

RSX112 - Sécurité des Réseaux



Structure de l'Examen



KERCKHOFFS PRINCIPLE

Algorithm
(Dissere my key!)

A

Principe de Kerckhoff

"La sécurité d'un système cryptographique ne doit pas dépendre du secret de l'algorithme, mais uniquement du secret de la clé"

Pourquoi c'est important ?



Transparence

Audit possible



Confiance

Validation par la communauté



Robustesse

Résiste à l'analyse

Exemples d'Application du Principe de Kerckhoffs

Respectent le principe

- AES (standard public)
- RSA (algorithme connu)
- SHA-256 (spécification ouverte)

Ne respectent pas

- Algorithmes propriétaires
- Obscurité par sécurité
- Chiffrement "maison"

Exercice Rapide

Question : Une entreprise utilise un algorithme de chiffrement secret développé en interne. Quels sont les risques ?

Réponse attendue :

- Violation du principe de Kerckhoffs
- Pas d'audit externe possible
- Failles potentielles non détectées
- Fausse impression de sécurité

Fonctions de Hachage

Propriétés Essentielles

Résistance à la collision

- Difficile de trouver $x \neq y$ tel que $H(x) = H(y)$
- Exemple : Birthday attack

Résistance à la préimage

- Impossible de retrouver x depuis $H(x)$
- Unidirectionnel

Pourquoi MD5/SHA-1 sont obsolètes ?

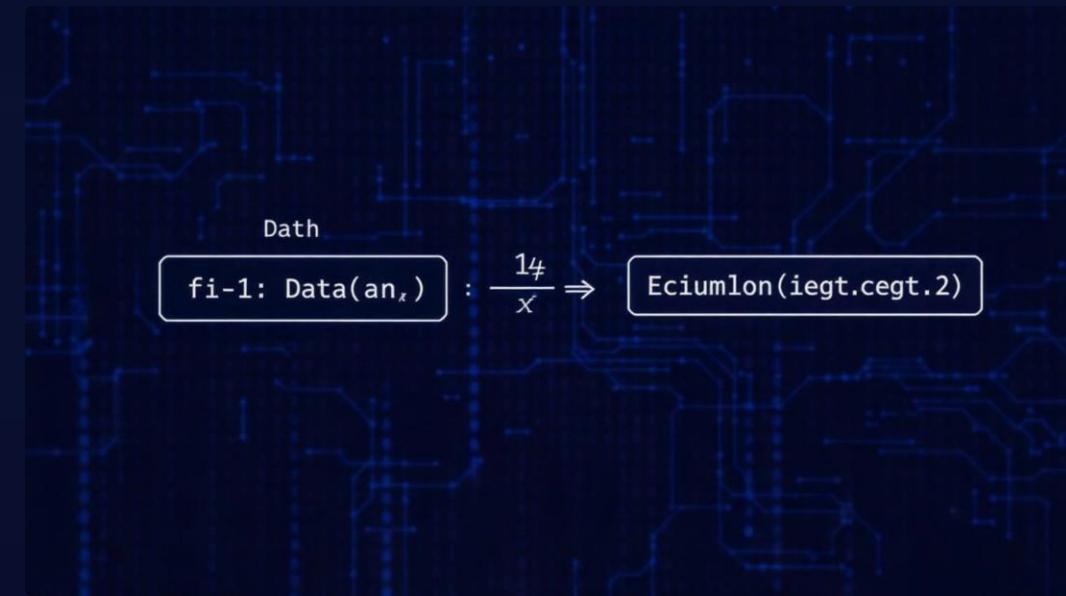
- MD5 (1991) → Collisions démontrées (2004)
- SHA-1 (1995) → Collisions pratiques (2017)
- SHA-256 (2001) → Toujours sûr ✓
- SHA-3 (2015) → Alternative moderne ✓

Algorithmes de Hachage Recommandés



SHA-256/512

Standard actuel largement utilisé dans l'industrie pour la sécurisation des données et des communications.



SHA-3

Basé sur l'algorithme Keccak, offrant une alternative robuste aux fonctions de la famille SHA-2.



BLAKE3

Algorithme moderne offrant des performances élevées tout en maintenant un excellent niveau de sécurité.

Architecture PKI

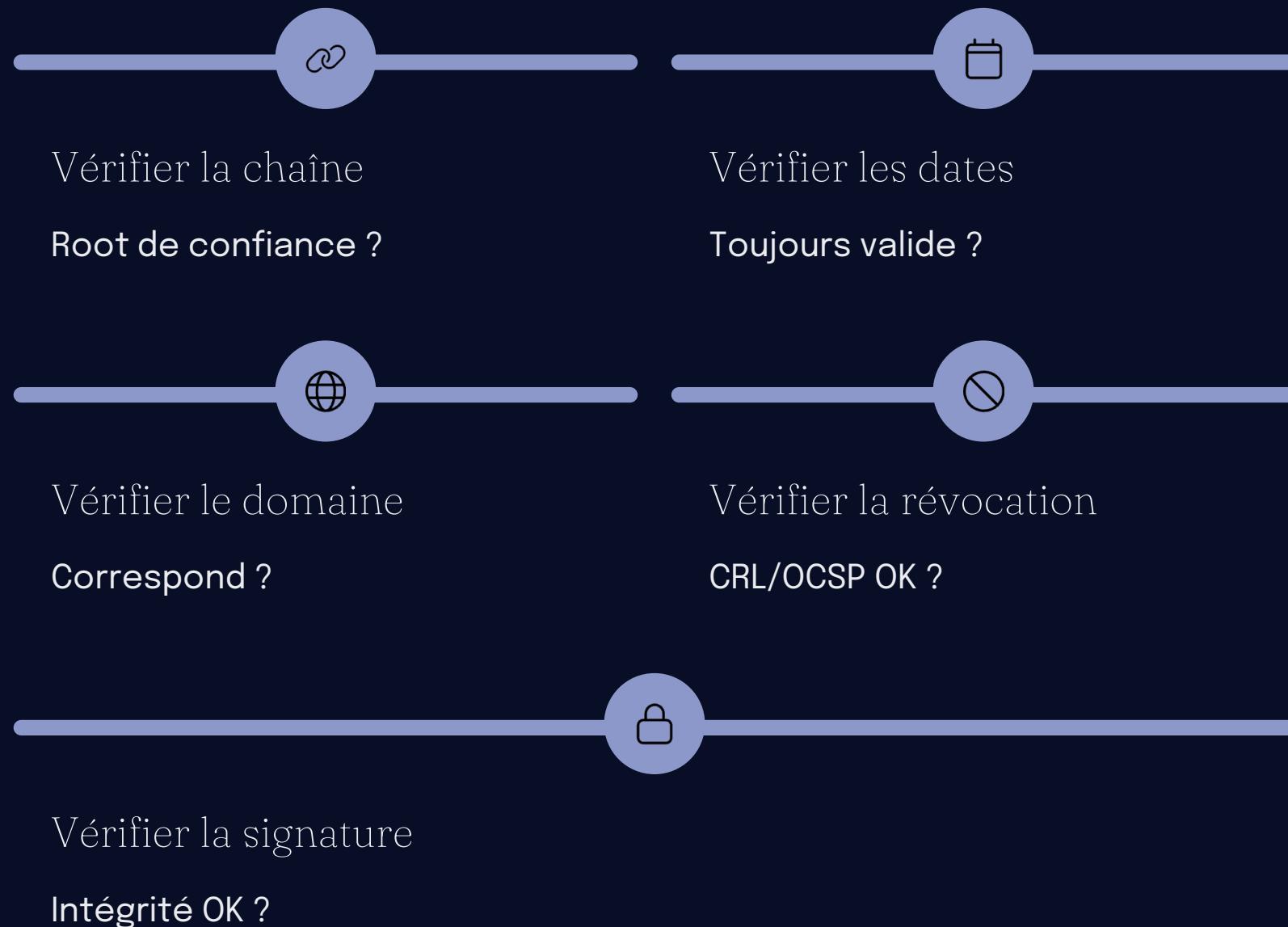
Hiérarchie des CA

```
[Root CA] ← Offline, HSM, 20 ans | ┌─ [CA TLS] ← Online, 10 ans | ┌─ Certificats serveurs | ┌─ [CA Email] ← Online, 10  
ans | ┌─ Certificats S/MIME | ┌─ [CA Code] ← Online, 10 ans | ┌─ Signature de code
```

Pourquoi séparer Root et Intermédiaires ?

1. Sécurité : Root CA hors ligne
2. Flexibilité : Révocation granulaire
3. Organisation : Spécialisation par usage
4. Risque : Impact limité si compromission

Validation d'un Certificat



Anatomie d'un Certificat X.509

Structure Clé

```
Certificate: Version: 3          # X.509 v3
Serial Number: unique_id # Identifiant unique Signature
Algorithm: sha256RSA # Algo de signature Issuer: CN=CA
Name      # Qui a signé Validity:      #
Période de validité Not Before: date Not After:
date Subject: CN=domain.com # Pour qui Public Key
Info:      # Clé publique Extensions:      #
Capacités - Basic Constraints - Key Usage - SAN
```

Extensions Critiques

Basic Constraints

CA:FALSE → Certificat final (ne peut pas signer)

CA:TRUE → Certificat CA (peut signer)

Key Usage

digitalSignature → Signatures numériques

keyEncipherment → Chiffrement de clés

keyCertSign → Signer des certificats

Subject Alternative Name (SAN)

DNS:example.com

DNS:*.example.com

IP:192.168.1.1

Commandes OpenSSL Essentielles

Lecture de Certificat

```
# Afficher tout le certificat
openssl x509 -in cert.pem -text -noout# Extraire la clé publique
openssl x509 -in cert.pem -pubkey -noout > pubkey.pem# Vérifier la validité temporelle
openssl x509 -in cert.pem -checkend 0# Retourne 0 si valide, 1 si expiré# Afficher le SAN
openssl x509 -in cert.pem -text -noout | grep -A1 "Subject Alternative Name"
```

Génération et Signature

```
# Générer une clé privée
openssl genrsa -out private.key 2048#
Créer une CSR
openssl req -new -key private.key -out request csr
\subj "/C=FR/O=CNAME/CN=example.com"#
Auto-signer (pour tests)
openssl x509 -req -days 365 -in request csr -signkey private.key -out cert.pem
```



Procédure de Renouvellement de Certificat

Côté Client

1. Générer nouvelle paire de clés (recommandé)
2. Créer CSR avec mêmes informations
3. Soumettre à la CA
4. Installer nouveau certificat
5. Tester avant expiration ancien

Bonnes Pratiques

- ⌚ Renouveler 30 jours avant expiration
- 🔑 Nouvelle clé à chaque renouvellement
- 💻 Automatiser le processus
- 🔍 Vérifier après installation

Côté CA

1. Vérifier identité du demandeur
2. Valider informations CSR
3. Vérifier autorisation
4. Signer le certificat
5. Publier dans repository



Exercice Pratique

Analysez ce certificat :

Basic Constraints: CA:FALSE

Key Usage: Digital Signature, Key Encipherment

Extended Key Usage: TLS Web Server Authentication

SAN: DNS:api.company.com, DNS:*.api.company.com

Architecture PKI pour SecureTech

Analyse des Besoins

- 200 employés
- 5 serveurs web internes
- Email sécurisé (S/MIME)
- VPN télétravail
- Signature de code

Architecture Proposée

```
[Root CA SecureTech] Offline | HSM | 20 ans | [Issuing CA] Online | 10 ans
|-----+-----+-----+|      |      |      | [TLS CA]
[User CA] [Code CA] [Device CA] 5 ans   5 ans   5 ans   5 ans
```



Durée de Vie des Certificats et Sécurisation

Type	Durée	Justification
Root CA	20 ans	Offline, changement coûteux
Issuing CA	10 ans	Balance sécurité/opéra tions
Sub-CA	5 ans	Flexibilité technologique
Serveur TLS	1 an	Standard industrie
Utilisateur	2 ans	Turnover employés
Code Signing	3 ans	Stabilité applications

Sécurisation CA Racine

Physique:

- Salle sécurisée, biométrie
- HSM FIPS 140-2 Level 3
- Pas de réseau (air gap)
- Coffre-fort pour backups

Procédures:

- Cérémonie de clés (3 personnes)
- Dual control
- Audit vidéo
- Activation 2 fois/an max

Conseils pour l'Examen

Structure des Réponses

- **Définition** → Court et précis
- **Explication** → Développement technique
- **Exemple** → Cas concret
- **Conclusion** → Synthèse/ouverture

Erreurs Fréquentes

- ✗ Confondre chiffrement et signature
- ✗ Oublier la validation de certificat
- ✗ Mélanger CRL et OCSP
- ✗ Négliger les aspects organisationnels

Pour le Projet PKI

- Justifier TOUS les choix
- Penser sécurité ET opérationnel
- Inclure les coûts
- Prévoir l'évolution

🎓 Bonne Chance pour l'Examen !

Remember : La PKI c'est 20% de technique et 80% de processus !