

Private-Key Encryption

Perfectly secret ciphers are not practical because their keys are too long.

Kerckhoffs has another principle:

A cipher must be practically, if not mathematically, indecipherable.

Today this means Computational Security.

1. A cipher is required to be secure against efficient adversaries that run in a feasible amount of time.
2. Adversaries are allowed to succeed with negligible probability, meaning they will never really succeed.

A scheme is (t, ϵ) -secure if every adversary running for time $\leq t$ succeeds in breaking the scheme with probability $\leq \epsilon$.

Another approach to practical security uses a security parameter n that may tend to ∞ .

“Efficient” or “feasible” means the adversary runs in probabilistic polynomial time in n .

“Negligible probability” means reciprocal of a polynomial in n .

These relaxations of security requirements rule out brute force attacks.

Definition: f is *negligible* if for every polynomial $p(\cdot)$ with positive leading term there exists an integer N so that $f(n) < 1/p(n)$ for all $n > N$.

Proposition 3.6: The sum of two negligible functions is negligible. A polynomial times a negligible function is negligible.

A scheme is *secure* if for every PPT adversary performing a specified attack, its probability of success is negligible.

Security is usually proved by reduction to a simple problem considered to be hard:

To show Π is secure, we assume that some efficient adversary \mathcal{A} can break it with probability $\epsilon(n)$.

Design an efficient algorithm \mathcal{A}' to solve a hard problem X using \mathcal{A} as a subroutine. Given an instance x of X , \mathcal{A}' builds an instance of Π and gives it to \mathcal{A} to break. When \mathcal{A} returns with its answer, \mathcal{A}' uses the answer to solve x , at least with probability $1/p(n)$.

Thus, if $\epsilon(n)$ is not negligible, then \mathcal{A}' solves X with non-negligible probability $\epsilon(n)/p(n)$ with the help of \mathcal{A} , which contradicts the assumption that X is hard.

Therefore, assuming X is hard, Π is computationally secure.

A *private-key encryption scheme* with security parameter n is $(\text{Gen}, \text{Enc}, \text{Dec})$ with

1. Gen generates a key $k \leftarrow \text{Gen}(1^n)$ of length $|k| \geq n$.
2. $c \leftarrow \text{Enc}_k(m)$ for $m \in \{0, 1\}^*$.
3. $m := \text{Dec}_k(c)$.

Require $\forall n \forall k \forall m$ we have $\text{Dec}_k(\text{Enc}_k(m)) = m$.

$(\text{Gen}, \text{Enc}, \text{Dec})$ might be defined only for messages of fixed length $\ell(n)$.

Eavesdropping Indistinguishability

Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be a private-key encryption scheme. Let \mathcal{A} be an adversary.

$\text{PrivK}_{\mathcal{A}, \Pi}^{\text{eav}}(n)$ is this experiment:

1. Given 1^n , \mathcal{A} outputs $m_0, m_1 \in \mathcal{M}$ of same length.
2. Let $k \leftarrow \text{Gen}(1^n)$ and $b \leftarrow \{0, 1\}$. Compute $c \leftarrow \text{Enc}_k(m_b)$ and give c to \mathcal{A} .
3. \mathcal{A} outputs a bit b' .
4. The output is 1 if $b' = b$, else 0. Write $\text{PrivK}_{\mathcal{A}, \Pi}^{\text{eav}} = 1$ and say \mathcal{A} *succeeded* if the output is 1.

Definition: Π has *indistinguishable encryptions in the presence of an eavesdropper* if for all PPT \mathcal{A} there is a negligible function negl such that

$$\Pr [\text{PrivK}_{\mathcal{A}, \Pi}^{\text{eav}}(n) = 1] = \frac{1}{2} + \text{negl}(n),$$

where the probability is taken over all random coins used by \mathcal{A} and in the experiment.

This definition has an equivalent form which says that \mathcal{A} behaves the same way (gives the same output) when given m_0 as it does when given m_1 .

We will need the notion of pseudorandomness before we can construct any encryption schemes with this property.

Pseudorandomness

A string is pseudorandom if it looks like a random string as long as you look at it only in polynomial time.

In more detail, a probability distribution on strings (of length $\ell(n)$) is pseudorandom if no polynomial-time algorithm (called a *distinguisher*) can distinguish a string chosen with this distribution from a string of length $\ell(n)$ chosen at random with a uniform distribution.

Although no specific string is either random or pseudorandom, we sometimes abuse the notation.

Our goal is to design an encryption scheme like the one-time pad, but with a pseudorandom key instead of a random one.

A pseudorandom generator will produce a long pseudorandom key from a short random one.

Definition: Let $\ell(\cdot)$ be a polynomial. Let G be a deterministic polynomial time algorithm that for any input $s \in \{0, 1\}^n$ outputs a string of length $\ell(n)$. Call G a *pseudorandom generator* if

1. We have $\ell(n) > n$ for all n .
2. For all PPT distinguishers D there is a negligible function negl such that

$$|\Pr[D(r) = 1] - \Pr[D(G(s)) = 1]| \leq \text{negl}(n),$$

where $r \leftarrow \{0, 1\}^{\ell(n)}$, the seed $s \leftarrow \{0, 1\}^n$, and the probabilities are taken over the random coins used by D and the choices of r and s .

The function $\ell(\cdot)$ is called the *expansion factor* of G .

It is easy to distinguish between a random string and a pseudorandom string (whose generator is known), given unlimited time.

Suppose G accepts a (seed) string of length n and outputs a pseudorandom string of length $2n$.

A distinguisher D , given input string w , searches all of $\{0, 1\}^n$ seeking an s with $G(s) = w$. D outputs 1 if such s is found, else 0.

If w was generated by G , then D outputs 1.

If w was a random string, then D outputs 0 usually. It would output 1 only if there happened to be a string s so that $G(s) = w$. This event happens with probability at most $2^n/2^{2n} = 2^{-n}$. (Perhaps G is not one-to-one.) Thus

$$|Pr[D(r) = 1] - Pr[D(G(s)) = 1]| \geq 1 - 2^{-n},$$

which is not negligible.

Polynomial time distinguishers do not have time to perform a brute force attack trying all possible keys s . Therefore, pseudorandom strings are just as good as truly random ones, so long as the seed remains secret and attackers run in polynomial time.

We do not actually know how to prove the existence of pseudorandom generators, but we believe they exist. We say more in Chapter 6, after we study one-way functions.

We are now ready to construct a fixed-length encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper.

Construction 3.15.

Let G be a pseudorandom generator with expansion factor ℓ . Define a private-key encryption scheme for messages of length ℓ as follows:

Gen: On input 1^n , choose $k \leftarrow \{0, 1\}^n$ uniformly and random and output it as the key.

Enc: On input $k \in \{0, 1\}^n$ and a message $m \in \{0, 1\}^{\ell(n)}$, output the ciphertext $c := G(k) \oplus m$.

Dec: On input $k \in \{0, 1\}^n$ and a ciphertext $c \in \{0, 1\}^{\ell(n)}$, output the plaintext $m := G(k) \oplus c$.

Theorem 3.16. If G is a pseudorandom generator, then Construction 3.15 gives a fixed-length private-key encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper.

Sketch of proof of Theorem 3.16.

Let Π be Construction 3.15. We show that if there is a PPT adversary \mathcal{A} for which Π does not have indistinguishable encryptions in the presence of an eavesdropper, then we can construct a PPT algorithm that can distinguish the output of G from a truly random string.

If Π used a truly random string instead of the pseudorandom string $G(k)$, then this modified Π would be just a one-time pad, so \mathcal{A} would be unable to distinguish between two plaintexts with probability greater than $1/2$. This would mean that \mathcal{A} is somehow distinguishing $G(k)$ from a truly random string.

When the distinguisher D is given a string w , it runs \mathcal{A} to get two messages m_0, m_1 of length $|w|$. D chooses a random bit b and sets $c := w \oplus m_b$. It gives c to \mathcal{A} and gets a bit b' . D outputs 1 if $b' = b$, else 0.

Variable-length messages

Definition. A deterministic polynomial-time algorithm G is a *variable output-length pseudo-random generator* if:

1. When s is a string and ℓ is a positive integer, $G(s, 1^\ell)$ outputs a string of length ℓ .
2. For all s, ℓ, ℓ' with $\ell < \ell'$, $G(s, 1^\ell)$ is a prefix of $G(s, 1^{\ell'})$.
3. If we let $G_\ell(s) = G(s, 1^{\ell(|s|)})$, then for every polynomial $\ell(\cdot)$, $G_\ell(s)$ is a pseudorandom generator with expansion factor ℓ .

Modify Construction 3.15 in the obvious way: $c := G(k, 1^{|m|}) \oplus m$.

Encryption schemes that use a pseudorandom generator are sometimes called *stream ciphers*.

One problem with stream ciphers arises when multiple messages are enciphered, each using the same key. Construction 3.15, in which Enc is deterministic, becomes insecure in this situation. For example, an adversary could easily distinguish between enciphering one message twice and two different messages.

Enc must not be deterministic if the key is allowed to be reused. Encryption must be probabilistic.

Two ways to handle multiple encryption with a deterministic Enc:

Synchronized mode: Remember the state. Treat the messages as one long message.

Unsynchronized mode: Let G take two inputs, a seed s and an initialization vector IV . Send IV in the clear as the first block of ciphertext.

Chosen Plaintext Attacks

are modeled by giving the adversary \mathcal{A} oracle access to $\text{Enc}_k(\cdot)$, written as $\mathcal{A}^{\text{Enc}_k(\cdot)}$.

\mathcal{A} queries the oracle by providing m ; the oracle replies with $c \leftarrow \text{Enc}_k(m)$. When $\text{Enc}_k(\cdot)$ is randomized, it used fresh coins (new random choices) each time it is called.

$\text{PrivK}_{\mathcal{A}, \Pi}^{\text{cpa}}(n)$ is this experiment:

1. Run $k \leftarrow \text{Gen}(1^n)$ to get a key.
2. Given 1^n and oracle access to $\text{Enc}_k(\cdot)$, \mathcal{A} outputs $m_0, m_1 \in \mathcal{M}$ of same length.
3. Let $b \leftarrow \{0, 1\}$. Compute $c \leftarrow \text{Enc}_k(m_b)$ and give c to \mathcal{A} .
4. \mathcal{A} continues to have oracle access to $\text{Enc}_k(\cdot)$ and outputs a bit b' .
5. The output is 1 if $b' = b$, else 0. Write $\text{PrivK}_{\mathcal{A}, \Pi}^{\text{cpa}} = 1$ and say \mathcal{A} *succeeded* if the output is 1.

Definition: Π has *indistinguishable encryptions under a chosen plaintext attack* if for all PPT \mathcal{A} there is a negligible function negl such that

$$\Pr \left[\text{PrivK}_{\mathcal{A}, \Pi}^{\text{cpa}}(n) = 1 \right] = \frac{1}{2} + \text{negl}(n),$$

where the probability is taken over all random coins used by \mathcal{A} and in the experiment.

No deterministic encryption scheme has indistinguishable encryptions under a chosen plaintext attack.

Real world: Midway Island = AF.

Proposition 3.22. Any private-key encryption scheme with indistinguishable encryptions under a chosen plaintext attack also has indistinguishable *multiple* encryptions under a chosen plaintext attack.

To construct a CPA-secure encryption scheme we need the notion of pseudorandom function.

A keyed function is a function $F(k, x)$ of two variables. The first variable k is the key.

For simplicity, we assume $F(k, x)$ is defined only when $|k| = |x|$ and that, when this is so, we also have $|F(k, x)| = |x|$.

We assume that $F(k, x)$ is efficient, that is, there is a deterministic polynomial time algorithm to compute it given input k and x .

We will look at the family of parametrized functions $F_k(\cdot)$ defined by $F_k(x) = F(k, x)$.

We say that F is *pseudorandom function* if $F_k(\cdot)$, where $k \leftarrow \{0, 1\}^n$ cannot be distinguished in polynomial time from a function f chosen uniformly at random from the set of all functions mapping n -bit strings to n -bit strings.

Construction 3.24.

Let F be a pseudorandom function. Define a private-key encryption scheme for messages of length n as follows:

Gen: On input 1^n , choose $k \leftarrow \{0, 1\}^n$ uniformly at random and output it as the key.

Enc: On input $k \in \{0, 1\}^n$ and a message $m \in \{0, 1\}^n$, choose $r \in \{0, 1\}^n$ uniformly at random and output the ciphertext $c := (r, F_k(r) \oplus m)$.

Dec: On input $k \in \{0, 1\}^n$ and a ciphertext $c = (r, s)$, output the plaintext $m := F_k(r) \oplus s$.

Theorem 3.16. If F is a pseudorandom function, then Construction 3.24 gives a fixed-length private-key encryption scheme for messages of length n that has indistinguishable encryptions under a chosen-plaintext attack.

One can omit “fixed-length” in Theorem 3.16.

A function $F : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a *keyed permutation* if for every $k \in \{0, 1\}^n$ the function $F_k(\cdot)$ is one-to-one (and therefore a bijection). It is *efficient* if there are polynomial-time algorithms to compute both $F_k(x)$ and $F_k^{-1}(x)$ for every k .

Proposition 3.27. Every pseudorandom permutation is also a pseudorandom function.

Just as stream ciphers are constructed from pseudorandom generators, block ciphers are constructed from pseudorandom permutations.

Modes of operation of block ciphers:

ECB, CBC, OFB, CTR.