CyBoK: Cryptography Knowledge Area

N.P. Smart

COSIC,
KU Leuven, ESAT,
Kasteelpark Arenberg 10, bus 2452,
B-3001 Leuven-Heverlee,
Belgium.

April 10, 2019

KU LEUVEN

Outline

Copyright

The CyBOK project would like to understand how the CyBOK is being used and its uptake.

The project would like organisations using, or intending to use, CyBOK for the purposes of education, training, course development, professional development etc. to contact it at contact@cybok.org to let the project know how they are using CyBOK.

Overview

The aim of the talk is to give a rapid overview of the Cryptography Knowledge Area

Covering the basic primitives and security definitions.

In this talk we focus on encryption and digital signatures.

- Cryptography covers a lot more than this though
    - Key Agreement Protocols
    - Authentication Protocols
    - Zero-Knowledge Protocols
    - Multi-Party Computation
    - ....

Encryption, Signatures and MACs

Symmetric Key vs Public Key Encryption

Symmetric Key Encryption:

The basic idea of public key encryption is:

Message + Secret Key = Ciphertext
Ciphertext + Secret Key = Message

Both parties need the same secret key to encrypt and decrypt the message.

Public Key Encryption:

The basic idea of public key encryption is:

Message + Alice's Public Key = Ciphertext
Ciphertext + Alice's Private Key = Message

Anyone with Alice's public key can send Alice a secret message, but only Alice can decrypt.

Notation

Henceforth we denote a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$, and a symmetric key by sk.

A message is denoted m, an encryption algorithm is denoted by Enc, a decryption algorithm by Dec, and a ciphertext by c.

$$\mathsf{Enc}_{\mathsf{sk}}(m) = c \text{ and } \mathsf{Dec}_{\mathsf{sk}}(c) = m.$$

or (for public key schemes)...

$$\mathsf{Enc}_{\mathsf{pk}}(m) = c \text{ and } \mathsf{Dec}_{\mathsf{sk}}(c) = m.$$

Digital Signatures and MACs

Another very important public key primitive is the digital signature, with the associated secret key primitive being a MAC function.

Digital Signature:

Message + Alice's Private Key = Signature
Message + Signature + Alice's Public Key = YES/NO

Alice can sign a message using her private key, and anyone can verify Alice's signature, since everyone can obtain her public key.

MAC Functions:

Message + Secret Key = Tag
Message + Tag + Secret Key = YES/NO

Need the secret key to verify the tag.

Notation

Henceforth we denote a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.

A message is denoted m, a signing algorithm is denoted Sig, a verification algorithm is denoted Verify, a signature is denoted s.

$$\mathsf{Sig}_{\mathsf{sk}}(m) = s \text{ and } \mathsf{Verify}_{\mathsf{pk}}(s, m) = YES/NO.$$

In the case of MACs we have the tag production algorithm is MAC and the equations are

$$\mathsf{MAC}_{\mathsf{sk}}(m) = t \text{ and } \mathsf{Verify}_{\mathsf{sk}}(t, m) = YES/NO.$$

Security Definitions

Security Definitions

In much of cryptography security is defined by a game.

The game is between a Challenger and an Adversary.

The Adversary is given

- A goal to achieve (see OW/IND/UF etc)
- Powers it can use (see CPA/CCA/CMA)
- Restrictions on its operations (see ROM)

Security Goals for Encryption

There are two main security goals

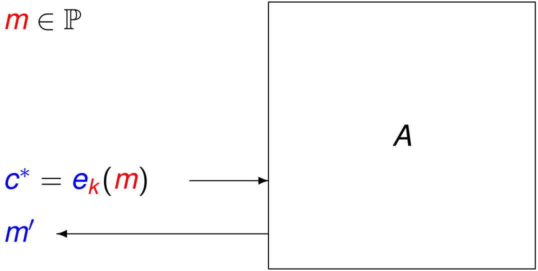
- OW: One way security. Can you decrypt a message?
- IND: Indistinguishability. Can you learn any information about a message?

The later is the one we aim for.

The former is what primitives sometimes achieve.

OW Security: Symmetric Key Case

Perhaps the most basic notion of security could be defined by the following game



IND-Security

This is the preferred security definition

Suppose that the challenger is given an encryption function f

- Defined by some key, i.e. $f(m) = e_k(m)$.

The attacker chooses two messages m_1 and m_2 of equal length.

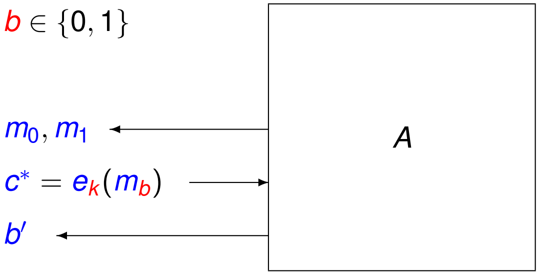
The challenger gives the attacker given a ciphertext c such that

$$c = f(m_1) \text{ or } c = f(m_2).$$

The goal is for the adversary to work out which message was encrypted.

IND-Security: Symmetric Key Case

It is simpler to present this in terms of pictures representing a game played with the adversary A

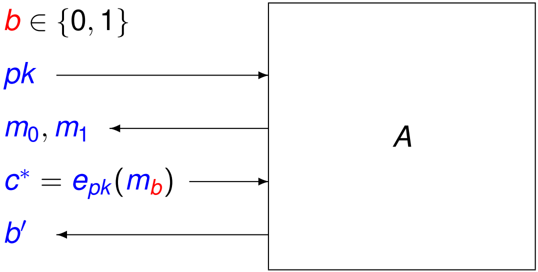


The ciphertext c^* is called the target ciphertext.

Remember we must have $|m_0| = |m_1|$.

IND-Security (Public Key Case)

For the public key case there is one main difference in the picture:



Adversarial Powers

IND and OW are definitions of adversarial goals.

- They say nothing about what powers we give the adversary

We define powers by giving the adversary access to various oracles.

Passive Attack

The adversary is given no oracles (the pictures are as above)

Chosen Plaintext Attack (CPA)

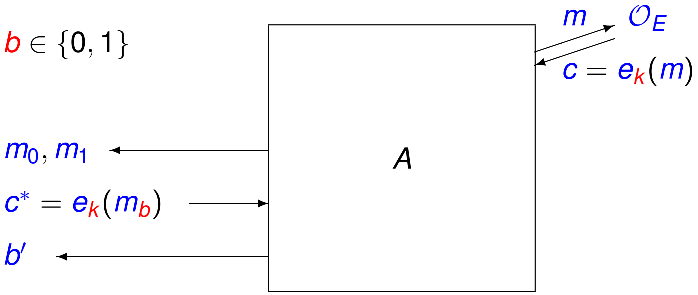
The adversary can encrypt any message of his choosing.

Chosen Ciphertext Attack (CCA)

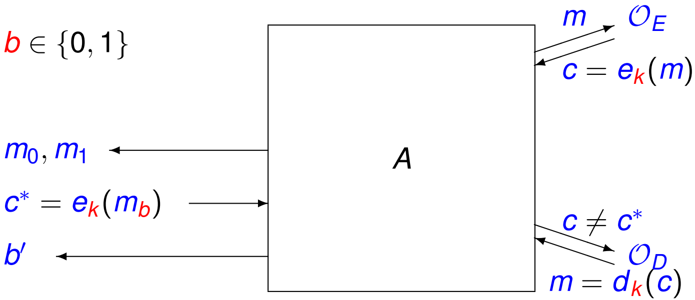
The adversary can decrypt any message of his choosing, except he is not allowed to decrypt c^*.

We say a scheme is IND-PASS, IND-CPA, IND-CCA, OW-PASS, OW-CPA, OW-CCA etc.

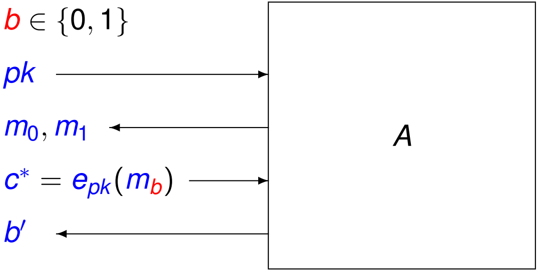
IND-CPA Symmetric Case



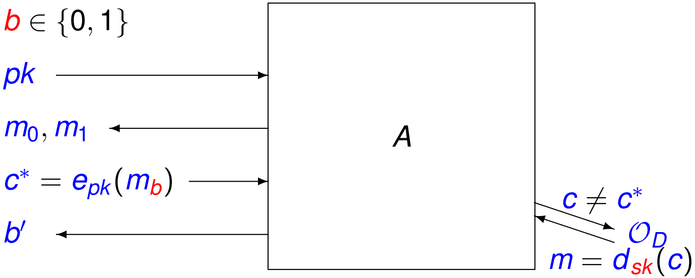
IND-CCA Symmetric Case



IND-CPA Public Key Case



IND-CCA Public Key Case



MAC Security Game

One can similarly define a security game for MAC security

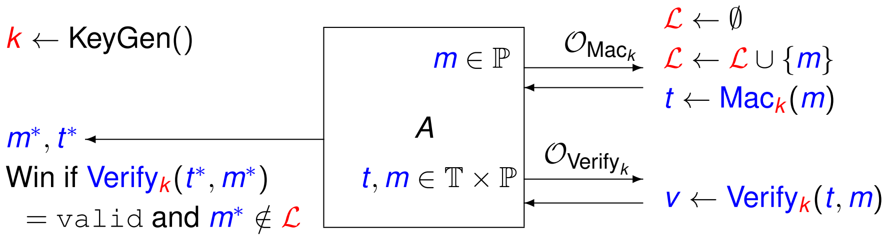


Figure : Security game for MAC security EUF-CMA

Signature Security Game

And for signature security

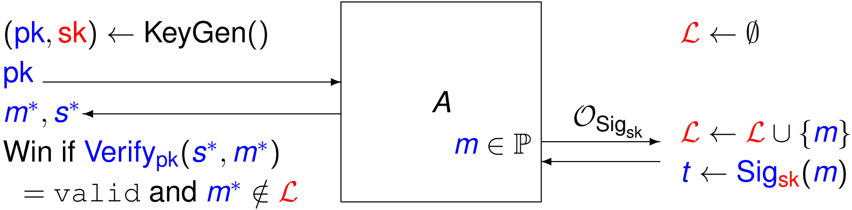


Figure : Security game for signature security EUF-CMA

Symmetric Key Primitives and Schemes

Block Ciphers

The basic building block of modern symmetric primitives is a block cipher

This is a keyed function which maps a block of bits to another block of bits

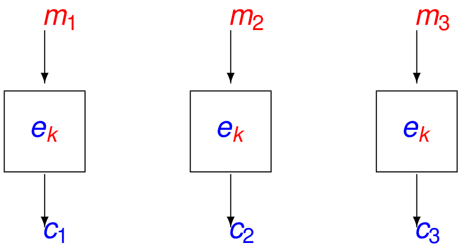
$$e_k : \{0,1\}^b \longrightarrow \{0,1\}^b$$

The "standard" block cipher is the AES (Advanced Encryption Standard).

- Has a block length of $b = 128$.
- Has a key length of 128, 192 or 256 bits.

On its own a block cipher is useless, it needs to be combined into a mode of operation

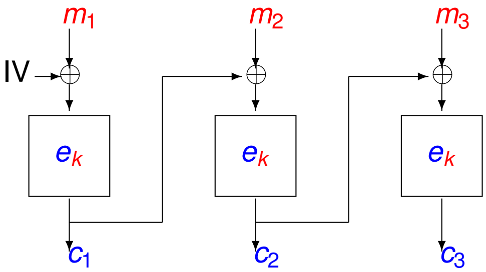
Security of Symmetric Modes of Operation: ECB



ECB Mode is OW-PASS and OW-CPA.

ECB Mode is not OW-CCA, IND-PASS, IND-CPA, IND-CCA.

Security of Symmetric Modes of Operation: CBC



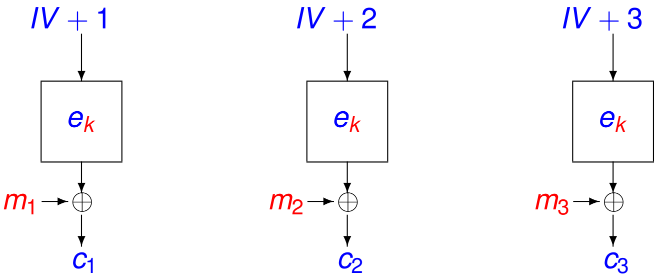
CBC Mode is OW-CPA and IND-CPA.

- With all zero IV it is only IND-PASS.

CBC Mode is not OW-CCA or IND-CCA.

So CBC is better than ECB at least!

Security of Symmetric Modes of Operation: CTR



CTR Mode is OW-CPA and IND-CPA.

- With all zero IV it is only IND-PASS.

CTR Mode is not OW-CCA or IND-CCA.

Producing an IND-CCA Mode of Operation

CBC and CTR Mode were only IND-CPA.

- Problem was the adversary could write down a valid ciphertext which was related to the target one
- He then calls his decryption oracle on this valid ciphertext

Idea is to stop the adversary writing down a valid ciphertext.

To construct an IND-CCA secure scheme we take

- An IND-CPA secure symmetric cipher E and
- An EUF-PASS secure MAC function MAC.
- The key for our new scheme E^* consists of a key k_0 for E and a key k_1 for MAC.

IND-CCA Symmetric Encryption

The function $E_k{}^*(m)$ is then constructed as follows.

- Split k into k_0 and k_1.
- $c_0 = E_{k_0}(m)$.
- $c_1 = \mathrm{MAC}_{k_1}(c_0)$.
- Return $c = (c_0, c_1)$.

Decryption, defined as $D_k{}^*(c)$, is then constructed as follows.

- Split k into k_0 and k_1.
- Split c into c_0 and c_1.
- $m = D_{k_0}(c_0)$.
- $c_1' = \mathrm{MAC}_{k_1}(c_0)$.
- If $c_1' \neq c_1$ then return $\perp$.
- Return m.

Public Key Encryption Schemes

RSA - Key Generation

Key generation:

- Generate two large primes p and q of at least 1024 bits.
- Compute $N = p \cdot q$ and $\phi(N) = (p-1)(q-1)$.
- Select a random integer e, $1 < e < \phi(N)$, such that

$$\gcd(e, (p-1)(q-1)) = 1.$$

- Using the XGCD compute the unique integer d, $1 < d < \phi(N)$ with

$$e \cdot d \equiv 1 \pmod{\phi(N)}.$$

Public key = (N, e) which can be published.
Private key = (d, p, q) which needs to be kept secret.

The RSA Function

The two keys define a *trapdoor one-way permutation*

$$\mathrm{RSA} : \left\{ \begin{array}{ccc} (\mathbb{Z}/N\mathbb{Z})^* & \longrightarrow & (\mathbb{Z}/N\mathbb{Z})^* \\ m & \longmapsto & m^e \pmod{N} \end{array} \right.$$

with trapdoor inverse...

$$\mathrm{RSA}^{-1} : \left\{ \begin{array}{ccc} (\mathbb{Z}/N\mathbb{Z})^* & \longrightarrow & (\mathbb{Z}/N\mathbb{Z})^* \\ c & \longmapsto & c^d \pmod{N} \end{array} \right.$$

The RSA-Problem is to invert the RSA function when you are not given d.

The Text-Book RSA encryption scheme is to encrypt messages using the RSA function, and decrypt them with the inverse function.

Discrete Logarithms

Suppose you are given a finite abelian group G of prime order q generated by P, so $q \cdot P = \mathcal{O}$.

The Discrete Logarithm Problem (DLP) is to invert the function

$$\text{DLP} : \begin{cases} (\mathbb{Z}/q\mathbb{Z})^* & \longrightarrow & G \\ m & \longmapsto & m \cdot P \end{cases}$$

This problem is believed to be hard if you select your group correctly

- e.g. certain Elliptic curve groups

Diffie–Hellman Problems

The Computational Diffie–Hellman (CDH) problem is given the tuple

$$(P, P_x, P_y) = (P, x \cdot P, y \cdot P)$$

to find

$$P_z = (x \cdot y) \cdot P.$$

The Decision Diffie–Hellman (DDH) problem is given the tuple

$$(P, P_x, P_y, P_z) = (P, x \cdot P, y \cdot P, z \cdot P)$$

where z is selected with probability $1/2$ to be uniformly random, and with probability $1/2$ to be equal to $x \cdot y \pmod{q}$. Then determine which case you are in.

ElGamal Encryption

The basic DLP based encryption algorithm is ElGamal

Key Generation:

- Secret Key: $x \in \mathbb{Z}/q\mathbb{Z}$.
- Public Key: $Q \leftarrow x \cdot P$.

Encryption: To encrypt $M \in G$.

- Generate a random ephemeral key $k \in \mathbb{Z}/q\mathbb{Z}$.
- Compute $C_1 \leftarrow k \cdot P$ and $C_2 \leftarrow M + k \cdot Q$.

Decryption:

- $-x \cdot C_1 + C_2 = (-x \cdot k \cdot P) + M + k \cdot Q = M$.

ElGamal Encryption

ElGamal is OW-CPA if the CDH problem is hard

ElGamal is IND-CPA if the DDH problem is hard.

Thus neither Text-Book RSA or ElGamal is IND-CCA

- Which is what we want

They also have restricted (small) message spaces

To solve these problems we use a hybrid cipher approach...

KEMs and DEMs

Transmitting a key is easier than transmitting a message

As this is the main purpose of public key encryption it is worth just concentrating on this only

Such a mechanism is called a Key Encapsulation Mechanism

The data is then transmitted using a Data Encapsulation Mechanism

- Think of this as an IND-CCA symmetric cipher

Key Encapsulation Mechanisms

Key Encapsulation Mechanism

A KEM is an algorithm which takes as input a public key pk and outputs a pair (k, c) where

- $k \in \mathbb{K}$ is a key for a symmetric encryption function
- c is an encapsulation (encryption) of k using pk.

The inverse, decapsulation algorithm takes as input (c, sk) where

- c is an encapsulation under pk of some key k
- sk is the private key corresponding to pk.

It outputs

- either $\perp$ if c is an invalid encapsulation, or
- k if c is an encapsulation of the key k.

A KEM-DEM Hybrid Cipher

Using the primitives we have been discussing, a KEM-DEM hybrid encryption scheme can then be created as follows.

Encryption

- $(k, c) = \text{KEM}(pk)$
- $e = \text{DEM}(m, k)$
- Return (c, e)

Decryption

- $k = \text{KEM}^{-1}(c, sk)$
- If $k = \perp$ then return $\perp$
- $m = \text{DEM}^{-1}(e, k)$
- If $m = \perp$ then return $\perp$
- Return m

Constructing a KEM

So the only (public key related) thing we have not shown is how to construct a KEM from a basic primitive.

We will now do this assuming the basic primitive is a trapdoor permutation like RSA

$$f_{pk} : X \longrightarrow X.$$

The KEM which we will call FDH-KEM (this is not a standard name) uses a hash function

$$H : X \longrightarrow \mathbb{K}.$$

When used with RSA this is called RSA-KEM

FDH-KEM

Encapsulation

- Generate $x \in X$ at random.
- Compute $c = f_{pk}(x)$.
- Compute $k = H(x)$.
- Output (k, c).

Decapsulation

- Given c compute $x = f_{sk}^{-1}(c)$.
- Output $k = H(x)$.

DH-KEM

The following is DH-KEM (or DHIES-KEM), the standard KEM used with DLP based schemes:

- Private Key: x
- Public Key: $Q = x \cdot P$
- Encapsulation: $C = r \cdot P$, for $r \in \mathbb{Z}/q\mathbb{Z}$.
- Encapsulated Key: $k = H(r \cdot Q)$
- Decapsulation: $k = H(x \cdot C)$

The function H is a hash function which maps elements in the group to keys of the DEM we aim to use.

Public Key Signature Schemes

RSA Based Signing

Using a cryptographic hash function H it is possible to create a signature scheme based on RSA.

Suppose we have an RSA key pair (e, N), (d, N) such that N has n-bits.

We use a hash function $H : \{0, 1\}^* \to (\mathbb{Z}/N\mathbb{Z})^*$.

To sign $m \in \{0, 1\}^*$:

- Compute $H(m)$.
- Compute signature by 'decrypting' $H(m)$, i.e. by computing $s = H(m)^d \bmod N$.

RSA Based Signing

To verify signature s on message m:

- 'Encrypt' s to recover $h' = s^e \bmod N$.
- Compute $H(m)$.
- Check whether $h' = H(m)$.
- If $h' = H(m)$, accept the signature. Otherwise reject.

This construct is called RSA-FDH as the codomain of the hash function is the entire set $(\mathbb{Z}/N\mathbb{Z})^*$.

DLP Based Signatures

The Digital Signature Algorithm makes use of a finite abelian group G of prime order q generated by an element P

Each user generates a secret signing key $x \in \mathbb{Z}/q\mathbb{Z}$ at random and such that

- $0 < x < q$.

Public key is Q where

$$Q = [x] \cdot P.$$

We assume a public "conversion" function

$$f : G \longrightarrow \mathbb{Z}/q\mathbb{Z}.$$

The exact function depends on the group G being used.

DSA : Signing

To sign a message m the signer proceeds as follows.

- Signer computes a hash value $e = H(m)$.
- Signer chooses a random ephemeral key: $0 < k < q$.
- Signer computes $r = f([k] \cdot P)$.
- Finally, signer computes

$$s = (e + x \cdot r)/k \pmod{q}.$$

The signature on m is the pair (r, s).

DSA : Verification

To verify a signature (r, s) on a message m under public key Q, the verifier proceeds as follows.

The verifier computes the following.

- $e = H(m)$
- $a = e/s \pmod{q}$
- $b = r/s \pmod{q}$

The verifier accepts the signature if and only if $v = r$ where

$$v = f([a] \cdot P + [b] \cdot Q).$$

Conclusion

Conclusion

We have covered the basics of cryptography, but there is much more to be found in the CyBoK Knowledge Area document

- Key Agreement Protocols
- Authentication Protocols
- Zero-Knowledge Protocols
- Multi-Party Computation
- Block Chain Applications
- Private Information Retrieval
- Implementation Aspects
- Fully Homomorphic Encryption
- ....