CRYPTO

Protocoles

Loïc Rouquette

2025-03-24

Correction du QCM

Question 1

Quel algorithme de chiffrement asymétrique utilise le principe des logarithmes discrets pour garantir la sécurité ?

- Diffie-Hellman
- AES
- RSA
- ECC
- SHA-3

Question 1

Quel algorithme de chiffrement asymétrique utilise le principe des logarithmes discrets pour garantir la sécurité ?

- **Diffie-Hellman**
- AES
- RSA -> Utilise mais repose principalement sur le problème de factorisation en nombres entiers
- ECC
- SHA-3

Question 2

Quel est le principal avantage du chiffrement asymétrique par rapport au chiffrement symétrique ?

- Il est plus rapide
- Il permet d'échanger des clés en toute sécurité sans canal confidentiel
- Il permet d'échanger des clés en toute sécurité sans canal authentifié
- Il n'a pas besoin de clés

Question 2

Quel est le principal avantage du chiffrement asymétrique par rapport au chiffrement symétrique ?

- Il est plus rapide
- **Il permet d'échanger des clés en toute sécurité sans canal confidentiel**
- Il permet d'échanger des clés en toute sécurité sans canal authentifié
- Il n'a pas besoin de clés

Question 3

Nous considérons ici l'algorithme d'échange de clé Diffie-Hellman. Cet échange se déroule en 3 étapes.

1. Un nombre premier est choisis, ainsi qu'un générateur g sur $\mathbb{Z}/p\mathbb{Z}$.
2. Alice choisit une valeur x et calcule $V = g^x \mod p$.
3. Bob choisit une valeur y et calcule $W = g^y \mod p$.
4. Après avoir envoyer leurs informations respectives, Alice et Bob connaissent un secret commun.

Dans Diffie-Hellman, que représente l'exposant x ?

- La clé secrète d'Alice
- La clé secrète de Bob
- La partie publique de la clé d'Alice
- La partie publique de la clé de Bob
- Le module de chiffrement
- Le secret commun

Question 3

Nous considérons ici l'algorithme d'échange de clé Diffie-Hellman. Cet échange se déroule en 3 étapes.

1. Un nombre premier est choisis, ainsi qu'un générateur g sur $\mathbb{Z}/p\mathbb{Z}$.
2. Alice choisit une valeur x et calcule $V = g^x \mod p$.
3. Bob choisit une valeur y et calcule $W = g^y \mod p$.
4. Après avoir envoyer leurs informations respectives, Alice et Bob connaissent un secret commun.

Dans Diffie-Hellman, que représente l'exposant x ?

- **La clé secrète d'Alice**
- La clé secrète de Bob
- La partie publique de la clé d'Alice
- La partie publique de la clé de Bob
- Le module de chiffrement
- Le secret commun

Question 4

Quel est le rôle de la fonction de hachage dans la signature numérique ?

- Chiffrer le message
- Réduire la taille de la clé publique
- Garantir l'intégrité du message

Question 4

Quel est le rôle de la fonction de hachage dans la signature numérique ?

- Chiffrer le message
- **Réduire la taille de la clé publique**
- **Garantir l'intégrité du message**

Question 5

Si Alice utilise la clé publique de Bob pour lui envoyer un message, qui peut déchiffrer ce message ?

- N'importe qui
- Alice
- Bob

Question 5

Si Alice utilise la clé publique de Bob pour lui envoyer un message, qui peut déchiffrer ce message ?

- N'importe qui
- Alice
- **Bob**

> Alice connaît le message, mais elle n'est pas capable de le déchiffré.

Question 6

Quel algorithme de chiffrement asymétrique repose sur le problème de factorisation des grands nombres premiers ?

- Diffie-Hellman
- AES
- RSA
- ECC
- SHA-3

Question 6

Quel algorithme de chiffrement asymétrique repose sur le problème de factorisation des grands nombres premiers ?

- Diffie-Hellman
- AES
- **RSA**
- ECC
- SHA-3

Question 7

Si Alice chiffre un message avec sa propre clé privée, que peut-on en déduire ?

- Le message est confidentiel
- Le message est authentifié
- Le message est anonymisé

Question 7

Si Alice chiffre un message avec sa propre clé privée, que peut-on en déduire ?

- Le message est confidentiel
- **Le message est authentifié**
- Le message est anonymisé

Question 8

Un algorithme de chiffrement asymétrique peut-il garantir la confidentialité et l'intégrité d'un message sans autre mécanisme complémentaire ?

- Oui
- Non
- Cela dépend de la longueur de la clé

Question 8

Un algorithme de chiffrement asymétrique peut-il garantir la confidentialité et l'intégrité d'un message sans autre mécanisme complémentaire ?

- Oui
- **Non**
- Cela dépend de la longueur de la clé

Question 9

Dans un système de chiffrement asymétrique, que se passe-t-il si la clé privée est compromise ?

- La sécurité du système est entièrement compromise, et les anciens messages peuvent être déchiffrés.
- Rien, car seule la clé publique est utilisée pour le déchiffrement.
- La clé publique devient inutilisable.

Question 9

Dans un système de chiffrement asymétrique, que se passe-t-il si la clé privée est compromise ?

- **La sécurité du système est entièrement compromise, et les anciens messages peuvent être déchiffrés.**
- Rien, car seule la clé publique est utilisée pour le déchiffrement.
- **La clé publique devient inutilisable.**

Question 10

Si un attaquant intercepte un message chiffré avec RSA, que peut-il faire sans la clé privée ?

- Déchiffrer immédiatement le message grâce à la clé publique.
- Briser le chiffrement en essayant toutes les clés publiques possibles.
- Essayer de factoriser le module n pour retrouver la clé privée.

Question 10

Si un attaquant intercepte un message chiffré avec RSA, que peut-il faire sans la clé privée ?

- Déchiffrer immédiatement le message grâce à la clé publique.
- Briser le chiffrement en essayant toutes les clés publiques possibles.
- **Essayer de factoriser le module n pour retrouver la clé privée.**

Rappels des Dernier Cours

Les 5 propriétés de la sécurité

- Disponibilité
- Authentification
- Intégrité
- Non-Répudiation
- Confidentialité

Les 5 propriétés de la sécurité(ii)

- ~~Disponibilité~~ -> ne concerne pas la cryptographie
- Authentification
- ~~Intégrité~~ -> les fonctions de hashage
- ~~Non-Répudiation~~ -> chiffrements asymétriques
- ~~Confidentialité~~ -> les chiffrements symétriques + asymétriques

Les 5 propriétés de la sécurité(iii)

- ~~Disponibilité~~ -> ne concerne pas la cryptographie
- **Authentification**
- ~~Intégrité~~ -> les fonctions de hashage
- ~~Non-Répudiation~~ -> chiffrements asymétriques
- ~~Confidentialité~~ -> les chiffrements symétriques + asymétriques

Les protocoles cryptographiques

Le protocole SSL/TLS

- Protocole de sécurisation des échanges sur Internet
- Mode client-serveur, au dessus de TCP
- Permet de garantir :
  - L'authentitifcation du serveur ;
  - La confidentialité des données échangées ;
  - L'intégrité et l'authentification de l'origine des données échangées ;
  - L'authentification du client (optionnel)
- Permet d'encapsuler de manière transparent des protocoles de la couche application
  - HTTP (80) -> HTTP**S** (443)
  - IMAP (143) -> IMAP**S** (993)
  - POP3 (110) -> POP3**S** (995)
  - **STARTTLS** (pour IMAP, POP3, SMTP, FTP, etc.) sur **le même port**

Un peu d'histoire

- Au début, **SSL** (Secure Sockets Layer), développé par Netscape
  - **SSL 1.0** (1994) : protocole théorique, jamais utilisé
  - **SSL 2.0** (1995 - 2011) : première version utilisée
  - **SSL 3.0** (1996 - en cours; RFC 6101) : dernière version sur laquelle TLS sera basé
- Puis **TLS** (Transport Layer Security), développé par l'**IETF** (Internet Engineering Task Force)
  - **TLS 1.0** (1990 - en cours ; RFC 2246) : sucesseur de SSL, diverses améliorations jusqu'en 2002
  - **TLS 1.1** (2006 - en cours ; RFC 4346)
  - **TLS 1.2** (2008 - en cours ; RFC 4346)
  - **TLS 1.3** (2018 - en cours ; RFC 4346)

SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
0.2%	1.6%	29.1 %	31.4%	99.9%	66.9%

Compatibilité des serveurs HTTPS (source SSL Pusle, 12/2024)

Mécanismes cryptographiques dans TLS

- TLS offre le choix entre **plusieurs mécanismes cryptographiques** pour être capable de s'adapter aux **contraintes** de chaque application et de chaque système.
- Authentification du serveur (et du client, optionnellement) :
  - clé publique : RSA, DSS, ECDSA
  - clé secrète ou mot de passe partagé : **PSK** (Pre-Shared Key), **SRP** (Secure Remote Password)
  - pas d'authentification : **ANON**
- Échange de clés :
  - clé publique (toujours la même clé privée côté serveur) : RSA
  - Diffie-Hellman statique (même problème) : DH, ECDH
  - clé secrète ou mot de passe partagé : PSK, SRP
  - Diffie-Hellman éphémère (clés privées **propres à chaque connexion ; garantit la** forward secrecy**) : DHE, ECDHE

Mécanismes cryptographiques dans TLS(ii)

- Chiffrement :
  - chiffrement par bloc : AES-CBC, 3DES-CBC, DES-CBC, etc.
  - chiffrement par bloc avec mode authentifiant : AES-CCM, AES-GCM, etc.
  - chiffrement par floc : RC4
  - pas de chiffrement : NULL
- Intégrité et authentification de l'origine des messages :
  - HMAC : HMAC-MD5, HMAC-SHA1, HMAC-SHA256, etc.
  - mode authentifiant du chiffrement par block : AEAD
- Une combinaison de ces mécanismes est appelée **cipher suite**
  - par exemple : TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- En bonus, TLS peut aussi supporter un **mode de compression** des données : NULL ou DEFLATE

Établissement d'une connexion SSL/TLS

- Contexte : un **client** souhaite établir une connexion SSL/TLS avec un **serveur**
- Objectifs :
  - se mettre d'accord sur une **cipher suite** commune
  - **authentifier** le serveur
  - échanger des **clés secrètes** pour le chiffrement et l'authentification des messages

Protocole de **handshake** en **4** étapes

Handshake SSL/TLS simple

1. Client -> Seveur

- `ClientHello`: plus haute version du protocole supportée, liste des **cipher suites** et des modes de compression supportés ;

Handshake SSL/TLS simple(ii)

2. Serveur -> Client

- `ServerHello` : version du protocole, **cipher suite** et mode de compression choisi ;
- `Certificate` (opt.) : la clé publique du serveur dans un certificat X.509 permettant d'en vérifier l'authenticité ;
- `ServerKeyExchange` (opt.) : une clé publique Diffie-Hellman pour l'échange de clés ;
- `ServerHelloDone`

Handshake SSL/TLS simple(iii)

3. Client -> Server

- `ClientKeyExchange` : soit un secret (*PreMasterKey*) chiffré avec la clé publique du serveur, soit une clé publique Diffie-Hellman pour l'échange de clés ;
- `ChangeCipherSpec` (marque la fin du *handshake* côté client)
- `Finished` : message chiffré et authentifié contenant un MAC des messages précédents du *handshake*

Handshake SSL/TLS simple(iv)

4. Serveur -> Client (si le `Finished` du client est valide)

- `ChangeCipherSpec` (marque la fin du *handshake* côté serveur)
- `Finished` : message chiffré et autentifié contenant un MAC des messages précédents du *handshake*

Handshake SSL/TLS simple(v)

Si le `Finished` du serveur est aussi valide, la connexion est établie

Authentification du serveur

- Dans le *handshake* prédédent, le serveur envoie **lui-même** sa **clé publique** afin d'**authentifier ses messages**

N'y a-t-il pas comme un problème ?

Authentification du serveur(ii)

Comment **vérifier l'authenticité de la clé publique** du serveur ?

Authentification du serveur(iii)

- Clé publique **signée par le serveur** ?
  - La vérification nécessite de faire confiance à la clé publique…
- Clé publique **signée par une tiece partie** (appelée **autorité de certification** (CA)) ?
  - OK, mais comment vérifier cette signature ?
- Clé publique **signée par un** CA**, et clé publique du** CA** ?
  - Comment vérifier l'**authenticité de la clé publique du CA**…

Authentification du serveur(iv)

Il s'agit d'un problème difficile qui nécessite la mise ne place d'une **Infrastructure à clés publiques** (PKI)

Infrastructures à clés publiques

Les Infrastructures à clés publiques

Permet de répondre à la question :

Comment faire confiance à une clé publique ?

- Certificat : **signature de la clé publique** par un tiers de confiance

Les Infrastructures à clés publiques(ii)

Infrastructures possibles :

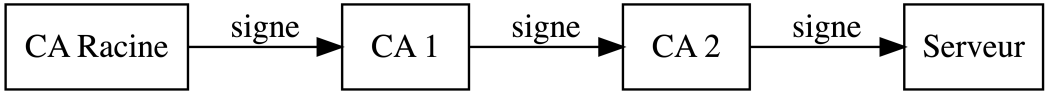
- Hiérarchique : **autorités de certification** (SSL/TLS et X.509, EMV)
- Décentralisée : **réseau de confiance** (PGP, GnuPG)

Les Infrastructures à clés publiques(iii)

Différents niveaux :

- **CA Racines** : peuvent certificer les clés **publiques d'autres CA**
- **CA intermédaires** : en général, **ne peuvent pas** certifier d'autres **CA**
  - certifient les **clés publiques des serveurs**
- Chaîne de certification :

Les Infrastructures à clés publiques(iv)



Les Infrastructures à clés publiques(v)

- **Authentification** : le serveur envoie trois certificats :
  - sa propre **clé publique**, **signée** par CA 2;
  - la **clé publique** de CA 2, **signée** par CA 1;
  - la **clé publique** de CA 1, **signée** par CA racine.
- **Vérification** : si le client connaît (et fait confiance à) la **clé publique de CA racine**, il peut vérifier la validité de **tous les certificats**.

Les Infrastructures à clés publiques(vi)

Toujours les mêmes problèmes : - Comment obtenir la **clé publique des CA racines** ? - **Quelle confiance** leur accorder ?

- Liste de **CA racines de confiance** maintenue par les **navigateurs**
  - Actuellement, **80+ CA racines** intégrés dans Firefox
  - mais comment **avoir confiance en le navigateur** que l'on télécharge ?!
    - **contrôle d'intégrité et d'authenticité** de l'archive téléchargée ?
    - **problème de la poule et de l'œuf...**
- **Attention à la sécurité des CA** (racines et intermédaires) !
  - si la **clé privée** d'un CA est compromise : **émission de faux certificats**
    - *e.g* **DigiNotar** en 2011 : 500 faux certificats, dont `*.google.com`
  - **audits de sécurité** réguliers
  - **mécanismes de révocation** de certificats compromis : CRL (*Certificate Revocation List*) ou OCSP (*Online Certificate Status Protocol*)

Failles dans SSL/TLS

Failles dans SSL/TLS

- Des vulnérabilités sont **tojours** découvertes dans SSL/TLS
  - *e.g* : **DROWN**, rendue publique le 1er mars 2016
- Des correctifs existent, mais les serveurs **doivent être mis à jour**
  - **renégotation non sécurisée** (2009, MITM) : 1.8 + 0.6% vulnérables
  - **BEAST** (_Browser Exploit Against SSL/TLS, 2011, violation de la contrainte d'origine des cookies) : 91.5%
  - **CRIME** (*Compression Ratio Info-Leak Made Easy, 2012*) : 3.2%
  - **dégradation du protocole** (force d'utilisation de SSL 3.0) : 28%
  - **attaques du RC4** (2013) : 8.5 + 34.8%
  - **POODLE sur TLS** (2014) : 3.0%
  - **Heartbleed** (2014, accès mémoire arbitraire dans OpenSSL) : 0.3%
  - **pas de forward secrecy** : 21.2 + 29.4%
  - **clés trop petites** : 0.1% (clé publique), 6.9 + 25.2 (éch. de clés)

Les standards

Recommandations du NIST

Mécanismes	Types	Primitives	Mécanismes complémentaires	Exemples de standards
Sans clé	Fonction de hashage	fonction de compression	Générateur de sel (aléa)	FIPS 180-4 SHA-384
	Générateur d'aléa	fonction de hachage chiffrement par flot, chiffrement par bloc MAC	Générateur physique d'aléa source d'entropie	NIST SP 800-90 `dev/random`

Recommandations du NIST(ii)

Mécanismes	Types	Primitives	Mécanismes complémentaires	Exemples de standards
À clé secrète	MAC	fonction de hachage, chiffrement par flot chiffrement par bloc fonction de hachage universel	Générateur d'aléa (clé)	NIST SP 800-38B AES-OMAC
	Chiffrement	fonction de rétroaction / filtrage / initialisation permutation aléatoire paramétrée (chiffrement par bloc)	Générateur d'aléa (clé, IV)	FIPS 197, NIST SP 800-38A AES-CTR

Recommandations du NIST(iii)

Mécanismes	Types	Primitives	Mécanismes complémentaires	Exemples de standards
À clé publique	Signature	RSA, ElGamal, ECDSA, EdDSA, NTRU, McEliece	Générateur d'aléa (clé, sel) fonction de hachage	FIPS 186-5 RSA-PPS
	Chiffrement	RSA, ElGamal, NTRU, McEliece	Générateur d'aléa (clé, sel) fonction de hachage	PKCS #1 v2.1 RSA-OAEP

Cryptographie Quantique

Algorithme de Shor

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer (Shor 1999)

Complexité : $O\left((\log N)^2(\log\log N)(\log\log\log N)\right)$

Appel du NIST

- Call for Proposal (2017) - Round 1 - Identification des forces & faiblesses
  - 82 soumissions
  - 69 candidats valides
  - 26 candidats sélectionnés
- Round 2 (2019) - Évaluation des performances matérielles & logicielles
  - 26 candidats
  - 8 alternatives
  - 7 candidats sélectionnés
- Round 3 (2020) - Identification des forces & faiblesses
  - 7 candidats
  - 4 PKE/KEM
  - 3 signatures
- Round 4 (2022)- Sélection & Standardisation

Appel du NIST(ii)

- 4 candidats PKE/KEM
- **Attaques sur SIKE (finaliste)**

- **FIPS** (2024)

Résultats

Selected algorithms 2022-2024

CRYSTALS-Kyber CRYSTALS-DILITHIUM Falcon SPHINCS+

FIPS

Numéro	Nom	Standard	Algorithme
203	Module-Lattice-Based Key-Encapsulation Mechanism Standard	ML-KEM	CRYSTALS-Kyber
204	Module-Lattice-Based Digital Signature Standard	ML-DSA	CRYSTALS-DILITHIUM
205	Stateless Hash-Based Digital Signature Standard	SLH-DSA	SPHINCS+

Résultats(ii)

ML-DSA et ML-KEM sont déjà en cours d'implémentation dans la bibliothèque `libgcrypt`

Et après ?

Les perspectives de recherche

Internet of Things

- Besoins de chiffrements légers
  - Pour les chiffrements symétriques : débit, puissance de calcul ;
  - pour les chiffrememnts asymétriques : taille de clé (3168 pour ML-KEM, 4896 pour ML-DSA-87)

Chiffrement quantiques

- Cryptanalyse à ses débuts (*e.g* SIKE)