Risposte domande IS

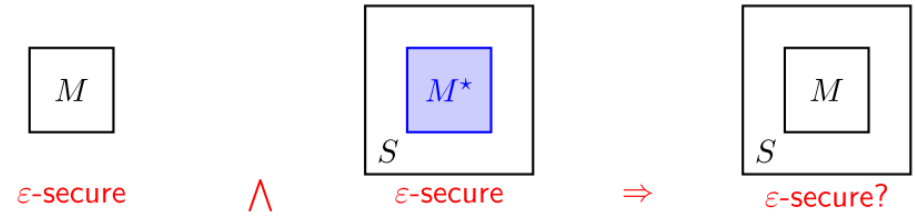
1)What is a composition theorem?

Context and motivation:

Consider a security mechanism S that makes use of another mechanism M, and denote this occurrence by $S[M]$.
Let $S[M^\star]$ denote the same mechanism S where M is replaced by its ideal counterpart $M^\star$.

The *composability* question

Is it possible to derive the security of $S[M]$ from those of M and $S[M^\star]$?



Teorema:

La triangular inequality è usata perché è one norm between statisical distribution

The composition theorem

Theorem (unconditional)

If M is ε_1-unconditionally secure and $S[M^\star]$ is ε_2-unconditionally secure, then $S[M]$ is $(\varepsilon_1 + \varepsilon_2)$-unconditionally secure

Proof.

Follows from the triangular inequality property of distinguishability. In fact:

$$d(S[M], S^\star) \leq d(S[M], S[M^\star]) + d(S[M^\star], S^\star)$$
$$\leq d(M, M^\star) + d(S[M^\star], S^\star) \leq \varepsilon_1 + \varepsilon_2$$

□

By repeatedly applying the above result, we can generalize to N-fold uses of M in S

Corollary

If M is ε_1-unconditionally secure and $S[M^\star]$ is ε_2-unconditionally secure, then $S[M^N]$ is $(N\varepsilon_1 + \varepsilon_2)$-unconditionally secure

2)We extended composition theorem to computational theorem? What is asymptotic version formulation in composition theorem?

Theorem (computational, concrete)

If M is (ε_1, T_0)-computationally secure and $S[M^\star]$ is (ε_2, T_0)-computationally secure, then $S[M]$ is $(\varepsilon_1 + \varepsilon_2, T_0)$-computationally secure

In the asymptotic form, the asymptotic security is retained even if M is used in S a number of times that is upper bounded by a polynomial in n, as follows

Theorem (computational, asymptotic)

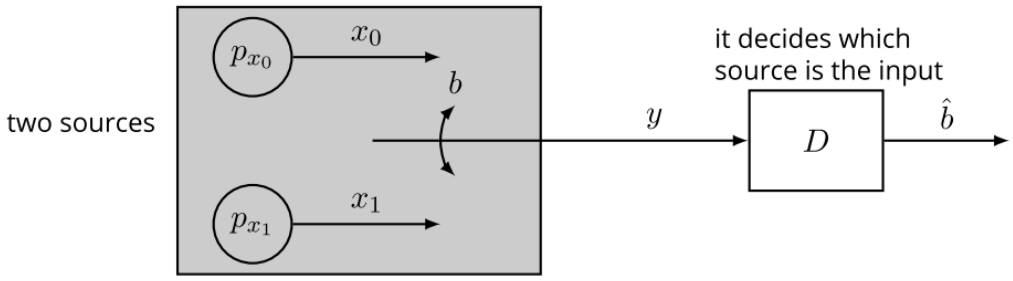
In the asymptotic formulation, if $\{M_n\}$ is computationally secure and $S_n[M_n^\star]$ is computationally secure, then for any polynomial $p(\cdot)$, $S_n[M_n^{p(n)}]$ is computationally secure

How many times do we use (m) in asymptotic version in polynomial? (??)

3)What type of distance is this? Variational Distance

Context:

Variable distinguishers (or binary hypothesis testing)



A distinguisher between two random variables x_0 and x_1 is a system D that is allowed to observe a realization of y without knowing in advance if $b = 0$ or $b = 1$ and should then guess which one holds

- x_0 and x_1 are characterized by their PMDs p_{x_0}, p_{x_1}
- D is composed of a decision function $g : \mathcal{Y} \mapsto \{0, 1\}$, i.e. $\hat{b} = g(y)$

It is a common situation in security (e.g., intrusion detection, authenticity verification, etc.)

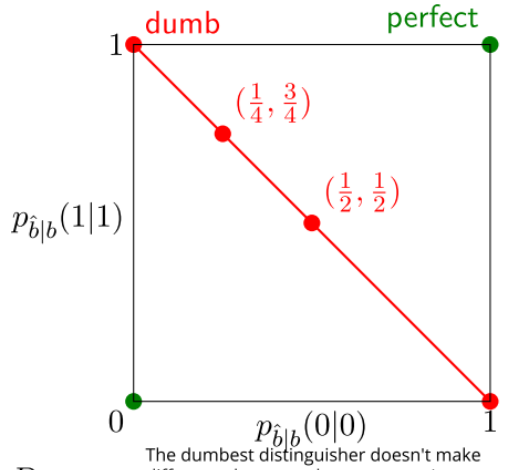
Distinguisher performance

The performance of a distinguisher D is given by the pair of correct decision probabilities

$$\left(p_{\hat{b}|b}(0|0), p_{\hat{b}|b}(1|1)\right)$$

or complementarily by the pair of error probabilities

$$\left(p_{\hat{b}|b}(1|0), p_{\hat{b}|b}(0|1)\right)$$



The dumbest distinguisher doesn't make difference between the two cases. It says "the input is always 0 or its always 1" so its dumb

We define the distinguishability between x_0 and x_1 with D as

$$d_D(x_0, x_1) = |p_{\hat{b}|b}(0|0) + p_{\hat{b}|b}(1|1) - 1| = |p_{\hat{b}|b}(1|0) + p_{\hat{b}|b}(0|1) - 1|$$

Note that $d_D(x_0, x_1) = 1$ for a perfect distinguisher while $d_D(x_0, x_1) = 0$ for a dumb distinguisher

even if its always wrong its good for us because we can just invert it

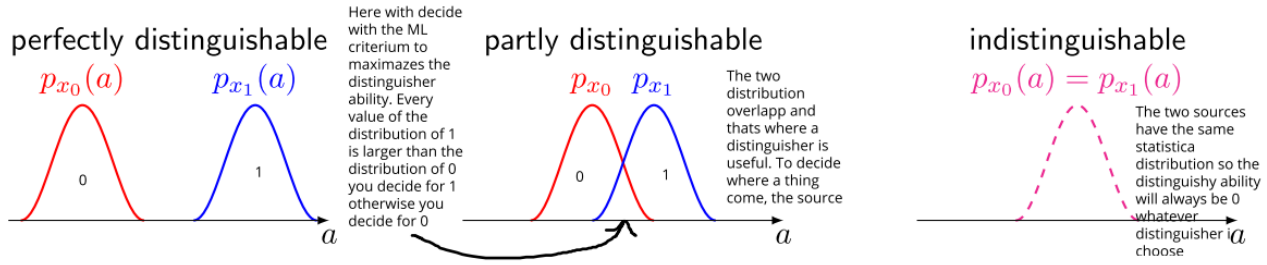
Indistinguishability and statistical distance

It is not always possible to find a perfect or even a good distinguisher

Definition (unconditional)

Two variables x_0 and x_1 are said to be ε-unconditionally indistiguishable if, for any distinguisher D, it is $d_D(x_0, x_1) \leq \varepsilon$

Unconditonal distinguishability is a measure of statistical distance between two variables



The distinguisher that maximizes $d_D(x_0, x_1)$ is the ML estimator of b from observation y

(Maximum likelihood detection)

Risposta:

Variational statistical distance

Definition

The variational distance between two rvs (random variables) x, y with alphabet $\mathcal{A}$ (A is the set of the possible messages) is defined as

$$d_{\mathsf{V}}(x,y) = \frac{1}{2}\sum_{a\in\mathcal{A}} |p_x(a) - p_y(a)|$$

It is a 1-norm distance between their PMD, and it holds

(indistinguishable) $\quad 0 \le d_{\mathsf{V}}(x,y) \le 1 \quad$ (perfectly distinguishable)

Relationship with distinguishability

$$\sup_D d_D(x,y) = d_{\mathsf{V}}(x,y)$$

4)Can you describe the scheme protocols in 2G mobile? 2G è GSM (primo appello scritto)

Security services

GSM security was designed to provide the following services:

user privacy against attackers trying to identify and/or trace a specific user's location

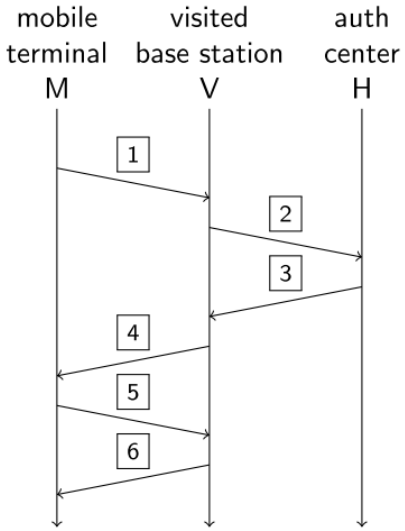
access control against network usage by unauthorized entities

user authentication against billing frauds

user data secrecy against eavesdropping on the radio channel

. . . no data integrity protection

GSM mobile user authentication protocol



1 $\mathsf{M} \to \mathsf{V}$: $\mathrm{id}_{\mathsf{M}}, \mathrm{id}_{\mathsf{H}}$

2 $\mathsf{V} \to \mathsf{H}$: $\mathrm{id}_{\mathsf{M}}, \mathrm{id}_{\mathsf{V}}$

3 H : for $n = 1, \ldots, N$:

generate random challenge (128 bit) $c_n \sim \mathcal{U}(\mathcal{C})$

compute expected response (32 bit) $\hat{r}_n = A3(k_{\mathsf{M}}, c_n)$

compute session key (64 bit) $\hat{k}'_n = A8(k_{\mathsf{M}}, c_n)$

$\mathsf{H} \to \mathsf{V}$: $[c_1, \hat{r}_1, \hat{k}'_1, \ldots, c_N, \hat{r}_N, \hat{k}'_N]$

4 $\mathsf{V} \to \mathsf{M}$: c_1

5 M : compute response $r_1 = A3(k_{\mathsf{M}}, c_1)$

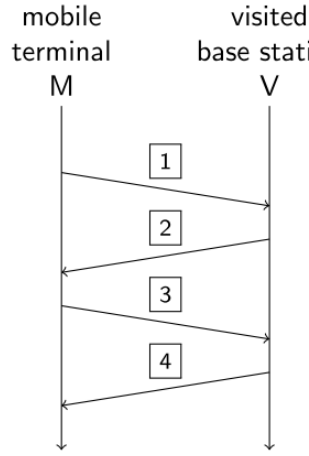
compute session key $k'_1 = A8(k_{\mathsf{M}}, c_1)$

$\mathsf{M} \to \mathsf{V}$: r_1

6 V : if $r_1 = \hat{r}_1$, accept M and generate temporary $\mathrm{id}'_{\mathsf{M},1}$

$\mathsf{V} \to \mathsf{M}$: $[\mathrm{id}_{\mathsf{V}}, \mathrm{id}'_{\mathsf{M},1}]$

With the same VLR V:



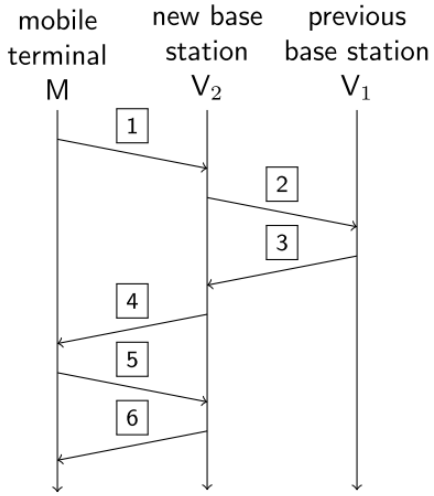
1 $\mathsf{M} \to \mathsf{V}$: $\mathrm{id}'_{\mathsf{M},n}, \mathrm{id}_{\mathsf{V}}$

2 $\mathsf{V} \to \mathsf{M}$: c_{n+1}

3 M : compute $r_{n+1} = A3(k_{\mathsf{M}}, c_{n+1})$ and $k'_{n+1} = A8(k_{\mathsf{M}}, c_{n+1})$
$\mathsf{M} \to \mathsf{V}$: r_{n+1}

4 V : if $\hat{r}_{n+1} = \hat{r}_{n+1}$, accept M and generate temporary $\mathrm{id}'_{\mathsf{M},n+1}$
$\mathsf{V} \to \mathsf{M}$: $[\mathrm{id}_{\mathsf{V}}, \mathrm{id}'_{\mathsf{M},n+1}]$

Handover from a VLR V_1 to another VLR V_2:



1 $\mathsf{M} \to \mathsf{V}_2$: $\mathrm{id}'_{\mathsf{M},n}, \mathrm{id}_{\mathsf{V}_1}$

2 $\mathsf{V}_2 \to \mathsf{V}_1$: $\mathrm{id}'_{\mathsf{M},n}, \mathrm{id}_{\mathsf{V}_1}$

3 $\mathsf{V}_1 \to \mathsf{V}_2$: $\mathrm{id}'_{\mathsf{M},n}, \mathrm{id}_{\mathsf{M}}, [c_{n+1}, \hat{r}_{n+1}, \hat{k}'_{n+1}, \ldots, c_N, \hat{r}_N, \hat{k}'_N]$

4 $\mathsf{V}_2 \to \mathsf{M}$: c_{n+1}

5 M : compute $r_{n+1} = A3(k_{\mathsf{M}}, c_{n+1})$ and
$k'_{n+1} = A8(k_{\mathsf{M}}, c_{n+1})$
$\mathsf{M} \to \mathsf{V}_2$: r_{n+1}

6 V_2 : if $r_{n+1} = \hat{r}_{n+1}$, accept M
and generate temporary $\mathrm{id}'_{\mathsf{M},n+1}$
$\mathsf{V}_2 \to \mathsf{M}$: $[\mathrm{id}_{\mathsf{V}_2}, \mathrm{id}'_{\mathsf{M},n+1}]$

5)What are the limitations of 2G? What kind of information

The GSM cipher A5/1: vulnerabilities

Specific vulnerabilities

- State update of A5/1 is not one-to-one
- Long time with the same BSC, states will concentrate
- 64 bit key / state are too short

→

Biased birthday state guessing attack

1. precompute the 64-bit outputs that correspond to the most likely states
2. observe until any of them appears in the actual transmission
3. the state is known

The security level of k'_{M} was initially set to 54 bits (with 10-bit zero padding), then extended to 64 actual bits

GSM security vulnerabilities

In the authentication protocol

- No authentication of V to M $\Rightarrow$ M will respond to any challenge
- Weakness of the A3 function: for some $c_n = \gamma_i$, r_n leaks information about k_{M}
- A3 is used in a time invariant way

$\rightarrow$

k_{M} recovery attack

- By simulating a fake base station in the vicinity of the victim mobile,
- or by directly accessing the victim SIM (phone resellers, repair shops, ...)

an attacker can submit challenges $\{\gamma_i\}$ and recover k_{M} (aka SIM cloning)

In the security negotiation

- Negotiation of the encryption mechanism (which A5/X) is carried out between V and M without H being aware
- M cannot enforce a minimum security level

$\rightarrow$

Security downgrade attack

- a forged V' can force a low security level (A5/2) or sometimes none (A5/0)

6)What is the secret key rate?

Memoryless sources

Consider n-symbol sequences, $\mathbf{x} = [x_1, \ldots, x_n]$ (and similarly for $\mathbf{y}$ and $\mathbf{z}$) and memoryless sources, $p_{\mathbf{xyz}}(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \prod_{i=1}^{n} p_{xyz}(a_i, b_i, c_i)$

Definition

$R_{\mathsf{k}} \geq 0$ is an achievable secret key rate for the source p_{xyz} if, $\forall n$, there exist: key spaces $\{\mathcal{K}_n\}$ and schemes $f_{\mathsf{A},n}(\cdot,\cdot)$ and $f_{\mathsf{B},n}(\cdot,\cdot)$ such that

cardinality: $|\mathcal{K}_n| \geq 2^{nR_{\mathsf{k}}}$

correctness: $\lim_{n\to\infty} \mathrm{P}\left[k_{\mathsf{A}} \neq k_{\mathsf{B}}\right] = 0$

secrecy: $\lim_{n\to\infty} I(k_{\mathsf{A}}, k_{\mathsf{B}}; \mathbf{z}, c_{\mathsf{A}}, c_{\mathsf{B}}) = 0$

uniformity: $\lim_{n\to\infty} nR_{\mathsf{k}} - H(k_{\mathsf{A}}) = 0$

Definition

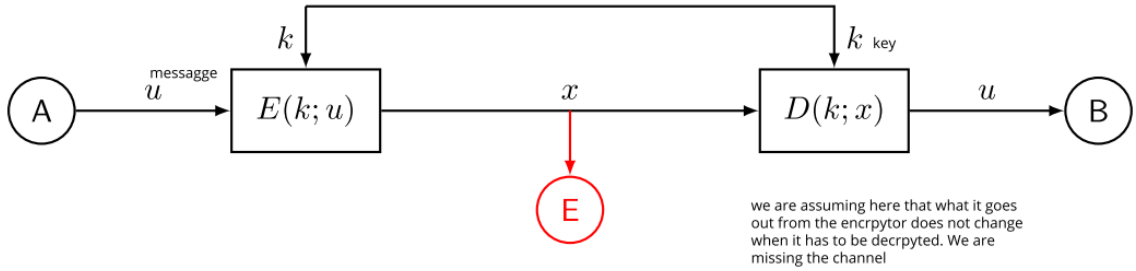
The secret key capacity of the memoryless source p_{xyz} is

$$C_{\mathsf{k}} = \sup\{R_{\mathsf{k}} \ : \ R_{\mathsf{k}} \text{ is an achievable secret key rate}\}$$

7)What is the best layer to apply a secret? Physical layer? Why is physical layer the best?

Penso di sì per questo:

Unconditional secrecy [Shannon, '49]



Kerchoff's Assumption

E knows:

- the functions $C(\cdot;\cdot)$, $D(\cdot;\cdot)$
- the distributions $p_u(\cdot)$, $p_k(\cdot)$

Secrecy of u is only based on hiding the key k

Perfect secrecy

u statistically independent of x

$p_u(\alpha) = p_{u|x}(\alpha|\beta) \quad , \quad I(u;x) = 0$

Theorem

Perfect secrecy requires $H(k) \geq H(u)$

8)Can you give an example why we use to protect application layer data?

5G?

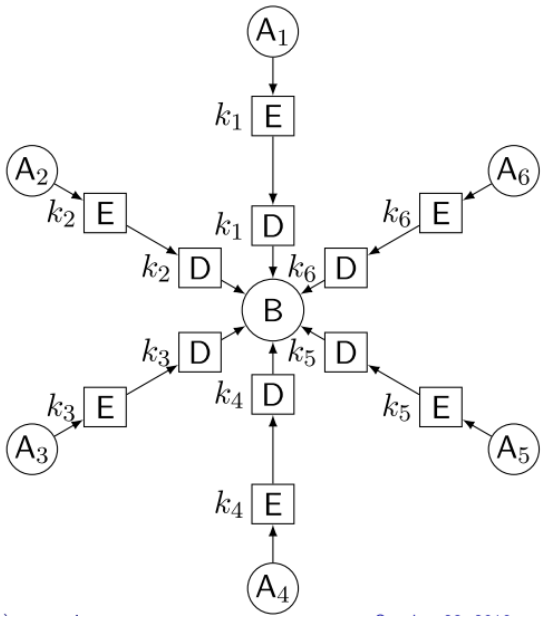
9)What is public key encryption? What are the requirements for the public key?

TLDR: chiave pubblica per tutti per cifrare, chiave privata del proprietario per decifrare. Si utilizza nella firma digitale. Quello che è un requisito fondamentale è il one way function cioè easy to compute e hard to invert(no reversibility). Asymmetric encryption è digital signature

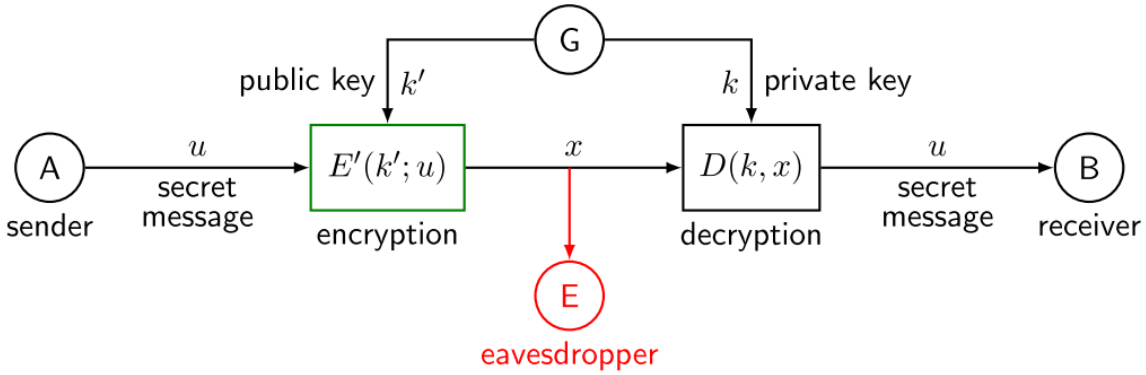
Consider the problem of a single user B having to receive confidential messages $u_1, \ldots, u_N$ from each of N different sources A_i, so that B obtains message u_i but any A_i cannot learn any message u_j, $j \neq i$.

With a symmetric encryption mechanism $(\mathcal{M}, \mathcal{X}, \mathcal{K}, E, D, p_k, p_u)$, B must agree and share a different key k_i with any A_i

Can we build a mechanism where B uses a single key k_B ?



General model of an asymmetric encryption system



private key $k \in \mathcal{K}$ private key space

public key $k' \in \mathcal{K}'$ public key space

(reparametrized) encryption map $E' : \mathcal{K}' \times \mathcal{M} \mapsto \mathcal{X}$

$E_{k'} : \mathcal{M} \mapsto \mathcal{X} \quad E_{k'}(u) \doteq E(k', u)$

decryption map $D : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{M}$

$D_k : \mathcal{X} \mapsto \mathcal{M} \quad D_k(x) \doteq D(k, x)$

Keys are random with joint probability mass distribution $p_{kk'} : \mathcal{K} \times \mathcal{K}' \mapsto [0, 1]$
typically $(k, k') \not\sim \mathcal{U}(\mathcal{K} \times \mathcal{K}')$ are uniform but not independent
often $k \sim \mathcal{U}(\mathcal{K})$ is random and uniform, $k' = f(k)$ is computed with $f : \mathcal{K} \mapsto \mathcal{K}'$ deterministic
The encryption system is completely specified as:

$$\mathcal{S} = (\mathcal{M}, \mathcal{X}, \mathcal{K}, \mathcal{K}', E', D, p_u, p_{kk'})$$

General assumptions

- (perfect reliability)The receiver must be able to recover the secret message perfectly

$$D_c = E_c^{-1} = (E'_{c'})^{-1} \quad \forall c \in \mathcal{K}, c' \in \mathcal{K}' : p_{kk'}(c, c') > 0 \quad (\text{or } c' = f(c))$$

- (Kerchoff's assumption) The eavesdropper knows the system $\mathcal{S}$ (in particular the maps $E'(\cdot, \cdot)$ and $D(\cdot, \cdot)$)

Where does secrecy come from?

Secrecy can only be computational and is based on the following requirements

1. it is hard to derive k from k' (i.e., f is one-way)
2. it is hard to derive u from (k', x) (i.e., $E'_{k'}$ is one-way)
3. it is hard to derive k from (u, x) (i.e., $D(\cdot, x)$ is one-way)

One-way function: definitions

One-way functions are a fundamental tool in many computationally secure mechanisms and their analysis. They are informally referred to as "easy to compute and hard to invert".

Definition (concrete)

A function $f : \mathcal{X} \mapsto \mathcal{Y}$ is said to be $(\varepsilon_0, T_0; \varepsilon_1, T_1)$-one-way if

(easy to compute) there exists a probabilistic algorithm $\mathtt{A}$ such that

$$\forall x \in \mathcal{X} \quad , \quad \mathrm{P}\left[\{\mathtt{A}[x] \to f(x)\} \cap \{T_{\mathtt{A}} \leq T_1\}\right] \geq 1 - \varepsilon_1$$

(hard to invert) for any probabilistic algorithm $\mathtt{B}$

$$\forall y \in \mathcal{Y}, \forall x \in f^{-1}(y) \quad , \quad \mathrm{P}\left[\{\mathtt{B}[y] \to x\} \cap \{T_{\mathtt{B}} \leq T_0\}\right] \leq \varepsilon_0$$

A deterministic variant for the easy to compute requires that there exists a deterministic algorithm $\mathtt{A}$ such that $T_{\mathtt{A}} \leq T_1$ and $\mathtt{A}[x] \to f(x)$, $\forall x \in \mathcal{X}$

In order to provide an asymptotic definition we introduce a security parameter n

Definition (asymptotic)

A sequence $\{f_n\}, n \in \mathbb{N}$ of functions $f_n : \mathcal{X}_n \mapsto \mathcal{Y}_n$ is one-way if

(easy to compute) $\forall \varepsilon > 0$, there exists a sequence of probabilistic algorithms $\mathtt{A}_n$ and a polynomial $p(\cdot)$ such that

$$\forall n \in \mathbb{N}, \forall x \in \mathcal{X}_n \quad , \quad \mathrm{P}\left[\{\mathtt{A}_n[x] \to f_n(x)\} \cap \{T_{\mathtt{A}_n} \leq p(n)\}\right] \geq 1 - \varepsilon$$

(hard to invert) for any sequence of probabilistic algorithms $\mathtt{B}_n$, and any polynomials $q(\cdot), s(\cdot)$, there is a n_0 such that

$$\forall n > n_0, \forall y \in \mathcal{Y}_n, \forall x \in f_n^{-1}(y) \ , \ \mathrm{P}\left[\{\mathtt{B}_n[y] \to x\} \cap \{T_{\mathtt{B}_n} \leq q(n)\}\right] \leq \frac{1}{s(n)}$$

Deterministic easy to compute requires a sequence of deterministic algorithms $\mathtt{A}_n$ such that $T_{\mathtt{A}_n} \leq p(n)$ and $\mathtt{A}_n[x] \to f_n(x)$, $\forall x \in \mathcal{X}_n$

10)How can we say k prime(public key) is compatible with one another?

???

11)What is a universal hashing function? Why do we need universal hashing function? What is its purpose? (Purpose = MAC)

Universal hashing

For unconditionally secure A+IP, we take $\{T_k(\cdot)\}_{k\in\mathcal{K}}$ to be a ε-almost strongly universal$_2$ family of hashing functions $\mathcal{M} \to \mathcal{T}$, for some parameter $\varepsilon \in (0, 1)$, that is

1. (uniform mapping) $\forall u \in \mathcal{M}, t \in \mathcal{T}$, and with $\mathcal{K}_{u\to t} = \{k \in \mathcal{K} \ : \ T_k(u) = t\}$ it must be
$$|\mathcal{K}_{u\to t}| \le \varepsilon\,|\mathcal{K}|$$
the set of indeces k that map u into t
2. (uniform collisions) $\forall u_1 \neq u_2 \in \mathcal{M}$, and with $\mathcal{K}_{u_1u_2} = \{k \in \mathcal{K} \ : \ T_k(u_1) = T_k(u_2)\}$ it must be
$$|\mathcal{K}_{u_1u_2}| \le \varepsilon\,|\mathcal{K}|$$
3. (uniform pairwise mapping) $\forall u_1 \neq u_2 \in \mathcal{M}$ and $\forall t_1, t_2 \in \mathcal{T}$, it must be
$$|\mathcal{K}_{u_1\to t_1} \cap \mathcal{K}_{u_2\to t_2}| \le \varepsilon\,|\mathcal{K}_{u_1\to t_1}|$$

(actually, property 3 $\Rightarrow$ property 2)

Message authentication codes (MACs)

A Message authentication code (MAC) is a symmetric mechanism providing authentication and integrity protection of the "tag-appending" kind

$$x = (u, t) \quad , \quad t = T(k, u)$$

that offers computational security.

Security requirements

It must be hard for F to find t, given u but not k, possibly even under

known message attacks (KMA) F has observed previous $x_i = (u_i, t_i)$ signed with the same k, i.e. $t_i = T(k, u_i)$

chosen message attacks (CMA) F can choose $u_1, \ldots, u_n \neq u$, have them signed with the same k, i.e. $t_i = T(k, u_i)$ and observe the resulting $x_i = (u_i, t_i)$

Message authentication codes (MACs)

The requirements for $T(\cdot,\cdot)$ are similar to those for the decryption function $D(\cdot,\cdot)$ in a symmetric encryption scheme

Can we use a well designed $D : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{M}$ as tag computation function $T : \mathcal{K} \times \mathcal{M} \mapsto \mathcal{T}$?

There is a difference:

- in encryption/decryption, $|\mathcal{X}| \geq |\mathcal{M}|$ because each E_k is injective, and typically $|\mathcal{X}| \approx |\mathcal{M}|$
- in MACs, $H(t)$ and $|\mathcal{T}|$ are dictated by the target security level, and nearly independent of $H(u)$, $|\mathcal{M}|$

We could use as T_k a block cipher decryptor with output range $\mathcal{T}$, but the authentic message u may be too long / short.

Two possible solutions

CBC-MAC pad or split u into blocks $u_1, \ldots, u_n$ of proper length, then compute their tags

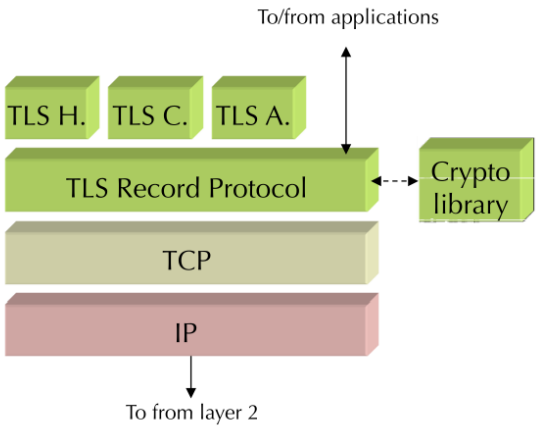
Hash & Sign compress u with a hash function $h : \mathcal{M} \mapsto \mathcal{T}$ before computing the tag

12)How does the handshake work in TLS? What is its purpose?

The purpose is to create a secure connection with server and choose system keys.

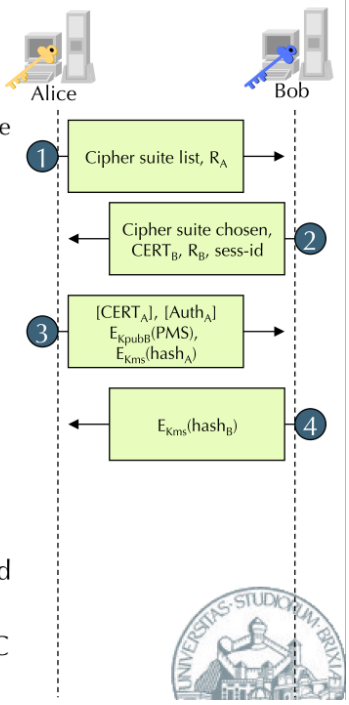
The TLS architecture
TLS Handshake Protocol

- Protocol used to manage security all parameters, such as cipher suite, ephemeral keys, etc., and handle authentication
- Deriving ephemeral keys is expensive, so TLS introduces the concept of ***session***, over which multiple ***connections*** can be setup

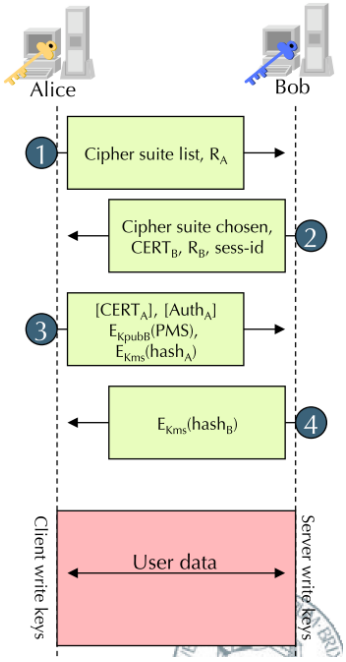


TLS Handshake protocol: high level overview

- Goals
  - Authentication of B to A and, optionally, of A to B
  - Setup of ephemeral ***session*** key K_{ms} (*Master Secret*), as a "seed" for one or more ***connection*** keys
- In TLS, A (*initiator*) is the ***client***, while B (*responder*) is the ***server***
- The Handshake protocol messages are transported by the TLS Record protocol
- Each record message can contain more than one handshake messages
  - For example, record message (2) usually contains messages "server_hello, certificate, [certificate_request], server_hello_done"
- R_A, R_B: random numbers
- $Auth_A = SIG_{KprivA}(hash(previous\ messages))$
- PMS: pre-master secret, random number chosen by A
- $K_{ms} = f_{PMS}(R_A, R_B)$, where *f* is a MAC function derived from both SHA1 and MD5
- $hash_{A/B} = g_{Kms}(\{client/server\}, previous\ messages)$, where *g* is another MAC function derived from both SHA1 and MD5



- At this point we have a ***TLS session***
  - Authentication [optionally mutual]
    - A authenticates B by the fact that B can prove they are able to decrypt PMS (i.e., calculate the correct K_{ms}) with the private key associated to $CERT_B$
    - Optionally, B authenticates A through $Auth_A$
  - TLS session, identified by (K_{ms}, sess-id). K_{ms} is a 384 bit (48 byte) string
- One or more ***TLS connections*** can now be instantiated, by deriving the necessary ephemeral keys from K_{ms}. User traffic will be protected by these TLS connections inside the TLS record protocol
- By key expansion, ***three key pairs*** are derived from K_{ms}
  - Client write MAC, client write, client IV (K_{cm}, K_c, IV_c)
  - Server write MAC, server write, server IV (K_{sm}, K_s, IV_s)
  - Key expansion is similar to the one used to derive K_{ms} from PMS, i.e., it is based on a MAC function that works on K_{ms}, R_A, R_B
  - Note that there are three keys ***for each of the two directions***



TLS Handshake protocol: notes

- ❑ PFS is optional: K_{ms} can optionally be derived from a PMS=DH key
- ❑ CA hierarchies
  - When A or B send a certificate, they can also include ***chains of (CA) certificates***
  - When B request that A authenticates with their certificate, usually B includes in the request a list of ***known and admissible CAs***

13)What is the difference between Digital signature and message authentication code? What services do they provide?

These types of cryptographic primitive can be distinguished by the security goals they fulfill (in the simple protocol of "appending to a message"):

- **Integrity:** Can the recipient be confident that the message has not been accidentally modified?
- **Authentication:** Can the recipient be confident that the message originates from the sender?
- **Non-repudiation:** If the recipient passes the message and the proof to a third party, can the third party be confident that the message originated from the sender?

	Hash	MAC	Digital signature
Integrity	Yes	Yes	Yes
Authentication	No	Yes	Yes
Non-repudiation	No	No	Yes
Kind of keys	None	Symmetric	Asymmetric

A (unkeyed) hash of the message, if appended to the message itself, only protects against accidental changes to the message (or the hash itself), as an attacker who modifies the message can simply calculate a new hash and use it instead of the original one. So this only gives integrity. If the hash is transmitted over a different, protected channel, it can also protect the message against modifications. This is sometimes be used with hashes of very big files (like ISO-images), where the hash itself is delivered over HTTPS, while the big file can be transmitted over an insecure channel.

A message authentication code (MAC) (sometimes also known as keyed hash) protects against message forgery by anyone who doesn't know the secret key (shared by sender and receiver).This means that the receiver can forge any message – thus we have both integrity and authentication (as long as the receiver doesn't have a split personality), but not non-repudiation. Also an attacker could replay earlier messages authenticated with the same key, so a protocol should take measures against this (e.g. by including message numbers or timestamps). (Also, in case of a two-sided conversation, make sure that either both

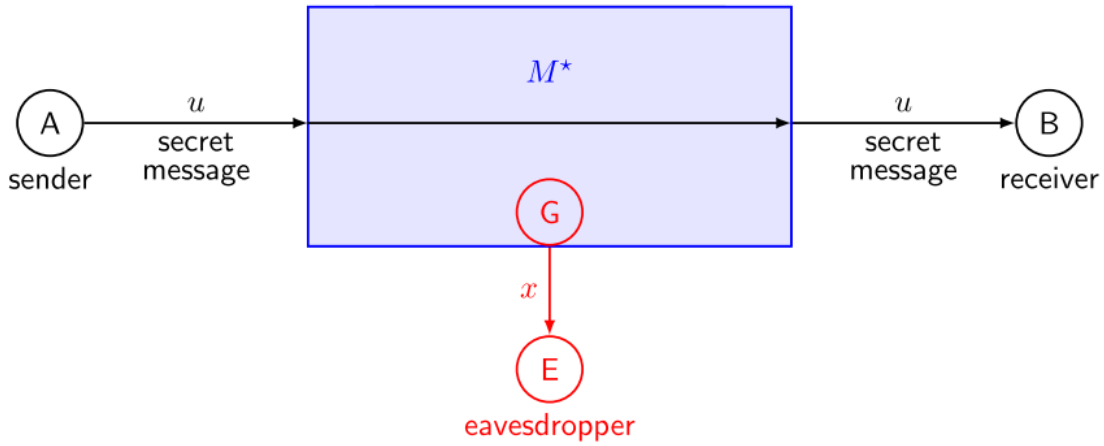
sides have different keys, or by another way make sure that messages from one side can't sent back by an attacker to this side.) MACs can be created from unkeyed hashes (e.g. with the HMAC construction), or created directly as MAC algorithms.

A (digital) signature is created with a private key, and verified with the corresponding public key of an asymmetric key-pair. Only the holder of the private key can create this signature, and normally anyone knowing the public key can verify it. Digital signatures don't prevent the replay attack mentioned previously. There is the special case of designated verifier signature, which only ones with knowledge of another key can verify, but this is not normally meant when saying "signature".So this provides all of integrity, authentication, and non-repudiation.

14)What is the difference between message source authentication and entity source authentication?

15)What is the definition of perfect secrecy? Why is it indipendent?

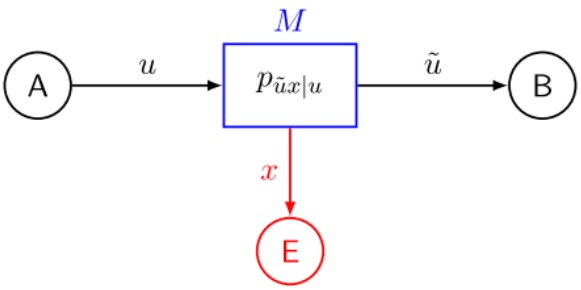
Ideal world model



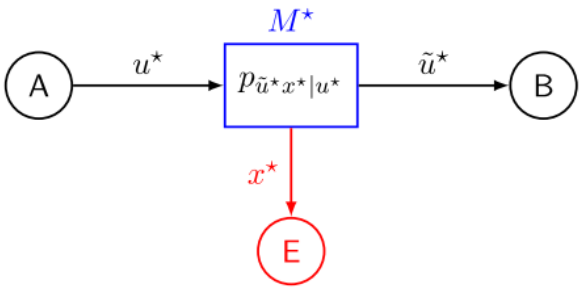
In the ideal counterpart of encryption, the secret message u is directly delivered, unmodified to B, and the message observed by E is generated independently from u

The best we can hope for, is an encryption system M that is statistically identical to its ideal counterpart $M^\star$

real



ideal



$$
\begin{aligned}
p_{\tilde{u}x|u}(b,c|a) &= p_{\tilde{u}^\star x^\star|u^\star}(b,c|a) \\
(\text{independence}) &= p_{\tilde{u}^\star|u^\star}(b|a)p_{x^\star}(c) \\
(\text{correctness}) &= \delta(a,b)p_{x^\star}(c)
\end{aligned}
$$

Definition

An encryption system is perfect if it provides 0-unconditional security based on indistinguishability, i.e. the plaintext is statistically independent of the ciphertext

$$p_{x|u}(b|a) = p_x(b) \quad \forall a \in \mathcal{M}, b \in \mathcal{X}$$

or equivalently

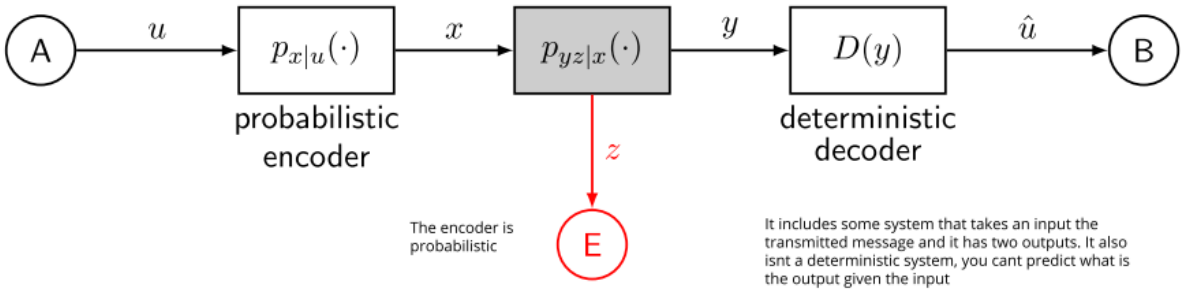
$$p_{ux}(a,b) = p_u(a)p_x(b) \quad \forall a \in \mathcal{M}, b \in \mathcal{X}$$
$$p_{u|x}(a|b) = p_u(a) \quad \forall a \in \mathcal{M}, b \in \mathcal{X}$$

In a system with perfect secrecy, since $p_{u|x} = p_u$ the optimal informed guessing strategy coincides with the optimal ignorant guessing

Indipendent?

16)What is a wiretap channel in physical layer?

The wiretap channel [Wyner, '75]



We aim for reliable transmissions to B, and secrecy with respect to E

perfect reliability: $\hat{u} = u$

perfect secrecy: z, u statistically independent

The attacker can see only z and z must be indipendent from the message x

In terms of unconditional distinguishability from the ideal counterpart

reliability is measured by the error probability $\mathrm{P}\left[\hat{u} \neq u\right] = d_{\mathsf{V}}(p_{\hat{u}|u}, p_{u|u})$

secrecy is measured by the mutual information $I(u;z) = \mathrm{D}\left(p_{uz} \| p_u p_z\right)$

17)How do we solve a problem in wiretap channel?

???

18)How do we order the wiretap channel?

19)How does McElice Cryptosystem work in encryption?

The McEliece cryptosystem [McEliece, '78]

Based on NP problem

minimum Hamming distance (mHd) decoding of binary codes

In a (n, ℓ, t) linear binary FEC code (e.g., Goppa codes) with

- n codeword length
- ℓ code dimension = information word length
- t maximum nr. of correctable errors

easy given an information word $\boldsymbol{b} \in \mathbb{B}^\ell$ and a generating matrix $\boldsymbol{G} \in \mathbb{B}^{n \times \ell}$, compute the codeword $\boldsymbol{c} = \boldsymbol{G}\boldsymbol{b} \in \mathbb{B}^n$

hard given a received word (not necessarily a codeword) $\tilde{\boldsymbol{c}} \in \mathbb{B}^n$ and a generating matrix $\boldsymbol{G} \in \mathbb{B}^{n \times \ell}$, compute $\hat{\boldsymbol{b}} = \arg\min_{\boldsymbol{\beta} \in \mathbb{B}^\ell} d_{\mathsf{H}}(\tilde{\boldsymbol{c}}, \boldsymbol{G}\boldsymbol{\beta})$

easy given a received word (not necessarily a codeword) $\tilde{\boldsymbol{c}} \in \mathbb{B}^n$ and a generating matrix $\boldsymbol{G} \in \mathbb{B}^{n \times \ell}$ in canonical form, compute $\hat{\boldsymbol{b}} = \arg\min_{\boldsymbol{\beta} \in \mathbb{B}^\ell} d_{\mathsf{H}}(\tilde{\boldsymbol{c}}, \boldsymbol{G}\boldsymbol{\beta})$

The McEliece cryptosystem

Key generation

1. B chooses $\boldsymbol{G} \in \mathbb{B}^{n \times \ell}$ canonical generating matrix of a (n, ℓ, t) Goppa code
2. generates $\boldsymbol{S} \in \mathbb{B}^{\ell \times \ell}$ non singular
3. generates $\boldsymbol{P} \in \mathbb{B}^{n \times n}$ a permutation matrix (exactly one '1' in each row and column)
4. computes $\boldsymbol{S}^{-1}, \boldsymbol{P}^{-1}$, and $\boldsymbol{G}' = \boldsymbol{P}^{-1}\boldsymbol{G}\boldsymbol{S}^{-1} \in \mathbb{B}^{n \times \ell}$ noncanonical generating matrix of an equivalent (n, ℓ, t) Goppa code

private key $k = (\boldsymbol{G}, \boldsymbol{P}, \boldsymbol{S})$, $\mathcal{K} = \mathbb{B}^{n \times \ell} \times \mathbb{B}^{n \times n} \times \mathbb{B}^{\ell \times \ell}$

public key $k' = f(k) = (\boldsymbol{G}', t)$, $\mathcal{K}' = \mathbb{B}^{n \times \ell} \times \mathbb{N}$

The McEliece cryptosystem

Encryption by A (public key, probabilistic)

$$\mathcal{M} = \mathbb{B}^\ell \quad , \quad \mathcal{X} = \mathbb{B}^n$$

A generates a random $\boldsymbol{e} \in \mathbb{B}^n$ such that $w_{\mathsf{H}}(\boldsymbol{e}) \leq t$ (i.e., a correctable error pattern)

$$E'_{k'} \; : \; \boldsymbol{x} = \boldsymbol{G}'\boldsymbol{u} + \boldsymbol{e}$$

Decryption by B (private key)

$\hat{u} = D(k, x) = D(\boldsymbol{G}, \boldsymbol{P}, \boldsymbol{S}, x)$ is computed as follows

1. B computes $\boldsymbol{x}' = \boldsymbol{P}\boldsymbol{x}$
2. solves the mHd decoding of $\boldsymbol{x}'$ in the Goppa code with canonical $\boldsymbol{G}$, i.e.,
$$\boldsymbol{u}' = \arg\min_{\boldsymbol{\beta} \in \mathbb{B}^\ell} d_{\mathsf{H}}(\boldsymbol{x}', \boldsymbol{G}\boldsymbol{\beta})$$
3. computes $\hat{\boldsymbol{u}} = \boldsymbol{S}\boldsymbol{u}'$

The McEliece cryptosystem

Correctness

We prove that $\hat{\boldsymbol{u}} = \boldsymbol{u}$

$$\begin{aligned} \boldsymbol{x}' &= \boldsymbol{P}\boldsymbol{x} \\ &= \boldsymbol{P}(\boldsymbol{G}'\boldsymbol{u} + \boldsymbol{e}) \\ &= \boldsymbol{P}\boldsymbol{P}^{-1}\boldsymbol{G}\boldsymbol{S}^{-1}\boldsymbol{u} + \boldsymbol{P}\boldsymbol{e} \\ &= \boldsymbol{G}\boldsymbol{S}^{-1}\boldsymbol{u} + \boldsymbol{e}' \\ &= \boldsymbol{G}\boldsymbol{u}' + \boldsymbol{e}' \end{aligned}$$

where $\boldsymbol{u}' = \boldsymbol{S}^{-1}\boldsymbol{u}$ is an information word, too, and $\boldsymbol{e}' = \boldsymbol{P}\boldsymbol{e}$ has $w_{\mathsf{H}}(\boldsymbol{e}') = w_{\mathsf{H}}(\boldsymbol{e}) \leq t$, so it is a correctable error pattern, too.
Therefore the mHd decoding of $\boldsymbol{x}'$ with $\boldsymbol{G}$ is $\boldsymbol{u}'$ and

$$\hat{\boldsymbol{u}} = \boldsymbol{S}\boldsymbol{u}' = \boldsymbol{S}\boldsymbol{S}^{-1}\boldsymbol{u} = \boldsymbol{u}$$

The McEliece cryptosystem

Security

$x = E'_{k'}(u)$ is one-way given x and noncanonical G', it is hard to find u (mHd decoding)
$k' = f(k)$ is one-way given the non canonical $\boldsymbol{G}' = \boldsymbol{P}^{-1}\boldsymbol{G}\boldsymbol{S}^{-1}$ it is hard to factor it into $\boldsymbol{P}, \boldsymbol{G}, \boldsymbol{S}$ with a canonical $\boldsymbol{G}$

20)How does Elgamal Cryptosystem work in encryption and in signature? Is Elgamal probalistic? Where does the probabilistic come from?

The Elgamal cryptosystem [Elgamal, '85]

Based on NP problem

finite logarithm

In a group $(\mathbb{G}, \circ)$, we denote $\alpha \overset{n}{\circ} = \underbrace{\alpha \circ \cdots \circ \alpha}_{n \text{ times}}$

easy given $\alpha \in \mathbb{G}, n \in \mathbb{N}$, compute $\beta = \alpha \overset{n}{\circ}$

hard given $\alpha, \beta \in \mathbb{G}$, find $n \in \mathbb{N}$ such that $\alpha \overset{n}{\circ} = \beta$

Key generation

Let $(\mathbb{G}, \circ)$ be a group with a primitive element $\alpha \in \mathbb{G}$, i.e. such that $\forall \beta \in \mathbb{G}, \exists n : \alpha \overset{n}{\circ} = \beta$.

private key space $\mathcal{K} = \{1, \ldots, |\mathbb{G}| - 1\} \subset \mathbb{N}$

public key space $\mathcal{K}' = \mathbb{G}$

Let $(\mathbb{G}, \circ)$ and α be publicly known. B generates $k \sim \mathcal{U}(\mathcal{K})$, then computes $k' = f(k) = \alpha \overset{k}{\circ}$

The Elgamal cryptosystem

Encryption by A (public key, probabilistic)

$\mathcal{M} = \mathbb{G}$, $\mathcal{X} = \mathbb{G}^2$

A generates $b \sim \mathcal{U}(\mathcal{K})$

Unlike RSA the encryption is probabilistic

$$x = E'_{k'}(u, b) = (x_1, x_2) \quad , \quad \begin{cases} x_1 = \alpha \overset{b}{\circ} \\ x_2 = u \circ (k' \overset{b}{\circ}) \end{cases}$$

a operating himself b times

Decryption by B (private key)

B need not know b

$$\hat{u} = D_k(x) = D_k(x_1, x_2) = x_2 \circ \left((x_1 \overset{k}{\circ}) \overset{-1}{\circ} \right)$$

where $\cdot \overset{-1}{\circ}$ denotes the inverse in $(\mathbb{G}, \circ)$

Probabilistico sì e dipende da finite log problem

The finite logarithm problem

A strong requirement for security of the Elgamal encryption is that "exponentiation" $f_\alpha(n) = \alpha \overset{n}{\circ}$ is a one-way function of n and this depends on the choice of the group $(\mathbb{G}, \circ)$.
If computing $\circ$ has linear complexity in $\ell = \log |\mathbb{G}|$, exponentiation can be computed with complexity $O(\ell^2)$, by iterative squaring and multiplying

For a general group $(\mathbb{G}, \circ)$

Consider the computation of $y = x \overset{n}{\circ}$, with $n < |\mathbb{G}|$.
Let $\boldsymbol{b} = [b_0, b_1, \ldots, b_{\ell-1}]$ be the binary representation of n

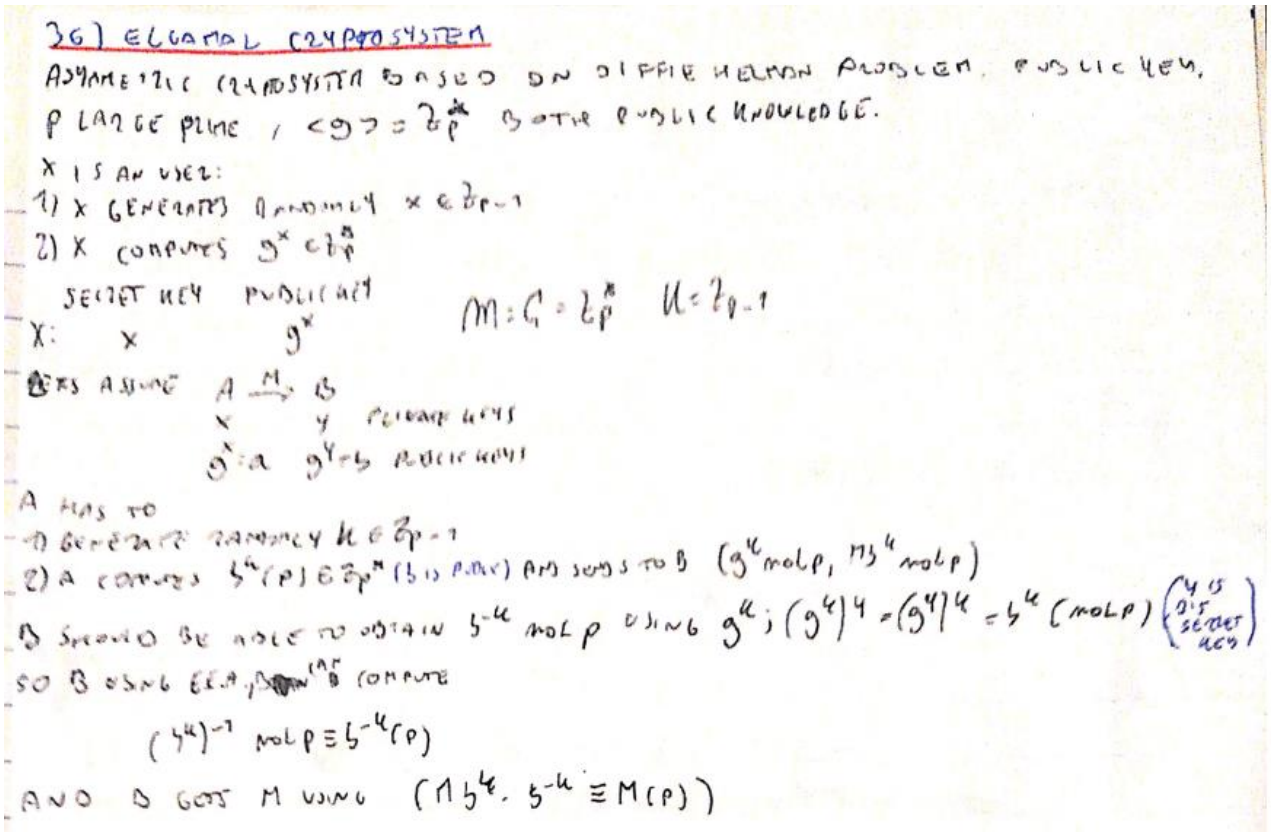
$$n = \sum_{i=0}^{\ell-1} b_i 2^i$$

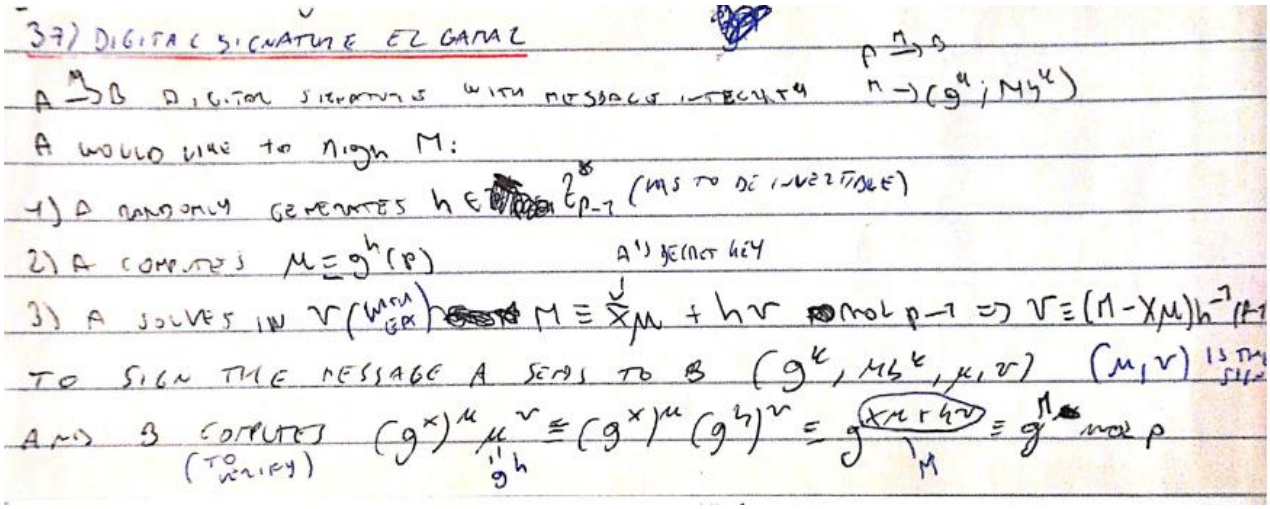
Then $y = x \overset{n}{\circ} = x \overset{\sum_{i=0}^{\ell-1} b_i 2^i}{\circ} = \left(x \overset{b_0 2^0}{\circ}\right) \circ \cdots \circ \left(x \overset{b_{\ell-1} 2^{\ell-1}}{\circ}\right)$

Iterative square and multiply

$c \leftarrow e$ (identity in $\mathbb{G}$)
$a \leftarrow x$
for $i = 0$ to $\ell - 1$ **do**
  if $b_i = 1$ **then**
    $c \leftarrow c \circ a$
  end if
  $a \leftarrow a \circ a$
end for
$y \leftarrow c$

Vedi slide languasco





21)Talk and explain memoryless channels, what results do we have?

Memoryless channels

By considering n-symbol sequences, $\boldsymbol{x} = [x_1, \ldots, x_n]$ (and similarly for $\boldsymbol{y}$ and $\boldsymbol{z}$) and memoryless channels, $p_{\boldsymbol{yz}|\boldsymbol{x}}(\boldsymbol{b}, \boldsymbol{c}|\boldsymbol{a}) = \prod_{i=1}^{n} p_{yz|x}(b_i, c_i|a_i)$, we define:

Definition

$R_s \geq 0$ is an achievable secrecy rate for memoryless channel $p_{yz|x}$ if, $\forall n \geq n_0$, there exist: a message set $\mathcal{M}_n$, an encoder and decoder such that

- $|\mathcal{M}_n| \geq 2^{nR_s}$
- $\lim_{n\to\infty} \mathrm{P}\left[\hat{u} \neq u\right] = 0$
- $\lim_{n\to\infty} I(u; \boldsymbol{z}) = 0$

Definition

The secrecy capacity of memoryless channel $p_{yz|x}$ is

$$C_s = \sup\{R_s : R_s \text{ is an achievable secrecy rate}\}$$

Memoryless channels

Theorem

PMD probability distribution

If there exists some input PMD p_x such that $R_s < I(x;y) - I(x;z)$, then R_s is an achievable secrecy rate for channel $p_{yz|x}$.

Intuition

By fixing p_x and making use of typical sequences, we have as $n \to \infty$

$$|\mathcal{Y}| \to 2^{nH(y)} \quad , \quad N_{y|x} \to 2^{nH(y|x)} \quad , \quad |\mathcal{Z}| \to 2^{nH(z)} \quad , \quad N_{z|x} \to 2^{nH(z|x)}$$

so that, by the hypothesis on R_s, $2^{nR_s} < 2^{n[I(x;y)-I(x;z)]} = 2^{n[H(y)-H(y|x)-H(z)+H(z|x)]}$

It is therefore possible, for a large enough n, to find $\mathcal{M}_n$ such that

more unpredictible the larger is the alphabet

$$2^{nR_s} \leq |\mathcal{M}_n| \leq \frac{2^{nH(y)}}{2^{nH(y|x)}} \frac{2^{nH(z|x)}}{2^{nH(z)}} \approx \frac{|\mathcal{Y}|}{N_{y|x}} \frac{N_{z|x}}{|\mathcal{Z}|}$$

and leverage the uniform channel result for the existence of encoder and decoder.

22)Talk about secret key capacity

Memoryless channels

Consider n-symbol sequences, $\mathbf{x} = [x_1, \ldots, x_n]$ (and similarly for $\mathbf{y}$ and $\mathbf{z}$) and memoryless channels, $p_{\mathbf{yz}|\mathbf{x}}(\mathbf{b}, \mathbf{c}|\mathbf{a}) = \prod_{i=1}^{n} p_{yz|x}(b_i, c_i|a_i)$

Definition

$R_k \geq 0$ is an achievable secret key rate for the memoryless channel $p_{yz|x}$ if there exists a memoryless source p_x such that R_k is an achievable secret key rate for the resulting joint memoryless source p_{xyz}

$$p_{xyz}(a, b, c) = p_{yz|x}(b, c|a)p_x(a)$$

Definition

The secret key capacity of the memoryless channel $p_{yz|x}$ is

$$C_k = \sup \{R_k \ : \ R_k \text{ is an achievable secret key rate}\}$$

Memoryless channels

Theorem

If there exists some input PMD p_x such that $R_k < I(x;y) - I(x;z)$ or $R_k < I(x;y) - I(y;z)$, then R_k is an achievable secret key rate for channel $p_{yz|x}$.

Proof.

Follows from the definition and the corresponding theorem for joint memoryless sources □

Corollary

For a memoryless channel, the secret key capacity satisfies the bounds

$$\max_x \left[I(x;y) - \min\left\{I(x;z), I(y;z)\right\}\right] \leq C_k \leq \max_x \left[\min\left\{I(x;y), I(x;y|z)\right\}\right]$$

23)What do we require in an entity authentication protocol?

General model for entity authentication

An entity A (called the prover) wants to prove his identity to another entity B (called the verifier), typically through an iteractive protocol. At the end of the protocol, the entity B must decide whether he trusts A (accept) or not (reject)

Attack scenario

Masquerade A malicious F wants to pose as A while interacting with B

Requirements

Correctness If A is honest, B accepts hime with high probability

Security / robustness (to false provers) If the prover is not honest (i.e., it is some F posing as A) it is hard for F to be accepted

Non transferability (against malicious verifiers) Even after the protocol has taken place between A and B it is hard for B to pose as A in an exchange with another entity C and be accepted

24)Give an example of an entity that does offer non-transferability and which doesn't

Hash password does not offer non transferability

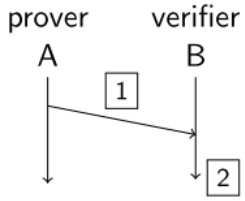
Hashed password authentication protocols

its an improve of the previous one

entities the prover A, the verifier B

tools a hash function $h : \mathcal{W} \mapsto \mathcal{T}$

setup A chooses a password $w_A \in \mathcal{W}$ and securely delivers it to B
B computes $t_A = h(w_A)$ and stores a copy of (id_A, t_A) in database $\mathcal{D}$



1 $A \to B : \ u = (u_1, u_2) = (\mathrm{id}_A, w_A)$

2 B : computes $\tilde{t} = h(u_2)$ and checks if $(u_1, t) \in \mathcal{D}$ and, if so, accepts A

the verify is the hash not the password

Improvement

- w_A no longer stored in clear

Weaknesses

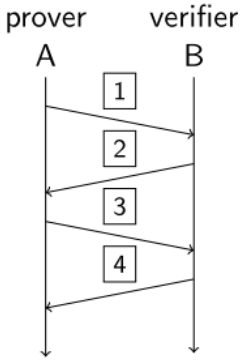
- transferability, as B learns w_A
- w_A still transmitted in clear
- a forger could carry on a 2nd preimage attack

Zero knowledge e challenge response + A + IP

Challenge-response protocols with symmetric A+IP

entities the prover A, the verifier B

tool a symmetric message A+IP mechanism, of the tag appending type with key k_A and tag function $T(\cdot, \cdot)$



1 $A \to B : \ u_1 = \mathrm{id}_A$

2 B : generates a random challenge $r \sim \mathcal{U}(\mathcal{R})$
$B \to A : \ u_2 = r$

3 A : builds $u_3 = \mathrm{id}_A, r$
signs $t_3 = T(k_A, u_3)$
$A \to B : \ t_3$

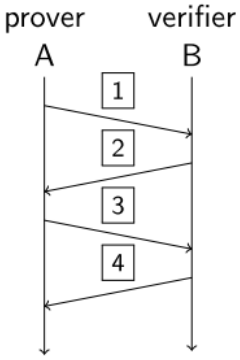
4 B : verifies whether $V(k_A, u_3, t_3) = (r, \mathrm{ok})$ and, if so, accepts A

The challenge r must be changed at every run of the protocol, otherwise a dishonest prover F, pretending to be A, can replay 1 and 3 even without knowing k_A, and would be accepted

Challenge-response protocols with asymmetric A+IP

entities the prover A, the verifier B

tool a digital signature mechanism, with keys k_A, k'_A; a certificate c_A for k'_A and k'_B



1 A → B : $u_1 = \mathrm{id}_A$

2 B : generates a random challenge $r_B \sim \mathcal{U}(\mathcal{R})$
B → A : $u_2 = r_B$

3 A : generates a random challenge $r_A \sim \mathcal{U}(\mathcal{R})$
builds $u_3 = [\mathrm{id}_B, r_B, r_A]$ signs $x_3 = S(k_A, u_3)$
A → B : x_3

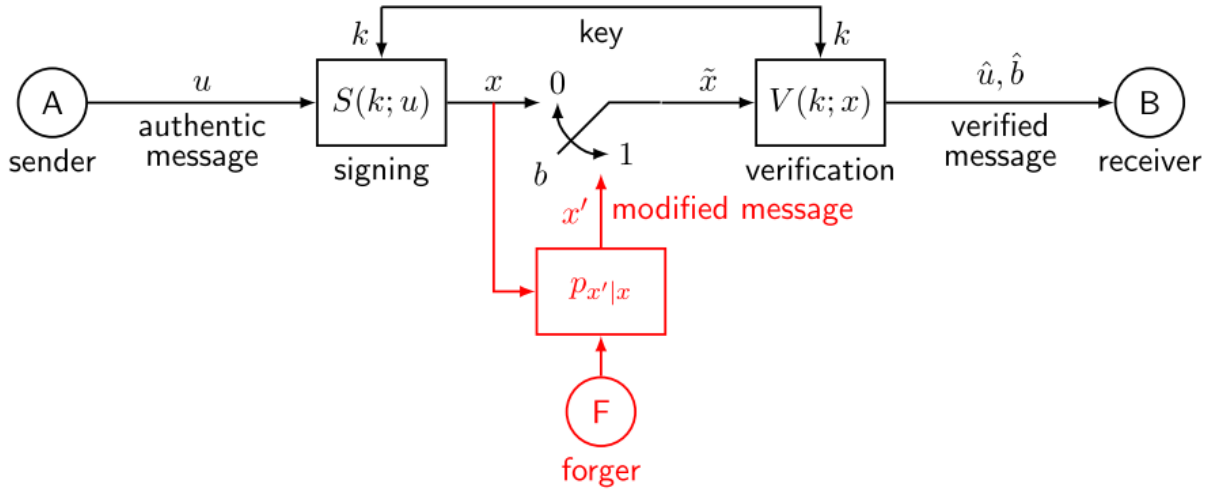
4 B : verifies whether $V(k'_A, x_3) = ([\mathrm{id}_B, r_B, r_A], \mathrm{ok})$ and, if so, accepts A

The challenge r must be changed at every run of the protocol, otherwise an dishonest prover F, pretending to be A, can replay 1 and 3 even without knowing k_A, and would be accepted

25)Talk about integrity protection, what are the requirements for symmetrics encryption?

Integrity protection makes it possible to detect whether a message was intecepted and modified.

General model of the integrity protection problem



Illegitimate modification (alteration) attack

F can block x and wants to replace it with x' such that $\hat{u} \neq u$ and $\hat{b} = 0$

Symmetric encryption is a type of encryption where only one key (a secret key) is used to both encrypt and decrypt. Stessa chiave condivisa che serve per scambiare in maniera sciura (DiffieHellmann). La lunghezza della chiave è related alla robustness.

26)In Digital signature using RSA, how to generate keys? What is needed for public and private keys? How to sign and verify messages?

Based on NP problems

- integer factorization
  - easy given $p, q \in \mathbb{Z}$, compute $n = pq$
  - hard given $n \in \mathbb{Z}$, find $p, q \in \mathbb{Z}$ such that $pq = n$
- finite logarithm and finite root
  - easy given $n \in \mathbb{Z}$, $x, d \in \mathbb{Z}_n$ compute $y = x^d \bmod n$ (finite exponential)
  - hard given $n \in \mathbb{Z}$, $x, y \in \mathbb{Z}_n$ find $d \in \mathbb{Z}_n$ such that $x^d \bmod n = y$
  - hard given $n \in \mathbb{Z}$, $d, y \in \mathbb{Z}_n$ find $x \in \mathbb{Z}_n$ such that $x^d \bmod n = y$

The RSA cryptosystem

Key generation (ℓ-bit)

B chooses $p, q < 2^{\ell/2}$ primes
computes $n = pq$, $\varphi = (p-1)(q-1)$
chooses $d \in \mathbb{Z}_n$ such that $\gcd(\varphi, d) = 1$
computes $e \in \mathbb{Z}_n$ such that $ed = 1 \pmod{\varphi}$

private key $k = (p, q, d)$, $\mathcal{K} = \mathbb{Z}_{2^\ell}^3$
public key $k' = (n, e)$, $\mathcal{K}' = \mathbb{Z}_{2^\ell}^2$

Encryption by A (public key)

$\mathcal{M} = \mathcal{X} = \mathbb{Z}_n$
$E' : \mathcal{K}' \times \mathcal{M} \mapsto \mathcal{X}$
$x = E'(k', u) = E'(n, e, u) = u^e \bmod n$

Decryption by B (private key)

$D : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{M}$
$\hat{u} = D(k, x) = D(n, d, x) = x^d \bmod n$

The RSA signature scheme

Necessity of hashing

Besides existential forgery, hashing also prevents the following chosen message attack

- F knows one pair $x = (t, u)$ with $t = T(k, u) = u^d \bmod n$ and aims to forge $x' = (u', t')$, for some target forged message u', with $t' = T(k, u') = (u')^d \bmod n$.
- he computes $u_1 = u'u \bmod n$ and lures A into signing u_1, obtaining

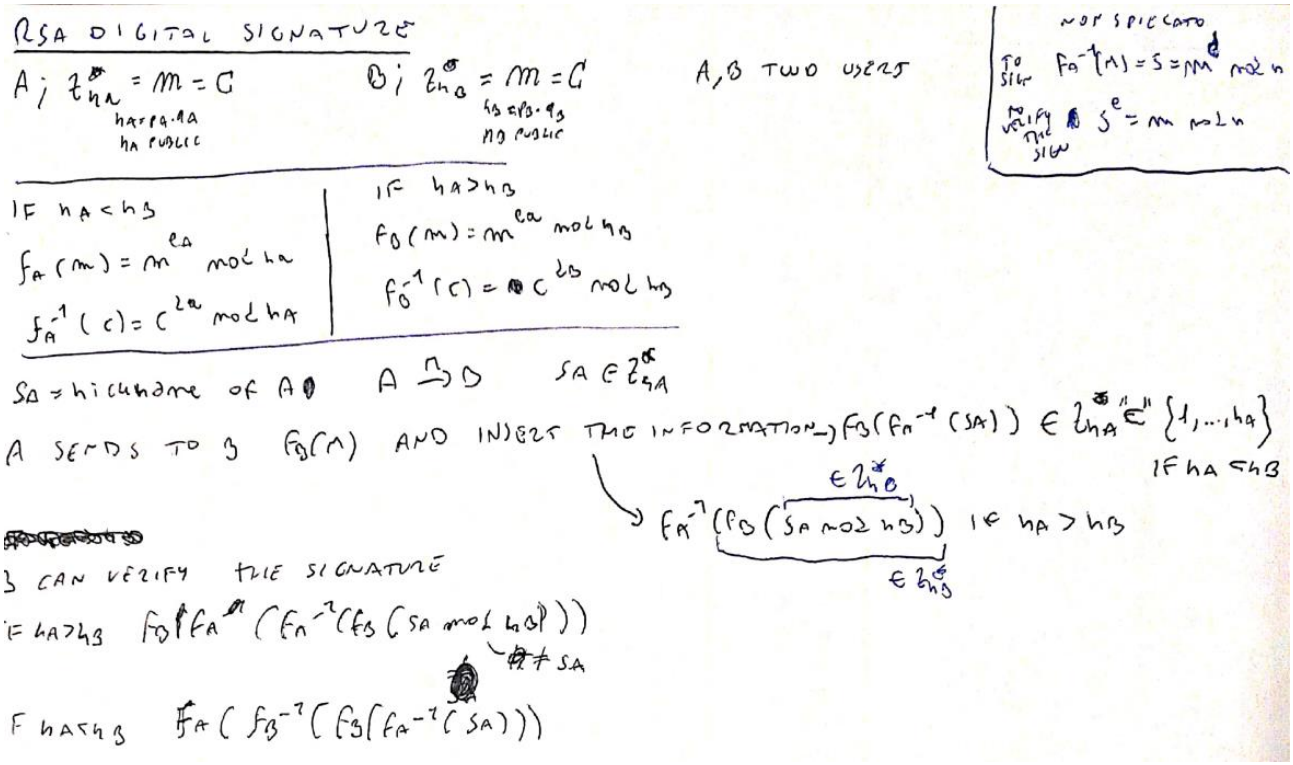
$$t_1 = (u_1)^d = (u'u)^d = (u')^d u^d = t't \pmod{n}$$

- he derives $t' = t_1 t^{-1} \bmod n$

With hashing the attacker can compute

$$h(u')h(u) \bmod n = v'v \bmod n = v_1 = h(u_1)$$

but can not derive u_1 (preimage reistance)



Mancano un po' di cose, sicuramente qualcosa sul **distance bounding (lecture 20)** perché è stato chiesto al primo appello quindi ci fa qualche domanda.

Distance bounding cryptographic protocols

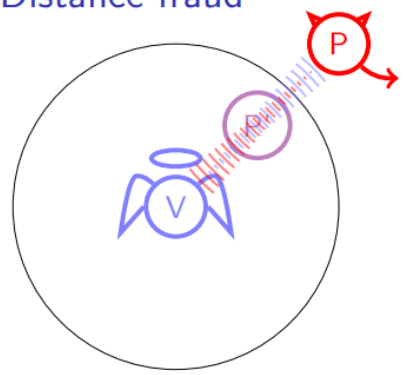
General formulation

1. preliminary setup: sharing a long-term cryptographic secret
2. initialization phase: sharing a temporary session key
3. timing phase: many (single bit) challenge-response iterations
4. verification phase: checking answers vs key, and times vs distance bound

- very short response process times (~ 1 ns)
- cryptographic computations at initialization

Verifier $\mathcal{V}$		Prover $\mathcal{P}$
$x_{\mathcal{VP}}$		$x_{\mathcal{VP}}$
Initialization phase		
$N_{\mathcal{V}} \leftarrow \{0,1\}^m$	$\xrightarrow{N_{\mathcal{V}}}$	$N_{\mathcal{P}} \leftarrow \{0,1\}^m$
	$\xleftarrow{N_{\mathcal{P}}}$	
$a_0 = f_{x_{\mathcal{VP}}}(N_{\mathcal{V}}, N_{\mathcal{P}})$		$a_0 = f_{x_{\mathcal{VP}}}(N_{\mathcal{V}}, N_{\mathcal{P}})$
$a_1 = \mathsf{Enc}_{a_0}(x_{\mathcal{VP}})$		$a_1 = \mathsf{Enc}_{a_0}(x_{\mathcal{VP}})$
Distance-bounding phase		
for $i = 1, \ldots n$		
pick $c_i \in \{0,1\}$		
Start Clock	$\xrightarrow{c_i}$	if $c_i \notin \{0,1\}$, halt
	$\xleftarrow{r_i}$	else $r_i = (a_{c_i})_i$
Stop Clock		
Verification phase		
Check that $\Delta t_i < t_{\mathsf{max}}$ $\forall i = 1, \ldots, n$		
Verify r_i		

Distance fraud



scenario a dishonest prover P is far from the verifier V

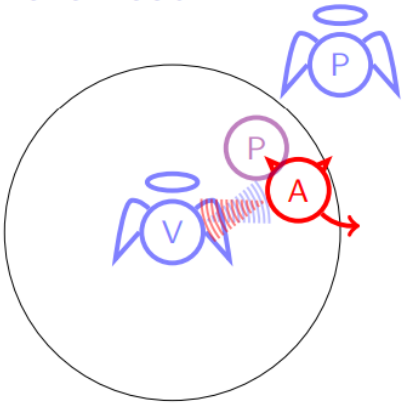
aim P attempts to convince V that P is close to V

attack P reduces the response computation time, or guesses the challenge in advance

Requirements against distance fraud

- The response computation time must be $\tau_c \leq (d' - d)/c \ll t_{\max}$
- The response must depend on the challenge, which must be unpredictable $c_i \sim U(\{0, 1\})$
- The challenge and response waveforms must have short duration

Mafia fraud



scenario an honest prover P is far from the verifier V
a malicious attacker A is close to V

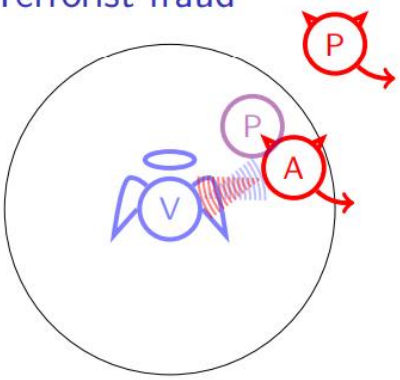
aim A attempts to convince V that P is close to V

attack A intercepts c_i and answers r_i in place of P
or forwards c_i to P and uses P as an oracle

Requirements against mafia fraud

- The response must depend on some secret shared between P and V
- The shared secret must be derived from P's credentials
- The shared secret must be renewed in each session

Terrorist fraud



scenario a dishonest prover P is far from the verifier V
a malicious attacker A is close to V

aim P and A collude and attempt to convince V that P is close to V
without P giving his/her long term credentials to A (otherwise it would be unstoppable)

attack P shares with A his/her temporary secret, so A can compute r_i and answer immediately

Requirements against terrorist fraud

- The response must depend on some secret shared between P and V
- The shared secret must leak out significant information from P's credentials (but the pairs (c_i, r_i) must not)

Altre cose:

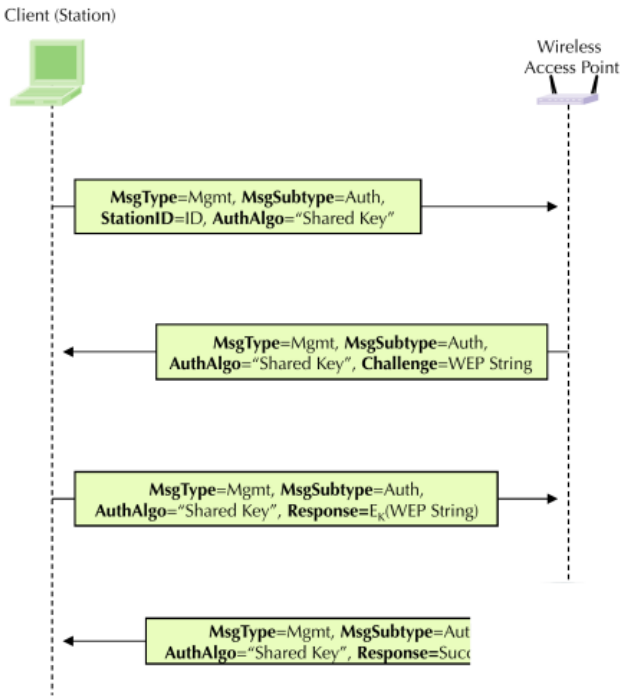
Wireless LAN (Lecture 22)

WEP: goals and main components

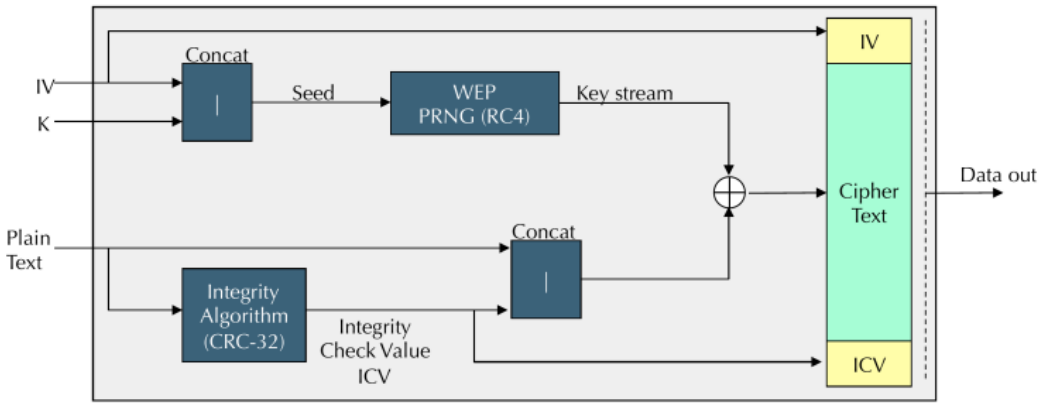
- Security at layer-2: cryptographic protection of most of the fields of pseudo-ethernet frames
- Goals
  - Authentication
  - Confidentiality
  - Integrity protection
- Main components
  - A pre-shared key *K*, shared among all APs and STAs that make up a WLAN: security goals are "shared" within a (potentially large and dynamic) group
    - In theory the standard specifies the possibility of using four different pre-shared keys, thus dividing users and APs in four different sub-groups
    - The same key is used for authentication and confidentiality
  - A very simple authentication protocol
  - A single cryptographic primitive: RC4
  - A non-crypto primitive for integrity protection: CRC-32

WEP: authentication protocol

- Authentication is one way (client to network)
- STA proves knowledge of *K* by encrypting the challenge with RC4
- As we will see, there are several security pitfalls to this approach
  - Authentication is not mutual
  - The way RC4 is used in WEP makes the authentication protocol leak security information that should not be exposed
- The protocol is easily breakable (remember: RC4, if used properly, ***is not***)



The use of RC4 in WEP



- K: the pre-shared key between all APs and STAs in a given WLAN
  - 40 or 104 bit
- IV: 24 bit
  - To generate different key-streams, the IV should be different for each packet [think about what would happen if an intruder could get a hold of c_1 (with $c_1=k_1 \oplus m_1$) and c_2 (with $c_2=k_2 \oplus m_2$), with $\mathbf{k_1=k_2}$]
- ICV: CRC-32, 32 bit (non-crypto, linear function)

The main problems with WEP

- K is a ***group key***
  - The system is not scalable
  - Key management is nearly impossible: what happens when a client leaves the group for good?
  - All clients are equal WRT authentication (same key)
- RC4 is used incorrectly
  - WEP design choices
    - Wireless devices ⇒ reduced computing power ⇒ stream cipher
    - Problem: how do we keep transmitter and receiver in sync
    - How does WEP address the issue: ***different (random) IVs for each packet***
  - The crux of the issue
    - The IV space is too small: the probability of collision is too high
- The authentication protocol is particularly ill-designed
  - Non mutual
  - It uses the same RC4-based engine (as it is used for confidentiality) as a MAC. The same key and IV space are used in both cases
- Integrity protection is left to a non-crypto function
  - It is relatively easy to create colliding messages

One Time Pad

One-time pad

Let $(\mathbb{G}, \circ)$ be a finite group, [e.g., $(\mathbb{Z}_N, +\bmod N)$]. A one-time pad (OTP) over $(\mathbb{G}, \circ)$ is the encryption system described by

$$\begin{aligned}
&\text{equal spaces} && \mathcal{M} = \mathcal{X} = \mathcal{K} = \mathbb{G} \\
&\text{uniform key} && k \sim \mathcal{U}(\mathbb{G}) \quad \Leftrightarrow \quad p_k(a) = \frac{1}{|\mathbb{G}|}\ \forall a \in \mathbb{G} \\
&\text{encrypt by add} && E(a,b) = b \circ a \\
&\text{decrypt by subtract} && D(a,c) = c \circ a^{-1}
\end{aligned}$$

Example

Let $\mathbb{G} = \mathbb{B}^N$, with $\mathbb{B} = \{0,1\}$, $N = 5$, and $\circ =$ bitwise XOR. Then, e.g.,

$$u = 01101, k = 10110 \quad \Rightarrow \quad x = u \circ k = 11011$$

B can recover the message with $k^{-1} = k = 10110$

$$u = x \circ k^{-1} = 01101$$

Secrecy of one-time pad

Theorem

The one-time pad offers perfect reliability and perfect secrecy for any message distribution

Proof.

Perfect reliability is guaranteed by the existence and uniqueness of $k^{-1} \in \mathbb{G}$.
As regards perfect secrecy, we prove that $p_{u,x}(b,c) = p_u(b)p_x(c), \forall b \in \mathcal{M}, c \in \mathcal{X}$. In fact,

$$\begin{aligned} p_{u,x}(b,c) &= \mathrm{P}\left[u = b, x = c\right] = \mathrm{P}\left[u = b, k = b^{-1} \circ c\right] \\ &= p_u(b)p_k(b^{-1} \circ c) = p_u(b)/|\mathcal{K}| \\ p_x(c) &= \sum_{b \in \mathcal{M}} p_{u,x}(b,c) = \sum_{b \in \mathcal{M}} p_u(b)/|\mathcal{K}| = 1/|\mathcal{K}| \end{aligned}$$

Observe that this result holds for any $p_u(\cdot)$. □

One time pad authentication

We aim for ε-unconditionally secure authentication against forging, i.e.

$$\mathrm{P}\left[S_{\mathsf{f}}; M, A\right] \leq \varepsilon \quad , \quad \forall A \in \mathcal{A}_{\mathsf{f}}$$

A possible solution is a mechanism M of the tag appending type, described by

equal tag and key spaces $\mathcal{T} = \mathcal{K}$ — the tag and the key are the same thing

uniform distributed key $k \sim \mathcal{U}(\mathcal{K}) \quad \Leftrightarrow \quad p_k(a) = \dfrac{1}{|\mathcal{K}|} \forall a \in \mathcal{K}$

sign by appending the key $t = T(k,u) = k \quad , \quad x = (u,k)$

verify by checking the key $\hat{b} = \begin{cases} 0 & , \text{ if } \tilde{t} = k \\ 1 & , \text{ if } \tilde{t} \neq k \end{cases}$ — if its the same you accept the message, if not you discard the message

Correctness

Trivially, it is of the tag appending type

One time pad authentication

Security

x' is the forged message

Consider the class $\mathcal{A}_\mathsf{f}$ of forging attacks, where x' is independent of k and observe that

Message forgery is the sending of a message to deceive the recipient as to whom the real sender is. Forgery -> messaggio contraffatto per ingannare il destinatario su chi sia il vero mittente

$$S_\mathsf{f} = \left\{\hat{u} = u', \hat{b} = 0\right\} = \left\{t' = k\right\}$$

$$\text{so that} \quad \mathrm{P}\left[S_\mathsf{f}\right] = \mathrm{P}\left[t' = k\right] = \sum_{a \in \mathcal{K}} \mathrm{P}\left[t' = a, k = a\right]$$

$$\text{(by independence)} \quad = \sum_{a \in \mathcal{K}} p_{t'}(a) p_k(a) = \sum_{a \in \mathcal{K}} p_{t'}(a) \frac{1}{|\mathcal{K}|} = \frac{1}{|\mathcal{K}|}$$

its not possible to get perfect authentication(epsilon=0 is not possible) with this method. If i increase the key entropy(and key space) i can go close to 0 so close to the perfect authentication

yielding ε-unconditional security, for $\varepsilon \geq 1/|\mathcal{K}|$, that is, if $H(k) \geq \log_{1/2} \varepsilon$

OTP authentication cannot offer integrity protection

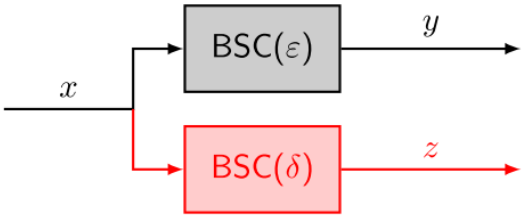
Trivial attack from $\mathcal{A}_\mathsf{m}$: F blocks $x = (u, k)$, replaces u with u', transmists $x' = (u', k)$
B verifies that $t' = k$ and accepts u'

problema, b verifica solo che k=t', non controlla che u (il messaggio codificato) sia cambiato

AWGN, BSC

Secrecy capacity for the wiretap BSC

Let the channels from A to B and from A to E be memorlyess binary symmetric with error rates ε and δ, respectively



If $\left|\varepsilon - \frac{1}{2}\right| < \left|\delta - \frac{1}{2}\right|$
legitimate channel is more noisy
(e.g., $0 < \delta < \varepsilon < \frac{1}{2}$)

$$C_\mathsf{s} = 0$$

no secrecy is possible

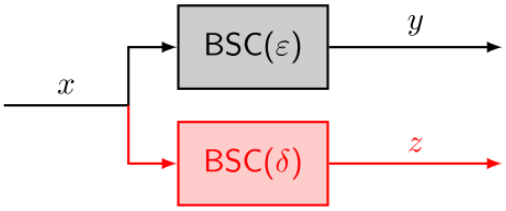
If $\left|\varepsilon - \frac{1}{2}\right| > \left|\delta - \frac{1}{2}\right|$
eavesdropper channel is more noisy
(e.g., $0 < \varepsilon < \delta < \frac{1}{2}$)

$$C_\mathsf{s} = C_\mathsf{AB} - C_\mathsf{AE} = h_2(\delta) - h_2(\varepsilon)$$

where
$h_2(\varepsilon) = \varepsilon \log_{1/2} \varepsilon + (1 - \varepsilon) \log_{1/2}(1 - \varepsilon)$

Secret key capacity for the wiretap BSC

Let the channels from A to B and from A to E be memorlyess binary symmetric with error rates ε and δ, respectively



y and z are indipendent. The probability they are different is the sigma

For all values of ε, δ the secret key capacity is attained with $x \sim \mathcal{U}(\{0,1\})$ and reverse reconciliation B $\to$ A: it yields

$$C_{\mathsf{k}} = I(x;y) - I(y;z) = h_2(\varepsilon) - h_2(\gamma)$$

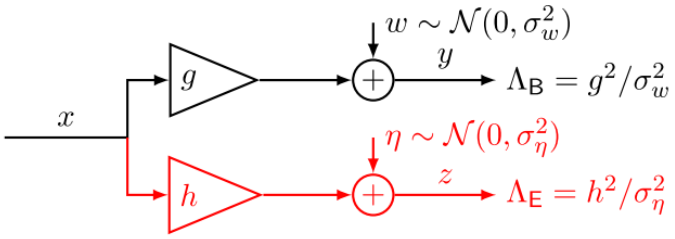
where $\gamma = \varepsilon + \delta - 2\varepsilon\delta$ and $h_2(\varepsilon) = \varepsilon \log_{1/2} \varepsilon + (1-\varepsilon)\log_{1/2}(1-\varepsilon)$.

Observe that $\left|\varepsilon - \frac{1}{2}\right| > \left|\gamma - \frac{1}{2}\right|$ for all $\delta \in (0,1)$, so that $C_k > 0$ unless the channel from A to E is perfect.

Secrecy capacity for the wiretap AWGN channel

Let the channels A $\to$ B and A $\to$ E be additive white Gaussian noise

g legitimate channel
h drop channel



here im fixing the input power, we put a bound

If $\Lambda_{\mathsf{E}} \geq \Lambda_{\mathsf{B}}$
legitimate channel is degraded

$$C_{\mathsf{s}} = 0$$

no secrecy is possible

If $\Lambda_{\mathsf{E}} < \Lambda_{\mathsf{B}}$
eavesdropper channel is degraded

shannon formula

$$C_{\mathsf{s}} = C_{\mathsf{AB}} - C_{\mathsf{AE}} = \frac{1}{2}\log_2 \frac{1+\Lambda_{\mathsf{B}} P}{1+\Lambda_{\mathsf{E}} P}$$

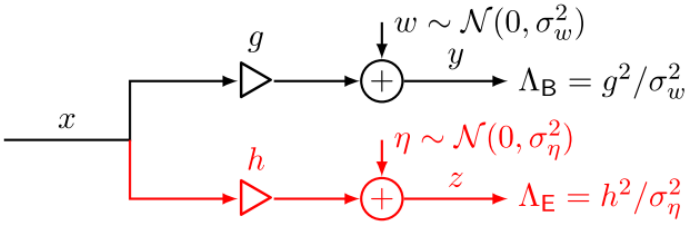
and it is achieved with $x \sim \mathcal{N}(0, P)$.

$$\lim_{P\to\infty} C_{\mathsf{s}} = \frac{1}{2}\log_2 \frac{\Lambda_{\mathsf{B}}}{\Lambda_{\mathsf{E}}}$$

P is the maximal power

Secret key capacity for the wiretap AWGN channel

Let the channels A $\to$ B and A $\to$ E be additive white Gaussian noise



For all values of Λ_B, Λ_E the secret key capacity is achieved with $x \sim \mathcal{N}(0, P)$ and reverse reconciliation. It is given by

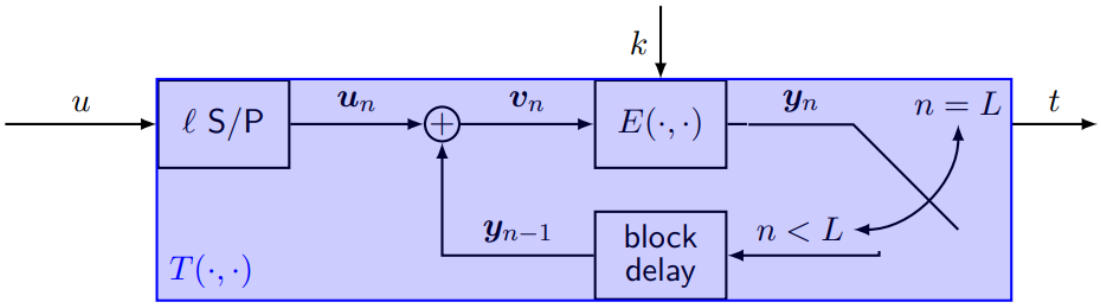
$$C_k = I(x;y) - I(y;z) = \frac{1}{2}\log_2 \frac{1-\rho_{yz}^2}{1-\rho_{xy}^2} = \frac{1}{2}\log_2 \frac{1+(\Lambda_B+\Lambda_E)P}{1+\Lambda_E P}$$

Observe that $C_k > 0$ for all $\Lambda_E < \infty$, that is unless the channel from A to E is noiseless.

$$\lim_{P\to\infty} C_k = \frac{1}{2}\log_2\left(1+\frac{\Lambda_B}{\Lambda_E}\right) \quad , \quad C_k \asymp \frac{\Lambda_B P}{2\ln 2} \quad \text{as } P \to 0$$

CBC MAC

Cipher block chaining MAC (CBC-MAC)



- Choose block length ℓ and tag space $\mathcal{T} = \mathcal{A}^\ell$ according to target security level:

$$|\mathcal{T}| \geq 1/\mathrm{P}\left[S_f\right], 1/\mathrm{P}\left[S_m\right]$$

- Choose a block encryptor $E : \mathcal{K} \times \mathcal{T} \mapsto \mathcal{T}$
- Make $(\mathcal{T}, +)$ a group, and the message space $\mathcal{M} = \cup_{L\in\mathcal{L}} \mathcal{T}^L$

Security of CBC-MAC

If $E(.,.)$ is secure, i.e., a pseudo random permutation (computationally indistinguishable from ideal random permutation)

- and $\mathcal{M}$ is prefix-free, i.e., no message u can be the prefix of another, different message u' (e.g., all messages in $\mathcal{M}$ have the same length), then CBC-MAC is computationally secure against forging and modification
- but $\mathcal{M}$ is not prefix-free, there is a known message modification attack which succeeds deterministically, can be avoided by prepending the length L to message u

Deterministic KMA attack

F observes two messages, $x = (u, t)$ with $u = (u_1, \ldots, u_L)$ and $x' = (u', t')$ with $u' = (u'_1, \ldots, u'_{L'})$, then constructs $x'' = (u'', t'')$ with $u'' = (u_1, \ldots, u_L, u'_1 - t, u'_2, \ldots, u'_{L'}) \in \mathcal{T}^{L+L'}$ and $t'' = t'$. Then at verification

$$y''_i = y_i \ , \quad i = 1, \ldots, L \quad \Rightarrow \quad y''_{L+1} = E_k(u''_{L+1} + y''_L) = E_k(u'_1 - t + y_L) = E_k(u'_1) = y'_1$$

$$\Rightarrow \quad y''_{L+i} = y'_i \ , \quad i = 2, \ldots, L' \quad \Rightarrow \quad t'' = t'$$

Elliptic curve DSA

Elliptic curve DSA

An important variant of the Elgamal signature is the elliptic curve version

Uses an elliptic curve group $(\mathcal{E}, \circ)$ on a field $\mathbb{F} = \mathbb{Z}_p$ with p prime, with cardinality $|\mathcal{E}| = q$ and a primitive point $P \in \mathcal{E}$, and mixes operations on both $\mathbb{Z}_q$ and $\mathcal{E}$.

The use of elliptic curve cryptography increases the security wrt DSA for the same key length. Hence ECDSA is very appropriate when short keys are needed (especially the public key that must be distributed)

Elliptic curve DSA

Key generation

private key space $\mathcal{K} = \{1, \ldots, q-1\}$, public key space $\mathcal{K}' = \mathcal{E}$

A randomly generates $k \sim \mathcal{U}(\mathcal{K})$, computes $k' = P \overset{k}{\circ}$

Denote by $c_1(Q) \in \mathbb{F}$ the first coordinate of a point $Q \in \mathbb{F}^2$

Signing (private, probabilistic)

$$\mathcal{T} = \mathcal{K}^2 \ , \ \mathcal{V} = \mathbb{Z}_q$$

A generates $r \sim \mathcal{U}(\mathcal{K})$

hashes message $v = h(u)$

computes $t = (t_1, t_2) \in \mathcal{T}$

$$\begin{cases} t_1 = c_1(P \overset{r}{\circ}) \bmod q \\ t_2 = (v + kt_1) r^{-1} \bmod q \end{cases}$$

Verification (public key)

B computes

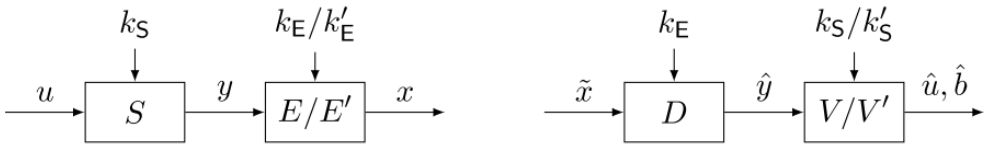
$$\begin{cases} m = t_2^{-1} h(u) \bmod q \\ n = t_2^{-1} t_1 \bmod q \\ Q = (P \overset{m}{\circ}) \circ (k' \overset{n}{\circ}) \end{cases}$$

$$\hat{b} = \begin{cases} 0 & , \text{ if } c_1(Q) = t_1 \pmod q \\ 1 & , \text{ otherwise} \end{cases}$$

Sign-then-encrypt

What if we want to both keep our message secret and guarantee is authenticity and integrity (aka build a secure channel) ?

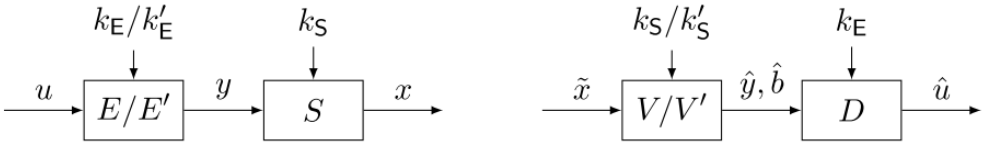
One possible solution is to first sign the information message, then encrypt the signed message



- the two mechanisms (E, D) and (S, V) can be separately designed, each with its target security requirements
- needs two distinct key pairs
- can use symmetric or asymmetric mechanisms both for signature and encryption
- was used in the Transport Layer Security (TLS) Record protocol
- vulnerable to padding oracle attacks

Encrypt-then-sign

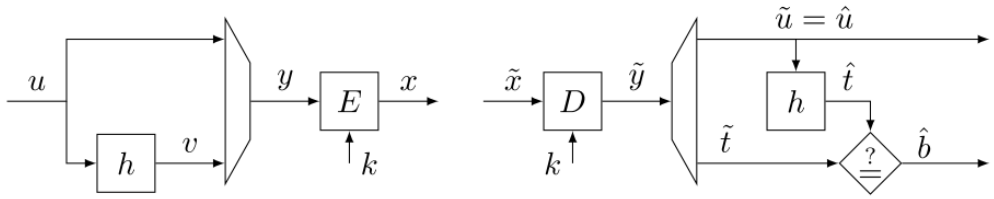
Another possible solution is to first encrypt the information message, then sign the encrypted message



- the two mechanisms (E, D) and (S, V) can be separately designed, each with its target security requirements
- needs two distinct key pairs
- if a message is not accepted ($\hat{b} = 1$), do not decrypt it: save workload and avoid padding oracle attacks
- can use symmetric or asymmetric mechanisms both for signature and encryption
- used in the Internet Protocol Security (IPSec) Encapsulating Security Payload (ESP) protocol

Hash-then-encrypt

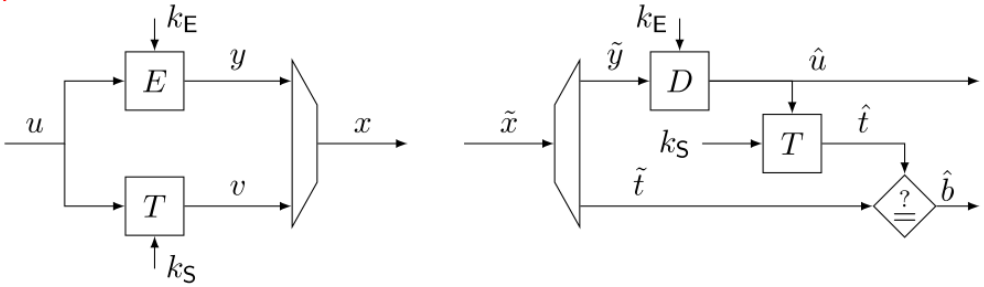
Another possible solution is to first hash the information message, then encrypt the concatenation of message and its hash



- encryption acts also as signature
- needs only one key pair
- cannot use asymmetric mechanisms (public encryption $\Rightarrow$ public signing, public verification $\Rightarrow$ public decryption)
- used in the ill-famed IEEE 802.11 Wired Equivalent Privacy (WEP) protocol (epic fail: the hash was a linear CRC code and the encryption was the RC4 additive stream cipher)

Sign-and-encrypt

Another possible solution is to compute the authentication tag on the plaintext, then append it to the ciphertext

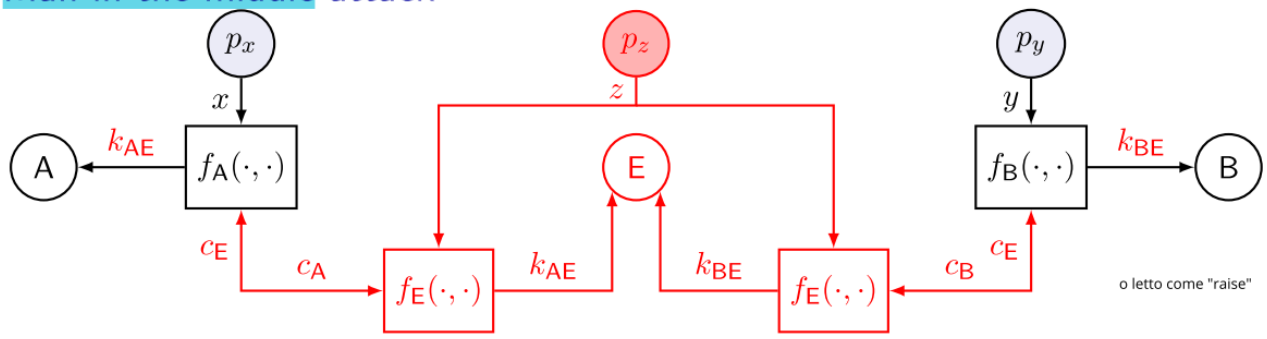


- the two mechanisms (E, D) and T can be separately designed, each with its target security requirements
- needs two distinct key pairs
- can use symmetric or asymmetric mechanisms
- in asymmetric signature, cryptographic hashing is needed to avoid revealing $u = T'(k', t)$ (beside preventing existential forgery)

Man in the middle attack

Man-in-the-middle attack

this attack only work online

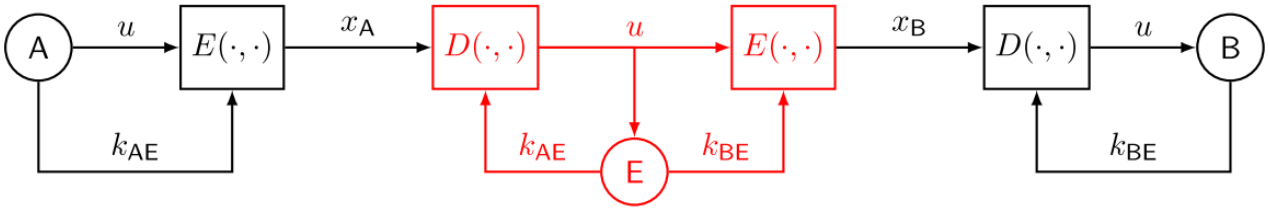


g is public

- E generates $z \in \mathbb{Z}_n \setminus \{0\}$ and computes $c_E = g \overset{z}{\circ}$
- E intercepts c_A, c_B and replaces each with c_E Ca,b,e sono le comunicazioni
- A receives c_E, computes $k_{AE} = c_E \overset{x}{\circ} = g \overset{xz \bmod n}{\circ}$
- B receives c_E, computes $k_{BE} = c_E \overset{y}{\circ} = g \overset{yz \bmod n}{\circ}$
- E can compute $k_{AE} = c_A \overset{z}{\circ} = g \overset{xz \bmod n}{\circ}$ as well as $k_{BE} = c_B \overset{z}{\circ} = g \overset{yz \bmod n}{\circ}$

Man-in-the-middle attack

If the attack succeeds, E can violate the symmetric protocol between A and B without them knowing



In order to avoid the man-in-the-middle attack, messages c_A, c_B must be authenticated and integrity protected

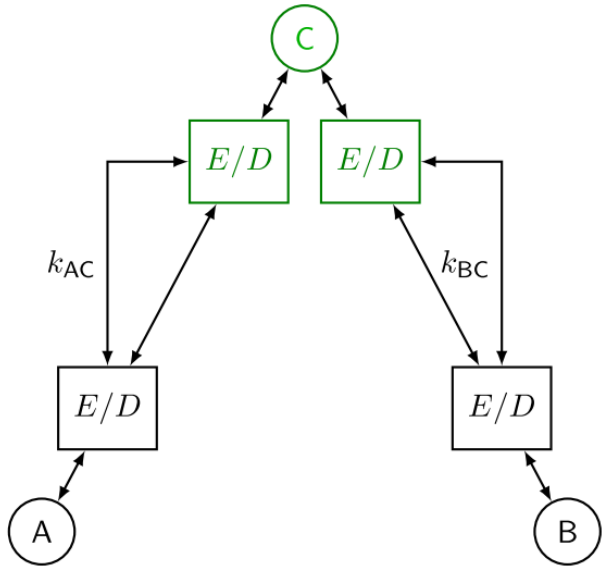
Is this a "chicken and egg" problem? No, can use digital signature

Forward secrecy

Even if E later learns the authentication provate keys, he will no longer be able to retrieve k_A

Needham-Schroeder symmetric protocol

Needham-Schroeder symmetric protocol



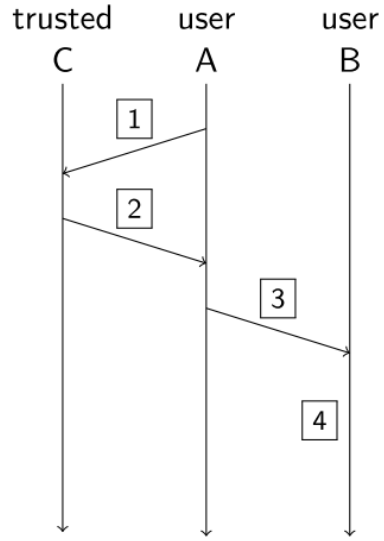
entities two parties A and B, a trusted third party C

tools a symmetric encryption mechanism $E(\cdot,\cdot)$ with keys shared between A and C, and between B and C; random generators at all entities

aim to securely distribute a key k_{AB} between A and B for a symmetric mechanism

Needham-Schroeder symmetric protocol (cont.)

phase I: key distribution



nonce= used only once

1 A : generates nonce n_A
A → C : $u_1 = (\mathrm{id}_A, \mathrm{id}_B, n_A)$

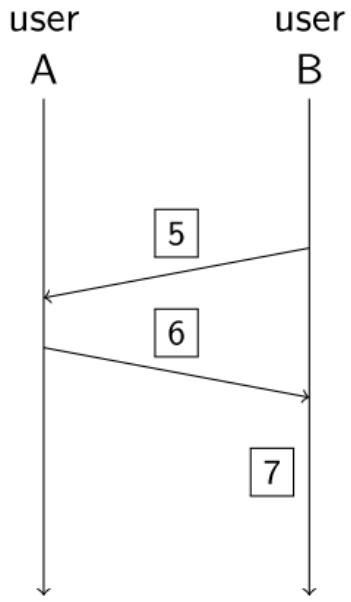
2 C : generates $k_{AB} \sim \mathcal{U}(\mathcal{K})$
encrypts $x_2 = E_{k_{BC}}([\mathrm{id}_A, k_{AB}])$
and $x_3 = E_{k_{AC}}([n_A, \mathrm{id}_B, k_{AB}, x_2])$
C → A : x_3

3 A : decrypts $[n_A, \mathrm{id}_B, k_{AB}, x_2] = D_{k_{AC}}(x_3)$
checks n_A and id_B
A → B : x_2

4 B : $[\mathrm{id}_A, k_{AB}] = D_{k_{BC}}(x_2)$

Needham-Schroeder symmetric protocol (cont.)

phase II: key confirmation



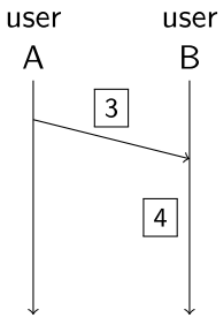
5 B : generates random $r_B \sim \mathcal{U}(\mathcal{R})$
encrypts $x_4 = E_{k_{AB}}(r_B)$
$B \to A$: x_4

6 A : decrypts $r_B = D_{k_{AB}}(x_4)$
computes $r_A = r_B - 1$
encrypts $x_5 = E_{k_{AB}}(r_A)$
$A \to B$: x_5

7 B : decrypts $r_A = D_{k_{AB}}(x_5)$
checks if $r_A = r_B - 1$

The Dennning-Sacco attack

Assume that E has learnt an old k_{AB}^{old} somehow and that he had recorded the corrsponding session between A and B
Then, E can replay message 3 to B



3 $E \to B$: x_2^{old}

4 B : $[id_A, k_{AB}^{old}] = D_{k_{BC}}(x_2^{old})$

B will use k_{AB}^{old} as if it were a good new key shared with A
solution needs a nonce known to B, too