Part I : Security Architectures

A] Symmetric Key Infrastructure - Kerberos → tickets

B] Public Key Infrastructure - PKI

1°) Principle

2°) Functions

3°) Elements

4°) Certificates & CRL

5°) Trust Models

6°) PKI Admin

7°) Revocation CRL + OCSP

RiP C]

D] Cross Certification

E] Embedded Model / Trust Anchor stores

F] Peer to Peer Architecture

1°) PGP

2°) Blockchain based

↳ Bitcoin

↳ Blockchains

↳ Architectures (DID + VC)

↳ Verified Credentials

G] ID - based Security Architecture

↳ ID based Encryption = IBE

↳ Pub Key Gen

↳ ID - based Infra

↳ Certificate less

lecture 1

lecture 2

lecture 3

lecture 4

lecture 5

lecture 6
6,5
7

lecture 8

Part II - Web Security

A] Secure Channel : IPSec

B] Secure Transport : TLS

C] Secure Messaging

↳ OTR

↳ DNSsec

D] Secure Transactions

↳ Set

↳ 3D-S

↳ EMV

↳ e-IDAS

lecture 9

lecture 10

Security Architecture

Lecture 1

Refresh week = Tools : Hash / MAC / e-sign / key exchange

Part 1 : Security Architectures

↳ Attack in the middle ? — Here to exchange keys securely (without AitM)

→ Standards : Request For Comments (RFC)

https://datatracker.ietf.org/doc/html/rfc5280

[RFC 5280]

PKCS = Public Key Crypto Standard

FIPS from the NIST

ANSSI
BSI
ENISA

Part I :

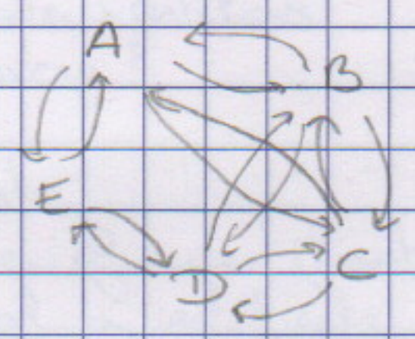
A) Symmetric Key Infrastructure : Kerberos

B) Public Key Infrastructure

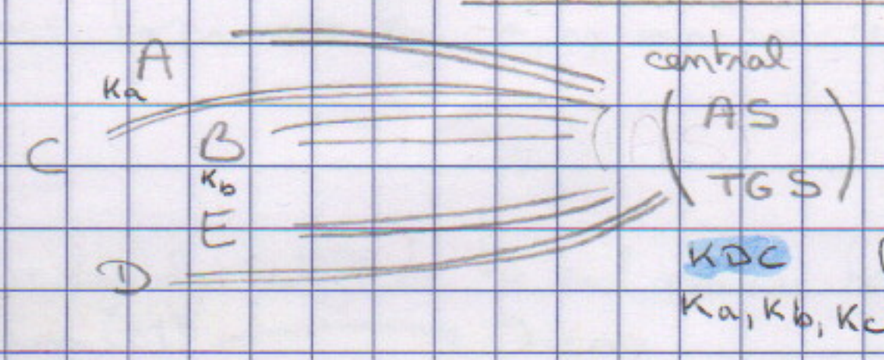
↳ Hierarchical Ours (PKIX)

Kerberos : · Authentication Server (AS)
· Ticket Granting Service (TGS)) Realm
· User
↳ id : name / role @ domain
service / hostname @ domain

AK base



Kerberos idea



- less key exchange
- one way to attack : registration

Kerberos : 3 phases

① User Authentication → Request to Authentication server → give a ticket
② Service Request → User present the ticket to the TGS to access a service
"Have access to Bob" → if all ok ⇒ new ticket
③ Actual Request → Present the new ticket to Bob → accessing the service

Tickets : $\boxed{T_{a,s} = (id_s \parallel E_{K_s}(id_a \parallel time \parallel time\ end \parallel K_{a,s}))}$

id — cipher with K_s — validity period ≈ some hours — fresh, random, ... session key

Authentication : $\boxed{A_{a,s} = E_{K_{a,s}}(id_a \parallel time)}$

cipher with session key

$A_{a,tgs}$ = Alice wants to request tgs

⚠ Symmetric → to decipher use the same key

Main outcome : Alice is sure that she is talking with Bob.
Bob is sure also

⚠ No Revocation = main drawback
↳ Short Time To Live (TTL) ≈ 8 hours
needed

- KDC knows all the keys = in company don't change so much
company has the right to read email
(no non repudiation)

advantages :
- Fast (symm keys)
- Little Surface
- Work

Can be in troubles only at registration phase

Attacks on security architecture :
- replay during TTL ? → want good cryptosystem → short TTL
- time stamping ? ← if you can change the time ← lose security
- Ka = ~~Hash (password)~~ RDF (Password)

Interoperability : Possible to have ways to access other Realm
↳ Realms

ex : une personne d'une entreprise peut chercher à contacter quelqu'un d'une autre

B) Public Key Infrastructure → certificats
→ revocation mechanisms
→ Only the public keys will be known

- Certification Authority (CA)
  ↳ issues $\underbrace{\text{certificates for}}_{\text{public data}}$ $\begin{cases} \text{public key} \\ \text{id} \end{cases}$ ← no tickets, but certificate

pub_{CA} A
$\begin{cases} pub_A \\ priv_A \end{cases}$

pub_{CA} B
$\begin{cases} pub_B \\ priv_B \end{cases}$

CA
$\begin{cases} pub_{CA} \\ priv_{CA} \end{cases}$

to decipher: just need to have public key for signature

Kerberos is secure if first secret exchange is secure
PKI is secure if first public exchange is secure

Public Key Infrastructure

B) Hierachical PKI

1°) Principle : CA = Certification Authority represents the PKI
2°) Functions
3°) Elements
4°) Certificat & CRL
5°) Trust models
6°) PKI Admin
7°) Revocation

2°) Functions

main functions
* Creation of a pair of pub / private keys
* Public key authentication → generation of a certificate = { · ID · pub key · Validity period ← months more than hours ↳ revocation · Signed by the CA }
* Certificate Delivery
* Certificate Publication ← certificates are publics
* Certificate Verification ← we have to verify the signature ; the validity period ; revocation
* Certificate Revocation

+ private key protection
+ Logging of actions
+ Archive (certificate)
+ Recovery ? ← if CA has access to private key ← no more non repudiation ← someone else knows the key

3°) Elements

* Owner of a certificate
* User of a certificate ← user use public key of a certificate to send message to the owner of it

User $E_{pub\ owner}(M)$ ⟶ Owner

* Certification Authority : top of the hierarchy
* CRL issuer (Certificate Revocation List)
* Registration Authority (RA)
* Repository
* Archive

Def PKI : A PKI is an infrastructure formed by certificats, servers, to create, manages, make available, numerical certificates which authenticity is guaranted by the certification Authority that represents this PKI.

4°) X.509 Certificates & CRL

Exam Q° → What are the 4 main component of a certificate ?

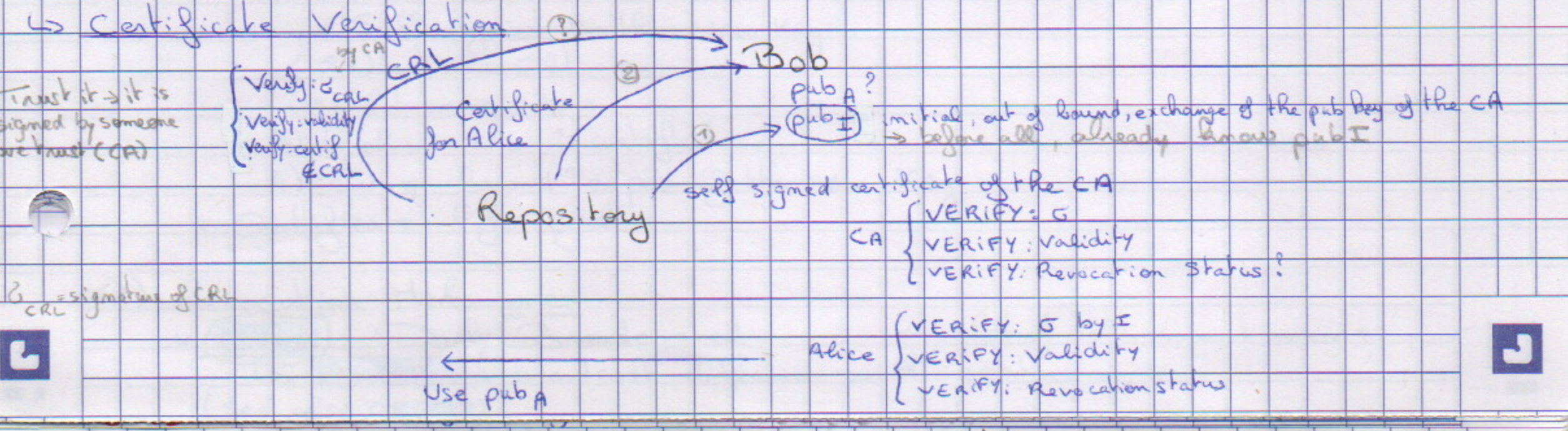
{
- ID of owner = Subject (+ algorithm)
- pub key
- Validity
- Signature by the CA "issuer
}

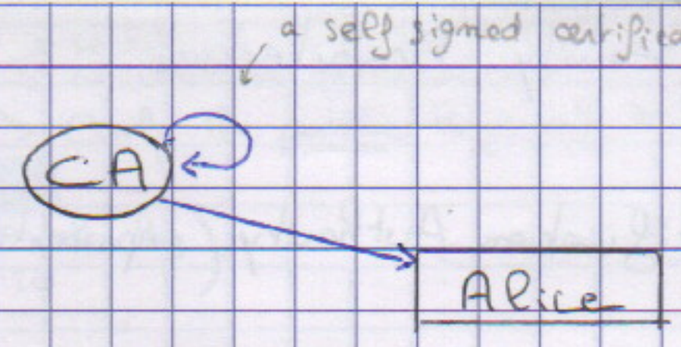
The CA sign all informations.

Serial number + CA — should be unique in the world

ASN.1 ← langage // data structures

↳ Certificate Verification





certificate :

$x \longrightarrow y$

Issuer Owner

1st Revocation Mechanism = CRL = list of revoked serial number

Unicity of certification = Serial + Issuer + Revocation Date

→ main issue of CRL : Daily Broadcasts

↳ Δ-CRL can be a solution "delta CRL"

⊕ caching

↳ pas obligé de tout vérifier à chaque fois

⊖ Black list ?

5°) Trust Models

certificates, X509 a format / Tool

→ Used in ≠ settings

a] Simple Hierarchy : CA = root of the hierarchy

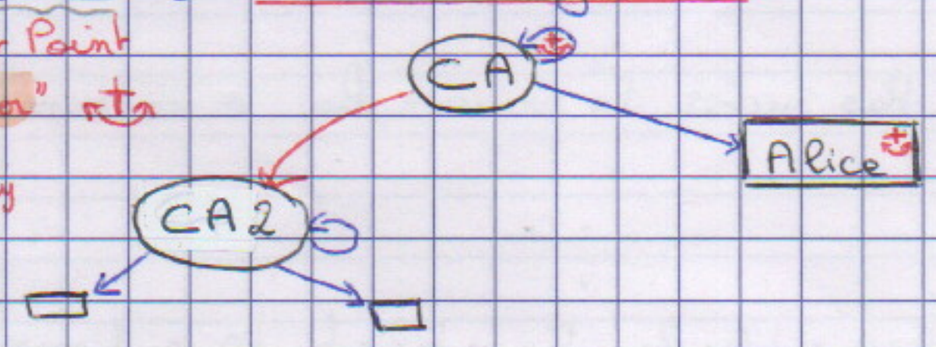
trust your CA root self signed certificates

b] Meshed Hierarchies & Cross Certification

⊕ Trust of the weakest Point

⊕ Notion of "Trust Anchor"

relative info about pub key exchanged "out-of-bound"



Trust anchor of Alice is the CA she knows

c] embedded Model

Trust Anchor Stores → browsers, OS

The trust is on what we used

browser has some trust Anchor

↳ reputation of the browser maker

Trust by Reputation of the Software Maker

d] P2P model

Trust by Accumulation

← if many people say the same thing

→ more confident this is correct

6°) PKIX → Interoperability and admin
7°) Revocation : CRL + OCSP +
D) Cross Certification
E) Embedded PKI → browser (https...) / OS
⇒ lab - Openssl

6°) PKIX : Interoperability & administration

Axiome 1 : ∃ Norm ISO-9594-8
Axiome 2 : Possibility to update (transparently) the keys
Axiome 3 : Several Algorithms could be used
Axiome 4 : User must be able to generate their own keys
Axiome 5 : Anybody could publish the certificates & CRL (⚠ D.S)
(+ several transports means, mecanisms)
Axiome 6 : CA is the final authority for certificate Creation
Axiome 7 : CA migration should be transparent too
Axiome 8 : Users should be ready to Prove Knowledge of their private key

Administration

* CA establishment
  a) Key pair gen
  b) Cert Requirement
  c) Self Signed Cert. gen
  d) CRL gen
  e) "Out-of-bounds" CA's public key publication
* CA migration
* End Entities Registration : Certification

At least 2
- Centralized (CA generates the keys) ⚠ non repudiation
- Authenticated (User generates the keys)

$IR = \{\text{"Bob"} \| N_1 \| CertReqMsg\}_{iak}$ e.g MAC $_{iak}$

$IP = \{\text{"Bob"} \| Time \| N_1 \| N_2 \| CertResp\}_{iak}$

$certConf = \{ \ldots\ldots N_2 \| N_3 \| \ldots\ldots \}$

$confack = \{ \ldots\ldots \| N_3 \| \ldots\ldots \}$

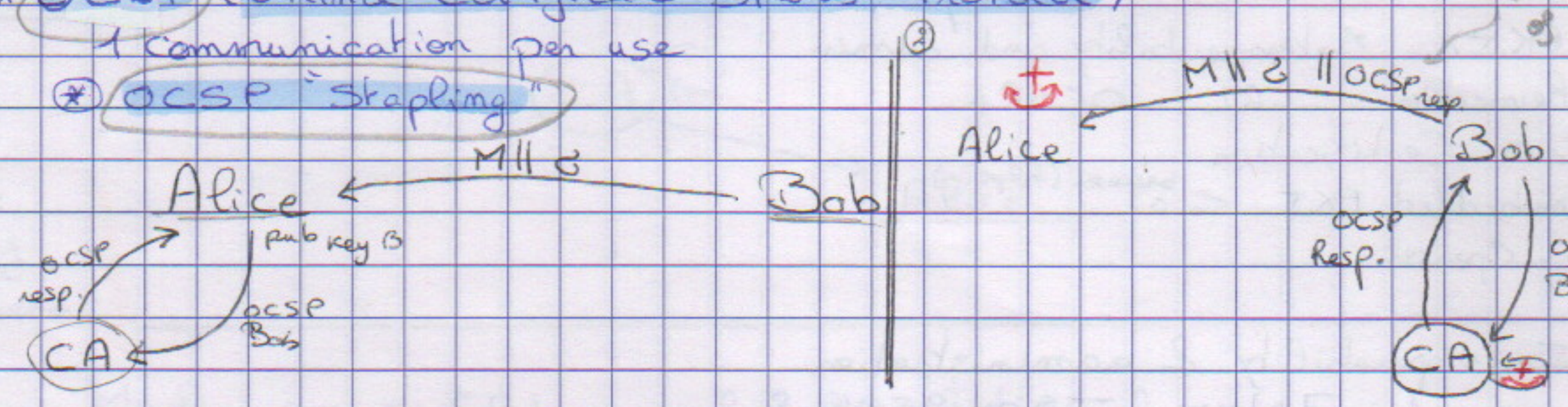
* Users Initialization
  - Recovering the CA's public key
  - Supported algorithms
  - Personal Security Envi
* Publications (Repository)
* Revocation
* CA migration
  1) New Key Pair generation
  2) 3 certificates
     - → Self signed for the New Key
     - → New → Old
     - → Old → New
  3) Publish the 3 certificates
  4) "out-of-bound" export of the new key
* Certificate life cycle

7°) Revocation Mechanismes

* CRL : Daily Broadcasted ← lot of data ⇒ not amazing
  ↳ Δ-CRL : broadcast difference with yesterday
  (see also CRLite)

Au lieu de tout le temps devoir demander au Serveur si le certificat est valide
→ Avoir un message sur l'état de son propre certificat (signé par la CA), et l'ajouter dans son message ⇑ ⑨

* OCSP (Online Certificate Status Protocol)
  1 communication per use
  ⊛ OCSP "Stapling"



* Certificate Revocation System (Naromodo ou Novomodo)
  ↳ One way Hash-Chain

CA : { • $X_0 \xleftarrow{\$}$
• $Y_0 \xleftarrow{\$}$ random secrets

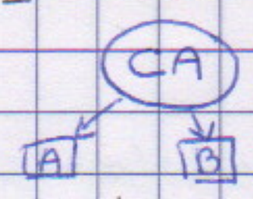
H : Crypto-Hash f°
{ • $Y_1 = H(Y_0)$: revocation target
• $X_m = H^m(X_0)$: validation target

→ added (publicly !) to the certificate (extension)
at each time interval ($i = 1 \dots n$) the CA publishes 1 hash value

if $H(Y_0) = Y_1$: Cert revoked Y_0 ← $H_{m-i} = H^{m-i}(X_0)$

Cert still valid $H^i(X_{m-i}) \overset{?}{=} X_m$

D] Cross Certification



Simple Hierarchy

↳ Cross-Certification & Trust Anchors

Sub CA : • Self signed Cert
• Ability to create certificates

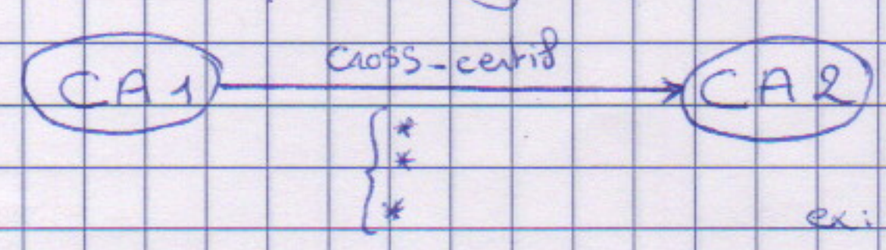
{ Relative
→ the pub key that are known to user A

↳ Hierarchical Cross Cert
↳ P2P Cross Certification

cross-certificate = Certificate of a CA by another one

↳ ⚠ Security of the whole system is the security of the weakest point

⇒ Trust Extension & Limitation
Constraints ∈ Cross-Certificate
* Naming Constraints
* Policy Constraints
* path length constraints



ex : • name count = permitted : IP : 192.168.0.0
• unhibit Policy Mapping
• path length = max # of intermediate cert. that can follow in a valid path

Trust anchor in the root

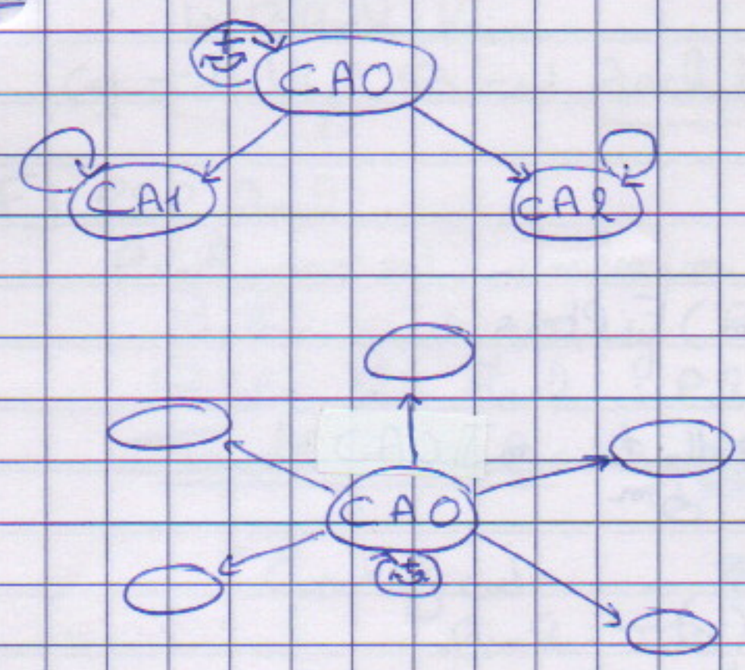
Security Architectures

lecture 4

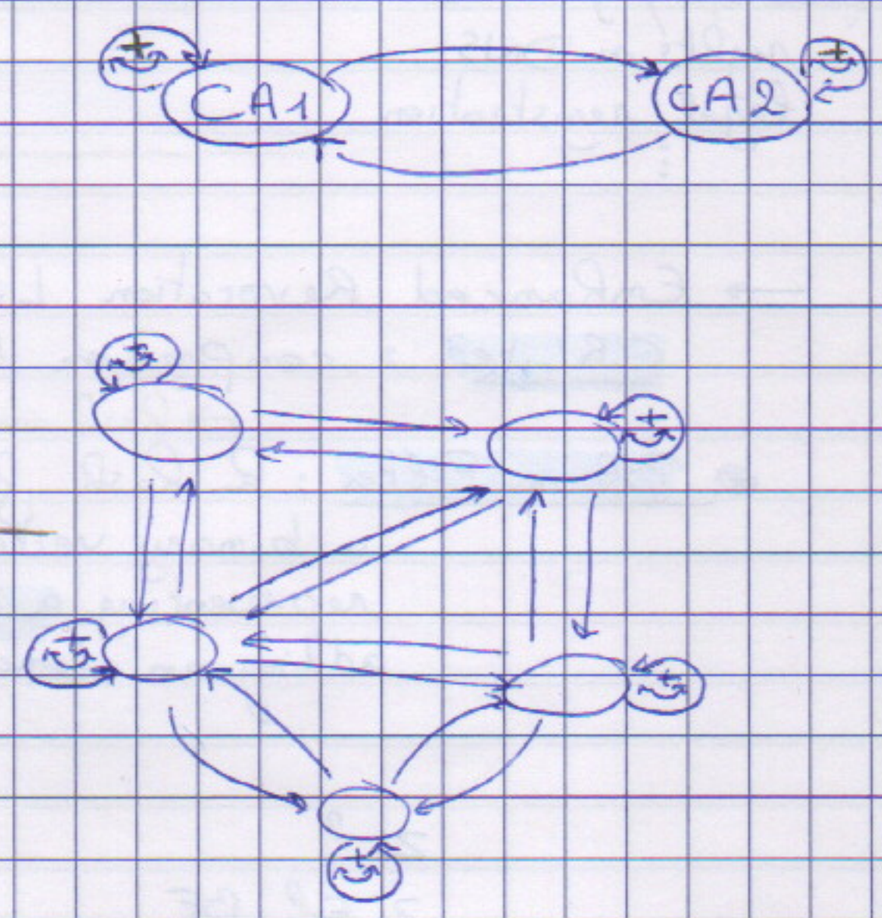
C) PKIX

D) Cross-Certification

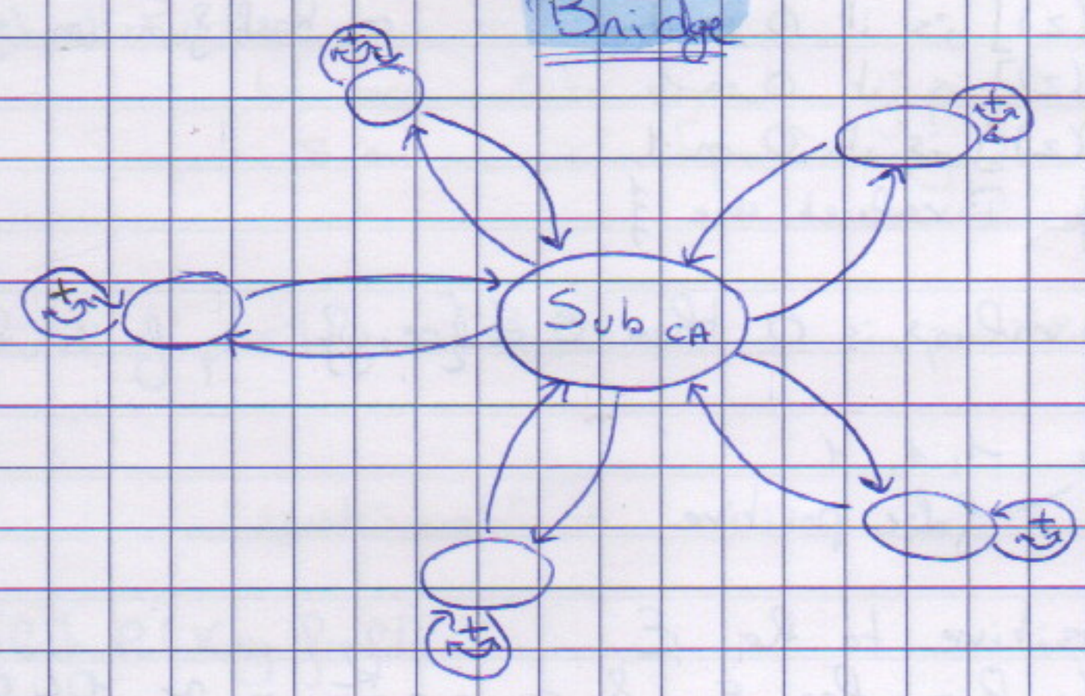
Hierachical Cross Cert.



Distributed Cross Cert.



Bridge



E) Embedded Model

↳ X.509 hierachical

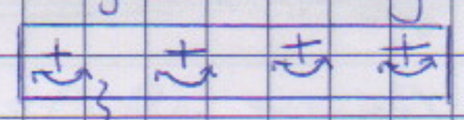
↳ Trust Anchor Store → browsers / → OS

Many Anchors embedded within the software.

↳ trust in the reputation of the software Maker

← trust is now on the reputation of the software maker

firefox, chrome, edge: certificate Manager



Store

→ https presents a certificate

Let'Encrypt → 60% market share

→ "page dangereuse" Juste le certificat n'est pas dans ceux en stock

Stores:

Same on OS: Android: Trusted Credentials → CA cert.

Windows: certmgr.msc

Linux: /etc/ssl/ca-certificates.crt

Additional Security Layers ← on top of the architectures

→ Pinning (Obsolete)

↳ warning

→ Append ← only logs (now called blockchain)

database DB → certificate Transparency

→ blockchain (...)

→ Notaries (Convergence, Perspective, Dns, ...)

↳ Extended Validation

DV < OV < EV (coût faible, coût moyen, coût élevé)

↳ domain validation — "nom du domaine" — DV ← you are accessing https website

↳ organisational — "vérification du propriétaire du domaine" — OV ← identity of the owner, legal registration

↳ extended — EV ← niveau de certification le plus élevé

Identity of owner
rights on DNS
legal registration
⋮

→ Enhanced Revocation Lists ← issue of CRL important (too many lines)

CRLite : compression by a cascade of Bloom Filters

→ hash multiple times

⊛ Bloom Filter : k hash functions : $\{0,1\}^* \to [\![1 \dots m]\!] \subseteq \mathbb{N}$

a binary vector of size m, initially 0^m

representing a set of elements

adding an element x : $h_1(x), \dots, h_k(x)$

		1			1			1	

z ?

$z \in$? BF

BF $[h_1(z)]$ is it 0 or 1 ← hash function gives an index

BF $[h_2(z)]$ is it 0 or 1

BF $[h_3(z)]$ is it 0 or 1

- if $z \in \{x, y\}$ 3 values are 1 1 1
- if one of the 3 values is 0 then $z \notin \{x, y\}$
- if 3 values are 1, 1, 1
  $z \in \{x, y\}$ → false positive ← "coincidence"

proba of false positive to be ε

↳ $k = -\log_2 \varepsilon$ & $m = -\frac{k}{\ln 2} n \approx 1.4\, k n$ ← size of the set

Revoked BF certificates

Test(z) $\in 0$? $\xrightarrow{0}$ Safe, $z \notin$ CRL

↓ 1 can be false positive

f.p

0 → $z \in$ CRL

↓ 1 can be false positive

- CRL Large Set
- All the certificates $\mathbb{U}_{cert}$ ← universe

$CRL \subseteq \mathbb{U}_{cert}$ (f.p)

At the end $\mathbb{U}$ is finite ← deterministic to know f.p

Bloom filter smaller and smaller

CRL : 10 Million cert. revoked
$\mathbb{U}$: 30 Million active
BF cascade : 10 MB
daily update ≈ 500 KB

order of magnitude gain over CRL and Δ-CRL

Security Architecture — Lecture 5

D] Cross Certification
E] Embedded Trust Anchor Stores
F] Peer to Peer Architecture
↳ PGP
↳ Blockchains
G] Identity-Based Architecture

certificate ← possible to see phishing

F] P2P Arch.
PGP
Pretty good privacy (P. Zimmerm)
Public Key Arch ! P2P
Trust Model: By accumulation

"Certificate" = Combinaison of Packets
link ID with PubKey
+ Validity
+ (many) signature

- PubKey Packet —Hash→ Keyid
- ID Packet
- Signature Packet

} slide avec le contenu

self signature

↳ accumulate signature of people we know
Si on ne connait pas, on regarde les signatures
↳ si on en connait une → start to have trust
↳ transitivity
→ possible de mettre le niveau de confiance de la sign.

Many kind of possible packet

* Packets
* Signatures
* Key ID
  - Select PubKey fields
  ↳ Hash
  ↳ 64 lower bits (used to be 32)
* levels of Trust
  ↳ Partial trust
    - None (don't trust)
    - Partial = at least $m \geq 2$ → keys with partial trust have to sign a key for you to
    - Complete = at least n keys with $m > n \geq 1$ → have to sign a key for you to → trust it
    - Ultimate = Secret Key KNOWN ← Know the private key links with the public → it is yourself

idea of accumulation

* Verification levels (when signing)
  - generic : no verification
  - persona : think this is a correct person
  - casual : partial verif (checksum, passeport, picture ...)
  - positive : extended verification (in person IRL, ...)

* Key rings
  Private Key Rings
  Public Key Rings

* Revocation ← to have long validity time
  ↳ Key Revocation : only by the owner ! ← (you, not the CA)
  Revocation request signed by the owner
  ↳ Made at the key creation time and stored elsewhere → or sent to a "Designated Revoker"
  ↳ Signature Revocation

* Key Servers ← toutes les clés sont stockées dedans (partout dans le monde)
  ↳ Synchronized ! ← not delete things

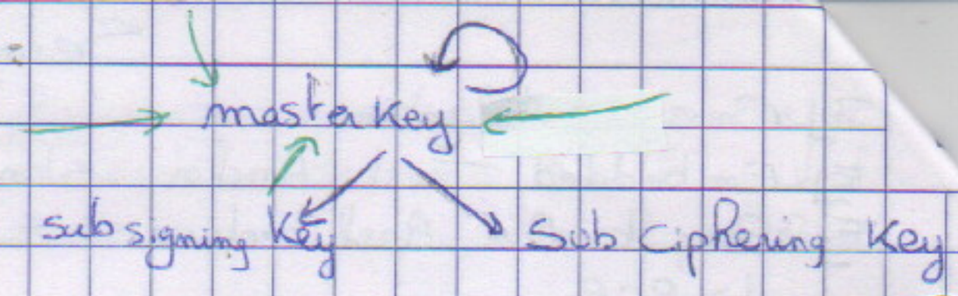
⚠ DoS

gpg

create pgp key ⚠ info mondiale ← lab 2

- sign-key
- edit-key
- export

in PGP :



not https → http
https
↳ to be ciphered

→ infos données avec le packets

OpenKeyChain → Android

$$E_{pub_1}(K_S) \| E_{pub_2}(K_S) \dots \| AES_{K_S}(Message) \| \sigma_{pub}$$

Security Architecture

lecture 6

C
D
E
F) P2P
F.1) PGP
F.2) Blockchain based
G) ID-based Infra
II) Network security

Interlude

PKI-based
- Authentication (SASL
- Key Transport (KEM
- Key Exchange (SIGMA

} secured

Security services

⊛ Certificate based Authentication

FIPS-196 ·····> SASL

Token : challenges (random Nounce)

Token $BA_1 \ni N_1$

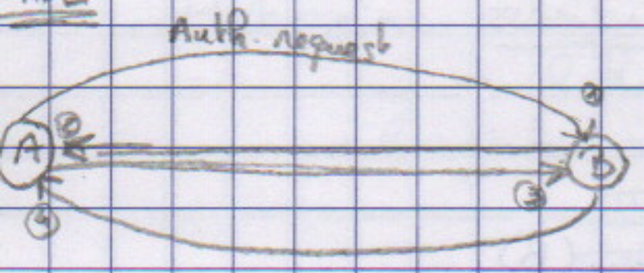
Token $AB \supseteq \{N_1, N_2\}$ ③ after ②

Token $BA_2 \supseteq \{N_2, \dots\}$ ④ after ③

Token ID : identities

[Cert x] : optional

SASL



(le certificat est public, on est authentifié grace à la signature)

in ③ : Alice is authenticate towards Bob (by using Ker priv A)

in ④ : Bob is authenticate towards Alice (by using Ker priv B)

Security : by the Sec. Arch that provides guarantees on the certificates that certify $\begin{cases} \text{pub A} \\ \text{pub B} \end{cases}$

⊛ Key Transport (Key encapsulation mecanism)

Alice — Bob

$Ks \xleftarrow{\$}$ Session Key

$Pub_B \leftarrow$ Sec. Arch.

$c = E_{pub_B}(Ks)$ —c→ $D_{priv_B}(c) \to Ks$

$AES_{Ks}(\dots)$ → $AES_{Ks}(\dots)$

C'est l'idée mais pas fait comme ça en pratique

Slide PKE / KEM

on génère de la randomness

Alice — Bob

$r \xleftarrow{\$}$

$C = E_{pub_B}(r)$ —c→ $r \leftarrow D_{priv_B}(c)$

$K = KDF(r)$ — $K = KDF(r)$

shared secret

Mais comme ça

KDF : Key Generation Function

ex: HKDF : IKM = initial randomness

⊛ Key Exchange (SIGMA)

↳ Diffie-Hellman } Prevented by Authentication and SIGNATURES

Mais insuffisant

Alice — Person-in-the-middle Attack — Bob

$a \xleftarrow{\$}$ — $u' = g^{\alpha}$, $v' = g^{\beta}$ — $b \xleftarrow{\$}$

$u = g^a$ —u→ — $K_1 = u^{\beta}$, $K_2 = v^{\alpha}$ — ←v— $v = g^v$

$K = v^a$ — $K = u^b$

Shared Secret

She cannot decipher, only pretends it is her that create the message ← not very powerful

But : Identity misbinding ⚠

Alice — Eve — Bob

$A \| g^x$ → $E \| g^x$ →

← $B \| g^y \| Sig_B(g^x, g^y)$

$Sig_A(g^x, g^y)$ → $Sig_E(g^x, g^y)$ →

To prevent it : Alice — Bob

1) $A \| g^x$ →

← 2) $B \| g^y \| Sig_B(g^x, g^y, A)$

3) $Sig_A(g^x, g^y, B)$ →

$Sig_B \leftarrow priv_B$

$Sig_B \leftarrow decrypt : pub_B$ } à vérifie

So man in the middle will be detect ← and we can block the comm°

Station to station protocol

Alice → Bob

1) g^x →

← 2) $g^y \| E_K(B, SIG(g^x, g^y))$

3) $E_K(A, SIG_A(g^x, g^y))$ →

E cannot get K by DLOG hardness

Mais on encrypte trop ← un MAC suffit

SIGMA (Sign and Mac)

Alice → Bob

g^x →

← $B \| g^y \| SIG_B(g^x, g^y) \| MAC_K(B)$

$A \| SIG_A(g^x, g^y) \| MAC_K(A)$ →

Good way to prevent identity misbinding = MACS
Good way to prevent man in the middle = SIGNATURES } SIGMA

← la base de toutes protocoles

Variants : SIGMA-I and SIGMA-R

Soit KEM ou SIGMA
↳ Symmetric

Possible to have only authentication : SASL

Encrypt then MAC ↳ TLS

Encrypt & MAC ↳ SSH

MAC then Encrypt ↳ SSL/TLS

F.2) Blockchain - Based Architectures
↳ Bitcoin
↳ Blockchains
↳ Architectures
(DID + VC) ← verified credentials

Money = Medium of exchange / Store / ...

Bitcoin ← privacy and want to avoid third party

unforgeability : don't create money
double spending : can't spend 2 times 10$ if you have only 10$

Blockchain : have a book where we know all transactions ← everybody has it and not only the bank

Sign the hash of all previous chain ← chain signature
weak privacy

Nb of bitcoin is finite 21 millions BTC

Security by randomness → difficult to know which person can write

Hard puzzle to solve but easy to check

Creation by the reward

$$P(H(H(\ldots; \$) < Target) \approx \frac{1}{2^{74}}$$

Merkle Has trees ← don't recompute all each time

Forks : Longest chain rule
← 2 people mine at the same time ← take the longest chain.

Hard fork or soft fork

worst : power requirement → TWR (small state ≈ AURA region just for bitcoin)

become rich with mining ?
Mining farms
↳ party that have money creation

Alt coins
↳ like etherum
Classification I
II
III
IV : other consensus algorithm

PoW
↳ Proof of work

PoS
↳ Proof of stake
↳ richer write in the book
↳ rich will be the things better to don't lose all

Etherum
validation : not 10 minute → 12 s
↓ more simultaneous validation
many forks at every blocks
reward everybody that mine / reward for having find a fork

Peercoin
Si on decide que l'argent doit tourner
→ "pénie au bout de 120 days"

Stablecoins
USDt : same amount than 1 $ but more privacy

Security Architecture — lecture 7

F.2] Blockchain-based Sech. Architecture
⋮
G] ID-based Sec. Arch. ← tomorrow
II / Network security ← after the break

Blockchain fundamentals

security by a chain (hashed) of block
- 1°] Ledger ← distributed
- 2°] Consensus ← who is allowed to write
- 3°] Incentive ← cryptocurrency

Unstoppable since distributed

Auditable ← all participants have copy, locally of the ledger

Storage "on" blockchain : integrity + time stamping
Proof of stake ← PoS : si tu as beaucoup de crypto : tu as tout intérêt que la crypto ne s'effondre pas ← must be "rich"
etherum : PoW + PoS ← hybrid
↑ work
Together: better Proof of Stake

Smart contract — ex: "The transaction is valid" "Alice has enought money" + "Bob fullfill the contract"
etherum contains programming language ← possible to write contracts with it

Tokens — fungible ← completely exchangeable
Build Tokens on the cryptocurrency
a token is a smart contract.
Crypto mining are necessary
Tokens : ERC-20 ≃ API specification
coins :

smart contracts
↳ ⚠ highy parallel settings
↳ Token → new crypto (with secure underlying chain — another cryptocurrency
↳ Sidechains (after lighting Network). ← do the addition → add transaction at the end of the day
↳ possible to have a sidechain of a sidechain — for gathering transactions
↳ improve scaling

↳ Decentralized Autonomous Organisation (DAO)
↳ Decentralized Application (DApp)
want to restrit but without transaction

Dentralized Finance

Coins vs Tokens

Non fungible Tokens (NFT) = unique and non interchangeable unit of data stored on a digital ledger (blockain)
↳ cf ERC-721

Crypto_Kitties

message ... car on ne l'a pas écrit. Mais si la clé est ... → pas possible de savoir le contenu ← $AES_{K_S}(m_2) = c_2$

↳ Realms

ex: une personne d'une entreprise peut chercher à contacter quelqu'un d'une autre

Blockchain-based security Architectures

PKI → CA
PGP → Web of Trust } → pub keys

ex: certificate transparency with append-only logs

↳ Blockchain based Security Architecture.

- Append Only
- Timestamped
- Integrity
- + Distributed

Main Principle ⇒ PKI
ex: cert Coin

Not manage by 1 company but by the ledger

ex: BPKI (Blockchain PKI)
↳ X.509 extensions = { Blockchain name
smart contract of the CA
↳ managing logs / revocation ...

- IKP: behavior of CAs
- IoTchains: for access control

⊛ DID + VC

decentralized identifier ← decentralized database
Verifiable Credentials

→ cf. W3C Decentralized IDentifiers
→ verifiable Credentials : generalization of both { PKI certificates
PGP keys

Holder

"Needs to "trust" the verifier.

Needs to "trust" the holder

Issuer

Verifier

Needs to "trust" the holder

Proof by Signature

Trust?

decentralized blockchain

Security Architecture

Lecture 8

6] ID-based Secu. Arch.

* ID based Enc = IBE
* Pub Key Gen
* ID-based Infra
* Certificate less

the identity is the public key

← de base, sert à lier l'ID et la public key ⇒ si l'ID est la pub key...

IBE: (ID Based Encryption)
- Setup
- Extract
- Encryption
- Decryption

Correctness: $D_{K_{ID}} \circ E_{ID} = \text{Identity}$

→ IB signature: A. Shamir 1984
→ IBE: Cocks 2001: quadratic residues
Boneh & Franklin 2001: pairings
Trust Third Party ⇒ PKG (pub. key Gen)

Setup → $K_{pub\,PKG}$, $K_{priv\,PKG}$
Extract → K_{ID} from (ID, $K_{priv\,PKG}$)
⇒ PKG {can reconstruct / knows all the secret keys

On ne peut pas créer une clé privé avec ID ← tout le monde pourrait le faire

Encrypt(m, ID, $K_{pub\,PKG}$) → c
Decrypt(c, ID, K_{ID}, $K_{pub\,PKG}$) → m
but after key Extraction K_{priv} PKG can be destroyed... ← n'est plus utilisé ?

* Boneh & Franklin: pairing scheme IBE

pairings = $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$
bilinear map

$$\begin{cases} e(P +_{G_1} Q; R) = e(P, R) +_{G_3} e(Q; R) \\ e(P; R +_{G_2} S) = e(P, R) +_{G_3} e(P, S) \end{cases}$$

$$\begin{cases} e(xy; z) = e(x; z) * e(y; z) \\ e(x; zt) = e(x; z) * e(x; t) \end{cases}$$

Corollary:
$$\begin{cases} \forall (\alpha, \beta) \in \mathbb{Z}^2 \quad e([\alpha]P; [\beta]Q) \\ \qquad = [\alpha\beta]e(P; Q) \end{cases}$$
$\forall \alpha, \beta \quad e(x^\alpha; z^\beta) = e(x, z)^{\alpha\beta}$

Hyp: Bilinear Diffie Hellman (BDH)
$\exists\, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e$ st from $\begin{cases} e(x, z)^\alpha = e(x^\alpha; z) = e(x; z^\alpha) \\ e(x; z)^\beta = e(x, z^\beta) = e(x^\beta; z) \end{cases}$
still HARD to compute $e(x, z)^{\alpha\beta}$

e should be efficiently computable and non degenerate
$$\begin{cases} \forall P, \exists Q, e(P, Q) \neq 1 \\ \forall Q, \exists P, e(P, Q) \neq 1 \end{cases}$$
"Not everything is 1"

ex: Weil Pairings
$E(p; 0; 1) = (\{(x, y) / y^2 = x^3 + 1 \bmod p\}$ ← Eliptic curve
$+$ group | addition / doubling)
double and add ⇒ fast exponentiation

with $p \in \mathbb{P}$ (prime), $p = 2 \bmod 3$ and with q points (solutions) $\in \mathbb{P}$, $q > 3$
and $q \mid p+1$ and a point P of order q
Then $\mathbb{G}_1 = \mathbb{G}_2 = \{[i]P, i = 0 \ldots q-1\}$ and $\mathbb{G}_T \subseteq (\mathbb{F}_{p^2}^*, \times)$ of order q
then $\exists$ e that seems to satisfy BDH
ex: Modified Tate, etc

B&F
PKG Setup: $\begin{cases} s \xleftarrow{\$} \text{secret} \\ K_{pub\,PKG} = R^s \end{cases}$

public $\begin{cases} \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e \text{ satisfy BDH} \\ R \text{ generator of } \mathbb{G}_2,\ H_1 \text{ hash}: \{0,1\}^* \to \mathbb{G}_1 \setminus \{0\} \\ \qquad H_2 \text{ hash}: \mathbb{G}_3 \to \{0,1\}^n \end{cases}$

↳ symmetric Encryption

quand on échange, message est de l'ai mais pas de mots car on ne l'a pas écrit, Mais si la conversation est leak → pas possible de savoir le contenu

$\xrightarrow{m_2}$
$\xleftarrow{AES_{K_S}(m_2) = c_2}$

↳ Realms

ex: une personne à une entreprise peut chercher à contacter quelqu'un d'une autre

Extract : $priv_{Bob} = H_1(\text{"bob@email.fr"})^s$

secure channel

Bob — Alice

Encrypt

$g_{bob} = H_1(\text{"bob@email.fr"})$

$r \xleftarrow{\$}$

$\gamma = e(g_{bob}; K_{pub}\ PKG)$

$\begin{cases} c_1 = R^r \\ c_2 = M \oplus H_2(\gamma^r) \end{cases}$

$c_1; c_2$

$c_2 \oplus H_2(e(priv_{Bob}; c_1))$

Proof: $\gamma^r = e(g; R^s)^r = e(g; R)^{rs}$

$e(priv_{Bob}; c_1) = e(g^s; R^r) = e(g, R)^{rs}$

PKG

LRA : Local Registration Authority (possible to have several)

↳ offline authentication

↳ uniqueness of the ID

Secure channel ?

1°] Alice : random temporary classical pub key and associate temporary priv. key

$\begin{cases} TK_{pubA} \\ ID_A \end{cases}$ out of band to LRA

2°] XOF = extendable Output Function. Hash $\{0,1\}^* \to G_2$

$a = XOF(ID_A)$ unique ?

Request = (a, TK_{pubA}) signed by the LRA

→ Validity ? ~~Revocation~~ ? ← because of the system (Key = ID)

impossible to change an identity

↳ Secret Key renewal → for everybody

↳ Periodic Renewal of every key (use the old key to get the new key)

∃ ways to improve to log n by log n change of keys

⚠ slide over routes les archi (Kerberos / PKIx, IBE, PGP)

ID-based : PKG create all secrets !

↳ mitigation = "Certificate less" ← double authentication

Prevents "Honest-but-Curious PKG"

Part I: Security Architecture Web Security Architecture / Web Part — lecture 9

↳ just after the break

Part II : Web Security

Secure Channel : IPSec
- transport : TLS
- Messaging : DNSSec & OTR
- Payment.

security architecture → SIG : protect against man-in-the-middle
MAC : identity missbinding } Countermeasure

A) IPSec

provides → Authentication + Integrity
↳ AH (Authentication Header)
↳ ESP (Encapsulating Security Payload) → Authentication + Integrity + Confidentiality

2 sub protocols

} → tunnel mode → seules les données du paquet sont protégées
→ transport mode → l'integralité du paquet est encapsulé dans un nouveau paquet.
↓ utilisen pour les connections VPN car masque l'adresse IP d'origine.

↳ 4 ≠ modes in total

IPSec Specificities
- SEQ : freshness (against reply)
- ICV : Integrity Check Value ← MAC
- SPI : identifier (for IKE) ← do the link between packets

IPv4 & IPv6

1°) IKE (Internet Key Exchange) → phase 1. SIG / phase 2. MAC — 2 parties dans le protocole.
2°) Auth & Integrity & Confidentiality of Data
↳ formats AH, ESP, transport or tunnel

Internet Key Exchange (IKE) : the PKI of IPSec

1°) IKE

1°) a) ISAKMP = Signature = SIG
1°) b) Phase 2 = MAC

Perfect Forward Secrecy (PFS) ≈ Key Exchange (with PFS)
↓ $K = (g^a)^b = (g^b)^a$
g^a public key, g^B public key

vs (without PFS) → Key Transport
↓ $E_{pubK}(K_s)$

Après la phase 1,
on a créé une clé de session : $K_s = (g^a)^b$ $K_s = (g^b)^a$
Phase 2. MAC

B) Secure Transport : TLS ← Transport Layer Security

→ TLS 1.3
5 sub protocols : Handshake / Change cipher spec / Alert / HHp / SSL record protocol

Handshake : SIG & MAC ← secu architecture
* garantir que la clé est authentique et provient d'une entité fiable
* vérifier l'identité des parties impliquées dans la communication
→ * garantir l'intégrité des données (données non modifiées ou corrompues durant le transfert)
* l'authenticité des données, s'assurer qu'elles proviennent bien de l'expéditeur attendu.

1.2 → finished = Hash all previous messages
→ 1.3
- no more compression
- no more Key Transport
- ciphered from the beginning

formal proof of security } ⊕ Session Resumption

k_1 ← not secure
$k_2 = TKDF_2(m_1, \text{trans})$ ← secure cause SIG & MAC before
k_2 = secure
k_n = just to make attacker's life worst.

C) Secure Messaging

variant of SIGMA protocol

* OTR (Off The Record) ← whatsapp & Signal
- Authentication
- Confidentiality
- Integrity
- Deniability
  ↳ : symmetric Encryption

quand on échange, on sait que le message est de lui mais pas de moins car on ne l'a pas écrit. Mais si la conversation est leak → pas possible de savoir l'auteur

Key Exch Auth.
K_s → ← K_s
$c_1 = AES_{K_s}(m_1)$ →
← $AES_{K_s}(m_2) = c_2$

OTR exchange → renew the session key for each message

Post Compromise Security → PCS ← combien de clé de session il faut briser

SIGNAL → asynchronous double ratchet → possible d'envoyer plusieurs message avant d'en recevoir.

↳ à chaque fois que l'on reçoit un message : on change le Diffie Hellman paramètre de l'autre

↳ à chaque fois que l'on envoie un message : on change notre Diffie-Hellman paramètre

Une matrice :

$$\begin{array}{ccccc} & g^{y_1} & \longrightarrow & g^{y_2} & \longrightarrow & g^{y_3} \\ g^{x_1} & & & & & \\ \downarrow & & & & & \\ g^{x_2} & & & & & \\ \downarrow & & & & & \\ g^{x_3} & & & & & \end{array}$$

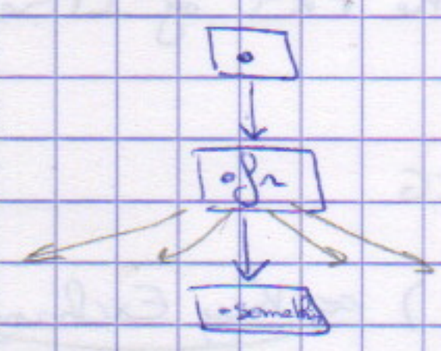
} il faut trouva la bonne combinaison

* DNSSec

↳ a PKI on top of the hierarchie :

"certificate"
- DNS Public Key (DNSKEY)
- Resource Record Signature (RRSIG : validity)
- Delegation Signer (DS : Signature by the CA).
-

Certification path (or chain)



NSec : Signed list of all the existing domains

NSec3 : " " " the hashes existing domains ← une white list mais avec hash

white list of servers

* Time Stamping ← Security Arch.

DNSSec → ne chiffre pas les données

→ garantit que les utilisateurs atteignent le site légitime associé à un nom de domaine