receiver learns $\hat{a}_i$ where $\hat{a}_i = a_i$ or $\hat{a}_i = a_i x_i$. The sender does not learn anything.
(2) The sender computes $\alpha \leftarrow x_0 \left[\prod_{i=1}^n a_i\right]^{-1}$ and sends $s \leftarrow g^{\alpha}$ to the receiver.
(3) The receiver computes $\hat{\alpha} = \prod_{i=1}^n \hat{a}_i$ and outputs $s^{\hat{a}}$.

We claim that $s^{\hat{a}}$ is the value of F_{NR} evaluated at $(b_1, \ldots, b_n)$. To see this, first observe that by construction,

$$\hat{\alpha} = \prod_{i=1}^n \hat{\alpha}_i = \prod_{i=1}^n a_i \prod_{i : b_i = 1} x_i.$$

Then,

$$\alpha\hat{\alpha} = x_0 \left[\prod_{i=1}^n a_i\right]^{-1} \prod_{i=1}^n a_i \prod_{i : b_i = 1} x_i = x_0 \prod_{i : b_i = 1} x_i.$$

Thus, we conclude that

$$s^{\hat{\alpha}} = g^{\alpha\hat{\alpha}} = g^{x_0 \prod_{i : b_i = 1} x_i} = F_{\mathsf{NR}}(b_1, \ldots, b_n).$$

Using an oblivious PRF, we have another natural protocol for private set membership testing. In this private set membership testing protocol, the sender holds a set $\{a_1, \ldots, a_n\} \subseteq \mathcal{X}$ and the recipient has a point $x \in \mathcal{X}$. To decide if $x \in \mathcal{X}$, the receiver engages in the following protocol with the server:

Private Set Membership from Oblivious PRFs

(1) The sender chooses a random key $k \xleftarrow{\$} \mathcal{K}$ for the oblivious PRF F. The sender computes and sends $S' = \{F(k, a_1), \ldots, F(k, a_n)\}$ to the recipient.
(2) To test whether $x \in \{a_1, \ldots, a_n\}$, the recipient engages in an oblivious evaluation protocol to learn $F(k, x)$. The recipient checks whether $F(k, x) \in S'$.

Security of the above construction follows directly from the underlying security of the oblivious PRF scheme. One advantage of this scheme in contrast to the scheme based on oblivious polynomial evaluation is that the recipient can now test for multiple values without having to go through the entire protocol each time. Each additional set membership query reduces to an oblivious PRF evaluation, which may be performed efficiently. Thus, after a preprocessing step (where the sender computes and sends $F(k, a_1), \ldots, F(k, a_n)$ to the receiver), each of the recipient's queries translates is a single oblivious PRF evaluation.

7.3. **Private Information Retrieval.** Private information retrieval can be viewed as a weak version of OT. As before, we have two parties, the sender $\mathcal{S}$, and the recipient $\mathcal{R}$. The sender holds a database $x_1, \ldots, x_n \in \{0,1\}^\ell$ and the recipient has an index $i \in \{1, \ldots, n\}$. At the end of the protocol, the receiver should learn x_i (and possibly other information) and the server should learn nothing. The security property that we are interested in is thus recipient privacy. Here, we will use the same indistinguishability definition as in our discussion of OT (Definition 6.1). Observe first that we can trivially achieve this property by simply having the sender send the entire database to the recipient. Our goal in PIR is then to reduce the communication complexity.

7.3.1. *Multi-Server* PIR *Constructions.* We begin by defining a 2-server PIR solution. In this case, we have two servers $\mathcal{S}_1, \mathcal{S}_2$, each with a copy of the database $x_1, \ldots, x_n \in \mathbb{F}_p$. In this construction, we assume that $\mathcal{S}_1$ and $\mathcal{S}_2$ does not collude. We begin with a simple solution that achieves $O(\sqrt{n})$ complexity. First, we represent the database in matrix form (without loss of generality, we can take n to be a perfect square):

$$A = \begin{bmatrix} x_{1,1} & \cdots & x_{1.\sqrt{n}} \\ \vdots & \ddots & \vdots \\ x_{\sqrt{n},1} & \cdots & x_{\sqrt{n},\sqrt{n}} \end{bmatrix}.$$

Suppose the receiver wants to query for the value $x_{i,j}$. The receiver chooses two random vectors $v_1, v_2 \in \mathbb{F}_p^{\sqrt{n}}$ such that $v_1 + v_2 = e_i$, where e_i is the i^{th} canonical basis vector. The recipient sends v_1 and v_2 to $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. Each server computes and replies with $\mu_i = Av_i$,. The recipient then computes

$$\mu_i + \mu_j = A(v_i + v_j) = Ae_i,$$

which gives the recipient the i^{th} column of A, from which the recipient learns $x_{i,j}$. The total communication complexity of this protocol is $O(\sqrt{n})$. We leave as an exercise how to achieve $O(\sqrt[3]{n})$ communication complexity in the two-server model. In the three-server setting, we can achieve communication complexity $O\left(2^{6\sqrt{\log n \log\log n}}\right)$. As a final note, we note that all of these constructions achieve information-theoretic security.

7.3.2. *Single-Server* PIR *Constructions.* If we use complexity assumptions, we can construct single-server PIR schemes. Currently, we have protocols that achieve $O(n^\varepsilon)$ complexity based on additive homomorphic encryption [KO97] and $O(\log n)$ complexity based on the ϕ-hiding assumption [CMS99]. We present the construction based on the ϕ-hiding assumption . Let $\ell_1, \ldots, \ell_n$ be the first n primes (note that $\ell_n = O(n \log n)$). The sender has a database of bits $x_1, \ldots, x_n \in \{0,1\}$ and the recipient has an index $i \in \{1, \ldots, n\}$. Then, we have the following:

Single Server PIR from the ϕ-Hiding Assumption

(1) The recipient begins by choosing $p, q \xleftarrow{\$} \mathsf{Primes}(\lambda)$ and where $p = 1 \pmod{\ell_i}$. The recipient sets $N \leftarrow pq$ and chooses $w \xleftarrow{\$} \mathbb{Z}_N$ such that w^{1/ℓ^i} does not exist in $\mathbb{Z}_N$, that is, w_i is not an ℓ_i-residue. Equivalently, $w^{\frac{p-1}{\ell_i}} \neq 1 \pmod{p}$. The recipient sends (N, w) to the server. Note that we write $\mathsf{Primes}(\lambda)$ to denote a distribution over prime numbers parametrized by a security parameter λ.

(2) The server computes and replies with

$$a \leftarrow w^{\prod_{j=1}^{n} \ell_j^{x_j}} \in \mathbb{Z}_N.$$

(3) The recipient computes

$$a^{\frac{p-1}{\ell_i}} \pmod p.$$

If the result is equal to 1, then $x_i = 1$, otherwise, $x_i = 0$.

First, we demonstrate correctness. Consider the case $x_i = 1$. Then, we have

$$a^{\frac{p-1}{\ell_i}} = \left[\left[w^{\prod_{j\neq i} \ell_j^{x_j}}\right]^{\ell_i}\right]^{\frac{p-1}{\ell_i}} = \left[w^{\prod_{j\neq i} \ell_j^{x_j}}\right]^{p-1} = 1 \pmod p.$$

Similarly, we van verify the corresponding condition when $x_i = 0$. Next, consider the communication complexity of the scheme. By construction, $N > \ell_n$, and since $\ell_n \sim n \log n$ so the number of bits needed to represent an element of $\mathbb{Z}_N$ is of order $O(\log n)$.

To prove security, we define a new assumption: the ϕ-hiding assumption, which states that the following two distributions are computationally indistinguishable:

- **Distribution 1:** Take $p, q \xleftarrow{\$} \mathsf{Primes}(\lambda)$ where $p = 1 \pmod \ell$, and set $N \leftarrow pq$. Choose $w \xleftarrow{\$} \mathbb{Z}_n$ where w is not an ℓ^{th} residue. Output (N, w).
- **Distribution 2:** Take $p, q \xleftarrow{\$} \mathsf{Primes}(\lambda)$, and set $N \leftarrow pq$. Choose $w \xleftarrow{\$} \mathbb{Z}_n$, and output (N, w).

Observe that by construction, $\ell \mid \phi(n)$ in Distribution 1. This is not necessarily the case in Distribution 2. With this assumption, security of the above PIR system naturally follows.

Theorem 7.2. *Under the ϕ-hiding assumption, the* PIR *scheme described above is recipient-secure.*

Proof. To show that the scheme is recipient-secure, we demonstrate that the view of the sender $\mathsf{View}_{\mathcal{S}}(j, (x_1, \dots, x_n))$ when the recipient queries at index i is computationally indistinguishable from its view $\mathsf{View}_{\mathcal{S}}(j, (x_1, \dots, x_n))$ when the recipient queries at index j. We use a simple hybrid argument. Define the following three experiments:

- **Experiment 1.** The receiver makes a query to index i. Write the receiver's query in the first step of the protocol as (N_1, w_1).
- **Experiment 2.** In the first step of the protocol, the receiver chooses $p, q \xleftarrow{\$} \mathsf{Primes}(\lambda)$, and sets $N_2 \leftarrow pq$. The receiver then chooses $w_2 \xleftarrow{\$} \mathbb{Z}_n$, and sends (N_2, w_2) to the sender.
- **Experiment 3.** The receiver makes a query to index j. Write the receiver's query in the first step of the protocol as (N_3, w_3).

Observe that in Experiment 1, (N_1, w_1) is a uniformly random draw from Distribution 1 in the ϕ-hiding definition. In Experiment 2, (N_2, w_2) is a uniformly random draw from Distribution 2 in the ϕ-hiding assumption. By the ϕ-hiding assumption, it follows that $(N_1, w_1) \overset{c}{\approx} (N_2, w_2)$. Analogously, we have $(N_2, w_2) \overset{c}{\approx} (N_3, w_3)$, from which we may conclude that

$$\mathsf{View}_{\mathcal{S}}(i, (x_1, \dots, x_n)) = ((N_1, w_1), (x_1, \dots, x_n)) \overset{c}{\approx} ((N_3, w_3), (x_1, \dots, x_n)) = \mathsf{View}_{\mathcal{S}}(j, (x_1, \dots, x_n)).$$

□

8. 4/29: Interactive Proof and Zero-Knowledge Proof Systems

8.1. Interactive Proofs.

In this lecture, we discuss interactive proofs and zero-knowledge proofs. First, we review and define some basic complexity classes.

Definition 8.1. A language L is in NP if there exists a *deterministic* polynomial time verifier $V : \{0,1\}^{p(n)} \times \{0,1\}^{q(n)} \to \{0,1\}$ such that $x \in L$ if and only if there exists a witness $w \in \{0,1\}^{q(n)}$ such that $V(x, w) = 1$. In this case, $p(n)$ and $q(n)$ are both polynomials of the input length n.

Definition 8.2. A language L is in MA if there exists a *deterministic* polynomial time verifier $V : \{0,1\}^{p(n)} \times \{0,1\}^{q(n)} \times \{0,1\}^{r(n)} \to \{0,1\}$ such that the following conditions hold:

- If $x \in L$, then
$$\Pr\left[r \xleftarrow{\$} \{0,1\}^{r(n)} : \exists w \in \{0,1\}^{q(n)}\ V(x,w,r) = 1\right] = 1.$$
- If $x \notin L$, then
$$\Pr\left[r \xleftarrow{\$} \{0,1\}^{r(n)} : \exists w \in \{0,1\}^{q(n)}\ V(x,w,r) = 1\right] < \frac{1}{2}.$$

As in the previous definition, $p(n), q(n), r(n)$ are all polynomials in the input length n.

In other words, we can view MA as an extension of NP where we allow the verifier to be randomized. Clearly, $\mathsf{NP} \subseteq \mathsf{MA}$ since any NP-verifier also suffices as an MA-verifier. Before continuing, we remark that we can also consider verifiers that can only check a few bits of the witness. We can define a class of languages that have verifiers that only check $O(1)$ of the bits of the witness. If we additionally allow the verifier access to $O(\log n)$ bits of randomness, this yields the complexity class PCP (the class of languages with probabilistically checkable proofs). A major result in complexity theory states that $\mathsf{NP} = \mathsf{PCP}$, that is all NP languages can be verified by a probabilistic algorithm using $O(\log n)$ bits of randomness and that only needs to read a constant number of bits of the witness (proof).

A fourth possible way of generalizing NP is to make the verification procedure interactive. Thus, rather than have the verifier be a single algorithm (such as in the case of NP languages), we now allow for interactions between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. This brings up to the class IP, that is, the set of languages for which we have interactive proofs.

Definition 8.3. An interactive proof system for a language $L \subseteq \{0,1\}^*$ is a 2-party protocol between verifier $\mathcal{V}$ (assumed to be probabilistic polynomial time) and prover $\mathcal{P}$ (assumed to be computationally unbounded) that satisfies the following properties. We write $\mathcal{P}(x) \leftrightarrow \mathcal{V}(x)$ to denote an interaction between the prover and verifier where the prover is trying to convince the verifier that $x \in L$. At the end of an interaction, the verifier either outputs accept or reject. Then, $(\mathcal{V}, \mathcal{P})$ must satisfy the following properties:

- **Completeness**: for all $x \in L$, $\mathcal{V}$ accepts x after interacting with $\mathcal{P}$:
$$\forall x \in L : \Pr[\mathcal{P}(x) \leftrightarrow \mathcal{V}(x) = \mathsf{accept}] = 1,$$
where the probability is taken over the randomness used in the protocol.
- **Soundness**: for all $x \notin L$ and for all provers $\mathcal{P}^\star$, $\mathcal{V}$ rejects with probability greater than $\frac{1}{2}$ after interacting with $\mathcal{P}^\star$:
$$\forall x \notin L : \Pr[\mathcal{P}(x) \leftrightarrow \mathcal{V}(x) = \mathsf{reject}] > \frac{1}{2}$$
where again the randomness is taken over the randomness used in the protocol.

Intuitively, completeness for the proof system states that an honest prover can *always* convince the verifier that $x \in L$ and soundness for the proof system states that a verifier will reject a false statement with probability at least $\frac{1}{2}$.

Definition 8.4. The complexity class $\mathsf{IP}[m]$ is the set of languages that can be recognized by interactive proofs with at most m rounds of interaction. A round of interaction is taken to be a message from either the prover or the verifier. The complexity class IP is the set of languages that can be recognized by some interactive proof system, that is, $\mathsf{IP} = \bigcup_{m \geq 1} \mathsf{IP}[m]$.

Fact 8.5. $\mathsf{NP} \subseteq \mathsf{MA} \subseteq \mathsf{IP}[1] \subseteq \mathsf{IP}[2] \subseteq \cdots \subseteq \mathsf{IP}$.

Proof. This basically follows by definition. All verifiers for NP-relations are also verifiers for MA-relations (that ignores the randomness). An MA protocol consists of just a single round (where the prover sends the proof), and so is trivially contained in $\mathsf{IP}(1)$. The other containments hold trivially by definition. □

Next, we demonstrate that randomness is very important in interactive proof systems. Specifically, we show that if the verifier is not allowed access to random bits, then the set of languages with interactive proof systems is just NP. In other words, without randomness, interactive proofs are useless.

Fact 8.6. *If $\mathcal{V}$ is deterministic, then the set of languages with an interactive proof system is* NP.

Proof. If the verifier is deterministic, the prover already knows how the verifier will respond to each message. Thus, the prover can compute the entire transcript of the interaction and send it to the verifier as its witness. Since the verifier is efficient, it can only read a polynomial number of bits, so this message is polynomial in size. To verify, the verifier can simply simulate the interaction using the prover's precomputed responses. □

Example 8.7. We give our first example of an interactive proof system. Consider the problem of graph isomorphism. Let $G_0 = (V, E_0)$ and $G_1(V, E_1)$ be two graphs with vertices V and edges E_1, E_2, respectively. We say that G_0 and G_1 are isomorphic ($G_0 \cong G_1$) if there exists a permutation $\pi : V \to V$ such that

$$(v_1, v_2) \in E_0 \iff (\pi(v_1), \pi(v_2)) \in E_1.$$

Note first that graph isomorphism is an NP problem: the isomorphism is the witness. To make things more interesting, consider the problem of graph non-isomorphism. We give an interactive proof for this problem. In this setting, both the prover and verifier have the graphs G_0 and G_1, and the prover seeks to convince the verifier that $G_0 \ncong G_1$.

Interactive Proof for Greaph Non-Isomorphism

(1) The verifier chooses a random bit $b \xleftarrow{\$} \{0,1\}$ and a random permutation $\pi \xleftarrow{\$} S_n$ where S_n denotes the symmetric group on n elements. The verifier applies π to the vertices of G_b to obtain an isomorphic copy G_b^π of G_b. The verifier sends $G^\star \leftarrow G_b^\pi$ to the prover.
(2) The prover replies with $b' \in \{0,1\}$.
(3) The verifier accepts if $b = b'$.

Theorem 8.8. *The protocol described in Example 8.7 is an interactive proof for graph non-isomorphism.*

Proof. We show that the protocol is complete and sound:

- **Completeness:** If $G_0 \ncong G_1$, then the verifier's challenge $G^\star$ will only be isomorphic to exactly one of G_0 and G_1. Since the prover is computationally unbounded, it can always determine which of G_0 or G_1 is isomorphic to $G^\star$. Thus, an honest prover will always convince the verifier.
- **Soundness:** If $G_0 \cong G_1$, then by construction, $G^\star$ is a uniform random graph isomorphic to G_1 and G_2. Thus, the best any prover can do is guess the bit b'. Therefore, the verifier rejects with probability $\frac{1}{2}$. This probability can be boosted arbitrarily close to 1 by repetition of the protocol.

□

As shown in Fact 8.6, randomness plays a critical role in the construction of interactive proof systems. In the context of interactive proofs systems, we can also distinguish between public-coin protocols and private-coin protocols. In a public-coin protocol, the prover is able to see all of the randomness used by the verifier, while in private-coin protocols, the prover does not get to observe the verifier's randomness. For example, the interactive proof for graph isomorphism that we gave above is an example of a private-coin protocol. In particular, the proof system would no longer be sound if the prover could see the random bits (in this case, the prover would learn both the bit b and the permutation π).

The canonical example of public-coin interactive proofs are Arthur-Merlin proofs. In an Arthur-Merlin proof, we have the same two parties: the prover (generally referred to as Merlin), and the verifier (generally referred to as Arthur). The interaction between Arthur and Merlin then takes the following form:

Structure of an Arthur-Merlin Proof

(1) Arthur begins by sending a random challenge to Merlin.
(2) Merlin responds to Arthur's challenge.

An Arthur-Merlin proof may consist of multiple rounds of such interactions, where the verifier sends a random string as a challenge and the prover replies with the challenge. At the end of the protocol, the verifier can perform some additional *deterministic* computation to decide whether to accept of reject the proof. Note however that the challenges are revealed iteratively. In particular, the prover does not immediately see all of the random bits that will be used by the verifier over the course of the protocol.

Definition 8.9. The complexity class $\mathsf{AM}[m]$ is the set of languages that can be recognized by Arthur-Merlin proofs with at most m rounds of interaction. A round of interaction is taken to be a message from either the prover or the verifier. We write $\mathsf{AM} = \mathsf{AM}[2]$.

While it might appear that this model is much more restrictive than private-coin interactive proofs, but it turns out that the two models are in fact equally expressive. In particular, we have the following result due to Goldwasser and Sipser.

Fact 8.10 ([GS86])**.** *For every $m \in \mathbb{N}$, $\mathsf{IP}[m] \subseteq \mathsf{AM}[m+2]$.*

In other words, any m-round private-coin interactive proof system can by simulated using a public-coin interactive proof system with just two additional rounds of interaction.

Fact 8.11. *It suffices to restrict the prover $\mathcal{P}$ to* PSPACE *algorithms.*

A natural question when studying interactive proofs is determining which languages have interactive proofs. It turns out that the class of languages with interactive proofs is precisely the class PSPACE, that is IP = PSPACE [Sha92]. We illustrate the main idea in this proof by proving a weaker statement that coNP $\subseteq$ IP.

Theorem 8.12. coNP $\subseteq$ IP*.*

Proof. We begin with a weaker statement that illustrates the techniques we will use to show PSPACE $\subseteq$ IP. To prove the claim, it suffices to describe an interactive proof for a coNP-Complete language. Specifically, we consider the language of graphs that are not 3-colorable. In fact, we will prove a stronger claim, that is, the problem of counting the number of valid 3-colorings for G is in IP. This particular problem of counting the number of 3-colorings for a graph G is in the complexity class #P. Observe that if we solve this problem, non-3-colorability amounts to simply checking that the number of valid 3-colorings of G is 0.

Let $G = (V, E)$ be a graph where $|V| = n$ and $|E| = m$. For all vertices u, let $x_u \in \{0, 1, 2\}$ denote the color assigned to vertex u. Consider the following polynomial

$$\hat{P}_G(x_1, \ldots, x_n) = \prod_{(u,v)\in E} (x_u - x_v).$$

If $(x_1, \ldots, x_u)$ is not a valid 3-coloring of G, then there must exist $(u, v) \in E$ such that $x_u = x_v$. In this case then, $\hat{P}_G(x_1, \ldots, x_n) = 0$. Note that since $x_i \in \{0, 1, 2\}$, $-2 \le x_u - x_v \le 2$ for all $(u, v) \in E$. We exploit this fact to now define a polynomial that evaluates to 1 on a valid coloring and 0 otherwise. Specifically, we need a function f such that

$$f(-2) = f(-1) = f(1) = f(2) = 1$$

and $f(0) = 0$. Using polynomial interpolation, we have that the polynomial $f(x) = \frac{5}{4}x^2 - \frac{1}{4}x^4$ satisfies these conditions (over the integers). Thus, define the polynomial

$$P_G(x_1, \ldots, x_n) = \prod_{(u,v)\in E} f(x_u - x_v).$$

By construction, $P_G(x_1, \ldots, x_n) = 1$ if and only if $(x_1, \ldots, x_n)$ is a valid coloring, Thus, the number of valid 3-colorings S of G can be computed by summing over P_G for all possible assignments to $(x_1, \ldots, x_n)$:

$$S = \sum_{x_1\in\{0,1,2\}} \sum_{x_2\in\{0,1,2\}} \cdots \sum_{x_n\in\{0,1,2\}} P_G(x_1, \ldots, x_n). \tag{8.1}$$

We describe an interactive proof system for (8.1). First, consider the following *univariate* polynomial in the first variable:

$$P_1(x) = \sum_{x_2\in\{0,1,2\}} \sum_{x_3\in\{0,1,2\}} \cdots \sum_{x_n\in\{0,1,2\}} P_G(x, x_2, \ldots, x_n).$$

Observe that if (8.1) holds, then $S = \sum_{x\in\{0,1,2\}} P_1(x)$. Moreover, since $\deg(f) = 4$, $\deg(P_1(x)) = 4m$. Thus, in the interactive proof, the prover can expand $P_1(x)$ into a polynomial of degree $4m$ and send the resulting polynomial to the verifier. This effectively amounts to sending the $4m$ coefficients of P_1. The verifier can then evaluate the polynomial at each $x \in \{0, 1, 2\}$ in polynomial time and verify that $S = \sum_{x\in\{0,1,2\}} P_1(x)$. Thus, if the prover is honest, then this relation will always hold. However, it is possible that a cheating prover can choose a different polynomial $\tilde{P}_1(x)$ such that $S = \sum_{x\in\{0,1,2\}} \tilde{P}_1(x)$. To protect against this attack, we leverage the fact that two distinct polynomials of degree d can agree on at most d points (a polynomial of degree d that is not identically 0 can have at most d roots). Specifically, the verifier chooses a random value r and evaluates $\tilde{P}_1(r)$. The verifier sends r to the prover, and now the prover must demonstrate that

$$\sum_{x_2\in\{0,1,2\}} \sum_{x_3\in\{0,1,2\}} \cdots \sum_{x_n\in\{0,1,2\}} P_G(r, x_2, \ldots, x_n) = \tilde{P}_1(r).$$

Observe that this is precisely the same problem, but in a polynomial with one fewer variable. Thus, we can apply the protocol recursively to prove each relation. The precise description of the protocol is given below. We will operate over the finite field $\mathbb{F}_p$ for some prime p.

Interactive Proof of $S = \sum_{x_1 \in \{0,1,2\}} \sum_{x_2 \in \{0,1,2\}} \cdots \sum_{x_n \in \{0,1,2\}} P_G(x_1, \dots, x_n)$.

$\mathsf{PolynomialSumProof}\left(S, (x_1, \dots, x_n), P_G\right)$:

(1) The prover constructs the univariate polynomial

$$P_1(x) = \sum_{x_2 \in \{0,1,2\}} \sum_{x_3 \in \{0,1,2\}} \cdots \sum_{x_n \in \{0,1,2\}} P_G(x, x_2, \dots, x_n).$$

Note that we will take $f(x) \in \mathbb{F}_p[x]$ to be the unique polynomial of degree 4 where $f(0) = 0$ and

$$f(-2) = f(-1) = f(1) = f(2) = 1.$$

The prover expands this to a polynomial of degree $4m$ and sends the coefficients to the verifier.

(2) If $n = 1$, the verifier checks whether $S = \sum_{x \in \{0,1,2\}} P_G(x)$. If so, the verifier accepts, and otherwise, rejects. If $n > 1$, then the verifier checks that $S = \sum_{x \in \{0,1,2\}} \tilde{P}_1(x)$, rejecting if this is not true. If the equality holds, the verifier chooses a random $r \xleftarrow{\$} \mathbb{F}_p$, and computes $\tilde{P}_1(r)$. Set $\tilde{P}'_G(x_2, \dots, x_n) \leftarrow P_G(r, x_2, \dots, x_n)$. Now, recursively invoke the protocol $\mathsf{PolynomialSumProof}\left(\tilde{P}_1(r), (x_2, \dots, x_n), \tilde{P}'_G\right)$ to prove the statement

$$\tilde{P}_1(r) = \sum_{x_2 \in \{0,1,2\}} \sum_{x_3 \in \{0,1,2\}} \cdots \sum_{x_n \in \{0,1,2\}} \tilde{P}'_G(r, x_2, \dots, x_n).$$

We argue that the above protocol constitutes a interactive proof for (8.1). First, we note that an honest prover will always be able to convince the verifier, so the proof system is complete. To argue soundness, we prove the following claim.

Claim 8.13. If (8.1) does not hold, the verifier will reject with probability at least $\left(1 - \frac{4m}{p}\right)^n$.

Proof. We proceed by induction on n. Consider the base case where $n = 1$. In this case, the verifier can directly compute $\sum_{x \in \{0,1,2\}} P_G(x)$. By assumption, since (8.1) does not hold, the verifier immediately rejects with probability 1. Next, for the inductive step, suppose we invoke $\mathsf{PolynomialSumProof}$ on a polynomial P_G over n variables. Consider the behavior of the prover on the first round. If the prover behaves honestly, that is, send the verifier $\tilde{P}_1(x) = P_1(x)$, then the verifier rejects since by assumption, $\sum_{x \in \{0,1,2\}} P_1(x) \neq S$. Thus, consider the case where the prover chooses a polynomial $\tilde{P}_1(x)$ such that $\sum_{x \in \{0,1,2\}} P_1(x) = S$. Now, we note that $\tilde{P}_1(x)$ and $P_1(x)$ are polynomials with degree $4m$, and thus, can agree on at most $4m$ points. Thus, with probability $1 - \frac{4m}{p}$, the verifier will choose a value $r \in \mathbb{F}_p$ where $\tilde{P}_1(r) \neq P_1(r)$. It this is the case, then in the recursive invocation of $\mathsf{PolynomialSumProof}$ (on a polynomial $\tilde{P}'_G$ over $n-1$ variables), the prover is left proving a false statement. By the inductive hypothesis, the verifier will reject with probability $\left(1 - \frac{4m}{p}\right)^{n-1}$. Thus, the probability that the verifier rejects is at least

$$\Pr\left[\text{verifier rejects}\right] \geq \left(1 - \frac{4m}{p}\right)\left(1 - \frac{4m}{p}\right)^{n-1} = \left(1 - \frac{4m}{p}\right)^n,$$

which proves the claim. ☐

Thus, by setting the size of the finite field appropriately (choosing sufficiently large values of p), the soundness error can be arbitrarily small. Since we have an interactive proof system for the number of 3-colorings a graph has, we can correspondingly prove that a graph is not 3-colorable. We conclude that $\mathsf{coNP} \subseteq \mathsf{IP}$. ☐

8.2. **Zero-Knowledge Proofs.** The concept of a zero-knowledge proof extends beyond an interactive proof in a critical way. Specifically we impose the additional requirement that at the end of the protocol, the verifier does not learn any additional information beyond the statement that was proved. We formalize this notion now:

Definition 8.14. An zero-knowledge proof system for a language $L \subseteq \{0,1\}^*$ is a 2-party protocol between a verifier $\mathcal{V}$ (probabilistic polynomial time) and a prover $\mathcal{P}$ (computationally unbounded) that satisfies the following properties:

- **Completeness**: For all $x \in L$, $\mathcal{V}$ accepts x after interacting with $\mathcal{P}$:
$$\forall x \in L : \Pr\left[\mathcal{P}(x) \leftrightarrow \mathcal{V}(x) = \mathsf{accept}\right] = 1,$$
where the probability is taken over the randomness used in the protocol.
- **Soundness**: For all $x \notin L$ and for all provers $\mathcal{P}^\star$, $\mathcal{V}$ rejects with probability greater than $\frac{1}{2}$ after interacting with $\mathcal{P}^\star$:
$$\forall x \notin L : \Pr\left[\mathcal{P}(x) \leftrightarrow \mathcal{V}(x) = \mathsf{reject}\right] > \frac{1}{2}$$
where again the randomness is taken over the randomness used in the protocol.
- **Zero-Knowledge**: For every PPT verifier $\mathcal{V}^\star$, there exists a PPT simulator $\mathcal{S}^\star$ such that for all $x \in L$,
$$\{\mathsf{transcript}(\mathcal{P}(x) \leftrightarrow \mathcal{V}^\star(x))\} = \{\mathcal{S}^\star(x))\},$$
where $\{\mathsf{transcript}\left(\mathcal{P}(x) \leftrightarrow \mathcal{V}^\star(x)\right)\}$ denotes the set of messages exchanged in the interaction between the prover $\mathcal{P}$ and the (possibly cheating) verifier $\mathcal{V}^\star$.

Intuitively, the zero-knowledge property states that any information that a cheating verifier $\mathcal{V}^\star$ could learn from interacting with the prover $\mathcal{P}$ could have been simulated by a simulator given only access to x. Thus, at the end of a zero-knowledge protocol, the verifier learns only that $x \in L$, and nothing more beyond what can be derived from that information alone. We can also prescribe a weaker notion of zero-knowledge, where we just require a simulator to exist for an interaction between the prover and an honest verifier.

Definition 8.15. A zero-knowledge proof system is (perfect) honest-verifier zero-knowledge (HVZK) if there exists a PPT simulator $\mathcal{S}$ such that for all $x \in L$,
$$\{\mathsf{transcript}\left(\mathcal{P}(x) \leftrightarrow \mathcal{V}(x)\right)\} = \{\mathcal{S}(x)\}.$$

Fact 8.16. *If perfectly hiding commitments exist, then* PSPACE *has zero-knowledge proofs.*

Proof. Show that NP has ZKP. Boost to PSPACE. $\square$

Example 8.17. We consider a ZKP for quadratic residuosity. Fix $N = pq$ and take $x \in \mathbb{Z}_N^*$. We want to prove in zero knowledge that x has a square root in $\mathbb{Z}_N^*$. We describe the following protocol. Suppose the prover and verifier have values (N, x) and the prover wants to convince the verifier that x has a square root in $\mathbb{Z}_N^*$. We use the following protocol:

Zero Knowledge Proof for Quadratic Residuosity

(1) The prover chooses $r \xleftarrow{\$} \mathbb{Z}_N$ and sends $y \leftarrow r^2 \pmod N$ to the verifier.
(2) The verifier replies with a random challenge $c \xleftarrow{\$} \{0,1\}$.
(3) The prover replies with $z \leftarrow r \cdot \alpha^c$ where $\alpha^2 = x \pmod N$.
(4) The verifier checks that $z^2 = yx^c \pmod N$ and rejects otherwise.

To show that this is a ZKP, we demonstrate that the above protocol is complete, sound, and zero-knowledge. We defer this proof to the following lecture (see Theorem 9.10).

9. 5/1: Zero Knowledge Proofs

9.1. **Zero Knowledge Proofs.** We continue our previous discussion of zero knowledge proofs. There are three notions of zero knowledge that we can consider. If in the zero-knowledge requirement, we require that for all $x \in L$,
$$\{\mathsf{transcript}(\mathcal{P}(x) \leftrightarrow \mathcal{V}^\star(x))\} = \{\mathsf{transcript}(\mathcal{S}^\star(x))\},$$

then the resulting scheme is perfect zero-knowledge. If we relax this definition to allow for statistically indistinguishability, that is, for all $x \in L$,

$$\{\mathsf{transcript}(\mathcal{P}(x) \leftrightarrow \mathcal{V}^\star(x))\} \overset{s}{\approx} \{\mathsf{transcript}(\mathcal{S}^*(x))\}\,,$$

then the resulting scheme is statistically zero-knowledge. Finally, if we further relax the definition to allow for computational indistinguishability, then the scheme is computationally zero-knowledge. We have the following facts:

Fact 9.1. *The class* SZK *of languages with statistical zero-knowledge proofs is contained in* $\mathsf{IP}[2]$.

Fact 9.2. *If a language* $L \in \mathsf{SZK}$*, then* $\bar{L} \in \mathsf{SZK}$.

Fact 9.3. *If* $L \in \mathsf{SZK}$*, then* L *is not* NP-Complete.

Proof. Otherwise, $\mathsf{NP} \subseteq \mathsf{SZK}$ and so $\mathsf{coNP} \subseteq \mathsf{SZP} \subseteq \mathsf{IP}[2]$, and the polynomial hierarchy collapses. □

Fact 9.4. *Quadratic residue, graph isomorphism all have statistical zero-knowledge proofs.*

Fact 9.5. *If perfectly hiding commitment exists, then* $\mathsf{PSPACE} = \mathsf{CZK}$*, where* CZK *is the class of languages that have computational zero-knowledge proofs.*

Proof. Can show that Hamiltonian cycle has computational zero-knowledge proofs. □

Definition 9.6. We say that in interaction proof system is black-box zero knowledge, if in addition to the usual completeness and soundness requirements, there exists a universal simulator $\mathcal{S}$ such that for all verifiers $\mathcal{V}^\star$,

$$\mathsf{transcript}\left(\mathcal{S}(x) \leftrightarrow \mathcal{V}^\star(x)\right) \overset{c}{\approx} \left\{\mathcal{S}(x)^{\mathcal{V}^\star(x)}\right\}.$$

Here, we write $\mathcal{S}(x)^{\mathcal{V}^\star(x)}$ to denote the simulator $\mathcal{S}$ with oracle access to $\mathcal{V}^\star$.

Theorem 9.7 ([GK96])**.** *A language* L *has 3-round black-box zero knowledge proofs with negligible soundness error if and only if* $L \in \mathsf{BPP}$.

There are potential ways of circumventing this limitation. There are several ways to construct 3-round zero knowledge proofs. One method leverages more powerful simulators. An alternative method is based on knowledge assumptions.

9.1.1. *Examples of Zero-Knowledge Proofs.* In this section, we give a few examples of zero-knowledge proofs. First, we show a perfect ZKP for the graph isomorphism problem [GMW86].

Example 9.8. Suppose we have two graphs G_0 and G_1 over n vertices and the prover is trying to prove $G_0 \cong G_1$. The protocol is very similar to the interactive proof for graph isomorphism described in 8.7.

Zero-Knowledge Proof of Graph Isomorphism

(1) The prover chooses a random permutation $\pi \xleftarrow{\$} S_n$ (S_n denotes the symmetric group over n elements) and a bit $b \xleftarrow{\$} \{0,1\}$. Then, the prover applies the permutation π to the vertices in G_b to obtain the graph G_b^π. The prover sends $G^\star \leftarrow G_b^\pi$ to the verifier.
(2) The verifier replies with a random challenge $c \xleftarrow{\$} \{0,1\}$.
(3) The prover replies with a permutation $\tau \in S_n$.
(4) The verifier applies τ to the vertices in $G^\star$ to obtain graph G'. It verifies that $G' = G_c$.

Theorem 9.9. *The protocol described in 9.8 is a perfect zero-knowledge proof of graph isomorphism.*

Proof. To show that the protocol is a zero-knowledge proof, we show that the protocol is complete, sound, and zero-knowledge. In this case, completeness is easy. If $G_0 \cong G_1$, then $G_0 \cong G^\star \cong G_1$. Therefore, for all $c \in \{0,1\}$, there exists a permutation $\tau \in S_n$ such that applying τ to $G^\star$ yields G'. Thus, an honest prover will always be able to answer the verifier's challenge.

To argue soundness, we observe that if $G_0 \ncong G_1$, the verifier will always reject with probability at least $\frac{1}{2}$. To see this, observe that the prover must send $G^\star$ before seeing the verifier's challenge. Therefore, if $G_0 \ncong G_1$, $G^\star$ will be congruent to *at most* one of G_0 or G_1. With probability at least $\frac{1}{2}$, the verifier will ask the prover to specify the isomorphism between $G^\star$ and G_c where $G^\star \ncong G_c$, but this is clearly impossible. Thus, with probability at least $\frac{1}{2}$, the verifier will reject when interacting with a cheating prover.

Finally, we show that the protocol is perfect zero-knowledge, that is, for all verifiers $\mathcal{V}^\star$, we show that there exists a simulator $\mathcal{S}^\star$ such that

$$\{\mathsf{transcript}\,(\mathcal{P}(G_0, G_1) \leftrightarrow \mathcal{V}^\star(G_0, G_1))\} = \{\mathcal{S}^\star(G_0, G_1)\}\,.$$

First, we note that a transcript between the prover and verifier will be given by $(G^\star, c, \tau)$. Our simulator $\mathcal{S}^\star$ will operate as follows:

Simulator $\mathcal{S}^\star$ for Graph Isomorphism ZKP

(1) Choose $b \xleftarrow{\$} \{0,1\}$ and $\pi \xleftarrow{\$} S_n$. Compute the permuted graph $G^\star \leftarrow G_b^\pi$.
(2) Send $G^\star$ to $\mathcal{V}^\star$. The verifier $\mathcal{V}^\star$ responds with a challenge $c \in \{0,1\}$.
   (a) If $b \neq c$, restart the protocol at step 1.
   (b) If $b = c$, then output $\left(G^\star, c, \pi^{-1}\right)$.

By construction, we have that $\{\mathsf{transcript}\,(\mathcal{P}(G_0, G_1) \leftrightarrow \mathcal{V}^\star(G_0, G_1))\} = \{\mathcal{S}^\star(G_0, G_1)\}$, since in the scenario where $b = c$, all steps of the actual protocol is correctly simulated. To argue that the simulator is efficient, it suffices to observe that b is a uniformly random bit, so $b = c$ with probability exactly $\frac{1}{2}$. We conclude that the protocol is a perfect zero-knowledge proof of graph isomorphism. □

A similar proof holds for the quadratic residuosity problem we discussed in the previous lecture (Example 8.17).

Theorem 9.10. *The protocol described in Example 8.17 is a perfect zero-knowledge proof for quadratic residuosity.*

Proof. We show that the protocol in Example 8.17 is complete, sound, and provides perfect zero-knowledge. First, we should that the protocol is complete. If x was indeed a quadratic residue in $\mathbb{Z}_N^*$, then an unbounded prover can always compute α such that $\alpha^2 = x \pmod N$. Thus, an honest prover will always be able to compute z and convince the verifier.

Next, consider the soundness of the scheme. We show that the verifier will always reject a cheating prover with probability at least $\frac{1}{2}$. Specifically, since x is not a quadratic residue, at most one of y and yx will be a quadratic residue. Thus, the prover can only find a z such that the verifier accepts with probability at most $\frac{1}{2}$ (since c is chosen uniformly at random by the verifier), so soundness holds.

Finally, we show that the proof is perfect zero-knowledge. Given a verifier $\mathcal{V}^\star$, we construct the following simulator $\mathcal{S}^\star$:

Simulator $\mathcal{S}^\star$ for Quadratic Residuosity ZKP

(1) Choose $b \xleftarrow{\$} \{0,1\}$ and $r \xleftarrow{\$} \mathbb{Z}_N$. Then, compute y as

$$y \leftarrow \begin{cases} r^2 \pmod N & b = 0 \\ \frac{r^2}{x} \pmod N & b = 1. \end{cases}$$

(2) Send y to $\mathcal{V}^\star$ to receive a challenge $c \in \{0,1\}$.
   (a) If $b \neq c$, restart the protocol at step 1.
   (b) If $b = c$, output (y, c, r).

We claim that $\{\mathsf{transcript}\,(\mathcal{P}(x) \leftrightarrow \mathcal{V}^\star(x))\} = \{\mathcal{S}^\star(x)\}$. Consider the output of the simulator. In the interactive proof system, y is a random quadratic residue, as is the case in the simulated transcript (since r is chosen randomly in $\mathbb{Z}_N$, both r^2 and $\frac{r^2}{x}$ are drawn uniformly at random from the set of quadratic residues). Thus, the first message to the verifier is properly distributed. The verifier chooses c, so the component is properly distributed in the simulated transcript. Lastly, given y and c, the value of z is also uniquely determined. Thus, we conclude that the simulated transcript is properly drawn from the distribution of valid transcripts of interactions between $\mathcal{P}(x)$ and $\mathcal{V}^\star(x)$. Finally, we argue that the simulator is efficient. Observe that the simulator succeeds if it "guesses" the verifier's challenge c correctly. Since the simulator chooses b uniformly from $\{0,1\}$, it will succeed with probability at least $\frac{1}{2}$. We conclude that $\mathcal{S}^\star$ efficiently simulates interactions between $\mathcal{P}(x) \leftrightarrow \mathcal{V}^\star(x)$. □

In the two schemes that we described above, we note that a prover will fool a verifier with probability $\frac{1}{2}$. We can boost the soundness probability to $1 - \varepsilon$ by repeated instantiations of the protocol. Observe that the zero-knowledge property is preserved since we can simulate each instantiation of the protocol sequentially. However, this is very impractical in practice; a natural extension would be to execute the protocol in parallel. For instance, in the quadratic residuosity zero-knowledge proof system, the prover can begin by sending $y_1 \leftarrow r_1^2, \ldots, y_n \leftarrow r_n^2$ and the verifier replies with $c_1, \ldots, c_n$. For each challenge $c_1, \ldots, c_n$, the prover replies accordingly. By construction, the soundness error is now reduces to $\frac{1}{2^n}$. Unfortunately, we no longer know how to prove that this protocol is zero-knowledge. Specifically, if we try to simulate the transcript, our simulator will run in 2^n time, which is no longer efficient. Note that if we were able to show that this protocol is zero-knowledge, then the proof cannot be black-box simulatable (otherwise, we have that quadratic residuosity is in BPP).

We consider an application of graph isomorphism to voting protocols using mixes. Each voter submits their vote v_i as $(g^r, h^r v_i)$ where $(g, h = g^x)$ is an ElGamal public key. The encrypted votes are submitted to the mixing authorities. Each mix applies a permutation π_i to the votes. At the end of the protocol, the votes are decrypted and the votes can be counted. Moreover, as long as at least one mix is honest, we cannot learn different voter's votes. One potential problem in this scenario is that the mixes could behave maliciously and introduce new votes, remove votes, and so on. To prevent this, at the end of the mixing process, each mix will include a zero-knowledge proof that its output is a valid permutation of the inputs. This is effectively the graph isomorphism problem. What remains is to demonstrate that two ElGamal ciphertexts encrypt the same underlying message.

9.2. **Chaum-Pederson Proofs.** As noted in the previous discussion of mixnets, we require a method of proving that two ElGamal ciphertexts are encryptions of the same message. To do this, let $g, h \in \mathbb{G}$ be elements in a group $\mathbb{G}$ of prime order q. Write $h = g^\alpha$ for some $\alpha \in \mathbb{Z}_q$. Then, $(g^r, h^r m)$ and $(g^s, h^s m')$ where $r, s \in \mathbb{Z}_q$ and $m, m' \in \mathbb{G}$ are ElGamal ciphertexts for messages m and m', respectively. Suppose $m = m'$ and consider the tuple

$$t = \left(g, h, \frac{g^r}{g^s}, \frac{h^r m}{h^s m'}\right) = \left(g, g^\alpha, g^{r-s}, g^{\alpha(r-s)}\right).$$

Thus, we see that t is a DDH tuple if and only if $m = m'$. Thus, proving that two ElGamal ciphertexts encrypt the same underlying message amounts to proving that t is a valid DDH tuple. We present the Chaum-Pederson zero-knowledge proof for demonstrating that a tuple is a DDH tuple [CP92]. In the Chaum-Pederson protocol, both the prover and verifier have a tuple $(g, x, y, z) = \left(g, g^a, g^b, g^c\right)$, and the prover wants to demonstrate that this is valid DDH tuple, namely, that $c = ab$.

Zero-Knowledge Proof that a Tuple is a DDH Tuple (Chaum-Pederson)

(1) The verifier chooses $t \xleftarrow{\$} \mathbb{Z}_q$ and sends a perfectly hiding (computationally binding) commitment of t to the prover.
(2) The prover chooses a random $s \xleftarrow{\$} \mathbb{Z}_q$ and sends $v \leftarrow g^s$ and $w \leftarrow x^s$ to the verifier.
(3) The verifier opens the commitment to t.
(4) The prover replies with $u \leftarrow s + bt$.
(5) The verifier accepts if $g^u = vy^t$ and $x^u = wz^t$.

Theorem 9.11. *The Chaum-Pederson protocol is a zero-knowledge proof that a tuple forms a* DDH*-tuple.*

Proof. We show that the Chaum-Pederson protocol is complete, sound, and zero-knowledge. First, we show that the protocol is complete by demonstrating that the verifier will always accept when interacting with an honest verifier. Suppose the input $(g, x, y, z) = (g, g^a, g^b, g^c)$ is a valid DDH tuple, in which case $c = ab$. Then, by definition we have that,

$$vy^t = g^s \left(g^b\right)^t = g^{s+bt} = g^u$$

and

$$wz^t = x^s g^{ct} = g^{as+ct} = g^{a(s+bt)} = x^u.$$

Next, we show that the Chaum-Pederson protocol is sound. Specifically, we demonstrate that if $(g, x, y, z) = \left(g, g^a, g^b, g^c\right)$ does not form a valid DDH tuple, then the prover cannot compute u such that the verifier accepts (except perhaps with negligible probability). In the second step of the protocol, the prover sends $v \leftarrow g^s$ and $w \leftarrow x^{s'}$ to the verifier. Note that a cheating prover might choose $s \neq s'$ in order to fool the verifier. Moreover, in step 2 of the protocol, we note that v, w can only be chosen independently of the verifier's challenge t, since at this point in the protocol, the prover only has a perfectly hiding commitment to t (which by definition, reveals no information about t). Then, after seeing the value t, the prover must produce u such that $g^u = vy^t$ and $x^u = wz^t$. In particular, we require

$$vy^t = g^{s+bt} = g^u \implies u = s + bt$$

and

$$wz^t = x^{s'} g^{ct} = g^{as'} g^{ct} = g^{au} \implies au = as' + ct.$$

Combining these two expressions, we note that such a u exists if and only if $a(s - s') = (c - ab)t$, or equivalently, $t = (c-ab)^{-1}a(s-s')$. Since t is chosen by the verifier uniformly at random (and independently of s, s'), this will only happen with probability $\frac{1}{q}$, which is negligible. Thus, a dishonest prover will be caught with probability $1 - \frac{1}{q} > \frac{1}{2}$. Note in fact that the soundness error in this case is negligible.

Finally, we show that the Chaum-Pederson protocol is zero-knowledge. First, we note that the transcript of the an interaction between the prover and verifier is specified by the tuple $(\mathsf{Commit}(t), v, w, t, u)$. Now, we show that given any dishonest verifier $\mathcal{V}^\star$, we can construct a simulator $\mathcal{S}^\star$ such that

$$\{\mathsf{transcript}\left(\mathcal{P}((g,x,y,z)) \leftrightarrow \mathcal{V}^\star((g,x,y,z))\right)\} \overset{c}{\approx} \{\mathcal{S}^\star((g,x,y,z))\}.$$

Simulator $\mathcal{S}^\star$ for Verifier $\mathcal{V}^\star$ for Chaum-Pederson Protocol

(1) Run $\mathcal{V}^\star$ to obtain a commitment $\mathsf{Commit}(t)$ to t.
(2) Choose $s \xleftarrow{\$} \mathbb{Z}_q$ and send $\hat{v} \leftarrow g^s$ and $\hat{w} \leftarrow x^s$ to $\mathcal{V}^\star$.
(3) $\mathcal{V}^\star$ replies with an opening to t.
(4) Choose $u \xleftarrow{\$} \mathbb{Z}_q$, and set $v \leftarrow g^u/y^t$ and $w \leftarrow x^u/z^t$.
(5) Output $(\mathsf{Commit}(t), v, w, t, u)$

We claim that this output is computationally indistinguishable from a valid transcript between $\mathcal{P}$ and $\mathcal{V}^\star$. First, we note that $\mathsf{Commit}(t)$ was chosen by $\mathcal{V}^\star$ so it must be properly distributed. Next, since the commitment scheme is computationally binding, given $\mathsf{Commit}(t)$ and the fact that $\mathcal{V}^\star$ is computationally bounded, the value of t is uniquely determined. Next, we note that since the commitment scheme is perfectly hiding, the values v, w are independent of t in the real scheme. Since u is chosen uniformly at random, $v = g^{u-bt}, w = g^{a(u-bt)}$ are properly distributed as in the real scheme. Finally given v, w, t, the value of u is uniquely determined (the only value that satisfies $g^u = vy^t$ and $x^u = wz^t$). Thus, since all the other components are properly distributed, we conclude that

$$\{\mathsf{transcript}\left(\mathcal{P}((g,x,y,z)) \leftrightarrow \mathcal{V}^\star((g,x,y,z))\right)\} \overset{c}{\approx} \{\mathcal{S}^\star((g,x,y,z))\}.$$

□

Observe that the Chaum-Pederson protocol is a 4-round zero-knowledge proof. Moreover, note what happens if we remove the commitment to t in the first step. In other words, suppose instead that $\mathcal{V}^\star$ just sent the value of t in step 3. It turns out that once we introduce this minor modification, we are no longer able to prove security. Specifically, if we were able to prove security of this modified protocol using a black-box simulator (such as the one we gave in the proof), that would immediately imply complexity theoretic lower bounds (Theorem 9.7). Nonetheless, the 3-round protocol without the commitments is still honest-verifier zero-knowledge. To see this, simply note that to simulate the transcript between the prover and an honest verifier, the simulator can simply choose $t \xleftarrow{\$} \mathbb{Z}_q$ and proceed with the rest of the simulation without modification. Since t is chosen uniformly at random, it is again independent of (u, v), so the proof carries through without modification.

10. 5/6: Proofs of Knowledge

10.1. Proofs of Knowledge.
In this lecture, we discuss proofs of knowledge.

Definition 10.1. Let $R(x, w)$ be an NP-relation and let L be the corresponding language. Then a protocol between a prover $\mathcal{P}(x, w)$ and a verifier $\mathcal{V}(x)$ is a proof of knowledge if the following two conditions holds:

- **Completeness**: For all $x \in L$, $\mathcal{V}$ accepts x after interacting with $\mathcal{P}$:
$$\forall x \in L : \Pr\left[\mathcal{P}(x, w) \leftrightarrow \mathcal{V}(x) = \mathsf{accept}\right] = 1,$$
where the probability is taken over the randomness used in the protocol.
- **Extraction:** For all efficient provers $\mathcal{P}^\star$ and for all x such that $[\mathcal{P}^\star(x, w) \leftrightarrow \mathcal{V}(x)]$ accepts, there exists an extractor $\mathcal{E}$ such that $\mathcal{E}^{\mathcal{P}^\star}$ (the extractor $\mathcal{E}$ given oracle access to $\mathcal{P}^\star$) that outputs w with non-negligible probability.

We can easily extend our definition to the notion of zero-knowledge proof of knowledge where we require the interactive proof to also be zero-knowledge. We begin with an example of an honest-verifier zero-knowledge PoK for the discrete log (Schnorr's protocol [Sch89]). Here, the prover and verifier have two group elements $g, h \in \mathbb{G}$ in a group $\mathbb{G}$ of prime order q. The prover's goal is to prove that it knows x such that $g^x = h$. The protocol proceeds as follows:

Proof of Knowledge of Discrete Log (Schnorr)

(1) The prover chooses $r \xleftarrow{\$} \mathbb{Z}_q$ and sends $t \leftarrow g^r$ to the verifier.
(2) The verifier replies with $c \xleftarrow{\$} \mathbb{Z}_q$.
(3) The prover replies with $s \leftarrow r + cx$.
(4) The verifier accepts if $g^s = th^c$.

Theorem 10.2. *Schnorr's protocol is a honest-verifier zero-knowledge* PoK *for discrete log.*

Proof. First, we show completeness. Suppose the prover knows x such that $h = g^x$. Then, it will always be able to compute $r + cx$. Completeness follows since
$$th^c = g^r \left(g^x\right)^c = g^{r+cx} = g^s,$$
so the verifier will always accept.

Next, we show that the proof is honest-verifier zero-knowledge. Specifically, we construct the following simulator $\mathcal{S}$ for an honest verifier $\mathcal{V}$:

Simulator $\mathcal{S}$ for Honest Verifier $\mathcal{V}$ for Schnorr's Protocol

(1) Choose a random $r \xleftarrow{\$} \mathbb{Z}_q$ and send $\hat{t} \leftarrow g^r$ to the verifier.
(2) $\mathcal{V}$ replies with $c \in\xleftarrow{\$} \mathbb{Z}_q$.
(3) Choose $s \xleftarrow{\$} \mathbb{Z}_q$ and compute $x \leftarrow g^s/h^c$ and output (t, c, s).

It is easy to see that the output of the simulator is distributed identically with a transcript between an honest prover and honest verifier. Specifically, the challenge c is drawn uniformly at random from $\mathbb{Z}_q$ (and independently of $\hat{x}$). Next, since s is chosen uniformly at random, x is a uniformly random element in $\mathbb{G}$. Once x and c have been chosen, s is uniquely determined in a valid transcript. Thus we conclude that any valid transcript between $\mathcal{P}(x, w)$ and $\mathcal{V}(x)$ can be simulated.

Finally, to show that the protocol is a proof of knowledge, we show that we can construct an extractor that can extract the witness x. Specifically, we require that if an honest verifier $\mathcal{V}$ accepts after interacting with a prover $\mathcal{P}^\star$, then there exists an efficient extractor $\mathcal{E}$ such that $\mathcal{E}^{\mathcal{P}^\star}$ extracts x. In other words, if $\Pr\left[[\mathcal{P}^\star \leftrightarrow \mathcal{V}] = \mathsf{accept}\right]$ is non-negligible, then $\Pr\left[\mathcal{E}^{\mathcal{P}^\star} \to x\right]$ is non-negligible. For simplicity, we first assume that $\mathcal{P}^\star$ convinces the verifier with probability 1. The extractor operates as follows:

Extractor for Schnorr's Protocol

(1) Run $\mathcal{P}^\star$ to obtain $t = g^r$ for some r.
(2) Choose $c_1 \xleftarrow{\$} \mathbb{Z}_q$ and sends c_1 to $\mathcal{P}^\star$.
(3) The prover will output s_1 such that $g^{s_1} = th^{c_1}$.

(4) Rewind the prover $\mathcal{P}^\star$ to step 2 of Schnorr's protocol. Specifically, we rollback $P^\star$ to the state immediately after it sends $t = g^r$.
(5) Choose $c_2 \xleftarrow{\$} \mathbb{Z}_q$ and send c_2 to $\mathcal{P}^\star$.
(6) Now, $\mathcal{P}^\star$ will output s_2 such that $g^{s_2} = th^{c_2}$.
(7) The extractor outputs $x = \frac{s_1 - s_2}{c_1 - c_2}$.

By construction, $g^{s_1} = th^{c_1}$ and $g^{s_2} = th^{c_2}$. Then, $g^{s_1 - s_2} = h^{c_1 - c_2}$, so we have $x = \frac{s_1 - s_2}{c_1 - c_2}$. To extend to the scenario where the prover succeeds with probability ε, then using the rewinding lemma, the extractor will work with probability ε^2. □

PoK protocols that consist of a commitment, a random challenge, and a response are also known as Σ-protocols. Concretely, Schnorr's protocol described above is an example of a Σ-protocol.

10.2. **Fiat-Shamir Heuristic.** We now demonstrate how to transform the interactive protocol described above to a non-interactive variant. Specifically, we replace the challenge with the output of a hash function evaluated on the statement and commitment. For example, in the case of Schnorr protocols, we have

(1) The prover chooses $r \xleftarrow{\$} \mathbb{Z}_q$ and computes $t = g^r$. Then, the prover computes the challenge $c \leftarrow H(g, h, t)$ and the response $s \leftarrow r + cx$. The prover outputs (t, s).
(2) The verifier then checks that $c = H(g, h, t)$ and that $g^s = th^c$.

If we model H as a random oracle, the above protocol remains secure.

We can use this approach to construct signature schemes from Σ-protocols. For example, the Schnorr signature scheme operates as follows:

- KeyGen: Choose $x \xleftarrow{\$} \mathbb{Z}_q$ and compute $h \leftarrow g^x$. The public key pk is then h and the secret key sk is x.
- Sign: Choose $r \xleftarrow{\$} \mathbb{Z}_q$ and $t = g^r$. To sign a message m, compute $c = H(g, h, t, m)$ and compute $s = r - xc$. Output $\sigma = (t, s)$.
- Verify: Given the message m, compute $c \leftarrow H(g, h, t, m)$ and verify that $g^s = th^c$.

We can optimize the above scheme by taking the signature to be (c, s). Then, to verify the signature, we first compute $t \leftarrow g^s / h^c$ and compute $\hat{c} \leftarrow H(g, h, u, m)$ and verify that $c = \hat{c}$. In practice, c (the challenge) is much smaller than t (a random group element): roughly 128 bits versus 2048 bits.

Theorem 10.3. *Schnorr signatures are existentially unforgeable under a chosen message attack assuming discrete log is hard in $\mathbb{G}$ and modeling H as a random oracle.*

Proof. Suppose we have an algorithm $\mathcal{A}$ that forges signatures. We use $\mathcal{A}$ to build an algorithm $\mathcal{B}$ that solves the discrete log problem in $\mathbb{G}$. The input to $\mathcal{B}$ consists of two group elements $g, h \in \mathbb{G}$, and our goal is to output $x \in \mathbb{Z}_q$ such that $g^x = h$. Our adversary proceeds as follows:

(1) Set the public key of the verification scheme to be $\mathsf{pk} = (g, h)$ and give pk to $\mathcal{A}$.
(2) At some point, algorithm $\mathcal{A}$ might issue a chosen-message query m. Using the honest-verifier simulator $\mathcal{S}$, generate (t, c, s) and reply with (t, z). Program the random oracle $H(g, h, t, m) = c$. Note c is drawn uniformly at random, so things are well-distributed.
(3) When $\mathcal{A}$ submits a forgery at the end of the game, use the extractor algorithm $\mathcal{E}$ to obtain x.

□

The Schnorr signature scheme serves as the basis for the DSA signature protocol.

11. 5/8: Witness-Indistinguishable Proofs

11.1. **Witness-Indistinguishable Proofs.** In this lecture, we discuss the notion of witness-indistinguishable proofs for NP languages, which is similar to, although much weaker than a zero-knowledge proof. The high level idea in a witness-indistinguishable proof for an NP language is that the verifier does not learn which witness the prover used in the proof. However, the proof might leak other information about the set of witnesses or in the case there there is only one witness, the actual witness to the relation. We define the specific notion below.

Definition 11.1. Let L be an NP language with relation R. A witness-indistinguishable proof system (WIP) proof system for L is a 2-party protocol between a verifier $\mathcal{V}$ (PPT) and a prover $\mathcal{P}$ such that

- For all PPT verifiers $\mathcal{V}^\star$ and for all witnesses w_0, w_1 such that $R(x, w_0) = 1 = R(x, w_1)$, then
$$\{\mathsf{transcript}\,(\mathcal{P}(x, w_0) \leftrightarrow \mathcal{V}^\star(x))\} \overset{c}{\approx} \{\mathsf{transcript}\,(\mathcal{P}(x, w_1) \leftrightarrow \mathcal{V}^\star(x))\}\,.$$
- $(\mathcal{P}, \mathcal{V})$ is a proof of knowledge for membership in L.

We now demonstrate that witness-indistinguishability is a weaker notion than zero-knowledge in that if a proof system is zero-knowledge, then it automatically satisfies witness-indistinguishability. In fact, witness-indistinguishability is a strictly weaker notion. As we will see, there exist schemes that are witness-indistinguishable, but certainly not zero-knowledge.

Lemma 11.2. *If $(\mathcal{P}, \mathcal{V})$ is a zero-knowledge* PoK *for an* NP *language L, then $(\mathcal{P}, \mathcal{V})$ is a witness-indistinguishable* PoK *for L:*

Proof. Suppose $R(x, w_0) = 1 = R(x, w_1)$. Then. for all PPT verifiers $\mathcal{V}^\star$, zero-knowledge of $(\mathcal{P}, \mathcal{V})$ implies that there exists a simulator $\mathcal{S}^\star$ such that
$$\{\mathsf{transcript}\,(\mathcal{P}(x, w_0) \leftrightarrow \mathcal{V}^\star(x))\} \overset{c}{\approx} \{\mathcal{S}^\star(x)\} \overset{c}{\approx} \{\mathsf{transcript}\,(\mathcal{P}(x, w_1) \leftrightarrow \mathcal{V}^\star(x))\}\,.$$
□

Recall from our discussion on zero-knowledge

Theorem 11.3 ([FS90])**.** *If $(\mathcal{P}, \mathcal{V})$ is witness-indistinguishable, then $\left(\mathcal{P}^\parallel, \mathcal{V}^\parallel\right)$ is also witness-indistinguishable, where $\left(\mathcal{P}^\parallel, \mathcal{V}^\parallel\right)$ denotes the parallel composition of $(\mathcal{P}, \mathcal{V})$.*

Proof. Follows by a simple hybrid. □

Since WIPs are much weaker than ZKPs, it is possible to have three-round WIPs. We give one such example below.

Example 11.4 ("Or" Proofs)**.** Consider the following relation (for an "or" language)
$$R\left((g, h_0, h_1), w\right) = 1 \iff g^w = h_0 \text{ or } g^w = h_1,$$
where $g, h_0, h_1 \in \mathbb{G}$ are elements in some group $\mathbb{G}$ (of prime order p). Specifically, the prover's goal is to convince the verifier that it knows the discrete log of at least one of h_0 and h_1. We give a 3-round witness-indistinguishable proof for this below. Here, the prover holds (g, h_0, h_1, w) and the verifier holds (g, h_0, h_1). Without loss of generality, suppose that $h_0 = g^w$. An analogous protocol applies when $h_1 = g^w$.

Witness-Indistinguishable Proof of Knowledge of One-of-Two Discrete Logs

(1) The prover chooses $r_0 \xleftarrow{\$} \mathbb{Z}_p$ and computes $t_0 = g^{r_0}$. Let $\mathcal{S}$ be the simulator for the Schnorr proof of knowledge of discrete log (see Theorem 10.2). The prover runs $\mathcal{S}$ on input (g, h_1) to obtain values (t_1, c_1, s_1) where $t_1 h_1^{c_1} = g^{s_1}$. The prover sends t_0 and t_1 to the verifier.
(2) The verifier replies with a challenge $c \xleftarrow{\$} \mathbb{Z}_p^\star$.
(3) The prover computes $c_0 \leftarrow c/c_1$ and $s_0 \leftarrow r_0 + wc_0$. The prover sends (c_0, s_0, s_1) to the verifier.
(4) The verifier first computes $c_1 \leftarrow c/c_0$. Then, the verifier checks that for $b \in \{0, 1\}$, $g^{s_b} = t_b h_b^{c_b}$.

Observe that the above protocol effectively consists of two Schnorr protocols run simultaneously. Using knowledge of the witness w, the prover is able to answer the verifier's challenge in one of the Schnorr instances; for the other instance, the prover simply simulates a valid transcript. We now show that the above protocol is correct and witness-indistinguishable.

Theorem 11.5. *The protocol given above is a witness-indistinguishable proof of knowledge for one-of-two possible discrete logs.*

Proof. First, we show that the proof system is complete. This is very similar to the proof of completeness for Schnorr proofs. Again, suppose without loss of generality that the prover knows w such that $h_0 = g^w$. Then, we have, for $b = 0$,

$$g^{s_0} = g^{r_0 + wc_0} = g^{r_0} h^{c_0} = t_0 h_0^{c_0}.$$

The $b = 1$ case holds by construction of c_1, t_1, s_1. Next, we show that the proof is a proof of knowledge. Specifically, we give an extractor that will extract a valid witness. Our extractor works very similarly to the extractor we gave for Schnorr proofs. Let $\mathcal{P}^\star$ be a prover such that $\Pr[[\mathcal{P}^\star \leftrightarrow \mathcal{V}] = \mathsf{accept}]$ is non-negligible. Then, our extractor works as follows:

Extractor for WIP of Knowledge of One-of-Two Discrete Logs

(1) Run $\mathcal{P}^\star$ to obtain $t_0 = g^{r_0}$ and $t_1 = g^{r_1}$ for some r_0, r_1.
(2) Choose $c_1 \xleftarrow{\$} \mathbb{Z}_q$ and sends c_1 to $\mathcal{P}^\star$.
(3) The prover will output (c_0, s_0, s_1) such that for $b \in \{0,1\}$, $g^{s_b} = t_b h^{c_b}$.
(4) Rewind the prover $\mathcal{P}^\star$ to step 2 of the protocol. Specifically, we rollback $P^\star$ to the state immediately after it sends t_0 and t_1.
(5) Choose $c_2 \xleftarrow{\$} \mathbb{Z}_q$ and send c_2 to $\mathcal{P}^\star$.
(6) Now, $\mathcal{P}^\star$ will output (c_0', s_0', s_1') such that for $b \in \{0,1\}$, $g^{s_b'} = t_b h^{c_b'}$.
(7) If $s_b = s_b'$ for both $b = 0$ and $b = 1$, abort. Otherwise, for b such that $s_b \neq s_b'$, output
$$w = \frac{s_b - s_b'}{c_b - c_b'}.$$

We claim that if $\Pr[[\mathcal{P}^\star \leftrightarrow \mathcal{V}] = \mathsf{accept}]$, then the above extractor will produce a witness w for (g, h_0, h_1) with non-negligible probability. For simplicity, we will assume that $\mathcal{P}^\star$ convinced the verifier with probability 1. First, we note that if there is $b \in \{0,1\}$ for which $s_b \neq s_b'$, then from the relations above, we have

$$g^{s_b - s_b'} = h_b^{c_b - c_b'} \implies w_b = \frac{s_b - s_b'}{c_b - c_b'},$$

where $g^{w_b} = h_b$. Just as in the construction of the extractor for Schnorr's protocol, we rewind the prover to the point after it has sent t_0, t_1, which effectively function as commitments to r_0, r_1. Having fixed r_0, r_1, we can extract one of the discrete logs by having the prover answer two different challenges. What remains is to show that with non-negligible probability, $s_b \neq s_b'$. This is not hard to see. As we just noted, the value of r_0 and r_1 is the same for the two challenges. Next, $c' \neq c$, so $c_b \neq c_b'$ for some $b \in \{0,1\}$. The values of r_b and c_b uniquely define s_b, and since r_b is the same in both challenges while c_b (for some b) is different in the two challenges, s_b is different for some b. Thus, the above extraction algorithm will extract w_b. Using the same type of boosting argument as in the proof of extractability for Schnorr's protocol, we can extract the witness even when $\mathcal{P}^\star$ succeeds in convincing the verifier with probability $\frac{1}{2} + \varepsilon$ for non-negligible ε.

We note that in this particular protocol, if the prover only knew one of the discrete logs w_b, then the extractor will extract the value w_b. Recall from the protocol that for the discrete log the prover did not know, it simulated the parameters. Once the prover commits to these simulated values (by committing to r_{1-b} in the first step of the protocol), the prover is bound to use the same value of c_{1-b} when responding to both challenges. If the prover tried to use a different value of c_{1-b}, it would not be able to compute s_{1-b} that satisfied the necessary relation (since that would require knowledge of w_{1-b}). Therefore, if the prover only knew only one of the discrete logs, it would be forced to reply with the same value s_{1-b} in response to both challenges. In this case then, $s_b \neq s_b'$, from which the extractor will extract the witness w_b.

Lastly, we demonstrate that the protocol is witness-indistinguishable. To show this, we first define the transcript of the protocol. In this case, the transcript consists of the 6-tuple $(t_0, t_1, c, c_0, s_0, s_1)$. We argue that regardless of which witness the prover used, the resulting transcripts are computationally indistinguishable. We consider each term separately. First, t_0 and t_1 are both drawn uniformly at random from $\mathbb{G}$ (irrespective of the choice of the witness). Next, c is a value chosen by the verifier, and thus, independent of the witness used by the prover. Next, c_0 is a uniform random element in $\mathbb{Z}_p$ (since we simulated c_1, and in the simulation, c_1 is drawn uniformly from $\mathbb{Z}_p$). Finally, we note that s_0 and s_1 are uniquely defined by the other parameters. Since none of the components in the transcript depend on the choice of witness, we conclude that the protocol is witness-indistinguishable. In fact, by the same argument, we can demonstrate that the protocol is honest-verifier zero-knowledge. □

While the above protocol for proving knowledge of one-of-two discrete logs is honest-verifier zero-knowledge, it is not zero-knowledge. Note that if it were zero-knowledge, then we would have a 3-round black box zero-knowledge protocol, which would immediately imply lower bounds (Theorem 9.7). However, we can modify the protocol to be zero-knowledge by having the verifier first commit to the challenge c. Note that this would result in a 4-round protocol. With this modification, note that the resulting protocol is no longer a proof of knowledge, since we cannot build an extractor anymore. Thus, if we constrain the number of rounds in a protocol, the zero-knowledge and proof of knowledge requirements can be in conflict with each other.

11.1.1. *Identification Protocols from Witness-Indistinguishability.* We give an application of witness-indistinguishability proofs to actively secure identification protocols. An identification protocol is a two-party protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ and in general, the prover is trying to authenticate to $\mathcal{V}$ (generally by proving knowledge of some secret). For example, in password-based authentication, the prover authenticates by sending a password to the verifier and the verifier checks the validity of the password. In our setting, we consider a discrete-log based identification protocol. In this setting, the prover holds a secret key $\mathsf{sk} = v$ and the verifier holds a public key $\mathsf{pk} = g^v$. To authenticate, the prover will prove knowledge of the secret exponent v. First, we restrict our attention to passive attacks on our system. A passive adversary is allowed to see many honest interactions $\mathcal{P} \leftrightarrow \mathcal{V}$ between the prover and the verifier, but cannot directly interact with the prover. The adversary then attempts to authenticate itself to the verifier.

We show that to construct a passive-secure identification scheme, it suffices to have an honest-verifier zero knowledge PoK. Suppose we have a passive adversary $\mathcal{A}$ for our identification scheme. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that computes the discrete log. Algorithm $\mathcal{B}$ takes as input (g, h) and its goal is to produce v such that $h = g^v$.

Adversary $\mathcal{B}$ that Computes Discrete Log Using Adversary $\mathcal{A}$

(1) We simulate sessions for $\mathcal{A}$.
(2) When $\mathcal{A}$ fools the verifier, use the extractor to extract the secret v.

To defend against an active attacker, we require a witness indistinguishable proof of knowledge. In an active attack, the adversary can first interact with the prover. Then, the attacker interacts with the verifier (without the prover), and wins if he is able to successfully authenticate. The standard solution is to use signature-based challenge response systems. We show how we can use witness-indistinguishability can achieve this. Here, the prover holds (g, h_0, h_1, w) and the verifier holds (g, h_0, h_1). To authenticate, the prover and verifier engage in the witness-indistinguishable proof of knowledge described above. The reduction to discrete log is below. As input, we are given $g, h = g^x$.

(1) Send to $\mathcal{A}$ the public key $\mathsf{pk} = (h, g_0, g_1)$. Choose $b \xleftarrow{\$} \{0,1\}$ and pick $r \xleftarrow{\$} \mathbb{Z}_p$. Set $h_b \leftarrow g^r$ and $h_{1-b} \leftarrow h$.
(2) $\mathcal{A}$ can now interact with the prover. We can answer all of these queries correctly since we know the discrete log of g_b.
(3) When $\mathcal{A}$ interacts and authenticates to the verifier, we use the extraction protocol, and with probability $\frac{1}{2}$, we extract x.

Note that because the adversary does not know which witness was used, the advantage of the above algorithm is always $\frac{1}{2}$. Additionally, note that the above protocol is a secure 3-round active identification protocol based on discrete log and does not require random oracles.

12. 5/13: Proofs of Relations and Elliptic Curves

12.1. **Zero Knowledge Proofs of Relations.** Before proceeding to the next topic, we show how to prove relations in zero-knowledge. Let $\mathsf{pk} = (g, h)$ and $\mathsf{sk} = \alpha$ be a ElGamal public/private key pair ($h = g^\alpha$) and suppose we have three ElGamal ciphertexts for messages m_0, m_1, m_2:

$$c_0 = (g^{r_0}, h^{r_0} g^{m_0}) \qquad c_1 = (g^{r_1}, h^{r_1} g^{m_1}) \qquad c_2 = (g^{r_2}, h^{r_2} g^{m_2}) .$$

We first note that proving $m_2 = m_0 + m_1$ is easy. Let $\hat{c}$ denote the first component of the ciphertext and $\tilde{c}$ denote the second component, that is, write the ciphertexts as $c = (\hat{c}, \tilde{c})$. To prove the sum relation, it suffices to check that

$$\left(g, h, \frac{\hat{c}_0 \cdot \hat{c}_1}{\hat{c}_2}, \frac{\tilde{c}_0 \cdot \tilde{c}_1}{\tilde{c}_2} \right)$$

is a valid DDH tuple. As shown in Section 9.2, this can be done via a Chaum-Pederson proof. To see that this suffices, observe that if $m_2 = m_0 + m_1$, then

$$\frac{\hat{c}_0 \cdot \hat{c}_1}{\hat{c}_2} = g^{\alpha_0+\alpha_1-\alpha_2} \qquad \frac{\tilde{c}_0 \cdot \tilde{c}_1}{\tilde{c}_2} = h^{\alpha_0+\alpha_1-\alpha_2}.$$

Next, we consider the case of proving a more complicated relation between the messages, namely that $m_2 = m_0 \cdot m_1$.

We describe an application of proving more expressive relations to voting protocols. Suppose we have n voters and a central voting authority that will be used to aggregate the votes. First, the voters in conjunction with the voting center generate an ElGamal public/private key pair $\mathsf{pk} = (g, h)$ and $\mathsf{sk} = \alpha$ with $h = g^\alpha$. Note that we share the secret key across the parties so no individual party can decrypt a ciphertext. To cast a vote for a candidate, each user i constructs an ElGamal encryption $(\hat{c}_i, \tilde{c}_i) = \left(g^{r_i}, h^{r_i} g^{b_i}\right)$ of its vote $b_i \in \{0, 1\}$ and sends it to the voting center. The voting center then homomorphically sums the votes by computing

$$(\hat{c}, \tilde{c}) = \left(\prod_{i=1}^n \hat{c}_i, \prod_{i=1}^n \tilde{c}_i\right) = \left(g^{\sum_i r_i}, h^{\sum_i r_i} g^{\sum_i b_i}\right).$$

Once all the votes have been aggregated, the parties engage in a joint decryption protocol where they compute

$$\left(g^{\sum_i r_i}\right)^\alpha = h^{\sum_i b_i},$$

from which they may recover $g^{\sum_i b_i}$. If everyone behaved honestly, then $0 \leq \sum_i b_i \leq n$, so we can simply try each possible value $0 \leq x \leq n$ and check whether $g^x = g^{\sum_i b_i}$. This is linear work in the number of voters, and thus, is efficient.

As we noted, the above protocol is only secure if everyone is honest. A dishonest voter can manipulate the vote by submitting votes that are not 0 or 1 (for instance, if someone wanted their vote to count twice for candidate 1, they might submit an encryption of 2 instead). To defend against this attack, we require the voters to submit a zero-knowledge proof that their vote corresponds to an encryption of $b_i \in \{0, 1\}$. There are two ways we can do this:

- **"Or" Proofs:** We can use the "or" proofs from Example 11.4 to show that one of the following tuples is a DDH tuple: $(g, h, \hat{c}_i, \tilde{c}_i)$ or $(g, h, \hat{c}_i, \tilde{c}_i/g)$. It is easy to check that $(g, h, \hat{c}_i, \tilde{c}_i)$ forms a DDH tuple when $b_i = 0$ and that $(g, h, \hat{c}_i, \tilde{c}_i/g)$ form a DDH tuple with $b_i = 1$.
- **Multiplication Proofs:** We can also leverage the multiplicative proofs from above. Specifically, we observe that $b_i^2 = b_i$ if and only if $b_i \in \{0, 1\}$. Thus, we use the multiplicative proof to demonstrate that $(\hat{c}_i, \tilde{c}_i)$ is an encryption of b_i^2 (given that encryption $(\hat{c}_i, \tilde{c}_i)$ of b_i).

12.2. Elliptic Curves.

In this section, we give a brief history into the development of elliptic curves, first from an algebraic perspective, then from an analytic perspective.

12.2.1. *Elliptic Curves: an Algebraic Perspective.*

Around 200 AD, the Greek mathematician Diophantus began examining rational solutions to equations for conic sections such as those of the form

$$ax^2 + by^2 + cxy + dx + fy + g = 0.$$

Diophantus' work built off of previous work by the Pythagoreans who were interested in rational solutions for $x^2 + y^2 = 1$. Diophantus discovered that for quadratic equations, the rational points on the conic sections coincided precisely with the rational numbers. Specifically, given a rational point, consider any line that goes through the point. The two points where the line intersects the conic are both rational points (the product of the roots of a quadratic with rational coefficients must be rational). Additionally, Diophantus studied the rational points on cubics such as $y^2 = x^3 + ax + b$ where $a, b \in \mathbb{Q}$. We say that the curve $f \in \mathbb{Q}[x, y]$ has a double point at (x, y) if $f(x, y) = 0$ and moreover, if

$$\left.\frac{\partial f}{\partial x}\right|_{(x,y)} = \left.\frac{\partial f}{\partial y}\right|_{(x,y)} = 0.$$

Then, if we take a rational line that crosses a double point, then the intersection of the line with the curve is again rational. Once again, the rational points on the curve correspond bijectively with the rational numbers. We characterize when (x_0, y_0) is a double point. Now,

$$\frac{\partial f}{\partial y} = 2y = 0.$$

Thus, a double point occurs when $x^2 + ax + b = 0$ has a double root, which happens if and only if the determinant of the cubic is 0, that is, if $-16\left(4a^3 + 27b^2\right) = 0$. We refer to these curves as genus-0 curves. Note that these are *not* elliptic curves since they correspond bijectively with the line.

We also consider curves that have no double roots. Given two rational points on the curve, if we consider the line going through that point, it will intersect the curve at a third rational point. This is suggestive of a group operation, where given two rational points on the curve, we can obtain a third rational point on the curve. We define some notation and state some theorems that will be useful when working with elliptic curves.

Definition 12.1. Let $f \in \mathbb{F}[x, y]$ be a polynomial in x, y over some field $\mathbb{F}$. Then, we write $E/\mathbb{F}$ to denote the curve $f(x, y) = 0$. We will typically just write E as shorthand to refer to the curve. For any field $\mathbb{K}$ (possibly equal to $\mathbb{F}$), we define $E(\mathbb{K}) = \{(x, y) \in \mathbb{K} \mid f(x, y) = 0\}$ to be the set of points on the curve E whose coefficients are in $\mathbb{F}$.

Definition 12.2. Let $\mathbb{G}$ be an abelian group. Then, $g \in \mathbb{G}$ is a torsion point if g has finite order in $\mathbb{G}$. The set of all torsion points $\mathbb{T}$ in $\mathbb{G}$ form the torsion subgroup of $\mathbb{G}$.

Theorem 12.3 (Poincaré)**.** *Let E be an elliptic curve. Then, $E(\mathbb{Q})$ is an abelian group. We specify more precisely what the group operation is in the following section.*

Theorem 12.4 (Mordell-Weil)**.** *Let E be an elliptic curve. Then, $E(\mathbb{Q}) \cong \mathbb{T} \times \mathbb{Z}^r$ where $\mathbb{T}$ is the finite torsion subgroup of $E(\mathbb{Q})$ and r is an integer (known as the rank of the elliptic curve). In particular, $E(\mathbb{Q})$ is a finitely generated abelian group. Thus, there exists a finite set of points $P_1, \ldots, P_n \in E(\mathbb{Q})$ such that for all points $P \in E(\mathbb{Q})$ on the elliptic curve, there exist $a_1, \ldots, a_n \in \mathbb{Z}$ such that $P = \sum_{i=1}^n a_i P_i$.*

12.2.2. *Elliptic Curves: an Analytic Perspective.* Next, we revisit the development of elliptic curves from the analytic perspective. Specifically, analysts in the 19th century were interested in studying the arc length of various curves. In the case of an ellipse, the arc length ℓ can be computed by taking an integral of the following form:

$$f(x) = \int_0^x \frac{dt}{\sqrt{t^3 + at + b}}.$$

Such integrals fall into a class of integrals called elliptic integrals. Euler developed an addition formula for evaluating these integrals. Specifically, given $f(x_0)$ and $f(x_1)$, Euler developed an addition formula for computing $f(x_0 + x_1)$. It turns out that this operation precisely corresponds to the group operation over an elliptic curve. Before continuing, we recall a simpler integral from calculus where the denominator contained the square root of a quadratic rather than a cubic. In this case, we have

$$\int_0^x \frac{dt}{\sqrt{t^2 - 1}} = \arcsin x,$$

which is the inverse of a periodic function. Abel and Jacob noticed that an analogous result held for $f(x)$. Specifically, if we work over the complex numbers the inverse function $f^{-1} : \mathbb{C} \to \mathbb{C}$ is a doubly-periodic function. In particular, there exist $\omega_1, \omega_2 \in \mathbb{C}$ where ω_1, ω_2 are linearly independent when viewed as vectors over $\mathbb{R}^2$, such that for all $x \in \mathbb{C}$,

$$f^{-1}(x + \omega_1) = f^{-1}(x) = f^{-1}(x + \omega_2).$$

An equivalent way of stating this condition is to first define the lattice spanned by ω_1, ω_2, that is, the lattice Λ where

$$\Lambda = \{\alpha_1\omega_1 + \alpha_2\omega_2 \mid \alpha_1, \alpha_2 \in \mathbb{Z}\} = \omega_1\mathbb{Z} + \omega_2\mathbb{Z}.$$

It suffices to view f^{-1} as a function from $\mathbb{C}/\Lambda$ to $\mathbb{C}$, that is for all $x \in \mathbb{C}$ and $\omega \in \Lambda$, $f^{-1}(x) = f^{-1}(x + \omega)$. The fundamental domain F of the lattice Λ is the set spanned by ω_1 and ω_2, that is,

$$F = \{\alpha_1\omega_1 + \alpha_2\omega_2 \mid 0 \leq \alpha_1, \alpha_2 \leq 1\}.$$

If we visualize this in the plane, the fundamental domain forms a parallelogram, with ω_1 and ω_2 defining two of the sides of the parallelogram. However, this picture is a bit misleading since the boundaries of the parallelogram in the plane are actually equivalent in $\mathbb{C}/\Lambda$. Thus, $\mathbb{C}/\Lambda$ is a torus. Specifically, if we take the parallelogram representing the fundamental region and fold opposing edges together, the resulting shape we obtain is a torus. For this reason, elliptic curves are regraded as genus-1 curves.

Later, in the 1800s, Weierstress discovered a class of functions ($\wp$-functions) $\wp : \mathbb{C} \to \mathbb{C}$ such that all infinitely differentiable (more precisely, the function is meromorphic), doubly periodic functions may be expressed as a rational function in $\wp$ and $\wp'$. Moreover, it turns out that these $\wp$ function can be used to construct elliptic curves over $\mathbb{C}$. The $\wp$ functions are doubly-periodic functions parameterized by a lattice Λ over $\mathbb{C}$:

$$\wp(z;\Lambda) = \frac{1}{z^2} + \sum_{\substack{\omega\in\Lambda \\ \omega\neq 0}} \left(\frac{1}{(z-\omega)^2} - \frac{1}{\omega^2} \right).$$

Note that all $\wp$-functions are parameterized by the choice of lattice Λ; for notational convenience, we will write $\wp(z)$ rather than $\wp(z;\Lambda)$. Moreover, the $\wp$ functions satisfy the following differential equation:

$$[\wp'(z)]^2 = 4\wp(z)^3 - g_2\wp(z) - g_3$$

with $g_2, g_3 \in \mathbb{C}$. The key theorem that we will state is an isomorphism between the complex numbers $\mathbb{C}$ and a family of elliptic curves over $\mathbb{C}$.

Theorem 12.5. *Let Λ be a lattice over $\mathbb{C}$ and let $E/\mathbb{C}$ be the elliptic curve given by $y^2 = 4x^3 - g_2x - g_3$. Define the mapping $\Phi : \mathbb{C}/\Lambda \to E(\mathbb{C})$ where for all $z \neq 0$,*

$$\Phi(z) = (\wp(z), \wp'(z))$$

and $\Phi(0) = \mathcal{O}$. Here, $\mathcal{O}$ is a special point (referred to as the point at infinity) that is the identity element of $E(\mathbb{C})$. The function Φ defines an isomorphism between $\mathbb{C}$ and $E/\mathbb{C}$.

We may view Theorem 12.5 as defining an embedding of the torus on an elliptic curve over the complex values (that is, in the complex projective plane). Moreover, Theorem 12.5 gives us a precise way of defining the group operation over $E/\mathbb{C}$, namely the chord-and-tangent method.

12.2.3. *Chord-and-Tangent Method .* We describe the chord-and-tangent method for "adding" points in $E(\mathbb{C})$. First, we have

$$E(\mathbb{C}) = \left\{(x,y) \in \mathbb{C} \mid y^2 = 4x^3 - g_2x - g_3\right\} \cup \{\mathcal{O}\}.$$

The point $\mathcal{O}$ is a special point at infinity. Given two points $P_1, P_2 \in E(\mathbb{C})$, the addition operation proceeds as follows. For now, consider $P_1 \neq \mathcal{O} \neq P_2$.

- If $P_1 \neq P_2$, take the line that passes through P_1 and P_2. Let P_3 be the third point where the line intersects the curve. The sum of P_1 and P_2 is the reflection of P_3 about the x-axis.
- If $P_1 = P_2$, take the tangent line through P_1. Letting P_3 denote the third point where the line intersects the curve, the sum is the reflection of P_3 about the x-axis.
- If $P_1 = -P_2$, the line passing through P and $-P$ will be vertical. The vertical line will intersect the curve at a third point, namely the point at infinity $\mathcal{O}$.

From the isomorphism between $\mathbb{C}$ and $E(\mathbb{C})$, we have that $\mathcal{O}$ is the identity element in $E(\mathbb{C})$. Thus, for all $P \in E(\mathbb{C})$, $P + \mathcal{O} = P = \mathcal{O} + P$. A graphical view of the addition operation over $E(\mathbb{C})$ is shown in Figure 12.1.

We note that in $\mathbb{C}/\Lambda$, there are exactly three points of order 2: $\left\{\frac{1}{2}(\omega_1, 0), \frac{1}{2}(0, \omega_2), \frac{1}{2}(\omega_1, \omega_2)\right\}$ where $\Lambda = \omega_1\mathbb{Z} + \omega_2\mathbb{Z}$. We note that these three points precisely correspond to the three points along the x-axis in $E(\mathbb{C})$. Refer to Figure 12.1 for a graphical representation of this fact. The chord-and-tangent method for adding points on the elliptic curve naturally lends itself to an algebraic definition for addition. For an elliptic curve written using the Weierstrass representation ($y^2 = x^3 + ax + b$), addition can be expressed as follows. Specifically, if we have two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ where $P_1 \neq P_2$, then we can compute $(x_3, y_3) = P_1 + P_2$ by first setting

$$s = \frac{y_2 - y_1}{x_2 - x_1},$$

and then computing

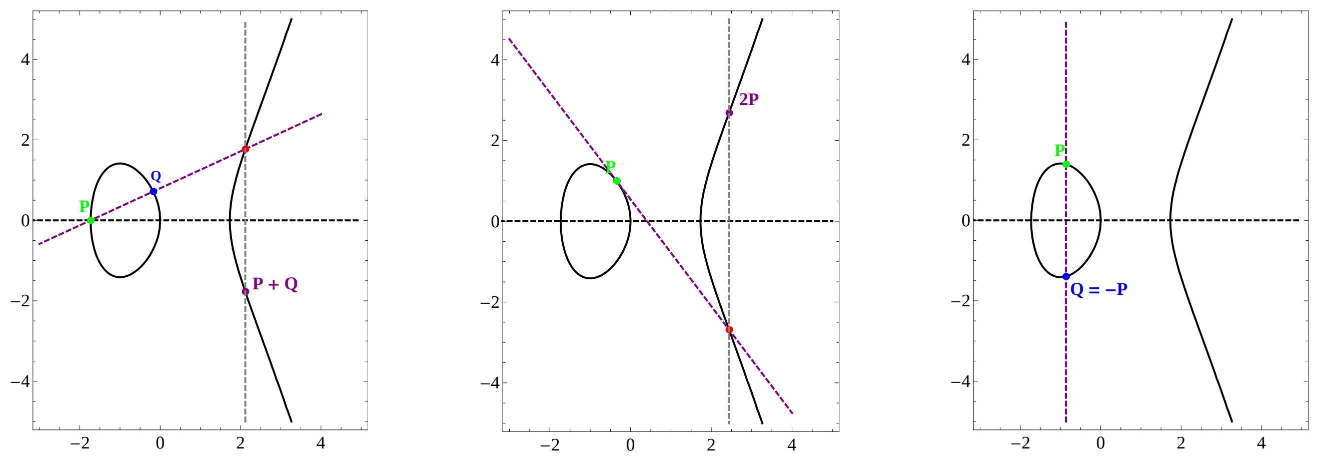
$$x_3 = s^2 - x_1 - x_2 \qquad y_3 = y_1 + s(x_3 - x_1).$$

When $P_1 = (x, y) = P_2$, then to compute $(x', y') = P_1 + P_2$, we write

$$s = \frac{3x^2 + a}{2y},$$

and then compute

$$x' = s^2 - 2x \qquad y' = s(x - x') - y.$$



(a) adding two distinct points (b) adding a point to itself (c) adding a point to its inverse

Figure 12.1. Illustration of chord-and-tangent method on the elliptic curve $y^2 = x^3 - 3x$

13. 5/15: Elliptic Curves and Pairings

13.1. **Elliptic Curves over Finite Fields.** In the previous lecture, we described elliptic curves over the rationals and the complex numbers. Here, we will apply our results to define elliptic curves over finite fields. Very often in these scenarios, we will work over extension fields $\mathbb{F}_{q^k}$ (recall that one way to construct $\mathbb{F}_{q^k}$ is to consider the quotient of the ring of polynomials $\mathbb{F}_q[x]$ with an irreducible polynomial $f \in \mathbb{F}_q[x]$ of degree k). We can now define elliptic curves over finite fields $\mathbb{F}_{q^k}$:

$$E_{a,b}\left(\mathbb{F}_{q^k}\right) = \left\{(x, y) \in \mathbb{F}_{q^k} : y^2 = x^3 + ax + b\right\} \cup \{\mathcal{O}\}.$$

It turns out that $E(\mathbb{F}_{q^k})$ is a finite abelian group, where the group operation is the chord-and-tangent operation described in Section 12.2.3.

Theorem 13.1 (Hasse-Weil). *For all $a, b \in \mathbb{F}_q$,*

$$|E_{a,b}(\mathbb{F}_q)| = q + 1 - t,$$

where $|t| < 2\sqrt{q}$. The quantity t is the trace (of Frobenius) of the elliptic curve.

Example 13.2. Let $E(\mathbb{F}_q)$ be the points on the elliptic curve $y^2 = x^3+1$ where $q = 2 \pmod 3$. Then $|E(\mathbb{F}_q)| = q+1$, so E has trace 0

Proof. The multiplicative group $\mathbb{F}_q^*$ has order $q - 1$, and since $q = 2 \pmod 3$, $\gcd(3, q - 1) = 1$. Then, x^3 is a permutation over $\mathbb{F}_q^*$, and since $0^3 = 0$, we conclude that $x^3 + 1$ is a permutation over $\mathbb{F}_q$. Thus,

$$E(\mathbb{F}_q) = \left\{(x, y) \in \mathbb{F}_q \mid y^2 = x^3 + 1\right\} \cup \{\mathcal{O}\} \cong \left\{(x, y) \in \mathbb{F}_q \mid y^2 = x\right\} \cup \{\mathcal{O}\}.$$

Next, we note that every quadratic residue $x \in \mathbb{F}_q^*$ has exactly two square roots. Thus, each quadratic residue in $\mathbb{F}_q^*$ corresponds to two points in $E(\mathbb{F}_q)$, one for each square root. The only remaining element in $\mathbb{F}_q$ (that is not in $\mathbb{F}_q^*$) is 0, which is also a quadratic residue (although with a single root). There are $\frac{q-1}{2}$ quadratic residues in $\mathbb{F}_q^*$, which correspond to $q-1$ points in $E(\mathbb{F}_q)$. The identity element 0 contributes one additional element, as does the point at infinity $\mathcal{O}$. Thus, $|E(\mathbb{F}_q)| = q - 1 + 2 = q + 1$. Thus $E(\mathbb{F}_q)$ has trace 0. $\square$

We remark that if we know $p = |E(\mathbb{F}_q)|$, then we can compute $\left|E\left(\mathbb{F}_{q^k}\right)\right|$ as a function in p, q, k. Next, consider the case where $p = |E(\mathbb{F}_q)|$ is prime, and moreover, that $p \nmid q(q - 1)$. Let $\alpha > 0$ be the smallest integer such that $p \mid q^\alpha - 1$. Then, $\mathbb{Z}_p^+ \times \mathbb{Z}_p^+ \subseteq E(\mathbb{F}_{q^\alpha})$ where $\mathbb{Z}_p^+$ denotes the additive group $\mathbb{Z}/p\mathbb{Z}$. Thus, over $E(\mathbb{F}_{q^\alpha})$, the points of order p on the elliptic curve form a lattice structure (similar to the case with the complex numbers).

Finally we note that there are alternative representations of elliptic curves that allow for better computational efficiency. The representation we have used in our exposition thus far is the Weierstrass representation.

Definition 13.3 (Weierstrass Curve)**.** An elliptic curve over a field $\mathbb{F}$ is described by an equation of the form $y^2 = x^3 + ax + b$ where $a, b \in \mathbb{F}$. The group law for a Weierstrass curve is specified by the chord-and-tangent method from Section 12.2.3.

We note that in a Weierstrass curve, the group operation on points P_1 and P_2 are different depending on whether $P_1 = P_2$ and whether $P_1 = \mathcal{O}$ or $P_2 = \mathcal{O}$. Using the Edwards representation [Edw07] of elliptic curves, we are able to express the group law explicitly.

Definition 13.4 (Edwards Curve)**.** An elliptic curve over a field $\mathbb{F}$ (that does not have characteristic 2) is described by an equation $x^2 + y^2 = c^2(1 + dx^2y^2)$ where $c, d \in \mathbb{F}$ and $cd\left(1 - c^4 d\right) \neq 0$ The identity element is represented as $(0, c)$. Two points on the curve (x_1, y_1) and (x_2, y_2) can be combined as

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1y_2 + x_2y_1}{c\,(1 + dx_1x_2y_1y_2)}, \frac{y_1y_2 - x_1x_2}{c\,(1 - dx_1x_2y_1y_2)}\right).$$

Finally, we note that every Edwards curve is birationally equivalent to a Weierstrass curve. Thus, in almost all cases, we can represent an elliptic curve using the Edwards representation. Because we have an explicit formula for the group law, the Edwards representation is often easier to work with in practice.

13.2. **Discrete Log in Elliptic Curve Groups.** The natural cryptographic question then concerns the hardness of discrete log in elliptic curve groups. For random $g, h \in E(\mathbb{F}_q)$, the best known discrete log algorithm runs in time roughly $O(\sqrt{q}) = O\left(e^{\frac{1}{2}\log q}\right)$. Thus, to achieve 128-bit security, we require that $q > 2^{256}$. Recall that in the case where we work over $\mathbb{F}_q^*$, the best algorithm runs in time $O\left(e^{\sqrt[3]{\log q}}\right)$. For 128 bit security, we require $q > 2^{2048}$. Thus, from a practical perspective, elliptic curves are much more efficient than working over the integers.

We illustrate the Diffie-Hellman protocol over elliptic curves. Let $E(\mathbb{F}_{q^\alpha})$ be an elliptic curve, and let $P \in E$ be an element of prime order p. The Diffie-Hellman key exchange protocol works as follows:

Diffie-Hellman Key Exchange over Elliptic Curve Groups

(1) Alice chooses $a \xleftarrow{\$} \{1, \ldots, p\}$ and computes $A \leftarrow aP$ via repeated doubling. Alice sends A to Bob.
(2) Bob chooses $b \xleftarrow{\$} \{1, \ldots, p\}$ and computes $B \leftarrow bP$ via repeated doubling. Bob sends B to Alice.
(3) Alice and Bob then apply a key derivation formula to $(ab) \cdot P$ to derive their shared session key.

We note that there are weak elliptic curves where the discrete log problem is easy. For instance, the discrete log problem over curves $E/\mathbb{F}_q$ where $|E(\mathbb{F}_q)| = q$, that is, curves with trace $t = 1$, is easy, and can be computed in polynomial time (requiring a linear number of group operations) [Sma99].

13.3. **Pairings in Elliptic Curve Groups.** It turns out that elliptic curves can have additional structure that enable many applications. Let $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T$ be finite groups of prime order p. Then, a pairing $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$ is a function that satisfies the following properties. Note that this is a generalization of Definition 6.7 to asymmetric groups.

- e is efficiently computable.
- **Non-degeneracy**: there exist $g_0 \in \mathbb{G}_0$ and $g_1 \in \mathbb{G}_1$ such that $e(g_0, g_1) \neq 1$.
- **Bilinearity**: for all $g_0, g_1 \in \mathbb{G}_0$, and $a, b \in \mathbb{Z}_p$, $e(g_0^a, g_1^b) = e(g_0, g_1)^{ab}$.

One application of pairings that we have already encountered is in the construction of short signatures (BLS signatures) in Section 6.2.2.

Example 13.5. Let $p = |E(\mathbb{F}_q)|$ where $p \mid q^\alpha - 1$. We can take $\alpha = 6$. As noted previously, $\mathbb{Z}_p \times \mathbb{Z}_p \subseteq E(\mathbb{F}_{q^\alpha})$. Denote the isomorphic copy of $\mathbb{Z}_p \times \mathbb{Z}_p$ by $\mathbb{G}$. Since $p \mid q^\alpha - 1$, then there exists a root of unity $\omega \in \mathbb{F}_{q^\alpha}$ such that $\omega^p = 1$ where $\omega \neq 1$. Then, let $G_T = \langle \omega \rangle$. Then, the Weil pairing is a function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The pairing can be evaluated in time $O\left(\log^3 p\right)$ via Miller's algorithm.

Before describing the operation of the pairing, we first describe an alternate viewpoint. Let $P_1, P_2 \in \mathbb{G}$ be a basis for $\mathbb{G}$. Then, given $Q_1, Q_2 \in \mathbb{G}$, there exist $a, b, c, d \in \mathbb{F}_{q^\alpha}$ such that

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \end{pmatrix} = \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}.$$

This immediately specifies the definition of the pairing:

$$e(Q_1, Q_2) = \omega^\alpha \text{ where } \alpha = \det \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

The Miller algorithm provides an efficient way of evaluating $e(Q_1, Q_2)$ without knowledge of a, b, c, d. We note that computing the coefficients a, b, c, d is at least as hard as computing the discrete log in $E(\mathbb{F}_{q^\alpha})$.

14. 5/20: Pairings in Elliptic Curve Groups

14.1. **Divisors in an Elliptic Curve Group.**

In this lecture, we describe Miller's algorithm for evaluating the Weil pairing over elliptic curve groups. Before describing the algorithm, we need to describe the structure of the elliptic curve group and introduce the concept of divisors. First, we note that all n where $q \nmid n$, there is an isomorphic copy of $\mathbb{Z}_n \times \mathbb{Z}_n$ in $E(\mathbb{F}_{q^\alpha})$ for some sufficiently large α.

Definition 14.1. Let $E(\mathbb{F})$ be an elliptic curve and let n be a positive integer. We write $E[n]$ to denote the points of order dividing n in E.

Theorem 14.2. *Let $E(\mathbb{F}_q)$ be an elliptic curve. Then, for all n such that $q \nmid n$, there exists $\alpha \in \mathbb{Z}$ such that*

$$E(\mathbb{F}_{q^\alpha})[n] \cong \mathbb{Z}_n \times \mathbb{Z}_n.$$

Let $E(\mathbb{F}_q)$ be an elliptic curve, and take n such that $q \nmid n$. By Theorem 14.2, there exists α such that $E(\mathbb{F}_{q^\alpha})$ contains an isomorphic copy $\mathbb{G}$ of $\mathbb{Z}_n \times \mathbb{Z}_n$. The Weil pairing is a function $e : \mathbb{G} \times \mathbb{G} \to \langle \omega \rangle$ where $\omega \in \mathbb{F}_{q^\alpha}$ is an element of order n. In words, the Weil pairing takes as input two elements of $E(\mathbb{F}_{q^\alpha})$ with order n and outputs an n^{th} root of unity in $\mathbb{F}_{q^\alpha}$. To make this more explicit, we will need some definitions.

Definition 14.3. Let E be an elliptic curve. A divisor $\mathcal{A}$ is a formal sum of finitely many points $S \subseteq E$ ($|S| < \infty$):

$$\mathcal{A} = \sum_{P \in S} a_P(P),$$

where $a_p \in \mathbb{Z}$. The degree of a divisor is the sum of its coefficients:

$$\deg(\mathcal{A}) = \deg\left(\sum_{P \in S} a_P(P)\right) = \sum_{P \in S} a_P.$$

The sum of a divisor is given by

$$\text{sum}(\mathcal{A}) = \text{sum}\left(\sum_{P \in S} a_P(P)\right) = \sum_{P \in S} a_P P.$$

In our presentation, we restrict our attention to divisors of degree 0. Next, we define ratioanl functions on an elliptic curve group. A rational function f over $E(\mathbb{F})$ is defined as the ratio of two polynomials $f_1, f_2 \in \mathbb{F}[x, y]$:

$$f(x, y) = \frac{f_1(x, y)}{f_2(x, y)}.$$

We require that f be defined for at least one point in $E(\mathbb{F})$. This means that $\left(y^2 - x^3 - ax - b\right)^{-1}$ is not a function in $E(\mathbb{F})$. The zeroes of f are the points $(x, y) \in E(\mathbb{F})$ where $f(x, y) = 0$; the poles of f are the points $(x, y) \in E(\mathbb{F})$ where $f(x, y) = \infty$. Two rational functions f_1, f_2 are equivalent if $\left(y^2 - x^3 - ax - b\right) \mid f_1 - f_2$. An equivalent definition is that f_1 and f_2 agree on all points in $E(\mathbb{F})$, that is, for all $P \in E(\mathbb{F})$, $f_1(P) = f_2(P)$.

Definition 14.4. The divisor (f) of a rational function f on $E(\mathbb{F})$ is given by the formal sum

$$(f) = \sum_{P \in E(\mathbb{F}_q)} \text{ord}_P(f) \cdot (P),$$

where $\text{ord}_P(f)$ denotes the order of the zero or pole P of f. Note that the order of a zero is always positive and the order of a pole is always negative.

Consider the function $f(x, y) = ax + by + c$, which defines a line. Then, the divisor of f is given by the formal sum

$$(f) = (P_1) + (P_2) + (P_3) - 3(\mathcal{O}),$$

where P_1, P_2, P_3 are the roots of f.

Definition 14.5. A divisor $\mathcal{A}$ is principal if there exists a function f such that $(f) = \mathcal{A}$. We say that two divisors $\mathcal{A}$ and $\mathcal{B}$ are equivalent if their difference $\mathcal{A} - \mathcal{B}$ is principal.

Fact 14.6. *A divisor $\mathcal{A} = \sum_p a_p \cdot (p)$ is principal if and only if $\deg(\mathcal{A}) = 0$ and $\operatorname{sum}(\mathcal{A}) = \mathcal{O}$.*

Fact 14.7. *The divisor of a function (f) uniquely define f (up to constant multiplicative factors).*

Example 14.8. Consider again the function $f(x, y) = ax + by + c$. As before, $(f) = (P_1) + (P_2) + (P_3) - 3\,(\mathcal{O})$, where P_1, P_2, P_3 are the roots of f. Then, (f) is a principal divisor. Additionally, we note that the zeroes and poles uniquely define a function. Note that 0 and ∞ are annihilators. If we take the function $f(x) = \frac{(x-1)(x-2)}{(x-3)}$, the formal sum would take the form $(1) + (2) - (3) - (\infty)$.

Theorem 14.9 (Riemann-Roch)**.** *Any divisor $\mathcal{A} = \sum_P a_P(P)$ is equivalent to some divisor of the form $(P) - (\mathcal{O})$.*

We note that the Riemann-Roch theorem uniquely defines the group operation over the curve. Specifically, given two points $(P_1) - (\mathcal{O})$ and $(P_2) - (\mathcal{O})$, we can combine them to obtain

$$(P_1) + (P_2) - 2\,(\mathcal{O}),$$

and from the Riemann-Roch theorem, there exists a point P_3 such that

$$(P_3) - (\mathcal{O}) = (P_1) + (P_2) - 2\,(\mathcal{O}).$$

We take P_3 to be the sum of P_1 and P_2. For a function f and a divisor $\mathcal{A} = \sum_P a_P \cdot (P)$, we define

$$f(\mathcal{A}) = \prod_P f\,(P)^{a_P}.$$

Example 14.10. Consider the divisor $\mathcal{A} = (4, 5) - (0, 2)$. Evaluating the above function $f(x, y) = \frac{(x-1)(x-2)}{x-3}$ at $\mathcal{A}$ yields

$$f(\mathcal{A}) = \left[\frac{(4-1)(4-2)}{(4-3)}\right]\left[\frac{(0-1)(0-2)}{(0-3)}\right]^{-1} = -\frac{6 \cdot 3}{2} = -9.$$

Finally, note that the value of the divisor is independent of constant multiplicative factors: $f(\mathcal{A}) = (cf)\,(\mathcal{A})$.

14.2. **Pairings and Miller's Algorithm.** With this preparation, we can define the Weil pairing. Take two points $P, Q \in E[n]$ that are of order n. Then $\mathcal{A}_P$ is equal to some divisor equivalent to $(P) - (\mathcal{O})$ and so $n \cdot \mathcal{A}_P = n\,(P) - n\,(\mathcal{O})$ is a principal divisor. Thus, there exists a function f_P such that $(f_P) = n \cdot \mathcal{A}_P$. We can define the function f_Q in an analogous fashion. Then, the Weil pairing is given by

$$e(P, Q) = \frac{f_P(\mathcal{A}_Q)}{f_Q(\mathcal{A}_P)}.$$

A more efficient pairing is the Tate-Lichtenbaum (commonly referred to as the Tate pairing). Take $n \in \mathbb{Z}$ where $n = -1 \pmod q$. As before, let ω be a primitive n^{th} root of unity in $\mathbb{F}_{q^\alpha}^*$. Then, the Tate pairing $\tau : E\left(\mathbb{F}_{q^\alpha}\right)[n] \times E\left(\mathbb{F}_{q^\alpha}\right)/nE\left(\mathbb{F}_{q^\alpha}\right) \to \langle\omega\rangle$ is given as

$$\tau(P, Q) = \left[f_P\left(\mathcal{A}_Q\right)\right]^{\frac{q^\alpha - 1}{n}},$$

We claim that this pairing is non-degenerate and bilinear (although the proof is nontrivial). With this preparation, we are now ready to describe Miller's algorithm for evaluating the pairing.

Miller's Algorithm

(1) Given P, Q, choose random $R \in E[n]$. Let $\mathcal{A}_Q = (Q + R) - (R)$. Note that $\mathcal{A}_Q$ is equivalent to the divisor $(Q) - (\mathcal{O})$. Then,

$$e(P, Q) = [f_P(\mathcal{A}_Q)]^{\frac{q^\alpha - 1}{n}} = \left[\frac{f_P(Q + R)}{f_P(R)}\right]^{\frac{q^\alpha - 1}{n}}.$$

We describe an efficient algorithm for computing $f_P(\cdot)$. Observe that this is sufficient for evaluating $\frac{f_P(Q+R)}{f_P(R)}$. We also note that oftentimes in practice, we only need to evaluate $f_P(Q + R)$ and not $f_P(R)$ because $f_P(R)$ will evaluate to the identity element.

(2) For $b > 0$, define the divisor $\mathcal{A}_b = b\,(P+R) - b(R) - (bP) + (\mathcal{O})$. Note that $\mathcal{A}_b$ is principal for all $b > 0$, so there exists a rational function f_b such that $(f_b) = \mathcal{A}_b$. Moreover, since $nP = \mathcal{O}$, we have that,

$$f_n(\mathcal{A}_Q) = f_P(\mathcal{A}_Q).$$

(3) Using repeated doubling (as described in Lemma 14.11), we can evaluate $f_n(\mathcal{A}_Q)$ in $\log n$ steps. Moreover, we note that if the point P is fixed a priori, we can precompute the lines $g_1, g_1', g_2, g_2', \ldots, g_{\log n}, g'_{\log n}$. With this optimization, evaluating the pairing becomes as efficient as exponentiation.

Lemma 14.11. *For all $b, c > 0$, there exists an efficient algorithm that on input $f_b(\mathcal{A}_Q)$, $f_c(\mathcal{A}_q)$ computes $f_{b+c}(\mathcal{A}_q)$.*

Proof. Consider the line g_1 going through bP and cP. Let $-(b+c)P$ be the third point of intersection on the elliptic curve. Let g_1' be the vertical line through $-(b+c)P$. Then,

$$(g_1) = (bP) + (cP) + (-(b+c)P) - 3\,(\mathcal{O})$$
$$(g_1') = ((b+c)P) + (-(b+c)P) - 2\,(\mathcal{O})\,.$$

Then, $\mathcal{A}_{b+c} = \mathcal{A}_b + \mathcal{A}_c + (g_1) - (g_1')$ and so

$$f_{b+c}\,(\mathcal{A}_Q) = f_b(\mathcal{A}_Q) \cdot f_c\,(\mathcal{A}_Q) \cdot \frac{g_1(\mathcal{A}_Q)}{g_1'\,(\mathcal{A}_Q)}.$$

□

Note that in order to apply Miller's algorithm, it is necessary to know the order of the group. Thus, if n is taken to be an RSA modulus, the pairing cannot be evaluated efficiently (assuming hardness of factoring). We can therefore view elliptic curve groups where the order is an RSA modulus as groups with a trapdoor DDH oracle.

14.3. Identity-Based Encryption (IBE).

One of the problems where pairings proved useful is in the construction of the first identity-based encryption (IBE) schemes [BF01]. IBE generalizes public-key encryption to the scenario where a user's public key can be arbitrary (for example, a name or an email address). To decrypt a message for a particular identity, we require an identity secret key derived from a master secret key for the IBE scheme. Specifically, an IBE scheme consists of the following algorithms:

- $\mathsf{Setup}(\lambda)$: Generates public parameters pp a master secret key msk.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$: Uses the master secret key to derive a identity secret key $\mathsf{sk}_{\mathsf{id}}$ for the identity id.
- $\mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m)$: Encrypts a message m to identity id (public key algorithm).
- $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, c)$: Decrypts a ciphertext c using the identity secret key $\mathsf{sk}_{\mathsf{id}}$ for identity id.

The security notion for IBE will be the usual notion of semantic security, but restricted to a particular identity $\mathsf{id}^\star$. However, we also allow the adversary to request secret keys for identities id of its choosing (provided that $\mathsf{id} \neq \mathsf{id}^\star$). We define the IBE security game with respect to a world bit $b \in \{0,1\}$.

IBE Security Game (Adaptive/Full Security)

(1) The challenger generates the public parameters and the master secret key for the IBE scheme by performing $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(\lambda)$. It sends pp to the adversary.
(2) The adversary can make adaptive identity queries $\mathsf{id}_1, \ldots, \mathsf{id}_k$ to obtain secret keys $\mathsf{sk}_{\mathsf{id}_1}, \ldots, \mathsf{sk}_{\mathsf{id}_k}$.
(3) The adversary chooses an identity $\mathsf{id}^\star$ ($\mathsf{id}^\star \notin \{\mathsf{id}_1, \ldots, \mathsf{id}_k\}$) and two messages m_0, m_1 ($|m_0| = |m_1|$).
(4) The challenger replies with $\mathsf{Encrypt}\,(\mathsf{pp}, \mathsf{id}^\star, m_b)$.
(5) The adversary can continue to make identity queries for identities $\mathsf{id} \neq \mathsf{id}^\star$. At the end of the game, the adversary outputs $b' \in \{0,1\}$.

We say an IBE scheme is secure if the distinguishing advantage of any PPT adversary is negligible in the security parameter λ. We note that an IBE scheme effectively compresses exponentially many public keys into the public parameters.

15. 5/22: Identity-Based Encryption

15.1. **Identity-Based Encryption.** We continue our discussion of identity-based encryption (IBE) from the previous lecture. Before presenting a construction of IBE, we first note that trapdoor functions and general public key primitives are insufficient for IBE. We begin by noting that the ElGamal encryption scheme is not an IBE scheme. Recall that in ElGamal encryption, the plaintext and ciphertext spaces is a group $\mathbb{G}$ with generator g. The public key consists of a group element h and the secret key is the exponent α such that $g^\alpha = h$. We note that while we can let the public component h be arbitrary, we are unable to compute the corresponding secret key in this case (requires computing the discrete log of h). Thus, ElGamal encryption is not an IBE system. Currently, we know of three families of constructions for IBE: from pairings, from lattices, and from quadratic residuosity. In this lecture, we will show some example of IBE constructions from pairings.

15.1.1. *Boneh-Franklin* IBE*.* In this section, we described the first IBE construction of IBE due to Boneh and Franklin [BF01]. The construction leverages pairings and can be proven secure assuming the Bilinear Diffie-Hellman (BDH) assumption in the random oracle model. Let $\mathbb{G}, \mathbb{G}_T$ be two groups with prime order p that admits an efficiently-computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Let g be a generator of $\mathbb{G}$, let $\mathcal{I}$ be the space of identities for the IBE scheme, and let $\{0,1\}^n$ be the space of messages for the IBE scheme. We describe the IBE scheme below:

Boneh-Franklin IBE Scheme

- $\mathsf{Setup}(\lambda)$: Choose $\alpha \xleftarrow{\$} \mathbb{Z}_p$, and compute $y \leftarrow g^\alpha$. Let $H_1 : \mathcal{I} \to \mathbb{G}$ be a hash function from the identity space to $\mathbb{G}$ and let $H_2 : \mathbb{G}_T \to \{0,1\}^n$ be a hash function from $\mathbb{G}_T$ to the message space. Output the public parameters $\mathsf{pp} \leftarrow (y, H_1, H_2)$ and the master secret key $\mathsf{msk} \leftarrow \alpha$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$: The secret key for an identity $\mathsf{id} \in \mathcal{I}$ is $\mathsf{sk}_{\mathsf{id}} \leftarrow H_1(\mathsf{id})^\alpha$. Recall that $\mathsf{msk} = \alpha$.
- $\mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m)$: As defined above, the public parameters pp are (y, H_1, H_2). To encrypt a message $m \in \{0,1\}^n$ to identity $\mathsf{id} \in \mathcal{I}$, first choose $s \xleftarrow{\$} \mathbb{Z}_p$ and output the ciphertext
$$c \leftarrow (g^s, m \oplus H_2 (e (y, H_1(\mathsf{id}))^s)).$$
- $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, c)$: To decrypt a ciphertext $c = (c_1, c_2)$ encrypted to identity id using $\mathsf{sk}_{\mathsf{id}}$, compute
$$m \leftarrow c_2 \oplus H_2 (e(c_1, \mathsf{sk}_{\mathsf{id}})).$$

First, we demonstrate correctness. Let $c = (c_1, c_2)$ be an encryption of message m for identity id. Then, the decryption algorithm computes

$$\begin{aligned} c_2 \oplus H_2 (e(c_1, \mathsf{sk}_{\mathsf{id}})) &= m \oplus H_2 (e (y, H_1(\mathsf{id}))^s) \oplus H_2 (e(g^s, H_1(\mathsf{id})^\alpha)) \\ &= m \oplus H_2 (e(g, H_1(\mathsf{id}))^{\alpha s}) \oplus H_2 (e (g, H_1(\mathsf{id}))^{\alpha s}) \\ &= m, \end{aligned}$$

using the bilinearity of e. Security of the scheme follows from the Bilinear Diffie-Hellman (BDH) assumption in the random oracle model. We describe the BDH problem, and the corresponding assumption below:

Definition 15.1. Let $\mathbb{G}, \mathbb{G}_T$ be groups of prime order p equipped with an efficiently-computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Let g be a generator for $\mathbb{G}$. Choose $a, b, c \xleftarrow{\$} \mathbb{Z}_p$. The Bilinear Diffie-Hellman (BDH) problem is to compute $e(g, g)^{abc}$ given values g^a, g^b, g^c. The advantage of an algorithm $\mathcal{A}$ in solving the BDH problem is defined to be

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{BDH}} = \Pr\left[a, b, c \xleftarrow{\$} \mathbb{Z}_p : \mathcal{A}\left((g, g^a, g^b, g^c)\right) = e\left(g, g\right)^{abc}\right].$$

The BDH problem can be viewed as a modified version of the CDH problem. In CDH, the adversary is given g^a, g^b and must produce g^{ab}. Note that the assumption is false if we only require the adversary to produce $e(g, g)^{ab}$ given g^a, g^b, since it can do this by simply evaluating the pairing $e(g^a, g^b)$. We say that a tuple $(\mathbb{G}, \mathbb{G}_T, e)$ satisfies the BDH assumption if the advantage of all efficient algorithms $\mathcal{A}$ in solving the BDH problem in $(\mathbb{G}, \mathbb{G}_T, e)$ is negligible in the security parameter (the assumption is generally phrased with respect to a BDH parameter generation algorithm parametrized by a security parameter).

Theorem 15.2. *The Boneh-Franklin* IBE *scheme is fully secure under the* BDH *assumption in the random oracle model.*

To illustrate some of the techniques that go into the proof, we will demonstrate here a weaker notion of security known as selective security. In the selective security game, the adversary must declare which identity it wants to target before seeing the IBE public key and before it starts making identity queries. Specifically, the selective security game for IBE is described below:

IBE Security Game (Selective Security)

(1) The adversary chooses an identity $\mathsf{id}^\star$ and sends it to the challenger.
(2) The challenger generates the public parameters and master secret key by invoking $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(\lambda)$. It sends pp to the adversary.
(3) The adversary can make adaptive identity queries $\mathsf{id} \neq \mathsf{id}^\star$ to the challenger to obtain secret keys $\mathsf{sk}_{\mathsf{id}}$.
(4) The adversary chooses two messages m_0, m_1 $(|m_0| = |m_1|)$ and sends them to the challenger.
(5) The challenger replies with $\mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}^\star, m_b)$.
(6) The adversary can continue to make identity queries for identities $\mathsf{id} \neq \mathsf{id}^\star$ to obtain secret keys $\mathsf{sk}_{\mathsf{id}}$. At the end of the game, the adversary outputs $b' \in \{0, 1\}$.

Clearly, all adaptively-secure IBE schemes are also selectively-secure. We now give a proof that the Boneh-Franklin IBE scheme is selectively secure in the random oracle model.

Theorem 15.3. *The Boneh-Franklin* IBE *scheme is selectively secure under the* BDH *assumption in the random oracle model.*

Proof. Suppose we have an adversary $\mathcal{A}$ that wins the selectively-secure IBE game (where the IBE system is instantiated over groups $\mathbb{G}, \mathbb{G}_T$ of prime order p) with non-negligible advantage. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks the BDH assumption in $\mathbb{G}, \mathbb{G}_T$. Specifically, adversary $\mathcal{B}$ takes as input a BDH challenge $(g, g^a, g^b, g^c) = (g, x, y, z)$, for $a, b, c \xleftarrow{\$} \mathbb{Z}_p$, and its goal is to compute $e(g,g)^{abc}$. We model the hash functions H_1 and H_2 as random oracles. Algorithm $\mathcal{B}$ then proceeds as follows:

Algorithm $\mathcal{B}$ for BDH Problem

(1) Algorithm $\mathcal{B}$ runs $\mathcal{A}$ to receive the target identity $\mathsf{id}^\star$.
(2) Algorithm $\mathcal{B}$ sets the public parameters $\mathsf{pp} \leftarrow y$ and sends pp to $\mathcal{A}$.
(3) Algorithm $\mathcal{B}$ can now make key generation queries as well as queries to H_1 and H_2. As is standard in random oracle constructions, we will model H_1 and H_2 as lookup tables, whose values are programmed on the spot. The queries from $\mathcal{A}$ are simulated as follows:
    - **Key Generation Request**: If there is an entry (id, t) in the table for H_1, then reply with y^t. Otherwise, choose $t \xleftarrow{\$} \mathbb{Z}_p$, add (id, t) to the table for H_1 and reply with y^t.
    - **Random Oracle Query for** H_1: If $\mathsf{id} = \mathsf{id}^\star$, then reply with z. Otherwise, if there is already an entry (id, t) in the table, then reply with g^t. Otherwise, choose $t \xleftarrow{\$} \mathbb{Z}_p$, add (id, t) to the table and reply with g^t.
    - **Random Oracle Query for** H_2: On input x, if $\mathcal{A}$ has previously queried H_2 on x, then return the value stored in the table. Otherwise, choose $r \xleftarrow{\$} \{0,1\}^n$, add (x, r) to the table and return r.

(4) At some point, $\mathcal{A}$ will submit two messages m_0 and m_1. Algorithm $\mathcal{B}$ chooses $b \xleftarrow{\$} \{0, 1\}$ and computes the ciphertext
$$\mathsf{ct} \leftarrow (x, m_b \oplus H_2(e(g, g)))$$

□

15.1.2. *Gentry's* IBE. The Boneh-Franklin IBE scheme from the previous section achieves security in the random oracle model. In this section, we present an alternative construction due to Gentry [Gen06] that does not require random oracles. Again, let $\mathbb{G}, \mathbb{G}_T$ be groups of prime order p that admits an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

- $\mathsf{Setup}(\lambda)$: Choose $g, h \xleftarrow{\$} \mathbb{G}$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p$. Next, compute $y \leftarrow g^\alpha$. Output the public parameters $\mathsf{pp} \leftarrow (g, y, h)$ and the master secret key $\mathsf{msk} \leftarrow \alpha$.

- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$: Choose $r \xleftarrow{\$} \mathbb{Z}_p$ and output the identity secret key
$$\mathsf{sk}_{\mathsf{id}} \leftarrow \left(r, \left(hg^{-r}\right)^{1/(\alpha-\mathsf{id})}\right).$$
Note that $\mathsf{msk} = \alpha$ and g, h are included in the public parameters pp.
- $\mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m)$: Choose $s \xleftarrow{\$} \mathbb{Z}_p$ and compute
$$c \leftarrow \left(y^s g^{-s\cdot\mathsf{id}}, e(g,g)^s, m \cdot e(g,h)^{-s}\right).$$
- $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, c)$: Write c as (c_1, c_2, c_3) and $\mathsf{sk}_{\mathsf{id}}$ as (s_1, s_2). Then, compute
$$m \leftarrow e(c_1, s_2) \cdot c_2^{s_1} \cdot c_3$$

First, we show that the algorithm is correct. Suppose $c = (c_1, c_2, c_3)$ is an encryption of a message m for identity id. First, we note that
$$c_1 = y^s g^{-s\cdot\mathsf{id}} = g^{s(\alpha-\mathsf{id})}.$$
Then,
$$e(c_1, s_2) = e\left(g^{s(\alpha-\mathsf{id})}, \left(hg^{-r}\right)^{1/(\alpha-\mathsf{id})}\right) = e\left(g^s, hg^{-r}\right) = e\left(g,h\right)^s e(g,g)^{-rs}.$$
Finally, we have
$$e(c_1, s_2) \cdot c_2^{s_1} \cdot c_3 = e(g,h)^s e(g,g)^{-rs} \cdot e(g,g)^{rs} \cdot e(g,h)^{-s} \cdot m = m.$$

We briefly describe the proof idea for this scheme. The simulator knows one private key for every id and thus can response to all private key queries. Hardness in this case reduces to the $q-\mathsf{BBDH}$ assumption: given $g, v, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^q}$, distinguish $e(g,v)^{\alpha^{q+1}}$ from a uniform element in $\mathbb{G}_T$. Our simulator works as follows. For a q-query adversary, the simulator is given $g, v, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^q}$. In the steup phase, the simulator will chooses a random polynomial $f(x) = a_0 + a_1 x + \cdots + a_q x^q \in \mathbb{F}_q[x]$. We sent the adversary the public parameters $\mathsf{pp} \leftarrow \left(g, y = g^{\alpha}, h \leftarrow g^{f(\alpha)}\right)$. Note that since we have the powers of g, we can compute $g^{f(\alpha)}$. Next, we need a way to respond to identity queries. On input id, the simulator needs to compute $\left(r \leftarrow \mathbb{Z}_p, \left(hg^{-r}\right)^{1/(\alpha-\mathsf{id})}\right)$. To do so, we let $t \leftarrow f(\mathsf{id})$ and compute
$$\left(hg^{-r}\right)^{1/(\alpha-\mathsf{id})} = g^{[f(\alpha)-f(\mathsf{id})]/(\alpha-\mathsf{id})}$$
and observe that $\left[f(x) - f(\mathsf{id})\right]/(x - \mathsf{id}) \in \mathbb{F}_p[x]$. Using the powers of g, we can compute this. Now, we can construct the challenge ciphertext accordingly so that to distinguish between the two distributions.

15.2. **Digital Signatures from IBE.** We note that IBE implies a signature scheme. Specifically, signing corresponds to key extraction: $\sigma \leftarrow \mathsf{KeyGen}(\mathsf{msk}, m)$ and to verify a signature $\mathsf{Verify}(\mathsf{pp}, m, \sigma)$, we check that σ decryps messages encrypted using m.

15.3. **Hierarchical IBE (HIBE).** In IBE, there is a master secret key msk that is used to derive secret keys for identities. In an hierarchical IBE system, we allow keys derived from the master secret key (level 0) can also issue keys (level 1), and so on. Then, we can encrypt a message to an $\mathsf{id} = (\mathsf{id}_1, \mathsf{id}_{11}, \mathsf{id}_{111})$. Thus, we have a hierarchical structure. In BBG05, we have constructions of HIBE where the ciphertext size is independent of hierarchy depths. It is still an open problem to have this kind of construction from LWE. Generally, in HIBE, the security degrades in the depth.

15.4. **Forward Secure Encryption.** Consider the following problem. Suppose that for each day t, we have a secret key sk_t and corresponding public key pk_t, where on day t, the user uses keypair $(\mathsf{sk}_t, \mathsf{pk}_t)$. Our goal is to ensure that given sk_t, ciphertexts encrypted under $\mathsf{pk}_1, \ldots, \mathsf{pk}_{t-1}$ are still semantically secure. The trivial construction is to simply let $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_T)$ and $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_T)$. Unfortunately, in this case $|\mathsf{pk}| = O(T) = |\mathsf{sk}|$. We can do much better using HIBE. Specifically, we construct a binary tree where the root corresponds to the public parameters pp. We construct a binary tree with T leaves. When we have 8 leaves, we use keys
$$\begin{aligned}\mathsf{sk}_1 &= (\mathsf{sk}_{000}, \mathsf{sk}_{001}, \mathsf{sk}_{01}, \mathsf{sk}_1)\\ \mathsf{sk}_2 &= (\mathsf{sk}_{001}, \mathsf{sk}_{01}, \mathsf{sk}_1)\\ \mathsf{sk}_3 &= (\mathsf{sk}_{010}, \mathsf{sk}_{011}, \mathsf{sk}_1)\end{aligned}$$
and so on. In particular, each day, we ensure that we have the parameters needed to derive all future keys, and delete the parameters that are necessary to derive secret keys in the past. Then $|\mathsf{pk}| = |\mathsf{pp}| = \log_2 T$. Similarly, $|\mathsf{ct}| = O(1)$ and $|\mathsf{sk}| = O(\log^2 T)$.

A similar notion that we can introduce is that of forward secure signatures. Specifically, on day t, we sign using sk_t and verify using pk_t. This will ensure that an adversary who compromises the keys on day t cannot forge signatures on messages in the past. This addresses the real-world issue of theft of the secret signing key. We note that any signature scheme is forward-secure. As we generate secret keys for child noeds, we destory the parent node used to generate the tree. The construction is very similar. We generate new keys on an as-needed basis, where we use the signature scheme to sign the next set of keys that are valid (signatures used to certify validitiy of signatures).

16. 5/27: Multilinear Maps

16.1. **Broadcast Encryption.** We consider another application of bilinear maps to broadcast encryption. A broadcast encryption system is a public-key encryption system where anyone with the public key can encrypt a message to any subset of the users (broadcas Let n be the total number of users in the broadcast encryption system, and let (E, D) be a symmetric key encryption system. Then, the broadcast encryption scheme consists of the following algorithms:

Broadcast Encryption Scheme
- $\mathsf{Setup}(\lambda, n)$: Outputs a public key pk and secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$.
- $\mathsf{Encrypt}(\mathsf{pk}, S, m)$: S is a subset of $\{1, \ldots, n\}$. The encryption algorithm outputs a symmetric encryption key k and an element hdr. The hdr element is also referred to as the header of the broadcast encryption ciphertext. The broadcast data for a message m is the tuple $(S, \mathsf{hdr}, E(k, m))$.
- $\mathsf{Decrypt}(\mathsf{sk}_i, \mathsf{pk}, \mathsf{ct})$: Write ct as the tuple $(S, \mathsf{hdr}, E(k, m))$. The decryption algorithm outputs the message m if $i \in S$ and $\perp$ otherwise.

In the trivial instantiation of the broadcast encryption system, we can simply give each user their own public key. To encrypt a message to a subset $S \subseteq \{1, \ldots, n\}$, we choose a random content key k. The hdr then contains the encryption of k under the public key for each user in S. In this case, observe that the size of the broadcast ciphertext (the hdr component) grows linearly in the size of the broadcast set. In broadcast encryption, our goal is to minimize the size of the ciphertext. Specifically, we aim to construct broadcast encryption systems where the size of the ciphertext is independent of the size of the broadcast set. Before we present the construction, we first describe the security game for a broadcast encryption system.

(Fully Collusion Resistant) Broadcast Encryption Security Game
(1) The adversary begins by submitting a set $S^\star \subseteq \{1, \ldots, n\}$.
(2) The challenger replies with the public key pk and secret keys $\{\mathsf{sk}_i \mid i \in S^\star\}$.
(3) The adversary submits two messages m_0, m_1.
(4) The challenger responds with $\mathsf{Encrypt}(\mathsf{pk}, \overline{S}^\star, m_b)$.
(5) The adversary outputs $b' \in \{0, 1\}$.

The adversary's advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{BE}}$ in the broadcast encryption security game is then given by the quantity

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{BE}} = \left|\Pr\left[b' = 1 \mid b = 0\right] - \Pr\left[b' = 1 \mid b = 1\right]\right|.$$

We say that a broadcast encryption system is secure if no PPT adversary wins the broadcast encryption security game with non-negligible advantage. We say that the broadcast encryption scheme is fully collusion-resistant since even an adversary that has access to an arbitrary subset of keys of its choosing is still unable to win the semantic security challenge for a message encrypted to a set for which it does not have the secret key.

Next, we present the BGW broadcast encryption scheme that leverages bilinear maps [BGW05]. In the BGW scheme, the ciphertext size is $O(1)$, the secret key is $O(1)$, and the public key is $O(n)$, where n is the number of users in the broadcast encryption system. The protocol is described below. Let $\mathbb{G}, \mathbb{G}_T$ be groups of prime order p, and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be an efficiently computable bilinear map. Let g be a generator of $\mathbb{G}$.

BGW Broadcast Encryption Scheme [BGW05]

- $\mathsf{Setup}(n, \lambda)$: Choose $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and $v \xleftarrow{\$} \mathbb{G}$. The public key is $\mathsf{pk} \leftarrow \left(g, g^{\alpha}, g^{(\alpha^2)}, \dots, g^{(\alpha^n)}, g^{(\alpha^{n+2})}, v\right)$. Note that we do *not* include the element $g^{(\alpha^{n+1})}$. The secret key for each user i is given by $\mathsf{sk}_i \leftarrow v^{(\alpha^i)}$.
- $\mathsf{Encrypt}(\mathsf{pk}, S, m)$: Choose $t \xleftarrow{\$} \mathbb{Z}_p$ and let $k = e\left(g, g^{(\alpha^{n+1})}\right)^t$. The broadcast encryption ciphertext hdr is then
$$\mathsf{hdr} \leftarrow \left(g^t, \left(v \cdot \prod_{j \in S} g^{(\alpha^{n+1-j})}\right)^t\right).$$
Output $\mathsf{ct} \leftarrow (S, \mathsf{hdr}, E(k, m))$.
- $\mathsf{Decrypt}(\mathsf{sk}_i, \mathsf{pk}, \mathsf{ct})$: Write ct as $(S, \mathsf{hdr}, E(k, m))$ and hdr as (c_1, c_2). Then, compute
$$k \leftarrow \frac{e\left(c_2, g^{(\alpha^i)}\right)}{e\left(c_1, \mathsf{sk}_i \cdot \prod_{j \in S, j \neq i} g^{(\alpha^{n+1-j+i})}\right)}.$$
Finally, output $m \leftarrow D(k, E(k, m))$.

First, we show that the scheme is correct. First, in a valid encryption to a set S, we have that
$$\mathsf{hdr} = (c_1, c_2) = \left(g^t, \left(v \cdot \prod_{j \in S} g^{(\alpha^{n+1-j})}\right)^t\right).$$
Take some $i \in S$. Then,
$$e\left(g^{(\alpha^i)}, c_2\right) = e\left(g^{(\alpha^i)}, \left(v \cdot \prod_{j \in S} g^{(\alpha^{n+1-j})}\right)^t\right) = e\left(g^t, v^{(\alpha^i)}\right) e\left(g^{(\alpha^i)}, \left[\prod_{j \in S} g^{(\alpha^{n+1-j})}\right]^t\right),$$
and
$$\begin{aligned}
\frac{e\left(g^{(\alpha^i)}, c_2\right)}{e\left(c_1, \mathsf{sk}_i \cdot \prod_{j \in S, j \neq i} g^{(\alpha^{n+1-j+i})}\right)} = \frac{e\left(g^t, v^{(\alpha^i)}\right) e\left(g^{(\alpha^i)}, \prod_{j \in S} g^{(\alpha^{n+1-j})}\right)^t}{e\left(g^t, v^{(\alpha^i)}\right) e\left(g^t, \prod_{j \in S, j \neq i} g^{(\alpha^{n+1-j+i})}\right)} &= \frac{e\left(g^{(\alpha^i)}, g^{(\alpha^{n+1-i})} \prod_{j \in S, j \neq i} g^{(\alpha^{n+1-j})}\right)^t}{e\left(g^{(\alpha^i)}, \prod_{j \in S, j \neq i} g^{(\alpha^{n+1-j})}\right)^t} \\
&= e\left(g^{(\alpha^i)}, g^{(\alpha^{n+1-i})}\right)^t \\
&= e\left(g, g^{(\alpha^{n+1})}\right)^t \\
&= k.
\end{aligned}$$
Correctness then follows from correctness of the underlying symmetric encryption scheme.

Security of the BGW scheme follows from the assumption that given $h, g, g^{\alpha}, g^{(\alpha^2)}, \dots, g^{(\alpha^n)}, g^{(\alpha^{n+2})}, \dots, g^{(\alpha^{2n})}$, $e\left(g, g^{(\alpha^{n+1})}\right)$ is computationally indistinguishable from a uniformly random element in $\mathbb{G}_T$. We note that this assumption holds in the generic group model, that is, the best attack in the generic group model runs in $O\left(\sqrt[3]{p}\right)$.

16.2. **Non-Interactive Key Exchange (NIKE).** We describe another application of pairings, this time for (static) non-interactive key exchange. Suppose we have n parties, and suppose each party holds a secret key $x_1, x_2, \dots, x_n$. Each party then publishes $g^{x_1}, g^{x_2}, \dots, g^{x_n}$. Recall first that in standard Diffie-Hellman key exchange, two parties can negotiate a shared key $H\left(g^{x_i x_j}\right)$ where $H : \mathbb{G} \to \{0,1\}^{128}$. Security of this bipartite key-exchange protocol follows from the Hash Diffie-Hellman assumption. Joux demonstrated a way to extend this to tripartite key-exchange. Specifically, the shared key between parties i, j, k is $k_{ij\ell} = H\left(e(g, g)^{x_i x_j x_\ell}\right)$. Security of this scheme follows from the Hash Bilinear Diffie-Hellman (HBDH) assumption:
$$\left\{g, g^{a_1}, g^{a_2}, g^{a_3}, H\left(e(g, g)^{a_1 a_2 a_3}\right)\right\} \overset{c}{\approx} \left\{g, g^{a_1}, g^{a_2}, g^{a_3}, T\right\}$$

where $T \xleftarrow{\$} \{0,1\}^{128}$ and $g \xleftarrow{\$} \mathbb{G}$, $a_1, \ldots, a_3 \xleftarrow{\$} \mathbb{Z}_p$. Note that we call this static key-exchange because we assume the adversary is not one of the parties, but can only observe the contents of the public bulletic board.

The natural question then is how to generalize to more than 3 parties. One approach is to extend our notion of a bilinear map to an n-linear map. Note that exponentiation can be viewed as a 1-linear map ($n = 1$), and pairings are bilinear maps ($n = 2$). Specifically, $e : \mathbb{G} \times \mathbb{G} \times \cdots \times \mathbb{G} \to \mathbb{G}_T$ is an n-linear map if the following conditions hold:

- **Multilinearity:** $e(g^{\alpha_1}, g^{\alpha_2}, \ldots, g^{\alpha_n}) = e(g, g, \ldots, g)^{\alpha_1 \alpha_2 \cdots \alpha_n}$.
- **Non-Degeneracy:** there exists $g \in \mathbb{G}$ such that $e(g, g, \ldots, g) \neq 1$.
- e is efficiently computable.

Using an n-linear map, we can trivially extend Joux's tripartite key exchange problem to achieve $(n+1)$-party key-exchange. Security in this setting follows from the Hash Multilinear Diffie-Hellman assumption. Note that even a $(\log n)$-linear model already has many applications such as efficient group communication and broadcast encryption.

16.3. **Randomized Encodings and Multilinear Maps.** It is an open problem to construct n-linear maps (from this point forward, we will refer to these n-linear maps as ideal multilinear maps). While we do not know of any construction of ideal multilinear maps, a construction known as randomized encodings (or graded encodings) suffices for many of the applications of ideal multilinear maps. In this section, we present the random encoding construction over the integers due to Coron, Lepoint, and Tibouchi. [CLT13]. First, we define the notion of a graded encoding [GGH13]:

Definition 16.1 (κ-Graded Encoding System [GGH13])**.** A κ-Graded Encoding System for a ring R consists of a system of sets $\mathcal{S} = \left\{ S_i^{(\alpha)} \mid \alpha \in R \text{ and } 0 \leq i \leq \kappa \right\}$ such that the following properties hold:

- For every fixed index $0 \leq i \leq \kappa$, the sets $\left\{ S_i^{(\alpha)} \mid \alpha \in R \right\}$ are pairwise disjoint.
- There are binary operations $+$ and $-$ such that for all $\alpha_1, \alpha_2 \in R$, all $0 \leq i \leq \kappa$, and all $s_1 \in S_i^{(\alpha_1)}, s_2 \in S_i^{(\alpha_2)}$,
$$s_1 + s_2 \in S_i^{(\alpha_1 + \alpha_2)} \text{ and } s_1 - s_2 \in S_i^{(\alpha_1 - \alpha_2)},$$
where $\alpha_1 \pm \alpha_2$ denote the addition and subtraction operations in R.
- There is a binary operation $\times$ such that for all $\alpha_1, \alpha_2 \in R$, all $0 \leq i_1, i_2 \leq \kappa$ where $0 \leq i_1 + i_2 \leq \kappa$, and all $s_1 \in S_{i_1}^{(\alpha_1)}, s_2 \in S_{i_2}^{(\alpha_2)}$ such that
$$s_1 \times s_2 \in S_{i_1 + i_2}^{(\alpha_1 \cdot \alpha_2)},$$
where $\alpha_1 \cdot \alpha_2$ denote the multiplication operation in R.

In a graded encoding system, we generally refer to i as the level of the encoding. At each level i, each ring element $\alpha \in R$ is associated with a corresponding set of encodings $S_i^{(\alpha)}$. One immediate consequence of this is that the encoding of each element need not be unique (in contrast to ideal multilinear maps). Finally, observe that if we have level-1 encodings of $\alpha_1, \ldots, \alpha_n$ in a graded encoding system (with $\kappa = n$), we can obtain the encoding of $\prod_{i=1}^n \alpha_i$ at level n. In this manner, we are able to simulate the operation of ideal multilinear maps. The graded encoding system should support the following operations:

κ-Graded Encoding Scheme Functionality

- $\mathsf{Setup}(\kappa, \lambda)$: Sets up the parameters for the κ-graded encoding system (with security parameter λ). Outputs a pair $(\mathsf{params}, p_{zt})$ consisting of the parameters for the graded encoding scheme as well as a zero-test element p_{zt}.
- $\mathsf{Sample}(\mathsf{params})$: Outputs a level-0 encoding $a \in S_0^{(\alpha)}$ of a nearly uniform element $\alpha \in R$. Note that α should be nearly uniform in R, not that a is nearly uniform in $S_0^{(\alpha)}$.
- $\mathsf{Encode}(\mathsf{params}, i, u)$: Given the level-0 encoding $u \in S_0^{(\alpha)}$ of $\alpha \in R$, outputs a level-i encoding $\hat{u} \in S_i^{(\alpha)}$ of α.
- $\mathsf{Rerandomize}(\mathsf{params}, i, u)$: Given a level-$i$ encoding $u \in S_i^{(\alpha)}$, outputs another encoding $\hat{u} \in S_i^{(\alpha)}$. For any two $u_1, u_2 \in S_i^{(\alpha)}$, the distributions $\{\mathsf{Rerandomize}(\mathsf{params}, i, u_1)\}$ and $\{\mathsf{Rerandomize}(\mathsf{params}, i, u_2)\}$ should be nearly the same.
- $\mathsf{Add}(\mathsf{params}, i, u_1, u_2)$: Given level-$i$ encodings $u_1 \in S_i^{(\alpha_1)}, u_2 \in S_i^{(\alpha_2)}$, outputs a level-$i$ encoding $\hat{u} \in S_i^{(\alpha_1 + \alpha_2)}$.

- $\mathsf{Negate}(\mathsf{params}, i, u)$: Given a level-$i$ encoding $u \in S_i^{(\alpha)}$, outputs a level-i encoding $\hat{u} \in S_i^{(-\alpha)}$.
- $\mathsf{Multiply}(\mathsf{params}, i_1, u_1, i_2, u_2)$: Given a level-$i_1$ encoding $u_1 \in S_{i_1}^{(\alpha_1)}$ and a level-i_2 encoding $u_2 \in S_{i_2}^{(\alpha_2)}$ such that $0 \leq i_1 + i_2 \leq \kappa$, outputs a level-$(i_1 + i_2)$ encoding $\hat{u} \in S_{i_1+i_2}^{(\alpha_1 \alpha_2)}$.
- $\mathsf{ZeroTest}(\mathsf{params}, p_{zt}, u)$: Outputs 1 if $u \in S_\kappa^{(0)}$ and 0 otherwise.
- $\mathsf{Extract}(\mathsf{params}, p_{zt}, u)$: Extracts a canonical and random representation of ring elements from their level-κ encoding, that is, they should satisfy the following properties:
  - For any $\alpha \in R$ and for all $u_1, u_2 \in S_\kappa^{(\alpha)}$, it should be the case that
$$\mathsf{Extract}(\mathsf{params}, p_{zt}, u_1) = \mathsf{Extract}(\mathsf{params}, p_{zt}, u_2).$$
  - The distribution $\left\{\alpha \xleftarrow{\$} R, u \leftarrow S_\kappa^{(\alpha)} : \mathsf{Extract}(\mathsf{params}, p_{zt}, u)\right\}$ is close to uniform.

The construction proceeds as follows.

(1) For an n-linear map, we begin by choosing w secret primes $p_1, \ldots, p_w$ where $w = O(n)$. Then, we publish $N = p_1 p_2 \cdots p_w$.
(2) Choose secret prime q where $q < p_i^{\frac{1}{2n+2}}$.
(3) Choose random $z \xleftarrow{\$} \mathbb{Z}_N$.
(4) An encoding of g^m is $c_m^{(1)} \in \mathbb{Z}_N$ then satisfies $c_m^{(1)} \equiv \frac{r_i q + m}{z} \pmod{p_i}$ and $r_i \xleftarrow{\$} \{1, \ldots, q\}$. Note that the integer $c_m \in \mathbb{Z}_N$ is thus defined via the Chinese Remainder Theorem.
(5) Publish $z^\star = \sum_{i=1}^w h_i z^n \left(q^{-1} \bmod p_i\right) \frac{N}{p_i} \in \mathbb{Z}_N$ where $h_i \xleftarrow{\$} \{1, \ldots, q\}$.

We note that these encodings satisfy the following properties:

- Discrete log is hard (assumed), that is given $c_m^{(1)}$, it is hard to extract m. The best known attacks are all based on lattice reduction attacks (based on LLL).
- Given $c_{m_1}^{(1)}, c_{m_2}^{(1)}, \ldots, c_{m_n}^{(1)}$, we can compute $\prod_{i=1}^n c_{m_i}^{(1)} = c_{m_1 m_2 \cdots m_n}^{(n)}$.
- There is a zero-test in $c^{(n)}$ where if we are given $c_m^{(n)}$, we can test if $m = 0$. In particular, if $m = 0$, then $c_m^{(n)} \cdot z^\star < N \cdot \frac{q}{\max(p_i)}$ (observe that by construction, only the r_i remains. Otherwise, if $m \neq 0$, $c_m^{(n)} \cdot z^\star$ will be on the order of N. Note that we can only test for 0 at the nth level.

17. 5/29: Obfuscation

17.1. **Obfuscation.** The goal of obfuscation is to embed secrets in code. One application of obfuscation is to software watermarking. Specifically, we might embed some secret data into software, and later on, by providing some secret input, we can extract the identifying information (say, if someone tried to pirate the software). Another application is in the construction of short signatures. In this scenario, we begin with a short MAC (for instance, a PRF based MAC). We can bootstrap the short MAC into a short signature scheme by obfuscating the MAC verification algorithm (with the key embedded in the algorithm).

We formally define obfuscators for circuits. Specifically, a circuit obfuscator $\mathcal{O}$ takes as input a circuit $\mathcal{C}$ and produces an obfuscated circuit $\mathcal{C}_\mathcal{O}$. We require the obfuscator $\mathcal{O}$ satisfy the following properties:

- For all inputs x to the circuit $\mathcal{C}$, $\mathcal{C}_\mathcal{O}(x) = \mathcal{C}(x)$.
- The obfuscator $\mathcal{O}$ is efficient.
- The size $|\mathcal{C}_\mathcal{O}|$ of the obfuscated circuit is polynomial in the size $|\mathcal{C}|$ of the original circuit: $|\mathcal{C}_\mathcal{O}| = \text{poly}(|\mathcal{C}|)$.

Our first notion of security for an obfuscator is the notion of virtual black-box obfuscation. At a high level, an obfuscator satisfies the virtual black-box property if anything that is learnable from $\mathcal{C}_\mathcal{O}$ is learnable by evaluating it on inputs of our choice.

Definition 17.1. For all PPT adversaries $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that the output of $\mathcal{S}$ given oracle access to $\mathcal{C}$ is indistinguishable from the output of $\mathcal{A}$ on $\mathcal{C}_\mathcal{O}$. In other words,

$$\left|\Pr\left[\mathcal{A}(\mathcal{C}_\mathcal{O}) = 1\right] - \Pr\left[\mathcal{S}^\mathcal{C}\left(1^{|C|}\right) = 1\right]\right| < \mathsf{negl}$$

is negligible.

Theorem 17.2. *Virtual black-box obfuscation is impossible (even for unbounded obfuscators $\mathcal{O}$).*

Since virtual black-box obfuscation is impossible, we weaken our security requirement to a notion called indistinguishability obfuscation ($i\mathcal{O}$). In $i\mathcal{O}$, we relax our security requirement and just require that the obfuscations of two circuits that compute the same functionality are computationally indistinguishable.

Definition 17.3. Let $i\mathcal{O}$ be an indistinguishability obfuscator. Then for all circuits $\mathcal{C}$ and $\mathcal{C}'$ such that on all inputs x, $\mathcal{C}(x) = \mathcal{C}'(x)$,

$$i\mathcal{O}(\mathcal{C}) \overset{c}{\approx} i\mathcal{O}(\mathcal{C}').$$

We sketch out the construction of indistinguishability obfuscators. The starting point for our construction will be an obfuscator for a shallow circuit. Using a result by Barrington, we can represent a shallow circuit (the circuit class $\mathsf{NC}[1]$) as a branching program. A branching program consists of a series of matrices

$$\begin{matrix} B_{10} & B_{20} & \cdots & B_{n0} \\ B_{11} & B_{21} & \cdots & B_{n1} \end{matrix}$$

and an input function $\text{inp}(\cdot)$ that maps a bit j to an index i_j. To evaluate a branching program $\text{BP}(\cdot)$ on an input x, we compute the product

$$\text{BP}(x) = \begin{cases} 1 & \prod_{j=1}^{\ell} B_{j x_{\text{inp}(j)}} = I \\ 0 & \text{otherwise}, \end{cases}$$

where I denotes the identity matrix. Next, we randomize the matrices in the branching program. To complete the construction, we place the matrices into the exponent in a multilinear map:

$$\begin{matrix} g^{B_{10}} & g^{B_{20}} & \cdots & g^{B_{n0}} \\ g^{B_{11}} & g^{B_{21}} & \cdots & g^{B_{n1}} \end{matrix}.$$

Using the multilinear map, it becomes possible to evaluate the matrix product in the exponent. If the product of the matrices evaluates to the identity matrix, then using the zero-test element in the graded encoding scheme, we can determine the result of the computation. Using fully homomorphic encryption, we can bootstrap the obfuscator for $\mathsf{NC}[1]$ to all circuits.

17.2. **Applications of Indistinguishability Obfuscation.** We describe some applications of $i\mathcal{O}$. First, we describe a short signature scheme. Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a secure PRF and f be a OWF.

- $\mathsf{Setup}(\lambda)$: Choose $k \xleftarrow{\$} \mathcal{K}$. Construct the circuit $\mathcal{C}(m, \sigma)$ that outputs 1 if $f(F(k, m)) = f(\sigma)$ and 0 otherwise. Let $\mathcal{C}_{i\mathcal{O}} = i\mathcal{O}(\mathcal{C})$. The public key is $\mathcal{C}_{i\mathcal{O}}$ and the secret key is k.
- $\mathsf{Sign}(\mathsf{sk}, m)$: Output the signature $\sigma = \mathsf{PRF}(k, m)$.
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: Accept if $\mathcal{C}_{i\mathcal{O}}(m, \sigma) = 1$.

Theorem 17.4. *If $i\mathcal{O}$ is an indistinguishability obfuscator, f is a* OWF*, and F is a secure puncturable* PRF*, then the above signature scheme is selectively secure.*

Note that we can convert the selectively secure signature scheme into a fully secure signature scheme in the random oracle model. We consider another application of $i\mathcal{O}$, this time, to multiparty key exchange. While it was possible to build non-interactive multiparty key exchange from multilinear maps, obfuscation enables a scheme that does not require a trusted setup phase (a trusted party was needed to set up the multilinear map).

We describe the non-interactive key exchange protocol based on obfuscation. Let $G : \mathcal{S} \to \mathcal{X}$ be a secure PRG. First, each party $i = 1, \ldots, n$ chooses a random seed $x_i \xleftarrow{\$} \mathcal{S}$. Each party then published $x_i \xleftarrow{\$} G(x_i)$. Then, one of the parties constructs the following circuit $\mathcal{C}$. On input $x_1, \ldots, x_n, i \in \{1, \ldots, n\}, s$, the circuit first checks that $x_i = G(s)$. If not, then the circuit outputs $\perp$. Otherwise, the circuit outputs $F(k, x_1 \| x_2 \| \cdots \| x_n)$, where F is a secure PRF. Then, the party published $\mathcal{C}_{i\mathcal{O}} = i\mathcal{O}(\mathcal{C})$. Effectively, the circuit constitutes a proof for knowing the preimage of their public value under G. The shared key is then $F(k, x_1 \| \cdots \| x_n)$. Clearly, each party can evaluate $\mathcal{C}_{i\mathcal{O}}$ to learn the shared key.

Theorem 17.5. *If $i\mathcal{O}$ is an indistinguishability obfuscator, G is a secure* PRG *with expansion factor 2, and F is a puncturable* PRF*, then the above protocol provides secure non-interactive key-exchange.*

18. 6/3: Applications of Elliptic Curves

18.1. **Factoring Algorithms.** Let $N = pq$ be a product of two primes, and without loss of generality, suppose $p \leq q$. Currently, the best known factoring algorithm is the general number field sieve, which runs in time roughly $e^{\sqrt[3]{\log N}}$. Using elliptic curves, we can construct a factoring algorithm with runtime roughly $e^{\sqrt{\log B}}$ where $p < B$. In other words, the runtime depends on the size of the smallest prime factor of N, and not the value of N (as in the general number field sieve). Specifically, this means that the elliptic curve factoring method can be more efficient than the general number field sieve when the prime factors of N differ greatly in magnitude.

To motivate the Lenstra's elliptic curve factorization method (ECM), we first present Pollard's $p-1$ factoring method [Pol74]. First, we define the notion of a smooth number, which frequently arises in the discussion of factorization algorithms. Then, we state some results pertaining to the distribution of smooth integers.

Definition 18.1. An integer $x \in \mathbb{Z}^+$ is s-smooth if whenever $p \mid x$, $p < s$. The function $\Psi(x, s)$ counts the number of s-smooth integers less than or equal to x:

$$\Psi(x, s) := \left|\{0 < y < x : y \text{ is } s\text{-smooth}\}\right|.$$

Example 18.2. Take $x = 900$. Then, the prime factorization of x is $2^2 \cdot 3^2 \cdot 5^2$. Since all prime factors of x are less than or equal to 5, we say that 900 is 5-smooth.

Theorem 18.3 (Canfield-Erdős-Pomerance [CEP83])**.** *Let $0 < \varepsilon < 1$, and suppose (for sufficiently large) x, s that*

$$(\log x)^{\varepsilon} < \log s < (\log x)^{1-\varepsilon}.$$

Define $z = \frac{\log x}{\log s}$. Then,

$$\Psi(x, s) = x \cdot z^{-z(1+o(1))}.$$

Corollary 18.4. *Let $L(x) = \exp\left(\sqrt{\log x \log\log x}\right)$. Then,*

$$\Pr\left[t \xleftarrow{\$} \{1, \ldots, x\} : t \textit{ is } L(x)\textit{-smooth}\right] \geq \frac{1}{L(x)^{\frac{1}{2}+o(1)}}.$$

Proof. We use the Canfield-Erdős-Pomerance (Theorem 18.3). Set $s = L(x)$. Then,

$$\log s = \log L(x) = \sqrt{\log x \log\log x},$$

which satisfies $(\ln x)^{\varepsilon} < \ln s < (\ln x)^{1-\varepsilon}$, for sufficiently large x, s and $0 < \varepsilon < \frac{1}{2}$. Thus, we conclude that

$$\Psi(x, s) = \Psi(x, L(x)) = x \cdot z^{-z(1+o(1))}$$

where

$$z = \frac{\log x}{\log s} = \frac{\log x}{\sqrt{\log x \log\log x}} = \sqrt{\frac{\log x}{\log\log x}}.$$

Next,

$$\Pr\left[t \xleftarrow{\$} \{1, \ldots, x\} : t \text{ is } L(x)\text{-smooth}\right] = \frac{\Psi(x, s)}{x} = z^{-z(1+o(1))}.$$

It remains to show that $z^{-z(1+o(1))} = L(x)^{-\frac{1}{2}+o(1)}$. First, we show that $z \log z = \left(\frac{1}{2} + o(1)\right) \log L(x)$:

$$\begin{aligned} z \log z = \sqrt{\frac{\log x}{\log\log x}} \left[\log\log x - \log\log s\right] &= \sqrt{\log x \log\log x} - z \log\log s \\ &= \log s - z\left[\frac{1}{2} \log\left(\log x \log\log x\right)\right] \\ &= \left[1 - \frac{1}{2} \frac{z \log\left(\log x \log\log x\right)}{\log s}\right] \log s. \end{aligned}$$

Next we note that

$$\frac{z \log\left(\log x \log\log x\right)}{\log s} = \frac{\log x \log\left(\log x \log\log x\right)}{\log^2 s} = \frac{\log\left(\log x \log\log x\right)}{\log\log x} = 1 + \frac{\log\log\log x}{\log\log x} = 1 + o(1),$$

and so we conclude that

$$z \log z = \log s \left[1 - \frac{1}{2}\left(1 + o(1)\right)\right] = \left[\frac{1}{2} - o(1)\right] \log s.$$

Exponentiating both sides, the claim follows.

$$z^{-z} = s^{-\frac{1}{2}+o(1)} = L(x)^{-\frac{1}{2}+o(1)}.$$

□

18.1.1. *Pollard's $p-1$ Method.* With this preparation, we are ready to describe Pollard's $p-1$ method. Suppose that we have found an integer ℓ such that $p-1 \mid \ell$ while $q-1 \nmid \ell$. Specifically, this means that there exist integers k_1, k_2, k_3 such that $\ell = k_1(p-1)$ and $\ell = k_2(q-1)+k_3$ with $0 < k_3 < q-1$. Take $g \xleftarrow{\$} \mathbb{Z}_N^*$. By Fermat's theorem, it follows that

$$g^{\ell} = g^{k_1(p-1)} = 1 \pmod{p}$$
$$g^{\ell} = g^{k_2(q-1)+k_3} = g^{k_3} \pmod{q}.$$

Since $k_3 \neq 0$, $g^{\ell} \neq 1 \pmod{q}$ with high probability. But this means that $a^{\ell}-1$ is a multiple of p and not a multiple of q. We can then recover p by computing $\gcd\left(a^{\ell}-1, N\right)$.

Thus, to factor N, it suffices to find ℓ such that $p-1 \mid \ell$ and $q-1 \nmid \ell$. This is easy to do so if $p-1$ is s-smooth and $q-1$ is not. Consider the product

$$L = \prod_{\text{prime } p<s} \ell^{\lfloor \log_{\ell} s \rfloor}.$$

By construction, $p-1 \mid L$ (since $p-1$ is s-smooth) and $q-1 \nmid L$ (since $q-1$ is not s-smooth). Then, taking a random $g \in \mathbb{Z}_N^*$, we have that with high probability, $g^L = 1 \pmod{p}$ and $g^L \neq 1 \pmod{q}$. As demonstrated above, evaluating $\gcd\left(g^L-1, N\right)$ will yield one of the factors of N. The specific algorithm is outlined below for a given value of s.

Pollard's $p-1$ Factorization Method

(1) Compute $L = \prod_{\text{prime } p<s} \ell^{\lfloor \log_{\ell} s \rfloor}$.

(2) Take $g \xleftarrow{\$} \mathbb{Z}_N^*$. Note that it suffices to sample $g \xleftarrow{\$} \mathbb{Z}_N$, since in the unlikely event that $\gcd(g, N) \neq 1$, we have recovered a factor of N.

(3) Compute $k \leftarrow \gcd\left(g^L-1, N\right)$. If $k=1$ or $k=N$, then output $\perp$. Otherwise, output k.

In practice, note that L can be a very large integer, so g^L might be very big. However, we note that it is not necessary to compute g^L over the integers. Since the quantity we are ultimately interested in is $\gcd\left(g^L-1, N\right)$, it suffices to compute $g^L \pmod{N}$. Additionally, for the smoothness parameter s, we generally choose $s = L(N) = \sqrt{\log N \log\log N}$. We can also invoke the above algorithm for different values of s (increasing and decreasing s) as necessary until we recover the factorization of N.

In light of Pollard's $p-1$ method, when choosing RSA moduli, it is essential to choose p, q such that $p-1$ and $q-1$ do not consist exclusively of small prime factors. It turns out, however, that this alone is not sufficient. We first note that Pollard's $p-1$ method works because $\left|\mathbb{Z}_p^*\right| = p-1$, and so, every element $g \in \mathbb{Z}_p^*$ satisfies $g^{p-1} = 1 \in \mathbb{Z}_p^*$. In a similar manner, we can construct an algorithm for the scenario where $p+1$ is smooth by working over an appropriate extension field - in this particular case, Williams describes an algorithm working over a quadratic extension [Wil82].

18.1.2. *Lenstra's Elliptic Curve Factorization (ECM) Method.* In the previous section, we noted that Pollard's $p-1$ method exploited the fact that the multiplicative group $\mathbb{Z}_p^*$ always has order $p-1$. Lenstra's method improves upon the Pollard $p-1$ method by instead working over a random elliptic curve group $E(\mathbb{F}_p)$ over $\mathbb{F}_p$. It turns out that if the order of $E(\mathbb{F}_p)$ is smooth, it is again possible to factor the modulus N.

Before describing the factoring algorithm, we briefly review some properties of elliptic curves. Fix a prime p and take $a, b \in \mathbb{F}_p$ where $4a^3 + 27b^2 \neq 0$ (non-zero determinant). Then, the Hasse-Weil Theorem (Theorem 13.1) states that

$$|E_{a,b}(\mathbb{F})| = \left|\left\{(x,y) \in \mathbb{F}_p \mid y^2 = x^3 + ax + b\right\}\right| = p+1-t$$

where $|t| < 2\sqrt{p}$. In particular, this means that $|E_{a,b}(\mathbb{F})| \in \left[p+1-2\sqrt{p}, p+1+2\sqrt{p}\right]$.

Fact 18.5. *Take $a, b \xleftarrow{\$} \mathbb{F}_p$. Then $|E_{a,b}(\mathbb{F})|$ is statistically close to uniform in $\left[p-2\sqrt{p}, p+2\sqrt{p}\right]$.*

Conjecture 18.6. *Corollary 18.4 still holds when we sample t uniformly at random from the interval $\left[p-2\sqrt{p}, p+2\sqrt{p}\right]$.*

We now describe Lenstra's elliptic curve factorization method. The high level idea is to first choose a random elliptic curve $E(\mathbb{Z}_N)$ with coordinates in $\mathbb{Z}_N$. Note that because $\mathbb{Z}_N$ is not a field, the set of points $E(\mathbb{Z}_N)$ with coordinates in $\mathbb{Z}_N$ do not form a field. Then, we hope that $|E_{a,b}(\mathbb{F}_p)|$ is s-smooth but $|E_{a,b}(\mathbb{F}_q)|$ is not s-smooth. If these conditions are met, then we can factor the modulus N.

Lenstra's Elliptic Curve Factorization Method (ECM)

(1) Choose random $a, b \xleftarrow{\$} \mathbb{Z}_N$ and a point $P = (x_0, y_0) \in E_{a,b}(\mathbb{Z}_N)$. This can be done by first choosing P and then choosing a, b at random subject to the constraint that $y_0^2 = x_0^3 + ax_0 + b$ is satisfied. Note that once x_0, y_0 are fixed, the relationship between a and b is linear.

(2) Let $s = L(B)$ and try to compute Q:

$$Q = \left(\prod_{\text{prime } \ell < s} \ell^{\lfloor \log_\ell B \rfloor} \right) \cdot P \in E_{a,b}(\mathbb{Z}_N).$$

Note that Q may not be well-defined.

(3) If we were able to compute Q in the previous step, then restart the algorithm. If in the process of computing Q, we needed to and were unable to invert an element v in $\mathbb{Z}_N$, then it follows that $\gcd(v, N) \neq 1$, in which case, we output v.

We note that the above algorithm succeeds when $E_{a,b}(\mathbb{F}_p^*)$ is s-smooth, since repeated additions will eventually require the computation of an inverse of 0 (mod p), which does not exist. The expected number of iterations needed to find a, b such that $|E_{a,b}(\mathbb{Z}_p)|$ is $L(B)$-smooth is of order $L(B)^{\frac{1}{2}+o(1)}$ (recall that we are assuming that $p < B$). The complexity of each iteration is $O\left(L(B)^2 \log^2 N\right)$ and so the total run time is of order $O(L(B)^3) = \exp\left(3\sqrt{\log B \log\log B}\right)$.

18.2. **Primality Testing.** The goal in primality testing is to decide, given $n \in \mathbb{Z}$, that n is prime. One of the earliest primality tests is Fermat's test which works as follows:

(1) Choose $a \xleftarrow{\$} \mathbb{Z}_n^*$.
(2) Compute $a^{n-1} = 1 \pmod{n}$. If so, output "prime", and otherwise, output "not prime."

First, we note that this test will always output "prime" when given a prime number as input. However, there are also many non-primes for which it will output "prime." Such numbers are called Carmichael numbers. For instance, $561 = 3 \cdot 11 \cdot 17$ is the first Carmichael number. To see that this is the case,

$$a^{560} = \left(a^{280}\right)^2 = \left(a^{56}\right)^{10} = \left(a^{35}\right)^{16} = 1 \pmod{561}$$

using the Chinese Remainder Theorem. Pomerance demonstrated that there are infinitely many Carmichael numbers, and the k^{th} Carmichael number is of order $2^{2^{2^k}}$.

A randomized primality test is the Miller-Rabin primality test. If $n = 3 \pmod{4}$, then we choose $a \xleftarrow{\$} \mathbb{Z}_N^*$, the test checks that $a^{\frac{n-1}{2}} = \pm 1$. It is possible to derandomize the Miller-Rabin primality test by testing all $a \in \left\{1, \ldots, \log^2 n\right\}$. The proof that this algorithm is correct requires the Extended Riemann Hypothesis.

Next, we describe a primality proving technique due to Pockington. Suppose $n - 1 = mp$ where $p > \sqrt{n}$ is prime.

Theorem 18.7 (Pocklington)**.** *Let $p \geq \sqrt{n}$ be a prime. If $g \in \mathbb{Z}_n^*$ such that $g^{n-1} = 1 \pmod{n}$ and $\gcd\left(g^{\frac{n-1}{p}} - 1, n\right) = 1$, then n is prime.*

Proof. Let ℓ be the smallest prime divisor of n. Then, we are given that $g^{n-1} = 1 \pmod{n}$ and since $\ell \mid n$, it is also the case that $g^{n-1} \pmod{\ell}$, and so we have that $\operatorname{ord}_\ell(g) \mid n - 1$. Moreover, from the second condition, $\operatorname{ord}_\ell(g) \nmid \frac{n-1}{p}$, since otherwise, $\gcd\left(g^{\frac{n-1}{p}} - 1, n\right) \geq \ell > 1$. Since p is prime, it must be the case that $p \mid \operatorname{ord}_\ell(g)$, and so $\operatorname{ord}_\ell(g) \geq p > \sqrt{n}$. Thus, $\ell > \sqrt{n}$, which is impossible (since ℓ is the smallest prime factor of n). $\square$

Observe that in order to leverage the above algorithm, we need to know a partial factorization of $n - 1$. To handle the case where we do not know the factorization, we leverage elliptic curves. Specifically, we first choose $a, b \xleftarrow{\$} \mathbb{Z}_n$. Then, let $t \leftarrow \left|(x, y) \in \mathbb{Z}_n : y^2 = x^3 + ax + b\right|$ be the number of points on the elliptic curve (using an algorithm due to Schoof). If t is not the product of a small prime (that we can easily verify) and a large prime, then we repeat and choose a new elliptic curve. Then, we invoke Pockington's algorithm on $E_{a,b}$. Finally, we recursive invoke the algorithm on the large prime. The certificate is then the curve $E_{a,b}$ along with the generator g used for the primality test.

References

[BBS86] Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(2):364–383, 1986.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*, pages 258–275, 2005.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, pages 514–532, 2001.

[BM82] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *FOCS*, pages 112–117, 1982.

[BM89] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In *CRYPTO*, pages 547–557, 1989.

[CEP83] E. R. Canfield, Paul Erdős, and Carl Pomerance. On a problem of oppenheim concerning 'factorisatio numerorum'. *J. Number Theory*, 17:1–28, 1983.

[Cha82] David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO (1)*, pages 476–493, 2013.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, pages 402–414, 1999.

[CNS07] Jan Camenisch, Gregory Neven, and Abhi Shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT*, pages 573–590, 2007.

[CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO*, pages 89–105, 1992.

[Edw07] Harold M. Edwards. A normal form for elliptic curves. In *Bulletin of the American Mathematical Society*, pages 393–422, 2007.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC*, pages 416–426, 1990.

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *STOC*, pages 25–32, 1989.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *FOCS*, pages 174–187, 1986.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *CRYPTO*, pages 216–231, 1988.

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *STOC*, pages 59–68, 1986.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions. In *STOC*, pages 12–24, 1989.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, pages 364–373, 1997.

[Näs96] Mats Näslund. All bits ax+b mod p are hard (extended abstract). In *CRYPTO*, pages 114–128, 1996.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *FOCS*, pages 458–467, 1997.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.

[Pol74] John M. Pollard. Theorems of factorization and primality testing. *Proceedings of the Cambridge Philosophical Society*, 76(3):521–528, 1974.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, pages 239–252, 1989.

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[Sma99] Nigel P. Smart. The discrete logarithm problem on elliptic curves of trace one. *J. Cryptology*, 12(3):193–196, 1999.

[Wil82] H. C. Williams. A p+1 method of factoring. *Mathematics of Computation*, 39(159):225–234, 1982.

Index