A perfectly secure symmetric encryption scheme: ONE-TIME PAD

This encryption scheme achieves information-theoric security.

Definition 1 (Symmetric encryption). Let $\mathscr{K}$ be a key space, $\mathscr{P}$ be a plain-text space and let $\mathscr{C}$ be a ciphertext space These three spaces are finite spaces.

A *symmetric encryption* scheme over $(\mathscr{K}, \mathscr{P}, \mathscr{C})$ is a tuple of three algorithms $(\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$:

- ▷ KeyGen provides a sample k of $\mathscr{K}$;
- ▷ $\mathrm{Enc} : \mathscr{K} \times \mathscr{P} \to \mathscr{C}$;
- ▷ $\mathrm{Dec} : \mathscr{K} \times \mathscr{C} \to \mathscr{P}$.

Without loss of generality, we will assume that $\operatorname{im}\mathrm{Enc} = \mathscr{C}$. We want to ensure **Correctness**: for any key $k \in \mathscr{K}$ and message $m \in \mathscr{P}$, we have that:

$$\mathrm{Dec}(k, \mathrm{Enc}(k, m)) = m.$$

The elements m and k are independent random variables and all the elements in $\mathscr{K}$ and $\mathscr{P}$ have non-zero probability.

Remark 1. The algorithm Enc could (and should[1]) be probabilistic. However, the algorithm Dec is deterministic.

So far, we did not talk about efficiency of these algorithms.

Definition 2 (Shannon, 1949). A symmetric encryption scheme is said to have *perfect security* whenever, for any $\bar{m}$ and any $\bar{c}$,

$$\Pr_{k,m}[m = \bar{m} \mid \mathrm{Enc}_k(m) = \bar{c}] = \Pr_{m}[m = \bar{m}].$$

The intuition is that knowing the encrypted message tells me *nothing* about the message.

Lemma 1 (Shannon). Given a symmetric encryption scheme $(\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$ has perfect security then $|\mathscr{K}| \geq |\mathscr{P}|$.

Proof. Let $\bar{c} \in \mathscr{C}$ and define

$$\mathscr{S} := \{\bar{m} \in \mathscr{P} \mid \exists \bar{k} \in \mathscr{K}, \bar{m} = \mathrm{Dec}(\bar{k}, \bar{c})\}.$$

Let $N := |\mathscr{S}|$. We have that $N \leq |\mathscr{K}|$ as Dec is deterministic. We also have that $N \leq |\mathscr{P}|$ as $\mathscr{S} \subseteq \mathscr{P}$. Finally, assume $N < |\mathscr{P}|$. This means, there exists $\bar{m} \in \mathscr{P}$ such that $\bar{m} \notin \mathscr{S}$. Then,

$$\Pr[m = \bar{m} \mid \mathrm{Enc}_k(m) = \bar{c}] = 0,$$

but by assumption, $\Pr[m = \bar{m}] \neq 0$. So this is not a perfectly secure scheme. We can conclude that

$$N = |\mathscr{P}| \leq |\mathscr{K}|.$$

$\square$

[1] If the algorithm is deterministic, if we see two identical ciphers we know that the messages are identical, and this can be seen as a vulnerability of this protocol.

Example 1 (One-Time PAD). Let $\mathscr{K} = \mathscr{C} = \mathscr{P} = \{0,1\}^\ell$. Here are the algorithms used:

- ▷ KeyGen samples from $\mathscr{U}(\{0,1\}^\ell)$.
- ▷ $\mathrm{Enc}(k, m)$ we compute the XOR $c = m \oplus k$.
- ▷ $\mathrm{Dec}(k, m)$ we compute the XOR $m = c \oplus k$.

Theorem 1. The One-Time PAD is a perfectly-secure symmetric encryption.

Proof. Correctness. We have that

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

Security. We have, by independence of m and k we have that

$$\begin{aligned}\Pr[m = \bar{m} \mid \mathrm{Enc}(k, m) = \bar{c}] &= \Pr[m = \bar{m} \mid k \oplus m = \bar{c}] \\ &= \Pr[m = \bar{m}].\end{aligned}$$

□

Remark 2. This example is not practical:

- ▷ keys need to be larger than the message;
- ▷ you cannot encrypt twice: for example, $c_1 = m_1 \oplus k$ and $c_2 = m_2 \oplus k$, then we have $c_1 \oplus c_2 = m_1 \oplus m_2$.

This last part is why that protocol is called a *One-Time secure encryption.*

We want to be able to encrypt arbitrarily long messages! We will have to make a trade-off and we choose to not care about *perfect* security. Why? In real life, we don't care about proving that something is proven to be absolutely infeasible, we only want to believe it is infeasible in practice.

***Computational complexity* is sufficient in practice.**

Let us be more precise in the next section.

1 Pseudo-random generators.

Definition 3. Let $\mathscr{D}_0$ and $\mathscr{D}_1$ be two distributions over $\{0,1\}^n$.

An algorithm $\mathscr{A} : \{0,1\}^n \to \{0,1\}$ is called a *distinguisher* between $\mathscr{D}_0$ and $\mathscr{D}_1$. We define its *distinguishing advantage* as:

$$\mathrm{Adv}_{\mathscr{A}} := \Big| \underbrace{\Pr_{x \leftarrow \mathscr{D}_1}[\mathscr{A}(X) = 1]}_{\substack{\text{probability of} \\ \text{being right}}} - \underbrace{\Pr_{x \leftarrow \mathscr{D}_0}[\mathscr{A}(X) = 1]}_{\substack{\text{probability of} \\ \text{being mistaken}}} \Big|.$$

We say that $\mathscr{D}_0$ and $\mathscr{D}_1$ are *computationally indistinguishable* if for any efficient distinguisher $\mathscr{A}$ its advantage $\mathrm{Adv}_{\mathscr{A}}$ is small. We will write, in this case, $\mathscr{D}_1 \simeq^{\mathrm{c}} \mathscr{D}_2$.

This definition is not very formal yet, we have not defined "efficient" and "small." This can be formalized by introducing a parameter $\lambda \in \mathbb{N}$ called the *security parameter.*

Definition 4. Let $(\mathscr{D}_{0,\lambda})_{\lambda\in\mathbb{N}}$ and $(\mathscr{D}_{1,\lambda})_{\lambda\in\mathbb{N}}$ be two distributions over $\{0,1\}^{n(\lambda)}$ for a non-decreasing polynomial $n(\lambda)$. The value of $\lambda \in \mathbb{N}$ is called the *security parameter.*

An algorithm $\mathscr{A} : \{0,1\}^{n(\lambda)} \to \{0,1\}$ is called a *distinguisher* between the distributions $\mathscr{D}_{0,\lambda}$ and $\mathscr{D}_{1,\lambda}$. We define its *distinguishing advantage* as:

$$\mathrm{Adv}_{\mathscr{A}}(\lambda) := \Big| \underbrace{\Pr_{x \leftarrow \mathscr{D}_{1,\lambda}}[\mathscr{A}(X) = 1]}_{\substack{\text{probability of} \\ \text{being right}}} - \underbrace{\Pr_{x \leftarrow \mathscr{D}_{0,\lambda}}[\mathscr{A}(X) = 1]}_{\substack{\text{probability of} \\ \text{being mistaken}}} \Big|.$$

We say that $\mathscr{D}_{0,\lambda}$ and $\mathscr{D}_{1,\lambda}$ are *computationally indistinguishable* if for any distinguisher $\mathscr{A}$ running in $\mathrm{O}(\lambda^c)$ for some $c > 0$[2] its advantage $\mathrm{Adv}_{\mathscr{A}}$ is a $\mathrm{o}(1/\lambda^c)$ for some $c > 0$.[3]

Our goal now is to extend the One-Time PAD to messages m larger than the key k. We want to construct some function G that takes as input the key $k \in \{0,1\}^n$ and expend it to a string $G(k) \in \{0,1\}^\ell$ for some $\ell > k$ that is computationally hard to distinguish from a uniform random string. This is called a *PGR* or *pseudo-random generator.*

Definition 5. A *pseudo-random generator* is a pair of poly-time algorithms (Setup, G) such that:

- ▷ Setup is an algorithm that takes as input a security parameter λ (taken as a string 1^λ of length λ, *i.e.* we write λ in unary) and returns a public parameter;
- ▷ $G_\lambda : \{0,1\}^{n(\lambda)} \to \{0,1\}^{\ell(\lambda)}$ is an algorithm which takes a string k of length $n(\lambda)$ and return a string $G(k)$ of length $\ell(\lambda)$ with $\ell(\lambda) > n(\lambda)$.

such that

- ▷ G is deterministic;
- ▷ $\ell(\lambda) > n(\lambda)$ (we say that it is *expanding*)
- ▷ the distributions $\{\mathscr{U}(\{0,1\}^{\ell(\lambda)})\}_{\lambda\in\mathbb{N}}$ and $\{G(\mathscr{U}(\{0,1\}^{n(\lambda)}))\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable (we call it *pseudo-randomness*).

Another way of defining a pseudo-random generator is with *unpredictability* instead of *pseudo-randomness.*

Definition 6. This is the same definition as before but replacing pseudo-randomness with *unpredictability.*

[2]This means it is polynomial in λ, which we will write $\mathrm{poly}(\lambda)$

[3]This means it is negligible in terms of λ, which we will write $\mathrm{negl}(\lambda)$.

A PRG (Setup, G) is *unpredictable* if, for any index $i \in \{0, \ldots, \ell(\lambda)\}$ and any efficient adversary $\mathscr{A} : \{0,1\}^n \to \{0,1\}$, we have that:

$$\left| \Pr_{k \leftarrow \mathscr{U}(\{0,1\}^{n(\lambda)})} \left[\mathscr{A}(G(k)_{|i}) = G(k)_{i+1} \right] - \frac{1}{2} \right| = \mathrm{negl}(\lambda).$$

We can now prove that the two definitions are equivalent.

Theorem 2. The two definitions of a PRG are equivalent.

Proof. To simplify, we will remove the security parameter from the notations.

On one side, assume we have a predictor $\mathscr{A} : \{0,1\}^i \to \{0,1\}$ that succeeds in guessing $G(k)_{i+1}$ with non-negligible probability. We then construct a distinguisher $\mathscr{B}$ against pseudo-randomness as $\mathscr{B}$ receive a sample x from either $\mathscr{D}_0 = \mathscr{U}(\{0,1\}^\ell)$ or $\mathscr{D}_1 = G(\mathscr{U}(\{0,1\}^n))$: algorithm $\mathscr{B}$ runs $\mathscr{A}$ on input $x_{|i}$ and checks if $\mathscr{A}(x_{|i}) \overset{?}{=} x_{i+1}$. In that case, $\mathscr{B}$ will return 1; otherwise it returns 0. What is the advantage of $\mathscr{B}$?

$$\begin{aligned} \mathrm{Adv}_{\mathscr{B}} &= \left| \Pr_{x \leftarrow \mathscr{D}_1} [\mathscr{B}(x) = 1] - \overbrace{\Pr_{x \leftarrow \mathscr{D}_0} [\mathscr{B}(x) = 1]}^{1/2} \right| \\ &= \left| \Pr_{x \leftarrow \mathscr{D}_1} [\mathscr{A}(x_{|i}) = x_{i+1}] - \frac{1}{2} \right|. \end{aligned}$$

This is the definition of the predictability advantage of $\mathscr{A}$ (which is non-negligible by assumption).

Next, we will use a technique called an *Hybrid Argument* (due to Yao in '82). Assume we have a distinguisher $\mathscr{A}$ such that

$$\mathrm{Adv}_{\mathscr{A}} = \left| \Pr_{x \leftarrow \mathscr{D}_1} [\mathscr{A}(x) = 1] - \Pr_{x \leftarrow \mathscr{D}_0} [A(x) = 1] \right|$$

is non-negligible, say $\mathrm{Adv}_{\mathscr{A}} \geq \varepsilon$. We then define $\ell+1$ distributions

$(\mathscr{D}_i)_{i=0,\dots,\ell}$ as

$$\mathscr{D}_i := \left\{ x \in \{0,1\}^\ell \;\middle|\; \begin{array}{l} x_{|i} = G(k)_{|i} \text{ for } k \leftarrow \mathscr{U}(\{0,1\}^n) \\ x_{|i+1,\dots,\ell} \leftarrow \mathscr{U}(\{0,1\}^{\ell-i}) \end{array} \right\}.$$

We then have, by all the terms cancelling (this is a telescoping sum), that:

$$\begin{aligned} \varepsilon \le \mathrm{Adv}_{\mathscr{A}}(\mathscr{D}_0, \mathscr{D}_n) &= \left| \sum_{i=0}^{\ell} \Big(\Pr_{x \leftarrow \mathscr{D}_{i+1}}[\mathscr{A}(x) = 1] - \Pr_{x \leftarrow \mathscr{D}_i}[\mathscr{A}(x) = 1] \Big) \right| \\ &\le \sum_{i=0}^{\ell} \left| \Pr_{x \leftarrow \mathscr{D}_{i+1}}[\mathscr{A}(x) = 1] - \Pr_{x \leftarrow \mathscr{D}_i}[\mathscr{A}(x) = 1] \right| \\ &\le \sum_{i=0}^{\ell} \mathrm{Adv}_{\mathscr{A}}(\mathscr{D}_i, \mathscr{D}_{i+1}). \end{aligned}$$

By the pigeonhole principle, we have that there exists an $i \in \{0, \dots, \ell\}$, such that

$$\left| \Pr_{x \leftarrow \mathscr{D}_{i+1}}[\mathscr{A}(x) = 1] - \Pr_{x \leftarrow \mathscr{D}_i}[\mathscr{A}(x) = 1] \right| \ge \frac{\varepsilon}{\ell + 1}.$$

As ε is non-negligible and $\ell + 1$ being polynomial in λ, we have that $\varepsilon/(\ell+1)$ is non-negligible. How to turn this into a predictor for i? Let us define $\mathscr{B}_i$ as a predictor which is given $G(k)_{|i}$ and supposed to predict $G(k)_{i+1}$. Algorithm $\mathscr{B}_i$ will computes $x \in \{0,1\}^\ell$ with $x \leftarrow G(k)_{|i} \,||\, y$ where $y \leftarrow \mathscr{U}(\{0,1\}^{\ell-i})$. Then $\mathscr{B}_i$ runs algorithms $\mathscr{A}$ on input x, and $\mathscr{A}$ returns a bit $b \in \{0,1\}$ and $\mathscr{B}_i$ outputs a prediction x_{i+1} for $G(k)_{i+1}$ if $b = 1$ and $1 - x_{i+1}$

otherwise. What is the prediction advantage of $\mathscr{B}_i$?

$$\begin{aligned}
&\Pr[\mathscr{B}_i(G(k)_{|i}) = G(k)_{i+1}]\\
&= \Pr\left[\begin{array}{c}\mathscr{A}(x) = 0 \wedge x_{i+1} = 1 - G(k)_{i+1}\\ \vee \\ \mathscr{A}(x) = 1 \wedge x_{i+1} = G(k)_{i+1}\end{array}\right]\\
&= \Pr_{x\leftarrow\mathscr{D}_i}[\mathscr{A}(x) = 0 \wedge x_{i+1} = 1 - G(k)_{i+1}]\\
&\quad + \Pr_{x\leftarrow\mathscr{D}_i}[\mathscr{A}(x) = 1 \wedge x_{i+1} = G(k)_{i+1}]\\
&= \frac{1}{2}\Pr_{x\leftarrow\bar{\mathscr{D}}_{i+1}}[\mathscr{A}(x) = 0] + \frac{1}{2}\Pr_{x\leftarrow\mathscr{D}_i}[\mathscr{A}(x) = 1]\\
&= \frac{1}{2}\Big(\Pr_{x\leftarrow\mathscr{D}_{i+1}}[\mathscr{A}(x) = 1] + 1 - \Pr_{x\leftarrow\bar{\mathscr{D}}_{i+1}}[\mathscr{A}(x) = 1]\Big)
\end{aligned}$$

.

where we write $\bar{\mathscr{D}}_{i+1}$ is the "flipped" of $\mathscr{D}_{i+1}$. We have that:

$$\begin{aligned}
&\Pr_{x\leftarrow\mathscr{D}_i}[\mathscr{A}(x) = 1]\\
&= \Pr_{x\leftarrow\mathscr{D}_i}[\mathscr{A}(x) = 1 \wedge x_{i+1} = G(k)_{i+1}]\\
&\quad + \Pr_{x\leftarrow\mathscr{D}_i}[\mathscr{A}(x) = 1 \wedge x_{i+1} = 1 - G(k)_{i+1}]\\
&= \frac{1}{2}\Big(\Pr_{x\leftarrow\mathscr{D}_i}[\mathscr{A}(x) = 1] + \Pr_{x\leftarrow\bar{\mathscr{D}}_{i+1}}[\mathscr{A}(x) = 1]\Big),
\end{aligned}$$

thus

$$\Pr_{x\leftarrow\bar{\mathscr{D}}_{i+1}}[\mathscr{A}(x) = 1] = 2\Pr_{x\leftarrow\mathscr{D}_i}[\mathscr{A}(x) = 1] - \Pr_{x\leftarrow\mathscr{D}_{i+1}}[\mathscr{A}(x) = 1].$$

Hence,

$$\Pr[\mathscr{B}_i(G(k)_{|i}) - G(k)_{i+1}] =$$
$$\frac{1}{2}\Pr_{x\leftarrow\mathscr{D}_{i+1}}[\mathscr{A}(x) = 1] + 1 - 2\Pr_{x\leftarrow\mathscr{D}_i}[\mathscr{A}(x) = 1] + \Pr_{x\leftarrow\mathscr{D}_{i+1}}[\mathscr{A}(x) = 1].$$

Finally, we can conclude that:

$$\operatorname{Adv}_{\mathscr{A}}(\mathscr{D}_i, \mathscr{D}_{i+1}) = \left| \Pr[\mathscr{B}_i(G(k)_{|i}) = G(k)_{i+1}] - \frac{1}{2} \right| \geq \frac{\varepsilon}{n}.$$

$\square$

Example 2. Let us go back to the One-Time PAD example. As said before, to get information-theoretic security, one needs the key's bit length to be no smaller than the message's length.

Now, how do we use the PRG to have a secure protocol? We encode using the PRG:

$$\operatorname{Enc}_k(m \in \{0,1\}^\ell) = m \oplus G(k) \in \{0,1\}^\ell.$$

We can use a key of length 128 bits but encode a 1 Gb message.

If we have a PRG $G : \{0,1\}^n \to \{0,1\}^{n+1}$ where n is the length of the key, then we can call G on itself a lot of times to get a string of any length $\ell > n$. (This is likely to be proven in the tutorials.)[4]

As seen before with the One-Time PAD, this kind of encryption can only be used once: you cannot re-use the key to encrypt multiple messages.

Definition 7. An encryption scheme $(\operatorname{KeyGen}, \operatorname{Enc}, \operatorname{Dec})$ is called *secure against a single message chosen plain-text attack* if, for all polynomial-time adversary $\mathscr{A}$, and all m_0, m_1 chosen by $\mathscr{A}$, we have that the two distributions are computationally indistinguishable:

$$\big(\operatorname{Enc}(k, m_0)\big)_{k \leftarrow \operatorname{KeyGen}(\,)} \simeq^{\mathrm{c}} \big(\operatorname{Enc}(k, m_1)\big)_{k \leftarrow \operatorname{KeyGen}(\,)}.$$

[4]The teacher gave us an explanation on how we can double the length of a string, then it is easy to go from 128 to 2^{30} bits. However, that construction is still using the $n \to n+1$ construction 2^{23} times.

Remark 3. Another way of thinking about this kind of security is to imagine two players, the adversary $\mathscr{A}$ and the challenger $\mathscr{C}$.

- ▷ The challenger generates a secret key $k \in \{0,1\}^n$ (which we assume to be uniform) and a uniform bit $b \leftarrow \mathscr{U}(\{0,1\})$.
- ▷ The adversary give two messages m_1 and m_2 to $\mathscr{C}$.
- ▷ Then, the challenger encrypt m_b using the key, and gives it to $\mathscr{A}$.
- ▷ Finally, $\mathscr{A}$ tries to "guess" b (*i.e.* which message was encrypted (m_0 or m_1).

Writing $b^\star$ for the guess of the adversary, we obtain a different formulation for the advantage of $\mathscr{A}$:

$$\operatorname{Adv}(\mathscr{A}) = \Big|2 \times \Pr[b^\star = b] - 1\Big|.$$

This definition of the advantage is equivalent (*c.f.* tutorials) to the one used before :

$$\operatorname{Adv}(\mathscr{A}) = \Big|\Pr[\mathscr{A} \text{ guesses } 1 \mid b = 0] - \Pr[\mathscr{A} \text{ guesses } 1 \mid b = 1]\Big|.$$

Proposition 1. The PRG-based construction is secure against a single message chosen plain-text attack.

Proof. We want to show that, if there is an attacker against the PRG-based scheme, then there is a distinguisher fo the PRG. We will use the "encryption security game" analogy in this proof. We define two games:

- ▷ Let Hybrid_0 be the game where $\mathscr{C}$ uses m_0.
- ▷ Let Hybrid_4 be the game where $\mathscr{C}$ uses m_1.

which we then complete with three other "intermediate" games:

- ▷ Let Hybrid_1 be the game similar to Hybrid_0 except that $c =$

$m_0 \oplus G(k)$ is replaced by $c = m_0 \oplus u$ where $u \leftarrow \mathscr{U}(\{0,1\}^\ell)$.

- ▷ Let Hybrid_2 be the game similar to Hybrid_1 except that m_0 is changed with m_1 and thus $c = m_1 \oplus u$.
- ▷ Let Hybrid_3 be the game similar to Hybrid_2 except that $c = m_1 \oplus u$ is replaced with $c = m_1 \oplus G(k)$.

We define

$$p_n := \Pr[\mathscr{A} \text{ guesses 1 in the game } \text{Hybrid}_n].$$

The goal is to show that $|p_0 - p_4|$ is negligible. To prove that we will prove that $|p_0 - p_1|$, $|p_1 - p_2|$, $|p_2 - p_3|$ and $|p_3 - p_4|$ are all negligible (we will then conclude by the triangle inequality). This strategy is called *Game Hopping*. By symmetry, we only need to consider $|p_0 - p_1|$ and $|p_1 - p_2|$.

- ▷ Consider the games Hybrid_0 and Hybrid_1. If $\mathscr{A}$ can see the difference between the two cyphers, then it can be used to break the PRG. To prove this, we proceed by reduction. We introduce a new player, $\mathscr{B}$, who will pretend to be $\mathscr{A}$ from the point of view of $\mathscr{C}$ and *vice-versa*.

  The players are then:

  - $\mathscr{A}$ is the encryption adversary;
  - $\mathscr{C}$ is the PRG challenger;
  - $\mathscr{B}$ is both the encryption challenger and the PRG adversary.

  We consider two cases: the “PRG” case and the “Uniform” case (depending on the choice for the key used to cypher the message. From the point of view of $\mathscr{A}$,

  - in the “PRG” case, it should be exactly as in Hybrid_0;
  - in the “Uniform” case, it should be exactly as in Hybrid_1.

The game will take place in the following way:

- $\mathscr{A}$ will give $\mathscr{B}$ two messages m_0 and m_1;
- $\mathscr{C}$ will give $\mathscr{B}$ a key y with the required length (either generated uniformly in the "Uniform" case, or with the PRG in the "PRG" case).
- $\mathscr{B}$ encrypts the message m_0 using the key y, and gives it to $\mathscr{A}$.
- $\mathscr{A}$ sends its guess $b^\star$ to $\mathscr{B}$, who directly sends it to $\mathscr{C}$.

Because $\mathscr{A}$'s view is consistent, it behaves as unexpected in Hybrid_0 or Hybrid_1. This means that:

- in the "PRG" case, $\mathscr{B}$ outputs 1 iff $\mathscr{A}$ outputs 1, which happens with probability p_0.
- in the "PRG" case, $\mathscr{B}$ outputs 1 iff $\mathscr{A}$ outputs 1, which happens with probability p_1.

If $|p_0 - p_1| = |\Pr[\mathscr{B} \leftarrow 1 \mid \text{PRG case}] - \Pr[\mathscr{B} \leftarrow 1 \mid \text{Uniform case}]|$ is non-negligible, then $\mathscr{B}$ breaks the PRG. And, if $\mathscr{A}$ is efficient, so is $\mathscr{B}$. Thus, if the PRG is secure, then $|p_0 - p_1|$ is negligible.

▷ For the games Hybrid_1 and Hybrid_2, we will prove that $p_1 = p_2$. As u is chosen uniformly, then $\mathscr{A}$ receives a uniform cypher c in both games. Then, as $\mathscr{A}$ has the same view, it has the same behavior. The rest of the proof is exactly the one for the perfect security of the One-Time PAD.

□

1.1 How to get PRGs? Cryptographic assumptions.

One example of a PRG is called RC4 (defined by Rivest in '87). It has some weaknesses. This PRG was used in WEP, an very old WiFi protocol (still used by 2 % of WiFi routers), and it has been totally broken (the WEP protocol added weaknesses on top of RC4's). It is also used by Bittorent. The state of the art is Salsa20 (software) or Trivium (hardware).

Definition 8. A function $f : \{0,1\}^k \to \{0,1\}^\ell$ is called *one way* (with no relation between l and k) if it is computable in polyomial-time and for any polynomial-time adversary $\mathscr{A}$, its advantage

$$\operatorname{Adv}(\mathscr{A}) = \Pr_{x \leftarrow \mathscr{U}(\{0,1\}^k)} [\mathscr{A}(f(x)) = x' \text{ where } f(x) = f(x')]$$

is negligible.

We have that:

- if there exists a PRG, then there is a one-way function
- if there is a one-way function, then there is a PRG (Goldreich-Levin hard-cord bits).

There also exists explicit universal functions: if a one-way function exists, then the universal function is one way.

This problem is connected to the **P** *vs.* **NP** problem (existence of one-way function implies $\mathbf{P} \neq \mathbf{NP}$).

Definition 9 (Discrete Logarithm Problem, DLP). The DLP is defined relative to a prime-order cyclic group G with a generator $g \in G$. This means that

$$G = \{g^k \mid k = 0, \dots, p-1\},$$

where $p = |G|$ is a prime number. The group G and the element g are publicly known. The goal is, given $h \in G$, find a x such that $g^x = h$.

Example 3. In $(\mathbb{Z}/p\mathbb{Z}, +)$, the DLP problem is quite easy.

In $G_p := ((\mathbb{Z}/p\mathbb{Z})^\times, \times)$ is cyclic of order $p-1$, but $p-1$ is not necessarily a prime! We take a prime p such that $p = 2q+1$ where q is prime (such primes are called *safe primes*). We have that

$$G_p = \{g, g^2, g^3, \dots, g^{p-1}\}$$

and

$$G_q = \{(g^2)^0, (g^2)^2, \dots, (g^2)^k, \dots, \overbrace{(g^2)^{(p-1)/2}}^{p^q}\}.$$

The group G_q is cyclic with prime order q. To find a generator for G_q, we simply sample uniformly an element g_0 of $(\mathbb{Z}/p\mathbb{Z})^\star$, then take $h := g_0^2$. This is in fact a generator as long as $g_0 \notin \{-1, 1\}$.

In the 2000s, cryptographers started using the group of elements of an elliptic curve over a finite field. For prime order subgroups of $((\mathbb{Z}/p\mathbb{Z})^\star, \times)$, the best known algorithms cost

$$\exp(\tilde{\mathrm{O}}(\sqrt[3]{\ln |G|})) \ll \exp(\mathrm{O}(\ln |G|)).$$

The other cost is for generic “black box” groups (hardness of DLP). This blackbox algorithm is the best known algorithm for elliptic curves with $\log_2 |G| \approx 256$.

Thus, p and q have to be quite large to be hard-to-solve (around 4 096 bits) on the case of prime order subgroups of $(\mathbb{Z}/p\mathbb{Z})^\star$. Given $h = g^x$, there is a baby-step-giant-step algorithm to find x.

- ▷ We start by computing $\{g^0, g, g^1, \dots, g^{\sqrt{q}}\}$ (baby steps);
- ▷ Then, we compute $\{hg^{-\sqrt{q}}, hg^{-2\sqrt{q}}, hg^{-3\sqrt{q}}, \dots, hg^{-\sqrt{q}\sqrt{q}}\}$ (giant steps).

The cost for each step is around $\sqrt{q}$. As $h = g^x = g^{x_0 + \sqrt{q}x_1}$, then we have that $g^{x_0} = h \cdot g^{\sqrt{-q}x_1}$. Each of these elements is in one set.

Then, if we find two elements g^{x_0} in the baby steps and $h \cdot g^{-\sqrt{q}x_1}$ in the giant steps that are equal, then we get $h = g^{x_0} \cdot g^{\sqrt{q}x_1} = g^{x_0 + \sqrt{q}x_1}$, thus we solve the DLP solution.

That's a $\mathrm{O}(\sqrt{|G|})$ time algorithm for finding the DLP.[5]