Crypto Symétrique
TD L'authentification de personnes

Exercice I

Un institut de recherche met en place un protocole d'authentification, qui a lieu entre un prouveur et un vérificateur. Le vérificateur est monté sur une porte, tandis que les prouveurs sont des cartes à puces. Chaque prouveur P partage une clé K_P avec le vérificateur. Pour chaque utilisation du protocole, le vérificateur choisit un nombre aléatoire N, et ensuite calcule H(N) pour une fonction de hachage donnée (pour ce protocole, ils utilisent SHA1). Le prouveur doit calculer $\mathrm{HMAC}_{K_P}(H(N))$.

Un attaquant qui écoute sur le canal de communication cible un certain prouveur, dont l'authentification les 6 deniers jours a résulté dans ces valeurs, que l'attaquant stocke dans un log.

Jour	**Requête : $H(N)$**	**Réponse : HMAC**
1	356a192b7913b04c54574d18c28d46e6395428ab	440d4154d5f1008aa1f39a4bf24068025c4836fb
2	da4b9237bacccdf19c0760cab7aec4a8359010b0	c85dbfd4f2c0f0296315b602da934be728289ccc
3	1b6453892473a467d07372d45eb05abc2031647a	0fad0409db56cd1648137a48d52b0d001a925b3e
4	fe5dbbcea5ce7e2988b8c69bcfdfde8904aabc1f	d07f1f27273e04601e00b8f3fd968af2a3568b16
5	1574bddb75c78a6fd2251d61e2993b5146201319	bc757ffc864e963a3ad60b35a523ba5619a070ef
6	cb4e5208b4cd87268b208e49452ed6e89a68e0b8	035a5cd41125dd3120f0454bfd4dc791a301ea20

Ces valeurs semblent aléatoires… mais notre attaquant, qui est malin et persistant, soupçonne une faiblesse au niveau des messages envoyés par le vérificateur. Par un coup de chance, il trouve que 356a192b7913b04c54574d18c28d46e6395428ab correspond au SHA1 de la valeur "1"; que fe5dbbcea5ce7e2988b8c69bcfdfde8904aabc1f correspond au SHA1 de "8" et que cb4e5208b4cd87268b208e49452ed6e89a68e0b8 est le SHA1 de "32".

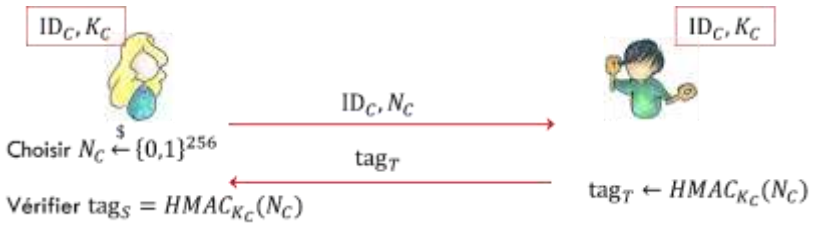
1. Un profitant du fait que l'accès est géré, côté prouveur, en utilisant des cartes à puces, l'attaquant utilise un émulateur du vérificateur pour envoyer un message challenge au prouveur ciblé, avant que ce dernier ait l'opportunité de s'authentifier le 7ème jour. Le prouveur lui répond avec un message response. En stockant et en réutilisant la valeur response, l'attaquant réussit à s'authentifier.

   Utilisez les valeurs ci-dessus ainsi qu'un calculateur de valeurs SHA1 (par exemple http://www.sha1-online.com/) pour trouver le message envoyé par l'attaquant au prouveur et à quoi correspond-il ?

2. Est-ce que la sécurité du schéma change si on utilise une meilleure fonction de hachage ou même un oracle aléatoire ?

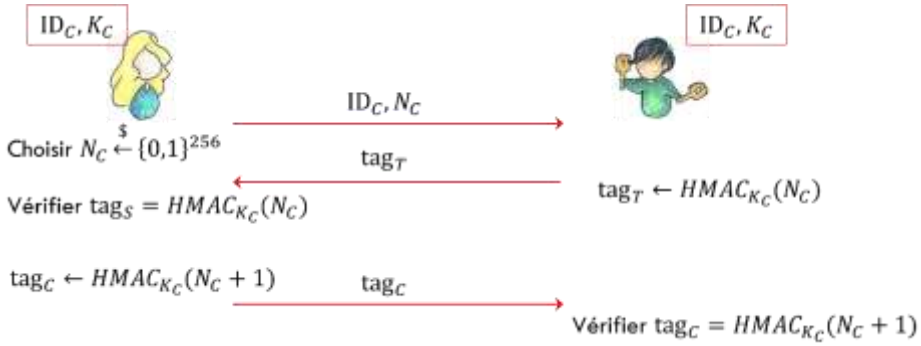
Exercice II

Dans un réseau de capteurs, chaque capteur possède un identifiant ID_C et partage un clé K_C avec un terminal T. Le protocole suivant permet au terminal de s'authentifier auprès d'un capteur :



Ce protocole est prouvablement sécurisé (authentification unilatérale terminal vers capteur) dans le modèle de l'oracle aléatoire.

On veut transformer ce protocole dans un schéma qui fournit une authentification mutuelle. Quelqu'un propose le protocole suivant :



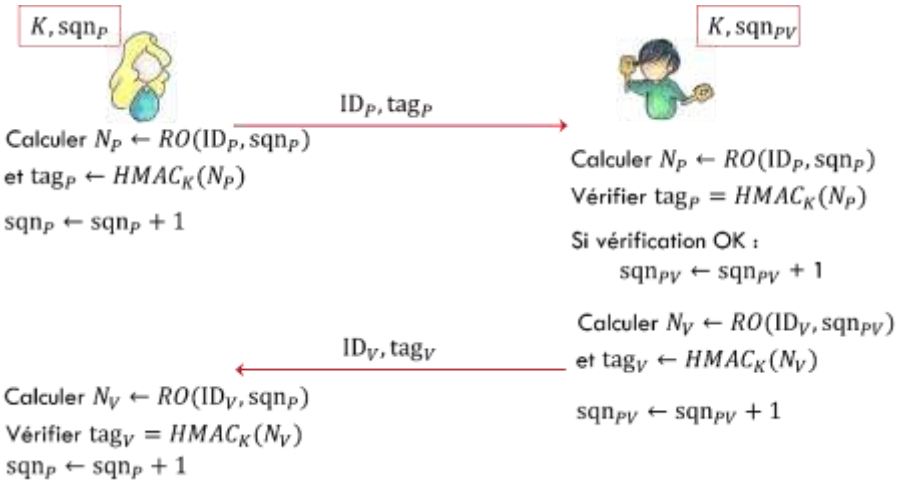
1. Pourquoi est-il nécessaire d'inclure l'identifiant ID_C dans le premier message du capteur ?
2. Comment un attaquant peut-il se faire passer par le terminal auprès du capteur ?
3. Et l'inverse ?
4. On idéalise HMAC en tant qu'oracle aléatoire. Est-ce que cela améliore la sécurité du protocole ?
5. Pouvez-vous modifier le protocole pour obtenir un schéma sécurisé ?
6. Pour le protocole modifié nous voulons fournir une preuve de sécurité dans le modèle de l'oracle aléatoire. Pour ce faire :
    1. Pouvez-vous raisonner pourquoi le protocole est sécurité, à un niveau intuitif ?
    2. Donnez la structure d'une preuve potentielle
    3. Pouvez-vous formaliser une preuve ?

Exercice III

Certains périphériques, comme par exemple les anciennes cartes SIM, ne peuvent pas générer des nombres aléatoires. Quand un protocole nécessite de la fraîcheur, mais les participants ne peuvent pas la générer, on peut utiliser un état -- c'est-à-dire une valeur (secrète ou publique), qui évolue au fil du temps.

Dans le schéma ci-dessous un prouveur (qui n'est pas capable de générer des nombres aléatoires) interagit avec un vérificateur. Chaque prouveur partage une clé K avec le vérificateur. De plus, le prouveur a un état sqn_P (et de l'autre côté, pour chaque prouveur, le vérificateur garde aussi un état sqn_{PV}). Le prouveur a un identifiant ID_P et le vérificateur, un autre identifiant ID_V.

Le protocole suivant veut assurer l'authentification mutuelle entre le prouveur et le vérificateur.



1. Montrez que, dans l'absence de l'attaquant, ce protocole marche.
2. Pouvez-vous trouver une attaque de déni de service sur ce protocole, mené par un attaquant au milieu tel que, en n'intervenant que dans une seule session, l'attaquant nie le service au prouveur pour toujours.
3. Quelle est la faille de design exploitée ?

Le prouveur n'est pas capable de générer des aléas, mais le vérificateur si. On change le protocole pour rajouter une procédure de resynchronisation. Si la vérification de la valeur tag_P ne réussit pas du côté du vérificateur, alors le vérificateur envoie un message spécial Resynchronisation et ensuite :

- Il génère une valeur aléatoire N_V, calcule $\mathrm{Blind} \leftarrow \mathrm{HMAC}_K(N_V)$ et envoie : $N_V, \mathrm{Blind} \oplus \mathrm{sqn}_{PV}$ et $\mathrm{HMAC}_K(\mathrm{ID}_P, \mathrm{ID}_V, \mathrm{Blind} \oplus \mathrm{sqn}_{PV})$
- Ayant reçu ce message, le prouveur vérifie $\mathrm{HMAC}_K(\mathrm{ID}_P, \mathrm{ID}_V, \mathrm{Blind} \oplus \mathrm{sqn}_{PV})$ et, si la vérification marche, alors le prouveur utilise Blind et retrouve sqn_{PV}. Le prouveur met $\mathrm{sqn}_P \leftarrow \mathrm{sqn}_{PV}$. Ensuite le prouveur recommence le protocole d'authentification du début, en utilisant la nouvelle valeur.

4. Montrez comment le rajout de la resynchronisation influence les attaques de déni de service que vous avez trouvées.