Table of contents

Cryptographie et Sécurité de l'Information

Fondamentaux de la Cryptographie

Principe de Kerckhoffs

Sécurité basée sur la clé : La sécurité du système repose uniquement sur le secret de la clé, jamais sur celui de l'algorithme.

Algorithme public : Le système doit rester sûr même si l'algorithme est connu de l'adversaire.

Pas de sécurité par l'obscurité : Kerckhoffs rejette explicitement ce principe dès le XIXe siècle en affirmant que la sécurité doit être mathématiquement démontrable.

Classification des systèmes de cryptage

Inconditionnelle : Sécurité parfaite et théorique, indépendante de la puissance de calcul. Exemple : *one-time pad*.

Provable security : La cryptanalyse est équivalente à la résolution d'un problème mathématique réputé difficile (comme la factorisation pour RSA).

Calculatoire : Sécurité pratique basée sur le coût irréaliste des attaques avec les ressources actuelles. C'est la catégorie la plus utilisée.

Entropie

Entropie : Mesure la quantité d'information effective contenue dans un message, approximant le nombre de bits nécessaires pour l'encoder.

Entropie conditionnelle : Quantifie l'incertitude restante sur le plaintext après avoir observé le ciphertext.

Redondance : Différence entre le codage effectif d'un message et son entropie minimale.

Oracles et Modèles de Sécurité

Oracle Aléatoire : Fonction de hachage "idéale" utilisée dans les preuves théoriques (modèle ROM), qui renvoie des valeurs uniformes et aléatoires.

Oracles CPA/CCA : Simulent un accès aux opérations de chiffrement/déchiffrement avec la clé secrète pour tester la résistance du système face à un adversaire.

IND-CPA : Propriété d'indiscernabilité garantissant qu'un adversaire ne peut distinguer les chiffrés de deux messages différents (équivalent à la sécurité sémantique).

Chiffrement Probabiliste : Ajoute de l'aléa au message pour éviter les attaques par dictionnaire, indispensable en cryptographie asymétrique.

OAEP : Méthode de padding (remplissage) qui ajoute l'aléa nécessaire au chiffrement RSA en combinant le message avec un nombre aléatoire via des fonctions de hachage.

Histoire de la Cryptographie

Historique : Les systèmes classiques reposent sur la substitution (César, Vigenère) et la transposition (permutation des caractères).

One-Time Pad : Seul système à sécurité absolue si la clé est aléatoire, unique et aussi longue que le message. Condition de Shannon : H(K) H(X).

Stéganographie : Technique qui cache l'existence même du message plutôt que de le rendre illisible (exemple : technique des LSB dans les images numériques).

Cryptographie Symétrique

Stream Ciphers

Caractéristiques générales

Stream Ciphers : Cryptage bit par bit en 2 phases (génération keystream + substitution). Taille de bloc = 1 bit.

Avantages :

- Rapides (cryptage au niveau registres)
- Légers (ressources CPU limitées)
- Pas de buffering
- Pas de propagation d'erreurs (retransmission suffisante pour WiFi)

Inconvénients :

- Qualité du keystream critique pour la robustesse
- Réutilisation du keystream = vulnérabilité majeure

Stream Ciphers Synchrones

Synchrone : Keystream dépend uniquement de la clé, indépendant du plaintext/ciphertext. Équations : $\sigma_{i+1} = f(\sigma_i, k)$, $z_i = g(\sigma_i, k)$, $c_i = h(z_i, m_i)$.

Exige synchronisation : Émetteur et récepteur doivent partager même clé ET même état σ_i.

Pas de propagation d'erreur : Modification de c_j affecte uniquement m_j, mais suppression = désynchronisation.

Vulnérable aux modifications de bits : Adversaire peut modifier des bits et analyser l'impact. Nécessite mécanismes d'authentification supplémentaires.

Cas fréquent : Stream cipher additif avec fonction h = XOR (comme one-time pad).

Stream Ciphers Asynchrones

Asynchrone (auto-synchronisé) : Keystream dépend de la clé ET des derniers ciphertexts. État σ_i = buffer de t ciphertexts précédents : $\sigma_i = (c_{i-t}, ..., c_{i-1})$.

Auto-synchronisation : Re-synchronisation automatique après insertion/élimination de ciphertexts grâce au buffer.

Propagation d'erreur limitée : Erreur se propage uniquement sur $\frac{n}{r}$ blocs (n = taille nominale, r = taille plaintexts). Après épuisement du buffer, décryption correcte reprend.

Meilleure diffusion des statistiques : Chaque bit du plaintext influence tous les ciphertexts subséquents. Idéal pour plaintexts redondants ou à faible entropie.

Générateurs LSFR

LSFR : Générateur de keystream long (m bits) depuis clé courte (l bits) avec $l \ll m$. Base = combinaisons linéaires de bits.

Avantages :

- Hardware efficace
- Périodes longues
- Bonne qualité aléatoire

Problème : Sécurité insuffisante comparé aux block ciphers modernes. Vulnérable à l'algorithme de Berlekamp-Massey qui calcule la complexité linéaire et génère arbitrairement de nouvelles séquences.

Solution : NLFSR (Non Linear Feedback Shift Registers) utilisant fonction non linéaire f au lieu de combinaison linéaire.

RC4

RC4 (Rivest Cipher 4, 1987) : Stream cipher logiciel à clé variable, mode synchrone, 10× plus rapide que DES.

Architecture : S-box 8×8 (permutation 0-255 dépendant de la clé) + XOR entre keystream et plaintext. Combinaisons linéaires et non linéaires.

2 étapes :

- KSA (Key Scheduling Algorithm, permutation S-box)
- PRGA (Pseudo Random Generator Algorithm, génération keystream)

Applications : SSL/TLS, Windows, Oracle, Lotus Notes. Usage commercial très répandu.

Vulnérabilité : Protocole WEP (WiFi) complètement cassé à cause d'une faille dans le mode d'utilisation, pas de l'algorithme RC4 lui-même.

Block Ciphers

Caractéristiques générales

Block cipher : Fonction bijective transformant blocs de n bits avec clé K de k bits. Chaque clé définit une bijection différente.

Critères qualité :

- Entropie clé 128 bits (protection brute force)
- Taille bloc 128 bits (éviter dictionnaires plaintext/ciphertext)
- Résistance cryptanalyse = effort brute force

Usage : Confidentialité, authentification, hachage, génération aléatoire (pierre angulaire de la cryptographie).

Modes d'Opération

ECB (Electronic Codebook)

ECB : Chaque bloc encrypté indépendamment avec même clé : $c_i = E_K(m_i)$.

Vulnérabilité majeure : Plaintexts identiques → ciphertexts identiques. Patterns visibles, structure du plaintext transparente.

Avantages :

- Parallélisable
- Pas de propagation d'erreurs

Ne JAMAIS utiliser pour données redondantes.

CBC (Cipher Block Chaining)

CBC : Chaque plaintext XORé avec ciphertext précédent avant encryption : $c_i = E_K(m_i \oplus c_{i-1})$ avec $c_0 = IV$.

Avantages :

- Plaintexts identiques → ciphertexts différents (si IV change)
- Patterns effacés par chaînage
- Propagation erreur limitée (c_j affecte m_j et m_{j+1} uniquement)

IV (Initialization Vector) : Doit être aléatoire ou pseudo-aléatoire, transmissible en clair, différent pour chaque message avec même clé.

Parallélisation : Non parallélisable en encryption (séquentiel), parallélisable en décryption.

CFB (Cipher Feedback)

CFB (asynchrone) : Fonctionne comme stream cipher où keystream dépend des ciphertexts précédents : $c_i = m_i \oplus E_K(c_{i-1})$ avec $c_0 = IV$.

Propagation erreur : Erreur sur c_j affecte $\frac{n}{r}$ blocs suivants (n = taille nominale, r = taille plaintexts).

Non parallélisable. IV non confidentiel mais doit être transmis.

Usage : Adapté aux transmissions avec pertes de paquets fréquentes.

OFB (Output Feedback)

OFB (synchrone) : Stream cipher où keystream dépend uniquement de clé + IV : $z_i = E_K(z_{i-1})$, $c_i = m_i \oplus z_i$ avec $z_0 = IV$.

Avantages :

- Pas de propagation d'erreurs (erreur sur c_j n'affecte que m_j)
- Keystream pré-calculable (parallélisable si pré-calculé)

CRITIQUE : Ne JAMAIS réutiliser même IV avec même clé (sinon keystream identique = vulnérabilité majeure). Modifier l'IV pour chaque nouveau message.

CTR (Counter Mode)

CTR : Keystream généré par encryption d'un compteur incrémenté : $c_i = m_i \oplus E_K(\text{counter} + i)$.

Avantages :

- Parallélisable (encryption + décryption)
- Accès aléatoire à chaque bloc
- Pas de propagation d'erreurs
- SIMD-friendly (pas de dépendances entre blocs)

Compteur : Taille 2^b (b = taille bloc), incrémenter modulo 2^b après chaque itération.

CRITIQUE : Ne jamais réutiliser même compteur avec même clé. Premier bloc du flux $i+1$ > dernier bloc du flux i.

Usage : ATM, IPsec, haut débit, vidéo, transmission sélective de blocs.

Product Ciphers et Feistel

Product cipher : Schéma combinant série de transformations successives (transpositions, substitutions, XOR, multiplications modulaires) pour renforcer résistance cryptanalyse.

Feistel cipher : Product cipher itératif transformant plaintext $2t$ bits $= (L_0, R_0)$ en ciphertext (R_r, L_r) après r rounds (généralement pair et ≥ 3).

Équations round i : $L_i = R_{i-1}$ et $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$ où K_i = sous-clés générées depuis clé principale K.

Décryption : Identique à encryption avec sous-clés appliquées en ordre inverse (K_r à K_1).

Exemple : DES (16 rounds).

DES (Data Encryption Standard)

Structure

DES : Feistel cipher, 64 bits blocs, 56 bits clé effective (64 totaux avec 8 parité), 16 rounds, 16 sous-clés de 48 bits.

Structure : Permutation initiale IP → 16 rounds Feistel → Permutation finale IP [1].

Fonction de round

Fonction f par round :

- Expansion E (32→48 bits)
- XOR avec sous-clé K_i (48 bits)
- 8 S-boxes (48→32 bits, chaque S-box : 6 bits entrée → 4 bits sortie)
- Permutation P (32 bits)

S-box : 6 bits input $a_1a_2a_3a_4a_5a_6$ → ligne = $a_1 + 2a_6$ (bits externes), colonne = $a_2 + 2a_3 + 4a_4 + 8a_5$ (bits internes) → 4 bits output.

Génération sous-clés

Génération sous-clés :

- PC-1 (sélection 56 bits)
- Division C_0, D_0 (28 bits)
- Rotations circulaires gauche (1 ou 2 positions)
- PC-2 (sélection 48 bits pour K_i)

Triple-DES et Sécurité

Vulnérabilité DES : Espace clés 2^{56} cassable en 24h (1999, brute force parallèle, 100'000 PCs).

Triple-DES : $C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$ avec deux clés 56 bits → espace 2^{112}.

Avantages :

- Sécurité satisfaisante
- Réutilise hardware/software DES existant
- Compatibilité
- Migration progressive vers AES

Inconvénient : 3× plus lent (3 exécutions DES successives).

DES groupe : Encryption composée renforce considérablement la sécurité. Si DES était un groupe, recherche exhaustive 2^{56} casserait l'algorithme indépendamment du nombre d'exécutions.

4 clés faibles : Génèrent sous-clés identiques par paires ($k_1 = k_{16}$, $k_2 = k_{15}$, ..., $k_8 = k_9$) → facilite cryptanalyse. 6 paires de clés semi-faibles.

AES (Advanced Encryption Standard)

Structure générale

AES (Rijndael, 2001) : Block cipher itératif (PAS Feistel), 128 bits blocs, clés 128/192/256 bits → 10/12/14 rounds respectivement.

State : Matrice 4×4 bytes (16 bytes = 128 bits). Opérations dans corps $GF(2^8)$ (polynômes degré 7 avec coefficients $GF(2)$).

Opérations par round

4 opérations par round :

- **SubBytes** : Substitution non linéaire via S-box (résistance cryptanalyse linéaire/différentielle)
- **ShiftRows** : Décalages lignes (ligne 0 : 0, ligne 1 : 1, ligne 2 : 2, ligne 3 : 3 positions gauche)
- **MixColumns** : Combinaisons linéaires des colonnes (multiplication matrices dans $GF(2^8)$, diffusion maximale)
- **AddRoundKey** : XOR State avec sous-clé du round

Round final : Identique SAUF pas de MixColumns.

Key Schedule et performances

Key Schedule : Expansion clé → matrice $4 \times 4 \times (N_e + 1)$ bytes (N_e = nombre rounds) → sélection sous-clés (rotations, substitutions S-box, XOR constantes Rcon).

Décryption : Opérations inverses (InvSubBytes, InvShiftRows, InvMixColumns) avec sous-clés en ordre inverse.

Avantages :

- 2× plus rapide que DES
- 10^{22} fois plus sûr (théoriquement)

- Processus ouvert
- Évolutif
- Fonctionne sur cartes 8 bits

Sécurité AES

Forces : Simplicité, performances (même sur plateformes limitées comme cartes à puce 8 bits), optimisations hardware/software.

Attaques publiées :

- **XSL (2002)** : Système 8000 équations quadratiques, 1600 inconnues binaires, effort 2^{100} (conjecture). Contestée car basée sur caractère "fortement algébrique" d'AES
- **Related Key Attacks (2009-2011)** : Résultats intéressants sur versions réduites, ne compromettent pas AES complet
- **Side Channel Attacks** : Sur implémentation (cache timing, power analysis, electromagnetic). Exemple : extraction clé 128 bits avec 6-7 couples plaintext/ciphertext via analyse accès cache (2005)
- **Meet-in-Middle bicyclique (2011-2015)** : Réduit effort AES-128 à 2^{126} (facteur 4 vs brute force), reste largement au-dessus capacités actuelles

Sécurité pratique : Hypothèse clé entropie maximale fondamentale. Attaques récentes (WPA2) = faiblesse passwords/passphrases, pas faille AES. Problème = génération clés depuis passwords faibles.

Techniques de Cryptanalyse

Cryptanalyse Différentielle

Différentielle : Attaque chosen plaintext analysant propagation des différences ($\Delta x = x_a \oplus x_b$) entre plaintexts à travers rounds.

Méthode : Attribuer probabilités aux clés selon changements observés dans ciphertexts. Clé la plus probable = clé correcte après nombreux essais.

Effort DES : 2^{47} couples chosen plaintext.

Très sensible au nombre de rounds : Chances succès augmentent exponentiellement quand rounds diminuent.

Cryptanalyse Linéaire

Linéaire : Attaque known plaintext créant simulateur linéaire du block cipher via approximations linéaires. Bits clé simulateur tendent à coïncider avec clé réelle (calcul probabiliste).

Effort DES :

- 2^{38} known plaintexts $\rightarrow$ 10% probabilité succès
- $2^{43} \rightarrow 85\%$ probabilité

Attaque analytique la plus puissante à ce jour sur block ciphers. Très sensible au nombre de rounds.

Meet-in-the-Middle

Meet-in-Middle : Attaque sur constructions composées $y = E_{K_2}(E_{K_1}(x))$.

Méthode : Construire 2 listes $L_1 = \{E_{K_1}(x)\}$ et $L_2 = \{D_{K_2}(y)\}$ pour tous K_1, K_2. Identifier éléments répétés. Vérifier avec deuxième known plaintext.

Effort DES composé : 2^{57} opérations + 2^{56} blocs stockage « 2^{112} attendu intuitivement.

Applications : Constructions composées, cryptanalyse interne des block ciphers.

Observations communes

Difficultés communes (différentielle/linéaire) :

- Parallélisation moins efficace que brute force parallèle
- Sensibilité rounds

DES et S-boxes : Conjecture répandue = concepteurs DES connaissaient ces attaques (inédites années 1970). Design S-boxes offre résistance très grande aux deux techniques.

Cryptographie Asymétrique

Fondements Mathématiques

Théorie des nombres

Décomposition unique : Tout entier = produit de nombres premiers (à l'ordre près).

$\phi(n)$: Fonction d'Euler comptant les entiers $< n$ premiers avec n.

Clé pour RSA : Si $n = pq$ avec p et q premiers, alors $\phi(n) = (p-1)(q-1)$.

Théorème d'Euler : Si $\gcd(a, n) = 1$, alors $a^{\phi(n)} \equiv 1 \pmod{n}$.

Petit Théorème de Fermat : Cas spécial si p premier : $a^{p-1} \equiv 1 \pmod{p}$.

Inverse modulaire : $a^{-1} \equiv a^{\phi(n)-1} \pmod{n}$. Si p premier, $a^{-1} \equiv a^{p-2} \pmod{p}$.

Base de RSA : Ces théorèmes permettent encryption/decryption avec exposants.

Groupes Multiplicatifs

$\mathbb{Z}_n^*$: Ensemble des éléments premiers avec n, cardinal = $\phi(n)$.

Générateur : Élément d'ordre $\phi(n)$ qui génère tout le groupe par puissances successives.

Crucial pour DH et ElGamal : Sécurité basée sur logarithme discret dans groupe cyclique.

Safe prime : Nombre premier $n = 2p + 1$ avec p également premier. Test de générateur simplifié : α est générateur ssi $\alpha^2 \not\equiv 1 \pmod{n}$ et $\alpha^p \not\equiv 1 \pmod{n}$.

Fast Exponentiation

Idée : Utiliser la représentation binaire de l'exposant pour calculer $a^k \bmod n$ efficacement.

Complexité : $O(\log^3 n)$ - polynomial et très efficace.

Essentiel : Rend RSA, ElGamal, Diffie-Hellman praticables en temps raisonnable.

Alternative : Algorithme d'Euclide étendu pour calculer inverses modulaires, même complexité $O(\log^3 n)$.

Théorème des Restes Chinois

Résout : Systèmes de congruences simultanées avec moduli premiers entre eux.

Solution unique : Modulo produit des moduli. Algorithme de Gauss donne la solution explicite.

Complexité : $O(\log^3 n)$ - polynomial.

Usage cryptographique :

- Optimisation des calculs RSA (utiliser p et q séparément)
- Partage de secret
- Certaines attaques si exposant petit avec messages multiples

Problèmes de Base et Complexité

Problèmes fondamentaux

FACTP (Factorisation) : Factoriser n en nombres premiers $\rightarrow$ base de RSA/Rabin.

DLP (Logarithme Discret) : Trouver x tel que $\alpha^x \equiv \beta \pmod{p}$ $\rightarrow$ base ElGamal/Diffie-Hellman.

SQROOTP (Racine Carrée mod Composite) : Trouver $\sqrt{a} \bmod n$ avec n composite $\rightarrow$ base de Rabin.

Équivalences prouvées : Casser l'algorithme = résoudre le problème de base correspondant.

Techniques de Factorisation

Sous-exponentiel : NFS (Number Field Sieve) actuellement le plus rapide, complexité $O(\exp(c \cdot (\ln(n))^{1/3}))$.

Record 2020 : RSA-829 (829 bits, 250 chiffres) factorisé en 2700 années-cœur avec GNFS.

Recommandation : Clés RSA 2048 bits minimum (3072-4096 bits pour sécurité long terme).

Menace future : Ordinateurs quantiques avec algorithme de Shor (complexité polynomiale $O(\log^c n)$).

RSA

Principe

Clé publique : (n, e) avec $n = pq$ (produit de deux grands nombres premiers).

Clé privée : d tel que $ed \equiv 1 \pmod{\phi(n)}$.

Chiffrement : $c = m^e \bmod n$.

Déchiffrement : $m = c^d \bmod n$.

Sécurité : Basée sur difficulté de factoriser n. Trouver d factoriser n (équivalence prouvée).

Taille recommandée : $n \geq 2048$ bits minimum.

Choix des paramètres

e **petit** : Souvent 3, 17 ou 65537 pour encryption rapide, mais nécessite padding obligatoire.

d **grand** : Doit être au moins taille$(n)/2$ pour sécurité. Exposant de décryption toujours grand.

Clés séparées : Utiliser des paires de clés distinctes pour encryption et signature.

Attaques sur RSA

Même message, petit e : Si message identique envoyé à plusieurs destinataires avec $e = 3$, le Théorème des Restes Chinois permet d'extraire m directement par racine cubique.

Message trop petit : Si $m < n^{1/e}$, alors $c = m^e$ dans $\mathbb{Z}$ (pas modulo) $\rightarrow$ racine e-ième directe possible.

Propriété multiplicative : $E(m_1) \cdot E(m_2) = E(m_1 \cdot m_2) \bmod n$. Permet attaques chosen-ciphertext et blind signatures.

Protection essentielle : Toujours utiliser padding/randomization (standard OAEP) avant encryption pour éviter ces attaques.

Attaque générale : Méthode la plus efficace reste la factorisation de n si paramètres bien choisis et implémentation correcte.

ElGamal

Principe

Base : Problème du logarithme discret (DLP) dans $\mathbb{Z}_p^*$.

Clé publique : (p, α, y) avec p premier, α générateur, $y = \alpha^a \bmod p$.

Clé privée : a (exposant secret).

Chiffrement : Pour message m, choisir k aléatoire unique, calculer $\gamma = \alpha^k \bmod p$ et $\delta = m \cdot y^k \bmod p$. Envoyer (γ, δ).

Déchiffrement : $m = \delta \cdot \gamma^{-a} \bmod p$.

Propriétés et limites

Sécurité : k doit être unique et grand pour chaque message.

Inconvénient majeur : Double la taille du message (expansion 1:2).

Équivalence : Basé sur DLP mais équivalence stricte non prouvée (contrairement à Rabin).

k **unique CRITIQUE** : Si k réutilisé pour deux messages, $\delta_1/\delta_2 = m_1/m_2$ révèle les messages.

Taille exposants : k et a doivent être grands, sinon vulnérable aux algorithmes baby-step giant-step.

Extensions : Peut se généraliser aux corps $GF(2^n)$ ou aux courbes elliptiques.

Rabin

Principe

Base : Problème SQROOTP (racine carrée modulo composite).

Clé publique : $n = pq$ (produit de deux grands nombres premiers).

Clé privée : (p, q) les facteurs premiers.

Chiffrement : $c = m^2 \bmod n$.

Déchiffrement : Calculer les 4 racines carrées possibles de $c \bmod n$ via algorithmes efficaces modulo p et q.

Sécurité

Avantage unique : Seul algorithme avec équivalence PROUVÉE à la factorisation (SQROOTP FACTP). Catégorie “provably secure”.

Problème : 4 solutions possibles au déchiffrement, nécessite mécanisme de redondance ou indication pour identifier le message correct.

Attaque chosen-ciphertext : Si adversaire M envoie $c = m^2 \bmod n$ et reçoit racine $m_x \neq m$, alors $\gcd(m - m_x, n)$ donne un facteur de n (probabilité 0.5 de succès).

Parade : Exiger redondance suffisante dans les messages permettant d’identifier sans ambiguïté la solution correcte et rejeter les autres.

Courbes Elliptiques

Structure mathématique

Équation : $y^2 = x^3 + ax + b$ avec discriminant $4a^3 + 27b^2 \neq 0$.

Structure : Forme un groupe commutatif avec point à l'infini $\mathcal{O}$ comme élément identité.

Opération fondamentale : Addition géométrique de points (3ème point d'intersection + symétrie).

Inverse : $-P = (x, -y)$ (symétrie par rapport à l'axe des x).

Addition : Tracer droite entre P et Q, trouver 3ème point d'intersection, prendre son symétrique.

Doublement : Utiliser tangente au point P au lieu de droite entre deux points.

Sécurité et avantages

Problème dur : ECDLP (Elliptic Curve Discrete Logarithm Problem) - trouver k dans $Q = kP$ nécessite effort exponentiel.

Gain majeur : Clés environ 6-10× plus courtes que RSA/DH classique pour sécuritééquivalente.

Limite : Représenter messages en points de la courbe reste opération complexe.

Adoption : NSA a acheté brevet Certicom en 2003 pour utilisation cryptographique.

Comparaison des tailles de clés

Pour sécurité équivalente à AES 128 bits :

- RSA nécessite 3072 bits
- Courbes elliptiques seulement 256 bits (rapport 1:12)

Pour sécurité équivalente à AES 256 bits :

- RSA nécessite 15360 bits
- Courbes elliptiques seulement 512 bits (rapport 1:30)

ElGamal sur Courbes Elliptiques

Principe : Adaptation directe d'ElGamal en remplaçant opérations dans $\mathbb{Z}_p^*$ par opérations sur courbe E_p.

Clé publique : (E_p, P_0, P_a) avec P_0 point de grand ordre, $P_a = xP_0$.

Clé privée : x (scalaire secret).

Chiffrement : Pour message $m_i \in E_p$, choisir k aléatoire, calculer $\gamma = kP_0$ et $\delta = kP_a + m_i$. Envoyer (γ, δ).

Déchiffrement : $m_i = \delta - x\gamma$.

Opérations : Addition de points et multiplication scalaire sur courbe elliptique.

Sécurité : Repose sur ECDLP (difficulté de calculer logarithme discret sur courbe).

Authentification : Nécessaire pour éviter attaques man-in-the-middle, comme pour ElGamal classique.

Avantage : Clés beaucoup plus courtes pour sécurité équivalente (facteur 6-30).

Fonctions de Hachage et Intégrité

Fonctions à Sens Unique

OWF : facile dans un sens ($f(x) \to y$), impossible dans l'autre ($y \to x$).

Exemples :

- Carrés modulaires
- $E_k(x) \oplus x$

OWF OWHF (hash functions = plus de contraintes).

Hash Functions

Types et propriétés

Hash function : compression + calcul facile

MDC (sans clé) pour intégrité

MAC (avec clé) pour authentification

Propriétés :

1. preimage resistance

2. 2nd-preimage resistance
3. collision resistance

OWHF = (1)+(2)

CRHF = (2)+(3)

Message Authentication Codes

MAC = hash avec clé k

Sans k : impossible de forger $(x, h_k(x))$ ou retrouver k

Garantit authentification d'origine + intégrité.

Attaques sur MDCs

Pour casser 2nd-preimage resistance avec digest de m bits : 2^{m-1} essais (prob 0.5).

Birthday paradox : pour casser collision resistance avec digest de m bits : $2^{m/2}$ essais (prob > 0.5).

Exemple : 23 personnes suffisent pour anniversaires identiques.

Résistance Calculatoire

Efforts :

- preimage 2^n
- 2nd-preimage 2^{n-1}
- collision $2^{n/2}$

Tailles :

- OWHF 128 bits
- CRHF 256 bits
- MAC clé 256 bits

MDCs Basés sur Cryptage

MDCs à partir de crypto symétrique : casser réversibilité + chaînage XOR.

Modèles :

- Matyas-Meyer-Oseas
- Davies-Meyer
- Miyaguchi-Preneel

MDC-2/4 avec DES → 128 bits.

Customized MDCs

Customized MDCs :

- MD5 (cassé)
- SHA-0 (cassé)
- SHA-1 (faible)
- SHA-2 (sûr)
- SHA-3/Keccak (standard actuel)

Construction : padding + constantes + rounds + chaînage.

MACs Basés sur Cryptage

CBC-MAC : mode CBC + IV=0, seul dernier bloc gardé. DES : clé 56/112 bits, MAC 64 bits.

Nested MACs et HMACs

HMAC : double hash avec clés dérivées (`ipad/opad`).

$\text{HMAC}_k(x) = H((k \oplus \text{opad}) \parallel H((k \oplus \text{ipad}) \parallel x))$

Standard, sûr, performant.

Applications

Intégrité

Intégrité : MAC seul, MDC+crypto, MDC+signature.

Vulnérable aux replay sans timestamps/nonces.

Blockchain

Blockchain : chaînage de blocs via hash.

Proof of Work : trouver nonce pour hash < target.

Sécurité = effort > tous mineurs.

Bitcoin : ~10 min/bloc, 10^{21} hash/sec.

Autres applications

Applications :

- authentification
- virus checking
- distribution clés publiques
- timestamp
- one-time passwords (chaîne de hash)

UNIX Salt

UNIX salt : 12 bits aléatoires ajoutés au password avant hash.

4096 variations possibles.

Empêche codebooks pré-calculés et détection duplications.

Signatures Digitales

Définitions et Classifications

Signature digitale = chaîne associant message + entité

Deux types :

- avec appendice (nécessite message original)
- avec reconstitution (reconstruit le message)

Basée sur crypto asymétrique

Signatures avec Appendice

Signature : $S_A(m_h) = s$ (clé privée).

Vérification : $V_A(m_h, s)$ (clé publique).

Impossible de forger sans clé privée.

Signatures avec Reconstitution

Avec reconstitution : Fonction redondance $R(m) = m_R$.

Signature $s = S_A(m_R)$.

Vérification : $m_R = V_A(s)$, reconstituer $m = R^{-1}(m_R)$.

Redondance cruciale pour sécurité.

Signature RSA

RSA signature : $s = m_R^d \bmod n$ (privée).

Vérif : $m'_R = s^e \bmod n$ (publique).

Avec appendice : $s = H(m)^d \bmod n$.

Signature lente, vérif rapide.

Signatures Aveugles

Blind signature : Exploite multiplicativité RSA.

Camouflage $f(m) = m \cdot k^e$

Décamouflage $g(m) = k^{-1} \cdot m$.

B signe $f(m)$, A obtient $S_B(m)$ sans que B voie m.

Signature Rabin

Rabin : $s = \sqrt{m_R} \bmod n$.

Vérif : $m'_R = s^2 \bmod n$.

Provably secure (équivalent factorisation).

Vulnérable attaques actives chosen-ciphertext.

Signature ElGamal

ElGamal : (r, s) avec $r = \alpha^k \bmod p$, $s = k^{-1}(m_h - ar) \bmod (p-1)$.

Vérif : $y^r r^s \stackrel{?}{=} \alpha^{H(m)} \bmod p$.

Base de DSA.

k unique crucial.

Signatures et Crypto-monnaies

Crypto-monnaies : Bitcoin/Ethereum utilisent **ECDSA** (ElGamal sur courbes elliptiques).

Chaque transaction signée avec clé privée détenteur.

Sécurité basée ECDLP.

Tableau Récapitulatif

Signatures classiques :

- RSA/Rabin (recovery)
- ElGamal/DSS (appendice)

Spécialisées :

- One-time
- Undeniable
- Fail-Stop
- Blind

Problèmes base : RSAP, SQROOTP, DLP, dépend de la OWF.

Types d'Attaques

Casser signature :

- Total break (clé privée)
- falsification sélective (message fixé)
- existentielle (un message)

Attaques :

- key-only
- known/chosen/adaptive-chosen-messages

Authentification

Méthodes d'Authentification de Messages

4 méthodes :

- MAC seul
- MDC+cryptage
- MDC+cryptage confidentiel
- MDC+signature

Attention : Vulnérable aux replay attacks sans mécanisme temporel

Authentification d'Entités

Niveaux d'authentification

3 niveaux :

- Faible (révèle secret)
- Forte (preuve de possession)
- Zero-knowledge (aucune info révélée)

4 objectifs :

- Acceptation si honnête
- non-réutilisation
- résistance usurpation
- résistance observation

Attaques Dictionnaire

Offline (via BdD ou capture) > **Online** (limitée par le système)

Protection :

- salting
- limitation tentatives
- authentification forte

Plaintext-Equivalence

Plaintext-equivalent : Donnée utilisable comme le password original

Danger : Si le système transmet `H(p)` et stocke `H(p)` → `H(p)` est plaintext-equivalent

Bon design : Système transmet `p`, stocke `H(p)` → pas plaintext-equivalent

Authentification Faible

Types et stockage

2 types :

- Password fixe (statique)
- Password variable (change à chaque instance)

Stockage :

- Clair (très vulnérable)
- Encrypté/Hashé (attaques offline)

Protections :

- Règles strictes
- limitation tentatives
- salting
- non-diffusion

Méthodes spécifiques

Lamport : wn+1 = H(wn), authentification par vérification de la chaîne de hash

Hardware : Générateur synchronisé (30-60s), limité au pre-play

Attention : Nécessite authentification de B pour éviter pre-play et small-n attacks

Authentification Forte Symétrique

Challenge-Response : B envoie challenge R, A répond avec E_k(R)

Alternative : MAC au lieu d'encryption (plus rapide)

Avec timestamp : Un message en moins mais nécessite synchronisation horloges

Reflection attack : Utiliser la réponse d'une session pour en authentifier une autre

Protection : Inclure identités + asymétrie dans challenges (R1,R2) vs (R2,R1)

Authentification Forte Asymétrique

Vulnérabilité : Chosen-ciphertext attacks si pas de structure

Protection : Inclure H(R), identité B dans le message encrypté, A vérifie avant de révéler R

Needham-Schroeder : 3 messages avec inclusion identités pour éviter chosen-ciphertext

Zero-Knowledge Proofs

Propriétés fondamentales

3 propriétés :

- Consistance (accepte si honnête)
- Significativité (nécessite secret)
- Zero-knowledge (aucune info révélée)

Structure : Témoin → Défi → Réponse (répéter n fois)

Perfect ZK : Indistinguable d'une simulation même avec ressources infinies

Exemple de la caverne

Caverne : A entre aléatoirement (y ou z), B demande sortie (gauche/droite)

Probabilité triche : 2^{-n} après n répétitions

ZK : B vérifie connaissance mais n'apprend pas le secret, ne peut convaincre tierce partie

ZKIP Isomorphisme de Graphes

Problème : Trouver permutation entre 2 graphes isomorphes = difficile

Protocole : A crée H aléatoire, B demande permutation vers G1 ou G2, A répond

Perfect ZK : Transcriptions indistinguables d'un simulateur

ZKIP Fiat-Shamir

Secret : s tel que $v = s^2 \bmod n$ (clé publique)

Protocole :

- Témoin r^2
- défi $e \in \{0, 1\}$
- réponse $y = r \cdot s^e$

Vérification : $y^2 \equiv x \cdot v^e \pmod{n}$

Perfect ZK : Paires (x,y) simulables par B

Implantations Pratiques

FSS : Témoins/défis multiples → probabilité 2^{-nk}

GQ : Domaine de défis élargi → moins d'échanges

Schnorr : DLP + grands défis → **3 échanges seulement**

Tous : Plus efficaces que RSA, adaptés aux cartes à puces

Attaque Mafia

Attaque Mafia : Relais des messages via complice → authentification frauduleuse transparente

Protections :

- Cage Faraday
- synchronisation forte
- distance bounding

Attaque	Protection

Tableau Attaques et Protections

Attaque	Protection
replay	zero-knowledge, challenge-response, one-time password
known/chosen-plaintext	zero-knowledge
chosen-ciphertext	zero-knowledge, témoin + structure
reflection	inclure identités, asymétrie messages
interleaving	inclure identités, chaînage cryptographique
collusion	cage Faraday, synchronisation forte

Établissement de Clés

KEP (Key Establishment Protocol)

Définitions

Mécanisme pour partager un secret destiné aux échanges cryptographiques.

Types de protocoles :

- **KTP** : Une entité crée et transmet la clé
- **KAP** : Les entités dérivent conjointement la clé
- **Pré-distribution** : Clés déterminées à priori
- **DKE (Dynamic Key Establishment)** : Clés changeant à chaque exécution

Propriétés d'authentification

Propriétés d'authentification :

- **Implicit key authentication** : Assurance que seul le correspondant peut accéder à la clé
- **Key confirmation** : Assurance que le correspondant possède effectivement la clé
- **Explicit key authentication** : Implicit + confirmation

Propriétés de sécurité

Propriétés de sécurité :

- **PFS (Perfect Forward Secrecy)** : Clés passées protégées même si clés long terme compromises
- **Future Secrecy** : Clés futures protégées même si compromission par attaquant passif
- **Deniability** : Participation non prouvable à un tiers

Protocoles Symétriques

KAP symétrique trivial (pré-distribution)

$n(n-1)/2$ clés pour n utilisateurs

Inconditionnellement sûr

Problème : $O(n^2)$ en stockage, $O(n)$ clés par utilisateur

KAP symétrique DKE simple

$K := E_S(r_a \oplus r_b)$

Propriétés :

- Implicit key authentication
- Entity authentication, key confirmation, PFS

AKEP2

Utilise MACs pour authentification, clé dérivée $K := h'_{S'}(r_b)$

Propriétés :

- Entity authentication mutuelle
- Implicit key authentication
- Key confirmation, PFS

Protocoles Asymétriques

Diffie-Hellman

$K := \alpha^{xy} \mod p$ calculée indépendamment par A et B

Sûr contre attaques passives (DHP DLP)

Vulnérable Man-in-the-Middle sans authentification

Générer clés symétriques : $K_{sym} := \text{SHA}(K)$

Propriétés :

- Entity authentication, implicit key authentication, key confirmation

Station to Station (STS)

DH authentifié avec signatures numériques

Propriétés :

- Entity authentication mutuelle
- Implicit + explicit key authentication
- PFS : clés session passées protégées même si clé signature compromise

Utilisé dans IPv6

OTR/Signal

Protocoles pour messagerie instantanée

SIGMA : signature + MAC

KDF génère K_e (chiffrement) et K_m (MAC)

Révélation anciennes clés MAC $\rightarrow$ répudiabilité

Propriétés :

- PFS, future secrecy, repudiability

Utilisés : WhatsApp, Facebook Messenger

SRP (Secure Remote Password)

KAP asymétrique basé mot de passe

B stocke vérificateur $v := \alpha^x$ (pas le password)

Résiste aux attaques dictionnaire

Propriétés :

- Toutes propriétés KEP

Attaques et Vulnérabilités

Logjam (2015)

Attaque sur DH :

- Downgrade vers groupes 512 bits via MIM
- Pré-calcul (1 semaine) + calcul individuel (1 minute)
- Réutilisation si même premier p

Acteurs étatiques peuvent casser PFS sur 1024 bits

KTP Asymétrique

KTP symétrique trivial

$K := r_a$ avec $E_S(r_a)$

Propriétés :

- Implicit key authentication
- Entity authentication, key confirmation, PFS

Shamir's No-key Protocol

Transport via exponentiations successives K^a, $(K^a)^b$, K^b

Vulnérable Man-in-the-Middle

Needham-Schroeder asymétrique

$K := H(k_1, k_2)$ avec échanges encryptés par clés publiques

Propriétés :

- Entity authentication, implicit key authentication, key confirmation
- PFS

EKE (Encrypted Key Exchange)

Protocole mixte symétrique + asymétrique

Password + crypto asymétrique

Robuste même si password faible

Propriétés :

- PFS si clés régénérées à chaque fois

Protocoles avec KDC

Needham-Schroeder symétrique (avec KDC)

KDC génère et distribue k_{AB}

Vulnérable replay et known-key attacks

Solution : ajouter timestamp

Base de Kerberos

Kerberos

Authentification et distribution de clés avec KDC

AS émet TGT (Ticket Granting Ticket)

TGS émet tickets pour services

Tickets contiennent clés de session encryptées

Authentification via authenticators

Propriétés :

- Entity authentication, implicit key authentication
- Key confirmation partielle
- PFS

Vulnérabilités : password guessing, replay attacks

Solutions : pré-authentification, timestamps

SSL/TLS

Meta-protocole entre TCP et couche application

Services :

- confidentialité
- intégrité
- authentification
- identification serveur

Handshake : négociation paramètres + authentification par certificats

Clés dérivées par cascade : $pre\_master\_secret \rightarrow master\_secret \rightarrow key\_block$

Propriétés :

- Entity authentication, implicit key authentication, key confirmation
- PFS dépend du protocole d'échange (DH oui, RSA non)

Standard HTTPS

Failles notables : génération aléatoire, heartbleed, renégociation

Bonnes pratiques KEP

Bonnes pratiques KEP :

- Définir objectifs (confidentialité, authentification, non-répudiation)
- Définir niveau de sécurité souhaité (key confirmation, PFS)
- Établir contraintes environnement
- Choisir solution prouvée et robuste
- Vérifier propriétés : analyse pratique + formelle
- Éviter pièges : reflection attacks, contrôle nombres aléatoires, redondance quantités encryptées

Tiers de Confiance (TTP)

Modes TTP

Modes TTP :

- **In-line** : TTP intermédiaire relaye tous les échanges
- **On-line** : TTP participe en temps réel, A et B communiquent directement
- **Off-line** : TTP ne participe pas en temps réel, info disponible à priori (ex: CA)

Off-line : pas de disponibilité permanente requise mais révocation plus complexe

KDC (Key Distribution Center)

Résout problème distribution : n^2 clés $\rightarrow$ n clés seulement

Scalable : +1 entité = +1 clé

Clés de session générées dynamiquement

Risques :

- single point of failure (sécurité + opérationnel)
- performance bottleneck

Solutions : mirroring, répartition charge

CA (Certification Authority)

Rôle et fonctionnement

Authentifie association entité clé publique

Crée et signe certificats

Mode off-line

CRLs (Certificate Revocation Lists) pour certificats invalides

Compromission CA = conséquences graves

PoP (Proof of Possession)

Vérifier possession clé privée (pas seulement identité)

Sans PoP : attaquant peut usurper identité via certificat frauduleux

Protocole : CA envoie challenge A, r, vérifie $S_{priv_A}(A, r)$

Introduit niveaux de confiance pour CAs

Separation of Duties

Certificats CRLs signés avec clés différentes

CA (certification) Revocation Authority (révocation)

Machines et security policies séparées

Évite attaques post-compromission clé CA

Entités fonctionnelles certification

Entités fonctionnelles certification :

- **Name Server** : gestion noms uniques + DNSSec
- **RA (Registration Authority)** : contact direct, vérifications identité/PoP
- **Key Generator** : génération paires clés (perd non-répudiation car clé privée connue)
- **Certificate Directory** : accès lecture certificats

Autres TTPs

Autres TTPs :

- **TA (Timestamp Agent)** : certifie existence document à moment précis
- **Notary Agent** : TA + validité/origine (support non-répudiation)
- **KEA (Key Escrow Agent)** : accès clés session sous conditions légales

Exemple historique : Clipper/Capstone/Fortezza (controversé)

Certificats

Structure

Certificat - Structure :

- **Issuer** : identité CA signataire
- **Subject** : nom entité certifiée
- **Subject Public Key** : clé publique (ex: RSA (n, e), DH (p, α, α^x))
- **Subject Public Key Algorithm** : RSA, DH, etc.
- **Validity** : période validité (UTC)
- **Signature** : CA signe tout, garantit authenticité

CRLs (Certificate Revocation Lists)

Listes certificats devenus invalides (clé compromise, changement algorithme, etc.)

Structure : issuer, dates émission, serial numbers révoqués, signature

Publication fréquente requise (large audience)

Talon d'Achille des systèmes PKI

Alternative : certificats validité très courte (quelques minutes) $\rightarrow$ retour mode on-line

Arbres d'authentification

Alternative certification via hash functions + structure arbre

Seule racine R nécessite signature

Vérification : fournir chemin depuis feuille ($\sim \log_2 n$ valeurs)

Construction : feuilles Y_i, arcs $h(Y_i)$, nœuds $h(h_1||h_2)$

Application principale : timestamping

TA publie R quotidiennement (journal) pour empêcher triche

Topologies certification

Topologies certification :

- **Cross-certification** : CA_A certifie pub_{CA_B}
- Chaîne : $CA_A\{CA_B\} \rightarrow CA_B\{B\}$
- **Modèle hiérarchique** (PEM/X.509) : racine universelle, clé publique supposée connue mondialement
- **Modèle graphe** (PGP) : utilisateurs agissent comme CAs, décentralisé
- **Hybride** : hiérarchie + cross-certification bidirectionnelle

Règle d'or : chaînes courtes (maillon le plus faible)

PKI - Infrastructure

Entités principales

PKI - Entités principales :

- **CA** : création et maintenance certificats
- **Certificate Repository** : stockage accessible (X.500, LDAP, WWW, DNS)
- **Revocation** : gestion CRLs
- **Key Backup/Recovery** : sauvegarde clés perdues (clés décryption uniquement, pas signature)
- **Automatic Key Update** : renouvellement clés
- **Key/Certificate History** : récupération clés obsolètes pour anciens documents
- **Cross-Certification** : validation certificats autres PKIs
- **Support Non-Répudiation** : data origin authentication, time-stamped signatures, signed receipts
- **Secure Time Stamping** : référence temps acceptée par tous
- **Logiciel Client** : opérations utilisateur (gestion certificats, signatures, périphériques)

Avantages

PKI - Avantages :

- **Sécurité** : environnement intégré sans maillons faibles
- **Tout-en-un** : multi-services (authentification forte, signatures, single sign-on, VPNs, B2C/B2B)
- **Interopérabilité** : standards répandus (X.509, PKCS, OCSP), compatibilité inter-entreprise

Inconvénients

PKI - Inconvénients :

- **Coût** : produits chers, compétences rares
- **Complexité** : mise en œuvre et gestion

Solution : sous-traitance service PKI