Cryptographie Appliquée Avancée
Midterm

8 Novembre 2023

Nom :

Prénom :

Durée : **90 minutes**
Matériel : Un **laptop offline** est autorisé ainsi que tout matériel papier
L'examen a **9 pages** et **3 exercices** pour un total de **46 points**.
Ne restez pas bloqués sur une question !

Page Vide

1 Questions en vrac (14 points)

1. Pour GCM avec un nonce aléatoire de 96 bits, le NIST recommande d'envoyer au maximum 2^{32} messages avec la même clef. Pourquoi cette valeur et non $2^{96/2} = 2^{48}$? **(2 pts)**

à cause des paradoxes d'anniversaires
2^{-32} c'est pour être sûr de n'avoir pas de collisions
c'est pour avoir une marge de sécurité. 2^{48} -> 50 % de tomber sur une collision..
pas assez précis 2^{32}, -> respecte les 2^{-32}.

2. Pour tirer un entier aléatoire modulo 60 (donc un élément de $\mathbb{Z}_{60}$), un ingénieur décide de tirer 60 bits parfaitement aléatoires et de les sommer. Est-ce un bon moyen de tirer un entier modulo 60 ? Justifiez. **(2 pts)**

C'est n'est pas bon car la distribution n'est pas uniforme. Les valeurs auront tendance à être au début de la distribution.

La distribution est vers le milieu

3. Dans un logiciel, l'aléa (*seed*) utilisée pour chiffrer avec RSA-OAEP est fixée à 0. Vous trouvez le salaire d'un collègue chiffré avec cette solution. Expliquez comment vous pouvez récupérer son salaire. **(3 pts)**

Vu que le seed est fixé ça rend tout le système déterministe
Ce qu'on pourrait faire, c'est qu'on a accès à un oracle de chiffrement et donc on retrouve le salaire.
On impose une grosse intervalle de salaire avec des incréments on continues et regarde la valeur de l'output

à Alice

4. Vous souhaitez envoyer un email chiffré à un collègue. Quel est l'avantage de ECIES par rapport à RSA-OAEP ? **(3 pts)**

ECIES est un schéma de chiffrement hybride avec un mac. Donc, il garanti intégrité et ~~authenticité~~ + chiffrement symétrique donc plus rapide.

+ taille des messages plus long car chiffrement sym.
Il n'y a pas l'authenticité.

5. Un équipement IoT doit effectuer des signatures digitales. Vous savez que cet équipement utilise RC4 pour générer des nombres aléatoires (les valeurs pseudo-aléatoires sont simplement les sorties de l'algorithme de chiffrement par flot). Quel algorithme de signatures proposez-vous ? Justifiez. **(4 pts)**

Vu que RC4 c'est casser, même pour générer des valeurs pseudo aléatoire ce n'est pas bon.

Il faudrait choisir un système de signature déterministique directement comme ED25519

à Alice pour toute une signature déterministique

2 SHAKE et SHA3 (14 points)

Pour rappel, SHAKE est une version spéciale de SHA3.

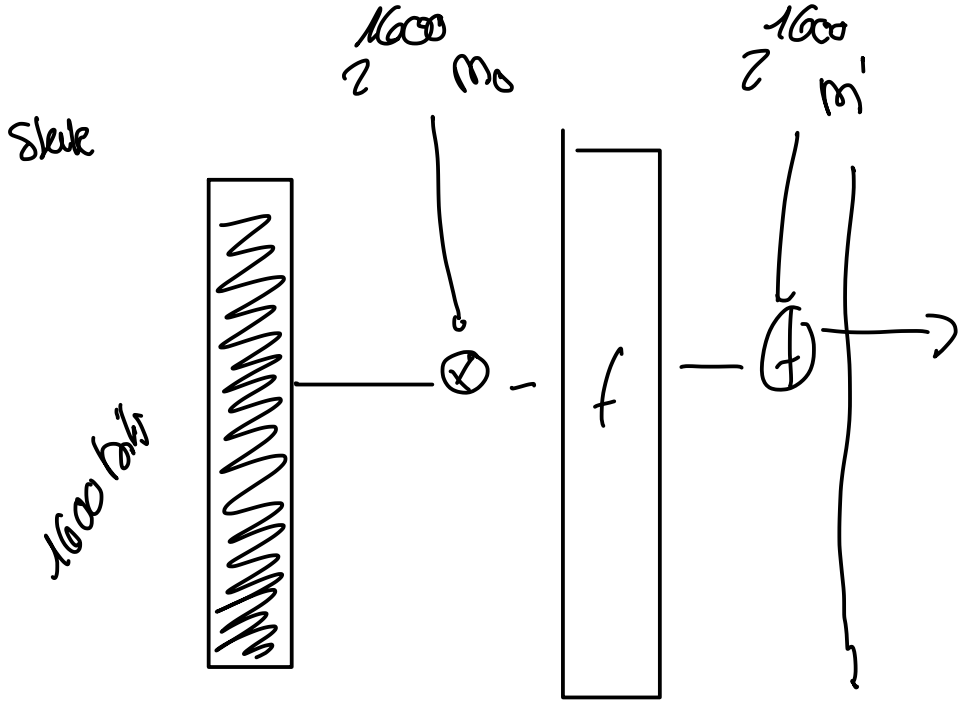
1. Quel peut être un avantage de SHAKE par rapport à SHA3 ? **(2 pts)**

Shake a un output variable.
On peut étendre et tronquer comme on veut pour la sortie.

2. Donnez un exemple concret d'utilisation de SHAKE (où il apporte un avantage par rapport à SHA2/SHA3). **(2 pts)**

Par exemple utiliser Shake pour générer des clefs de taille variable.

3. Pour optimiser SHA3, un "ingénieur" décide de supprimer l'état interne de la construction éponge. Plus précisément, des blocs de 1600 bits de messages sont XOR avec l'état (tout est état externe). Montrez comment vous pouvez trouver une collision dans cette fonction de hashage. **(5 pts)**

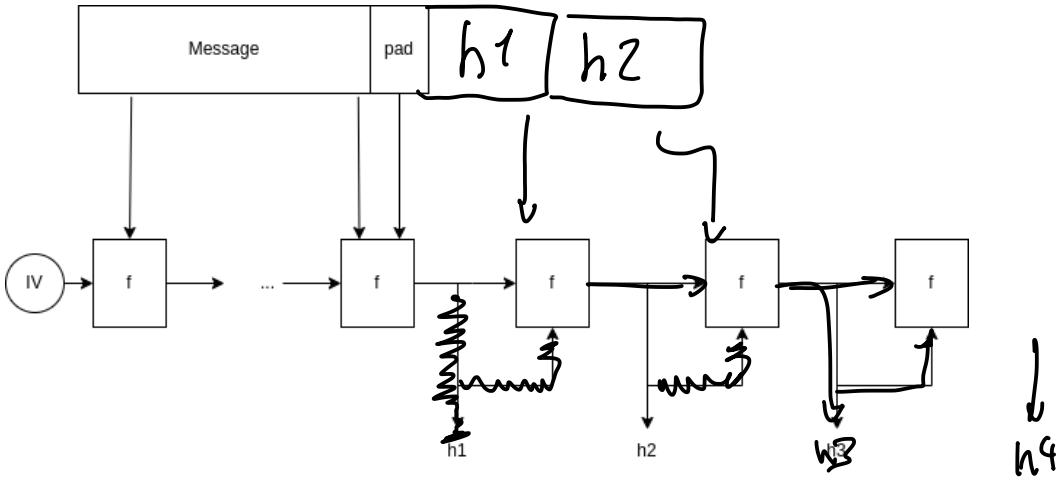


Vu que tout le message est absorbé, on peut trouver un message m' tel que

permutation (S0 ⊕ m0) ⊕ permutation (S1 ⊕ m') = 0

Ce qui veut dire notre m' est la première sortie, on peut contrôler l'input total de la prochaine permutation

4. Un adepte de la blockchain décide de modifier SHA2 afin qu'il puisse gérer une sortie de taille arbitraire. Pour ce faire, il modifie la construction de Merkle-Damgård afin que les sorties soient réinjectées dans la fonction de compression (f dans le schéma) pour générer plus de valeurs (h_1, h_2 et h_3 sur le schéma). Cassez la construction. **(5 pts)**



On peut append les prochain hash dans le message, qui modifie le message et qui garde un key valide

On prend notre message et on fait un nouveau m'
On prend calcule h1||h2 et on concate m'||h1||h2
notre nouveau second h3 et h4, mais le key secret quand même valide.

$(m||h_i, h_{i+n})$

$n \in \mathbb{N}$

3 Scénarios (18 points)

Nous allons analyser **trois scénarios différents** dans lesquels vous allez devoir faire un choix algorithmique. Plus précisément, vous allez devoir analyser la compatibilité des quatre algorithmes de chiffrement symétriques suivants :

- AES-ECB
- AES-CBC
- AES-CTR
- AES-GCM

→ piège des noces sur les params. taille IV, compteur, tag, clef, blocks

Pour **chacun des algorithmes** :

- Indiquez s'il est compatible avec le scénario. Soyez précis sur la manière dont vous comptez l'utiliser (taille des IV/compteurs, taille du tag, ...). Justifiez vos choix.

Recommandez ensuite **un algorithme** qui est le meilleur selon vous. Justifiez.

1. Dans ce premier scénario, nous souhaitons chiffrer un disque dur contenant des données numériques sensibles. Que proposez-vous ? XTS ne fait pas partie de choix proposés. **(6 pts)**

ECB -> problème de patterns avec les blocks qui se ressemble

CBC -> bien, mais comme clés différent de block à la chaine, on pourrait pas simplement déchiffrer un seul block.

CTR -> souci de compteur et de nonce qu'on doit stocker et faire attention car fait pas un integer overflow, donc faut gérer et stocker tout les nonces pour éviter les soucis de patterns.

AES-GCM → on veut pas sauver plus de données additionnelles, overhead & tag

On prend CBC

↳ on peut soit prendre un IV prédictible, en utilisant par exemple numéro de secteur. Ou sinon un IV aléatoire, mais on doit sauver où stocker chaque IV, donc une perte de stockage.

2. Le but ici est de chiffrer des communications entre deux équipements IoT. Ces équipements n'ont pas de mémoire persistante (sauf pour la clef secrète) mais 512Mo de RAM. A cause de leurs batteries, les équipements s'éteignent souvent et doivent être redémarrés. Nous supposons que les adversaires peuvent modifier les messages à leur guise. Finalement, on estime à 100'000'000'000 le nombre de messages échangés entre les deux équipements, chaque message faisant maximum 2^{20} bits. **(6 pts)**

messages $\leq 2^{36}$
n_blocks $\leq 2^{36} \cdot 2^{-20} = 2^{16}$ blocks

ECB pas de key

CBC pas de key

CTR pas de key

AES-GCM -> key

↳ serait les aspects de GCM

On aimerait un key car les cachesaires peuvent modifier.

GCM counter $= 2^{32} - 2$, donc nous avons assez

IV = 96 bits

Count = 32 bits

3. Le but est, cette fois, de chiffrer les communications qu'un capteur de température envoie à un serveur (et uniquement dans cette direction). Ce capteur envoie un message toutes les secondes pendant 2 ans. Ensuite, la clef est changée. Dans ce scénario, les textes clairs font toujours exactement 256 bits. Le message que vous envoyez au serveur a toujours la forme suivante : [date en ms, texte chiffré] et ne peut pas contenir d'autres valeurs. Le texte chiffré dans ce message doit faire au maximum 256 bits et vous ne pouvez envoyer qu'un seul message par seconde. Nous supposons que les adversaires ne font qu'écouter les communications. **(6 pts)**

$\leq 2^{26}$

ECB = déterministe

CBC = besoin de padding -> ajout des données deux fois

CTR = pas déterministe, nonce & counter -> counter $\geq 2^{26}$ + nonce

nonce = date

counter minimum 26 bits

mais la date en nos devrait suffire.

AES-GCM = sans le tag ici, mais ça serait à peice du CTR