

Symmetric Key Encryption

A symmetric key encryption scheme is defined by a message space M along with algorithms

$\text{Gen} \rightarrow$ Probabilistic algorithm that outputs a key k according to some distribution on the keyspace K

$\text{Enc} \rightarrow$ Takes as input a key $k \in K$, message m , outputs a ciphertext c . ($\text{Enc}_k(m) = c$)

$\text{Dec} \rightarrow$ Takes as input a key $k \in K$, ciphertext c , outputs a message m . ($\text{Dec}_k(c) = m$)

correctness $\rightarrow \text{Dec}_k(\text{Enc}_k(m)) = m$

Kerckhoff's Principle

The details of the encryption scheme are not secret.

Then what is?

The key!

security \rightarrow A ciphertext should leak nothing about the plaintext that the attacker doesn't already know

↑
— each security definition will state exactly what the attacker knows (i.e. how much power the attacker has)

— leak nothing usually means that the attacker can't win the security game (i.e. the attacker can't guess which message was encrypted)

Perfect Secrecy

For every $m, m' \in M, c \in C$ (for any messages m, m' & any ciphertext c)

$$\Pr[Enc_k(m) = c] = \Pr[Enc_k(m') = c]$$

i.e. the probability (over the choice of key k) that c is an encryption of m is equivalent to the probability that c is an encryption of m'

Fact: Perfect secrecy can only be achieved when the key is at least as long as the message.

Fact: The one-time pad is essentially the best we can do to achieve perfect secrecy.

(when $\text{length}(\text{key}) = \text{length}(\text{message})$ the only way to get perfect secrecy is to use the one time pad)

Computational Security

The definition of perfect secrecy makes no assumptions about the computational power of the attacker \mathcal{A} .

The following definitions in contrast all refer to an efficient (i.e computationally bounded) attacker. This allows us to encrypt messages using secret keys smaller than the message.

efficient attacker \rightarrow probabilistic polynomial time (PPT)

an attacker with polynomial amount of time in the length n of the message (eg: n^2 , $5n^{17} + n$) the attacker's algorithm can be probabilistic (eg: randomly output either x or y)

EAV-game

- \rightarrow The key k is generated according to the algorithm Gen
- \rightarrow \mathcal{A} outputs any two messages m_0, m_1 from the message space M
- \rightarrow A random bit $b \in \{0,1\}$ is chosen uniformly & the encryption of message m_b , $c \leftarrow \text{Enc}_k(m_b)$ is given to \mathcal{A}
- \rightarrow \mathcal{A} outputs a guess, b' , as to which message was encrypted

(The adversary \mathcal{A} succeeds if $b = b'$)

EAV-security

No efficient (PPT) attacker \mathcal{A} succeeds at the EAV-game with probability much higher than $\frac{1}{2}$

i.e. no efficient attacker can do much better than randomly guess their answer b' for the EAV-game

Chosen Plaintext Attack

Security against an attacker who can get encryptions of any messages they choose

CPA - game

- The key k is generated according to the algorithm Gen
- \mathcal{A} outputs any two messages m_0, m_1 from the message space M with access to encryption oracle $\text{Enc}_k(\cdot)$
- A random bit $b \in \{0,1\}$ is chosen uniformly & the encryption of message m_b , $c \leftarrow \text{Enc}_k(m_b)$ is given to \mathcal{A}
- \mathcal{A} outputs a guess, b' , as to which message was encrypted

CPA - security No efficient (PPT) attacker \mathcal{A} succeeds at the CPA-game with probability much higher than $\frac{1}{2}$

i.e. no efficient attacker can do much better than randomly guess bit b'



Note that since the attacker can get encryptions of any messages, a CPA secure encryption scheme must be randomized. i.e there must be many possible encryptions of a message m under a key k .

Chosen Ciphertext Attack

Security against an attacker who can get encryptions & decryptions of their choosing except for the decryption of the challenge ciphertext c itself.



decryption schemes are always deterministic.

This is exactly why we don't allow \mathcal{A} to get a decryption of

the challenge ciphertext

CCA - game

- The key k is generated according to the algorithm Gen
- \mathcal{A} outputs any two messages m_0, m_1 from M with access to encryption & decryption oracles $\text{Enc}_k(\cdot), \text{Dec}_k(\cdot)$
- A random bit $b \in \{0,1\}$ is chosen uniformly & the encryption of message m_b , $c \leftarrow \text{Enc}_k(m_b)$ is given to \mathcal{A}
- \mathcal{A} continues to have access to $\text{Enc}_k(\cdot), \text{Dec}_k(\cdot)$ but may not decrypt c . \mathcal{A} outputs a guess b'

CCA - security No efficient (PPT) attacker \mathcal{A} succeeds at the CCA-game with probability much higher than $\frac{1}{2}$

i.e. no efficient attacker can do much better than randomly guess bit b'

Pseudorandom Generator (PRG)

A function $G: \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ is a PRG if

- it expands the input (input length $n <$ output length $l(n)$)
- let s be a randomly chosen n bit input & r be a randomly chosen $l(n)$ bit string.

If given either $G(s)$ or r at random, no efficient (PPT) attacker can guess which one it was given with probability much better than random guessing ($\frac{1}{2}$)

$\left\{ \begin{array}{l} \text{The output of a PRG is indistinguishable from a randomly chosen string} \\ \text{from the output space. This is true even though } g(x) \text{ can only} \\ \text{take } 2^n \text{ values out of the output space of } 2^{l(n)} \text{ possible values} \end{array} \right.$

Pseudorandom Functions (PRF)

A pseudorandom function F is actually a distribution of functions F_k identified by their keys k . It is pseudorandom in the sense that F is indistinguishable from randomly choosing any function with the same domain and range as the functions in F .

How many functions $f: \{0,1\}^n \rightarrow \{0,1\}^n$ are there that map strings of length n to strings of length n ?

$$2^{2^n \cdot n}$$

How many functions in a PRF F whose keys are of length n ?

$$2^n$$

definition \rightarrow Let k be a randomly chosen key

Let f be a randomly chosen function

Given access to either the oracle $f(\cdot)$ or

the oracle $F_k(\cdot)$, no efficient attacker can guess which one it is interacting with

formal definitions

let $l_k, l_{\text{in}}, l_{\text{out}}$ be the key length, input length & output length respectively. Let F be a family of efficiently computable keyed functions.

$F: \{0,1\}^{l_k} \times \{0,1\}^{l_{\text{in}}} \rightarrow \{0,1\}^{l_{\text{out}}}$ is a PRF if no efficient attacker \mathcal{A} can win with probability much better than guessing $(\frac{1}{2})$ in the following game.

- choose a random key $k \in \{0,1\}^{l_k}$, and a random function f from $\{0,1\}^{l_{\text{in}}}$ to $\{0,1\}^{l_{\text{out}}}$
- choose a random bit $b \in \{0,1\}$
- if $b=0$, give \mathcal{A} the oracle $F_k(\cdot)$
- if $b=1$, give \mathcal{A} the oracle $f(\cdot)$
- \mathcal{A} outputs a guess b'



We provide oracles $F_k(\cdot)$ & $f(\cdot)$ because even though F_k is efficiently computable, the description of $f(\cdot)$ itself may be exponential in length in which case a polynomial time attacker would not even have time to read its description

Key Exchange

Eavesdropper Game

- ↳ Give the attacker \mathcal{A} all the messages exchanged between Alice & Bob
- ↳ Choose a random bit b
 - If $b=0$, give \mathcal{A} the key output by Alice & Bob
 - If $b=1$, give \mathcal{A} a key chosen randomly from the keyspace K
- ↳ \mathcal{A} outputs a guess b'

A key exchange protocol is secure if no efficient attacker can tell the real key from one chosen randomly even with access to the transcript of messages exchanged between Alice & Bob i.e. the transcript reveals nothing about the key

Decisional Diffie Hellman Assumption (DDH)

A group G with generator g satisfies the DDH assumption if no efficient adversary can distinguish (g^a, g^b, g^{ab}) from (g^a, g^b, g^c)

Computational Diffie Hellman Assumption (CDH)

A group G with generator g satisfies the CDH assumption if given g^a, g^b , no efficient adversary can compute g^{ab}

Diffie Hellman Key Exchange (Let G be a DDH group)

- ↳ Alice chooses $a \in \mathbb{Z}$ sends g^a to Bob
- ↳ Bob chooses $b \in \mathbb{Z}$ sends g^b to Alice
- ↳ Alice computes $(g^b)^a$ using her secret exponent a
- ↳ Bob computes $(g^a)^b$ using his secret exponent b

g^{ab} is indistinguishable from a random key g^c since the group satisfies the DDH assumption

Integrity of Communication

While encryption guarantees secrecy, message authentication codes (MACs) & signatures guarantee integrity. i.e each party can identify when a message it receives was sent by the party claiming to send it.

MACs

A message authentication code (MAC) is defined by the following algorithms :

- 1) Gen - probabilistic key generation algorithm

2) Mac - takes as input a key $k \in \mathcal{K}$ and message $m \in \mathcal{M}$ and outputs a tag $t \leftarrow \text{Mac}_k(m)$

3) Vrfy - takes as input a key k , tag t , message m and outputs a bit 0 or 1 where 1 means t is a valid tag for m with respect to key k and 0 means it isn't.

correctness — $\text{Vrfy}_k(m, \text{Mac}_k(m)) = 1$

security — No efficient adversary can generate a valid tag on any "new" message that wasn't previously sent by one of the communicating parties.

Secure MAC

We will give \mathcal{A} access to a MAC oracle $\text{Mac}_k(\cdot)$. The attacker can submit any message m of its choice to this oracle and get back tag $t \leftarrow \text{Mac}_k(m)$.

The attacker breaks the scheme if it can output m, t such that

- 1) $\text{Vrfy}_k(m, t) = 1$ (i.e t is a valid tag on the message)
- 2) \mathcal{A} had not previously requested a tag on the message m

Strong MAC

We will give \mathcal{A} access to a MAC oracle $\text{Mac}_k(\cdot)$. The attacker can submit any message m of its choice to this oracle & get back tag $t \leftarrow \text{Mac}_k(m)$.

The attacker breaks the scheme if it can output m, t such that

- 1) $\text{Vrfy}_k(m, t) = 1$ (i.e t is a valid tag on the message)
- 2) \mathcal{A} had not previously received the tag t on message m

The attacker may have received a different tag on the same message m .

Digital Signature Schemes

Both MACs & Signature Schemes ensure integrity of messages. In the case of MACs, each pair of parties share a secret key that can be used by either party to both tag & verify messages.

Digital Signature Schemes are asymmetric/public key protocols. The signer generates a pair of keys (pk, sk) called the public key & private key respectively. The private key is used by the

Signer to sign messages while the public key is used by any party to verify signatures.

Therefore, signature schemes are publicly verifiable. This means that if a receiver verifies that a signature on a given message is legitimate, then all other parties who received the message can also verify it as legitimate. This feature is not achieved by MACs since the signer would share a separate key with each receiver and calculate separate tags with respect to each key.

Public verifiability ensures that signatures are transferable. This means that a signature σ on a message m received by party A can be shown to a third party C for verification, who can use the signer's public key to verify the signature.

Signatures are also non-repudiable which means that the signer cannot deny having signed a message.

A (digital) Signature Scheme is defined by the following algorithms:

1) Gen - probabilistic algorithm that outputs a pair of keys (pk, sk)

2) Sign - takes as input a private key sk & a message m & outputs a signature $\sigma \leftarrow \text{Sign}_{sk}(m)$

3) Vrfy - takes as input a public key pk , signature σ & message m & outputs 1 or 0 to signify that the signature is valid or invalid respectively

$$\text{Vrfy}_{pk}(\text{Sign}_{sk}(m)) = 1$$

Security of Digital Signature Schemes

For a fixed public key pk , a forgery is a message m with a valid signature σ , where m was not previously signed by the signer. Security of a signature schemes means that an attacker should be unable to output a forgery even if it obtains signatures on many other messages of its choice. This is exactly analogous to a secure MAC.

