CS 355: Topics in Cryptography **Spring 2019**

Lecture 2: Basic Cryptographic Primitives, Hybrid Argument

Dima Kogan

Introduction

Ideally, we would like to prove the security of our cryptographic constructions unconditionally. However, in most cases, such a proof would imply that $\mathsf{P} \neq \mathsf{NP}$. Why? An unconditional security proof for some functionality would establish the following: the functionality can be efficiently computed given the secret key, whereas without access to the secret key, there is no efficient algorithm for the functionality. This roughly corresponds to a problem which is in NP but not in P (and not even in BPP).

Since we do not expect to resolve the P vs. NP problem anytime soon, we cannot hope for unconditional proofs of security. What would be the next best thing then? Perhaps, proving the security under the assumption that $\mathsf{P} \neq \mathsf{NP}$. For instance, we would hope to show that, there is no efficient adversary for our encryption scheme, as long as there are no polynomial-time algorithms for 3-SAT. We do not know how to this (it is a huge open problem). The problem is that NP only capture hardness in the worst case, whereas for crypto, we need our problems to be hard on the average case (e.g., decryption should be hard for most choices of the secret key, rather than only in the worst case).

1 One-way functions

The basic form of computational hardness we require is a one-way function.

Definition 1.1 (One-way function)**.** A function ensemble $\left\{f_\lambda \colon \{0,1\}^{n(\lambda)} \to \{0,1\}^{\ell(\lambda)}\right\}_{\lambda\in\mathbb{N}}$ is one-way if

1. there exists an efficient algorithm that given λ and x computes $f_\lambda(x)$, and
2. for every efficient algorithm $\mathcal{A}$, and sufficiently large $\lambda \in \mathbb{N}$, it holds that

$$\Pr\left[f(\mathcal{A}(f(x)) = f(x) \colon x \xleftarrow{R} \{0,1\}^n\right] < \mathrm{negl}(\lambda)\,.$$

A few remarks are in order:

1. Why do we require $f(\mathcal{A}(f(x))) = f(x)$ instead of $\mathcal{A}(f(x)) = x$? As f is not necessary one-to-one, the point $f(x)$ might have more than one preimage, and we require the adversary to find any one of them.
2. Why do we speak of function ensembles, rather than of simply a single function $f \colon \{0,1\}^n \to \{0,1\}^\ell$? Our definition of negligible probability is asymptotic, so a one-way function has to be defined for infinitely many values of λ. In fact, strictly speaking, a function with a finite image (e.g., SHA-256) cannot be one-way in this sense, since an algorithm can invert it with probability 1 by having in its code a preimage for every point in the image of the function. The code would be huge (and impractical), yet finite. However, in many cases, we will ignore this important, yet technical, aspect in our definitions, and we will write $f \colon \{0,1\}^n \to \{0,1\}^\ell$, where in fact we refer to a function ensemble parameterized by a security parameter λ, and $n = n(\lambda)$ and $\ell = \ell(\lambda)$ are some polynomials in λ.

As we said, we cannot prove that one-way functions exist, yet we have several plausible candidates. For example, variations of factoring and modular exponentiation (with inverse being the discrete logarithm problem) are all plausible one-way functions. It turns out that one-way functions are surprisingly powerful, and are sufficient to construct most symmetric cryptographic primitives. The following is known to hold:

$$OWF \Rightarrow PRG \Rightarrow PRF \Rightarrow \text{block ciphers, MACs}.$$

Where each arrow means that we can construct one primitive from the other one. The other direction is also true (and is much easier to show), so each of these primitives is in-fact sufficient and necessary for symmetric-key cryptography.

In CS255 you have seen how to construct block ciphers and MACs from PRFs, and in this class we'll focus on the other two steps.

2 Pseudorandom Generators

Definition 2.1. Pseudorandom generator An efficient algorithm G is a pseudorandom generator if it:

1. stretches its input: for $s \in \{0,1\}^n$, it holds that $G(s) \in \{0,1\}^\ell$ for $\ell > n$, and
2. pseudorandom: for every efficient algorithm $\mathcal{A}$, it holds that

$$\left|\Pr\left[\mathcal{A}(G(s)) = 1 \colon s \xleftarrow{R} \{0,1\}^n\right] - \Pr\left[\mathcal{A}(y) = 1 \colon y \xleftarrow{R} \{0,1\}^\ell\right]\right| < \text{negl}(n).$$

We can also rephrase the second requirement in terms of computational indistinguishability of probabilty distributions.

Definition 2.2 (Computational indistinguishability)**.** Let $\mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda\in\mathbb{N}}$ and $\mathcal{D}_2 = \{\mathcal{D}_{2,\lambda}\}_{\lambda\in\mathbb{N}}$ be two collections (ensembles) of probability distributions. We say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable (denoted $\mathcal{D}_1 \approx_c \mathcal{D}_2$), if for all efficient aglorithms $\mathcal{A}$,

$$\left|\Pr\left[A(1^\lambda, x) = 1 \colon x \xleftarrow{R} \mathcal{D}_{1,\lambda}\right] - \Pr\left[A(1^\lambda, x) = 1 \colon x \xleftarrow{R} \mathcal{D}_{2,\lambda}\right]\right| < \text{negl}(\lambda).$$

We can then rephrase the second requirement in Definition 2.1 as requiring that the ensembles $\{G(s) \colon s \xleftarrow{R} \{0,1\}^n\}_{\lambda\in\mathbb{N}}$ and $\{y \xleftarrow{R} \{0,1\}^\ell\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable.

Constructing a PRG from a OWF. We consider the simpler case of constructing a PRG from a *one-way permutation*. Let f be a one-way permutation, i.e., it is one-way (as defined in Definition 1.1), and additionally, for every $n \in \mathbb{N}$, $f \colon \{0,1\}^n \to \{0,1\}^n$ is a permutation on the elements of $\{0,1\}^n$. Then

$$G(r,x) = r \| f(x) \| \langle x, r\rangle$$

where $r, x \in \{0,1\}^n$, is a pseudorandom generator $\{0,1\}^{2n} \to \{0,1\}^{2n+1}$. Here $\|$ denotes string concatenation, and $\langle\cdot,\cdot\rangle$ denotes inner-product modulo 2. Since f is a permutation, $r\|f(x)$ is a truly random string of length $2n$. The heart of the proof is to show that $\langle x, r\rangle$ is unpredictable. This is called the Goldreich-Levin theorem. You will see a (relatively simple) variant of it in Homework 1.

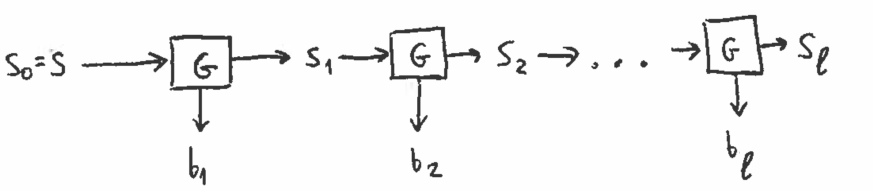
2.1 Increasing the stretch

We now show how to construct PRGs or polynomially large stretch from the above PRG that only stretches its input by a single bit. We use this proof as an opportunity to present a very important proof technique in cryptography, called a "Hybrid Argument".

The Blum-Micali PRG. Let G be a PRG with stretch 1, i.e., $|G(s)| = |s| + 1$, where $|\cdot|$ denotes the length of a string. Let $\ell(n)$ be any polynomial. We construct a PRG G' as follows:

on input $s \in \{0,1\}^n$:

- $s_0 \leftarrow s$
- $s_i b_i \leftarrow G(s_{i-1})$ for $i = 1, 2, \ldots, \ell$
- output $b_1, b_2, \ldots, b_\ell$



Theorem 2.3. *G' is a pseudorandom generator.*

Proof. Clearly, the PRG G' stretches its input from n to $\ell(n)$. Moreover, since G is efficient, than G' is also efficient: its running time is $t'(n) = t(n) \cdot \ell(n) + O(n)$ where $t(n)$ is the running time of G. Since $t(n)$ and $\ell(n)$ are both polynomials in n, so is $t'(n)$.

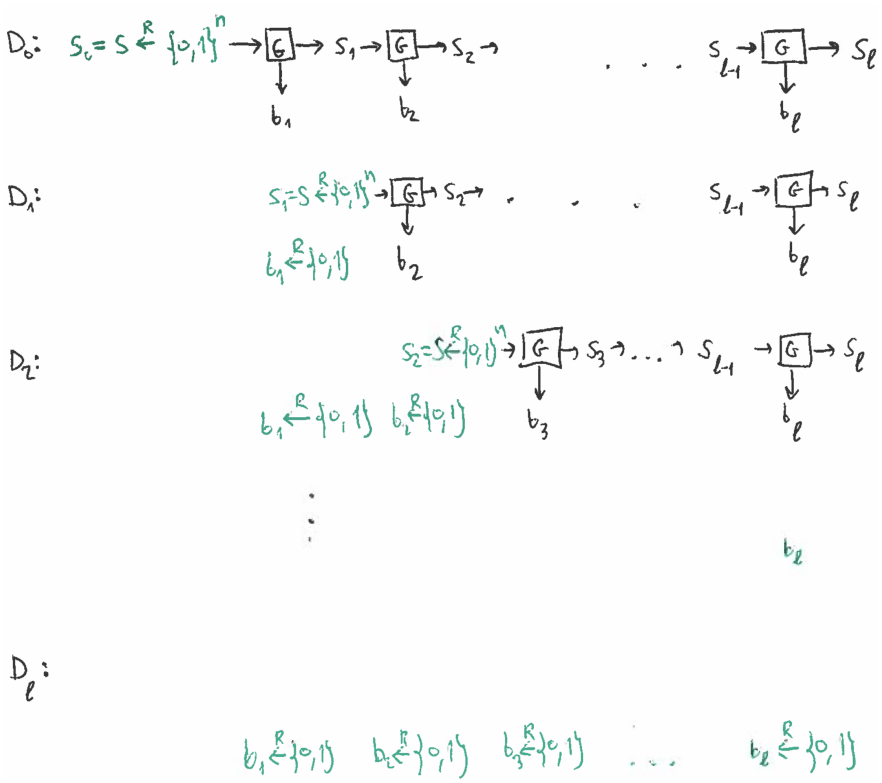
It remains to show that G' is indeed pseudorandom. We prove by contradiction: let $\mathcal{A}$ be an efficient algorithm that breaks G', namely assume

$$\left| \Pr\left[\mathcal{A}(G'(s)) = 1 : s \xleftarrow{R} \{0,1\}^n\right] - \Pr\left[\mathcal{A}(y) = 1 : y \xleftarrow{R} \{0,1\}^\ell\right] \right| = \epsilon(n)$$

for some *non-negligible* function $\epsilon(n)$.

We define the hybrid distributions $\{\mathcal{D}_i\}_{i=0}^{\ell}$, each over the set $\{0,1\}^\ell$, where $\mathcal{D}_i$ is obtained as follows:

- for $j = 1$ to i, let $b_i \xleftarrow{R} \{0,1\}$[1]
- let $s_i \xleftarrow{R} \{0,1\}^n$
- for $j = i+1, 2, \ldots, \ell$ set $s_j b_j \leftarrow G(s_{j-1})$
- output $b_1, b_2, \ldots, b_\ell$



Note that $\mathcal{D}_0$ is exactly the output distribution of G': $\{G'(s)\colon s \xleftarrow{R} \{0,1\}^n\}$, and $\mathcal{D}_\ell$ is exactly the truly random distribution $\{y \xleftarrow{R} \{0,1\}^\ell\}$.

Denote $p_i = \Pr[\mathcal{A}(y) = 1\colon y \xleftarrow{R} \mathcal{D}_i]$. By our assumption on $\mathcal{A}$ it thus holds $|p_0 - p_\ell| = \epsilon$ where ϵ is non-negligible. The intuition now is that there must exist an index $i \in \{1,2,\ldots,\ell\}$ and a neighboring pair of hybrids $\mathcal{D}_{i-1}, \mathcal{D}_i$ such that $|p_i - p_{i-1}|$ is also non-negligible. We can then use the adversary $\mathcal{A}$ and such an i to create a distinguisher $\mathcal{B}$ for G. In fact, a random i will be enough.

Algorithm $\mathcal{B}$ works as follows:

on input $z \in \{0,1\}^{n+1}$,

1. parse z as (s,b) where $s \in \{0,1\}^n$ and $b \in \{0,1\}$
2. set $I \xleftarrow{R} \{1,2,\ldots,\ell\}$
3. for $j = 1$ to $I-1$, let $b_j \xleftarrow{R} \{0,1\}$
4. set $b_I \leftarrow b$ and $s_I \leftarrow s$
5. for $j = I+1, 2, \ldots, \ell$ set $s_j b_j \leftarrow G(s_{j-1})$
6. Set $y = b_1, b_2, \ldots, b_\ell$, run and output $\mathcal{A}(y)$

We now observe the following. When $\mathcal{B}$ is given as input $z \xleftarrow{R} \{0,1\}^{n+1}$, and chooses a value $I = i$ in step 2 then it runs $\mathcal{A}$ on an input y sampled from the distribution $\mathcal{D}_i$. Conversely, when $\mathcal{B}$ is given an input

[1] We slightly abuse notation, and here and elsewhere the loop "for $j = 1$ to i" does not execute at all when $i = 0$.

$G(s)$ for $s \xleftarrow{R} \{0,1\}^n$, then it runs $\mathcal{A}$ on in input y sampled from the distribution $\mathcal{D}_{i-1}$. Using our notation

$$\left\{y \mid I = i : z \xleftarrow{R} \{0,1\}^{n+1}\right\} \approx \mathcal{D}_i$$
$$\left\{y \mid I = i : z = G(s), s \xleftarrow{R} \{0,1\}^n\right\} \approx \mathcal{D}_{i-1}$$

It therefore holds:

$$\begin{aligned}
&\left|\Pr\left[\mathcal{B}(z) = 1 : z \xleftarrow{R} \{0,1\}^{n+1}\right] - \Pr\left[\mathcal{B}(G(s)) = 1 : s \xleftarrow{R} \{0,1\}^n\right]\right| \\
&= \left|\sum_{i=1}^{\ell} \Pr\left[\mathcal{B}(z) = 1 \mid I = i : z \xleftarrow{R} \{0,1\}^{n+1}\right] \cdot \Pr[I = i] - \Pr\left[\mathcal{B}(G(s)) = 1 \mid I = i : s \xleftarrow{R} \{0,1\}^n\right] \cdot \Pr[I = i]\right| \\
&= \left|\sum_{i=1}^{\ell} \Pr\left[\mathcal{A}(y) = 1 : y \xleftarrow{R} \mathcal{D}_i\right] \cdot \tfrac{1}{\ell} - \Pr\left[\mathcal{A}(y) = 1 : y \xleftarrow{R} \mathcal{D}_{i-1}\right] \cdot \tfrac{1}{\ell}\right| \\
&= \tfrac{1}{\ell} \cdot \left|\sum_{i=1}^{\ell} p_i - p_{i-1}\right| = \tfrac{1}{\ell} \cdot |p_\ell - p_0| = \frac{\epsilon}{\ell}.
\end{aligned}$$

The first equality above holds by the law of total probability, and the second to last inequality holds since the sum is telescoping.

The advantage $\epsilon(n)$ is non-negligible, and the stretch $\ell(n)$ is a polynomial. Therefore $\epsilon(n)/\ell(n)$ is non-negligible, so we obtain a algorithm $\mathcal{B}$ with a non-negligible distingushing probability for PRG G,which is a contradiction to the premise of the theorem. □

Recap: Hybrid Argument. Abstractly, the key steps in our hybrid argument were:

1. Construct a sequence of polynomially many distributions between the two target distributions.
2. Argue that each pair of neighboring distributions are indistinguishable.
3. Conclude that the target distributions are indistinguishable.

The last two steps are often done using a proof by contradiction.

3 Pseudorandom functions

Let $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function. Consider the following attack game:

1. The challenger chooses $b \xleftarrow{R} \{0,1\}$. If $b = 0$, the challenger chooses $k \xleftarrow{R} \mathcal{K}$ and sets $f \leftarrow F(k, \cdot)$. If $b = 1$, the challenger chooses f at random $f \xleftarrow{R} \mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$.
2. The adversary can repeatedly query f on any point x, and the challenger responds with $f(x)$.
3. The adversary outputs $b' \in \{0,1\}$.

We define the adversary's advantage as:

$$\mathsf{PRFAdv}[\mathcal{A}, F] = \left|\Pr[b' = b] - \tfrac{1}{2}\right|.$$

Definition 3.1. A PRF is an efficient deterministic algorithm F that takes two inputs: a key $k \in \mathcal{K}$ and an input $x \in \mathcal{X}$, and outputs $y = F(k, x) \in \mathcal{Y}$, such that for every efficient adversary $\mathcal{A}$ the advantage $\mathsf{PRFAdv}[\mathcal{A}, F]$ is negligible.

Remark. As before, strictly speaking, the sets $\mathcal{K}$, $\mathcal{X}$, and $\mathcal{Y}$ have to be defined as ensembles of sets parameterized by the security parameter λ.

3.1 The GGM construction

Suppose we have a length-doubling PRG $G\colon \{0,1\}^n \to \{0,1\}^{2n}$. We can construct a PRF $F\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ as follows. For the sake of simplicity, we assume that n is a power of 2. Since G is length doubling, for a seed $s \in \{0,1\}^n$, we can write $G(s) = (s_0, s_1) = (G_0(s), G_1(s))$, where $G_0, G_1 \in \{0,1\}^n \to \{0,1\}^n$. For every $k, x \in \{0,1\}^n$,w define $F(x) := G_{x_n}(G_{x_{n-1}}(\ldots(G_{x_1}(k))\ldots))$.

The proof of security uses a hybrid argument, and can be found in the Boneh-Shoup book, Chapter 4.6.