1 Preliminaries

Let $\mathcal{G}$, $\mathcal{E}$, $\mathcal{D}$ enote the key *generation*, *encription*, and *decription* algorithms, respectively. Let $\mathcal{M}$ denote a finite and non-empty message space and $\mathcal{K}$ a finite *key space* which is the range of $\mathcal{G}$. Clearly, the domain of $\mathcal{E}$ is $\mathcal{K} \times \mathcal{M}$, and the range of $\mathcal{D}$ is $\mathcal{M}$. Let $\mathcal{C}$ denote the range of $\mathcal{E}$.

The algorithm $\mathcal{E}$ is allowed to be probabilistic, so that $\mathcal{E}(m)$ might produce different outputs even if m is fixed. We write $c \leftarrow \mathcal{E}_k(m)$ to denote the (possibly probabilistic) process by which m is encripted with key k producing a ciphertext c.

We assume (without loss of generality) that $\mathcal{G}$ chooses $k \in \mathcal{K}$ uniformely, and thus we say the uniform distribution. Thus, K is defined as the random variable which is the output of $\mathcal{G}$. Similarly, M is the random variable that denotes the message being encripted at a particular time, nad $Pr\,[M = m]$ denotes the probability that the message takes on the value $m \in M$. The distribution of M is not set by the encryption scheme but simply reflects the probability of particular messages being sent by the different parties involved. K and M are assumed to be independent.

Fixing an encryption scheme and a distribution over $\mathcal{M}$ determines a distribution over $\mathcal{C}$ given by choosing a key $k \in \mathcal{K}$ (via $\mathcal{G}$) and a message $m \in \mathcal{M}$ (according to the distribution of M). We let C denote the random variable de∉ $gtheresultingciphertext, sothatforc \in \mathcal{C}$, $Pr\,[C = c]$ is the probability of producing the cyphertext c.

Definition 1 *An encryption scheme* $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ *with message space* $\mathcal{M}$ *is* ***perfectly secret*** *if, for every probability distribution over* $\mathcal{M}$*, every* $m \in \mathcal{M}$*, and every* $c \in \mathcal{C}$ *such that* $P(C = c) > 0$*:*

$$P\,(M = m \mid C = c) = P(M = m)$$

Theorem 1 *An encryption scheme is perfectly secret if the distribution of the ciphertex does not depend on the plaintext; i.e. for any* $m_1, m_2 \in \mathcal{M}$*, and every* $c \in \mathcal{C}$*:*

$$P\,(\mathcal{E}_K(m_1) = c) = P\,(\mathcal{E}_K(m_2) = c)$$

The theorem simply states that if the ciphertext contains no information whatsoever about the plaintext, i.e. that if it's impossible to distinguish an encryption of m_1 from an encription of m_2, then there is perfect secrecy.

Definition 2 *Let* $PrivK^{eav}_{\mathcal{A},\pi}$ *be the computable function given by the following effective procedure:*

- *Simulate an adversary* $\mathcal{A}$ *which outputs a pair* $m_0, m_1 \in \mathcal{A}$*.*
- *Generate* $k = \mathcal{G}(m_i)$*, with i chosen uniformily from* $\{0, 1\}$*.*
- *Let* $c \leftarrow \mathcal{E}_k(m_i)$*.*
- *Let* $\mathcal{A}$ *randomly output a bit b.*
- *If* $b = i$ *output* 1*, if* $b \neq i$ *output 0.*

Definition 3 *An encryption scheme $\pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ with message space $\mathcal{M}$ is perfectly indistinguishable if, for every agent $\mathcal{A}$ which outputs a pair $m_0, m_1 \in \mathcal{M}$,*

$$P\left(PrivK^{eav}_{\mathcal{A},\pi} = 1\right) = \frac{1}{2}$$

Theorem 2 *An encryption scheme is perfectly secret if and only if it is perfectly indistinguishable.*

1.1 One-time pad

The one-time pad encryption scheme, given $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^\ell$, is defined by:

- $\mathcal{G}$ chooses $k \in \mathcal{K}$ uniformily.
- $\mathcal{E}$, given $k \in \mathcal{K}$ and $m \in \mathcal{M}$, outputs $c := k \circ m$
- $\mathcal{D}$, given $k \in \mathcal{K}$ and $c \in \mathcal{C}$, outputs $m := k \circ c$

Here, $a \circ b$ denotes the bitwise exclusive-or (XOR) operation. The scheme is correct because, fixing k, for every m it holds

$$\mathcal{D}_k\left(\mathcal{E}_k(m)\right) = k \circ k \circ m = m$$

Theorem 3 *One-time pad encryption is perfectly secret.*

Proof: Observe that

$$\begin{aligned} P\left(C = c \mid M = m\right) &= P\left(\mathcal{E}_K(m) = c\right) \\ &= P\left(m \circ K = c\right) \\ &= P\left(K = m \circ c\right) \\ &= 2^{-\ell} \end{aligned}$$

where the final equality holds because K is chosen uniformily as one in 2^ℓ binary strings. Now observe that, fixing a distribution over $\mathcal{M}$, for any $c \in \mathcal{C}$,

$$\begin{aligned} P\left(C = c\right) &= \sum_{m \in \mathcal{M}} P\left(C = c \mid M = m\right) P\left(M = m\right) \\ &= 2^{-\ell} \sum_{m \in \mathcal{M}} P(M = m') \\ &= 2^{-\ell} \end{aligned}$$

Furthermore,

$$
\begin{aligned}
P\,(M = m \mid C = c) &= \frac{P\,(C = c \mid M = m)\,P(M = m)}{P(C = c)} \\
&= \frac{2^{-\ell} P(M = m)}{2^{-\ell}} \\
&= P(M = m)
\end{aligned}
$$

$\therefore$ The scheme is perfectly secret.

1.2 Limitations of perfect secrecy

An inherent drawback of perfect secrecy is that the key space must always be at least as large as the message space. If all keys are the same length, and the message space consists of strings of some fixed length, this implies that the key is at least as long as the message.

A second limitation is that a key k may only be used once, insofar as using it twice or more leaks information.

Theorem 4 *If $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a perfectly secret encryption scheme, then $|\mathcal{K}| \geq |\mathcal{M}|$.*

Proof: Assume the scheme is perfectly secret and $|\mathcal{K}| < |\mathcal{M}|$. Consider the uniform distribution over $\mathcal{M}$ and let $c \in \mathcal{C}$ be a ciphertext occurring with non-zero probability. Let $\mathcal{M}(c)$ be the set of all possible messages that are possible decryptions of c:

$$\mathcal{M}(c) := \{m : m = \mathcal{D}_k(c) \text{ for some } k \in \mathcal{K}\}$$

Clearly, $|\mathcal{M}(c)| \leq |\mathcal{K}| < |\mathcal{M}|$. Then there is some $m \in \mathcal{M}$ such that $m \notin \mathcal{M}(c)$. But then

$$P(M = m \mid C = c) = 0 \neq P(M = m)$$

which means the scheme is not perfeclty secret.

Informally, if the consequent were not true, then given a ciphertext c, at least one possible message m could be ruled out, which contradicts the perfect secrecy assumption.

Theorem 5 (Shannon's theorem) *Assume $|\mathcal{M}| = |\mathcal{C}| = |\mathcal{K}|$. Then a scheme is perfectly secret if and only if:*

- *Every $k \in \mathcal{K}$ is chosen uniformily by $\mathcal{G}$.*
- *For every $m \in \mathcal{M}, c \in \mathcal{C}$, there is a unique $k \in \mathcal{K}$ such that $\mathcal{E}_k(m)$ outputs c.*

(Proof in page 36.) This theorem provides a way to decide if a scheme is perfectly secret. The first condition is easy to check, and the second condition can be proven or contradicted without having to deal with any probabilities.

2 Private-key (Symmetric) Cryptograph

Definition 4 (Informal concrete definition) *A scheme is (t, ϵ)-secure if any adversary running for some amount of time $w \leq t$ succeedes in breaking the scheme with probability $p \leq \epsilon$.*

Definition 5 (Informal asymptotic definition) *A scheme is secure if any adversary running for time $O(p(n))$, with n a security parameter, succeedes in breaking the code with probability smaller than any inverse polynomial in n.*

Example 1 *Assume an adversary running for n^3 minutes can succeede in breaking the scheme with probability $2^{40}2^{-n}$. When $n \leq 40$, 40^3 minutes (approx. 6 weeks) breaks the code with probability 1, so this is not useful. When $n = 500$, an adversary running for 200 years breaks the scheme with probability roughly 2^{-500}.*

The security parameter increases the time required to run the scheme, as well as the length of the key, so a balance between the desired security and the practicality of the scheme is to be reached.

2.1 The asymptotic approach

Definition 6 *An algorithm is efficient if it runs in polynomial time.*

We allow all algorithms to be randomized or probabilistic. Any such algorithm may make a non-deterministic decision at any step of its execution. Randomness is essential to cryptography, to the point that honest parties must be probabilistic. Furthermore, randomization gives attackers additional power, and it makes sense to consider the strongest possible opponent.

Definition 7 *A function $\varphi : \mathbb{N} \to \mathbb{R}_0^+$ is negligible if for every polynomial $p \in \mathcal{P}$ there is an N such that $f(n) < \frac{1}{p(n)}$ for all $n > N$.*

Example 2 *The functions $2^{-n}, n^{-\log n}$ are negligible, though they approach zero at different rates. For instance, $2^{-n} < n^{-5}$ if and only if $n > 5 \log n$, with $n = 23$ being the least number which satisfies this. Alternatively, $n^{-\log n} < n^{-5}$ if and only if $\log n > 5$, with $n = 33$ being the least number which satisfies this.*

Theorem 6 *Let φ_1, φ_2 be negligible functions. Then $\varphi_1 + \varphi_2$ is negligible, and for any poltnomial p, $p(n) \cdot \varphi_1(n)$ is negligible.*

Proof: Complete.

The theorem guarantees that if certain event occurs with only negligible probability, such event occurs with negligible probability even if the experiment is repeated polynomially many times.

A corollary of the theorem is that if a function g is not negligible, then neither is the function $f = g/p$ for any positive polynomial p.

Definition 8 (Informal security definition) *A scheme is secure if, for every probabilistic polynomial-time adversary $\mathcal{A}$ carrying out an attack of some formally specified type, the probability that $\mathcal{A}$ succeeds (where "succeed" is formally specified) is negligible.*

The definition is asymptotic because it is posible that, for small values of n (the security parameter), an adversary may succeed with high probability. But the point is that there is some N such that, if $n > N$, the probability of the attacker succeeding becomes less than $1/p(n)$ for any polynomial p.

2.2 Computationally secure encryption

Definition 9 *A private-key encryption scheme is a tuple of probabilistic polynomial-time algorithms* $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ *such that:*

- $\mathcal{G}$ *takes as input* 1^n *(the security parameter written in unary) and outputs a key* $k \in \mathcal{K}$*, where it is assumed that* $|k| \geq n$*.*
- $\mathcal{E}$ *takes as input* $k \in \mathcal{K}$ *and* $m \in \{0,1\}^*$ *and outputs a* $c \in \mathcal{C}$*.*
- $\mathcal{D}$ *takes as input* $k \in \mathcal{K}, c \in \mathcal{C}$*, and outputs* $m \in \mathcal{M}$ *deterministically.*
- *For every* $k \leftarrow \mathcal{G}(1^n), n \in \mathbb{N}, m \in \{0,1\}^*$, $\mathcal{D}_g\left(\mathcal{E}_k(m)\right) = m$

If $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is such that $\mathcal{E}_k$ is defined only for messages $m \in \{0,1\}^{\ell(n)}$, we say $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a fixed-length private-key encryption scheme for messages of length $\ell(n)$.

Considering an adersary $\mathcal{A}$ that runs in polynomial time, we define $\text{Priv}^{eav}_{\mathcal{A},\pi}$ just as before, except that we now admit that $\mathcal{A}$ may guess which of two messages was encrypted with probability *negligibly* better than 50%. Since now the scheme is parametrized by n, the security parameter, we write

$$P(\text{PrivK}^{eav}_{\mathcal{A},\pi}(n) = 1)$$

to denote the probability that $\mathcal{A}$ correctly guesses which of two messages was encrypted.

Definition 10 *A private-key encryption scheme* $\pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ *has indistinguishable encryptions in the presence of an eavesdropper, or is EAV-secure, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there is a negligible function* φ *such that:*

$$\forall n \in \mathbb{N} : P\left[PrivK^{eav}_{\mathcal{A},\pi}(n) = 1\right] \leq \frac{1}{2} + \varphi(n)$$

where the probability is taken over the randomness used by $\mathcal{A}$ *and the randomness used in the experiment.*

2.3 Psuedorandom generators and stream ciphers

A pseudorandom generator G is a deterministic algorithm which transforms a short, uniform string called the seed into a longer, pseudorandom output string.

Definition 11 *Let* ℓ *a polynomial and* G *a deterministic polynomial-time algorithm such that, for any natural* n *and binary string* s *of length* n*,* $G(s)$ *is a string of length* $\ell(n)$*. We say* G *is a pseudorandom generator if:*

- *For every* n*,* $\ell(n) > n$*.*

- *For any PPT algorithm D, there is some $\varphi \in \mathcal{N}$ such that*

$$|P\left[D\left(G(s)\right) = 1\right] - P\left[D(r) = 1\right]| \leq \varphi(n)$$

where the first probability is taken over the uniform choice of s and the randomness of D, and the second probability is taken over the uniform choice of $r \in \{0,1\}^{\ell}(n)$ and the randomness of D.

We call ℓ the expansion factor of G.

A stream cipher is a pair of deterministic algorithms $(Init, GetBits)$ where:

- Init takes an input seed s, an optional initialization vector $\overrightarrow{v}$, and outputs an initial state q_0. GetBits takes as input q_i and outputs a bit y and an updated state q_{i+1}. (In practice, y is usually a block of several bits.)

Given ℓ an expansion factor, an algorithm G_ℓ may be defined as mapping inputs of length n to outputs of length $\ell(n)$. The algorithm runs Init and then repeatedly runs GetBits ℓ times.

Input : $s, \overrightarrow{v}$
Output : $\overrightarrow{y}$

$q_0 := Init(s, \overrightarrow{v})$
for $i = 1$ **to** ℓ **do**
    $(y_i, q_i) := GetBits(q_{i-1})$
od
return $\overrightarrow{y}$

2.4 Reduction proofs

If a problem is hard, then we may prove a scheme is secure by showing that any efficient adversary $\mathcal{A}$ is reducible to an algorithm $\mathcal{A}'$ which is a solution to the hard problem. Let X be a hard problem. A security proof may look as follows:

- Fix some efficient (PPT) adversary $\mathcal{A}$ attacking the scheme π, and let $\epsilon(n)$ be its success probability.
- Construct $\mathcal{A}'$ that attempts to solve X using $\mathcal{A}$ as a subroutine. So, given an input instance x of the problem X, $\mathcal{A}'$ will simulate for $\mathcal{A}$ an instance of π such that:
  - Running as a subroutine of $\mathcal{A}'$, $\mathcal{A}$ attacks π.
  - If $\mathcal{A}$ succeeds, then $\mathcal{A}'$ solves x at least with inverse polynomial probability $1/p(n)$.
- The first two items entail that $\mathcal{A}'$ solves X with probability $\epsilon(n)/p(n)$. If $\epsilon(n)$ is not negligible, then this last probability isn't either. Furthermore, if $\mathcal{A}$ is efficient, we have found an efficient algorithm $\mathcal{A}'$ to solve X, which contradicts the initial assumption.

- We conclude no efficient adversare $\mathcal{A}$ can succeed in breaking π with non-negligible probability. In other words, π is computationally secure.

2.5 A secure fixed-length encryption scheme

One-time padding consisted in using a uniformly sampled pad $\mathfrak{p} \in \{0,1\}^{\ell}$ to encrypt/decrypt a message as

$$m \overbrace{\rightarrow}^{\text{encryption}} c := m + \mathfrak{p} \overbrace{\rightarrow}^{\text{decryption}} m = c + \mathfrak{p}$$

An insight from previous sections is that we can generate a pseudorandom pad $\mathfrak{p}$ via a seed $\mathfrak{s}$, and share only $\mathfrak{s}$ as key. Thus, we defeat one of the limitations faced; namely, our key becomes shorter than our message. In terms of security, a pseudorandom string "looks random" to any polynomial-time adversary.

The scheme is as follows. Fix ℓ some message length and let G be a pseudorandom generator with expansion factor ℓ. Let $\pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ be as follows:

- $\mathcal{G}(1^n)$ outputs a uniform $k \in \mathcal{K}$ of length n, the security parameter.
- $\mathcal{E}$ works by computing $\mathfrak{p} \leftarrow G(k)$, and then $c := m + \mathfrak{p}$.
- $\mathcal{D}$ computes $\mathfrak{p} \leftarrow G(k)$ and then $m = \mathfrak{p} + c$.

Theorem 7 *If G is a pseudorandom generator, the scheme provided above is a fixed-length private-key encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper.*

Proof: Long. Page 89.

2.6 CPA Security

Definition 12 *A private-key encryption scheme $\pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ has indistinguishable encryptions under a chosen-plaintext attack, or is CPA-secure, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there is a function $\varphi \in \mathcal{N}$ such that:*

$$P\left[PrivK^{cpa}_{\mathcal{A},\pi}(n) = 1\right] \leq \frac{1}{2} + \varphi(n)$$

The experiment $\text{PrivK}^{\text{cpa}}_{\mathcal{A},\pi}$ is simply the chosen-plaintext attack experiment:

- $k \in \mathcal{K}$ is generated (secretely) and $\mathcal{A}$ is given access to $\mathcal{E}_k$.
- $\mathcal{A}$ outputs two messages of equal length, $m_0, m_1 \in \mathcal{M}$.
- $i \in \{0,1\}$ is chosen uniformily and m_i is encoded with $\mathcal{E}$.
- $\mathcal{A}$ continues to have oracle access to $\mathcal{E}$, and ventures (outputs) a guess $b \in \{0,1\}$.
- If $b = i$ the result is 1, zero otherwise.

Theorem 8 *If $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a private-key encryption scheme that is CPA-secure, it is CPA-secure for multiple encryptions.*

Theorem 9 *If $\pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a fixed-length private-key encryption scheme that is CPA-secure, there is an arbitrary length private-key encryption scheme $\pi' = (\mathcal{G}, \mathcal{E}', \mathcal{D})$.*

Proof: Assume (w.l.g.) that π encodes messages of 1 bit. Let $m = m_0 m_1 \dots m_\ell$ be a message of ℓ bits. Then $c = \mathcal{E}(m_0)\mathcal{E}(m_1) \dots \mathcal{E}(m_\ell)$ is an encryption of m. That the encryption is secure follows from the theorem preceding the one being proved here.

3 Constructing CPA-secure schemes

3.1 Pseudorandom functions

For convenience, let us use $\mathcal{B}$ to denote $\{0,1\}^*$, the set of all binary strings, and $\mathcal{B}_k$ to denote $\{0,1\}^k$ the set of all binary strings of length k.

A keyed function is a function $f : \mathcal{B}^2 \to \mathcal{B}$ where the first argument is called the key and is denoted by k. We say f is efficient if there is a polynomial-time algorithm that computes $f(k,x)$ for any $k, x \in \mathcal{B}$. Typically, we treat k as fixed and we sometimes write $f_k(x)$. The security parameter n dictates the key length, input length, and output length. That is, three functions $\ell_{key}, \ell_{in}, \ell_{out}$ are associated with f: for any $k \in \mathcal{B}_{\ell_{key}(n)}$, the functoin f_k is only defined for inputs $x \in \mathcal{B}_{\ell_{in}(n)}$. For simplicity, we assume each f_k is length-preserving, i.e. the lengths of input, output and key are equal and of length n.

Since each keyed function f induces a k-indexed family of functions $\mathcal{f}_k = \{(k, f_k) : k \in \mathbb{N}\}$, we may uniformly choose $k \in \mathcal{B}_n$ and consider only the resulting f_k. No adversary can distinguish whether it is interacting with a uniform choice of k producing f_k or with a uniform choice of f from the set of all keyed functions.

One may recall that there are $|B|^{|A|}$ functions of the form $g : A \to B$. Since a keyed, length-preserving function is of the form $f : \mathcal{B}_n^2 \to \mathcal{B}_n$, and $|\mathcal{B}_n| = 2^n$, there are $(2^n 2^n)^{2^n} = 2^{4n^2}$ keyed functions for any security parameter n. Similar reasoning gives 2^{n2^n} functions in $\mathcal{f}_k$.

From now on, we define:

$$\mathcal{F}_n := \{f \mid f : \mathcal{B}_n \to \mathcal{B}_n\}$$

The intuition behind a pseudorandom function is this. A polynomial-time distinguisher D is given an oracle $\mathcal{O}$ which is either equal to f_k (for uniform k) or equal to f uniformly chosen from $\mathcal{F}_n$. We assume the oracle is deterministic, and observe that D can ask only polynomially many queries to $\mathcal{O}$ if it is to remain a polynomial-time algorithm.

Definition 13 *Let $f : \mathcal{B}^2 \to \mathcal{B}$ be an efficient, length preserving, keyed function. F is a pseudorandom function if for all probabilistic polynomial-time distinguisher D, there is some $\varphi \in \mathcal{N}$ such that:*

$$\left| P\left[D^{F_k(\cdot)}(1^n) = 1\right] - P\left[D^{f(\cdot)}(1^n) = 1\right] \right| \leq \varphi(n)$$

where the first probability is taken over uniform choice of $k \in \mathcal{B}_n$ and the randomness of D, and the second probability is taken over uniform choice of $f \in \mathcal{F}_n$ and the randomness of D.

Informally, f is a pseudorandom function if an efficient distinguisher cannot distinguish it from an arbitrary function uniformly sampled from the function space with more than negligible probability. We must observe that the pseudorandom function f_k is determined by the key, and hence the inability to distinguish f_k, with k sampled uniformly from $\mathcal{B}_n$, from f sampled uniformly from $\mathcal{F}_n$, corresponds to an inability of inferring the key.

Example 3 *Let $f_k(x) = k + x$. If k is uniform then the output of $f_k(x)$ is uniform for any x. However, f is not pseudorandom because its values on any two points are correlated.*

Concretely, consider D querying $\mathcal{O}$ on x_1, x_2 to obtain $\mathcal{O}(x_1) = y_1, \mathcal{O}(x_2) = y_2$, outputting 1 iff $y_1 + y_2 = x_1 + x_2$.

If $\mathcal{O} = f_k$, for any k, then D outputs 1. If $\mathcal{O} = g$, with g chosen uniformly from $\mathcal{F}_n$, then the probability that $y_1 + y_2 = x_1 + x_2$ is the probability that $y_2 = x_1 + x_2 + y_1$, which is 2^{-n}.

So,

$$\left| P\left[D^{F_k(\cdot)}(1^n) = 1 \right] - P\left[D^{f(\cdot)}(1^n) = 1 \right] \right| = |1 - 2^{-n}|$$

which is not negligible.

Let $\mathcal{P}_n$ denote the set of all permutations (bijections) on $\mathcal{B}_n$. The cardinality of $\mathcal{P}$ is $(2^n)!$.

We say F is a keyed permutation if $\ell_{in} = \ell_{out}$ and for all $k \in \mathcal{B}_{\ell_{key}(n)}$, the function $f_k : \mathcal{B}_{\ell_{in}(n)} \to \mathcal{B}_{\ell_{out}(n)}$ is one-to-one (i.e. a permutation). We call ℓ_{in} the block length of f, and again we assume the input, output and key lengths are equal unless stated otherwise.

A keyed permutation f_k is efficient if there is a polynomial-time algorithm for computing $f_k(x) = y$ and $x = f_k^{-1}(y)$.

Theorem 10 *If f is a pseudorandom permutation and $\ell_{in}(n) \geq n$, then f is a pseudorandom function.*

A pseudorandom permutation is said to be **strong** if it is indistinguishable even when a distinguisher is given oracle access to its inverse.

Definition 14 *A block cipher is an instantiation of a strong pseudorandom permutation with some fixed key length and block length.*

A pseudorandom function/block cipher f can be used to build pseudorandom generators by evaluating f on a series of different inputs. In other words, $G(s) := f_s(1) \overline{+} f_s(2) \overline{+} \ldots \overline{+} f_s(\ell)$ pseudorandomly generates a binary string of length ℓ from a seed s, where $\overline{+}$ denotes the bitwise or.

More generally, a stream cipher $(Init, GetBits)$ with input vector v can be built from a pseudorandom function f as follows:

- Let $Init$ with input $s, v \in \mathcal{B}_n$ set $q_0 := (s, v)$.
- On input $q_i = (s, v_i)$ let $GetBits$ compute $v_{i+1} = v_i + 1$, and set $y := f(v_{i+1})$. Let $q_{i+1} := (s, v_{i+1})$ and output (y, q_{i+1}).

Thus, the encryption is

$$f_s(v_0 + 1), f_s(v_0 + 2), \ldots$$

3.2 CPA-secure scheme from pseudorandom functions

We cannot let $\mathcal{E}_k(m) = f_k(m)$ because f_k is deterministic, and thus the scheme could not be CPA-secure. However, we may encrypt by applying f_k to a random value r instead of the message, and taking $\mathcal{E}_k(m) = f_k(r) + m$. This in fact constitutes a pseudorandom pad with the plaintext, but the pad $f_k(r)$ is different each time (assuming the same r is not obtained repeatedly).

Construction. Let f a pseudorandom function. Define

- $\mathcal{G}$ as sampling $k \in \mathcal{B}_n$.
- $\mathcal{E}$ as computing $c := \langle r, f_k(r) + m \rangle$ on input k, m, with r uniformly sampled from $\mathcal{B}_n$.
- $\mathcal{D}$ as takingi nput $k, c = \langle r, s \rangle$ and outputting $m := f_k(r) + s$.

Theorem 11 *If F is a pseudorandom function, then the above construction is a CPA-secure private key encryption scheme for messages of length n.*