Cryptography

Encryption Schemes and Definitions of Security

Prof. Dr. Heiko Knospe

TH Köln – University of Applied Sciences

March 23, 2024

Encryption

We have already defined *encryption schemes*. They consists of algorithms, which produce keys and transform plaintext into ciphertext and conversely. The encryption algorithm is either *deterministic* or *probabilistic (randomized)*, and decryption is always deterministic. All algorithms (key generation, encryption, decryption) must be *polynomial* with respect to the input size.

However, successful attacks against schemes must not run in polynomial time. In the following we will give precise definitions of security.

Kerkhoff's Principle

The security of a cryptographic scheme should be solely based on a *secret key*, not on the details of the system.

Definition (Kerkhoff's Principle)

A cryptosystem should be secure under the assumption that an attacker knows the encryption and decryption algorithms.

Which are reasons for Kerkhoff's principle?

One-Time Pad

The *one-time pad* is an example of a simple but very powerful fixed-length symmetric encryption scheme. It uses the binary alphabet and the key length is equal to the message length. The security parameter n defines the length of plaintexts, ciphertexts and keys:

$$\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$$

The key generation algorithm $Gen(1^n)$ outputs a uniform random key $k \overset{\$}{\leftarrow} \{0,1\}^n$. A key k of length n is *used only for one message* $m \in \{0,1\}^n$. Encryption $\mathcal{E}_k$ and decryption $\mathcal{D}_k$ are identical and defined by a simple vectorial XOR operation:

$$c = \mathcal{E}_k(m) = m \oplus k \qquad m = \mathcal{D}_k(c) = c \oplus k$$

Definition of Perfect Secrecy

One might want a cryptosystem that is *unbreakable* for any type of adversary, even if they have unlimited computing power. To this end, Claude Shannon defined *perfect secrecy*.

Definition

An encryption scheme is called *perfectly secret* if for all plaintexts $m_0, \ m_1 \in \mathcal{M}$ and all ciphertexts $c \in \mathcal{C}$:

$$Pr[\mathcal{E}_k(m_0) = c\,] = Pr[\mathcal{E}_k(m_1) = c\,]$$

The probabilities are computed over randomly generated keys $k \in \mathcal{K}$.

Perfect secrecy means that all plaintexts have the same probability for a given ciphertext. This is also called *perfect indistinguishability*.

One-Time Pad

Perfect secrecy is a very strong condition. However, it is achieved by the one-time-pad.

Theorem

The one-time pad is perfectly secret if the key is a true random bit sequence (the bits are independent and uniformly distributed) and used only once.

In fact, given a one-time-pad ciphertext $c \in \{0,1\}^n$, any plaintext m is possible and equally likely:

$$Pr[\mathcal{E}_k(m) = c] = \frac{1}{2^n}.$$

Example

A Vigenère cipher of key length 3 is perfectly secret for messages of length 3, if the key is uniform random and only used once.

However, the same cipher is not perfectly secure for messages of length 4 (Exercise).

Computational Security

Perfect secrecy is too strong for most applications. Concrete security takes the *computing power* of an adversary and the probability of *successful attacks* into account.

Definition

A scheme is (t, ε)-secure if any adversary running in time t (measured in CPU cycles) can break the scheme with probability of ε at most.

We still need to give a definition of a successful attack (a break). This is done in the next section with well-defined experiments. For now, we consider a scheme to be *broken*, if an adversary can learn something about the plaintext from the ciphertext, i.e., if they obtain the output of any function of the plaintext, for example a plaintext bit or a sum of several plaintext bits.

Example of Computational Security

Assume the best-known attack against a scheme is exhaustive key search (brute force) and the key has length n. If testing a single key takes c CPU cycles and in total N CPU cycles are executed, then $\frac{N}{c}$ keys can be tested and the probability of success is approximately $\frac{N}{c2^n}$, if $\frac{N}{c} \ll 2^n$. Hence the scheme is $(N, \frac{N}{c2^n})$-secure.

Example: An adversary uses a computer with one 2 GHz CPU and performs a brute force attack against a scheme with 128-bit key length over the course of a year. Let's assume that $c = 1$. Then roughly 2^{55} keys can be tested and the scheme is $(2^{55}, 2^{-73})$-secure.

Note: an event with a probability of 2^{-73} will never occur in practice.

Asymptotic Definition

We give an asymptotic definition of computational security.

Definition

An encryption scheme is called *computational secure*, if every probabilistic algorithm with polynomial running time can only break the scheme with *negligible probability* in the security parameter n.

Example: Suppose the best possible attack is a brute-force search of a key of length n and the running time of attackers is bounded by $N = p(n)$, where p is a polynomial.
Then the scheme is $(p(n), \frac{p(n)}{c \cdot 2^n})$-secure, where c is a constant. The probability $\frac{p(n)}{c \cdot 2^n}$ is *negligible*. The scheme is computationally secure.

Security under Different Types of Attacks

We now define the security goal in well-defined experiments: suppose an adversary chooses two plaintexts and a challenger encrypts one of them. Can a polynomial-time adversary (who does not know the secret key) find the correct plaintext from the given ciphertext? *Indistinguishability* (IND) means that efficient adversaries are unable to find the correct plaintext out of two possibilities better than guessing at random.

We specify a *threat model* and consider attackers with certain capabilities:

- *Eavesdropping Attack* (EAV): the adversary gets the ciphertext.
- *Chosen Plaintext Attack* (CPA): the adversary can additionally choose plaintexts and gets their ciphertexts.
- *Chosen Ciphertext Attack* (CCA): the adversary can additionally choose ciphertexts and gets the associated plaintext.

EAV Experiment

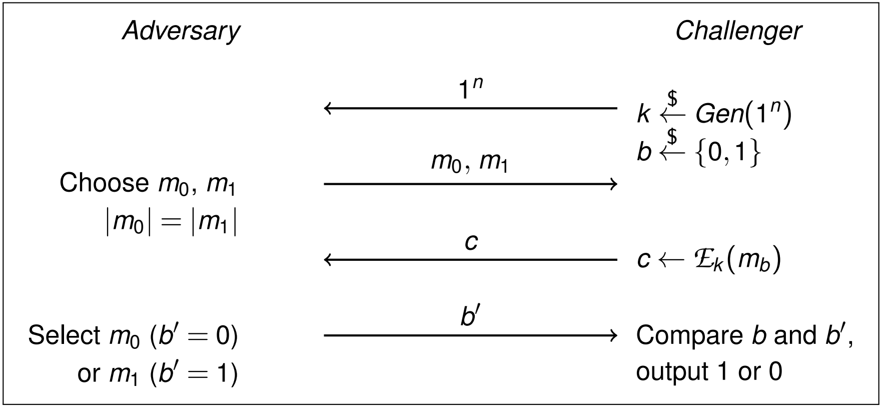
Definition (EAV Indistinguishability Experiment)

A challenger takes the security parameter 1^n as input, generates a key $k \in \mathcal{K}$ by running $Gen(1^n)$ and chooses a random bit $b \overset{\$}{\leftarrow} \{0,1\}$. A probabilistic polynomial-time adversary A is given 1^n, but neither k nor b are not known to A. The adversary chooses two plaintexts m_0 and m_1 that are equal in length. The challenger returns the ciphertext $\mathcal{E}_k(m_b)$ of one of them. A tries to guess b and outputs a bit b'. The challenger outputs 1 (success) if $b = b'$, and 0 (failure) otherwise. The *EAV advantage* of A is defined as

$$\mathrm{Adv}^{\text{ind-eav}}(A) = \mid Pr[b' = b] - Pr[b' \neq b] \mid .$$

The probability is taken over all random variables in this experiment, i.e., the key k, bit b, encryption $\mathcal{E}_k$ and randomness of A.

EAV Experiment



Indistinguishability experiment in the presence of an eavesdropper.

EAV Security

Definition

An encryption scheme has *indistinguishable encryptions in the presence of an eavesdropper* (IND-EAV secure or EAV-secure), if for every probabilistic polynomial-time adversary A, the advantage $\mathrm{Adv}^{\text{ind-eav}}(A)$ is negligible in the security parameter n.

EAV security is equivalent to *semantic security*: a polynomial-time adversary learns (almost) nothing from the ciphertext.

Example: Suppose a scheme does not encrypt the first bit. Then the scheme does not have EAV (or semantic) security: an adversary could choose two plaintexts which differ in their first bit. Then they can identify the correct plaintext by looking at the first ciphertext bit. So the EAV advantage is equal to 1.

CPA Experiment

The CPA (Chosen Plaintext Attack) experiment gives an adversary more power than in the EAV experiment: they can freely choose plaintexts and get the corresponding ciphertexts from the challenger. This might help to decrypt a challenge ciphertext, at least if there are only two plaintext candidates m_0 and m_1. They may even ask the challenger to encrypt m_0 and m_1.

Note: CPA security is stronger than EAV security and *deterministic* encryption schemes cannot be CPA-secure! CPA security is now a standard requirement for modern schemes.

CPA Experiment

Adversary		*Challenger*
	$\xleftarrow{1^n}$	$k \overset{\$}{\leftarrow} Gen(1^n)$
		$b \overset{\$}{\leftarrow} \{0,1\}$
Choose plaintext m'	$\xrightarrow{m'}$	
	$\xleftarrow{c'}$	$c' \leftarrow \mathcal{E}_k(m')$
Choose m_0, m_1	$\xrightarrow{m_0, m_1}$	
$\lvert m_0 \rvert = \lvert m_1 \rvert$	$\xleftarrow{c}$	$c \leftarrow \mathcal{E}_k(m_b)$
Choose plaintext m''	$\xrightarrow{m''}$	
	$\xleftarrow{c''}$	$c'' \leftarrow \mathcal{E}_k(m'')$
Select m_0 $(b' = 0)$ or m_1 $(b' = 1)$	$\xrightarrow{b'}$	Compare b and b', output 1 or 0

CPA Security

Definition

A scheme has *indistinguishable encryptions under chosen plaintext attack* (IND-CPA secure or CPA-secure), if for every probabilistic polynomial-time adversary A in the CPA experiment, the advantage

$$\mathrm{Adv}^{\text{ind-cpa}}(A) = \mid Pr[b' = b] - Pr[b' \neq b] \mid$$

is negligible in n.

CPA security is a key requirement for modern encryption schemes. However, it is hard to achieve! We will later see that secure block ciphers in certain operation modes can achieve this level of security.

CCA Security

The CCA2 (Chosen Ciphertext Attack) experiment is similar to the CPA experiment, but the adversary can also ask for the *decryption* of chosen ciphertexts (before and after receiving the challenge ciphertext, but except the challenge itself).

Definition

A scheme has *indistinguishable encryptions under chosen ciphertext attack* (IND-CCA2 secure or CCA2-secure), if for every probabilistic polynomial-time adversary A in the CCA2 experiment, the advantage

$$\mathrm{Adv}^{\text{ind-cca}}(A) = |\, Pr[b' = b] - Pr[b' \neq b] \,|$$

is negligible in n.

CCA2 security is significantly stronger than CPA security and can be achieved by combining a CPA-secure scheme and a secure MAC.

CCA2 Experiment

Adversary		*Challenger*
	$\xleftarrow{\quad 1^n \quad}$	$k \xleftarrow{\$} Gen(1^n)$
		$b \xleftarrow{\$} \{0,1\}$
Choose plaintext m'	$\xrightarrow{m' \text{ or } c'}$	
or ciphertext c'	$\xleftarrow{c' \text{ or } m'}$	$c' \leftarrow \mathcal{E}_k(m')$
		or $m' \leftarrow \mathcal{E}_k(c')$
Choose m_0, m_1	$\xrightarrow{m_0, m_1}$	
$\lvert m_0 \rvert = \lvert m_1 \rvert$	$\xleftarrow{\quad c \quad}$	$c \leftarrow \mathcal{E}_k(m_b)$
Choose plaintext m''	$\xrightarrow{m'' \text{ or } c''}$	$c'' \leftarrow \mathcal{E}_k(m'')$
or ciphertext $c'' \neq c$	$\xleftarrow{c'' \text{ or } m''}$	$m'' \leftarrow \mathcal{D}_k(c'')$
Select m_0 $(b' = 0)$		Compare b and b',
or m_1 $(b' = 1)$	$\xrightarrow{\quad b' \quad}$	output 1 or 0

Constructing Secure Encryption Schemes

For the construction of secure encryption schemes, we use cryptographic *primitives* as building blocks.

- Pseudorandom generators (*prg*) take a seed (or key) as input and output a sequence of randomly looking bits. They are building block for stream ciphers, which use a pseudorandom keystream for encryption.
- Pseudorandom functions or permutations (*prf*, *prp*) are function families which are parametrized by a key. They transform input blocks into output blocks such that the mapping looks randomly. Block ciphers can be combined with an operation mode and define encryption schemes.

We will see that there is often a *reduction* from the pseudorandomness of a primitive to the (EAV, CPA or CCA)-security of an encryption scheme.

Pseudorandom Generators

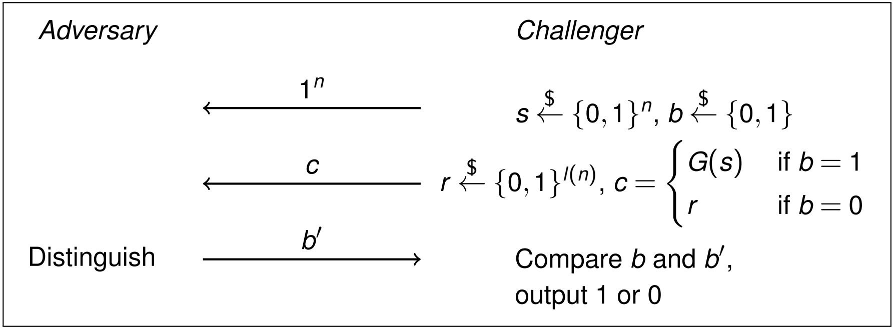
Randomness and pseudorandomness play a fundamental role in cryptography, especially for encryption, but also for other cryptographic operations. A pseudorandom string *looks like* a uniform binary string without being the output of a true random bit generator.

A *bit generator G* is a deterministic polynomial-time algorithm that takes an input seed $s \in \{0,1\}^n$ and outputs a string $G(s) \in \{0,1\}^{l(n)}$, where $l(.)$ is a polynomial and $l(n) > n$ for all $n \in \mathbb{N}$.

G is called a *pseudorandom generator* (prg), if the output cannot be distinguished from a uniform random sequence in polynomial time.

Pseudorandom Generator (prg) Experiment

The adversary has to distinguish between a true random sequence r and the output of the generator G.


Adversary
Challenger
1^n
$s \overset{\$}{\leftarrow} \{0,1\}^n,\ b \overset{\$}{\leftarrow} \{0,1\}$
c
$r \overset{\$}{\leftarrow} \{0,1\}^{l(n)},\ c = \begin{cases} G(s) & \text{if } b = 1 \\ r & \text{if } b = 0 \end{cases}$
Distinguish
b'
Compare b and b', output 1 or 0

Distinguishability experiment for a bit generator G.

Secure Pseudorandom Generators

Definition

A generator G is called a *pseudorandom generator*, if for every probabilistic polynomial-time adversary A in the prg experiment, the prg-advantage $\mathrm{Adv}^{\mathrm{prg}}(A) = \mid Pr[b' = b] - Pr[b' \neq b] \mid$ is negligible in the security parameter n.

The construction of a pseudorandom generator (prg) is a non-trivial task! The output of a *prg* can be used as *keystream*. Define a fixed-length encryption scheme by

$$c = \mathcal{E}_k(m) = m \oplus G(k) \quad \text{and} \quad m = \mathcal{D}_k(c) = c \oplus G(k).$$

Theorem

If G is a pseudorandom generator, then the associated encryption scheme is EAV-secure.

Pseudorandom Functions

Many cryptographic schemes are based on *pseudorandom functions*. Consider a family of functions

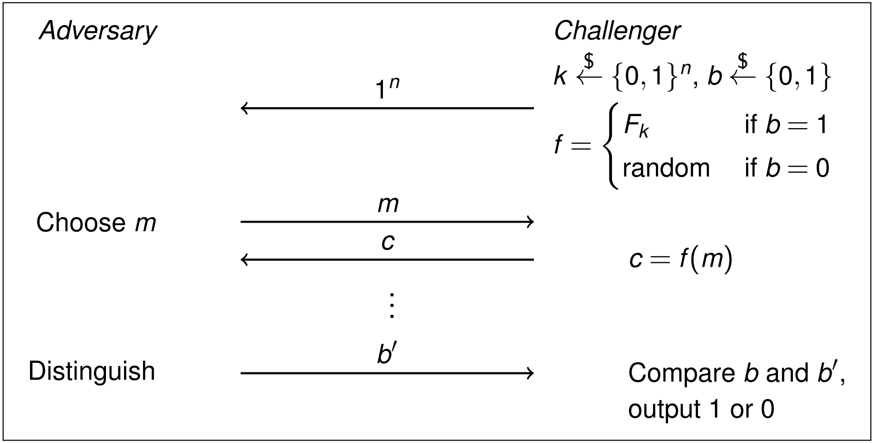
$$F: \ K_n \times D_n \to R_n,$$

where n is a security parameter and $K_n = \{0,1\}^n$ the set of keys. To simplify the notation, we assume that $D_n = R_n = \{0,1\}^l$ where $l = l(n)$ is polynomial in n. We suppose that F can be computed in *polynomial time*. Fixing $k \in K_n$ yields a function

$$F_k : D_n \to R_n.$$

F is called *pseudorandom* if the functions $F_k : D_n \to R_n$ *appear to be random* for a polynomial-time adversary who knows the input-output behavior of F_k, but is not given the secret key k. In other words, a polynomial-time adversary is not able to distinguish a pseudorandom function from a true random function.

Pseudorandom Function (prf) Experiment


Adversary
Challenger
$k \xleftarrow{\$} \{0,1\}^n$, $b \xleftarrow{\$} \{0,1\}$
1^n
$f = \begin{cases} F_k & \text{if } b = 1 \\ \text{random} & \text{if } b = 0 \end{cases}$
Choose m
m
c
$c = f(m)$
$\vdots$
Distinguish
b'
Compare b and b', output 1 or 0

Distinguishability experiment for a function family F.

Secure Pseudorandom Functions

Definition

A keyed function family F as described above is called a *pseudorandom function* (prf) if for every probabilistic polynomial time adversary A in the prf-experiment, the prf-advantage $\mathrm{Adv}^{\mathrm{prf}}(A) = \mid Pr[b' = b] - Pr[b' \neq b] \mid$ is negligible in n.

A pseudorandom generator G can be derived from a pseudorandom function F by the following construction: choose a uniform random counter *ctr* of length l and set

$$G(k) = F_k(ctr + 1) \parallel F_k(ctr + 2) \parallel F_k(ctr + 3) \parallel \ldots,$$

where *ctr* is viewed as an integer and addition is done modulo 2^l.

Pseudorandom Permutations

Pseudorandom permutations are an important special case of pseudorandom functions. They are given by a keyed family of *permutations* (bijections):

$$F : K_n \times D_n \to D_n$$

where $K_n = \{0,1\}^n$ and $D_n = \{0,1\}^l$. For any $k \in K_n$, the function $F_k : D_n \to D_n$ is a permutation, i.e., F_k is bijective.

A family F of permutations is called *pseudorandom*, if polynomial-time adversaries are not able to distinguish F from a true random permutation.

Modern block ciphers (for example AES) are modeled to be pseudorandom permutations (prp). We will see that they are a main building block in the construction of secure encryption schemes.

Block Cipher

Definition

A family of permutations

$$E = E(n): \ \{0,1\}^n \times \{0,1\}^l \to \{0,1\}^l$$

is said to be a *block cipher*. n is the security parameter, $\{0,1\}^n$ is the key space, n is the *key length* and $l = l(n)$ is the *block length*.

Block ciphers can be thought of as concrete instances of families of *pseudorandom permutations*. Each key $k \in \{0,1\}^n$ yields a bijective function

$$E_k: \ \{0,1\}^l \to \{0,1\}^l.$$

Operation Modes

Block ciphers can only encrypt messages of block length l. Therefore, an *operation mode* is needed to encrypt plaintexts of arbitrary length. The combination of a block cipher and an operation mode defines an encryption scheme for plaintexts (and ciphertexts) in $\{0,1\}^*$.

Here, we discuss the following operation modes:

- ECB (Electronic Codebook Mode)
- CBC (Cipher Block Chaining Mode)
- CTR (Counter Mode)

There are more operation modes, e.g., OFB, CFB and GCM. We will deal with GCM later.

ECB Mode

Definition (ECB Mode)

Let E be a block cipher. We define an encryption scheme based on E. The key generation algorithm $Gen(1^n)$ outputs a uniform random key $k \overset{\$}{\leftarrow} \{0,1\}^n$ of length n. For encryption, set $m = m_1 \| m_2 \| \dots \| m_N$, where each m_i is a block of length l and the last message is padded by $10\dots0$. Define

$$c_i = E_k(m_i) \text{ for } i = 1, 2, \dots, N \text{ and } c = \mathcal{E}_k(m) = c_1 \| c_2 \| \dots \| c_N.$$

Decryption works in a similar way:

$$m_i = E_k^{-1}(c_i) \text{ for } i = 1, 2, \dots, N \text{ and } m = \mathcal{D}_k(c) = m_1 \| m_2 \| \dots \| m_N.$$

Finally, the paddings bits in the last block are removed.

CBC Mode

Definition (Randomized CBC Mode)

The key generation and splitting of messages into blocks is the same as for the ECB mode. The CBC mode requires an initialization vector $IV \xleftarrow{\$} \{0,1\}^l$, which is chosen uniformly at random for each message. Then encrypt a message m by computing

$$c_0 = IV,\ c_i = E_k(m_i \oplus c_{i-1}) \text{ for } i = 1, 2, \dots, N. \text{ Set}$$
$$c = \mathcal{E}_k(m) = c_0 \| c_1 \| c_2 \| \dots \| c_N.$$

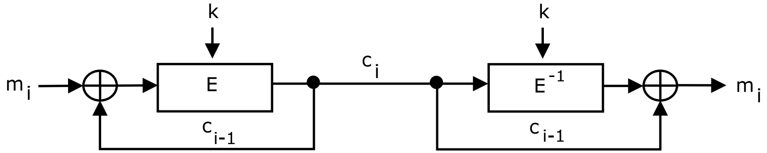
Decryption of a ciphertext c is defined by

$$m_i = E_k^{-1}(c_i) \oplus c_{i-1} \text{ for } i = 1, 2, \dots, N. \text{ Then}$$
$$m = \mathcal{D}_k(c) = m_1 \| m_2 \| \dots \| m_N.$$

CBC Mode

We can easily verify that the CBC mode has correct decryption:

$$E_k^{-1}(c_i) \oplus c_{i-1} = E_k^{-1}(E_k(m_i \oplus c_{i-1})) \oplus c_{i-1} = m_i \oplus c_{i-1} \oplus c_{i-1} = m_i$$



Encryption and decryption in CBC mode.

CTR Mode

Definition (Randomized CTR Mode)

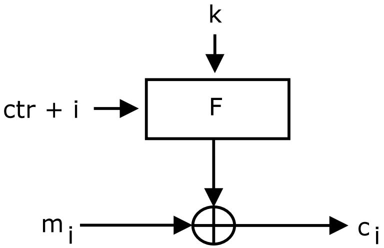
Let F be family of functions or permutations. We define an encryption scheme based on F. The key generation algorithm $Gen(1^n)$ outputs a uniform random key $k \overset{\$}{\leftarrow} \{0,1\}^n$ of length n. A plaintext message m is split into blocks of length l, i.e., $m = m_1 \| m_2 \| \dots \| m_N$; the last block can be shorter. A uniform random counter $ctr \overset{\$}{\leftarrow} \{0,1\}^l$ is chosen and incremented for each block. The encrypted counter gives keystream:

$$c_i = F_k(ctr + i) \oplus m_i \text{ for } i = 1, 2, \dots, N \text{ and}$$
$$c = \mathcal{E}_k(m) = ctr \| c_1 \| c_2 \| \dots \| c_N.$$

Decryption uses the same keystream:

$$m_i = F_k(ctr + i) \oplus c_i \text{ for } i = 1, 2, \dots, N \text{ and}$$
$$m = \mathcal{D}_k(c) = m_1 \| m_2 \| \dots \| m_N.$$

CTR Mode



Encryption in CTR mode. The keystream is XORed with the plaintext. Decryption is almost identical, with plaintext and ciphertext swapped.

Security of the CTR Mode

The CTR mode is secure since there is polynomial-time reduction from the pseudorandomness of F to the CPA security of the associated CTR mode encryption.

Theorem

If F is a pseudorandom function (prf or prp), then the randomized CTR mode has indistinguishable encryptions under a chosen-plaintext attack, i.e., the encryption scheme is IND-CPA secure.

Proof of Security

We have to show that any adversary A in the CPA experiment achieves only a negligible advantage if a pseudorandom function F is used. The proof has two steps and uses a a modified encryption scheme, where a truly random function f in place of a pseudorandom function F is used.

- One shows that the difference between the probabilities of the output of A in the original scheme (using F) and the modified scheme (using f) is negligible. To this end, use oracle access to A to construct an algorithm that distinguishes between the pseudorandom function F and the random function f. However, the advantage of this algorithm must be negligible since by assumption F is a *prf*.

Proof of Security

- The previous step shows that we can consider a random function f that generates the cipher stream. Then CTR mode encryption basically behaves as a *one-time-pad* where $\mathrm{Adv}^{\text{ind-cpa}}(A)$ is zero. However, this is only true if the input for f (i.e. the counter) is fresh and has not been used before. In the CPA experiment, A can ask for the encryption of chosen plaintexts. Each time a uniform counter value is generated and used as input for f. If the counter values of A's queries overlap with a counter used by the challenge ciphertext, then A can decrypt a block of the challenge ciphertext and win the experiment. Now a careful analysis shows that the probability of an overlap and hence $\mathrm{Adv}^{\text{ind-cpa}}(A)$ is negligible.

Security of the CBC Mode

The CBC mode provides a similar level of security as the CTR mode.

Theorem

If E is a pseudorandom permutation (prp), then the randomized CBC mode is IND-CPA secure.

The CPA security guarantee can easily be misunderstood.

- CBC and CTR mode security is not unconditional, but rather depends on the underlying *prp* or *prf*.
- Only certain type of attacks are considered, since we made assumptions about the setup and the adversary (uniform random secret keys, a polynomial-time adversary who is only looking at plaintexts and ciphertexts, e.g., no side-channel attacks).
- The security guarantee is only asymptotic, i.e., for large n.

CCA Security and Malleability

It is easy to see that neither CBC nor CTR mode yield a CCA2-secure encryption scheme. Security against chosen ciphertext attacks requires *non-malleability*. An encryption scheme is called *malleable* if it is possible to transform a ciphertext into another ciphertext which decrypts to a related plaintext.

Example: Consider the CTR mode. If a single bit of the ciphertext is flipped, then only the corresponding plaintext bit changes. Hence the CTR mode is clearly malleable.

CCA2 security can be achieved by adding a MAC (*Message Authentication Code*) or by other transformations. This is, for example, implemented by the Galois Counter Mode (GCM).