Cryptography

Corso di Laurea Magistrale in Informatica

Private-Key Encryption and Pseudorandomness

Ugo Dal Lago

Academic Year 2023-2024

Computational Cryptography

- We know that perfect secrecy, though satisfactory in terms of the offered guarantees, has strong restrictions.
- In computational cryptography, which emerged in the late 70s, perfect secrecy is *weakened*, working toward a definition that offers **milder guarantees**.
- This weakening unfolds in two directions:
  1. The impossibility of breaking the scheme is guaranteed only for **efficient adversaries**.
  2. In addition, the adversary *can* in principle force the underlying scheme, but its probability of success is **very small**.

Two Side Effects

1. **Need to Work with Strings**.
   - How to formalize the notion of *efficient* adversary?
   - Computational complexity provides the right tools, but it requires that we can at least measure *the length of the input*.
   - For this reason, we assume that keys, messages and ciphertexts are strings, without loss of generality.

Two Side Effects

1. **Need to Work with Strings**.
   - How to formalize the notion of *efficient* adversary?
   - Computational complexity provides the right tools, but it requires that we can at least measure *the length of the input*.
   - For this reason, we assume that keys, messages and ciphertexts are strings, without loss of generality.
2. **Impossibility of Unconditional Proofs of Security**.
   - With the exception of very few cases, proofs of security for primitives and protocols in computational cryptography are based on assumptions.
   - In most cases, proofs of security will be proofs by reduction.

Two Approaches

1. **The Concrete Approach**
   - In the concrete approach, an encryption scheme is defined (t, ε)-secure, if every adversary running for *time at most* t succeeds in breaking the scheme with *probability at most* ε.
   - The main problem with the concrete approach is that the boundary t, whatever the unit of measure in which it is expressed, makes the security evaluation dependent on the underlying HW.
2. **The Asymptotic Approach**
   - In the asymptotic approach, security evaluations depend on on a global parameter n, called *security parameter*.
   - The notion of *efficiency* is defined by means of **PPT-algorithms**, while that of *low probability of success* is captured by the concept of **negligible function**.
   - A scheme is secure if every PPT adversary succeeds in breaking the scheme with only negligible probability.
   - In this way, we realize independency from the model.

The Asymptotic Approach - PPT Algorithms

- We have already mentioned that adversaries and schemes, in modern cryptography, are probabilistic algorithms, that is, they can perform probabilistic choices.

Definition

A probabilistic algorithm A is said to be **PPT** (probabilistic polynomial time) if there is a polynomial p that defines an upper bound on the computation time of A independently on the outcomes of the performed probabilistic choices.

- An alternative would be to say that p bounds *the average* execution time of A. However, cryptography did **not** follow this path.
- This definition of algorithm is the same as the one on which the complexity class BPP is based.

The Asymptotic Approach - Negligible Functions

Definition

A function $f : \mathbb{N} \to \mathbb{R}$ is said to be *negligible* iff for every polynomial $p : \mathbb{N} \to \mathbb{N}$ there exists $N \in \mathbb{N}$ such that for every $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.

- **Examples**: $n \mapsto 2^{-n}$, $n \mapsto 2^{-\sqrt{n}}$, $n \mapsto n^{-\log n}$.

Lemma

The set of all negligible functions, called $\mathcal{NGL}$, is closed for sum, product, and product by an arbitrary polynomial.

- The closure properties of $\mathcal{NGL}$, the last one in particular, allow us to make "robust" our security definitions.

Limited Resources and Possible Attacks - Why?

- Let us assume that $|\mathcal{K}| \ll |\mathcal{M}|$, i.e., that the set of keys is *much smaller* than the set of messages.

Limited Resources and Possible Attacks - Why?

- Let us assume that $|\mathcal{K}| \ll |\mathcal{M}|$, i.e., that the set of keys is *much smaller* than the set of messages.
- We could construct the following attacks:
  1. In a ciphertext-only context, we could, given a ciphertext c, decrypt c **with all possible keys**. The set of messages potentially corresponding to c would be greatly reduced.
  2. We could also, in a known-plaintext context in which we observe $(m_1, c_1), \ldots, (m_\ell, c_\ell)$, construct a key k **completely randomly**, and then check if $Dec_k(c_i) = m_i$ for every i. The probability of success is nonzero, although very low.
- It is therefore necessary to bound the computational power of the adversary, allowing at the same time that the latter can break the cipher, but with very low probability.

The Role of Assumptions

- Proving the security of an encryption scheme in this new sense would require demonstrating the **impossibility** of constructing adversaries that satisfy certain criteria:

$$\forall A \in PPT . \neg \mathbf{BRK}(\Pi, A) \iff \neg \exists A \in PPT . \mathbf{BRK}(\Pi, A)$$

The Role of Assumptions

- Proving the security of an encryption scheme in this new sense would require demonstrating the **impossibility** of constructing adversaries that satisfy certain criteria:

$$\forall A \in PPT.\neg\mathbf{BRK}(\Pi, A) \iff \neg\exists A \in PPT.\mathbf{BRK}(\Pi, A)$$

- Theoretical computer science, and computational complexity, **are not currently able** to prove results like the previous ones, with the exception of very simple cases.
- The best we can do is nothing more than conditioning security to appropriate assumptions that have a similar form, namely to show that

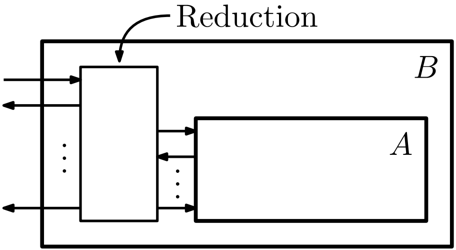
$$\forall B \in PPT.\neg\mathbf{BRK}(\Xi, B) \Longrightarrow \forall A \in PPT.\neg\mathbf{BRK}(\Pi, A)$$

Proofs by Reduction

- We observe that

$$\forall B \in PPT.\neg\mathbf{BRK}(\Xi, B) \Longrightarrow \forall A \in PPT.\neg\mathbf{BRK}(\Pi, A)$$
$$\Updownarrow$$
$$\exists A \in PPT.\mathbf{BRK}(\Pi, A) \Longrightarrow \exists B \in PPT.\mathbf{BRK}(\Xi, B)$$

- We will then proceed by constructing an adversary for Ξ starting from from an adversary for Π:



A New Definition of Encryption Scheme

- It is necessary to slightly modify the definition of the encryption scheme $\Pi = (Gen, Enc, Dec)$ by requiring that the three algorithms involved are PPT and that furthermore:
  - Gen takes as input a string of the form 1^n, to be interpreted as the security parameter, and outputs a key k, such that $|k| \geq n$.
  - The Enc algorithm can be genuinely probabilistic, that is, produce different ciphertexts starting from the same couple (m, k).
  - The algorithm Dec is deterministic (otherwise there is no chance of being correct).
- We assume, obviously, that the scheme is at least correct, i.e. that $Dec_k(Enc_k(m)) = m$.
- Often, Enc_k is defined only for messages of length equal to $\ell(n)$, where n is the security parameter that generated k. In this case, we say that encryption scheme is a **fixed-length** encryption scheme with length parameter ℓ.

Adapting the Experiment PrivK^{eav}

- There are basically two ways to define the security of an encryption scheme in the sense of computational cryptography:
  - The first one, called **semantic security**, formally traces the definition of perfect secrecy given by Shannon.
  - The second is based instead on the **indistinguishability** and is the one we will use in this course.
- It is necessary to adapt the notion of experiment that we saw talking about perfect secrecy:

$\mathsf{PrivK}^{eav}_{A,\Pi}(n)$:
$(m_0, m_1) \leftarrow A(1^n)$;
if $|m_0| \neq |m_1|$ **then**
  Result: 0
$k \leftarrow Gen(1^n)$; $b \leftarrow \{0,1\}$;
$c \leftarrow Enc(k, m_b)$;
$b^* \leftarrow A(c)$;
Result: $\neg(b \oplus b^*)$

The Definition

Definition

An encryption scheme Π is said to be *secure against passive attacks* or *secure with respect to* PrivK^{eav} iff for every PPT adversary A there exists a function $\varepsilon \in \mathcal{NGL}$ such that

$$Pr(\mathsf{PrivK}^{eav}_{\Pi,A}(n) = 1) = \frac{1}{2} + \varepsilon(n)$$

The Definition

Definition

An encryption scheme Π is said to be *secure against passive attacks* or *secure with respect to* PrivK^{eav} iff for every PPT adversary A there exists a function $\varepsilon \in \mathcal{NGL}$ such that

$$Pr(\mathsf{PrivK}^{eav}_{\Pi,A}(n) = 1) = \frac{1}{2} + \varepsilon(n)$$

- It can be observed that we indeed quantify over *all* adversaries, but restricting our attention to those with bounded complexity.
- Note that the adversary can win with probability strictly greater than $\frac{1}{2}$, but that his *advantage* must be negligible.

How to Construct a Secure Scheme?

From One-Time Pad...

$$\begin{tikzcd} & m \arrow[d] & \\ k \arrow[r] & \oplus \arrow[r] & c \end{tikzcd}$$

How to Construct a Secure Scheme?

From One-Time Pad…

$$\begin{tikzcd} & m \arrow[d] & \\ k \arrow[r] & \oplus \arrow[r] & c \end{tikzcd}$$

…to a version with shorter keys.

$$\begin{tikzcd} & & m \arrow[d] & \\ k \arrow[r] & G(k) \arrow[r] & \oplus \arrow[r] & c \end{tikzcd}$$

Towards Pseudorandomness

- What is reasonable to require of the algorithm G that we implicitly mentioned?

Towards Pseudorandomness

- What is reasonable to require of the algorithm G that we implicitly mentioned?
  - Certainly, it is necessary that G be an **deterministic** algorithm, i.e., that it doesn't use any form of random choice.

Towards Pseudorandomness

- What is reasonable to require of the algorithm G that we implicitly mentioned?
  - Certainly, it is necessary that G be an **deterministic** algorithm, i.e., that it doesn't use any form of random choice.
  - Next, we need to ensure that G is **efficiently computable**, otherwise the whole scheme we are building would become problematic from a computational point of view.
  - Then we would like G to allow us to **expand the input string**, making it longer and longer. If this were not the case, there would be no point to build a new scheme.

Towards Pseudorandomness

- What is reasonable to require of the algorithm G that we implicitly mentioned?
  - Certainly, it is necessary that G be an **deterministic** algorithm, i.e., that it doesn't use any form of random choice.
  - Next, we need to ensure that G is **efficiently computable**, otherwise the whole scheme we are building would become problematic from a computational point of view.
  - Then we would like G to allow us to **expand the input string**, making it longer and longer. If this were not the case, there would be no point to build a new scheme.
  - **Is that all**? No! We need to ensure that the string $G(k)$ has *all the properties* that we expect from a key.
    - What would happen, for example, if G did nothing more than return the concatenation of k with itself, that is, if $G(k) = k \cdot k$?
    - What would happen, instead, if the first and last bits of $G(k)$ were always the same bit?

Pseudorandom Generators

- Informally, we want $G(k)$ to be *pseudorandom*: even though it is not random, it should be indistinguishable from a random string for an *efficient* adversary.

Definition (Pseudorandom Generator)

Let $\ell : \mathbb{N} \to \mathbb{N}$ be a polynomial, called *expansion factor*, and let G be an deterministic algorithm that, for every input $s \in \{0,1\}^*$ outputs a string $G(s) \in \{0,1\}^{\ell(|s|)}$. We say that G is a *pseudorandom generator* (PRG) iff:

- For every $n \in \mathbb{N}$ it holds that $\ell(n) > n$
- G is polytime.
- For every PPT algorithm D there exists $\varepsilon \in \mathcal{NGL}$ such that

$$|Pr(D(s) = 1) - Pr(D(G(r)) = 1)| \leq \varepsilon(n)$$

where s, r are random of length $\ell(n)$ and n, respectively.

On Pseudorandom Generators

- The output of pseudorandom generators **is not random** in a strict sense
  - If, for example, $\ell(n) = 2n$, there are only 2^n strings (of length $2n$) that G outputs, while the strings of length $2n$ are 2^{2n}. We observe that
  $$\frac{2^n}{2^{2n}} = \frac{1}{2^n}$$
  - There is therefore always a distinguisher D that wins against G with very high probability, but this distinguisher has exponential complexity.
- From a more concrete point of view, it must be ensured that brute-force attacks, in the form of a distinguisher, are not possible, i.e. that n is sufficiently large.
- Unfortunately, pseudorandom generators *cannot* be proved to exist unless under appropriate assumptions.

The First Secure Encryption Scheme

Definition (PRG-Induced Scheme)

Given a PRG G with expansion factor ℓ, the scheme $\Pi^G = (Gen, Enc, Dec)$ is defined as follows:

- The algorithm Gen on input 1^n outputs each string of length n with same probability, i.e. $\frac{1}{2^n}$.
- $Enc(m, k) = G(k) \oplus m$
- $Dec(c, k) = G(k) \oplus c$.

- We observe how the scheme Π^G is for messages of fixed length equal to ℓ.
- Correctness is easy to prove: $Dec(Enc(m, k), k) = G(k) \oplus (G(k) \oplus m) = m$.

Theorem

If G is a pseudorandom generator, then Π^G is secure against passive attacks

Handling Variable-Length Messages

- The scheme Π^G allows to handle messages of fixed length, equal to the expansion factor of G.
- We would like, instead, to be able to handle variable-length messages. To do this, we need to generalize the notion of pseudorandom generator to the case of variable-length messages.

Handling Variable-Length Messages

- The scheme Π^G allows to handle messages of fixed length, equal to the expansion factor of G.
- We would like, instead, to be able to handle variable-length messages. To do this, we need to generalize the notion of pseudorandom generator to the case of variable-length messages.

Definition (Variable Output-Length Generators)

A deterministic polytime algorithm G is said to be *Variable Output-Length Pseudorandom Generators* if from a seed $s \in \{0,1\}^n$ and a string in the form 1^ℓ outputs a binary string $G(s, 1^\ell)$ such that

- If $\ell < \ell'$, the string $G(s, 1^\ell)$ is a prefix of $G(s, 1^{\ell'})$.
- For any polynomial $p : \mathbb{N} \to \mathbb{N}$, the algorithm G_p defined by setting $G_p(s) = G(s, 1^{p(|s|)})$ is a random (of fixed-length p), generator.

Handling Variable-Length Messages

- Given a variable output-length pseudorandom generator G, it is easy to generalize the construction Π^G:

$$Enc(m, k) = G(k, 1^{|m|}) \oplus m \qquad Dec(c, k) = G(k, 1^{|c|}) \oplus c$$

- The security properties of this new construction remain unchanged.

Handling Variable-Length Messages

- Given a variable output-length pseudorandom generator G, it is easy to generalize the construction Π^G:

$$Enc(m, k) = G(k, 1^{|m|}) \oplus m \qquad Dec(c, k) = G(k, 1^{|c|}) \oplus c$$

- The security properties of this new construction remain unchanged.

Lemma

For every pseudorandom generator G it is possible to construct from G a variable output-length pseudorandom generator H.

Handling Variable-Length Messages

- Given a variable output-length pseudorandom generator G, it is easy to generalize the construction Π^G:

$$Enc(m,k) = G(k, 1^{|m|}) \oplus m \qquad Dec(c,k) = G(k, 1^{|c|}) \oplus c$$

- The security properties of this new construction remain unchanged.

Lemma

For every pseudorandom generator G it is possible to construct from G a variable output-length pseudorandom generator H.

- In practice, so-called stream ciphers (such as, for example, RC4) are designed to satisfy the axioms of a variable output-length pseudorandom generator.
  - **It cannot be proved** that they satisfy these axioms.
  - **They are not** encryption schemes.

Multiple Encryptions

- Until now we have always assumed that the adversary sees only one ciphertext. Is this restrictive?

Multiple Encryptions

- Until now we have always assumed that the adversary sees only one ciphertext. Is this restrictive?
- The answer is unluckily yes.

Multiple Encryptions

- Until now we have always assumed that the adversary sees only one ciphertext. Is this restrictive?
- The answer is unluckily yes.
- To understand why this is the case, we need to **modify the experiment** PrivK^{eav} into an experiment that takes into account the possibility for the adversary to observe multiple encryptions and that we will call PrivK^{mult}.

Multiple Encryptions

- Until now we have always assumed that the adversary sees only one ciphertext. Is this restrictive?
- The answer is unluckily yes.
- To understand why this is the case, we need to **modify the experiment** PrivK^{eav} into an experiment that takes into account the possibility for the adversary to observe multiple encryptions and that we will call PrivK^{mult}.
  - PrivK^{mult} is quite similar to PrivK^{eav}.
  - The adversary, however, is allowed to produce two vectors of messages $\mathbf{m}_0 = (m_0^1, \ldots, m_0^t)$ and $\mathbf{m}_1 = (m_1^1, \ldots, m_1^t)$.
  - The so-called *challenge ciphertext* will become a vector $\mathbf{c} = (c^1, \ldots, c^t)$
  - As usual, we will require that for every PPT A there exists negligible ε with

$$Pr(\mathsf{PrivK}^{mult}_{\Pi,A}(n) = 1) = \frac{1}{2} + \varepsilon(n)$$

    obtaining the definition of **security with respect to to** PrivK^{mult}.

Multiple Encryptions

Lemma

The scheme Π^G *is not secure with respect to* PrivK^{mult}, *not even if* G *is pseudorandom.*

Multiple Encryptions

Lemma

The scheme Π^G is not secure with respect to PrivK^{mult}, not even if G is pseudorandom.

- Actually, we can go further and prove the following result:

Theorem

If Enc is deterministic, then the scheme $\Pi = (Gen, Enc, Dec)$ cannot be secure with respect to PrivK^{mult}.

- Does this mean that stream ciphers are *useless* in the context of multiple encryptions?

Multiple Encryptions

Lemma

The scheme Π^G is not secure with respect to PrivK^{mult}, not even if G is pseudorandom.

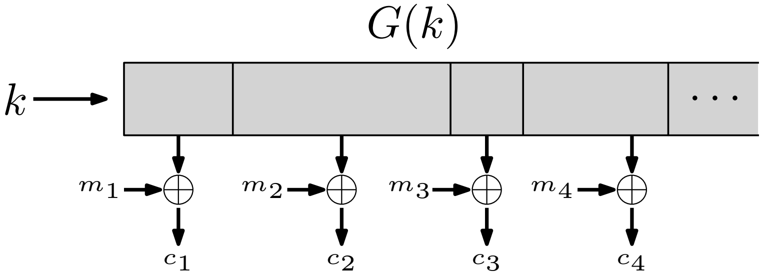
- Actually, we can go further and prove the following result:

Theorem

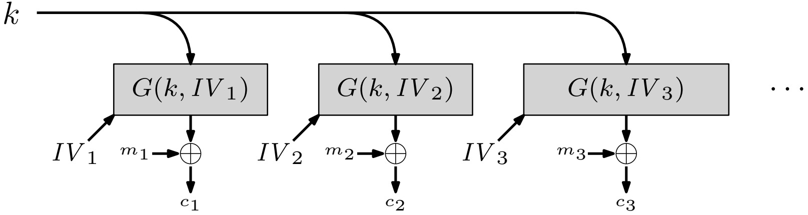
If Enc is deterministic, then the scheme $\Pi = (Gen, Enc, Dec)$ cannot be secure with respect to PrivK^{mult}.

- Does this mean that stream ciphers are *useless* in the context of multiple encryptions?
- No, but the penalty is the need of encryption schemes in which Enc has an *internal state*.

Synchronized Mode



Unsynchronized Mode



Security Against CPA Attacks

- So far, our adversary model has assumed that the adversary could interact with the experiment in a restricted way.
  - Sure, he could build m_0 and m_1, but having received c, he could do nothing more.

Security Against CPA Attacks

- So far, our adversary model has assumed that the adversary could interact with the experiment in a restricted way.
  - Sure, he could build m_0 and m_1, but having received c, he could do nothing more.
- In the CPA context, we will instead make the assumption that the adversary has access to an *oracle* for $Enc_k(\cdot)$.
  - At any time, the adversary can **invoke the oracle** on a message m of his choice, obtaining the corresponding ciphertext $c = Enc_k(m)$.
  - Otherwise, the experiment is essentially the same and is called PrivK^{CPA}.

Security Against CPA Attacks

- So far, our adversary model has assumed that the adversary could interact with the experiment in a restricted way.
  - Sure, he could build m_0 and m_1, but having received c, he could do nothing more.
- In the CPA context, we will instead make the assumption that the adversary has access to an *oracle* for $Enc_k(\cdot)$.
  - At any time, the adversary can **invoke the oracle** on a message m of his choice, obtaining the corresponding ciphertext $c = Enc_k(m)$.
  - Otherwise, the experiment is essentially the same and is called PrivK^{CPA}.

Definition

An encryption scheme Π is said to be *secure against CPA attacks* (or *CPA-secure*) iff for every adversary A there exists a negligible function ε such that

$$Pr(\mathsf{PrivK}_{A,\Pi}^{CPA}(n) = 1) \leq \frac{1}{2} + \varepsilon(n).$$

Necessary Conditions for CPA Security

Lemma

Any scheme Π that is secure with respect to PrivK^{CPA} is secure with respect to PrivK^{eav}.

Necessary Conditions for CPA Security

Lemma

Any scheme Π *that is secure with respect to* PrivK^{CPA} *is secure with respect to* PrivK^{eav}.

Lemma

Any scheme $\Pi = (Gen, Enc, Dec)$ *that is secure with respect to* PrivK^{CPA} *must be such that* Enc *is probabilistic.*

Necessary Conditions for CPA Security

Lemma

Any scheme Π *that is secure with respect to* PrivK^{CPA} *is secure with respect to* PrivK^{eav}.

Lemma

Any scheme $\Pi = (Gen, Enc, Dec)$ *that is secure with respect to* PrivK^{CPA} *must be such that* Enc *is probabilistic.*

- Generalizing our notion of security against CPA attacks to the case of multiple encodings is a very simple exercise.

Theorem

Every encryption scheme that is CPA-secure is secure even in case of multiple encodings.

Constructing a CPA-Secure Cipher

- But do **exist** CPA-Secure encryption schemes?
- Again, as in the case of security against passive attacks, the theory of pseudorandomness comes to our aid.
- It is, however, clear that pseudorandom generators are not very useful in this context, at least in the way we have used them so far.
  - They are deterministic, let us remember that.
  - As a result, Π^G has no hope of being CPA-secure.

Constructing a CPA-Secure Cipher

- But do **exist** CPA-Secure encryption schemes?
- Again, as in the case of security against passive attacks, the theory of pseudorandomness comes to our aid.
- It is, however, clear that pseudorandom generators are not very useful in this context, at least in the way we have used them so far.
  - They are deterministic, let us remember that.
  - As a result, Π^G has no hope of being CPA-secure.
- We must consider the notion of **pseudorandom function**, for which, however, we need some preliminary notions:
  - We will work with *binary partial functions*, i.e. partial functions from $\{0,1\}^* \times \{0,1\}^*$ to $\{0,1\}^*$.
  - A binary partial function F is *length-preserving* iff $F(k,x)$ is defined iff $|k| = |x|$ and in that case $|F(k,x)| = |x|$.
  - Given a length-preserving binary partial function F, we denote by F_k the function from $\{0,1\}^{|k|}$ to $\{0,1\}^{|k|}$ defined in the natural way.

Pseudorandom Functions — The Definition

- A binary partial function is *efficient* iff there exists a poltyime algorithm that computes it.
- We consider the space of functions from $\{0,1\}^n$ to $\{0,1\}^n$.
  - This space is *finite* and has cardinality $2^{n \cdot 2^n}$, because any of its function can be seen as a table of binary values with 2^n rows and n columns.
  - It therefore makes sense to consider **uniform distribution** on such a space, which assigns probability $\frac{1}{2^{n \cdot 2^n}}$ to every function.

Definition

Given a binary partial function F, which is length-preserving and efficient, we say that F is a *pseudorandom function* (PRF) iff for every distinguisher D that is PPT there exists a negligible function ε such that

$$|Pr(D^{F_k(\cdot)}(1^n) = 1) - Pr(D^{f(\cdot)}(1^n) = 1)| \leq \varepsilon(n)$$

Pseudorandom Functions — The Definition

- Note carefully how, in the definition of a pseudorandom function:
  - k is chosen among all strings of length n randomly.
  - $f(\cdot)$ is chosen among all functions from $\{0,1\}^n$ to $\{0,1\}^n$ randomly.
- About the existence of pseudorandom functions, we can say that:
  - As with pseudorandom generators, the existence of pseudorandom functions is not known in an absolute sense.
  - From a pseudorandom generator we can construct a pseudorandom function and viceversa.
  - So-called **block ciphers** (such as DES or AES) are constructed to satisfy the axiom of pseudorandom functions together with other axioms.

A CPA-Secure Encryption Scheme

Definition (PRF-Induced Scheme)

Given a pseudorandom function F, the scheme $\Pi^F = (Gen, Enc, Dec)$ is defined as follows:

- The algorithm Gen on input 1^n outputs each string of length n with same probability, i.e. $\frac{1}{2^n}$.
- $Enc(m, k)$ is defined as the (binary encoding of the) pair $\langle r, F_k(r) \oplus m \rangle$, where r is a random string $|k|$ bits long.
- $Dec(c, k)$ returns $F_k(r) \oplus s$ anytime c is the (binary encoding of the) pair $\langle r, s \rangle$.

A CPA-Secure Encryption Scheme

Definition (PRF-Induced Scheme)

Given a pseudorandom function F, the scheme $\Pi^F = (Gen, Enc, Dec)$ is defined as follows:

- The algorithm Gen on input 1^n outputs each string of length n with same probability, i.e. $\frac{1}{2^n}$.
- $Enc(m, k)$ is defined as the (binary encoding of the) pair $\langle r, F_k(r) \oplus m \rangle$, where r is a random string $|k|$ bits long.
- $Dec(c, k)$ returns $F_k(r) \oplus s$ anytime c is the (binary encoding of the) pair $\langle r, s \rangle$.

- It is easy to see that Π^F is at least a correct encryption scheme.

A CPA-Secure Encryption Scheme

Definition (PRF-Induced Scheme)

Given a pseudorandom function F, the scheme $\Pi^F = (Gen, Enc, Dec)$ is defined as follows:

- The algorithm Gen on input 1^n outputs each string of length n with same probability, i.e. $\frac{1}{2^n}$.
- $Enc(m, k)$ is defined as the (binary encoding of the) pair $\langle r, F_k(r) \oplus m \rangle$, where r is a random string $|k|$ bits long.
- $Dec(c, k)$ returns $F_k(r) \oplus s$ anytime c is the (binary encoding of the) pair $\langle r, s \rangle$.

- It is easy to see that Π^F is at least a correct encryption scheme.

Theorem

If F is a PRF, then Π^F is secure against CPA attacks.

Handling Variable-Length Messages

- The Π^F cipher is CPA-secure anytime F is a PRF, but it can only handle messages of length equal to the length of the key.
    - Therefore, it has the same restrictions that we have seen when talking about perfect secrecy.

Handling Variable-Length Messages

- The Π^F cipher is CPA-secure anytime F is a PRF, but it can only handle messages of length equal to the length of the key.
  - Therefore, it has the same restrictions that we have seen when talking about perfect secrecy.
- However, there is a way to generalize every CPA-secure cipher Π for messages of length n to a cipher Π^* for messages $n\ell(n)$, where ℓ is a polynomial:

$$Enc^*(k, m_0||\cdots||m_{\ell(n)}) = Enc(k, m_0)||\cdot||Enc(k, m_{\ell(n)})$$

Handling Variable-Length Messages

- The Π^F cipher is CPA-secure anytime F is a PRF, but it can only handle messages of length equal to the length of the key.
  - Therefore, it has the same restrictions that we have seen when talking about perfect secrecy.
- However, there is a way to generalize every CPA-secure cipher Π for messages of length n to a cipher Π^* for messages $n\ell(n)$, where ℓ is a polynomial:

$$Enc^*(k, m_0||\cdots||m_{\ell(n)}) = Enc(k, m_0)||\cdot||Enc(k, m_{\ell(n)})$$

Theorem

If Π is CPA-secure, then Π^* is also CPA-secure.

Pseudorandom Permutations

- An *permutation* of a set X is nothing else than a bijective function from X to X.
- The number of permutations of the set $\{0,1\}^n$ is $(2^n)!$, thus a number which is smaller than the number of functions on the set.
- The notion of **pseudorandom permutation** is given in a very similar way to that of a pseudorandom function, requiring, however, that the inverse of F_k is also efficient.
- Sometimes, however, adversaries also have access to the inverse function (which always exists), and not only to the function itself. In that case it makes sense to require that

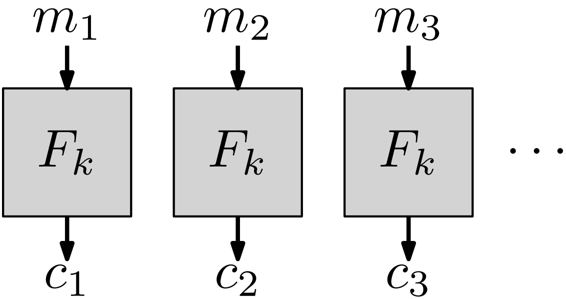
  $$|Pr(D^{F_k(\cdot),F_k^{-1}(\cdot)}(1^n)=1)-Pr(D^{f(\cdot),f^{-1}(\cdot)}(1^n)=1)|\leq\varepsilon(n)$$

  thus obtaining the notion of **strongly pseudorandom permutation**.
- Block ciphers are often thought to be strongly pseudorandom permutations.

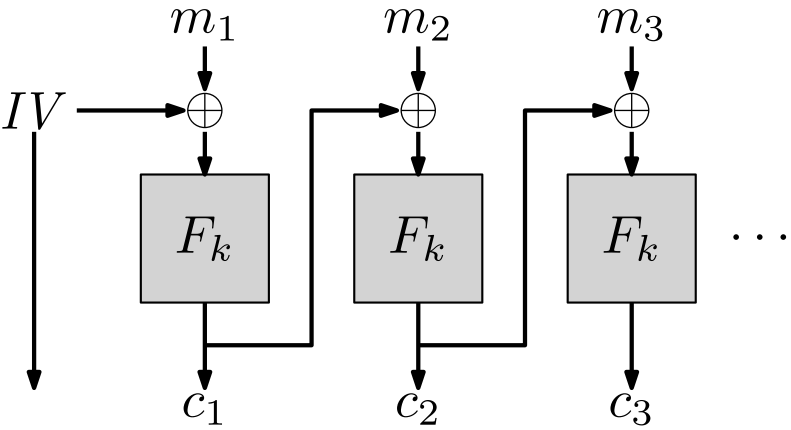
Modes of Operation

- The construction Π is just one of the many ways in which a block cipher (i.e., a strongly pseudorandom *permutation*) can be used to construct an encryption scheme for messages of length greater than $|k|$.
- The term **mode of operation** refers to that, and in the literature there is an infinite number of modes of operation.
- In the mode of operation we always proceed by splitting the message m to be encrypted into a sequence of submessages $m_1 \cdots m_\ell$ each of length equal to $|k|$, and so assuming that $|m|$ is a multiple of $|k|$.

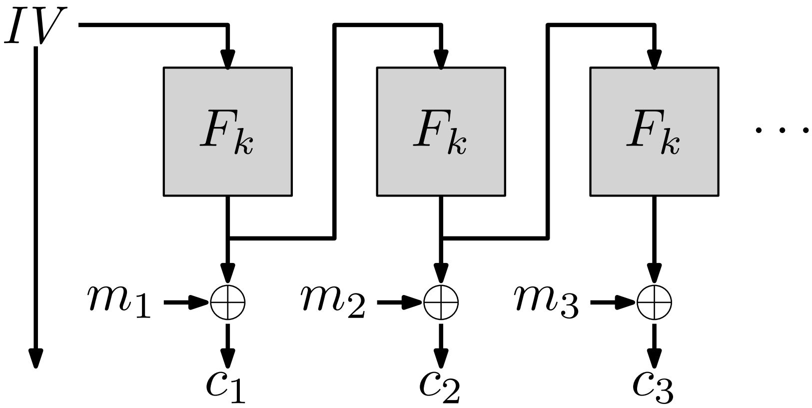
Electronic Code Book (ECB) Mode



Cipher Block Chaining (CBC) Mode



Output Feedback (OFB) Mode



Counter (CTR) Mode