Definitions and concepts

based on A Graduate Course in Applied Cryptography

Secret key Encryption -- Ciphers and security definitions

Suppose Alice and Bob share a secret key k. Alice wants to transmit a message m to Bob over a network while maintaining the secrecy of m in the presence of an eavesdropping adversary. Encryption does not solve all problems of **secure comunication**:

- Alice can send a *single message*. There are different concerns if we want to send multiple messages using the same key k.
- The eavesdropper / attacker **cannot modify** the message. Identifying a modified message is the problem of *message integrity* not *message secrecy*
- We assume a common secret key k is known in the first place - This is an assumption of **secret key cryptography** (symmetric cryptography). There are many ways to reach this key (Alice and Bob can meet in real life, they can use another protocol etc), but the way to reach that secret key is not the purpose of this type of cryptography.

Cipher -- Definition

Let $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ be key, message and ciphertext spaces. A **cipher** is defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ and is a pair of efficient encryption-decryption algorithms (E, D) such that:

- E may be randomized, D is deterministic.

$$E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}; \quad c = E(k, m)$$
$$D : \mathcal{K} \times \mathcal{C} \to \mathcal{M}; \quad m = D(k, c)$$

- has the **Corectness propriety** (decryption "undoes" encryption):

$$\forall m \in \mathcal{M}, k \in \mathcal{K} \Rightarrow D(k, (E(k, m))) = m$$

$$\boxed{m} \xrightarrow[E(k,m)]{\text{Alice encrypts}} \boxed{c} \xRightarrow{\text{insecure channel}} \boxed{c} \xrightarrow[D(k,c)]{\text{Bob decrypts}} \boxed{m}$$

In practice key, messages and ciphertexts are represented as sequences of bits. They can also be whatever mathemathical object (polynomials, matrices, group elements) but for practical purposes they must also be represented as bits for them to be passed over a network and stored.
Usually messages and ciphertexts can be of variable length and keys have a fixed length (Ex: 128b key).

Kerchoff's principle

The method must not be required to be secret, and it must be able to fall into the enemy's hands without causing inconvenience.

Intuition: **All** of the security of the system should be concentrated in the **secrecy of the key**, not the secrecy of the algorithms. In practice *security through obscurity* didn't work well and we want protocol / primitives / function (Ex: (E, D)) to pe public knowledge. This way the professionals (cryptanalysts) can analyze the security of the constructions and try to break them.

Example: One time pad
Let $(\mathcal{K}, \mathcal{M}, \mathcal{C}) = \{0,1\}^L$ be sets of L bit strings. The encryption and decryption are done by `xor`ing the key with the message / ciphertext.

- Encryption: $E(k, m) = k \oplus m$
- Decryption: $D(k, c) = k \oplus c$
- Corectness: $D(k, E(k, c)) = k \oplus k \oplus m = m$

Perfect security

The idea is that the message m remains "well hidden" even after the eavesdropper / attacker seees the ciphertext c. Intuitively we use the key k and the encryption algorithm E to turn the message m into "noisy garbage". When we think of "noisy garbage" we think of random stuff that makes no sense, and this helps us build the intuition for what we want our c to look like: **indistinguishable from random**.

Perfect security -- definition

A cipher (E, D) has perfect secrecy if $\forall m_0, m_1 \in \mathcal{M}, c \in \mathcal{C}$ and the uniform key distribution $\boldsymbol{K}$ we have:

$$Pr[E(\boldsymbol{K}, m_0) = c] = Pr[E(\boldsymbol{K}, m_1) = c]$$

where $E(\boldsymbol{K}, m)$ is now a distribution too.

Intuition:

- All messages are **equally** likely to be the ciphertext.
- No adversary can learn something about m given c. This means that if we have the distribution of messages $\boldsymbol{M}$ (not necessarily uniform) and the distribution of ciphertexts $\boldsymbol{C} = E(\boldsymbol{K}, \boldsymbol{M})$ the following holds for all ciphertexts c and messages m:

  $$Pr[\boldsymbol{M} = m | \boldsymbol{C} = c] = Pr[\boldsymbol{M} = m]$$

  In short, $\boldsymbol{C}$ and $\boldsymbol{M}$ must be **independent** when we want perfect security. Also, the choice of the message must have no impact on the distribution of ciphertext:

  $$Pr[\boldsymbol{C} = c | \boldsymbol{M} = m] = Pr[\boldsymbol{C} = c]$$

We can also develop another characterization of perfect security. Suppose we have an adversary that can tinker with the ciphertexts. We call this tinkering a predicate ϕ. Now we can rephrase the definition to include all possible predicates ϕ that can be applied on $\mathcal{C}$:

$$Pr[\phi(E(\boldsymbol{K}, m_0))] = Pr[\phi(E(\boldsymbol{K}, m_1))]$$

Example:
OTP with $(\mathcal{K}, \mathcal{M}, \mathcal{C}) = \{0,1\}^L$ is perfectly secure.
However, a variable length OTP is not perfectly secure (we leak the ciphertext length, this may or may not be a problem). If we have 2 messages m_0 and m_1 of length l_0 and l_1 and a ciphertext c of length l_0 then $P[E(\boldsymbol{K}, m_1) = c] = 0$, meaning that we know for sure that c cannot be the encryption of m_1 therefore ruining our definition of perfect security which says that the distributions $\boldsymbol{M}$ and $\boldsymbol{C}$ must be independent (the adversary learns that the message cannot be m_1).

However, the bad news:
If (E, D) has perfect secrecy $\Rightarrow |\mathcal{K}| \geq |\mathcal{M}| \Rightarrow$ So for each message we will need to send a key that is the same or bigger size $\Rightarrow$ very impractical

Proof:
Let $|\mathcal{K}| < |\mathcal{M}|$. Like in the OTP example above we want to prove that we can construct a situation where given 2 messages m_0, m_1 and an ecryption c there is a chance that $E(\boldsymbol{K}, m_0) = c$ and **no** chance that $E(\boldsymbol{K}, m_1) = c$.
We choose $c = E(k_0, m_0)$ for a key $k_0 \in \mathcal{K}$ to be the encryption of m_0. Then we consider the set $\mathcal{S}$ of all possible decryptions of c using keys from $\mathcal{K}$. We have $|\mathcal{S}| \leq |\mathcal{K}| < |\mathcal{M}|$ and $\mathcal{S} \subset \mathcal{M}$.
We choose $m_1 \in \mathcal{M} \setminus \mathcal{S}$. To prove that we can't decrypt c to m_1 we need to prove that there is no key k that can do the decryption. Assume the contrary, that there is a key. Then we could decrypt $c \Rightarrow m_1 \in \mathcal{S}$ which contradicts how we chose m_1.

Semantic security

video explanation

Semantic security -- Definition

Instead of insisting that the probabilities from perfect security definition are equal, we insist they are close:

$$Pr[\phi(E(\boldsymbol{K}, m_0))] - Pr[\phi(E(\boldsymbol{K}, m_1))] < \epsilon$$

where ϵ is negligible quantity.

Remark:

- In order to achieve computationally efficient results we need to relax the definition of security.
- Practical Relaxation:
  - For all practical purposes instead of considering all possible predicates ϕ (ways for an adversary to tinker with the ciphertext) we consider only the **efficient** ones.
  - Instead of taking into account all possible generated messages m_0, m_1 we consider only those that can be generated by efficient algorithms.

Security using attack games.

There are other ways to define security concepts. One of the most common ones is to formulate a game with a **challenger** and an **adversary**. The challenger sets up a game and offers some information and the adversary must win the game or gain a significant **advantage**.
This way the challenger can control the difficulty of the game. If the game is too hard we can choose to give the adversary more power or set a lower advantage threshold for the adversary to achieve. Using these tools we can adjust how strong the security definitions are.

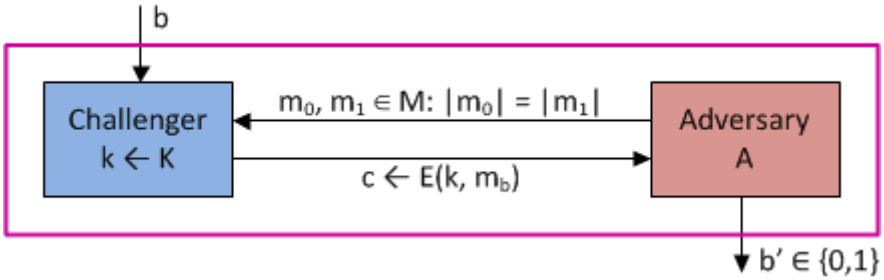
Semantic security -- Attack game

- Let $\mathcal{E} = (E, D)$ be a cipher
- Let $EXP(0)$, and $EXP(1)$ be two experiments
- An adversary A sends $m_0, m_1 \in M$ to the challenger
- The challenger sends an encryption of **one** of them ($EXP(0)$ or $EXP(1)$)
- The adversary must guess which one of the experiments was received.
- $W_b = \text{ event that } EXP(b) = 1$ = event that in $EXP(b)$ the Adversary outputs 1
- $\text{Adv}_{SS}(A, \mathcal{E}) = |Pr[W_0] - Pr[W_1]| \in [0, 1]$
  - *intuition*: We look if the adversary behaves diferently if he is given one ciphertext or the other
  - If Adv is close to 1 => The adversary can distinguish between the encryptions

$\mathcal{E}$ is semantically secure if for all efficient adversaries A

- $\text{Adv}_{SS}(A, \mathcal{E}) < \epsilon$

Intuition: This is similar to a bit guessing game. In this game the challenger rolls a bit b and based on the outcome decides which ciphertext $c \in \{c_0, c_1\}$ to send to the adversary. The adversary must guess the bit's value based on the received ciphertext c.



Attack games

In general, in cryptography we can use attack games to see if a property is satisfied. We need to consider and **carefully** define following when defining an attack game:

- The property that we want to satisfy (Ex: message secrecy, message integrity)
- The cryptographic system (primitive, protocol)
- The adversary (what is his computational power? what can he do in this game?)
- The challenger, the challenge and what the adversary needs to do to break security

Efficient adversaries and negligible quantities

Poly-bounded functions

A function is poly bounded if it's bounded in absolute value by a polynomial.

$$\exists c, d \in \mathbb{R}_{>0} \text{ such that } \forall n \in \mathbb{Z}_+, |f(n)| < n^c + d$$

Negligible function

A function is negligible if at some point it starts being bounded by the inverse of a polynomial.

$$\exists c \in \mathbb{R}_{>0}, n_0 \in \mathbb{Z}_{\geq 1} \text{ such that } \forall n > n_0 \in \mathbb{Z}_+, |f(n)| < \frac{1}{n^c}$$

Super poly

A function is superpoly if $1/f$ is negligible.

Security parameters

Cryptography constructions usually take a security parameter $\lambda \in \mathbb{Z}_+$ that tells how hard is to break this protocol. Generally a higher λ implies higher levels of security. However this usually comes at the cost of slower encryption and decryption speeds and bigger key sizes.

Efficient algorithm

An algorithm that takes a security parameter λ is efficient if there exists a poly-bounded function $t(\lambda)$ and a **negligible** quantity $\epsilon(\lambda)$ such that for all reasonable-sized inputs (poly bounded) the running time of the algorithm exceeds $t(\lambda)$ with the probability at most $\epsilon(\lambda)$

Efficient adversary

We say that an adversary is efficient if there exists a poly-bounded function t and a negligible quantity ϵ such that $\forall$ environments the probability that the total running time of the adversary's algorithm exceeds $t(\lambda)$ is at most $\epsilon(\lambda)$

Intuition:

- An efficient adversary is an adversary that has a reasonable compute power and time to break the system. In practice this is determined by the system (In an online system might have only a few seconds / minutes) but if the adversary gets your encryption offline he might have months or years to break it.
- If a quantity negligible that means that we can treat it as being equal to 0 for all practical purposes

Using the concepts of negligible quantities, efficient adversary and security parameters we can enhance the understanding of how big adversary's advantage can be in security games: this advantage is dependent on the security parameter λ.
Essentially we consider (E, D) a cipher insecure if there exists an adversary A that for infinitely many λ (this does not mean all possible λ, rather starting from one point forward) his advantage is:

$$\mathrm{Adv}_{SS}(A, (E, D)) > \frac{1}{\lambda^c}$$

for some $c > 0$