Crypto, Sécurité, Réseaux

TD 4

Le protocole TLS est un des trois protocoles au plus utilisés aujourd'hui, avec SSH et IPSec. Il assure une des fonctionnalités de sécurité les plus importantes aujourd'hui : notamment la confidentialité, authenticité et intégrité des messages envoyés sur l'Internet, par courriel, ou même par Voice-over-IP. Ce protocole se déroule entre *des serveurs* (qui stockent des données) et *des clients* (en pratique, les usagers qui cherchent envoyer ou recevoir des données).

Lorsque le protocole TLS se déclenche pour la navigation sur l'Internet, le protocole http: (utilisé pour permettre l'échange d'informations entre les clients et les serveurs) se transforme dans le protocole https:

Le protocole TLS consiste de deux phases : 1) l'échange de clé authentifié, et 2) l'échange sécurisé des messages. Dans la première phase du protocole, le client et le serveur échangent des informations leurs permettant de calculer une paire de clés privés. Dans une deuxième phase, les clés sont utilisés pour chiffrer et authentifier les messages qu'ils échangent.

Les clés calculées par TLS doivent être fraîches à chaque session : comme ça, si la clé d'une session est compromise, toute autre session du protocole reste sauve. De plus, nous voulons aussi que même si les clés long-terme que les parties peuvent avoir sont compromises, alors toute session qu'a été déjà finie doit être sauve. Cette dernière notion est critique pour les infrastructures en réseau et s'appelle **Perfect Forward Secrecy**.

TLS en mode RSA

Le protocole TLS opère par défaut dans un mode qui permet l'authentification seulement du serveur auprès du client, et non l'inverse. On considère que tout le monde peut accéder à un serveur, mais il sera impossible pour quelqu'un de se faire passer par un serveur légitime. Un système de certification est utilisé : le serveur est en possession d'une clé publique certifié et doit prouver sa connaissance de la clé privée pour finaliser l'échange de clé.

TLS à trois modes de fonctionnement pour assurer l'échange de clé : TLS-RSA, TLS-DH et TLS-DHE.

Pour TLS-RSA, le serveur est en possession d'un tuple de clés de chiffrement RSA (*sk, pk*), où sk dénote la clé privée et pk dénote la clé publique. Le certificat *Cert* dénote le certificat signé pour la clé *pk* par une autorité légitime de certification. Nous utiliserons les notations ci-dessous :

- $a \xleftarrow{\$} A$: la variable a est choisie aléatoirement de l'ensemble A.
- $\mathrm{RSA.Enc}(pk; m)$: un chiffrement RSA du message m avec la clé publique pk
- $\mathrm{RSA.Dec}(sk; c)$: le déchiffrement d'un chiffré c avec la clé privée sk
- $\{0,1\}^k$: l'ensemble de chaines de bits de taille k, pour une valeur de k entière

- $\mathrm{HMAC(k; m)}$: l'exécution de la fonction HMAC avec la clé k et sur le message m
- $\mathrm{H(a, b, c \ldots)}$: l'éxécution de la fonction de hachage H sur la concaténation des valeurs données en entrée
- $AE(k_1, k_2; m)$: un chiffrement authentifié du message m, avec la clé k_1 pour le chiffrement et la clé k_2 pour l'authentification. Les deux algorithmes sont à clé symétrique. Il y a plusieurs types de chiffrement authentifié en TLS. Les chiffrés par flot de la gamme RC sont découragés, tout comme le chiffrement AES en mode CBC. Le chiffrement authentifié qu'on encourage au plus aujourd'hui c'est une méthode qui s'appelle «Authenticated Encryption with Additional Data » -- AEAD, notamment en mode AES-GCM ou ChaCha-Poly1305.

Voici le protocole TLS en mode RSA :

Client		Serveur $(sk, pk, Cert)$	Messages :
$n_c \xleftarrow{\$} \{0,1\}^{224}$	$\xrightarrow{M_1}$		M_1 : le temps (sur 4 octets) n_C (sur 28 octets) Une liste d'options des algos de chiffrement/hachage/etc à utiliser
		$n_S \xleftarrow{\$} \{0,1\}^{256}$	
	$\xleftarrow{M_2, M_3}$		M_2 : n_S (sur 32 octets) Un choix parmi les options données en M_1
Vérifier certificat s'il ne vérifie pas, erreur			M_3 : le certificat $Cert$, qui inclue la clé publique pk
$\mathrm{pms} \xleftarrow{\$} \{0,1\}^{384}$			M_4 : le chiffré c
$c \coloneqq \mathrm{RSA.Enc}(pk; \mathrm{pms})$			M_5: $AE(K_C^{Enc}, K_C^{MAC}; H_C)$: le chiffrement authentifié de H_C avec les clés du client
$ms \coloneqq \mathrm{HMAC}(\mathrm{pms};\ n_C, n_S)$ $K \coloneqq \mathrm{HMAC}(ms; n_c, n_S)$	$\xrightarrow{M_4, M_5}$	Déchiffrer c : $\mathrm{pms} = \mathrm{RSA.Dec}(sk;\ c)$	M_6: $AE(K_S^{Enc}, K_S^{MAC}; H_S)$, le chiffrement authentifié de H_S avec les clés du serveur
$K_C^{Enc}, K_C^{MAC}, K_S^{Enc}, K_S^{MAC} \xleftarrow{} K$		Calcule (ms, K) et toute sous-clé comme le Client	
$H_C \coloneqq H(M_1..M_3, c)$ $CFin \coloneqq HMAC(ms; l, H_c)$ où l est une constante $H_S \coloneqq H(M_1..M_3, c, H_c)$ $SFin \coloneqq HMAC(ms; l', H_S)$ où l' est une constante		Vérifie et déchiffre M_5, vérifie H_C par rapport aux messages reçus et envoyés $H_S \coloneqq H(M_1..M_3, c, H_c)$ $SFin \coloneqq HMAC(ms; l', H_S)$	
	$\xleftarrow{M_6}$		
Vérifie et déchiffre M_6, vérifie H_S par rapport aux messages reçus et envoyés			

Précisions :

L'algorithme représenté ci-dessus n'est qu'une indication de la complexité du protocole TLS en mode RSA. J'ai ignoré quelques détails : par exemple la valeur maître ms a une taille de 48 octets = 384 bits. Elle s'obtient en coupant la sortie entière de HMAC (dont la taille dépend de la taille de sortie de la fonction de hachage). La taille des clés $K_C^{Enc}, K_C^{MAC}, K_S^{Enc}, K_S^{MAC}$ n'est pas une constante : elle dépend de la méthode de chiffrement authentifié choisie par les deux parties. Dans le cas de certaines méthodes deux autres la clé K doit être partagée en 6 morceaux : les 4 clés, ainsi que deux vecteurs d'initialisation

Bien comprendre le protocole

Regardez le protocole et répondrez aux questions suivantes :

- Pourquoi est-il important de fournir le certificat du serveur ?
- Comment l'authentification du serveur est-elle assurée ?
- Qu-est-ce qu'assure la sécurité de la valeur pms ? Pourquoi est-elle tellement importante ? (la technique qu'on utilise pour le pms s'appelle un mécanisme d'encapsulation de clé -- Key Encapsulation Mechanism)
- Les messages M_5, M_6 assurent ce qu'on appelle la « confirmation de clés ». Selon vous, quelle est leur fonctionnalité ?
- Quel est le rôle de la fonction de hachage utilisé dans les messages de confirmation de clé ? De quelle propriété de sécurité a-t-on besoin ?
- Disons qu'on peut obtenir la clé privée du serveur. Est-ce que la sécurité des sessions passées est assurée (Forward Secrecy) ?
- Pourquoi a-t-on besoin des deux nonces n_C et n_S ? Est-ce qu'une seule nonce (par exemple venant du client) suffirait ?

TLS en mode DH

TLS-DH assure le calcul des clés à partir d'un échange Diffie-Hellman, qui est authentifié du côté du serveur. Il y a deux différences principales entre les modes TLS-DH et TLS-DHE : bien que les deux soient basés sur l'échange de clé Diffie Hellman, la valeur DH utilisée par le serveur est *statique* : notamment, elle ne change pas d'une session à l'autre ; par contre, pour le mode TLS-DHE, le serveur choisit les paramètres du groupe DH, ainsi que l'élément DH à chaque session. La deuxième différence entre les deux modes concerne le certificat du serveur. En TLS-DH, le certificat atteste la valeur publique DH que le serveur utilise (rappelez-vous qu'il s'agit d'une valeur statique, ainsi que les paramètres du groupe), tandis que pour TLS-DHE, le certificat est fait pour une clé de signatures.

Voyons donc le protocole TLS en mode DH (voire la page suivante ; le texte rouge sur jaune indique des différences par rapport à TLS-RSA). Répondez aux questions suivantes :

- Comment l'authentification du serveur est-elle assurée ?
- L'élément DIffie Hellman du serveur est statique. Qu'est-ce qu'assure la fraîcheur de la valeur pms (qui devrait être différente à chaque session) ?

- En mode TLS-RSA, le secret pms est choisi aléatoirement (sur l'ensemble de valeurs à 384 bits). Quel type de valeur prend le secret pms ? Qu'est-ce que garantit sa sécurité ? (voir les problèmes considérées difficiles pour les valeurs DH)
- Quels sont les paramètres de groupe qu'il faut spécifier dans le certificat ? Pourquoi faut-il les spécifier ?
- Est-ce que le protocole en mode DH assure la propriété de Forward Secrecy ?

Client		Serveur $(x, g^x, Cert)$	Messages :
$n_c \xleftarrow{\$} \{0,1\}^{224}$	$\xrightarrow{M_1}$		M_1 : le temps (sur 4 octets) n_C (sur 28 octets) Une liste d'options des algos de chiffrement/hachage/ etc.
		$n_S \xleftarrow{\$} \{0,1\}^{256}$	
	$\xleftarrow{M_2, M_3}$		M_2 : n_S (sur 32 octets) Un choix parmi les options données en M_1
Vérifier certificat s'il ne vérifie pas, erreur			M_3 : le certificat $Cert$, qui inclue la partie publique de l'élément DH, notamment g^x et les paramètres du groupe DH
Choisir y et calculer g^y pms := $(g^x)^y$			M_4 : l'élément g^y
$ms := \text{HMAC}(\text{pms};\ n_C, n_S)$ $K := \text{HMAC}(ms; n_c, n_S)$	$\xrightarrow{M_4, M_5}$	Calculer : pms = $(g^y)^x$	M_5: $AE(K_C^{Enc}, K_C^{MAC}; H_C)$: le chiffrement authentifié de H_C avec les clés du client
$K_C^{Enc}, K_C^{MAC}, K_S^{Enc}, K_S^{MAC} \leftarrow K$		Calcule (ms, K) et toute sous-clé comme le Client	
$H_C := H(M_1..M_3, c)$ $CFin := HMAC(ms; l, H_c)$ où l est une constante $H_S := H(M_1..M_3, c, H_c)$ $SFin := HMAC(ms; l', H_S)$ où l' est une constante		Vérifie et déchiffre M_5, vérifie H_C par rapport aux messages reçus et envoyés	M_6: $AE(K_S^{Enc}, K_S^{MAC}; H_S)$, le chiffrement authentifié de H_S avec les clés du serveur
	$\xleftarrow{M_6}$	$H_S := H(M_1..M_3, c, H_c)$ $SFin := HMAC(ms; l', H_S)$	
Vérifie et déchiffre M_6, vérifie H_S par rapport aux messages reçus et envoyés			

TLS en mode DHE

Comme l'on avait commenté dans la section précédente, TLS en mode DHE diffère légèrement du protocole TLS-DH. Le « E » dans TLS-DHE dénote le mot « éphémère » : notamment, l'élément DH du serveur diffère à chaque session. Le serveur possède une paire de clés de signature (en pratique pour des signatures RSA ou DSA), qu'on va noter (sk, pk). On fait la notation suivante (en plus de celles déjà indiquées) :

- $\mathrm{Sign}(sk; m)$: la signature sur le message m avec la clé sk.

Client		Serveur $(sk, pk, Cert)$	Messages :
$n_c \xleftarrow{\$} \{0,1\}^{224}$	$\xrightarrow{M_1}$	$n_S \xleftarrow{\$} \{0,1\}^{256}$ Générer des paramètres DH : p, q, g Choisir $x \xleftarrow{\$} \{1 \dots q-1\}$, calculer $g^x \bmod p$	M_1 : le temps (sur 4 octets) n_C (sur 28 octets) Une liste d'options des algos de chiffrement/hachage/ etc. M_2 : n_S (sur 32 octets) Un choix parmi les options données en M_1 M_3 : le certificat $Cert$, qui inclue la cle publique pk du serveur M_4 : les éléments d'échange de clé utilisés par le serveur, notamment (p, q, g, g^x) M_5 : $\mathrm{Sign}(sk; H(n_C, n_S, p, g, q, g^x))$, notamment une signature avec la clé privée sk sur l'hachée des valeurs indiquées M_4 : l'élément g^y M_5: $AE(K_C^{Enc}, K_C^{MAC}; H_C)$: le chiffrement authentifié de H_C avec les clés du client M_6: $AE(K_S^{Enc}, K_S^{MAC}; H_S)$, le chiffrement authentifié de H_S avec les clés du serveur
Vérifier certificat s'il ne vérifie pas, erreur Vérifier M_5 ; s'il ne vérifie pas, erreur	$\xleftarrow{M_2, M_3, M_4}$		
Choisir y et calculer g^y pms $:= (g^x)^y \bmod p$	$\xleftarrow{M_5}$		
$ms := \mathrm{HMAC}(\mathrm{pms};\ n_C, n_S)$ $K := \mathrm{HMAC}(ms; n_c, n_S)$ $K_C^{Enc}, K_C^{MAC}, K_S^{Enc}, K_S^{MAC} \leftarrow K$ $H_C := H(M_1 .. M_3, c)$ $CFin := HMAC(ms; l, H_c)$ où l est une constante $H_S := H(M_1 .. M_3, c, H_c)$ $SFin := HMAC(ms; l', H_S)$ où l' est une constante	$\xrightarrow{M_6, M_7}$	Calcule (ms, K) et toute sous-clé comme le Client Vérifie et déchiffre M_7, vérifie H_C par rapport aux messages reçus et envoyés $H_S := H(M_1 .. M_3, c, H_c)$ $SFin := HMAC(ms; l', H_S)$	
Vérifie et déchiffre M_8, vérifie H_S par rapport aux messages reçus et envoyés	$\xleftarrow{M_8}$		

Répondez aux questions suivantes :

- Quels inconvénients peut avoir le fait de laisser le serveur choisir les paramètres du groupe DH ?

- Comment s'authentifie le serveur dans ce scénario ? Quelle(s) autre(s) valeur(s) authentifie-t-il ? Pourquoi ?
- Est-ce que ce mode garantit la Forward Secrecy ?
- Un mode alternatif d'exécution pour TLS-DHE prévoit de ne pas envoyer de signature, donc de ne pas envoyer le message M_5 . Qu'est-ce qu'on perd dans ce cas ? Qu'est-ce qu'on gagne ?

Un manque de sécurité en TLS

Dans son format actuel, TLS est un protocole souvent utilisé en pratique. Toutefois, depuis 2008 (quand la version actuelle de TLS a été déployée), des attaques diverses contre ce protocole ont visé l'échange de clé, les méthodes de chiffrement authentifié, ainsi que les implémentations de TLS. Dans la suite, nous allons essayer de comprendre quelques vulnérabilités de ce protocole.

- En dehors des nonces n_C et n_S, le premier message du protocole contient également des choix sur les algorithmes à utiliser pour l'échange de clé : la taille de la clé RSA et/ou du groupe DH, les algorithmes de chiffrement authentifié, la fonction de hachage, etc. Est-ce que ces valeurs sont jamais confirmées dans le cours du protocole ? Dans quel message(s) et par quel mécanisme ?

- Pour les attaques FREAK et LogJam, l'attaquant joue le rôle d'un homme au milieu, qui se pose entre un client et un serveur légitime. Son but est d'apprendre quels messages les deux parties échangent (alors, sont but est de casser la sécurité du canal établi par TLS). Pour FREAK, l'attaquant se base sur le fait qu'il est possible de factoriser un module RSA à 512 bits dans quelques secondes. Comment peut un tel attaquant casser la sécurité de TLS-RSA ?

- Comment peut-on résoudre ce problème ?