Université de Sherbrooke - Département d'informatique
IFT814 – Cryptographie
Martin Fiset, chargé de cours

Devoir 3
Date de remise : Vendredi 24 octobre à 23h59

Effectuez ce devoir en équipe d'au plus 4 personnes. Utilisez TurninWeb, le système de soumission de travaux du Département d'informatique, pour soumettre votre travail. Soumettez un seul fichier pdf par équipe (n'oubliez pas d'y mettre tous les noms), contenant le code source nécessaire comme texte dans le corps de votre devoir. Ce devoir compte pour 12% de la note finale. Les questions bonus ou supplémentaires ne sont pas nécessaire pour le calcul de la note, mais pourraient servir à enrichir votre note finale.

1. Protocole d'échange de clé avec variante de Diffie Hellman

Vous allez implémenter une variante du protocole d'échange de clé de Diffie Hellman avec des paramètres différents et une analyse de sécurité approfondie.

Soit le protocole* suivant:

1. Alice choisit aléatoirement un exposant secret $x \in_R [2^{40}] = \{1,2, \ldots, 2^{40}-1, 2^{40}\}$ de 40 bits, calcule $h_A = 5^x \bmod 2^{61}-1$, et transmet h_A à Bob.
2. Bob choisit aléatoirement un exposant $y \in_R [2^{40}]$secret de 40 bits, calcule $h_B = 5^y \bmod 2^{61}$- 1, et transmet h_B à Alice.
3. Alice, après réception de h_B, output la clé $k_A = h_B^x \bmod 2^{61}-1$.
4. Bob, après réception de h_A, output la clé $k_B = h_A^y \bmod 2^{61}-1$.

Les parties 1 et 3 du protocole correspondent au code de Alice, et les parties 2 et 4 au code de Bob.

Écrivez le code pour ces 4 algorithmes en langage C, python, Java ou autre. N'oubliez pas de vous assurer que votre protocole soit calculatoirement efficace. En particulier, utilisez l'exponentiation modulaire rapide (square-and-multiply**).

---

*Les exposants sont définis comme des Nombres de Mersenne (1588-1648)
Marin Mersenne a étudié les propriétés des nombres entiers, nommés en son hommage, nombre de Mersenne.
Un nombre de Mersenne : Pour tout entier n, le n-ième nombre de Mersenne M_n est un entier de la forme d'une puissance n-ième de 2 moins un soit :

$$\forall n \in \mathbb{N}^* \; ; \; M_n = 2^n - 1$$

Ce résumé est incomplet, aussi il ne vise qu'à expliquer l'établissement du contexte de la création des clés.

**Square-and-multiply: $a^b \bmod n$
1. Il faut convertir l'exposant b en binaire
2. Parcourir les bits de gauche à droite (ou droite à gauche selon l'implémentation)
3. À chaque bit :
   - Square : on élève le résultat courant au carré
   - Multiply : Si le bit est 1, on multiplie par la base a.

Dans ce qui suit, une personne jouera Alice, une jouera Bob, et une Eve.

(a) Exécutez le protocole dans deux contextes distincts :

- **Scénario 1 (Écoute passive)** : Eve intercepte et observe tous les messages entre Alice et Bob sans les modifier.
- **Scénario 2 (Attaque de l'homme du milieu, ou MITM)** : Eve intercepte les messages et établit deux sessions distinctes, une avec Alice et une avec Bob, en se faisant passer pour l'autre partie.

Pour chaque scénario, fournissez les traces d'exécution complètes incluant :

- Les valeurs locales (secrets, calculs intermédiaires, clés finales) pour Alice, Bob et Eve
- Les messages observés ou transmis sur le réseau
- Les transcripts complets du point de vue de chaque participant

(b) Analysez la distinguabilité des scénarios : du point de vue d'Alice et de Bob considérés individuellement (sans communication directe hors protocole), existe-t-il un moyen de détecter la différence entre les deux scénarios ? Justifiez votre réponse en termes de vues locales.

(c) Si vous comparer maintenant les outputs k_A et k_B d'Alice et de Bob dans les deux scénarios, dans quel(s) cas a-t-on $k_A = k_B$? Expliquez pourquoi cette égalité est vérifiée ou non selon le scénario

(d) En utilisant la notation formelle des notes de cours (voir révision IKE), définissez précisément les instances de session quelles sont les $\Pi_A^i,\ \Pi_B^i$ pour chaque scénario. Identifiez également les partenaires de session selon la définition du modèle de sécurité.

(e) (QUESTION BONUS/ et difficile) Supposons maintenant qu'Eve dispose d'un oracle capable de calculer le logarithme discret en base 5 modulo $2^{61}-1$ pour environ 1% des valeurs aléatoires. Décrivez une stratégie qu'Eve pourrait utiliser pour augmenter ses chances de

déchiffrer les communications. Quelle serait approximativement sa probabilité de succès après avoir observé *n* sessions indépendantes du protocole ?

2. Analyse du chiffrement ‘’textbook’’ RSA avec module spécifique

Soit la sortie *(N,e,d)=(221,7,103)* d'un algorithme GenModulus pour RSA.

Soit $\Pi_{ch-t-RSA}$ = *(Gen1, E1, D1),* le schéma de chiffrement ‘’textbook’’ RSA (vous pouvez supposer pour les besoins du devoir que *N* est au plus 8 ou 16 bits, et que les messages sont dans $\mathbb{Z}_N^*$ sauf indication contraire).

(a) Implémentez les algorithmes de chiffrement *E1* et de déchiffrement *D1* en langage C, Python, Java ou autre. Assurez-vous d'utiliser l'exponentiation modulaire efficace (comme la décomposition de l’exposant en base 2).

Dans ce qui suit, une personne jouera Alice, une jouera Bob, et une Eve.

(b) Alice génère les paramètres correspondant à *(N,e,d)=(221,7,103)* via Gen1 et diffuse publiquement pk=(N,e). Répétez le scénario suivant pour chacun des messages : $m \in \{2,6,13,17\}$.

Pour chaque message:

- Bob calcule le cryptogramme *c=E1(pk,m)* et l'envoie à Alice via Eve (qui observe sans modifier)
- Alice déchiffre c pour récupérer le message *m*

Fournissez les traces d'exécution pour chacun des quatre messages, incluant :

- Côté Alice : *m, sk, c* (reçu), *m′* (déchiffré)
- Côté Eve : *pk, c* (observé)
- Côté Bob : *pk, m, c* (calculé)

(c) Pour chacune des quatre exécutions, déterminez si Eve peut récupérer le message m avec l'information dont elle dispose. Si oui, expliquez la méthode précise qu'elle doit utiliser. Si non, expliquez quelle propriété cryptographique l'en empêche.

(d) Exécutez maintenant le protocole avec les messages *m=0, m=1*et *m=221*. Que remarquez-vous dans chacun de ces trois cas ? Expliquez mathématiquement pourquoi ces comportements se produisent.

(e) (QUESTION BONUS/ et difficile) Bob souhaite envoyer deux messages *m1* et *m2* à Alice. Par erreur, il envoie *c1=E1(pk, m1)* et *c2=E1(pk, m2)* où *m1* m2≡ m3 (mod N)* pour un certain

m3 connu d'Eve. Décrivez comment Eve peut exploiter cette information pour obtenir une relation entre les messages. Cette attaque illustre quelle propriété problématique du ‘’textbook’’ RSA?

3. Signatures ‘’textbook’’ RSA et forgerie

Soit $\Pi_{sign-t-RSA}$ = *(Gen2, Sign2, Verif2)*, le schéma de signature digitale ‘’textbook’’ RSA (vous pouvez supposer pour les besoins du devoir que *N* est au plus 8 ou 16 bits, et que les messages sont valides).

(a) Écrivez le code pour les deux algorithmes *Sign2* et *Verif2*, en langage C, python, Java ou autre.

Dans ce qui suit, une personne jouera Alice, une jouera Bob, et une Eve.

(b) Alice génère les paramètres via *Gen2* avec les mêmes valeurs qu'à la question 2 et diffuse publiquement pk. Répétez le scénario suivant quatre fois pour les messages : $m \in \{2,6,13,17\}$.

Pour chaque message :

- Alice génère la signature $\sigma = Sign2\ (sk, m)$ et envoie (m, σ) à Bob via Eve
- Pour les messages m = 2 et $m = 6$: Eve transmet fidèlement
- Pour le message $m = 13$: Eve modifie la signature en $\sigma' = \sigma + 1 \bmod N$
- Pour le message $m = 17$: Eve modifie le message en $m' = 15$ (mais garde σ inchangé)
- Bob reçoit (m, σ) ou (m', σ') et calcule le bit de vérification *v*

Fournissez les traces complètes des quatre exécutions, incluant :

- Côté Alice : *m, sk, σ*
- Côté Eve : (m, σ) reçu, (m', σ') envoyé (si modification)
- Côté Bob : *m (*ou *m’), pk, σ (*ou *σ’), v*

(c) Eve souhaite maintenant envoyer un message m_e de son choix à Bob avec une signature valide, sans connaître la clé secrète d’Alice. Peut-elle y parvenir avec une probabilité de succès raisonnable ? Décrivez précisément une stratégie qu’Eve pourrait utiliser en exploitant les propriétés du RSA textbook.

(d) (QUESTION BONUS/ et difficile) Supposons qu’Eve ait observé les signatures de plusieurs messages $(m_1, \sigma_1), (m_2, \sigma_2), \ldots, (m_k, \sigma_k)$. Expliquez comment Eve peut utiliser ces observations pour forger une signature valide pour un message $m^* = m_i * m_j \bmod N$ (où $i \neq j$) sans connaître la clé privée d’Alice. Implémentez cette attaque avec les messages de la partie **(b)** . Cette vulnérabilité illustre quelle propriété fondamentale qui rend ‘’texbook’’ RSA inadéquat pour la signature ?

Suggestions :

Bibliographie :
Un de vos collègues a fait la recommandation suivante : L'utilisation de Zotero (extension dans votre navigateur) afin de créer un catalogue des liens de vos consultations sur le Net. À la fin, cela donne une bibliographie que vous pouvez insérer à la fin de votre PDF. C'est gratuit comme outil et très pratique pour la correction de vos travaux par les auxiliaires d'enseignement.
https://zotero.org

Code dans un PDF :
Nous avons reçu des fichiers avec l'extension py comme document en plus du document PDF de vos travaux. Ce n'est pas l'approche qu'on favorise. Il existe d'autres options et je ne vais pas en imposer une, mais on m'indique que celle-ci fonctionne correctement :
Easy Code Formatter
Outil pour faciliter l'intégration du code dans un document Word.