Computer Science 308-547A
Cryptography and Data Security

Claude Crépeau

These notes are, largely, transcriptions by Anton Stiglic of class notes from the former course *Cryptography and Data Security (308-647A)* that was given by prof. Claude Crépeau at McGill University during the autumn of 1998-1999. These notes are updated and revised by Claude Crépeau.

6 Information theory

The security of most modern cryptographic system is based on a computational assumption. In large, a system that is computationally secure relies on the fact that an attacker does not have enough **time** to break the system, while on the other hand a system that is unconditionally secure relies on the fact that an attacker will never have enough **information**. Computational security uses results from *complexity theory*, unconditional security uses *information theory* which we present in this chapter.

6.1 Notations and probability theory

Let X, Y be distributions on finite sets of events (we will only consider finite sets of probabilities), we denote $\Pr_X(x)$ as the probability of x occurring, following the distribution X (when the situation is clear, we will not explicitly write the distribution on the index).
$\Pr_{X,Y}(x, y)$ is the **mutual probability**, it is the probability that both x and y occur.
$\Pr_{X,Y}(x|y)$ is the **conditional probability**, it is the probability that x occurs knowing that y has occurred.

Definition 6.1 *X and Y are two* **independent** *distributions if*

$$\forall_{x\in X, y\in Y}\ \Pr(x, y) = \Pr(x)\Pr(y)$$

By definition, we have that $\Pr(x, y) = \Pr(x|y)\Pr(y) = \Pr(y|x)\Pr(x)$.
We can now deduce the following theorem

Theorem 6.2 *(Bayes)*

$$If\ \Pr(y) > 0\ then\ \Pr(x|y) = \frac{\Pr(y|x)\Pr(x)}{\Pr(y)}$$

We can easily convince ourselves of the following corollary

Corollary 6.3 *X and Y are independent variables iff* $\forall_{x\in X, y\in Y}$

$$\Pr(x|y) = \Pr(x)$$

6.2 Shannon's information theory and entropy

Consider a random source $\mathcal{S} \longrightarrow 100110100111011...$ We would like to characterize its output. $\mathcal{S}$ outputs symbols with probabilities $\Pr(0), \Pr(1)$. How much uncertainty do we have about the output of this source (what is the entropy $\mathbf{H}[\mathcal{S}]$ of $\mathcal{S}$)?

Example 6.1 $\mathcal{S}_0 \longrightarrow 00000000...$ *(always outputs* 0*), then* $\mathbf{H}[\mathcal{S}_0] = 0$.
$\mathcal{S}_1 \longrightarrow 111111111...$ *(always outputs* 1*), then* $\mathbf{H}[\mathcal{S}_1] = 0$.
$\mathcal{S}_U \longrightarrow 100110100111011...$ *(Independent and Uniform), then* $\mathbf{H}[\mathcal{S}_U] = 1$.

In a more general case, how do we define $\mathbf{H}[\mathcal{S}]$?

6.2.1 Entropy

The entropy of a source $\mathcal{S}$ can be considered as the average length, in bits, needed to represent the output of $\mathcal{S}$ (this was introduced in [**?**]).

Example 6.2 *Consider a random source* $\mathcal{S} \longrightarrow a, b, c$*, where*

$$\Pr(a) = 1/2, \Pr(b) = 1/4 = \Pr(c) = 1/4$$

and the coding $a \longrightarrow 0, b \longrightarrow 10, c \longrightarrow 11$ *(a prefix coding), then the average length needed to represent the output of* S *is*

$$1/2 \cdot 1 \textit{ bit } + 1/4 \cdot 2 \textit{ bits } + 1/4 \cdot 2 \textit{ bits } = 3/2 \textit{ bits.}$$

More formally, we state the following definition of the entropy of a source $\mathcal{S}$, where the probabilities of the outputs are $p_1, p_2, ..., p_n$,

Definition 6.4 (Entropy)

$$\mathbf{H}[p_1, p_2, ..., p_n] = \sum_{i=1}^{n} p_i \lg \frac{1}{p_i} = -\sum_{i=1}^{n} p_i \lg p_i$$

where $\lg$ *represents the logarithm base 2. We assume in this definition that* $0 \lg 0 = 0$ *(the real fact is that* $\lim_{x \longrightarrow 0^+} x \lg x = 0$*).*

Another way of defining the entropy is through the following set of properties

6.2.2 Properties of entropy

$\mathbf{H}[p_1, p_2, ..., p_n]$, where $\sum_{i=1}^{n} p_i = 1$, satisfies the following:

1. $\mathbf{H}[p_1, p_2, ..., p_n]$ is maximum when $p_1 = p_2 = ... = p_n = 1/n$ *(the uniform distribution should have the greatest entropy...).*
2. $\mathbf{H}[p_1, p_2, ..., p_n] = \mathbf{H}\left[p_{\pi(1)}, p_{\pi(2)}, ..., p_{\pi(n)}\right]$ for any permutation π.
3. $\mathbf{H}[p_1, p_2, ..., p_n] \geq 0$, with $= 0$ exactly if one $p_i = 1$.
4. $\mathbf{H}[p_1, p_2, ..., p_n, 0] = \mathbf{H}[p_1, p_2, ..., p_n]$
5. $\mathbf{H}\left[\underbrace{\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}}_{n}\right] \leq \mathbf{H}\left[\underbrace{\frac{1}{n+1}, \frac{1}{n+1}, ..., \frac{1}{n+1}}_{n+1}\right]$
6. $\mathbf{H}[*]$ is continuous

Theorem 6.5 *Let $H'(p_1, p_2, ..., p_n)$ satisfy the above properties.*
There exists a constant λ such that

$$H'(p_1, p_2, ..., p_n) = \lambda \mathbf{H}[p_1, p_2, ..., p_n].$$

6.2.3 Entropy of random variables

For a random variable X such that $\Pr_X(x_1) = p_1, ..., \Pr_X(x_n) = p_n$ we abuse notation and write $\mathbf{H}[X]$ instead of $\mathbf{H}[p_1, p_2, ..., p_n]$.

Look ahead: For a cryptosystem $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, we will be interested in random variables P, K, C describing the uncertainty about the plaintext, ciphertext and key, and in $\mathbf{H}[P|C]$ (uncertainty of the message given the ciphertext) and $\mathbf{H}[K|C]$ (uncertainty of the encryption key given the ciphertext).

Now, a couple more theorems and definitions...

Theorem 6.6

$$\mathbf{H}[X, Y] \leq \mathbf{H}[X] + \mathbf{H}[Y]$$

(equal iff X and Y are mutually independent)

Theorem 6.7

$$\mathbf{H}[X_1, X_2, ..., X_n] \le \sum_{i=1}^{n} \mathbf{H}[X_i]$$

(equal iff $X_1, X_2, .., X_n$ are mutually independent)

6.3 Conditional Entropy

Definition 6.8 (Conditional entropy)

$\mathbf{H}[X|Y=y] = -\sum_{x\in X} \Pr_{X|Y=y}(x) \lg \Pr_{X|Y=y}(x)$

$\mathbf{H}[X|Y] = \sum_{y\in Y} \Pr_Y(y)\, \mathbf{H}[X|Y=y]$

$= -\sum_{y\in Y}\sum_{x\in X} \Pr_Y(y) \Pr_{X|Y=y}(x) \lg \Pr_{X|Y=y}(x)$

Theorem 6.9 $\mathbf{H}[X|Y] = 0$ *exactly if* $X = f(Y)$ *for some function f.*

Theorem 6.10 $\mathbf{H}[X|Y] = \mathbf{H}[X]$ *exactly if X and Y are independent.*

Theorem 6.11

$$\mathbf{H}[X,Y] = \mathbf{H}[Y] + \mathbf{H}[X|Y]$$

and thus, $\mathbf{H}[X|Y] \le \mathbf{H}[X]$ *(using Theorem 6.6), with equality iff X, Y are indepenent.*

6.4 Mutual Information

Definition 6.12

$$\mathbf{I}[X;Y] = \mathbf{H}[X] - \mathbf{H}[X|Y]$$

Theorem 6.13

$$\mathbf{I}[X;Y] = \mathbf{I}[Y;X]$$

option vs vote	OUI	NON	und.
separation	30%	60%	10%
offer of partnership	39%	46%	15%
certainty of partnership	54%	35%	11%

6.5 Example:

$$\Pr[OPTION = \text{“separation”}] = 1/4$$
$$\Pr[OPTION = \text{“offer of partnership”}] = 1/2$$
$$\Pr[OPTION = \text{“certainty of partnership”}] = 1/4$$

$$\Pr[VOTE = \text{“OUI”}] = 40.5\%$$
$$\Pr[VOTE = \text{“NON”}] = 46.75\%$$
$$\Pr[VOTE = \text{“undec.”}] = 12.75\%$$

$$\mathbf{H}\,[VOTE] = -.405 \lg .405 - .4675 \lg .4675 - .1275 \lg .1275 = 1.4198$$
$$\mathbf{H}\,[OPTION] = -.25 \lg .25 - .5 \lg .5 - .25 \lg .25 = 1.5$$

$$\mathbf{H}\,[VOTE|OPTION = \text{“separation”}]$$
$$= -.3 \lg .3 - .6 \lg .6 - .1 \lg .1 = 1.2955$$
$$\mathbf{H}\,[VOTE|OPTION = \text{“offer of partnership”}]$$
$$= -.39 \lg .39 - .46 \lg .46 - .15 \lg .15 = 1.4557$$
$$\mathbf{H}\,[VOTE|OPTION = \text{“certainty of partnership”}]$$
$$= -.54 \lg .54 - .35 \lg .35 - .11 \lg .11 = 1.3604$$

$$\mathbf{H}\,[VOTE|OPTION] =$$
$$\frac{\mathbf{H}\,[VOTE|OPTION = \text{“separation”}]}{4} +$$
$$\frac{\mathbf{H}\,[VOTE|OPTION = \text{“offer of partnership”}]}{2} +$$
$$\frac{\mathbf{H}\,[VOTE|OPTION = \text{“certainty of partnership”}]}{4} =$$
$$\frac{1.2955}{4} + \frac{1.4557}{2} + \frac{1.3604}{4} = 1.3918$$

$$\mathbf{I}\,[VOTE; OPTION] = \mathbf{H}\,[VOTE] - \mathbf{H}\,[VOTE|OPTION]$$
$$= 1.4198 - 1.3918 = 0.0280$$

6.6 Jensens Lemma

The following theorem will be used to prove some more interesting statements about entropy functions. First, a preliminary definition.

Definition 6.14 *A real function f is said to be concave on an interval I if $\forall_{x,y \in I}$*

$$f\left(\frac{x+y}{2}\right) \geq \frac{f(x)+f(y)}{2}$$

Theorem 6.15 (Jensens inequality) *If f is continuous and strictly concave on I, and $\sum_{i=1}^{n} a_i = 1$, $a_i \geq 0$, $1 \leq i \leq n$, then, $\forall_{x_i \in I}$*

$$\sum_{i=1}^{n} a_i f(x_i) \leq f(\sum_{i=1}^{n} a_i x_i)$$

with equality iff $x_1 = x_2 = ... = x_n$

6.6.1 Application

Theorem 6.16 *if X takes values x_1 with probability p_1, x_2 with probability p_2, ..., x_n with probability p_n, then*

$$\mathbf{H}[X] \leq \lg\ n$$

(with equality when $p_1 = p_2 = ...p_n = 1/n$)

Proof.

$$\begin{aligned}
\mathbf{H}[X] &= -\sum_{i=1}^{n} p_i \lg p_i \\
&= \sum_{i=1}^{n} p_i \lg 1/p_i \\
&\leq \lg \sum_{i=1}^{n} p_i 1/p_i \\
&= \lg n \ \ (eq.\ when\ p_1 = p_2 = ... = p_n = 1/n)
\end{aligned}$$

6.7 Perfect Secrecy, Key Equivocation, Unicity Distance

Definition 6.17 (Perfect Secrecy) *Let P, K, C be random variables describing the uncertainty about the plaintext, ciphertext and key of a cryptosystem $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$. We say that a cryptosystem has perfect secrecy if* $\forall_{p \in \mathcal{P}, c \in \mathcal{C}} \Pr_{P,C}(p|c) = \Pr_P(p)$

Example 6.3 *Vernam's one-time pad has perfect secrecy. Here, $\mathcal{P} = \mathcal{K} = \mathcal{C} = \{0,1\}^n$, with K the uniform distribution on $\mathcal{K}$ (no distribution condition necessary on $\mathcal{P}$). Encryption and decryption are given by $e_k(p) = p \oplus k$, $d_k(c) = c \oplus k$.*

$$\begin{aligned} \Pr_{P,C}(p|c) &= \Pr_{P,C}(p|p \oplus k) \\ &= \Pr_P(p) \Pr_{P,C}(p \oplus k|p) / \Pr_C(p \oplus k) \ (\mathit{Bayes}) \end{aligned}$$

for any fixed p, $p \oplus K$ is the uniform distribution, so $\Pr_{P,C}(p \oplus k|p) = 1/2^n = \Pr_C(p \oplus k)$ *and so*

$$\begin{aligned} \Pr_{P,C}(p|c) &= \Pr_P(p) \frac{1}{2^n} / \frac{1}{2^n} \\ &= \Pr_P(p) \end{aligned}$$

An equivalent definition is the following

Definition 6.18 *Cryptosystem $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ has perfect secrecy iff*

$$\mathbf{H}[P|C] = \mathbf{H}[P]$$

Theorem 6.19

$$\mathbf{H}[C] \geq \mathbf{H}[P]$$

when K and P are independent.

Proof.

$$\begin{aligned} \mathbf{H}[C] &\geq \mathbf{H}[C|K] \\ &= \mathbf{H}[C,K] - \mathbf{H}[K] \\ &= \mathbf{H}[P,K] - \mathbf{H}[K] \ \text{(encryption function is bijective)} \\ &= \mathbf{H}[P] + \mathbf{H}[K|P] - \mathbf{H}[K] \ \text{(Thm 6.11)} \\ &= \mathbf{H}[P] + \mathbf{H}[K] - \mathbf{H}[K] \ \text{(we suppose } P, K \text{ independent)} \\ &= \mathbf{H}[P] \end{aligned}$$

We could also show the following

Theorem 6.20 *(when we know the key...)*

$$\mathbf{H}[C|K] = \mathbf{H}[P|K]$$

Theorem 6.21

$$\mathbf{H}[P|C] \leq \mathbf{H}[K]$$

Proof.

$$\begin{aligned}
\mathbf{H}[P|C] &\leq \mathbf{H}[P,K|C] \\
&= \underbrace{\mathbf{H}[P|K,C]}_{0} + \mathbf{H}[K|C] \ \text{(Bayes)} \\
&= \mathbf{H}[K|C] \\
&\leq \mathbf{H}[K]
\end{aligned}$$

This shows that if there is not much doubt about the key, there can't be much doubt about the plaintext knowing the ciphertext.

Definition 6.22 (key equivocation) $\mathbf{H}[K|C]$ *is called the key equivocation, it is a measure of how much we know about the key, knowing the ciphertext.*

Theorem 6.23

$$\mathbf{H}[K|C] = \mathbf{H}[K] + \mathbf{H}[P] - \mathbf{H}[C]$$

Proof.

$$\begin{aligned}
(1)\ \mathbf{H}[K,P,C] &= \underbrace{\mathbf{H}[C|K,P]}_{0} + \mathbf{H}[K,P] \\
&= \mathbf{H}[K,P] \\
&= \mathbf{H}[K] + \mathbf{H}[P] \ \text{(by independence)} \\
(2)\ \mathbf{H}[K,P,C] &= \mathbf{H}[K,C] \ \text{(because from } K \text{ and } C\text{, we can induce } P\text{)}
\end{aligned}$$

we conclude

$$\begin{aligned}\mathbf{H}[K|C] &= \mathbf{H}[K,C] - \mathbf{H}[C] \\ &= \mathbf{H}[K,P,C] - \mathbf{H}[C] \\ &= \mathbf{H}[K] + \mathbf{H}[P] - \mathbf{H}[C]\end{aligned}$$

Let $P^n = (P_1, \ldots, P_n)$ be a distribution on $n - grams$ of $\mathcal{P}$, and $C^n = (C_1, \ldots, C_n)$ be a distribution on $n - grams$ of $\mathcal{C}$, Suppose that we are working in a cryptographic system $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ where a plaintext $x_1 x_2 \ldots x_n$ is encrypted into a ciphertext $y_1 y_2 \ldots y_n$ by some permutation function. We want to know how much ciphered letters we have to see to know what the key is. The distribution depends on the language in use.

Definition 6.24 *Rate of a Language L*

$$H_L = lim_{n \longrightarrow \infty} \frac{\mathbf{H}[P^n]}{n}$$

For english we have $1 \leq H_L \leq 1.5$

Definition 6.25 *Redondancy of language L*

$$R_L = 1 - \frac{H_L}{\lg\ |\mathcal{P}|}$$

If we take $1,25$ as an estimation of H_L in english, we have a redondancy of 0.75.

For a $c \in \mathcal{C}$, it is possible that many keys can be found such that there exists plaintexts that encrypt to c.

Definition 6.26 (Spurious Keys) *Given some $c \in C^n$, we define*

$$K(c) = \{k \in \mathcal{K} | \exists_x \in \mathcal{P}^n, \Pr_{P^n}(x) > 0 \; and \; c = e_k(x)\}$$

Finally we can define the notion of unicity distance, which is a measure of the amount of ciphered letters we have to see to know, without doubt, the unique key that was used.

Definition 6.27 *(Unicity Distance)*

$$\overline{S_n} = \sum_{c \in \mathcal{C}^n} \Pr(c)(|K(c)| - 1)$$

We are interested in knowing for what value of n will $\overline{S_n} = 0$.

Theorem 6.28 *([**?**]) For large n,*

$$\overline{S_n} \geq \frac{|\mathcal{K}|}{|\mathcal{P}|^{nR_L}} - 1$$

If we take $\overline{S_n} = 0$ and solve for n we get

$$n \geq \frac{\lg |\mathcal{K}|}{R_L \ \lg |\mathcal{P}|}$$

Example 6.4

- *Shift Cipher*
  $|\mathcal{P}| = 26, |\mathcal{K}| = 26, R_L = 0.75,\ n \geq 1/R_L = 4/3$
- *Substitution Cipher*
  $|\mathcal{K}| = 26!,\ \lg 26! \approx 88.4,\ n \geq \frac{88.4}{0.75\ 4.7} \approx 25$
- *One-Time Pad*
  $|\mathcal{K}| = 26^n,\ \lg 26^n = n\ \lg 26,\ n \geq \frac{n\ \lg 26}{0.75\ \lg 26} = n/0.75$ *($n = 0$ is the only solution!).*

7 Pseudo-Randomness

7.1 Pseudo-Random Bit Generator (PRBG)

Informally, a PRBG is a deterministic algorithm that stretches a random n-bit string to an $l(n)$-bit string (where $l(n) > n$) which is indistinguishable from a truly random $l(n)$-bit string, in a time that is polynomial in $|n|$. Random bits are useful in probabilistic computation, we could always flip a coin $l(n)$ times to get an $l(n)$-bit random string, but if $l(n)$ is large this is costly, this is but one motivation to have PRBGs.

7.1.1 Indistinguishability

We give some definitions that will be useful from here on.

For the tree following definitions, let $X = \{X_n\}_n$ and $Y = \{Y_n\}_n$ be families of distributions.

Definition 7.1 (Absolute Indistinguishability) *X is absolutely indistinguishable from Y if*

$$\forall_f \, \forall_n \, Pr(f(X_n) = 1) = Pr(f(Y_n) = 1)$$

Definition 7.2 (Statistical Indistinguishability) *X is statistically indistinguishable from Y if*

$$\forall_f \, \forall_{P:polynomial} \, \forall_{n \geq n_0} \, |Pr(f(X_n) = 1) - Pr(f(Y_n) = 1)| \leq 1/P(n)$$

Example 7.1 *Let U_n be the uniform distribution on $\{0,1\}^n$ and let V_n be the uniform distribution on $\{0,1\}^n - \{0^n\}$. Then U and V are statistically indistinguishable.*

Definition 7.3 (Computational Indistinguishability) *X is computationally indistinguishable from Y if*

$$\forall_{A:fast\ algo.} \, \forall_{P:polynomial} \, \forall_{n \geq n_0} \, |Pr(A(X_n) = 1) - Pr(A(Y_n) = 1)| \leq 1/P(n)$$

Example 7.2 *Let N be a random n-bit integer, product of two primes p, q. Let U_n be the uniform distribution on QR_N and let V_n be the uniform distribution on J_N. Then U and V are computationally indistinguishable if deciding quadratic residuosity modulo a composite number is a hard problem.*

We now formally define a secure PRBG

Definition 7.4 (SPRBG) *We say that a family of functions*

$$g_n : \{0,1\}^n \to \{0,1\}^{l(n)}$$

for a superpolynomial function $l(n)$, is a secure PRBG if, for all $x \in \{0,1\}^n$, $g_n(x)$ can be computed in polynomial time (in n) and if for every polynomial P, the families

$$X_n = g(U_n)|_{P(n)} \text{ and } Y_n = U_{P(n)}$$

are computationally indistinguishable.

7.2 Examples of PRBG

We will give examples of PRBGs that take a seed $s_0 \in \{0,1\}^n$ and expand it to an element of $\{0,1\}^{l(n)}$

We first give an example of a PRBG that is not a SPRBG,

Algorithm 7.1 (*Linear_Congruential_BG*(s_0))

```
1: $M \leftarrow 2^n$, and $a, b$ such that $1 \leq a, b \leq M - 1$
2: FOR $i \leftarrow 0$ TO $l(n)$
3: $s_{i+1} \leftarrow (a\, s_i + b) \bmod M$
4: $z_{i+1} \leftarrow s_{i+1} \bmod 2$
5: ENDFOR
6: RETURN $z_1 \| z_2 \| ... \| z_{l(n)}$.
```

Theorem 7.5 *([?], [?])*
Given a segment $z_1 z_2 ... z_i$ of Linear_Congruential_BG(s_0), it is computationally easy to find a, b, and to predict z_{i+1}

So the generator is not cryptographically "secure". In particular, if one uses this generator to do one-time-pad encryption and the adversary knows a sufficient portion of the plaintext, then he can deduce the sequence and obtain the rest of the pad (key) and the plaintext. Nevertheless, this generator is very often used for its well behaved statistical properties. It is very well adopted to applications outside of cryptography.

The next bit generator we present is from [?], it is similar to the one above, but with a variation that makes it a cryptographically secure PRBG. Let $N \leftarrow pq$, where p, q are random $(n/2)$-bit primes such that $p \equiv q \equiv 3 \bmod 4$.

Algorithm 7.2 (*Blum_Blum_Shub_PRBG*($s_o \in QR_N$))

```
1: FOR $i \leftarrow 0$ TO $l(n)$
2: $s_{i+1} \leftarrow {s_i}^2 \bmod N$
3: $z_{i+1} \leftarrow s_{i+1} \bmod 2$
4: ENDFOR
5: RETURN $z_1 \| z_2 \| ... \| z_{l(n)}$.
```

One intriguing property of the Blum Blum Shub bit generator is the fact that knowledge of the seed s_0 and the factorization of the modulus N allows direct access to each of the first $exp(n)$ bits. To compute $s_0^{2^\ell} \bmod N$ for large values of ℓ compute $s_\ell \leftarrow s_0^{2^\ell \bmod \phi(N)} \bmod N$.

The security of generator is reflected in the following result (due to [?])

Theorem 7.6 *The Blum-Blum-Shub bit generator is a SPRBG if factoring Blum-integers is hard (*N *is a Blum-integer if it is of the form* pq*, where* $p \equiv q \equiv 3 \bmod 4$*).*

We will see a similar result using the RSA function in the public-key cryptosystem section.

The next example is from Blum-Micali [?], it is also a cryptographically secure PRBG. Let p be an odd prime and g a generator of $\in \mathbb{Z}_p^*$.

Algorithm 7.3 (*Blum_Micali_PRBG*(s_0))

```
1: FOR $i \leftarrow 0$ TO $l(n)$
2: $s_{i+1} \leftarrow g^{s_i} \bmod p$
3: IF $s_i \leq (p+1)/2$ THEN $z_{i+1} \leftarrow 0$, ELSE $z_{i+1} \leftarrow 1$
4: ENDFOR
5: RETURN $z_1 \| z_2 \| ... \| z_{l(n)}$.
```

Theorem 7.7 *The Blum Micali bit generator is a SPRBG if DLP (discrete log problem, finding x given $p, g, g^x \bmod p$) is hard.*

7.3 Examples of application of PRBGs

Example 7.3 (Stream-Cipher from PRBG) *Alice and Bob share a random secret $s \in \{0,1\}^n$. To encrypt a message m of size $l(n)$, Alice generates a pseudo-key $k \in \{0,1\}^{l(n)}$ using the PRBG on s, sets $c \leftarrow k \oplus m$, and transmits the ciphertext c. To decrypt, Bob computes $m \leftarrow k \oplus c$ where k is computed similarily.*

Example 7.4 (Message Authentication from PRBG) *Alice and Bob share a random secret $s \in \{0,1\}^n$. To authenticate a message m of size $l(n)$, Alice generates a pseudo-key $(i,j) \in \{0,1\}^{2*l(n)}$ using the PRBG on s, sets $t \leftarrow i * m + j|_{50}$, and transmits the pair $\langle m, t \rangle$. To check authenticity, Bob computes (i,j) and t from s and m and verify the equality with the received tag.*

Example 7.5 (Identification from PRBG) *Alice and Bob share a random secret $s \in \{0,1\}^n$. When Alice wants to identify herself to Bob, they both generate a pseudo-key $k \in \{0,1\}^{l(n)}$ using the PRBG on s, and run the interactive identification protocol seen in section 5.4.*

7.4 General Results on PRBGs

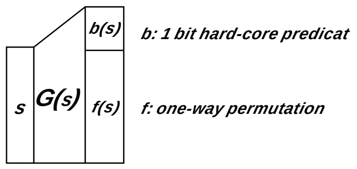
Theorem 7.8 ***Definition 7.9 (one-way function)*** *A function f is* one-way *if there exists a polynomial time algorithm to compute $f(x)$ from x, whereas for any polynomial time inverting algorithm I,* $\Pr[f(I(f(x))) = f(x)]$ *is negligeable when x is picked at random from the domain of f.*

Definition 7.10 (hard-core predicate) *A predicate b is* hard-core *for a one-way function f if there exists a polynomial time algorithm to compute $b(x)$ from x, whereas for any polynomial time predicting algorithm P,* $\Pr[P(f(x)) = b(x)]$ *is no more than 1/2+negligeable when x is picked at random from the domain of f.*

Given a PRBG that has an expansion factor $l(n) = n + 1$, it is possible to construct a PRBG with any polynomial expansion factor.

Construction: Let G_1 be a PRBG from $\{0,1\}^n$ to $\{0,1\}^{n+1}$. Define $G(s) = \sigma_1 \dots \sigma_{p(n)}$, where σ_i is the first bit of $G_1(s_{i-1})$ and s_i is the n-bit suffix of $G_1(s_{i-1})$ (i.e. $\sigma_i s_i = G_1(s_{i-1})$), with $s_0 = s$ being the initial random value.

Theorem 7.11 ([?]) *Given any one-way permutation f and a hard-core predicate b for f, $f||b$ is a PRBG.*



Idea of proof. Let U_f be the uniform distribution on the elements of the domain of f. We use theorem 7.8 and the fact that if the distribution $\{f(U_f)||b(U_f)\}$ was not pseudorandom, then it would be possible to efficiently compute $b(U_f)$ from $f(U_f)$.

Goldreich and Levin gave a hard-core predicate with respect to any one-way function:

Theorem 7.12 **(([?]))** *Let f be a one-way function from $\{0,1\}^n$ to $\{0,1\}^{n'}$ and define*

$$f' : \ (x, r) \rightarrow (f(x), r), \text{ where } r, x \in \{0,1\}^n$$

(f' is also one-way)
Define $x \odot r$ to be the inner product bit of x and r ($x \odot r = \oplus_{i=1}^{n} x_i \wedge r_i$).
The predicate $x \odot r$ is hard-core for f' !

Impagliazzo, Levin and Luby generalize theorem 7.11 to any one-way function:

Theorem 7.13 **([?])** *Given a function that is one-way , it is possible to construct a PRBG.*

8 Pseudo-Random Function Generators (PRΦG)

For some cryptographic tasks it would be useful to agree on a random function. Let H_n be the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$. The cardinality of this set is 2^{n2^n}, thus to specify a function in H_n we would need $n2^n$ bits, which is a lot. Moreover, if we randomly select a subset $H'_n \subseteq H_n$ of cardinality 2^n so that each function in H'_n has a unique n-bit index, then there is no polynomial time bounded algorithm that, given an index to a function $f \in H'_n$ and a $x \in \{0,1\}^n$, can evaluate $f(x)$. Pseudo-Random Function Generators are used instead of truly random functions.

Informally, a PRΦG is such that it generates a family $F = F_n$ of functions from $\{0,1\}^n$ to $\{0,1\}^n$ where each function $f \in F_n$ has a unique n-bit index, for all $x \in \{0,1\}^n$ $f(x)$ can be efficiently computed and no polynomial time algorithm (in n) can distinguish F_n from H_n.
Formally, we have the following definition

Definition 8.1 (PRΦG) *Let $H = \{H_n\}$ and $F = \{F_n\}$ where*

$$H_n = \{h \in \{0,1\}^n \to \{0,1\}^n\} \text{ and } F_n = \{f_k \in H_n : k \in \{0,1\}^n\}.$$

F is a pseudo-random function generator (PRΦG) if for all large enough n, for h picked at random from H_n and a random n-bit string k
$\forall_{T:\text{fast algo.}} \forall_{P,p:\text{polynomials}} \forall_{x_1,x_2,\ldots,x_{p(n)}}$
$|Pr(T[(x_1,h(x_1)),(x_2,h(x_2)),...,(x_{p(n)},h(x_{p(n)}))] = 1) -$
$Pr(T[(x_1,f_k(x_1)),(x_2,f_k(x_2)),...,(x_{p(n)},f_k(x_{p(n)}))] = 1)| \leq 1/P(n)$

8.1 Constructing PRΦGs

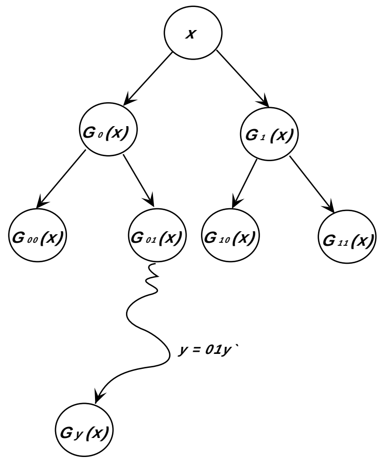
[**?**] showed how to construct a PRΦG from any given PRBG. Let G be a PRBG that stretches a seed $x \in \{0,1\}^n$ into a $2n$-bit long sequence,
$G(x) = b_1^x \ldots b_{2n}^x$. Denote $G_0(x)$ to be the first n bits output ($G_0(x) = b_1^x \ldots b_n^x$) and $G_1(x)$ the last n bits ($G_1(x) = b_{n+1}^x \ldots b_{2n}^x$). Let $\alpha = \alpha_1\alpha_2 \cdot \alpha_t$ be a binary string. We define

$$G_{\alpha_1\alpha_2\cdot\alpha_t}(x) = G_{\alpha_t}(\ldots(G_{\alpha_2}(G_{\alpha_1}(x)))\ldots).$$

The function $f_k : \{0,1\}^n \to \{0,1\}^n$ is defined as follows: For $y = y_1y_2\ldots y_n$,

$$f_k(y) = G_{y_1y_2\ldots y_n}(k).$$

Set $F_n = \{f_k\}_{k\in\{0,1\}^n}$ and $F = \{F_n\}_n$. F is a PRΦG. It might be useful to picture the function f_x as a full binary tree of depth n with n-bit strings stored in the nodes and edges labelled 0 or 1.



8.2 Examples of application of PRΦGs

Example 8.1 (Block-Cipher from PRΦG) *Alice and Bob share a secret $k \in \{0,1\}^n$. To encrypt an n-bit message m, Alice picks a random $y \in \{0,1\}^n$, sets $c \leftarrow \phi_k(y) \oplus m$, and transmits the ciphertext $\langle c, y\rangle$.*
To decrypt, Bob computes $m \leftarrow \phi_k(y) \oplus c$.

Example 8.2 (Message Authentication and Time-Stamping) *Say Alice wants to send authenticated messages to Bob, Alice and Bob need just share a random value k. When Alice wants to send a message m she simply appends $\phi_k(m)$ for authentication. To prevent replay attacks, Alice could concatenate m with the date and time, then apply ϕ_k. One can tradeoff security for efficiency by simply appending the first $t < n$ bits of the result of the function.*

Example 8.3 (Identification from PRΦG) *Alice and Bob share a secret $k \in \{0,1\}^n$. When Alice wants to identify herself to Bob, Bob sends Alice an $x \in \{0,1\}^n$, if Alice returns $\phi_k(x)$ to Bob, then Bob is convinced that it is indeed Alice, with small probability of error.*