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

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

□

[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.

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:

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

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.

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:

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

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 $\{\mathcal{U}(\{0,1\}^{\ell(\lambda)})\}_{\lambda\in\mathbb{N}}$ and $\{G(\mathcal{U}(\{0,1\}^{n(\lambda)}))\}_{\lambda\in\mathbb{N}}$ 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.*

A PRG (Setup, G) is *unpredictable* if, for any index $i \in \{0,\ldots,\ell(\lambda)\}$

[2] This means it is polynomial in λ.

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

and any efficient adversary $\mathscr{A} : \{0,1\}^n \to \{0,1\}$, we have that:

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

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 $\mathscr{B}$?

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

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

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

is non-negligible, say $\operatorname{Adv}_{\mathscr{A}} \geq \varepsilon$. 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:

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

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

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

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 $\mathscr{B}_i$?

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

.

where we write $\bar{\mathscr{D}}_{i+1}$ is the “flipped” of $\mathscr{D}_{i+1}$. □