SANJAM GARG

A COURSE IN THEORY OF CRYPTOGRAPHY

This draft was compiled on Monday 16th September, 2024.

Contents

Preface

Cryptography enables many paradoxical objects, such as public key encryption, verifiable electronic signatures, zero-knowledge protocols, and fully homomorphic encryption. The two main steps in developing such seemingly impossible primitives are (i) defining the desired security properties formally and (ii) obtaining a construction satisfying the security property provably. In modern cryptography, the second step typically assumes (unproven) computational assumptions, which are conjectured to be computationally intractable. In this course, we will define several cryptographic primitives and argue their security based on well-studied computational hardness assumptions. However, we will largely ignore the mathematics underlying the assumed computational intractability assumptions.

Acknowledgements

These lecture notes are based on scribe notes taken by students in CS 276 over the years. Also, thanks to Peihan Miao, Akshayaram Srinivasan, and Bhaskar Roberts for helping to improve these notes.

1
Mathematical Background

In modern cryptography, (1) we typically assume that our attackers cannot run in unreasonably large amounts of time, and (2) we allow security to be broken with a *very small*, but non-zero, probability.

Without these assumptions, we must work in the realm of information-theoretic cryptography, which is often unachievable or impractical for many applications. For example, the one-time pad [1] – an information-theoretically secure cipher – is not very useful because it requires very large keys.

[1] For a message $m \in \{0,1\}^n$ and a random key $k \in \{0,1\}^n$, the encryption of m is $c = m \oplus k$. The decryption is $m = c \oplus k$.

In this chapter, we define items (1) and (2) more formally. We require our adversaries to run in polynomial time, which captures the idea that their runtime is not unreasonably large (sections 1.1). We also allow security to be broken with negligible – very small – probability (section 1.2).

1.1 Probabilistic Polynomial Time

A probabilistic Turing Machine is a generic computer that is allowed to make random choices during its execution. A probabilistic *polynomial time* Turing Machine is one which halts in time polynomial in its input length. More formally:

Definition 1.1 (Probabilistic Polynomial Time). *A probabilistic Turing Machine M is said to be PPT (a Probabilistic Polynomial Time Turing Machine) if $\exists c \in \mathbb{Z}^+$ such that $\forall x \in \{0,1\}^*$, $M(x)$ halts in $|x|^c$ steps.*

A *non-uniform* PPT Turing Machine is a collection of machines one for each input length, as opposed to a single machine that must work for all input lengths.

Definition 1.2 (Non-uniform PPT). *A non-uniform PPT machine is a sequence of Turing Machines $\{M_1, M_2, \cdots\}$ such that $\exists c \in \mathbb{Z}^+$ such that $\forall x \in \{0,1\}^*$, $M_{|x|}(x)$ halts in $|x|^c$ steps.*

1.2 Noticeable and Negligible Functions

Noticeable and negligible functions are used to characterize the "largeness" or "smallness" of a function describing the probability of some event. Intuitively, a noticeable function is required to be larger than some inverse-polynomially function in the input parameter. On the other hand, a negligible function must be smaller than any inverse-polynomial function of the input parameter. More formally:

Definition 1.3 (Noticeable Function)**.** *A function $\mu(\cdot) : \mathbb{Z}^+ \to [0,1]$ is noticeable iff $\exists c \in \mathbb{Z}^+, n_0 \in \mathbb{Z}^+$ such that $\forall n \geq n_0,\ \mu(n) > n^{-c}$.*

Example. Observe that $\mu(n) = n^{-3}$ is a noticeable function. (Notice that the above definition is satisfied for $c = 4$ and $n_0 = 1$.)

Definition 1.4 (Negligible Function)**.** *A function $\mu(\cdot) : \mathbb{Z}^+ \to [0,1]$ is negligible iff $\forall c \in \mathbb{Z}^+\ \exists n_0 \in \mathbb{Z}^+$ such that $\forall n \geq n_0,\ \mu(n) < n^{-c}$.*

Example. $\mu(n) = 2^{-n}$ is an example of a negligible function. This can be observed as follows. Consider an arbitrary $c \in \mathbb{Z}^+$ and set $n_0 = c^2$. Now, observe that for all $n \geq n_0$, we have that $\frac{n}{\log_2 n} \geq \frac{n_0}{\log_2 n_0} > \frac{n_0}{\sqrt{n_0}} = \sqrt{n_0} = c$. This allows us to conclude that

$$\mu(n) = 2^{-n} = n^{-\frac{n}{\log_2 n}} < n^{-c}.$$

Thus, we have proved that for any $c \in \mathbb{Z}^+$, there exists $n_0 \in \mathbb{Z}^+$ such that for any $n \geq n_0$, $\mu(n) < n^{-c}$.

Gap between Noticeable and Negligible Functions. At first thought it might seem that a function that is not negligible (or, a non-negligible function) must be a noticeable. This is not true![2] Negating the definition of a negligible function, we obtain that a non-negligible function $\mu(\cdot)$ is such that $\exists c \in \mathbb{Z}^+$ such that $\forall n_0 \in \mathbb{Z}^+$, $\exists n \geq n_0$ such that $\mu(n) > n^{-c}$. Note that this requirement is satisfied as long as $\mu(n) > n^{-c}$ for infinitely many choices of $n \in \mathbb{Z}^+$. However, a noticeable function requires this condition to be true for every $n \geq n_0$.

[2] Mihir Bellare. A note on negligible functions. *Journal of Cryptology*, 15 (4):271–284, September 2002. DOI: 10.1007/s00145-002-0116-x

Below we give example of a function $\mu(\cdot)$ that is neither negligible nor noticeable.

$$\mu(n) = \begin{cases} 2^{-n} & : x \mod 2 = 0 \\ n^{-3} & : x \mod 2 \neq 0 \end{cases}$$

This function is obtained by interleaving negligible and noticeable functions. It cannot be negligible (resp., noticeable) because it is greater (resp., less) than an inverse-polynomially function for infinitely many input choices.

Properties of Negligible Functions. Sum and product of two negligible functions is still a negligible function. We argue this for the sum function below and defer the problem for products to Exercise 2.2. These properties together imply that any polynomial function of a negligible function is still negligible.

Exercise 1.1. *If $\mu(n)$ and $\nu(n)$ are negligible functions from domain $\mathbb{Z}^+$ to range $[0,1]$ then prove that the following functions are also negligible:*

1. $\psi_1(n) = \frac{1}{2} \cdot (\mu(n) + \nu(n))$
2. $\psi_2(n) = \min\{\mu(n) + \nu(n), 1\}$
3. $\psi_3(n) = \mu(n) \cdot \nu(n)$
4. $\psi_4(n) = \mathsf{poly}(\mu(n))$*, where* $\mathsf{poly}(\cdot)$ *is an unspecified polynomial function. (Assume that the output is also clamped to* $[0,1]$ *to satisfy the definition)*

function.

Proof.

1. We need to show that for any $c \in \mathbb{Z}^+$, we can find n_0 such that $\forall n \geq n_0$, $\psi_1(n) \leq n^{-c}$. Our argument proceeds as follows. Given the fact that μ and ν are negligible we can conclude that there exist n_1 and n_2 such that $\forall n \geq n_1$, $\mu(n) < n^{-c}$ and $\forall n \geq n_2$, $g(n) < n^{-c}$. Combining the above two facts and setting $n_0 = \max(n_1, n_2)$ we have that for every $n \geq n_0$,

$$\psi_1(n) = \frac{1}{2} \cdot (\mu(n) + \nu(n)) < \frac{1}{2} \cdot (n^{-c} + n^{-c}) = n^{-c}$$

   Thus, $\psi_1(n) \leq n^{-c}$ and hence is negligible.

2. We need to show that for any $c \in \mathbb{Z}^+$, we can find n_0 such that $\forall n \geq n_0$, $\psi_2(n) \leq n^{-c}$. Given the fact that μ and ν are negligible, there exist n_1 and n_2 such that $\forall n \geq n_1$, $\mu(n) \leq n^{-c-1}$ and $\forall n \geq n_2$, $g(n) \leq n^{-c-1}$. Setting $n_0 = \max(n_1, n_2, 3)$ we have that for every $n \geq n_0$,

$$\psi_2(n) = \min\{\mu(n) + \nu(n), 1\} < n^{-c-1} + n^{-c-1} < n^{-c}$$

□

1.3 *Computationally Hard Problems*

We will next provide certain number theoretical problems that are conjectured to be computationally intractable. We will use the conjectured hardness of these problems in subsequent chapters to o provide concrete instantiations.

1.3.1 The Discrete-Log Family of Problem

Consider a group $\mathbb{G}$ of prime order. For example, consider the group $\mathbb{Z}_p^*$ where p is a large prime. Let g be a generator of this group $\mathbb{G}$. In this group, given g^x for a random $x \in \{1, \ldots p-1\}$ consider the problem of finding x. This problem, referred to as the discrete-log problem, is believed to be computationally hard.

The asymptotic definition of the discrete-log problem needs to consider an infinite family of groups or what we will a group ensemble.

Group Ensemble. A group ensemble is a set of finite cyclic groups $\mathcal{G} = \{\mathbb{G}_n\}_{n\in\mathbb{N}}$. For the group G_n, we assume that given two group elements in G_n, their sum can be computed in polynomial in n time. Additionally, we assume that given n the generator g of $\mathbb{G}_n$ can be computed in polynomial time.

Definition 1.5 (Discrete-Log Assumption)**.** *We say that the discrete-log assumption holds for the group ensemble $\mathcal{G} = \{\mathbb{G}_n\}_{n\in\mathbb{N}}$, if for every non-uniform PPT algorithm $\mathcal{A}$ we have that*

$$\mu_{\mathcal{A}}(n) := \Pr_{x \leftarrow |G_n|}[\mathcal{A}(g, g^x) = x]$$

is a negligible function.

The Diffie-Hellman Problems. In addition to the discrete-log assumption, we also define the Computational Diffie-Hellman Assumption and the Decisional Diffie-Hellman Assumption.

Definition 1.6 (Computational Diffie-Hellman (CDH) Assumption)**.** *We say that the Computational Diffie-Hellman Assumption holds for the group ensemble $\mathcal{G} = \{\mathbb{G}_n\}_{n\in\mathbb{N}}$, if for every non-uniform PPT algorithm $\mathcal{A}$ we have that*

$$\mu_{\mathcal{A}}(n) := \Pr_{x,y \leftarrow |G_n|}[\mathcal{A}(g, g^x, g^y) = g^{xy}]$$

is a negligible function.

Definition 1.7 (Decisional Diffie-Hellman (DDH) Assumption)**.** *We say that the Computational Diffie-Hellman Assumption holds for the group ensemble $\mathcal{G} = \{\mathbb{G}_n\}_{n\in\mathbb{N}}$, if for every non-uniform PPT algorithm $\mathcal{A}$ we have that*

$$\mu_{\mathcal{A}}(n) = \left| \Pr_{x,y \leftarrow |G_n|}[\mathcal{A}(g, g^x, g^y, g^{xy}) = 1] - \Pr_{x,y,z \leftarrow |G_n|}[\mathcal{A}(g, g^x, g^y, g^z) = 1] \right|$$

is a negligible function.

It is not hard to observe that the discrete-log assumption is the weakest of the three assumptions above. In fact, it is not difficult to show that the Discrete-Log Assumption for $\mathcal{G}$ implies the CDH and the DDH Assumptions for $\mathcal{G}$. Additionally, we leave it as an exercise to show that the CDH Assumption for $\mathcal{G}$ implies the DDH Assumptions for $\mathcal{G}$.

Examples of Groups where these assumptions hold. Now we provide some examples of group where these assumptions hold.

1. Consider the group $\mathbb{Z}_p^*$ for a prime p.[3] For this group the CDH Assumption is conjectured to be true. However, using the Legendre symbol,[4] the DDH Assumption in this group can be shown to be false. Can you show how?[5]
2. Let $p = 2q + 1$ where both p and q are prime.[6] Next, let $\mathbb{Q}$ be the order-q subgroup of quadratic residues in $\mathbb{Z}_p^*$. For this group, the DDH assumption is believed to hold.
3. Let $N = pq$ where $p, q, \frac{p-1}{2}$ and $\frac{q-1}{2}$ are primes. Let $\mathbb{QR}_N$ be the cyclic subgroup of qudratic resides of order $\phi(N) = (p-1)(q-1)$. For this group $\mathbb{QR}_N$, the DDH assumption is also believed to hold.

Is DDH strictly stronger than Discrete-Log? In the example cases above, where DDH is known believed to be hard, the best known algorithms for DDH are no better than the best known algorithms for the discrete-log problem. Whether the DDH assumption is strictly stronger than the discrete-log assumption is an open problem.

1.3.2 *CDH in $\mathbb{QR}_N$ implies Factoring*

In this section, we will show that the CDH assumption in $\mathbb{QR}_N$ implies the factoring assumption.

Lemma 1.1. *Given an algorithm $\mathcal{A}$ that breaks the CDH assumption in $\mathbb{QR}_N$, we construct an non-uniform PPT adversary $\mathcal{B}$ that on input N outputs its prime factors p and q.*

Proof. Given that $\mathcal{A}$ is an algorithm that solves the CDH problem in $\mathbb{QR}_N$ with a non-negligible probability, we construct an algorithm $\mathcal{B}$ that can factor N. Specifically, $\mathcal{B}$ on input N proceeds as follows:

1. Sample $v \leftarrow \mathbb{QR}_N$ (such a v can be obtained by sampling a random value in $\mathbb{Z}_N^*$ and squaring it) and compute $g := v^2 \mod N$.
2. Sample $x, y \leftarrow [N]$.[7]
3. Let $u := \mathcal{A}(g, g^x \cdot v, g^y \cdot v)$[8] and compute $w := \frac{u}{g^{xy} \cdot v^{x+y}}$.

[3] Since the number of primes is infinite we can define an infinite family of such groups. For the sake of simplicity, here we only consider a single group.

[4] Let p be an odd prime number. An integer a is said to be a *quadratic residue* modulo p if it is congruent to a perfect square modulo p and is said to be a *quadratic non-residue* modulo p otherwise. The *Legendre symbol* is a function of a and p defined as

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a \text{ is quadration residue mod } p \text{ and } a \not\equiv 0 \\ -1 & \text{if } a \text{ is quadration non-residue mod } p \\ 0 & \text{if } a \equiv 0 \mod p \end{cases}$$

Legendre symbol can be efficiently computed as $\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \mod p$.

[5] This is because given g^x, g^y one can easily compute deduce the Legendre symbol of g^{xy}. Observe that if $\left(\frac{g}{p}\right) = -1$ then we have that $\left(\frac{g^{xy}}{p}\right) = 1$ if and only if $\left(\frac{g^x}{p}\right) = 1$ or $\left(\frac{g^y}{p}\right) = 1$. Using this fact, we can construct an adversary that breaks the DDH problem with a non-negligible (in fact, noticeable) probability.

[6] By Dirichet's Theorem on primes in arithmetic progression, we have that there are infinite choices of primes (p, q) for which $p = 2q + 1$. This allows us to generalize this group to a group ensemble.

[7] Note that sampling x, y uniformly from $[N]$ is statistically close to sampling x, y uniformly from $[\phi(N)]$.

[8] Note that $g^x \cdot v$ where $x \leftarrow [N]$ is statistically close to g^x where $x \leftarrow [N]$.

4. If $w^2 = v^2 \mod N$ and $u \neq \pm v$, then compute the factors of N as $\mathsf{gcd}(N, u+v)$ and $N/\mathsf{gcd}(N, u+v)$. Otherwise, output $\perp$.

Observe that if $\mathcal{A}$ solves the CDH problem then the returned values $u = g^{(x+2^{-1})(y+2^{-1})} = v^{2xy+x+y+2^{-1}}$. Consequently, the computed value $w = v^{2^{-1}}$. Furthermore, with probability $\frac{1}{2}$ we have that $w \neq v$. In this case, $\mathcal{B}$ can factor N. $\square$

2
One-Way Functions

Cryptographers often attempt to base cryptographic results on conjectured computational assumptions to leverage reduced adversarial capabilities. Furthermore, the security of these constructions is no better than the assumptions they are based on.

> *Cryptographers seldom sleep well.*[1]

[1] Quote by Silvio Micali in personal communication with Joe Kilian.

Thus, basing cryptographic tasks on the *minimal* necessary assumptions is a key tenet in cryptography. Towards this goal, rather can making assumptions about specific computational problem in number theory, cryptographers often consider *abstract primitives*. The existence of these abstract primitives can then be based on one or more computational problems in number theory.

The weakest abstract primitive cryptographers consider is one-way functions. Virtually, every cryptographic goal of interest is known to imply the existence of one-way functions. In other words, most cryptographic tasks would be impossible if the existence of one-way functions was ruled out. On the flip side, the realizing cryptographic tasks from just one-way functions would be ideal.

2.1 *Definition*

A one-way function $f : \{0,1\}^n \to \{0,1\}^m$ is a function that is easy to compute but hard to invert. This intuitive notion is trickier to formalize than it might appear on first thought.

Definition 2.1 (One-Way Functions)**.** *A function $f : \{0,1\}^* \to \{0,1\}^*$ is said to be one-way function if:*

- ***Easy to Compute:*** *$\exists$ a (deterministic) polynomial time machine M such that $\forall x \in \{0,1\}^*$ we have that*

$$M(x) = f(x)$$

- ***Hard to Invert:*** *$\forall$ non-uniform PPT adversary $\mathcal{A}$ we have that*

$$\mu_{\mathcal{A},f}(n) = \Pr_{x \xleftarrow{\$} \{0,1\}^n} [\mathcal{A}(1^n, f(x)) \in f^{-1}(f(x))] \tag{2.1}$$

is a negligible function, $x \xleftarrow{\$} \{0,1\}^n$ denotes that x is drawn uniformly at random from the set $\{0,1\}^n$, $f^{-1}(f(x)) = \{x' \mid f(x) = f(x')\}$, and the probability is over the random choices of x and the random coins of $\mathcal{A}$[2].

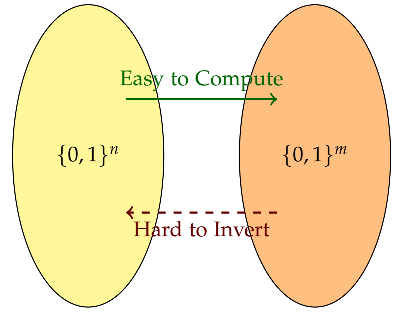

Figure 2.1: Visulizing One-way Funcations

We note that the function is not necessarily one-to-one. In other words, it is possible that $f(x) = f(x')$ for $x \neq x'$ – and the adversary is allowed to output any such x'.

The above definition is rather delicate. We next describe problems in the slight variants of this definition that are insecure.

[2] Typically, the probability is only taken over the random choices of x, since we can fix the random coins of the adversary $\mathcal{A}$ that maximize its advantage.

1. What if we require that $\Pr_{x \xleftarrow{\$} \{0,1\}^n}[\mathcal{A}(1^n, f(x)) \in f^{-1}(f(x))] = 0$ instead of being negligible?

   This condition is false for every function f. An adversary $\mathcal{A}$ that outputs an arbitrarily fixed value x_0 succeeds with probability at least $1/2^n$, as $x_0 = x$ with at least the same probability.

2. What if we drop the input 1^n to $\mathcal{A}$ in Equation 2.1?

   Consider the function $f(x) = |x|$. In this case, we have that $m = \log_2 n$, or $n = 2^m$. Intuitively, f should not be considered a one-way function, because it is easy to invert f. Namely, given a value y any x such that $|x| = y$ is such that $x \in f^{-1}(y)$. However, according to this definition the adversary gets an m bit string as input, and hence is restricted to running in time polynomial in m. Since each possible x is of size $n = 2^m$, the adversary doesn't even have enough time to write down the answer! Thus, according to the flawed definition above, f would be a one-way function.

   Providing the attacker with 1^n (n repetitions of the 1 bit) as additional input avoids this issue. In particular, it allows the attacker to run in time polynomial in m and n.

Candidate One-way Functions. It is not known whether one-way functions exist. In fact, the existence of one-way functions would imply that $P \neq NP$ (see Exercise 2.3).

However, there are candidates of functions that could be one-way functions, based on the difficulty of certain computational problems. (See Section 1.3)

Let's suppose that the discrete-log assumption hold for group ensemble $\mathcal{G} = \{\mathbb{G}_n\}$ then we have that the function family $\{f_n\}$ where $f_n : \{1, \ldots |\mathbb{G}_n|\} \to \mathbb{G}_n$ is a one-way function family. In particular, $f_n(x) = g^x$ where g is the generator of the group $\mathbb{G}_n$. The proof that $\{f_n\}$ is one-way based on the Discrete-Log Assumption (see Definition 1.5) is left as as an exercise.

2.2 *Robustness and Brittleness of One-way Functions*

What operations can we perform on one-way functions and still have a one-way function? In this section, we explore the robustness and brittleness of one-way functions and some operations that are safe or unsafe to perform on them.

2.2.1 *Robustness*

Consider having a one-way function f. Can we use this function f in order to make a more structured one-way function g such that $g(x_0) = y_0$ for some constants x_0, y_0, or would this make the function no longer be one-way?

Intuitively, the answer is yes - we can specifically set $g(x_0) = y_0$, and otherwise have $g(x) = f(x)$. In this case, the adversary gains the knowledge of how to invert y_0, but that will only happen with negligible probability, and so the function is still one-way.

In fact, this can be done for an exponential number of x_0, y_0 pairs. To illustrate that, consider the following function:

$$g(x_1 \| x_2) = \begin{cases} x_1 \| x_2 & : x_1 = 0^{n/2} \\ f(x_1 \| x_2) & : \text{otherwise} \end{cases}$$

However, this raises an apparent contradiction - according to this theorem, given a one-way function f, we could keep fixing each of its values to 0, and it would continue to be a one-way function. If we kept doing this, we would eventually end up with a function which outputs 0 for *all* of the possible values of x. How could this still be one-way?

The resolution of this apparent paradox is by noticing that a one-way function is only required to be one-way in the limit where n grows very large. So, no matter how many times we fix the values of f to be 0, we are still only setting a finite number of x values to 0. However, this will still satisfy the definition of a one-way function

- it is just that we will have to use larger and larger values of n_0 in order to prove that the probability of breaking the one-way function is negligible.

2.2.2 Brittleness

Example: OWFs do not always compose securely. Given a one-way function $f : \{0,1\}^n \to \{0,1\}^n$, is the function $f^2(x) = f(f(x))$ also a one-way function? Intuitively, it seems that if it is hard to invert $f(x)$, then it would be just as hard to invert $f(f(x))$. However, this intuition is incorrect and highlights the delicacy when working with cryptographic assumptions and primitives. In particular, assuming one-way functions exists we describe a one-way function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ such that f^2 can be efficiently inverted. Let $g : \{0,1\}^n \to \{0,1\}^n$ be a one-way function then we set f as follows:

$$f(x_1, x_2) = 0^n \| g(x_1)$$

Two observations follow:

1. f^2 is not one-way. This follows from the fact that for all inputs x_1, x_2 we have that $f^2(x_1, x_2) = 0^{2n}$. This function is clearly not one-way!

2. f is one-way. This can be argued as follows. Assume that there exists an adversary $\mathcal{A}$ such that $\mu_{\mathcal{A},f}(n) = \Pr_{x \xleftarrow{\$} \{0,1\}^n}[\mathcal{A}(1^{2n}, f(x)) \in f^{-1}(f(x))]$ is non-negligible. Using such an $\mathcal{A}$ we will describe a construction of adversary $\mathcal{B}$ such that $\mu_{\mathcal{B},g}(n) = \Pr_{x \xleftarrow{\$} \{0,1\}^n}[\mathcal{B}(1^n, g(x)) \in g^{-1}(g(x))]$ is also non-negligible. This would be a contradiction thus proving our claim.

   Description of $\mathcal{B}$: $\mathcal{B}$ on input $y \in \{0,1\}^n$ outputs the n lower-order bits of $\mathcal{A}(1^{2n}, 0^n \| y)$.

   Observe that if $\mathcal{A}$ successfully inverts f then we have that $\mathcal{B}$ successfully inverts g. More formally, we have that:

   $$\mu_{\mathcal{B},g}(n) = \Pr_{x \xleftarrow{\$} \{0,1\}^n} \left[\mathcal{A}(1^{2n}, 0^n || g(x)) \in \{0,1\}^n || g^{-1}(g(x)) \right].$$

   But

   $$\begin{aligned} \mu_{\mathcal{A},f}(2n) = & \Pr_{x_1, x_2 \xleftarrow{\$} \{0,1\}^{2n}} [\mathcal{A}(1^{2n}, f(x_1, x_2)) \in f^{-1}(f(\tilde{x}))] \\ = & \Pr_{x_1 \xleftarrow{\$} \{0,1\}^n} [\mathcal{A}(1^{2n}, 0^n || g(x_2)) \in \{0,1\}^n || g^{-1}(g(x_2))] \\ = & \mu_{\mathcal{B},g}(n). \end{aligned}$$

   Hence, we have that $\mu_{\mathcal{B},g}(n) = \mu_{\mathcal{A},f}(2n)$ which is non-negligible as long as $\mu_{\mathcal{A},f}(2n)$ is non-negligible.

Example: Dropping a bit is not always secure. Below is another example of a transformation that does not work. Given any one-way function g, let $g'(x)$ be $g(x)$ with the first bit omitted.

Claim 2.1. *g' is not necessarily one-way. In other words, there exists a OWF function g for which g' is not one-way.*

Proof. We must (1) construct a function g, (2) show that g is one-way, and (3) show that g' is not one-way.
Step 1: Construct a OWF g. To do this, we first want to come up with a (contrived) function g and prove that it is one-way. Let us assume that there exists a one-way function $h : \{0,1\}^n \to \{0,1\}^n$. We define the function $g : \{0,1\}^{2n} \to \{0,1\}^{2n}$ as follows:

$$g(x\|y) = \begin{cases} 0^n\|y & \text{if } x = 0^n \\ 1\|0^{n-1}\|g(y) & \text{otherwise} \end{cases}$$

Step 2: Prove that g is one-way.

Claim 2.2. *If h is a one-way function, then so is g.*

Proof. Assume for the sake of contradiction that g is not one-way. Then there exists a polynomial time adversary $\mathcal{A}$ and a non-negligible function $\mu(\cdot)$ such that:

$$\Pr_{x,y}[\mathcal{A}(1^n, g(x\|y)) \in g^{-1}(g(x\|y))] = \mu(n)$$

We will use such an adversary $\mathcal{A}$ to invert h with some non-negligible probability. This contradicts the one-wayness of h and thus our assumption that g is not one-way function is false.

Let us now construct an $\mathcal{B}$ that uses $\mathcal{A}$ and inverts h. $\mathcal{B}$ is given $1^n, h(y)$ for a randomly chosen y and its goal is to output $y' \in h^{-1}(h(y))$ with some non-negligible probability. $\mathcal{B}$ works as follows:

1. It samples $x \leftarrow \{0,1\}^n$ randomly.
2. If $x = 0^n$, it samples a random $y' \leftarrow \{0,1\}^n$ and outputs it.
3. Otherwise, it runs $\mathcal{A}(10^{n-1}\|h(y))$ and obtains $x'\|y'$. It outputs y'.

Let us first analyze the running time of $\mathcal{B}$. The first two steps are clearly polynomial (in n) time. In the third step, $\mathcal{B}$ runs $\mathcal{A}$ and uses its output. Note that the running time of since $\mathcal{A}$ runs in polynomial (in n) time, this step also takes polynomial (in n) time. Thus, the overall running time of $\mathcal{B}$ is polynomial (in n).

Let us now calculate the probability that $\mathcal{B}$ outputs the correct inverse. If $x = 0^n$, the probability that y' is the correct inverse is at least $\frac{1}{2^n}$ (because it guesses y' randomly and probability that a random y' is the correct inverse is $\geq 1/2^n$). On the other hand, if $x \neq 0^n$, then the probability that $\mathcal{B}$ outputs the correct inverse is $\mu(n)$. Thus,

$$\begin{aligned}
\Pr[\mathcal{B}(1^n, h(y)) \in h^{-1}(h(y))] &\geq \Pr[x = 0^n](\frac{1}{2^n}) + \Pr[x \neq 0^n]\mu(n) \\
&= \frac{1}{2^{2n}} + (1 - \frac{1}{2^n})\mu(n) \\
&\geq \mu(n) - (\frac{1}{2^n} - \frac{1}{2^{2n}})
\end{aligned}$$

Since $\mu(n)$ is a non-negligible function and $(\frac{1}{2^n} - \frac{1}{2^{2n}})$ is a negligible function, their difference is non-negligible.[3] This contradicts the one-wayness of h.

[3] Exercise: Prove that if $\alpha(\cdot)$ is a non-negligible function and $\beta(\cdot)$ is a negligible function, then $(\alpha - \beta)(\cdot)$ is a non-negligible function.

$\square$

Step 3: Prove that g' is not one-way. We construct the new function $g' : \{0,1\}^{2n} \to \{0,1\}^{2n-1}$ by dropping the first bit of g. That is,

$$g'(x\|y) = \begin{cases} 0^{n-1}\|y & \text{if } x = 0^n \\ 0^{n-1}\|g(y) & \text{otherwise} \end{cases}$$

We now want to prove that g' is not one-way. That is, we want to design an adversary $\mathcal{C}$ such that given 1^{2n} and $g'(x\|y)$ for a randomly chosen x, y, it outputs an element in the set $g^{-1}(g(x\|y)$. The description of $\mathcal{C}$ is as follows:

- On input 1^{2n} and $g'(x\|y)$, the adversary $\mathcal{C}$ parses $g'(x\|y)$ as $0^{n-1}\|\overline{y}$.
- It outputs $0^n\|\overline{y}$ as the inverse.

Notice that $g'(0^n\|\overline{y}) = 0^{n-1}\|\overline{y}$. Thus, $\mathcal{C}$ succeeds with probability 1 and this breaks the one-wayness of g'.

$\square$

2.3 *Hardness Amplification*

In this section, we show that even a very *weak* form of one-way functions suffices from constructing one-way functions as defined previously. For this section, we refer to this previously defined notion as strong one-way functions.

Definition 2.2 (Weak One-Way Functions). *A function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ is said to be a weak one-way function if:*

- *f is computable by a polynomial time machine, and*
- *There exists a noticeable function $\alpha_f(\cdot)$ such that $\forall$ non-uniform PPT adversaries $\mathcal{A}$ we have that*

$$\mu_{\mathcal{A},f}(n) = \Pr_{x \xleftarrow{\$} \{0,1\}^n}[\mathcal{A}(1^n, f(x)) \in f^{-1}(f(x))] \leq 1 - \alpha_f(n).$$

Theorem 2.1. *If there exists a weak one-way function, then there exists a (strong) one-way function.*

Proof. We prove the above theorem constructively. Suppose $f : \{0,1\}^n \rightarrow \{0,1\}^m$ is a weak one-way function, then we prove that the function $g : \{0,1\}^{nq} \rightarrow \{0,1\}^{mq}$ for $q = \lceil \frac{2n}{\alpha_f(n)} \rceil$ where

$$g(x_1, x_2, \cdots, x_q) = f(x_1)||f(x_2)|| \cdots ||f(x_q),$$

is a strong one-way function. Let us discuss the intuition. A weak one-way function is "strong" in a small part of its domain. For this construction to result in a strong one-way function, we need just one of the q instantiations to be in the part of the domain where our weak one-way function is strong. If we pick a large enough q, this is guaranteed to happen.

Assume for the sake of contradiction that there exists an adversary $\mathcal{B}$ such that $\mu_{\mathcal{B},g}(nq) = \Pr_{x \xleftarrow{\$} \{0,1\}^{nq}}[\mathcal{B}(1^{nq}, g(x)) \in g^{-1}(g(x))]$ is non-negligible. Then we use $\mathcal{B}$ to construct $\mathcal{A}$ (see Figure 2.2) that breaks f, namely $\mu_{\mathcal{A},f}(n) = \Pr_{x \xleftarrow{\$} \{0,1\}^n}[\mathcal{A}(1^n, f(x)) \in f^{-1}(f(x))] > 1 - \alpha_f(n)$ for sufficiently large n.

1. $i \xleftarrow{\$} [q]$.
2. $x_1, \cdots, x_{i-1}, x_i, \cdots, x_q \xleftarrow{\$} \{0,1\}^n$.
3. Set $y_j = f(x_j)$ for each $j \in [q] \backslash \{i\}$ and $y_i = y$.
4. $(x'_1, x'_2, \cdots, x'_q) := \mathcal{B}(f(x_1), f(x_2), \cdots, f(x_q))$.
5. $f(x'_i) = y$ then output x'_i else $\perp$.

Figure 2.2: Construction of $\mathcal{A}(1^n, y)$

Note that: (1) $\mathcal{A}(1^n, y)$ iterates at most $T = \frac{4n^2}{\alpha_f(n)\mu_{\mathcal{B},g}(nq)}$ times each call is polynomial time. (2) $\mu_{\mathcal{B},g}(nq)$ is a non-negligible function. This implies that for infinite choices of n this value is greater than some noticeable function. Together these two facts imply that for infinite choices of n the running time of $\mathcal{A}$ is bounded by a polynomial function in n.

It remains to show that $\Pr_{x \xleftarrow{\$} \{0,1\}^n}[\mathcal{A}(1^n, f(x)) = \perp] < \alpha_f(n)$ for arbitrarily large n. A natural way to argue this is by showing that at least one execution of $\mathcal{B}$ should suffice for inverting $f(x)$. However, the technical challenge in proving this formally is that these calls to $\mathcal{B}$ aren't independent. Below we formalize this argument even when these calls aren't independent.

Define the set S of "bad" x's, which are hard to invert:

$$S := \left\{ x \,\middle|\, \Pr_{\mathcal{B}}[\mathcal{A} \text{ inverts } f(x) \text{ in a single iteration}] \leq \frac{\alpha_f(n)\mu_{\mathcal{B},g}(nq)}{4n} \right\}.$$

Lemma 2.1. *Let A be any an efficient algorithm such that $\Pr_{x,r}[A(x,r) = 1] \geq \epsilon$. Additionally, let $G = \{x \mid \geq \Pr_r[A(x,r) = 1] \geq \frac{\epsilon}{2}\}$. Then, we have $\Pr_x[x \in G] \geq \frac{\epsilon}{2}$.*

Proof. The proof of this lemma follows by a very simple counting argument. Let's start by assuming that $\Pr_x[x \in G] < \frac{\epsilon}{2}$. Next, observe that

$$\begin{aligned}
&\Pr_{x,r}[A(x,r) = 1] \\
&\quad = \Pr_x[x \in G] \cdot \Pr_{x,r}[A(x,r) = 1 \mid x \in G] \\
&\quad + \Pr_x[x \notin G] \cdot \Pr_{x,r}[A(x,r) = 1 \mid x \notin G] \\
&\quad < \frac{\epsilon}{2} \cdot 1 + 1 \cdot \frac{\epsilon}{2} \\
&\quad < \epsilon,
\end{aligned}$$

which is a contradiction. □

We start by proving that the size of S is small. More formally,

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n}[x \in S] \le \frac{\alpha_f(n)}{2}.$$

Assume, for the sake of contradiction, that $\Pr_{x \xleftarrow{\$} \{0,1\}^n}[x \in S] > \frac{\alpha_f(n)}{2}$. Then we have that:

$$\begin{aligned}
\mu_{\mathcal{B},g}(nq) = & \Pr_{(x_1,\cdots,x_q) \xleftarrow{\$} \{0,1\}^{nq}}[\mathcal{B}(1^{nq}, g(x_1,\cdots,x_q)) \in g^{-1}(g(x_1,\cdots,x_q))] \\
= & \Pr_{x_1,\cdots,x_q}[\mathcal{B}(1^{nq}, g(x_1,\cdots,x_q)) \in g^{-1}(g(x_1,\cdots,x_q)) \wedge \forall i : x_i \notin S] \\
& + \Pr_{x_1,\cdots,x_q}[\mathcal{B}(1^{nq}, g(x_1,\cdots,x_q)) \in g^{-1}(g(x_1,\cdots,x_q)) \wedge \exists i : x_i \in S] \\
\le & \Pr_{x_1,\cdots,x_q}[\forall i : x_i \notin S] + \sum_{i=1}^{q} \Pr_{x_1,\cdots,x_q}[\mathcal{B}(1^{nq}, g(x_1,\cdots,x_q)) \in g^{-1}(g(x_1,\cdots,x_q)) \wedge x_i \in S] \\
\le & \left(1 - \frac{\alpha_f(n)}{2}\right)^q + q \cdot \Pr_{x_1,\cdots,x_q,i}[\mathcal{B}(1^{nq}, g(x_1,\cdots,x_q)) \in g^{-1}(g(x_1,\cdots,x_q)) \wedge x_i \in S] \\
= & \left(1 - \frac{\alpha_f(n)}{2}\right)^{\frac{2n}{\alpha_f(n)}} + q \cdot \Pr_{x \xleftarrow{\$} \{0,1\}^n, \mathcal{B}}[\mathcal{A} \text{ inverts } f(x) \text{ in a single iteration} \wedge x \in S] \\
\le & e^{-n} + q \cdot \Pr_x[x \in S] \cdot \Pr[\mathcal{A} \text{ inverts } f(x) \text{ in a single iteration} \mid x \in S] \\
\le & e^{-n} + \frac{2n}{\alpha_f(n)} \cdot 1 \cdot \frac{\mu_{\mathcal{B},g}(nq) \cdot \alpha_f(n)}{4n} \\
\le & e^{-n} + \frac{\mu_{\mathcal{B},g}(nq)}{2}.
\end{aligned}$$

Lemma 2.2. *Let A be any an efficient algorithm such that $\Pr_{x,r}[A(x_1,\dots x_n, r) = 1] \ge \epsilon$. Additionally, let $G = \{x \mid \ge \Pr_{x_1,\dots,x_n,r}[A(x,r) = 1 \mid \exists i, x = x_i] \ge \frac{\epsilon}{2}\}$. Then, we have $\Pr_x[x \in G] \ge \frac{\epsilon}{2}$.*

Proof. The proof of this lemma follows by a very simple counting argument. Let's start by assuming that $\Pr_x[x \in G] < \frac{\epsilon}{2}$. Next, observe that

$$\begin{aligned}
&\Pr_{x,r}[A(x,r) = 1] \\
&= \Pr_x[x \in G] \cdot \Pr_{x,r}[A(x,r) = 1 \mid x \in G] \\
&+ \Pr[x \notin G] \cdot \Pr[A(x,r) = 1 \mid x \notin G] \\
&< \frac{\epsilon}{2} \cdot 1 + 1 \cdot \frac{\epsilon}{2} \\
&< \epsilon,
\end{aligned}$$

which is a contradiction. □

Hence $\mu_{\mathcal{B},g}(nq) \le 2e^{-n}$, contradicting with the fact that $\mu_{\mathcal{B},g}$ is non-negligible. Then we have

$$\begin{aligned}
& \Pr_{x \xleftarrow{\$} \{0,1\}^n}[\mathcal{A}(1^n, f(x)) = \perp] \\
& = \Pr_x[x \in S] + \Pr_x[x \notin S] \cdot \Pr[\mathcal{B} \text{ fails to invert } f(x) \text{ in every iteration} | x \notin S] \\
& \le \frac{\alpha_f(n)}{2} + (\Pr[\mathcal{B} \text{ fails to invert } f(x) \text{ a single iteration} | x \notin S])^T \\
& \le \frac{\alpha_f(n)}{2} + \left(1 - \frac{\mu_{\mathcal{A},g}(nq) \cdot \alpha_f(n)}{4n}\right)^T \\
& \le \frac{\alpha_f(n)}{2} + e^{-n} \le \alpha_f(n)
\end{aligned}$$

for sufficiently large n. This concludes the proof. □

2.4 *Levin's One-Way Function*

In this section, we discuss Levin's one-way function, which is an explicit construction of a one-way function that is secure as long as a one-way function exists. This is interesting because unlike a typical cryptographic primitive that relies on a specific hardness assumption (which may or may not hold in the future), Levin's one-way function is future-proof in the sense that it will be secure as long as atleast one hardness assumption holds (which we may or may not discover).

The high-level intuition behind Levin's construction is as follows: since we assume one-way functions exist, there exists a uniform machine $\tilde{M}$ such that $|\tilde{M}|$ is a constant and $\tilde{M}(x)$ is hard to invert for a random input x. Now, consider a function h that parses the first $\log(n)$ bits of its n-bit input as the code of a machine M and the remaining bits as the input to M. For a large enough n that is exponential in $|\tilde{M}|$, note that we will hit the code of $\tilde{M}$ with noticeable probability in n, and for those instances, h will be hard to invert. It is easy to see that this gives us a weak one-way function which has a noticeable probability of being hard to invert, and we can amplify the hardness of this weak one-way function to get an explicit construction of a one-way function.

Theorem 2.2. *If there exists a one-way function, then there exists an explicit function f that is one-way (constructively).*

Before we look at the construction and the proof in detail, we first prove a lemma that will be useful in the proof. In particular, we need a bound on the running time of the one-way function $\tilde{M}$ so that we can upper bound the execution time of h, since there could be inputs to g that do not terminate in polynomial time. To this end, we prove the following lemma which shows that if a one-way function exists, then there is also a one-way function that runs in time n^2, and thus, we can bound h to n^2 steps.

Lemma 2.3. *If there exists a one-way function computable in time n^c for a constant c, then there exists a one-way function computable in time n^2.*

Proof. Let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way function computable in time n^c. Construct $g : \{0,1\}^{n+n^c} \to \{0,1\}^{n+n^c}$ as follows:

$$g(x,y) = f(x)||y$$

where $x \in \{0,1\}^n, y \in \{0,1\}^{n^c}$. $g(x,y)$ takes time $2n^c$, which is linear in the input length.

We next show that $g(\cdot)$ is one-way. Assume for the purpose of contradiction that there exists an adversary $\mathcal{A}$ such that $\mu_{\mathcal{A},g}(n + n^c) = \Pr_{(x,y)\xleftarrow{\$}\{0,1\}^{n+n^c}}[\mathcal{A}(1^{n+n^c}, g(x,y)) \in g^{-1}(g(x,y))]$ is non-negligible. Then we use $\mathcal{A}$ to construct $\mathcal{B}$ such that $\mu_{\mathcal{B},f}(n) = \Pr_{x\xleftarrow{\$}\{0,1\}^n}[\mathcal{B}(1^n, f(x)) \in f^{-1}(f(x))]$ is also non-negligible.

$\mathcal{B}$ on input $z \in \{0,1\}^n$, samples $y \xleftarrow{\$} \{0,1\}^{n^c}$, and outputs the n higher-order bits of $\mathcal{A}(1^{n+n^c}, z||y)$. Then we have

$$\begin{aligned}\mu_{\mathcal{B},g}(n) = & \Pr_{x \xleftarrow{\$} \{0,1\}^n, y \xleftarrow{\$} \{0,1\}^{n^c}} \left[\mathcal{A}(1^{n+n^c}, f(x)||y) \in f^{-1}(f(x))||\{0,1\}^{n^c}\right] \\ \geq & \Pr_{x,y}\left[\mathcal{A}(1^{n+n^c}, g(x,y)) \in f^{-1}(f(x))||y\right] \\ = & \Pr_{x,y}\left[\mathcal{A}(1^{n+n^c}, g(x,y)) \in g^{-1}(g(x,y))\right]\end{aligned}$$

is non-negligible. □

Now, we provide the explicit construction of h and prove that it is a weak one-way function. Since h is an (explicit) weak one-way function, we can construct an (explicit) one-way function from h as we discussed in Section 2.3, and this would prove Theorem 2.2.

Proof of Theorem 2.2. $h : \{0,1\}^n \to \{0,1\}^n$ is defined as follows:

$$h(M,x) = \begin{cases} M||M(x) & \text{if } M(x) \text{ takes no more than } |x|^2 \text{ steps} \\ M||0 & \text{otherwise} \end{cases}$$

where $|M| = \log n$, $|x| = n - \log n$ (interpreting M as the code of a machine and x as its input).

It remains to show that if one-way functions exist, then h is a weak one-way function, with $\alpha_h(n) = \frac{1}{n^2}$. Assume for the purpose of contradiction that there exists an adversary $\mathcal{A}$ such that $\mu_{\mathcal{A},h}(n) = \Pr_{(M,x) \xleftarrow{\$} \{0,1\}^n}[\mathcal{A}(1^n, h(M,x)) \in h^{-1}(h(M,x))] \geq 1 - \frac{1}{n^2}$ for all sufficiently large n. By the existence of one-way functions and Lemma 2.3, there exists a one-way function $\tilde{M}$ that can be computed in time n^2. Let $\tilde{M}$ be the uniform machine that computes this one-way function. We will consider values n such that $n > 2^{|\tilde{M}|}$. In other words for these choices of n, $\tilde{M}$ can be described using $\log n$ bits. We construct $\mathcal{B}$ to invert $\tilde{M}$: on input y outputs the $(n - \log n)$ lower-order bits of $\mathcal{A}(1^n, \tilde{M}||y)$. Then

$$\begin{aligned}\mu_{\mathcal{B},\tilde{M}}(n-\log n) = & \Pr_{x \xleftarrow{\$} \{0,1\}^{n-\log n}} \left[\mathcal{A}(1^n, \tilde{M}||\tilde{M}(x)) \in \{0,1\}^{\log n}||\tilde{M}^{-1}(\tilde{M}((x))\right] \\ \geq & \Pr_{x \xleftarrow{\$} \{0,1\}^{n-\log n}} \left[\mathcal{A}(1^n, \tilde{M}||\tilde{M}(x)) \in \tilde{M}||\tilde{M}^{-1}(\tilde{M}((x))\right].\end{aligned}$$

Observe that for sufficiently large n it holds that

$$\begin{aligned}1 - \frac{1}{n^2} \leq & \mu_{\mathcal{A},h}(n) \\ = & \Pr_{(M,x) \xleftarrow{\$} \{0,1\}^n} \left[\mathcal{A}(1^n, h(M,x)) \in h^{-1}(h(M,x))\right] \\ \leq & \Pr_M[M = \tilde{M}] \cdot \Pr_x\left[\mathcal{A}(1^n, \tilde{M}||\tilde{M}(x)) \in \tilde{M}||\tilde{M}^{-1}(\tilde{M}((x))\right] + \Pr_M[M \neq \tilde{M}] \\ \leq & \frac{1}{n} \cdot \mu_{\mathcal{B},\tilde{M}}(n - \log n) + \frac{n-1}{n}.\end{aligned}$$

Hence $\mu_{\mathcal{B},\tilde{M}}(n - \log n) \geq \frac{n-1}{n}$ for sufficiently large n which is a contradiction. $\square$

2.5 *Hardness Concentrate Bit*

We start by asking the following question: Is it possible to concentrate the strength of a one-way function into one bit? In particular, given a one-way function f, does there exist one bit that can be computed efficiently from the input x, but is hard to compute given $f(x)$?

Definition 2.3 (Hard Concentrate Bit). *Let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way function. $B : \{0,1\}^n \to \{0,1\}$ is a hard concentrate bit of f if:*

- *B is computable by a polynomial time machine, and*
- *$\forall$ non-uniform PPT adversaries $\mathcal{A}$ we have that*

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n} [\mathcal{A}(1^n, f(x)) = B(x)] \leq \frac{1}{2} + \mathsf{negl}(n).$$

A simple example. Let f be a one-way function. Consider the one-way function $g(b,x) = 0||f(x)$ and a hard concentrate bit $B(b,x) = b$. Intuitively, the value $g(b,x)$ does not reveal any information about the first bit b, thus no information about the value $B(b,x)$ can be ascertained. Hence $\mathcal{A}$ cannot predict the first bit with a non-negligible advantage than a random guess. However, we are more interested in the case where the hard concentrate bit is hidden because of computational hardness and not information theoretic hardness.

Remark 2.1. *Given a one-way function f, we can construct another one-way function g with a hard concentrate bit. However, we may not be able to find a hard concentrate bit for f. In fact, it is an open question whether a hard concentrate bit exists for every one-way function.*

Intuitively, if a function f is one-way, it seems that there should be a particular bit in the input x that is hard to compute given $f(x)$. However, we show that is not true:

Claim 2.3. *If $f : \{0,1\}^n \to \{0,1\}^n$ is a one-way function, then there exists a one-way function $g : \{0,1\}^{n+\log n} \to \{0,1\}^{n+\log n}$ such that $\forall i \in [1, n + \log n]$, $B_i(x) = x_i$ is not a hard concentrate bit, where x_i is the i^{th} bit of x.*

Proof. Define $g : \{0,1\}^{n+\log(n)} \to \{0,1\}^{n+\log(n)}$ as follows.

$$g(x,y) = f(x_{\bar{y}})||x_y||y,$$

where $|x| = n$, $|y| = \log n$, $x_{\bar{y}}$ is all bits of x except the y^{th} bit, and x_y is the y^{th} bit of x.

First, one can show that g is still a one-way function. (We leave this as an exercise!) Next, we show that B_i is not a hard concentrate bit for $\forall i \in [1, n]$ (clearly B_i is not a hard concentrate bit for $i \in [n+1, n+\log n]$). Construct an adversary $\mathcal{A}_i(1^{n+\log n}, f(x_{\bar{y}})||x_y||y)$ that "breaks" B_i:

- If $y \neq i$ then output a random bit;
- Otherwise output x_y.

$$\begin{aligned}
&\Pr_{x,y}[\mathcal{A}(1^{n+\log n}, g(x,y)) = B_i(x)] \\
=&\Pr_{x,y}[\mathcal{A}(1^{n+\log n}, f(x_{\bar{y}})||x_y||y) = x_i] \\
=&\frac{n-1}{n} \cdot \frac{1}{2} + \frac{1}{n} \cdot 1 = \frac{1}{2} + \frac{1}{2n}.
\end{aligned}$$

Hence $\mathcal{A}_i$ can guess the output of B_i with greater than $\frac{1}{2} + \mathsf{negl}(n)$ probability. $\square$

2.5.1 *Hard Concentrate Bit of any One-Way Permutation*

We now show that a slight modification of every one-way function has a hard concentrate bit. More formally,

Theorem 2.3. *Let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way function. Define a function $g : \{0,1\}^{2n} \to \{0,1\}^{2n}$ as follows:*

$$g(x,r) = f(x)||r,$$

where $|x| = |r| = n$. Then we have that g is one-way and that it has a hard concentrate bit, namely $B(x,r) = \sum_{i=1}^{n} x_i r_i \mod 2$.

Remark 2.2. *If f is a (one-to-one) one-way function, then g is also a (one-to-one) one-way function with hard concentrate bit $B(\cdot)$.*

Proof. We leave it as an exercise to show that g is a one-way function and below we will prove that the function $B(\cdot)$ describe a hard concentrate bit of g. More specifically, we need to show that if there exists a non-uniform PPT $\mathcal{A}$ s.t. $\Pr_{x,r}[\mathcal{A}(1^{2n}, g(x,r)) = B(x,r)] \geq \frac{1}{2} + \epsilon(n)$, where ϵ is non-negligible, then there exists a non-uniform PPT $\mathcal{B}$ such that $\Pr_{x,r}[\mathcal{B}(1^{2n}, g(x,r)) \in g^{-1}(g(x,r))]$ is non-negligible. Below we use E to denote the event that $\mathcal{A}(1^{2n}, g(x,r)) = B(x,r)$. We will present our proof in three steps, where each step progressively increases in complexity: (1) the super simple case where we restrict to $\mathcal{A}$ such that $\Pr_{x,r}[E] = 1$, (2) the simple case where we restrict to $\mathcal{A}$ such that $\Pr_{x,r}[E] \geq \frac{3}{4} + \epsilon(n)$, and finally (3) the general case where $\Pr_{x,r}[E] \geq \frac{1}{2} + \epsilon(n)$.

Super simple case. Suppose $\mathcal{A}$ guesses $B(\cdot)$ with perfect accuracy:

$$\Pr_{x,r}[E] = 1.$$

We now construct $\mathcal{B}$ that inverts g with perfect accuracy. Let e^i denote the one-hot n-bit string $0 \cdots 010 \cdots 0$, where only the i-th bit is 1, the rest are all 0. $\mathcal{B}$ gets $f(x)||r$ as input, and its algorithm is described in Figure 2.3.

for $i = 1$ **to** n **do**
  $x'_i \leftarrow \mathcal{A}(1^{2n}, f(x)||e^i)$
end for
return $x'_1 \cdots x'_n||r$

Figure 2.3: Super-Simple Case $\mathcal{B}$

Observe that $B(x, e^i) = \sum_{j=1}^n x_j e^i_j = x_i$. Therefore, the probability that $\mathcal{B}$ inverts a single bit successfully is,

$$\Pr_x\left[\mathcal{A}(1^{2n}, f(x)||e^i) = x_i\right] = \Pr_x\left[\mathcal{A}(1^{2n}, f(x)||e^i) = B(x, e^i)\right] = 1.$$

Hence $\Pr_{x,r}[\mathcal{B}(1^{2n}, g(x,r)) = (x,r)] = 1$.

Simple case. Next moving on to the following more demanding case.

$$\Pr_{x,r}[E] \geq \frac{3}{4} + \epsilon(n),$$

where $\epsilon(\cdot)$ is non-negligible. We describe $\mathcal{B}$'s algorithm for inverting g in Figure 2.4. Here we can no longer use the super simple case algorithm because we no longer know if $\mathcal{A}$ outputs the correct bit on input $f(x)\|e^i$. Instead, we introduce randomness to $\mathcal{A}$'s input expecting that it should be able to guess the right bit on majority of those inputs since it has a high probability of guessing $B(\cdot)$ in general. We now also need to make two calls to $\mathcal{A}$ to isolate the i-th bit of x. Note that an iteration of $\mathcal{B}$ outputs the right bit if calls to $\mathcal{A}$ output the correct bit because $B(x,s) \oplus B(x, s \oplus e^i) = x_i$:

for $i = 1$ **to** n **do**
  for $t = 1$ **to** $T = \frac{n}{2\epsilon(n)^2}$ **do**
    $s \xleftarrow{\$} \{0,1\}^n$
    $x_i^t \leftarrow \mathcal{A}(f(x)||s)$
      $\oplus\ \mathcal{A}(f(x)||(s \oplus e^i))$
  end for
  $x'_i \leftarrow$ the majority of $\{x_i^1, \cdots, x_i^T\}$
end for
return $x'_1 \cdots x'_n||R$

Figure 2.4: Simple Case $\mathcal{B}$

$$\begin{aligned} B(x,s) \oplus B(x, s \oplus e^i) &= \sum_j x_j s_j \oplus \sum_j x_j (s_j \oplus e^i_j) \\ &= \sum_{j \neq i} (x_j s_j \oplus x_j s_j) \oplus x_i s_i \oplus x_i (s_i \oplus 1) \\ &= x_i \end{aligned}$$

The key technical challenge in proving that $\mathcal{B}$ inverts g with non-negligible probability arises from the fact that the calls to $\mathcal{A}$ made during one iteration of $\mathcal{B}$ are not independent. In particular, all calls to $\mathcal{A}$ share the same x and the calls $\mathcal{A}(f(x)||s)$ and $\mathcal{A}(f(x)||(s \oplus e^i))$ use correlated randomness as well.

We solve the first issue by showing that there exists a large set of x values for which $\mathcal{A}$ still works with large probability. The latter issue of lack of independence between $\mathcal{A}(f(x)||s)$ and $\mathcal{A}(f(x)||(s \oplus e^i))$ can be solved using a union bound since the success probability of the adversary $\mathcal{A}$ is high enough.

Formally, define the set G of "good" x's, for which it is easy for $\mathcal{A}$ to predict the right bit:

$$G := \left\{ x \,\middle|\, \Pr_r[E] \geq \frac{3}{4} + \frac{\epsilon(n)}{2} \right\}.$$

Now we prove that G is not a small set. More formally, we claim that:

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n} [x \in G] \geq \frac{\epsilon(n)}{2}.$$

Assume that $\Pr_{x \xleftarrow{\$} \{0,1\}^n} [x \in G] < \frac{\epsilon(n)}{2}$. Then we have the following contradiction:

$$\begin{aligned}
\frac{3}{4} + \epsilon(n) \leq & \Pr_{x,r}[E] \\
= & \Pr_x[x \in G] \Pr_r[E | x \in G] + \Pr_x[x \notin G] \Pr_r[E | x \notin G] \\
< & \frac{\epsilon(n)}{2} \cdot 1 + 1 \cdot \left(\frac{3}{4} + \frac{\epsilon(n)}{2} \right) = \frac{3}{4} + \epsilon(n).
\end{aligned}$$

Now consider a single iteration for a fixed $x \in G$:

$$\begin{aligned}
& \Pr_s \left[\mathcal{A}(f(x), s) \oplus \mathcal{A}(f(x), s \oplus e^i) = x_i \right] \\
= & \Pr_s [\text{Both } \mathcal{A}\text{'s are correct}] + \Pr_s [\text{Both } \mathcal{A}\text{'s are wrong}] \\
\geq & \Pr_s [\text{Both } \mathcal{A}\text{'s are correct}] = 1 - \Pr_s [\text{Either } \mathcal{A} \text{ is wrong}] \\
\geq & 1 - 2 \cdot \Pr_s [\mathcal{A} \text{ is wrong}] \\
\geq & 1 - 2 \left(\frac{1}{4} - \frac{\epsilon(n)}{2} \right) = \frac{1}{2} + \epsilon(n).
\end{aligned}$$

Let Y_i^t be the indicator random variable that $x_i^t = x_i$ (namely, $Y_i^t = 1$ with probability $\Pr[x_i^t = x_i]$ and $Y_i^t = 0$ otherwise). Note that $Y_i^1, \cdots, Y_i^T$ are independent and identical random variables, and for all $t \in \{1, \ldots, T\}$, we have $\Pr[Y_i^t = 1] = \Pr[x_i^t = x_i] \geq \frac{1}{2} + \epsilon(n)$. Next we argue that majority of x_i^t coincide with x_i with high probability.

$$\begin{aligned}
\Pr[x_i' \neq x_i] = & \Pr \left[\sum_{t=1}^{T} Y_i^t \leq \frac{T}{2} \right] \\
= & \Pr \left[\sum_{t=1}^{T} Y_i^t - \left(\frac{1}{2} + \epsilon(n) \right) T \leq \frac{T}{2} - \left(\frac{1}{2} + \epsilon(n) \right) T \right] \\
\leq & \Pr \left[\left| \sum_{t=1}^{T} Y_i^t - \left(\frac{1}{2} + \epsilon(n) \right) T \right| \geq \epsilon(n) T \right]
\end{aligned}$$

Let $X_1, \cdots, X_m$ be i.i.d. random variables taking values 0 or 1. Let $\Pr[X_i = 1] = p$.

By Chebyshev's Inequality, $\Pr \left[\left| \sum X_i - pm \right| \geq \delta m \right] \leq \dfrac{1}{4\delta^2 m}$.

$$\leq \frac{1}{4\epsilon(n)^2 T} = \frac{1}{2n}.$$

Then, completing the argument, we have

$$
\begin{aligned}
&\Pr_{x,r}[\mathcal{B}(1^{2n}, g(x,r)) = (x,r)] \\
\geq\ & \Pr_x[x \in G] \Pr[x_1' = x_1, \cdots x_n' = x_n | x \in G] \\
\geq\ & \frac{\epsilon(n)}{2} \cdot \left(1 - \sum_{i=1}^{n} \Pr[x_i' \neq x_i | x \in G]\right) \\
\geq\ & \frac{\epsilon(n)}{2} \cdot \left(1 - n \cdot \frac{1}{2n}\right) = \frac{\epsilon(n)}{4}.
\end{aligned}
$$

Real Case. Now, we describe the final case where $\Pr_{x,r}[E] \geq \frac{1}{2} + \epsilon(n)$ and $\epsilon(\cdot)$ is a non-negligible function. The key technical challenge in this case is that we cannot make two related calls to $\mathcal{A}$ as was done in the simple case above since we can't argue that both calls to $\mathcal{A}$ will be correct with high enough probability. However, just using one call to $\mathcal{A}$ seems insufficient. The key idea is to just guess one of those values. Very surprisingly, this idea along with careful analysis magically works out. Just like the previous two cases, we start by describing the algorithm $\mathcal{B}$ in Figure 2.5.

$T = \frac{2n}{\epsilon(n)^2}$
for $\ell = 1$ **to** $\log T$ **do**
  $s_\ell \xleftarrow{\$} \{0,1\}^n$
  $b_\ell \xleftarrow{\$} \{0,1\}$
end for
for $i = 1$ **to** n **do**
  for all $L \subseteq \{1, 2, \cdots, \log T\}$ **do**
    $S_L := \bigoplus_{j \in L} s_j$
    $B_L := \bigoplus_{j \in L} b_j$
    $x_i^L \leftarrow B_L \oplus \mathcal{A}(f(x) || S_L \oplus e^i)$
  end for
  $x_i' \leftarrow$ majority of $\{x_i^{\emptyset}, \cdots, x_i^{[\log T]}\}$
end for
return $x_1' \cdots x_n' || R$

Figure 2.5: Real Case $\mathcal{B}$

In the beginning of the algorithm, $\mathcal{B}$ samples $\log T$ random strings $\{s_\ell\}_\ell$ and bits $\{b_\ell\}_\ell$. Since there are only $\log T$ values, with probability $\frac{1}{T}$ (which is polynomial in n) all the b_ℓ's are correct, i.e., $b_\ell = B(x, s_\ell)$. In the rest of this proof, we denote this event as F. Now note that if F happens, then B_L as defined in the algorithm is also equal to $B(x, S_L)$ (we denote the k^{th}-bit of s with $(s)_k$):

$$
\begin{aligned}
B(x, S_L) &= \sum_{k=1}^{n} x_k (\bigoplus_{j \in L} s_j)_k \\
&= \sum_{k=1}^{n} x_k \sum_{j \in L} (s_j)_k \\
&= \sum_{j \in L} \sum_{k=1}^{n} x_k (s_j)_k \\
&= \sum_{j \in L} B(x, s_j) \\
&= \sum_{j \in L} b_j \\
&= B_L
\end{aligned}
$$

Thus, with probability $\frac{1}{T}$, we have all the right guesses for one of the invocations, and we just need to bound the probability that $\mathcal{A}(f(x) || S_L \oplus e^i) = B(x, S_L \oplus e^i)$. However there is a subtle issue.

Now the events $Y_i^{\emptyset}, \cdots, Y_i^{[\log T]}$ are no longer independent. Nevertheless, we can still show that they are pairwise independent, and the Chebyshev's Inequality still holds. Now we give the formal proof.

Just as in the simple case, we define the set G as

$$G := \left\{ x \,\middle|\, \Pr_r [E] \geq \frac{1}{2} + \frac{\epsilon(n)}{2} \right\},$$

and with an identical argument we obtain that:

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n} [x \in G] \geq \frac{\epsilon(n)}{2}$$

Next, given $\{b_\ell = B(x, s_\ell)\}_{\ell \in [\log T]}$ and $x \in G$, we have:

$$\begin{aligned}
&\Pr_r \left[B_L \oplus \mathcal{A}(f(x)||S_L \oplus e^i) = x_i \right] \\
&= \Pr_r \left[B(x, S_L) \oplus \mathcal{A}(f(x)||S_L \oplus e^i) = x_i \right] \\
&= \Pr_r \left[\mathcal{A}(f(x)||S_L \oplus e^i) = B(x, S_L \oplus e^i) \right] \\
&\geq \frac{1}{2} + \frac{\epsilon(n)}{2}
\end{aligned}$$

For the same $\{b_\ell\}_\ell$ and $x \in G$, let Y_i^L be the indicator random variable that $x_i^L = x_i$. Notice that $Y_i^{\emptyset}, \cdots, Y_i^{[\log T]}$ are pairwise independent and $\Pr[Y_i^L = 1] = \Pr[x_i^L = x_i] \geq \frac{1}{2} + \frac{\epsilon(n)}{2}$.

$$\begin{aligned}
\Pr[x_i' \neq x_i] &= \Pr\left[\sum_{L \subseteq [\log T]} Y_i^L \leq \frac{T}{2} \right] \\
&= \Pr\left[\sum_{L \subseteq [\log T]} Y_i^L - \left(\frac{1}{2} + \frac{\epsilon(n)}{2}\right) T \leq \frac{T}{2} - \left(\frac{1}{2} + \frac{\epsilon(n)}{2}\right) T \right] \\
&\leq \Pr\left[\left| \sum_{L \subseteq [\log T]} Y_i^L - \left(\frac{1}{2} + \frac{\epsilon(n)}{2}\right) T \right| \geq \frac{\epsilon(n)}{2} T \right] \\
&\quad \text{(By Theorem 2.4)} \\
&\leq \frac{1}{4\left(\frac{\epsilon(n)}{2}\right)^2 T} = \frac{1}{2n}.
\end{aligned}$$

Completing the proof, we have that:

$$\begin{aligned}
&\Pr_{x,r}[\mathcal{B}(1^{2n}, g(x, r)) = (x, r)] \\
&\geq \Pr_{\{b_\ell, s_\ell\}_\ell} [F] \cdot \Pr_x [x \in G] \cdot \Pr[x_1' = x_1, \cdots x_n' = x_n \mid x \in G \wedge F] \\
&\geq \frac{1}{T} \cdot \frac{\epsilon(n)}{2} \cdot \left(1 - \sum_{i=1}^{n} \Pr[x_i' \neq x_i \mid x \in G \wedge F] \right) \\
&\geq \frac{\epsilon(n)^2}{2n} \cdot \frac{\epsilon(n)}{2} \cdot \left(1 - n \cdot \frac{1}{2n} \right) = \frac{\epsilon(n)^3}{8n}
\end{aligned}$$

By pairwise independence, for $i \neq j$,
$\mathbb{E}\left[X_i X_j\right] = \mathbb{E}\left[X_i\right] \mathbb{E}\left[X_j\right]$.

$$= \sum_{i=1}^{m} \mathbb{E}\left[X_i^2\right] - \mathbb{E}\left[X_i\right]^2$$

$$= mp(1-p).$$

□

Hence

$$\Pr\left[\left|\sum_{i=1}^{m} X_i - pm\right| \geq \delta m\right] \leq \frac{mp(1-p)}{\delta^2 m^2} \leq \frac{1}{\delta^2 m}.$$

□

Exercises

Exercise 2.1. *If $\mu(\cdot)$ and $\nu(\cdot)$ are negligible functions then show that $\mu(\cdot) \cdot \nu(\cdot)$ is a negligible function.*

Exercise 2.2. *If $\mu(\cdot)$ is a negligible function and $f(\cdot)$ is a function polynomial in its input then show that $\mu(f(\cdot))$*[4] *are negligible functions.*

[4] Assume that μ and f are such that $\mu(f(\cdot))$ takes inputs from $\mathbb{Z}^+$ and outputs values in $[0, 1]$.

Exercise 2.3. *Prove that the existence of one-way functions implies $P \neq NP$.*

Exercise 2.4. *Prove that there is no one-way function $f : \{0,1\}^n \rightarrow \{0,1\}^{\lfloor \log_2 n \rfloor}$.*

Exercise 2.5. *Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be any one-way function then is $f'(x) \stackrel{def}{=} f(x) \oplus x$ necessarily one-way?*

Exercise 2.6. *Prove or disprove: If $f : \{0,1\}^n \rightarrow \{0,1\}^n$ is a one-way function, then $g : \{0,1\}^n \rightarrow \{0,1\}^{n-\log n}$ is a one-way function, where $g(x)$ outputs the $n - \log n$ higher order bits of $f(x)$.*

Exercise 2.7. *Explain why the proof of Theorem 2.1 fails if the attacker $\mathcal{A}$ in Figure 2.2 sets $i = 1$ and not $i \stackrel{\$}{\leftarrow} \{1, 2, \cdots, q\}$.*

Exercise 2.8. *Given a (strong) one-way function construct a weak one-way function that is not a (strong) one-way function.*

Exercise 2.9. *Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be a weak one-way permutation (a weak one way function that is a bijection). More formally, f is a PPT computable one-to-one function such that $\exists$ a constant $c > 0$ such that $\forall$ non-uniform PPT machine A and $\forall$ sufficiently large n we have that:*

$$\Pr_{x,A}[A(f(x)) \notin f^{-1}(f(x))] > \frac{1}{n^c}$$

Show that $g(x) = f^T(x)$ is not a strong one way permutation. Here f^T denotes the T times self composition of f and T is a polynomial in n.

Interesting follow up reading if interested: With some tweaks the function above can be made a strong one-way permutation using explicit constructions of expander graphs. See Section 2.6 in `http://www.wisdom.weizmann.ac.il/~oded/PSBookFrag/part2N.ps`

3
Pseudorandomness

3.1 Distinguishability Between Two Distributions

Sanjam: add

3.2 Computational Indistinguishability

Defining indistinguishability between two distributions by a computationally bounded adversary turns out be tricky. In particular, It is tricky to define for a single pair of distributions because the length of the output of a random variable is a constant. Therefore, in order for "computationally bounded" adversaries to make sense, we have to work with infinite families of probability distributions.

Definition 3.1. *An* ensemble *of probability distributions is a sequence of random variables* $\{X_n\}_{n\in\mathbb{N}}$. *Two ensembles of probability distributions* $\{X_n\}_n$ *and* $\{Y_n\}_n$ *(which are* samplable *in time polynomial in* n*) are said to be* computationally indistinguishable *if for all (non-uniform) PPT machines* $\mathcal{A}$, *the quantities*

$$p(n) := \Pr[\mathcal{A}(1^n, X_n) = 1] = \sum_x \Pr[X_n = x] \Pr[\mathcal{A}(1^n, x) = 1]$$

and

$$q(n) := \Pr[\mathcal{A}(1^n, Y_n) = 1] = \sum_y \Pr[Y_n = y] \Pr[\mathcal{A}(1^n, y) = 1]$$

differ by a negligible amount; i.e. $|p(n) - q(n)|$ *is negligible in* n. *This equivalence is denoted by*

$$\{X_n\}_n \approx \{Y_n\}_n$$

We now prove some properties of computationally indistinguishable ensembles that will be useful later on.

Lemma 3.1 (Sunglass Lemma). *If $\{X_n\}_n \approx \{Y_n\}_n$ and P is a PPT machine, then*

$$\{P(X_n)\}_n \approx \{P(Y_n)\}_n$$

Proof. Consider an adversary $\mathcal{A}$ that can distinguish $\{P(X_n)\}_n$ from $\{P(Y_n)\}_n$ with non-negligible probability. Then the adversary $\mathcal{A} \circ P$ can distinguish $\{X_n\}_n$ from $\{Y_n\}_n$ with the same non-negligible probability. Since P and $\mathcal{A}$ are both PPT machines, the composition is also a PPT machine. This proves the contrapositive of the lemma. □

Lemma 3.2 (Hybrid Argument). *For a polynomial $t : \mathbb{Z}^+ \to \mathbb{Z}^+$ let the t-product of $\{Z_n\}_n$ be*

$$\{Z_n^{(1)}, Z_n^{(2)}, \ldots, Z_n^{(t(n))}\}_n$$

where the $Z_n^{(i)}$s are independent copies of Z_n. If

$$\{X_n\}_n \approx \{Y_n\}_n$$

then

$$\{X_n^{(1)}, \ldots, X_n^{(t)}\}_n \approx \{Y_n^{(1)}, \ldots, Y_n^{(t)}\}_n$$

as well.

Proof. Consider the set of tuple random variables

$$H_n^{(i,t)} = (Y_n^{(1)}, \ldots, Y_n^{(i)}, X_n^{(i+1)}, X_n^{(i+2)}, \ldots, X_n^{(t)})$$

for integers $0 \leq i \leq t$. Assume, for the sake of contradiction, that there is a PPT adversary $\mathcal{A}$ that can distinguish between $\{H_n^{(0,t)}\}_n$ and $\{H_n^{(t,t)}\}_n$ with non-negligible probability difference $r(n)$. Suppose that $\mathcal{A}$ returns 1 with probability ϵ_i when it runs on samples from $H_n^{(i,t)}$. By definition, $|\epsilon_t - \epsilon_0| \geq r(n)$. By the Triangle Inequality and the Pigeonhole Principle, there is some index k for which $|\epsilon_{k+1} - \epsilon_k| \geq r(n)/t$. However, using Sunglass Lemma, note that the computational indistinguishability of X_n and Y_n implies that $\{H_n^{(k,t)}\}_n$ and $\{H_n^{(k+1,t)}\}_n$ are computationally indistinguishable. This is a contradiction. □ □

3.3 Pseudorandom Generators

Now, we can define pseudorandom generators, which intuitively generates a polynomial number of bits that are indistinguishable from being uniformly random:

Definition 3.2. *A function $G : \{0,1\}^n \to \{0,1\}^{n+m}$ with $m = poly(n)$ is called a* pseudorandom generator *if*

- *G is computable in polynomial time.*
- *$U_{n+m} \approx G(U_n)$, where U_k denotes the uniform distribution on $\{0,1\}^k$.*

3.3.1 PRG Extension

In this section we show that any pseudorandom generator that produces one bit of randomness can be extended to create a polynomial number of bits of randomness.

Construction 3.1. *Given a PRG $G : \{0,1\}^n \to \{0,1\}^{n+1}$, we construct a new PRG $F : \{0,1\}^n \to \{0,1\}^{n+l}$ as follows (l is polynomial in n).*

(a) Input: $S_0 \xleftarrow{\$} \{0,1\}^n$.

(b) $\forall i \in [l] = \{1, 2, \cdots, l\}$, $(\sigma_i, S_i) := G(S_{i-1})$, where $\sigma_i \in \{0,1\}, S_i \in \{0,1\}^n$.

(c) Output: $\sigma_1\sigma_2 \cdots \sigma_l S_l$.

Theorem 3.1. *The function F constructed above is a PRG.*

Proof. We prove this by hybrid argument. Define the hybrid H_i as follows.

(a) Input: $S_0 \xleftarrow{\$} \{0,1\}^n$.

(b) $\sigma_1, \sigma_2, \cdots, \sigma_i \xleftarrow{\$} \{0,1\}$, $S_i \leftarrow S_0$.
$\forall j \in \{i+1, i+2, \cdots, l\}$, $(\sigma_j, S_j) := G(S_{j-1})$, where $\sigma_j \in \{0,1\}, S_j \in \{0,1\}^n$.

(c) Output: $\sigma_1\sigma_2 \cdots \sigma_l S_l$.

Note that $H_0 \equiv F$, and $H_l \equiv U_{n+l}$.

Assume for the sake of contradiction that there exits a non-uniform PPT adversary $\mathcal{A}$ that can distinguish H_0 form H_l. Define $\epsilon_i := \Pr[\mathcal{A}(1^n, H_i) = 1]$ for $i = 0, 1, \cdots, l$. Then there exists a non-negligible function $v(n)$ such that $|\epsilon_0 - \epsilon_l| \geq v(n)$. Since

$$|\epsilon_0 - \epsilon_1| + |\epsilon_1 - \epsilon_2| + \cdots + |\epsilon_{l-1} - \epsilon_l| \geq |\epsilon_0 - \epsilon_l| \geq v(n),$$

there exists $k \in \{0, 1, \cdots, l-1\}$ such that

$$|\epsilon_k - \epsilon_{k+1}| \geq \frac{v(n)}{l}.$$

l is polynomial in n, hence $\frac{v(n)}{l}$ is also a non-negligible function. That is to say, $\mathcal{A}$ can distinguish H_k from H_{k+1}. Then we use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that can distinguish U_{n+1} from $G(U_n)$ (which leads to a contradiction): On input $T \in \{0,1\}^{n+1}$ (T could be either from U_{n+1} or $G(U_n)$), $\mathcal{B}$ proceeds as follows:

- $\sigma_1, \sigma_2, \cdots, \sigma_k \xleftarrow{\$} \{0,1\}$, $(\sigma_{k+1}, S_{k+1}) \leftarrow T$.
- $\forall j \in \{k+2, k+3, \cdots, l\}$, $(\sigma_j, S_j) := G(S_{j-1})$, where $\sigma_j \in \{0,1\}, S_j \in \{0,1\}^n$.
- Output: $\mathcal{A}(1^n, \sigma_1\sigma_2 \cdots \sigma_l S_l)$.

First, since $\mathcal{A}$ and G are both PPT computable, $\mathcal{B}$ is also PPT computable.

Second, if $T \leftarrow G(U_n)$, then $\sigma_1\sigma_2 \cdots \sigma_l S_l$ is the output of H_k; if $T \xleftarrow{\$} U_{n+1}$, then $\sigma_1\sigma_2 \cdots \sigma_l S_l$ is the output of H_{k+1}. Hence

$$\begin{aligned}
&\left|\Pr[\mathcal{B}(1^n, G(U_n)) = 1] - \Pr[\mathcal{B}(1^n, U_{n+1}) = 1]\right| \\
=&\left|\Pr[\mathcal{A}(1^n, H_k) = 1] - \Pr[\mathcal{A}(1^n, H_{k+1}) = 1]\right| \\
=&|\epsilon_k - \epsilon_{k+1}| \geq \frac{v(n)}{l}.
\end{aligned}$$

□

3.3.2 *PRG from OWP (One-Way Permutations)*

In this section we show how to construct pseudorandom generators under the assumption that one-way permutations exist.

Construction 3.2. *Let $f : \{0,1\}^n \to \{0,1\}^n$ be a OWP. We construct $G : \{0,1\}^{2n} \to \{0,1\}^{2n+1}$ as*

$$G(x, r) = f(x)||r||B(x, r),$$

where $x, r \in \{0,1\}^n$, and $B(x, r)$ is a hard concentrate bit for the function $g(x, r) = f(x)||r$.

Remark 3.1. *The hard concentrate bit $B(x, r)$ always exists. Recall Theorem 2.3,*

$$B(x, r) = \left(\sum_{i=1}^{n} x_i r_i\right) \mod 2$$

is a hard concentrate bit.

Theorem 3.2. *The G constructed above is a PRG.*

Proof. Assume for the sake of contradiction that G is not PRG. We construct three ensembles of probability distributions:

$$H_0 := G(U_{2n}) = f(x)||r||B(x, r), \text{ where } x, r \xleftarrow{\$} \{0,1\}^n;$$

$$H_1 := f(x)||r||\sigma, \text{ where } x, r \xleftarrow{\$} \{0,1\}^n, \sigma \xleftarrow{\$} \{0,1\};$$
$$H_2 := U_{2n+1}.$$

Since G is not PRG, there exists a non-uniform PPT adversary $\mathcal{A}$ that can distinguish H_0 from H_2. Since f is a permutation, H_1 is uniformly distributed in $\{0,1\}^{2n+1}$, i.e., $H_1 \equiv H_2$. Therefore, $\mathcal{A}$ can distinguish H_0 from H_1, that is, there exists a non-negligible function $v(n)$ satisfying

$$\big|\Pr[\mathcal{A}(H_0) = 1] - \Pr[\mathcal{A}(H_1) = 1]\big| \geq v(n).$$

Next we will construct an adversary $\mathcal{B}$ that "breaks" the hard concentrate bit (which leads to a contradiction). Define a new ensemble of probability distribution

$$H_1' = f(x)||r||(1 - B(x,r)), \text{ where } x, r \xleftarrow{\$} \{0,1\}^n.$$

Then we have

$$\begin{aligned}\Pr[\mathcal{A}(H_1) = 1] =& \Pr[\sigma = B(x,r)] \Pr[A(H_0) = 1] + \Pr[\sigma = 1 - B(x,r)] \Pr[A(H_1') = 1]\\ =& \frac{1}{2}\Pr[A(H_0) = 1] + \frac{1}{2}\Pr[A(H_1') = 1].\end{aligned}$$

Hence

$$\Pr[A(H_1) = 1] - \Pr[A(H_0) = 1] = \frac{1}{2}\Pr[A(H_1') = 1] - \frac{1}{2}\Pr[A(H_0) = 1],$$
$$\frac{1}{2}\big|\Pr[A(H_0) = 1] - \Pr[A(H_1') = 1]\big| = |\Pr[A(H_1) = 1] - \Pr[A(H_0) = 1]| \geq v(n),$$
$$\big|\Pr[A(H_0) = 1] - \Pr[A(H_1') = 1]\big| \geq 2v(n).$$

Without loss of generality, we assume that

$$\Pr[A(H_0) = 1] - \Pr[A(H_1') = 1] \geq 2v(n).$$

Then we construct $\mathcal{B}$ as follows:

$$\mathcal{B}(f(x)||r) := \begin{cases} \sigma, & \text{if } \mathcal{A}(f(x)||r||\sigma) = 1 \\ 1 - \sigma, & \text{if } \mathcal{A}(f(x)||r||\sigma) = 0 \end{cases},$$

where $\sigma \xleftarrow{\$} \{0,1\}$. Then we have

$$\begin{aligned}&\Pr[\mathcal{B}(f(x)||r) = B(x,r)]\\ =& \Pr[\sigma = B(x,r)] \Pr[\mathcal{A}(f(x)||r||\sigma) = 1 | \sigma = B(x,r)] +\\ &\Pr[\sigma = 1 - B(x,r)] \Pr[\mathcal{A}(f(x)||r||\sigma) = 0 | \sigma = 1 - B(x,r)] +\\ =& \frac{1}{2}\big(\Pr[\mathcal{A}(f(x)||r||B(x,r)) = 1] + 1 - \Pr[\mathcal{A}(f(x)||r||1 - B(x,r)) = 1]\big)\\ =& \frac{1}{2} + \frac{1}{2}\big(\Pr[A(H_0) = 1] - \Pr[A(H_1') = 1]\big)\\ \geq& \frac{1}{2} + v(n).\end{aligned}$$

Contradiction with the fact that B is a hard concentrate bit. $\square$

3.4 Pseudorandom Functions

In this section, we first define pseudorandom functions, and then show how to construct a pseudorandom function from a pseudorandom generator.

Considering the set of all functions $f : \{0,1\}^n \to \{0,1\}^n$, there are $(2^n)^{2^n}$ of them. To describe a random function in this set we need $n \cdot 2^n$ bits. Intuitively, a pseudorandom function is one that cannot be distinguished from a random one, but needs much fewer bits (e.g., polynomial in n) to be described. Note that we restrict the distinguisher to only being allowed to ask the function $\text{poly}(n)$ times and decide whether it is random or pseudorandom.

3.4.1 Definitions

Definition 3.3 (Function Ensemble). *A* function ensemble *is a sequence of random variables $F_1, F_2, \cdots, F_n, \cdots$ denoted as $\{F_n\}_{n\in\mathbb{N}}$ such that F_n assumes values in the set of functions mapping n-bit input to n-bit output.*

Definition 3.4 (Random Function Ensemble). *We denote a random function ensemble by $\{R_n\}_{n\in\mathbb{N}}$.*

Definition 3.5 (Efficiently Computable Function Ensemble). *A function ensemble is called* efficiently computable *if*

(a) ***Succinct****: $\exists$ a PPT algorithm I and a mapping ϕ from strings to functions such that $\phi(I(1^n))$ and F_n are identically distributed. Note that we can view I as the description of the function.*

(b) ***Efficient****: $\exists$ a poly-time machine V such that $V(i,x) = f_i(x)$ for every $x \in \{0,1\}^n$, where i is in the range of $I(1^n)$, and $f_i = \phi(i)$.*

Definition 3.6 (Pseudorandom Function Ensemble). *A function ensemble $F = \{F_n\}_{n\in\mathbb{N}}$ is* pseudorandom *if for every non-uniform PPT oracle adversary $\mathcal{A}$, there exists a negligible function $\epsilon(n)$ such that*

$$\left| \Pr[\mathcal{A}^{F_n}(1^n) = 1] - \Pr[\mathcal{A}^{R_n}(1^n) = 1] \right| \le \epsilon(n).$$

Here by saying "oracle" it means that $\mathcal{A}$ has "oracle access" to a function (in our definition, the function is F_n or R_n), and each call to that function costs 1 unit of time.

Note that we will only consider efficiently computable pseudorandom ensembles in the following.

3.4.2 Construction of PRF from PRG

Construction 3.3. *Given a PRG $G : \{0,1\}^n \to \{0,1\}^{2n}$, let $G_0(x)$ be the first n bits of $G(x)$, $G_1(x)$ be the last n bits of $G(x)$. We construct $F^{(K)} : \{0,1\}^n \to \{0,1\}^n$ as follows.*

$$F_n^{(K)}(x_1 x_2 \cdots x_n) := G_{x_n}(G_{x_{n-1}}(\cdots(G_{x_1}(K))\cdots)),$$

where $K \in \{0,1\}^n$ is the key to the pseudorandom function. Here i is an n-bit string, which is the seed of the pseudorandom function.

The construction can be viewed as a binary tree of depth n, as shown in Figure 3.1.

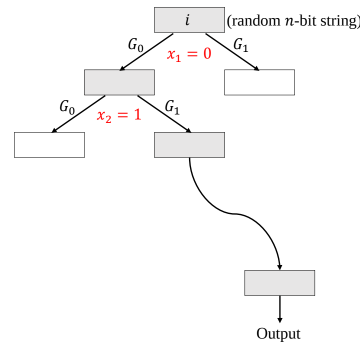

Figure 3.1: View the construction as a binary tree

Theorem 3.3. *The function ensemble $\{F_n\}_{n\in\mathbb{N}}$ constructed above is pseudorandom.*

Proof. Assume for the sake of contradiction that $\{F_n\}_{n\in\mathbb{N}}$ is not PRG. Then there exists a non-uniform PPT oracle adversary $\mathcal{A}$ that can distinguish $\{F_n\}_{n\in\mathbb{N}}$ from $\{R_n\}_{n\in\mathbb{N}}$. Below, via a hybrid argument, we prove that this contradicts the fact that G is a PRG.

Consider the sequence of hybrids H_i for $i \in \{0, 1, \cdots, n\}$ where the hybrid i is defined as follows:

$$H_{n,i}^{(K)}(x_1 x_2 \ldots x_n) := G_{x_n}(G_{x_{n-1}}(\cdots(G_{x_{i+1}}(K(x_i x_{i-1} \ldots x_1)))\cdots)),$$

where K is a random function from $\{0,1\}^i$ to $\{0,1\}^n$. Intuitively, hybrid H_i corresponds to a binary tree of depth n where the nodes of levels 0 to i correspond to random values and the nodes at levels $i+1$ to n correspond to pseudorandom values. By inspection, observe that hybrids H_0 and H_n are identical to a pseudorandom function and a random function, respectively. There it suffices to prove that hybrids H_i and H_{i+1} are computationally indistinguishable for each $i \in \{0, 1, \cdots, n\}$.

We show that H_i and H_{i+1} are indistinguishable by considering a sequence of sub-hybrids $H_{i,j}$ for $j \in \{0, \ldots q_{i+1}\}$, where q_{i+1} is the number of the distinct $i - bit$ prefixes of the queries of $\mathcal{A}$.[1]

We define hybrid $H_{i,j}$ for $j = 0$ to be same as hybrid H_i. Additionally, for $j > 0$ hybrid $H_{i,j}$ is defined to be exactly the same as hybrid $H_{i,j-1}$ except the response provided to the attacker for the j^{th} distinct $i - bit$ prefix query of $\mathcal{A}$. Let this prefix be $x_n^* x_{n-1}^* \ldots x_i^*$. Note that in hybrid $H_{i,j-1}$ the children of the node $x_n^* x_{n-1}^* \ldots x_i^*$ correspond to two pseudorandom values. In hybrid $H_{i,j}$ we replace these two children with random values. By careful inspection, it follows that hybrid $H_{i,q_{i+1}}$ is actually H_{i+1}. All we are left to prove is that hybrid $H_{i,j}$ and $H_{i,j+1}$ are indistinguishable for the appropriate choices of j and we prove this below.

[1] Observe that q_{i+1} for each appropriate choice of i is bounded by the running time of $\mathcal{A}$. Hence, this value is bounded by a polynomial in the security parameter.

Now we are ready to construct an adversary $\mathcal{B}$ that distinguishes U_{2n} from $G(U_n)$: On input $T \in \{0,1\}^{2n}$ (T could be either from U_{2n} or $G(U_n)$), construct a full binary tree of depth n that is exactly the same as $H_{i,j}$ except replacing the children of $x_n^* x_{n-1}^* \dots x_i^*$ by the value T. Observe that the only difference between $H_{i,j}$ and $H_{i,j+1}$ is that values corresponding to nodes $x_n^* \dots x_i^* 0$ and $x_n^* \dots x_i^* 1$ are pseudorandom or random respectively. $\mathcal{B}$ uses the value T to generate these two nodes. Hence success in distinguishing hybrids $H_{i,j}$ and $H_{i,j+1}$ provides a successful attack for $\mathcal{B}$ in violating security of the pseudorandom generator. □

3.5 PRFs from DDH: Naor-Reingold PRF

We will now describe a PRF function family $F_n : \mathcal{K} \times \{0,1\}^n \to \mathbb{G}_n$ where DDH is assumed to be hard for $\{\mathbb{G}_n\}$ and $\mathcal{K}$ is the key space. The seed for the PRF F_n will be $K = (h, u_1, \dots u_n)$, where $u, u_0 \dots u_n$ are sampled uniformly from $|\mathbb{G}_n|$, g is the generator of $\mathbb{G}_n$ and $h = g^u$.

$$F_n(K, x) = h^{\prod_i u_i^{x_i}}$$

Next, we will prove that the function F_n is a pseudo-random function or that $\{F_n\}$ is a pseudo-random function ensemble.[2]

[2] Here, we require that adversary distinguish the function F_n from a random function from $\{0,1\}^n$ to $\mathbb{G}_n$. Note that the output range of the function is $\mathbb{G}_n$. Note that the distribution of random group elements in $\mathbb{G}_n$ might actually be far from uniformly random strings.

Lemma 3.3. *Assuming the DDH Assumption (see Definition 1.7) for $\{\mathbb{G}_n\}$ is hard, we have that $\{F_n\}$ is a pseudorandom function ensemble.*

Proof. The proof of this lemma is similar to the proof of Theorem 3.3.

Let R_n^j be random function from $\{0,1\}^j \to \mathbb{G}_n$. Then we want to prove that for all non-uniform PPT adversaries $\mathcal{A}$ we have that:

$$\mu(n) = \left|\Pr[\mathcal{A}^{F_n}(1^n) = 1] - \Pr[\mathcal{A}^{R_n^n}(1^n) = 1]\right|$$

is a negligible function.

For the sake of contradiction, we assume that the function F_n is not pseudorandom. Next, towards a contradiction, we consider a sequence of hybrid functions $F_n^0 \dots F_n^n$. For any $j \in \{0, \dots n\}$, let $F_n^j((h, u_j \dots u_n), x) = (R_n^j(x_1 \dots x_j))^{\prod_{i=j+1}^n u_i^{x_i}}$, where $R_n^0(\epsilon)$ is the constant function with output h. Observe that F_n^0 is the same as the function F_n and F_n^n is the same as the function R_n^n. Thus, by a hybrid argument, we conclude that there exists $k \in \{0, \dots n-1\}$, such that

$$\left|\Pr[\mathcal{A}^{F_n^k}(1^n) = 1] - \Pr[\mathcal{A}^{F_n^{k+1}}(1^n) = 1]\right|$$

is a non-negligible function. Now all we are left to show is that this implies an attacker that refutes the DDH assumption. The proof of this claim follows by a sequence of T sub-hybrids, where T is the running time of $\mathcal{A}$. Without loss of generality we assume that $\mathcal{A}$ never makes the same query twice.

More specifically, we consider a sequence of functions $F_n^{k,t}$ where $t \in \{0, T\}$, $F_n^{k,0}$ is same as F_n^k and $F_n^{k,T}$ is same as F_n^{k+1}. In particular, we explain how $F_n^{k,t}$ answers queries by $\mathcal{A}$.[3] Let $x^1, \ldots x^t$ be the first t queries made by $\mathcal{A}$. For any query, x made by $\mathcal{A}$ such that the first k bits of x match the first k bits of one of $x_1, \ldots x_y$ answer as F_n^{k+1} else answer as F_n^k. Now we can conclude that there exists a t such that $F_n^{k,t}$ and $F_n^{k,t+1}$ are distinguishable with non-negligible probability.

[3] As assumed earlier, keep in mind that $\mathcal{A}$ never makes the same query twice.

Finally, we will show that using an adversary that can distinguish between $F_n^{k,t}$ and $F_n^{k,t+1}$ we need to construct an adversary $\mathcal{B}$ that refutes the DDH assumption. We leave construction of this adversary as an exercise. $\square$

Exercises

Exercise 3.1. *Prove or disprove: If f is a one-way function, then the following function $B : \{0,1\}^* \to \{0,1\}$ is a hardconcentrate predicate for f. The function $B(x)$ outputs the inner product modulo 2 of the first $\lfloor |x|/2 \rfloor$ bits of x and the last $\lfloor |x|/2 \rfloor$ bits of x.*

Exercise 3.2. *Let $\phi(n)$ denote the first n digits of $\pi = 3.141592653589\ldots$ after the decimal in binary (π in its binary notation looks like $11.001001000011111101101010100010001000001\ldots$).*

Prove the following: if one-way functions exist, then there exists a one-way function f such that the function $B : \{0,1\}^* \to \{0,1\}$ is not a hard concentrate bit of f. The function $B(x)$ outputs $\langle x, \phi(|x|)\rangle$, where

$$\langle a, b\rangle := \sum_{i=1}^{n} a_i b_i \mod 2$$

for the bit-representation of $a = a_1 a_2 \cdots a_n$ and $b = b_1 b_2 \cdots b_n$.

Exercise 3.3. *If $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is PRF, then in which of the following cases is $g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ also a PRF?*

1. $g(K, x) = f(K, f(K, x))$
2. $g(K, x) = f(x, f(K, x))$
3. $g(K, x) = f(K, f(x, K))$

Exercise 3.4 (Puncturable PRFs.)**.** *Puncturable PRFs are PRFs for which a key can be given out such that, it allows evaluation of the PRF on all inputs, except for one designated input.*

A puncturable pseudo-random function F is given by a triple of efficient algorithms (Key_F,$\mathsf{Puncture}_F$, and Eval_F), satisfying the following conditions:

- ***Functionality preserved under puncturing**: For every $x^*, x \in \{0,1\}^n$ such that $x^* \neq x$, we have that:*

  $$\Pr[\mathsf{Eval}_F(K, x) = \mathsf{Eval}_F(K_{x^*}, x) : K \leftarrow \mathsf{Key}_F(1^n), K_{x^*} = \mathsf{Puncture}_F(K, x^*)] = 1$$

- ***Pseudorandom at the punctured point**: For every $x^* \in \{0,1\}^n$ we have that for every polysize adversary $\mathcal{A}$ we have that:*

  $$|\Pr[\mathcal{A}(K_{x^*}, \mathsf{Eval}_F(K, x^*)) = 1] - \Pr[\mathcal{A}(K_{x^*}, \mathsf{Eval}_F(K, U_n)) = 1]| = \mathsf{negl}(n)(n)$$

  where $K \leftarrow \mathsf{Key}_F(1^n)$ and $K_S = \mathsf{Puncture}_F(K, x^*)$. U_n denotes the uniform distribution over n bits.

Prove that: If one-way functions exist, then there exists a puncturable PRF family that maps n bits to n bits.

***Hint:** The GGM tree-based construction of PRFs from a length doubling pseudorandom generator (discussed in class) can be adapted to construct a puncturable PRF. Also note that K and K_{x^*} need not be the same length.*

4
Private-Key Cryptography

Definition 4.1 (Pri-IND-CPA). *A private-key encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* Pri-IND-CPA*-secure if*

$$\mathsf{Adv}^{\text{ind-cpa}}_{\Pi,\mathcal{A}}(n) = \mid \Pr[\text{Pri-IND-CPA}^{\mathcal{A}}_{\Pi}(n) = 1] - \frac{1}{2} \mid$$

is a negligible function.

$\text{Pri-IND-CPA}^{\mathcal{A}}_{\Pi}(n)$

$b \leftarrow\!\!\$ \{0,1\}$

$\mathsf{k} \leftarrow\!\!\$ \mathsf{Gen}(1^n)$

$(\mathsf{state}, m_0, m_1) \leftarrow\!\!\$ \mathcal{A}^{\mathsf{Enc}(\mathsf{k},\cdot)}(1^n)$

$c \leftarrow\!\!\$ \mathsf{Enc}(\mathsf{k}, m_b)$

$b' \leftarrow\!\!\$ \mathcal{A}^{\mathsf{Enc}(\mathsf{k},\cdot)}(\mathsf{state}, c)$

return $b = b'$

5
Digital Signatures

In this chapter, we will introduce the notion of a digital signature. At an intuitive level, a digital signature scheme helps providing authenticity of messages and ensuring non-repudiation. We will first define this primitive and then construct what is called as one-time secure digital signature scheme. An one-time digital signature satisfies a weaker security property when compared to digital signatures. We then introduce the concept of collision-resistant hash functions and then use this along with a one-time secure digital signature to give a construction of digital signature scheme.

5.1 Definition

A digital signature scheme is a tuple of three algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ with the following syntax:

1. $\mathsf{Gen}(1^n) \to (vk, sk)$: On input the message length (in unary) 1^n, Gen outputs a secret signing key sk and a public verification key vk.

2. $\mathsf{Sign}(sk, m) \to \sigma$: On input a secret key sk and a message m of length n, the Sign algorithm outputs a signature σ.

3. $\mathsf{Verify}(vk, m, \sigma) \to \{0, 1\}$: On input the verification key vk, a message m and a signature σ, the Verify algorithm outputs either 0 or 1.

We require that the digital signature to satisfy the following correctness and security properties.

Correctness. For the correctness of the scheme, we have that $\forall m \in \{0,1\}^n$,

$$\Pr\left[(vk, sk) \leftarrow \mathsf{Gen}(1^n), \sigma \leftarrow \mathsf{Sign}(sk, m) : \mathsf{Verify}(vk, m, \sigma) = 1\right] = 1.$$

Security. Consider the following game between an adversary and a challenger .

1. The challenger first samples $(vk, sk) \leftarrow \mathsf{Gen}(1^n)$. The challenger gives vk to the adversary.

2. **Signing Oracle.** The adversary is now given access to a signing oracle. When the adversary gives a query m to the oracle, it gets back $\sigma \leftarrow \mathsf{Sign}(sk, m)$.

3. **Forgery.** The adversary outputs a message, signature pair (m^*, σ^*) where m^* is different from the queries that adversary has made to the signing oracle.

4. The adversary wins the game if $\mathsf{Verify}(vk, m^*, \sigma^*) = 1$.

We say that the digital signature scheme is secure if the probability that the adversary wins the game is $\mathsf{negl}(n)(n)$.

5.2 *One-time Digital Signature*

An one-time digital signature has the same syntax and correctness requirement as that of a digital signature scheme except that in the security game the adversary is allowed to call the signing oracle only once (hence the name one-time). We will now give a construction of one-time signature scheme from the assumption that one-way functions exists.

Let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way function.

- $\mathsf{Gen}(1^n)$: On input the message length (in unary) 1^n, Gen does the following:
  1. Chooses $x_{i,b} \leftarrow \{0,1\}^n$ for each $i \in [n]$ and $b \in \{0,1\}$.
  2. Output $vk = \begin{bmatrix} f(x_{1,0}) & \dots & f(x_{n,0}) \\ f(x_{1,1}) & \dots & f(x_{n,1}) \end{bmatrix}$ and $sk = \begin{bmatrix} x_{1,0} & \dots & x_{n,0} \\ x_{1,1} & \dots & x_{n,1} \end{bmatrix}$
- $\mathsf{Sign}(sk, m)$: On input a secret key sk and a message $m \in \{0,1\}^n$, the Sign algorithm outputs a signature $\sigma = x_{1,m_1} \| x_{2,m_2} \| \dots \| x_{n,m_n}$.
- $\mathsf{Verify}(vk, m, \sigma)$: On input the verification key vk, a message m and a signature σ, the Verify algorithm does the following:
  1. Parse $\sigma = x_{1,m_1} \| x_{2,m_2} \| \dots \| x_{n,m_n}$.
  2. Compute $vk'_{i,m_i} = f(x_{i,m_i})$ for each $i \in [n]$.
  3. Check if for each $i \in [n]$, $vk'_{i,m_i} = vk_{i,m_i}$. If all the checks pass, output 1. Else, output 0.

Before we prove any security property, we first observe that this scheme is completely broken if we allow the adversary to ask for two signatures. This is because the adversary can query for the signatures on 0^n and 1^n respectively and the adversary gets the entire secret key. The adversary can then use this secret key to sign on any message and break the security.

We will now argue the one-time security of this construction. Let $\mathcal{A}$ be an adversary who breaks the security of our one-time digital signature scheme with non-negligible probability $\mu(n)$. We will now construct an adversary $\mathcal{B}$ that breaks the one-wayness of f. $\mathcal{B}$ receives a one-way function challenge y and does the following:

1. $\mathcal{B}$ chooses i^* uniformly at random from $[n]$ and b^* uniformly at random from $\{0,1\}$.

2. It sets $vk_{i^*,b^*} = y$

3. For all $i \in [n]$ and $b \in \{0,1\}$ such that $(i,b) \neq (i^*,b^*)$, $\mathcal{B}$ samples $x_{i,b} \leftarrow \{0,1\}^n$. It computes $vk_{i,b} = f(x_{i,b})$.

4. It sets $vk = \begin{bmatrix} vk_{1,0} & \dots & vk_{n,0} \\ vk_{1,1} & \dots & vk_{n,1} \end{bmatrix}$ and sends vk to $\mathcal{A}$.

5. $\mathcal{A}$ now asks for a signing query on a message m. If $m_{i^*} = b^*$ then $\mathcal{B}$ aborts and outputs a special symbol abort_1. Otherwise, it uses it knowledge of $x_{i,b}$ for $(i,b) \neq (i^*,b^*)$ to output a signature on m.

6. $\mathcal{A}$ outputs a valid forgery (m^*,σ^*). If $m^*_{i^*} = m_{i^*}$ then $\mathcal{B}$ aborts and outputs a special symbol abort_2. If it does not abort, then it parses σ^* as $1, m_1 \| x_{2,m_2} \| \dots \| x_{n,m_n}$ and outputs x_{i^*,b^*} as the inverse of y.

We first note that conditioned on $\mathcal{B}$ not outputting abort_1 or abort_2, the probability that $\mathcal{B}$ outputs a valid preimage of y is $\mu(n)$. Now, probability $\mathcal{B}$ does not output abort_1 or abort_2 is $1/2n$ (this is because abort_1 is not output with probability $1/2$ and conditioned on not outputting abort_1, abort_2 is not output with probability $1/n$). Thus, $\mathcal{B}$ outputs a valid preimage with probability $\mu(n)/2n$. This completes the proof of security.

We now try to extend this one-time signature scheme to digital signatures. For this purpose, we will rely on a primitive called as collision-resistant hash functions.

5.3 *Collision Resistant Hash Functions*

As the name suggests, collision resistant hash function family is a set of hash functions H such that for a function h chosen randomly from the family, it is computationally hard to find two different inputs x, x' such that $h(x) = h(x')$. We now give a formal definition.

5.3.1 *Definition of a family of CRHF*

A set of function ensembles

$$\{H_n = \{h_i : D_n \to R_n\}_{i \in I_n}\}_n$$

where $|D_n| < |R_n|$ is a family of collision resistant hash function ensemble if there exists efficient algorithms $(\mathsf{Sampler}, \mathsf{Eval})$ with the following syntax:

1. $\mathsf{Sampler}(1^n) \to i$: On input 1^n, $\mathsf{Sampler}$ outputs an index $i \in I_n$.
2. $\mathsf{Eval}(i, x) = h_i(x)$: On input i and $x \in D_n$, Eval algorithm outputs $h_i(x)$.
3. $\forall$ PPT $\mathcal{A}$ we have

$$\Pr[i \leftarrow \mathsf{Sampler}(1^n), (x, x') \leftarrow \mathcal{A}(1^n, i) : h_i(x) = h_i(x') \wedge x \neq x'] \leq \mathsf{negl}(n)(n)$$

5.3.2 *Collision Resistant Hash functions from Discrete Log*

We will now give a construction of collision resistant hash functions from the discrete log assumption. We first recall the discrete log assumption:

Definition 5.1 (Discrete-Log Assumption). *We say that the discrete-log assumption holds for the group ensemble $\mathcal{G} = \{\mathbb{G}_n\}_{n \in \mathbb{N}}$, if for every non-uniform PPT algorithm $\mathcal{A}$ we have that*

$$\mu_{\mathcal{A}}(n) := \Pr_{x \leftarrow |\mathbb{G}_n|}[\mathcal{A}(g, g^x) = x]$$

is a negligible function.

We now give a construction of collision resistant hash functions.

- $\mathsf{Sampler}(1^n)$: On input 1^n, the sampler does the following:
  1. It chooses $x \leftarrow |\mathbb{G}_n|$.
  2. It computes $h = g^x$.
  3. It outputs (g, h).
- $\mathsf{Eval}((g, h), (r, s))$: On input (g, h) and two elements $(r, s) \in |\mathbb{G}_n|$, Eval outputs $g^r h^s$.

We now argue that this construction is collision resistant. Assume for the sake of contradiction that an adversary gives a collision $(r_1, s_1) \neq (r_2, s_2)$. We will now use this to compute the discrete logarithm of h. We first observe that:

$$\begin{aligned} r_1 + xs_1 &= r_2 + xs_2 \\ (r_1 - r_2) &= x(s_2 - s_1) \end{aligned}$$

We infer that $s_2 \neq s_1$. Otherwise, we get that $r_1 = r_2$ and hence, $(r_1, s_1) = (r_2, s_2)$. Thus, we can compute $x = \frac{r_1 - r_2}{s_1 - s_2}$ and hence the discrete logarithm of h is computable.

5.4 Multiple-Message Digital Signature

We now explain how to combine collision-resistant hash functions and one-time signatures to get a signature scheme for multiple messages. We first construct an intermediate primitive wherein we will still have the same security property as that of one-time signature but we would be able to sign messages longer than the length of the public-key.[1]

[1] Note that in the one-time signature scheme that we constructed earlier, the length of message that can be signed is same as the length of the public-key.

5.4.1 One-time Signature Scheme for Long Messages

We first observe that the CRHF family H that we constructed earlier compresses $2n$ bits to n bits (also called as 2-1 CRHF). We will now give an extension that compresses an arbitrary long string to n bits using a 2-1 CRHF.

Merkle-Damgard CRHF. The sampler for this CRHF is same as that of 2-1 CRHF. Let h be the sampled hash function. To hash a string x, we do the following. Let x be a string of length m where m is an arbitrary polynomial in n. We will assume that $m = kn$ (for some k) or otherwise, we can pad x to this length. We will partition the string x into k blocks of length n each. For simplicity, we will assume that k is a perfect power of 2 or we will again pad x appropriately. We will view these k-blocks as the leaves of a complete binary tree of depth $\ell = \log_2 k$. Each intermediate node is associated with a bit string y of length at most ℓ and the root is associated with the empty string. We will assign a $\mathsf{tag} \in \{0,1\}^n$ to each node in the tree. The i-th leaf is assigned tag_i equal to the i-block of the string x. Each intermediate node y is assigned a $\mathsf{tag}_y = h(\mathsf{tag}_{y\|0} \| \mathsf{tag}_{y\|1})$. The output of the hash function is set to be the tag value of the root. Notice that if there is a collision for this CRHF then there are exists one intermediate node y such that for two different values $\mathsf{tag}_{y\|0}, \mathsf{tag}_{y\|1}$ and $\mathsf{tag}'_{y\|0}, \mathsf{tag}'_{y\|1}$ we have, $h(\mathsf{tag}_{y\|0}, \mathsf{tag}_{y\|1}) = \mathsf{tag}'_{y\|0}, \mathsf{tag}'_{y\|1}$. This implies that there is a collision for h.

Construction. We will now use the Merkle-Damgard CRHF and the one-time signature scheme that we constructed earlier to get a one-time signature scheme for signing longer messages. The main idea is simple: we will sample a (sk, vk) for signing n-bit messages and to sign a longer message, we will first hash it using the Merkle-

Damgard hash function to n-bits and then sign on the hash value. The security of the construction follows directly from the security of the one-time signature scheme since the CRHF is collision-resistant.

5.4.2 *Signature Scheme for Multiple Messages*

We will now describe the construction of signature scheme for multiple messages. Let $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify}')$ be a one-time signature scheme for signing longer messages.

1. $\mathsf{Gen}(1^n)$: Run $\mathsf{Gen}'(1^n)$ using to obtain sk, vk. Sample a PRF key K. The signing key is (sk, K) and the verification key is vk.
2. $\mathsf{Sign}((sk, K), m)$: To sign a message m, do the following:
   (a) Parse m as $m_1 m_2 \ldots m_\ell$ where each $m_i \in \{0, 1\}$.
   (b) Set $sk_0 = sk$ and $m_0 = \epsilon$ (where ϵ is the empty string).
   (c) For each $i \in [\ell]$ do:
      i. Evaluate $\mathsf{PRF}(m_1 \| \ldots \| m_{i-1} \| 0)$ and $\mathsf{PRF}(m_1 \| \ldots \| m_{i-1} \| 1)$ to obtain r_0 and r_1 respectively. Run $\mathsf{Gen}'(1^n)$ using r_0 and r_1 as the randomness to obtain $(sk_{i,0}, vk_{i,1})$ and $(sk_{i,1}, vk_{i,1})$.
      ii. Set $\sigma_i = \mathsf{Sign}(sk_{i-1,m_{i-1}}, vk_{i,0} \| vk_{i,1})$
      iii. If $i = \ell$, then set $\sigma_{\ell+1} = \mathsf{Sign}(sk_{i,m_i}, m)$.
   (d) Output $\sigma = (\sigma_1, \ldots, \sigma_{\ell+1})$ along with all the verification keys as the signature.
3. $\mathsf{Verify}(vk, \sigma, m)$: Check if all the signatures in σ are valid.

To prove security, we will first use the security of the PRF to replace the outputs with random strings. We will then use the security of the one-time signature scheme to argue that the adversary cannot mount an existential forgery.

Exercises

Exercise 5.1. ***Digital signature schemes can be made deterministic.*** *Given a digital signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *for which* Sign *is probabilistic, provide a construction of a digital signature scheme* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify}')$ *where* Sign' *is deterministic.*

5.5 *Cramer-Shoup Construction*

Bibliography

Mihir Bellare. A note on negligible functions. *Journal of Cryptology*, 15 (4):271–284, September 2002. DOI: 10.1007/s00145-002-0116-x.