Résumé de Révision
Sécurité Informatique

Cours de Sécurité Informatique - Niveau Universitaire
Préparation Examens Janvier 2024 & 2025

12 janvier 2026

Document complet de révision couvrant les 6 chapitres du cours

Table des matières

1 Cryptographie Symétrique

1.1 Perfect Security (One-Time Pad)

Définition

Perfect Security : Un chiffrement a la perfect security si :

$$\Pr[M = m \mid C = c] = \Pr[M = m] \quad \forall m, c$$

L'observation du chiffré ne donne AUCUNE information sur le message.

One-Time Pad (OTP) :

- Chiffrement : $c = m \oplus k$
- Déchiffrement : $m = c \oplus k$
- **Conditions** :
  - Clé aussi longue que le message : $|K| = |M|$
  - Clé uniformément aléatoire et usage unique

Théorème de Shannon : Si $|M| > |K|$, alors pas de perfect security possible.

Limitations OTP :

- Clé aussi longue que le message (impraticable)
- Clé à usage unique (distribution difficile)
- Sécurité inconditionnelle (même contre adversaire avec puissance infinie)

1.2 Sécurité Computationnelle

Secure PRG (Pseudo-Random Generator) :

$$\text{PRGadv}[A, G] := |\Pr[A(G(s)) = 1] - \Pr[A(r) = 1]|$$

où $s \leftarrow S$ (seed), $r \leftarrow R$ (vraiment aléatoire)

Définition

G est un **Secure PRG** si $\text{PRGadv}[A, G]$ est négligeable pour tout adversaire efficace (polynomial-time).

Stream Cipher : $c = m \oplus \text{PRG}(k)$
Exemples : RC4 (cassé), ChaCha20 (moderne)

1.3 Block Ciphers

AES (Advanced Encryption Standard) :

- Taille de bloc : 128 bits
- Tailles de clés : 128, 192, 256 bits
- Structure : Substitution-Permutation Network (SPN)

Modes d'opération :

Mode	Chiffrement	Parallèle	IV	Sécurité
ECB	$c_i = E(k, m_i)$	Oui	Non	Patterns visibles
CBC	$c_i = E(k, m_i \oplus c_{i-1})$	Non	Oui	Padding oracle
CTR	$c_i = m_i \oplus E(k, IV + i)$	Oui	Oui	CPA-secure

CPA Security : L'adversaire peut choisir des messages et obtenir leurs chiffrés. CTR mode est CPA-secure, ECB ne l'est pas.

2 Intégrité des Messages

2.1 Fonctions de Hachage Cryptographiques

Propriétés requises :

1. **Collision Resistance** : Difficile de trouver $m_1 \neq m_2$ tel que $H(m_1) = H(m_2)$
2. **Pre-image Resistance** : Difficile de trouver m tel que $H(m) = h$ (one-way)
3. **Second Pre-image Resistance** : Difficile de trouver $m_2 \neq m_1$ tel que $H(m_1) = H(m_2)$

Exemples :

— SHA-256 : 256 bits, utilisé massivement

— SHA-3 (Keccak) : Différente structure (sponge construction)

— MD5, SHA-1 : CASSÉS, ne plus utiliser

2.2 MAC (Message Authentication Code)

Objectif : Authentifier l'intégrité d'un message avec une clé secrète partagée.
HMAC (Hash-based MAC) :

$$\text{HMAC}_k(m) = H\left((k \oplus \text{opad}) \| H((k \oplus \text{ipad}) \| m)\right)$$

Résiste aux collisions de H sous-jacent.

2.3 AEAD (Authenticated Encryption with Associated Data)

AES-GCM : Combine chiffrement (CTR) + authentification (GMAC)

— Chiffre et authentifie en une passe

— Données associées (AD) : métadonnées non chiffrées mais authentifiées

3 Protocoles d'Authentification

3.1 Hachage de Mots de Passe

Avertissement

Jamais faire : `hash = SHA256(password)` (vulnérable aux rainbow tables !)
Toujours faire :

```
salt = random(16 bytes)
hash = bcrypt(password, salt, cost=12)
```

Salting :

— Salt unique par utilisateur

— Rend les rainbow tables inutiles

— Stockage : (salt, hash)

Password KDF (Key Derivation Functions) :

Algorithme	Coût	Résistance	Recommandation
SHA-256	Très rapide	Faible	Jamais utiliser
PBKDF2	Itérations (100k+)	Moyen	Acceptable
bcrypt	Cost (12-14)	Bon	**Recommandé**
scrypt	Mémoire + CPU	Excellent	Recommandé
Argon2	Mémoire + CPU	Meilleur	**Standard moderne**

Benchmark bcrypt (cost = 12) : 300ms par hash sur CPU moderne

3.2 Rainbow Tables et Tables de Hellman

Problème : Comment casser un hash sans sel efficacement ?
Approches naïves :

1. **Attaque en temps** : Calculer $H(w)$ pour chaque tentative (très lent)
2. **Table complète** : Stocker $(w, H(w))$ pour tous les mots (énorme mémoire, 400 GB)

Solution : Compromis Temps-Mémoire (TMTO)

3.2.1 Tables de Hellman (1980)

Principe : Chaînes de réduction avec UNE fonction R

```
p  --H--> h  --R--> p  --H--> h  --R--> p  --H--> ... --R--> p
```

Stockage : Seulement (p_0, p_t) pour chaque chaîne
Paramètres :

— m chaînes de longueur t

— Couverture : $\sim 0.5 \times m \times t$ mots de passe (50% à cause des collisions)

— Espace : $O(m)$ stockage

— Temps recherche : $O(t^2)$ opérations

Problème : Collisions de chaînes (Merging chains) : Si deux chaînes produisent le même hash intermédiaire, elles fusionnent. Perte de 50% de couverture.

3.2.2 Rainbow Tables (Oechslin, 2003)

Idée clé : Utiliser des fonctions de réduction DIFFÉRENTES à chaque étape

```
p --H--> h --R--> p --H--> h --R--> p --H--> ... --R--> p
```

Avantage : Élimine les collisions de chaînes !
Comparaison Hellman vs Rainbow :

Caractéristique	Hellman	Rainbow
Fonctions réduction	1 seule (R)	t différentes ($R_1 \dots R_t$)
Collisions chaînes	Oui (50%)	Non (éliminées)
Couverture ($m \times t$)	$\sim 0.5 \times m \times t$	$\sim 0.86 \times m \times t$
Temps recherche	$O(t^2)$	$O(t^2)$
Efficacité	Moyenne	**2× meilleure**

Exemple Concret :

— Espace : alphanumériques 8 chars = $62^8 \approx 2.18 \times 10^{14}$
— Hash : MD5 (rapide)
— Chaînes : $m = 10^8$ (100 millions)
— Longueur : $t = 10^6$ (1 million)
— Couverture : $\sim 8.6 \times 10^{13}$ (40% de l'espace)
— **Espace disque : 1.6 GB seulement !**

Important

Défense : Le salage rend Rainbow Tables inutiles
Avec sel de 128 bits : $2^{128} \approx 3.4 \times 10^{38}$ sels possibles
Espace requis : $2^{128} \times 1.6$ GB $\approx 10^{38}$ exaoctets (totalement infaisable !)
Conclusion : Le salage ($\geq$ 128 bits) rend Rainbow Tables complètement inefficaces.

3.3 Protocoles Challenge-Response

Objectif : Ne JAMAIS transmettre le mot de passe sur le réseau.
Protocole basique (HMAC) :

1. Client → Server : username
2. Server → Client : challenge (nonce aléatoire)
3. Client → Server : response = HMAC(password_key, challenge)
4. Server vérifie : response $\stackrel{?}{=}$ HMAC(stored_key, challenge)

Avantages :

— Mot de passe jamais transmis
— Protection contre replay attack

3.4 Authentification Multi-Facteurs (MFA)

TOTP (Time-based One-Time Password) :

$$\text{TOTP} = \text{Truncate}(\text{HMAC-SHA1}(K, \lfloor T/30 \rfloor))$$

Standard : RFC 6238 (Google Authenticator, Authy)
FIDO2/WebAuthn : Standard moderne (passwordless), utilise cryptographie à clé publique, résiste au phishing.

4 Sécurité Réseau

4.1 Attaques DoS/DDoS

SYN Flood :

1. Attaquant → Server : SYN (IP source forgée)
2. Server → (void) : SYN-ACK (va nulle part)
3. Server alloue ressources et attend ACK qui ne vient jamais
4. Table de connexions saturée → Denial of Service

Défense : SYN Cookies :

$$\text{seq\_num} = \text{Hash}(\text{src\_ip}, \text{src\_port}, \text{dst\_ip}, \text{dst\_port}, \text{time}, \text{secret})$$

Ne stocke PAS l'état avant connexion complète.
Amplification Attacks :

Protocole	**Requête**	**Réponse**	**Facteur**
DNS (ANY)	60 B	3000 B	**50x**
NTP (monlist)	48 B	468 B	9.7x
Memcached	15 B	750 KB	**51,000x**

4.2 Firewalls

Types :

1. **Packet Filtering** (stateless) : Filtre IP source/dest, ports, protocole
2. **Stateful Firewall** : Suit l'état des connexions TCP
3. **Application Layer / WAF** : Inspecte contenu HTTP/HTTPS, détecte SQLi, XSS

Principe du moindre privilège : Tout bloquer par défaut, autoriser explicitement.

4.3 IDS/IPS (Intrusion Detection/Prevention Systems)

IDS : Détecte et alerte (passif)
IPS : Détecte et bloque (actif)
Types de détection :

1. **Signature-based** : Compare trafic à signatures d'attaques connues (rapide, vulnérable aux zero-days)
2. **Anomaly-based** : Détecte déviations du comportement normal (détecte attaques inconnues, taux élevé de faux positifs)

Métriques IDS :

— **TPR** (True Positive Rate) : Pr[Alerte | Attaque] (sensibilité)
— **FPR** (False Positive Rate) : Pr[Alerte | Normal]
— **PPV** (Positive Predictive Value) : Pr[Attaque | Alerte] = TP / (TP + FP)

4.4 Base-Rate Fallacy et IDS

Le problème : Un IDS avec 99% de précision est-il vraiment efficace ?

4.4.1 Exemple Concret : IDS "99% précis"

Paramètres :

— TPR = 99% (détecte 99% des attaques)

— FPR = 1% (1% du trafic légitime déclenche alerte)

— **Base rate** : Pr[Attaque] = 0.1% (0.1% du trafic est malveillant)

Théorème de Bayes :

$$\Pr[\text{Attaque} \mid \text{Alerte}] = \frac{\Pr[\text{Alerte} \mid \text{Attaque}] \times \Pr[\text{Attaque}]}{\Pr[\text{Alerte}]}$$

où :

$$\begin{aligned}\Pr[\text{Alerte}] &= \Pr[\text{Alerte} \mid \text{Attaque}] \times \Pr[\text{Attaque}] + \Pr[\text{Alerte} \mid \text{Normal}] \times \Pr[\text{Normal}] \\ &= 0.99 \times 0.001 + 0.01 \times 0.999 \\ &= 0.00099 + 0.00999 = 0.01098\end{aligned}$$

Donc :

$$\Pr[\text{Attaque} \mid \text{Alerte}] = \frac{0.99 \times 0.001}{0.01098} = \frac{0.00099}{0.01098} \approx 0.09 = 9\%$$

Avertissement

Résultat choquant : Seulement **9% des alertes sont vraies** !
→ **91% sont des faux positifs** !

4.4.2 Table de Confusion (100,000 connexions)

	Attaque réelle	**Trafic normal**	**Total**
Alerte	99 (TP)	999 (FP)	1,098
Pas alerte	1 (FN)	98,901 (TN)	98,902
Total	100	99,900	100,000

Calcul PPV :

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{99}{99 + 999} = \frac{99}{1{,}098} \approx 9\%$$

4.4.3 Le Phénomène de Base-Rate Fallacy

Intuition erronée : "Mon IDS a 99% de précision, donc si une alerte se déclenche, il y a 99% de chances que ce soit une attaque."

FAUX ! Cette intuition ignore :

— Le taux de base des attaques est très faible (0.1%)

— Il y a BEAUCOUP PLUS de trafic légitime que malveillant

— Même 1% de FPR génère énormément de fausses alertes

Calcul mental rapide :

— 100,000 connexions : 100 attaques, 99,900 normales

— FP générés : $99{,}900 \times 1\% = 999$ fausses alertes

— TP générés : $100 \times 99\% = 99$ vraies alertes

— Ratio FP :TP = 999 :99 $\approx$ 10 :1

4.4.4 Solutions pour Améliorer le PPV

1. Réduire le FPR (impact énorme !)

FPR	Alertes totales	PPV
1%	1,098	**9%**
0.5%	599	16%
0.1%	199	**50%**
0.01%	109	**91%**

→ Diviser FPR par 10 améliore PPV de 9% à 50% !

2. Corrélation d'événements :

— Combiner 2 IDS indépendants (FPR = 1% chacun)

— FPR combiné : $0.01 \times 0.01 = 0.01\%$

— PPV passe à $\sim 91\%$!

3. Machine Learning et Tuning : Feature engineering, ensemble methods, anomaly scoring

4. Context-Aware Detection : Historique utilisateur/IP, géolocalisation, reputation scoring

Important

Conclusion : Le taux de base (base rate) est CRUCIAL pour interpréter les alertes. Toujours calculer PPV, pas seulement TPR. Réduire FPR est plus important qu'augmenter TPR dans de nombreux cas.

5 Sûreté Mémoire (Buffer Overflow)

5.1 Stack Frame x86 32-bit

Layout mémoire :

```
Adresses hautes
+-----------------+ <- EBP + 8
| Arguments       |
+-----------------+ <- EBP + 4
| Return Address  | <- CIBLE de l'attaque !
+-----------------+ <- EBP
| Saved EBP       |
+-----------------+
| Variables loc.  |
+-----------------+
| buffer[N]       | <- Débordement
+-----------------+ <- ESP
Adresses basses
```

5.2 Exploitation Buffer Overflow

Mécanisme :

1. Déborder le buffer pour écraser return address
2. Rediriger vers shellcode injecté
3. Shellcode s'exécute avec privilèges du programme

Payload typique :

```
[NOP sled][Shellcode][Padding][Return Address]
```

Shellcode Linux x86 (execve("/bin/sh"), 23 bytes) :

```
\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e
\x89\xe3\x50\x53\x89\xe1\xb0\x0b\xcd\x80
```

5.3 Défenses Modernes

1. **Stack Canaries** : Valeur aléatoire placée avant return address, vérifiée avant `ret`
2. **NX bit (DEP)** : Stack non-exécutable (`gcc -z noexecstack`)
3. **ASLR** : Randomisation adresses mémoire (stack, heap, libraries)
4. **ROP (Return-Oriented Programming)** : Attaque contournant NX en chaînant gadgets existants

Exemple

Compilation sécurisée :

```
gcc -fstack-protector-all -D_FORTIFY_SOURCE=2 \
    -pie -fPIE -Wl,-z,relro,-z,now,-z,noexecstack \
    program.c
```

6 Formules et Concepts Clés

6.1 Cryptographie

Perfect Security :

$$\Pr[M = m \mid C = c] = \Pr[M = m]$$

PRG Advantage :

$$\text{PRGadv}[A, G] = |\Pr[A(G(s)) = 1] - \Pr[A(r) = 1]|$$

HMAC :

$$\text{HMAC}_k(m) = H\left((k \oplus \text{opad}) \| H((k \oplus \text{ipad}) \| m)\right)$$

6.2 Authentification

TOTP :

$$\text{TOTP} = \text{Truncate}(\text{HMAC-SHA1}(K, \lfloor T/30 \rfloor))$$

6.3 Sécurité Réseau

Bayes Theorem :

$$\Pr[A \mid B] = \frac{\Pr[B \mid A] \times \Pr[A]}{\Pr[B]}$$

où :

$$\Pr[B] = \sum_i \Pr[B \mid A_i] \times \Pr[A_i]$$

PPV (Positive Predictive Value) :

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

7 Checklist Examens

7.1 Examens Janvier 2024

Q1 : Vrai/Faux (10 points)

— Perfect Security, OTP, théorème de Shannon
— PRG sécurisé, indistinguabilité
— AES-CBC et IV aléatoire
— HMAC et collisions
— bcrypt vs SHA-256 pour mots de passe
— Firewall stateful et attaques applicatives
— Buffer overflow et exécution code arbitraire
— ASLR et impossibilité buffer overflow
— E2EE et protection MITM

Q2 : Rainbow Tables (5 points)

— Principe Hellman vs Rainbow
— Avantages Rainbow Tables
— Impact du salage

Q3 : Buffer Overflow avec Protection (5 points)

— Dessiner stack frame
— Calculer offsets
— Contourner canary maison
— Améliorations proposées

7.2 Examens Janvier 2025

Q1 : PRG Security (7 points)
— Définition formelle sécurité PRG
— Analyse $G(s) = s \| \text{AES}_s(0^{128})$
— Proposer PRG insécurisé et prouver
Q2 : IDS base-rate fallacy (6 points)
— Calcul avec théorème de Bayes
— TPR = 99%, FPR = 1%, base rate = 0.1%
— Expliquer phénomène base-rate fallacy
Q3 : Stack Frame (7 points)
— Dessiner stack frame complet
— Calculer offset exact
— Proposer payload exploitation
— Lister vulnérabilités

8 Conseils de Révision

Important

Points clés à maîtriser :

1. **Formules** : Mémoriser Perfect Security, PRG, HMAC, TOTP, Bayes
2. **Tableaux** : KDF comparison, AES modes, amplification factors, FPR vs PPV
3. **Stack frames** : Savoir calculer offsets (EBP ± X)
4. **Base-rate fallacy** : Toujours calculer PPV avec Bayes, comprendre impact FPR
5. **Rainbow tables** : Comprendre Hellman vs Rainbow, impact du salage (2^{128} sels)
6. **Défenses** : Connaître limitations de chaque mécanisme (ASLR, NX, canaries)
7. **Cryptographie** : Différence perfect vs computational security
8. **Authentification** : Pourquoi bcrypt > SHA-256, rôle du sel

Méthode de travail :
— Refaire les calculs de Bayes (IDS question)
— Dessiner stack frames pour différents scénarios
— Expliquer oralement les concepts (rainbow tables, base-rate fallacy)
— Comparer les approches (Hellman vs Rainbow, ECB vs CBC vs CTR)

"Security is a process, not a product."
— Bruce Schneier