

ZZ2 F5 - Sécurité des applications web

Vie privé et Cryptographie

Vie privée

- ▶ « Je n'ai rien à cacher »
- ▶ Chiffrement de bout-en-bout
- ▶ Anonymat sur le web

Cryptographie

- ▶ Propriétés clés (confidentialité, authenticité,...)
- ▶ (Brève) histoire de la cryptographie
- ▶ Primitives classiques
- ▶ Les pièges des implémentations



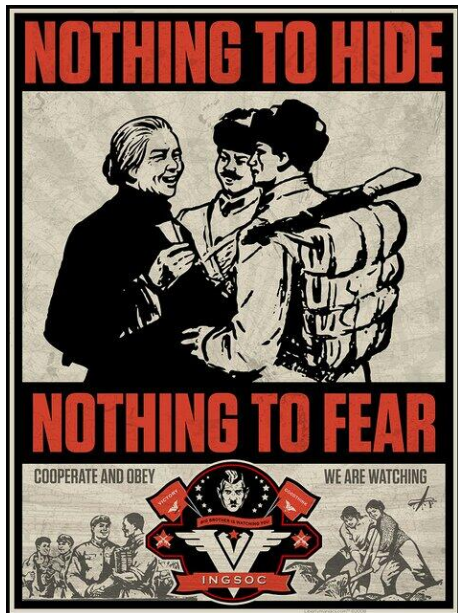
Section 1 : **Vie privée**



Je n'ai rien à cacher ...



Besoin d'arguments supplémentaires ?



... donc rien à craindre



Du panoptique à un projet global



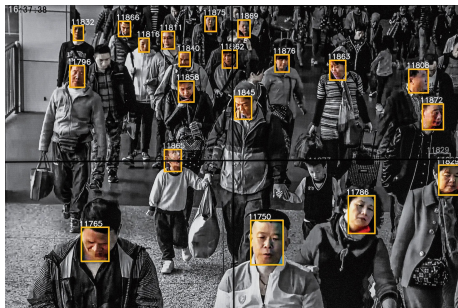
Edward Snowden ✓

@Snowden

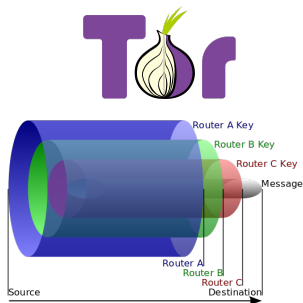
Abonné

Surveillance is not about safety, it's about power. It's about control.

Système de crédit social (chinois)



Category	Evaluation standard	Points
Performance of contract	Failed to perform obligations in accordance with the contract during activities such as borrowing, credit purchase, guarantee, lease, or insurance	-20
	The other party filed a lawsuit and was sentenced for breach of contract	-50
Bank loans	Had overdue records for loans (each)	-10
	Had overdue records for credit cards (each)	-10
Private loans	Had default records	-20
	Investigated and punished by relevant departments for illegal fundraising and illegal lending	-40
	[...]	
Social goods	Voluntary blood donation	+10
	Marrow donation	+50
	Voluntary body or organ donation (add the score for the	+100



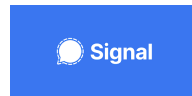
Tor : anonymise le trafic (onion routing, 3 nœuds).

PGP : Pretty Good Privacy

Se mettre en sécurité : outils



Tails : OS “amnésique” (tout passe par Tor, rien n’est gardé).



Signal : messagerie E2EE.

Rappel

Aucun outil n’est infailible : la sécurité est une **chaîne**. Casser le maillon le plus faible suffit.



Même si vous pensez être protégés

Vos conversations WhatsApp sont chiffrées de bout en bout.

Le chiffrement protège le **contenu**... mais pas forcément les **métadonnées**.

Ce qui est protégé

- ▶ contenu des messages
- ▶ contenu des appels
- ▶ fichiers échangés

Ce qui peut fuiter

- ▶ correspondants
- ▶ fréquence et horaires
- ▶ durée des appels
- ▶ volume et taille des échanges

**Même sans lire les messages,
on peut reconstruire des relations entre personnes.**

N. Rastogi, J. Hendler, "WhatsApp security and role of metadata in preserving privacy", 2017.
[arXiv:1701.06817](https://arxiv.org/abs/1701.06817)

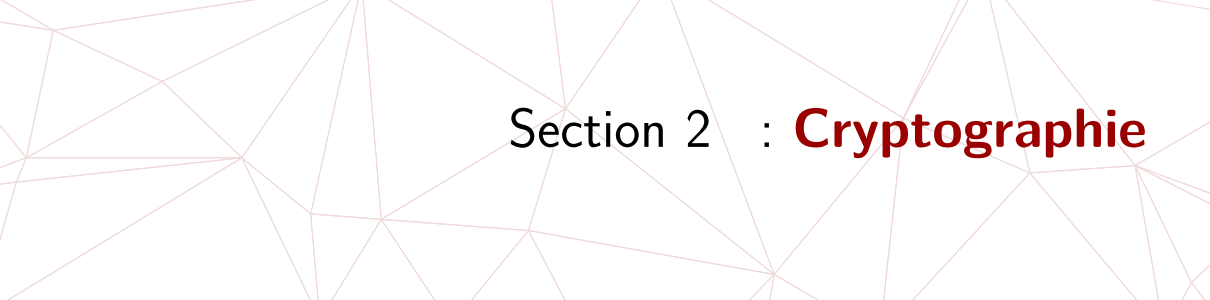
La vérification de l'âge par :

La vérification de l'âge par :

- ▶ validation de la carte de paiement
- ▶ estimation sur la base d'une analyse faciale
- ▶ “ cartes à gratter ”
- ▶ analyse de documents d'identité
- ▶ par inférence



Développement de nouvelles solutions.



Section 2 : **Cryptographie**



Pourquoi utilise-t-on de la cryptographie ?

► Confidentialité



► Authentification



► Intégrité



► Non répudiation



Agit sur une donnée

- ▶ une chaîne caractère ($m \in \{0, 1\}^*$)
- ▶ un fichier texte ($m \in \{0, 1\}^*$)
- ▶ un fichier multimédia ($m \in \{0, 1\}^*$)
- ▶ une archive ($m \in \{0, 1\}^*$)

Appelé aussi le **clair**

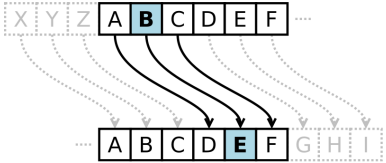
Le but de la protéger le contenu (de différentes manières).

Chiffrement par décalage (César, 50 av. J.-C.)

Décalage de chaque lettre d'un nombre fixe :

$$E_k(x) = (x + k) \bmod 26$$

Exemple avec $k = 3$: A → D, B → E, ...



Chiffrement par substitution (Vigenère, 1586)

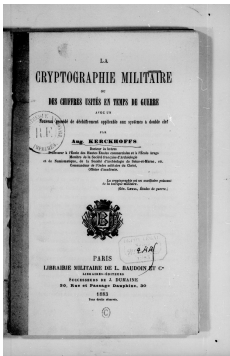
Clé = mot clé répété, décalage variable par position :

$$E_k(x_i) = (x_i + k_{i \bmod |k|}) \bmod 26$$

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

En 1883, un linguiste et cryptographe néerlandais, Auguste Kerchoff von Nieuwenhof, affirmait dans son ouvrage « *La Cryptographie Militaire* » que :

La sécurité d'un système cryptographique doit reposer entièrement sur le secret de la clé, et non sur celui de l'algorithme.



Texte original en français : <https://gallica.bnf.fr/ark:/12148/bd6t57758983.texteImage>

La machine Enigma (1918–1945)

- ▶ **Principe** : rotors électromécaniques, permutation alphabétique à chaque touche
- ▶ **Clé** : position initiale des rotors + connections du tableau de criblage
- ▶ $\sim 10^{20}$ configurations possibles
- ▶ Utilisée par l'Allemagne nazie pendant la Seconde Guerre mondiale

Cryptanalyse

Équipe de Bletchley Park (Alan Turing, Marian Rejewski) : brise Enigma grâce aux faiblesses statistiques et à la **bombe** électromécanique.



Théorie de l'information (Shannon, 1949)

Premier cadre mathématique :

$$H(X) = - \sum_i p(x_i) \log_2 p(x_i)$$

notion de **confusion** et de **diffusion**.

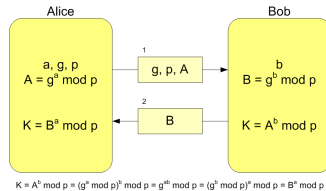


Clé publique / privée (Diffie & Hellman, 1976)

Révolution : échanger une clé sans canal sécurisé.

Alice envoie $g^a \bmod p \rightarrow$ Bob envoie $g^b \bmod p$

Clé partagée : $g^{ab} \bmod p$.



Standards de chiffrement

- ▶ **DES** (1977) : block cipher 64 bits, clé 56 bits — cassé en 1998
- ▶ **AES** (2001) : successeur, 128/192/256 bits de clé
- ▶ **ChaCha20** (2013) : chiffrement par flux, alternative à AES en logiciel

Chiffrement asymétrique

- ▶ **RSA** (1977) : sécurité basée sur la factorisation
- ▶ **ECC** (1985) : courbes elliptiques, clés plus courtes

Période contemporaine

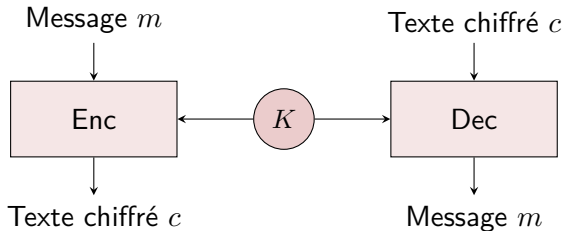
Applications modernes

- ▶ **PGP** (1991) : chiffrement de mails
- ▶ **TLS** (1994–) : navigation sécurisée
- ▶ **Signal Protocol** (2013) : E2EE, Double Ratchet
- ▶ **Messaging** : chiffrement de bout en bout par défaut

Menace quantique

Ordinateurs quantiques → attaque de Shor sur RSA/ECC. Recherche de **post-quantum cryptography** (NIST, 2024).

Principe du chiffrement symétrique



Définition

Une clé secrète K est partagée entre l'émetteur et le récepteur.

$$\text{Enc}(K, m) = c \quad \text{Dec}(K, c) = m \quad \text{Dec}(K, \text{Enc}(K, m)) = m$$

Avantage : rapide, adapté aux gros volumes

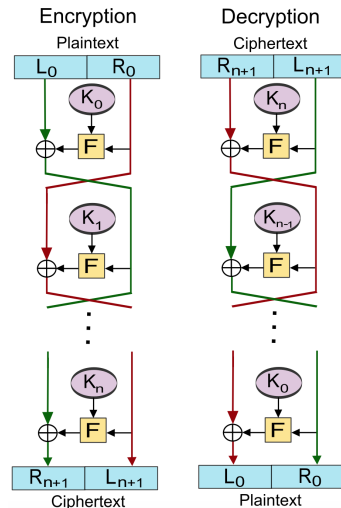
Inconvénient : problème de distribution de la clé

DES (Data Encryption Standard, 1977)

- Bloc de 64 bits, clé de 56 bits
- 16 tours de structure de Feistel
- **Cassé** en 1998 (EFF, $\sim 22h$)

AES (Advanced Encryption Standard, 2001)

- Bloc de 128 bits, clés de 128/192/256 bits
- 10/12/14 tours
- Opérations : SubBytes, ShiftRows, MixColumns, AddRoundKey
- **Standard actuel**



Chiffrement par flux

Génère un **stream de clés** $z_1, z_2, \dots$ à partir d'une graine.

$$c_i = m_i \oplus z_i$$

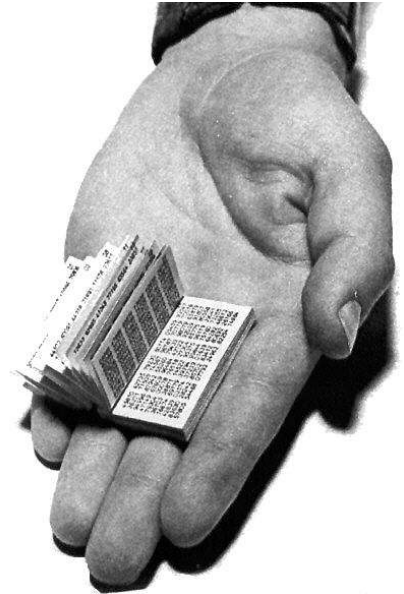
Exemples : RC4 (obsolète), ChaCha20, Salsa20.

One-Time Pad (OTP, Vernam 1917)

Stream = clé aléatoire de même longueur que m , utilisée **une seule fois**.

$$c_i = m_i \oplus k_i \quad k_i \stackrel{\$}{\leftarrow} \{0, 1\}$$

Prouvé sûr (Shannon) : $H(K) \geq H(M)$.



Modes d'opération

Un block cipher ne chiffre qu'un bloc fixes. Les **modes** gèrent les messages plus longs :

Modes simples

ECB Chaque bloc indépendant —
fuit des motifs

CBC IV + chaînage par blocs —
nécessite padding

Modes modernes

CTR Transforme bloc en stream
via compteur

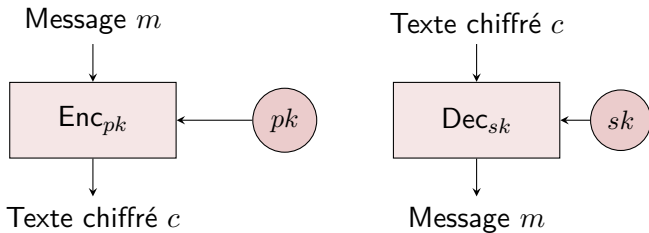
GCM CTR + authentication —
AEAD, recommandé

Règle d'or

Jamais réutiliser un couple (clé, IV/Nonce). En GCM, le nonce doit être **unique** par clé.

Mode	Parallélisable	Authentifié	Padding requis
ECB	Oui	Non	Oui
CBC	Déchiffr.	Non	Oui
CTR	Oui	Non	Non
GCM	Oui	Oui	Non

Principe du chiffrement asymétrique



Définition

Chaque utilisateur possède une paire de clés :

Clé publique pk (connue de tous)

Clé privée sk (secrète)

$$Enc_{pk}(m) = c$$

$$Dec_{sk}(c) = m$$

$$Dec_{sk}(Enc_{pk}(m)) = m$$

Échange de clés Diffie-Hellman

Protocole (1976) — deux utilisateurs sur un canal public :

- 1 Choix public : nombre premier p , générateur g
- 2 Alice choisit $a \xleftarrow{\$} \llbracket 2, p-2 \rrbracket$, envoie $A = g^a \bmod p$
- 3 Bob choisit $b \xleftarrow{\$} \llbracket 2, p-2 \rrbracket$, envoie $B = g^b \bmod p$
- 4 Clé partagée :

$$K = g^{ab} \bmod p = A^b = B^a$$



Sécurité

Problème du logarithme discret : difficile en general (HTTPS, SSH, TLS).

RSA (Rivest–Shamir–Adleman, 1977)

Génération de clés

- 1 Choisir p, q premiers
- 2 $n = p \times q$, $\varphi(n) = (p - 1)(q - 1)$
- 3 e tel que $\gcd(e, \varphi(n)) = 1$
- 4 $d = e^{-1} \bmod \varphi(n)$

Clé publique : (n, e) . Clé privée : (n, d) .

Chiffrement / Déchiffrement

$$\text{Enc}_{(n,e)}(m) = m^e \bmod n = c$$

$$\text{Dec}_{(n,d)}(c) = c^d \bmod n = m$$

Complexité

Sécurité reposant sur la **factorisation** de n . Nécessite des clés de ≥ 2048 bits aujourd'hui.

Propriétés d'une fonction de hachage

`sha256("password") = 5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8`

Définition

$h : \{0, 1\}^* \rightarrow \{0, 1\}^n$ — fonction déterministe, pas de clé.

Préimage

Étant donné y , trouver x tel que $h(x) = y$ est **difficile**. (anti-pré-image à sens unique)

Préimage secondaire

Étant donné x , trouver $x' \neq x$ tel que $h(x) = h(x')$ est **difficile**.

Collision

Trouver (x, x') avec $h(x) = h(x')$ est **difficile**. Attaque de l'anniversaire : $\sim 2^{n/2}$ opérations.

MD5 (Rivest, 1991)

Sortie : 128 bits. **Cassé** (collisions trouvées en 2004). **Ne plus utiliser.**

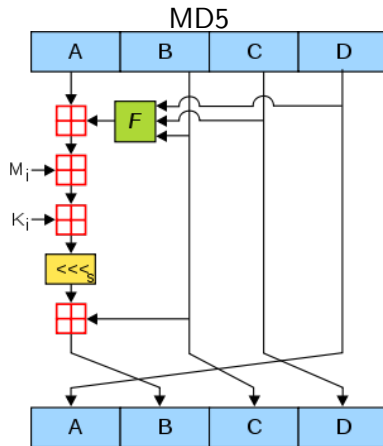
SHA-1 (NSA, 1995)

Sortie : 160 bits. **Cassé** (Google/INRIA, 2017, collision SHattered).

SHA-256 (SHA-2, 2001)

Sortie : 256 bits. **Standard actuel.** Utilisé dans Bitcoin, TLS, signatures numériques.

[Calculateur en ligne](#)



Intégrité de fichier

`sha256sum fichier.zip` Comparer le hash pour détecter une altération.

Stockage de mots de passe

$$\text{hash} = H(\text{sel} \parallel \text{motdepasse})$$

jamais stocker en clair. Utiliser **bcrypt**, **Argon2**.

Applications du hachage

HMAC

Authentification avec clé :

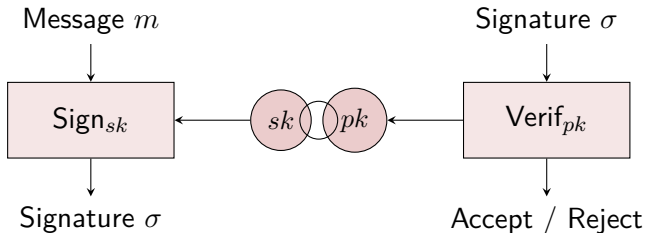
$$\begin{aligned} & \text{HMAC}_K(m) \\ &= H((K \oplus \text{opad}) \parallel H((K \oplus \text{ipad}) \parallel m)) \end{aligned}$$

Utilisé dans les API (JWT), TLS.

Merkle Tree

Arbre de hashes pour vérifier l'intégrité de grandes structures. Utilisé dans les blockchains (Bitcoin).

Principe de la signature électronique



Définition

$$Sign_{sk}(m) = \sigma \quad Verif_{pk}(m, \sigma) \in \{\text{accept}, \text{reject}\}$$

Authentification

L'émetteur est identifié

Intégrité

m n'a pas été modifié

Non-répudiation

L'émetteur ne peut nier

Signatures numériques : algorithmes et sécurité

RSA (signature)

Clé privée (n, d) : $\sigma = H(m)^d \bmod n$

Clé publique (n, e) : vérifier

$$\sigma^e \bmod n \stackrel{?}{=} H(m)$$

DSA / ECDSA

Basé sur le logarithme discret / courbes elliptiques. Utilisé dans TLS, Bitcoin (ECDSA avec secp256k1).

EdDSA (Ed25519)

Signature rapide, résistante aux implémentations malveillantes. Utilisée dans SSH, Signal, Tor.

Attaques courantes

- ▶ Réutilisation de nonce (ECDSA) → clé privée exposée
- ▶ Attaques sur l'implémentation (timing, power analysis)
- ▶ Failles mathématiques (collision MD5 + signature tiers → Flame, 2012)

ECDSA (Elliptic Curve Digital Signature Algorithm) – I

Clés d'Alice :

- ▶ Clé privée : d_A
- ▶ Clé publique : $Q_A = d_A G$

Génération de la signature

- 1 Calculer $e = \text{HASH}(m)$ (fonction de hachage cryptographique, ex. SHA-1).
- 2 Choisir un entier aléatoire $k \in [1, n - 1]$.
- 3 Calculer $(x_1, y_1) = kG$ et $r = x_1 \bmod n$.
Si $r = 0$, recommencer à l'étape 2.
- 4 Calculer $s = k^{-1}(e + rd_A) \bmod n$.
Si $s = 0$, recommencer à l'étape 2.
- 5 La signature est le couple (r, s) .

Vérification de la signature

- 1 Vérifier que $r, s \in [1, n - 1]$.
Sinon, la signature est invalide.
- 2 Calculer $e = \text{HASH}(m)$ (même fonction que pour la signature).
- 3 Calculer $w = s^{-1} \bmod n$.
- 4 Calculer $u_1 = ew \bmod n$ et $u_2 = rw \bmod n$.
- 5 Calculer $(x_1, y_1) = u_1G + u_2Q_A$.
- 6 La signature est **valide** si $r = x_1 \bmod n$.

Certificats X.509 (PKI)

Une **Autorité de Certification** (CA) signe la clé publique :

$$\text{cert} = \text{CA}_{sk}(\text{identité} || pk_{\text{serveur}})$$

Utilisé dans TLS/HTTPS, courrier signé.

Signatures de code

Vérifier l'origine d'un exécutable ou d'un paquet logiciel. Ex. : apt vérifie les signatures des paquets.

PGP / GPG

Chaîne de confiance décentralisée : chaque utilisateur signe les clés de ses contacts.

Documents légaux

Signature électronique (eIDAS en Europe) : valeur juridique équivalente à une signature manuscrite.



Les pièges des implémentations

Timing, padding oracle, réutilisation de nonce/IV, aléa faible, attaques par faute : même les bibliothèques **éprouvées** ont eu des failles (Heartbleed, ...). $\Rightarrow$ **ne pas réimplémenter** soi-même.

Outils pour valider les bibliothèques

Wycheproof (Google & C2SP)

Suite de **tests** des implémentations contre les **attaques connues** (RSA, ECDSA, AES-GCM, ChaCha20, ...).

github.com/C2SP/wycheproof

TLS-Attacker

Framework open source pour analyser et **fuzzer** les implémentations TLS 1.2/1.3.

github.com/tls-attacker/TLS-Attacker

Vie privée :

pas juste une utopie, une nécessité

- ▶ le chiffrement (même de bout en bout) protège le **contenu**, pas les **métadonnées**
- ▶ l'**anonymat** est un enjeu démocratique, pas une preuve de culpabilité

Primitives cryptographiques :

- ▶ **symétrique** : AES, GCM (AEAD), OTP ; **asymétrique** : Diffie–Hellman, RSA
- ▶ **hachage** : SHA-256, HMAC ; **signatures** : ECDSA, Ed25519
- ▶ confidentialité, intégrité, authentification, non-répudiation

Implémentation crypto :

- ▶ ne jamais **implémenter sa propre cryptographie** : utiliser des bibliothèques **éprouvées** (et les **tester**)