Cryptography

Michele Laurenti

January 5, 2017

Abstract

Notes taken during the Cryptography lectures held by Daniele Venturi (`http://danieleventuri.altervista.org/crypto.shtml`) in fall 2016 at Sapienza.

Contents

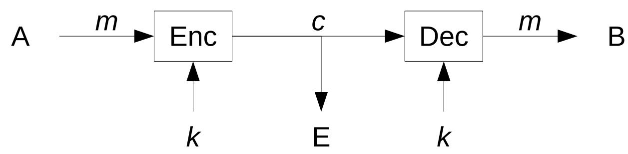

Figure 1: Message exchange between A and B using symmetric encryption. E is the eavesdropper.

1 Introduction

Solomon,
I'm concerned about security; I think, when we email each other,
we should use some sort of code.

Confidentiality is our goal. We want to encrypt and decrypt a (plaintext) message m, using a key, to obtain a cyphertext c. As per Kirkoff's principle, only the key is secret.

Our encryption schemes have the following syntax:

$$\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})\,.$$

A and B, the actors of our communication exchange (fig. 1), share k, the key, taken from some key space $\mathcal{K}$. The elements of our encryption scheme play the following roles:

1. Gen outputs a random key from the key space $\mathcal{K}$, and we write this as $k \leftarrow \$\mathrm{Gen}$;
2. $\mathrm{Enc} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ is the encryption function, mapping a key and a message to a cyphertext;
3. $\mathrm{Dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$ is the decryption function, mapping a key and a cyphertext to a message.

We expect an encryption scheme to be at least correct:

$$\forall k \in \mathcal{K}, \forall m \in \mathcal{M}. \mathrm{Dec}(k, \mathrm{Enc}(k, m)) = m.$$

1.1 Perfect secrecy

Shannon defined "perfect secrecy", *i.e.*, the fact that the cyphertext carries no information about the plaintext.

Definition 1. [Perfect secrecy] Let M be a random variable (RV) over $\mathcal{M}$, and K be a uniform distribution over $\mathcal{K}$.

$(\mathrm{Enc}, \mathrm{Dec})$ has perfect secrecy if

$$\forall M, \forall m \in \mathcal{M}, c \in \mathcal{C}. \Pr\left[M = m\right] = \Pr\left[M = m | C = c\right]$$

where $C = \mathrm{Enc}(k, m)$ is a third RV. $\diamond$

We have equivalent definitions for perfect secrecy.

Theorem 1. *The following definitions are equivalent:*

1. *definition 1;*
2. *M and C are independent;*
3. *$\forall m, m' \in \mathcal{M}, \forall c \in \mathcal{C}$*

   $$\Pr\left[\mathrm{Enc}(k, m) = c\right] = \Pr\left[\mathrm{Enc}(k, m') = c\right]$$

   where k is a random key in $\mathcal{K}$ chosen with uniform probability. $\diamond$

Proof of theorem 1. First, we show that 1 implies 2.

$$\begin{aligned}\Pr\left[M = m\right] &= \Pr\left[M = m | C = c\right] \\ &= \frac{\Pr\left[M = m \wedge C = c\right]}{\Pr\left[C = c\right]} && \text{(by Bayes)}\end{aligned}$$

$$\Longrightarrow$$

$$\Pr\left[M = m\right] \Pr\left[C = c\right] = \Pr\left[M = m \wedge C = c\right]$$

which is the definition of independence.

Now we show that 2 implies 3. Fix $m \in \mathcal{M}$ and $c \in \mathcal{C}$.

$$\begin{aligned}\Pr\left[\mathrm{Enc}(k, m) = c\right] &= \Pr\left[\mathrm{Enc}(k, M) = c | M = m\right] && \text{(we fixed } m\text{)} \\ &= \Pr\left[C = c | M = m\right] && \text{(definition of the RV } C\text{)} \\ &= \Pr\left[C = c\right]. && \text{(by 2)}\end{aligned}$$

Since m is arbitrary, we can do the same for m', and obtain

$$\Pr\left[\mathrm{Enc}(k, m') = c\right] = \Pr\left[C = c\right]$$

which gives us 3.

Now we want to show that 3 implies 1. Take any $c \in \mathcal{C}$.

$$
\begin{aligned}
\Pr[C=c] &= \sum_{m' \in \mathcal{M}} \Pr[C=c \wedge M=m'] \\
&= \sum_{m' \in \mathcal{M}} \Pr[C=c|M=m'] \Pr[M=m'] && \text{(by Bayes)} \\
&= \sum_{m' \in \mathcal{M}} \Pr[\mathrm{Enc}(k,M)=c|M=m'] \Pr[M=m'] \\
&= \sum_{m' \in \mathcal{M}} \Pr[\mathrm{Enc}(k,m')=c] \Pr[M=m'] \\
&= \Pr[\mathrm{Enc}(k,m)=c] \underbrace{\sum_{m' \in \mathcal{M}} \Pr[M=m']}_{1} \\
& && \text{(Enc is indepenendent of } M\text{, so we take it out)} \\
&= \Pr[\mathrm{Enc}(k,M)=c|M=m] = \Pr[C=c|M=m].
\end{aligned}
$$

We are left to show that $\Pr[M=m] = \Pr[M=m|C=c]$, but this is easy with Bayes. □

One Time Pad

Now we'll see a perfect encryption scheme, the One Time Pad (OTP). The message space, the cyphertext space, and the key space are all the same, *i.e.*, $\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0,1\}^l$, with $l \in \mathbb{N}^+$.

Encryption and decryption use the xor operation:

- $\mathrm{Enc}(k,m) = k \oplus m = c$;
- $\mathrm{Dec}(k,c) = c \oplus k = (k \oplus m) \oplus k = m$.

Seeing that this is correct is immediate.

This can actually be done in any finite abelian group $(\mathbb{G}, +)$, where you just do $k+m$ to encode and $c-k$ to decode.

Theorem 2. *OTP is perfectly secure.* ◇

Proof of theorem 2. Fix $m \in \mathcal{M}, c \in \mathcal{C}$, and choose a random key.

$$\Pr[\mathrm{Enc}(k,m)=c] = \Pr[k=c-m] = \frac{1}{|\mathbb{G}|}.$$

This is true for any m, so we are done. □

OTP has two problems:

1. the key is long (as long as the message);

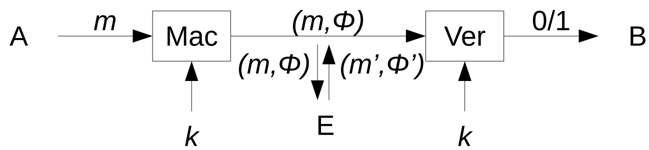

Figure 2: Message exchange between A and B using symmetric authentication. E is the eavesdropper.

2. we can't reuse the key:

$$\begin{array}{c} c = k + m \\ c' = k + m' \end{array} \implies c - c' = m - m' \implies m' = m - (c - c').$$

Theorem 3. *[Shannon, 1949] In any perfectly secure encryption scheme the size of the key space is at least as large as the size of the message space, i.e.,* $|\mathcal{K}| \geq |\mathcal{M}|$. ◇

Proof of theorem 3. Assume, for the sake of contradiction, that $|\mathcal{K}| < |\mathcal{M}|$. Fix M to be the uniform distribution over $\mathcal{M}$, which we can do as perfect secrecy works for any distribution. Take a cyphertext $c \in \mathcal{C}$ such that $\Pr[C = c] > 0$, *i.e.*, $\exists m, k$ such that $\mathrm{Enc}(k, m) = c$.

Consider $\mathcal{M}' = \{\mathrm{Dec}(k, c) : k \in \mathcal{K}\}$, the set of all messages decrypted from c using any key. Clearly, $|\mathcal{M}'| \leq |\mathcal{K}| < |\mathcal{M}|$, so $\exists m' \in \mathcal{M}$ such that $m' \notin \mathcal{M}'$. This means that

$$\Pr[M = m'] = \frac{1}{|\mathcal{M}|} \neq \Pr[M = m' | C = c] = 0$$

in contradiction with perfect secrecy. □

In the rest of the course we will forget about perfect secrecy, and strive for computational security, *i.e.*, bound the computational power of the adversary.

1.2 Authentication

The aim of authentication is to avoid tampering of E with the messages exchanged between A and B (fig. 2).

A Message Authentication Code (MAC) is defined as a tuple $\Pi = (\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$, where:

- Gen, as usual, outputs a random key from some key space $\mathcal{K}$;
- $\mathrm{Mac} : \mathcal{K} \times \mathcal{M} \to \Phi$ maps a key and a message to an authenticator in some authenticator space Φ;
- $\mathrm{Vrfy} : \mathcal{K} \times \mathcal{M} \times \Phi \to \{0, 1\}$ verifies the authenticator.

As usual, we expect a MAC to be correct, *i.e.*,

$$\forall m \in \mathcal{M}, \forall k \in \mathcal{K}. \mathrm{Vrfy}(k, m, \mathrm{Mac}(k, m)) = 1.$$

If the Mac function is deterministic, then it must be that $\mathrm{Vrfy}(k, m, \phi) = 1$ if and only if $\mathrm{Mac}(k, m) = \phi$.

Security for MACs is that *forgery* must be hard: you can't come up with an authenticator for a message if you don't know the key.o

Definition 2. [Information theoretic MAC] $(\mathrm{Mac}, \mathrm{Vrfy})$ has ε-statistical security if for all (possibly unbounded) adversary $\mathcal{A}$, for all $m \in \mathcal{M}$,

$$\Pr\left[\mathrm{Vrfy}(k, m', \phi') = 1 \wedge m' \neq m : \begin{array}{l} k \leftarrow \$\mathrm{KeyGen}; \\ \phi = \mathrm{Mac}(k, m); \\ (m', \phi') \leftarrow \mathcal{A}(m, \phi) \end{array}\right] \leq \varepsilon$$

i.e., the adversary forges a (m', ϕ') that verifies with key k with low probability, even if it knows a valid pair (m, ϕ). $\diamond$

As an exercise, prove that the above is impossible if $\varepsilon = 0$.

Information theoretic security is also called unconditional security. Later we'll see *conditional* security, based on computational assumptions.

Definition 3. [Pairwise independence] Given a family $\mathcal{H} = \{h_k : \mathcal{M} \to \Phi\}_{k \in \mathcal{K}}$ of functions, we say that $\mathcal{H}$ is pairwise independent if for all distinct m, m' we have that $(h_k(m), h_k(m')) \in \Phi^2$ is uniform over the choice of $k \leftarrow \$\mathcal{K}$. $\diamond$

We say straight away a construction of a pairwise independent family of function. Let p be a prime, the functions in our family will be

$$h_{a,b}(m) = am + b \mod p$$

with $\mathcal{K} = \mathbb{Z}_p^2$, and with $\mathcal{M} = \Phi = \mathbb{Z}_p$.

Theorem 4. *The above construction is pairwise independent.* $\diamond$

Proof of theorem 4. For any m, m', ϕ, ϕ', we want to find the value of

$$\Pr\left[am + b = \phi \wedge am' + b = \phi'\right]$$

for $a, b \leftarrow \$\mathbb{Z}_p^2$. This is the same as

$$\Pr_{a,b}\left[\begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \phi \\ \phi' \end{pmatrix}\right] = \Pr_{a,b}\left[\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix}^{-1}\begin{pmatrix} \phi \\ \phi' \end{pmatrix}\right] = \frac{1}{|\Phi|^2}.$$

This is true since $\left(\begin{smallmatrix} m & 1 \\ m' & 1 \end{smallmatrix}\right)^{-1}\left(\begin{smallmatrix} \phi \\ \phi' \end{smallmatrix}\right)$ is just a couple of (constant) numbers, so the probability of choosing (a, b) such that they equal the constant is just $\frac{1}{|\Phi|^2}$. $\square$

If h_k is part of a pairwise independent family of functions, then $\mathrm{Mac}(k, m) = h_k(m)$, and $\mathrm{Vrfy}(k, m, \phi)$ is simply computing $h_k(m)$ and comparing it with ϕ, *i.e.*,

$$\mathrm{Vrfy}(k, m, \phi) = 1 \iff h_k(m) = \phi.$$

We now prove that this is an information theoretic MAC.

Theorem 5. *Any pairwise independent function is $\frac{1}{|\Phi|}$-statistical secure.* $\diamond$

Proof of theorem 5. Take any two distinct m, m', and two ϕ, ϕ'. We show that the probability that $\mathrm{Mac}(k, m') = \phi'$ is exponentially small.

$$\Pr_k [\mathrm{Mac}(k, m) = \phi] = \Pr_k [h_k(m) = \phi] = \frac{1}{|\Phi|}.$$

Now look at the joint probabilities:

$$\begin{aligned}\Pr_k [\mathrm{Mac}(k, m) = \phi \wedge \mathrm{Mac}(k, m') = \phi'] &= \Pr_k [h_k(m) = \phi \wedge h_k(m') = \phi'] \\ &\qquad \text{(by definition)} \\ &= \frac{1}{|\Phi|^2} = \frac{1}{|\Phi|} \cdot \frac{1}{|\Phi|}.\end{aligned}$$

The last steps come from the fact that h_k is pairwise independent. To see that the construction is $\frac{1}{|\Phi|}$-statistical secure:

$$\begin{aligned}\Pr_k [\mathrm{Mac}(k, m') = \phi' | \mathrm{Mac}(k, m) = \phi] &= \Pr_k [h_k(m') = \phi' | h_k(m) = \phi] \\ &= \frac{\Pr_k [h_k(m) = \phi \wedge h_k(m') = \phi']}{\Pr_k [h_k(m) = \phi]} \\ &= \frac{1}{|\Phi|}.\end{aligned}$$

$\square$

Note that the previous construction ($h_k(m) = am + b \mod p$) is insecure if the same key $k = (a, b)$ is used for two messages.

Theorem 6. *Any t-time $2^{-\lambda}$-statistically secure MAC has key of size $(t + 1)\lambda$.* $\diamond$

1.3 Randomness Extraction

X is a random source (possibly not uniform). $\mathrm{Ext}(X) = Y$ is a uniform RV.

First, let's see a construction for a binary RV. Let B be a RV such that $\Pr [B = 1] = p$ and $\Pr [B = 0] = 1 - p$, with $p \neq 1 - p$. We take two samples, B_1 and B_2 from B, and we want to obtain an unbiased RV B'.

1. Take two samples, $b_1 \leftarrow \$ B_1$ and $b_2 \leftarrow \$ B_2$;
2. if $b_1 = b_2$, sample again;
3. if $(b_1 = 1 \wedge b_2 = 0)$, output 1; if $(b_1 = 0 \wedge b_2 = 1)$, output 0.

It's easy to verify that B' is uniform:

$$\Pr[B' = 1] = \Pr[B_1 = 1 \wedge B_2 = 0] = p(1-p)$$
$$\Pr[B' = 0] = \Pr[B_1 = 0 \wedge B_2 = 1] = (1-p)p.$$

How many trials do we have to make before outputting something? $2(1-p)p$ is the probability that we output something. The probability that we don't output anything for n steps is thus $(1 - 2(1-p)p)^n$.

2 Computational Cryptography

To introduce computational cryptography we first have to define a computational model. We assume the adversary is efficient, *i.e.*, it is a Probabilistic Polynomial Time (PPT) adversary.

We want that the probability of success of the adversary is tiny, *i.e.*, negligible for some $\lambda \in \mathbb{N}$. A function $\varepsilon : \mathbb{N} \to \mathbb{R}$ is negligible if $\forall c > 0. \exists n_0$ such that $\forall n > n_0. \varepsilon(n) < n^{-c}$.

We rely on computational assumptions, *i.e.*, in tasks believed to be hard for any efficient adversary. In this setting we make conditional statements, *i.e.*, if a certain assumption holds then a certain crypto-scheme is secure.

2.1 One Way Functions

A simple computational assumption is the existence of One Way Functions (OWFs), *i.e.*, functions for which is hard to compute the inverse.

Definition 4. [One Way Function] A function $f : \{0,1\}^\star \to \{0,1\}^\star$ is a OWF if $f(x)$ can be computed in polynomial time for all x and for all PPT adversaries $\mathcal{A}$ it holds that

$$\Pr\left[f(x') = y : x \leftarrow \$ \{0,1\}^\star;\ y = f(x);\ x' \leftarrow \mathcal{A}(1^\lambda, y)\right] \leq \varepsilon(\lambda). \qquad \diamond$$

The 1^λ given to the adversary $\mathcal{A}$ is there to highlight the fact that $\mathcal{A}$ is polynomial in the length of the input (λ).

Russel Impagliazzo proved that OWFs are equivalent to One Way Puzzles, *i.e.*, couples (Pgen, Pver) where $\text{Pgen}(1^\lambda) \to (y, x)$ gives us a puzzle (y) and a solution to it (x), while $\text{Pver}(x, y) \to 0/1$ verifies if x is a solution of y.

Another object of interest in this classification are average hard NP-puzzles, for which you can only get an instance, *i.e.*, $\text{Pgen}(1^\lambda) \to y$.

Impagliazzo says we live in one of five worlds:

1. Algorithmica, where P = NP;
2. Heuristica, where there are no average hard NP-puzzles, *i.e.*, problems without solution;
3. Pessiland, where you have average hard NP-puzzles;
4. Minicrypt, where you have OWF, one-way NP-puzzles, but no Public Key Cryptography (PKC);
5. Cryptomania, where you have both OWF and PKC.

We'll stay in Minicrypt for now.

OWF are hard to invert on average. Two examples:

- factoring the product of two large prime numbers;
- compute the discrete logarithm, *i.e.*, take a finite group $(\mathbb{G}, \cdot)$, and compute $y = g^x$ for some $g \in \mathbb{G}$. The find $x = \log_g(y)$. This is hard to compute in some groups, *e.g.*, $\mathbb{Z}_p^\star$.

2.2 Computational Indistinguishability

Definition 5. [Distribution Ensemble] A distribution ensemble $\mathcal{X} = \{X_n\}_{n\in\mathbb{N}}$ is a sequence of distributions X_i over some space $\{0,1\}^\lambda$. $\diamond$

Definition 6. [Computational Indistinguishability] Two distribution ensembles $\mathcal{X}_\lambda$ and $\mathcal{Y}_\lambda$ are computationally indistinguishable, written as $\mathcal{X}_\lambda \approx_c \mathcal{Y}_\lambda$, if for all PPT distinguishers $\mathcal{D}$ it holds that

$$\Big|\Pr\left[\mathcal{D}(\mathcal{X}_\lambda) = 1\right] - \Pr\left[\mathcal{D}(\mathcal{Y}_\lambda) = 1\right]\Big| \leq \varepsilon(\lambda).$$

$\diamond$

Lemma 1. *[Reduction] If $\mathcal{X} \approx_c \mathcal{Y}$, then for all PPT functions f, $f(\mathcal{X}) \approx_c f(\mathcal{Y})$.* $\diamond$

Proof of lemma 1. Assume, for the sake of contradiction, that $\exists f$ such that $f(\mathcal{X}) \not\approx_c f(\mathcal{Y})$: then we can distinguish $\mathcal{X}$ and $\mathcal{Y}$. Since $f(\mathcal{X}) \not\approx_c f(\mathcal{Y})$, then $\exists p = \text{poly}(\lambda), \mathcal{D}$ such that, for infinitely many λs

$$\Big|\Pr\left[\mathcal{D}(f(\mathcal{X}_\lambda)) = 1\right] - \Pr\left[\mathcal{D}(f(\mathcal{Y}_\lambda)) = 1\right]\Big| \geq \frac{1}{p(\lambda)}.$$

$\mathcal{D}$ distinguishes $\mathcal{X}_\lambda$ and $\mathcal{Y}_\lambda$ with non-negligible probability. Consider the following $\mathcal{D}'$, which is given

$$z = \begin{cases} x \leftarrow \$\mathcal{X}_\lambda; \\ y \leftarrow \$\mathcal{Y}_\lambda. \end{cases}$$

$\mathcal{D}'$ runs $\mathcal{D}(f(z))$ and outputs whatever it outputs, and has the same probability of distinguishing $\mathcal{X}$ and $\mathcal{Y}$ of $\mathcal{D}$, in contradiction with the fact that $\mathcal{X} \approx_c \mathcal{Y}$. $\square$

Now we show that computational indistinguishability is transitive.

Lemma 2. *[Hybrid Argument] Let $\mathcal{X} = \{X_\lambda\}$, $\mathcal{Y} = \{Y_\lambda\}$, $\mathcal{Z} = \{Z_\lambda\}$ be distribution ensembles. If $\mathcal{X}_\lambda \approx_c \mathcal{Y}_\lambda$ and $\mathcal{Y}_\lambda \approx_c \mathcal{Z}_\lambda$, then $\mathcal{X}_\lambda \approx_c \mathcal{Z}_\lambda$.* $\diamond$

Proof of lemma 2. This follows from the triangular inequality.

$$\begin{aligned}
\Big|\Pr[\mathcal{D}(\mathcal{X}_\lambda)=1]-\Pr[\mathcal{D}(\mathcal{Z}_\lambda)=1]\Big| &= \Big|\Pr[\mathcal{D}(\mathcal{X}_\lambda)=1]-\Pr[\mathcal{D}(\mathcal{Y}_\lambda)=1] \\
&\quad+\Pr[\mathcal{D}(\mathcal{Y}_\lambda)=1]-\Pr[\mathcal{D}(\mathcal{Z}_\lambda)=1]\Big| \\
&\le \Big|\Pr[\mathcal{D}(\mathcal{X}_\lambda)=1]-\Pr[\mathcal{D}(\mathcal{Y}_\lambda)=1]\Big| \\
&\quad+\Big|\Pr[\mathcal{D}(\mathcal{Y}_\lambda)=1]-\Pr[\mathcal{D}(\mathcal{Z}_\lambda)=1]\Big| \\
&\le 2\varepsilon(\lambda). \qquad\qquad \text{(negligible)}
\end{aligned}$$

□

We often prove $\mathcal{X} \approx_c \mathcal{Y}$ by defining a sequence $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_t$ of distributions ensembles such that $\mathcal{H}_0 \equiv \mathcal{X}$ and $\mathcal{H}_t \equiv \mathcal{Y}$, and that for all i, $\mathcal{H}_i \approx_c \mathcal{H}_{i+1}$.

2.3 Pseudo Random Generators

Let's see our first cryptographic primitive. Pseudo Random Generators (PRGs) take in input a random seed and generate pseudo random sequences with some stretch, *i.e.*, output longer than input, and indistinguishable from a true random sequence.

Definition 7. [Pseudo Random Generator] A function $\mathcal{G} : \{0,1\}^\lambda \to \{0,1\}^{\lambda+l(\lambda)}$ is a PRG if and only if

1. $\mathcal{G}$ is computable in polynomial time;
2. $|\mathcal{G}(s)| = \lambda + l(\lambda)$ for all $s \in \{0,1\}^\lambda$;
3. $\mathcal{G}(\mathcal{U}_\lambda) \approx_c \mathcal{U}_{\lambda+l(\lambda)}$.

$\diamond$

Theorem 7. *If $\exists$ PRG with 1 bit of stretch, then $\exists$ PRG with $l(\lambda)$ bits of stretch, with $l(\lambda) = \mathrm{poly}(\lambda)$.* $\diamond$

Proof of theorem 7. We'll prove this just for some fixed constant $l(\lambda) = l \in \mathbb{N}$.

First, let's look at the construction (fig. 3). We replicate our PRG $\mathcal{G}$ with 1 bit stretch l times. The PRG $\mathcal{G}^l$ that we define takes in input $s \in \{0,1\}^\lambda$, computes $(s_1, b_1) = \mathcal{G}(s)$, where $s_1 \in \{0,1\}^l$ and $b_1 \in \{0,1\}$, outputs b_1 and feeds s_1 to the second copy of PRG $\mathcal{G}$, and so on until the l-th PRG.

To show that our construction is a PRG, we define l hybrids, with $\mathcal{H}_0^\lambda \equiv \mathcal{G}^l(\mathcal{U}_\lambda)$, where $\mathcal{G}^l : \{0,1\}^\lambda \to \{0,1\}^{\lambda+l}$ is our proposed construction, and $\mathcal{H}_i^\lambda$

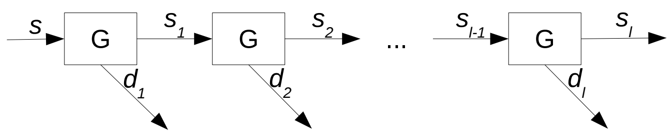

Figure 3: Extending a PRG with 1 bit stretch to a PRG with l bit stretch.

takes $b_1, \ldots, b_i \leftarrow \$\{0,1\}$, $s_i \leftarrow \$\{0,1\}^\lambda$, and outputs $(b_1, \ldots, b_i, s_l)$, where $s_l \in \{0,1\}^{\lambda+l-i}$ is $s_l = \mathcal{G}^{l-i}(s_i)$, *i.e.*, the output of our construction restricted to $l - i$ units.

$\mathcal{H}_l^\lambda$ takes $b_1, \ldots, b_l \leftarrow \$\{0,1\}$ and $s_l \leftarrow \$\{0,1\}^l$ and outputs $(b_1, \ldots, b_l, s_l)$ directly.

We need to show that $\mathcal{H}_i^\lambda \approx_c \mathcal{H}_{i+1}^\lambda$. To do so, fix some i. The only difference between the two hybrids is that s_{i+1}, b_{i+1} are pseudo random in $\mathcal{H}_i^\lambda$, and are truly random in $\mathcal{H}_{i+1}^\lambda$. All bits before them are truly random, all bits after are pseudo random.

Assume these two hybrids are distinguishable, then we can break the PRG. Consider the PPT function f_i defined by $f(s_{i+1}, b_{i+1}) = (b_1, \ldots, b_l, s_l)$ such that $b_1, \ldots b_i \leftarrow \$\{0,1\}$ and, for all $j \in [i+1, l]$ $(b_j, s_j) = \mathcal{G}(s_{j-1})$.

By the security of PRGs we have that $\mathcal{G}(\mathcal{U}_\lambda) \approx_c \mathcal{U}_{\lambda+1}$. By reduction, we also have that $f(\mathcal{G}(\mathcal{U}_\lambda)) \approx_c f(\mathcal{U}_{\lambda+1})$. Thus, $\mathcal{H}_i^\lambda \approx_c \mathcal{H}_{i+1}^\lambda$. $\square$

Acronyms

DL Discrete Log

CDH Computational Diffie-Hellman

DDH Decisional Diffie-Hellman

MAC Message Authentication Code

OTP One Time Pad

OWF One Way Function

PKC Public Key Cryptography

PPT Probabilistic Polynomial Time

PRG Pseudo Random Generator

RV random variable