Groupe de travail Réseau
Request for Comments : 4082
Catégorie : Information
Traduction Claude Brière de L'Isle

A. Perrig & D. Song, Carnegie Mellon University
R. Canetti, IBM
J. D. Tygar, University of California, Berkeley
B. Briscoe, BT
juin 2005

Authentification de flux tolérante aux pertes en temps efficace (TESLA) : introduction à la transformation d'authentification de source de diffusion groupée

Résumé
Le présent document introduit l'authentification de flux tolérante aux pertes en temps efficace (TESLA, *Timed Efficient Stream Loss-tolerant Authentication*). TESLA permet à tous les receveurs de vérifier l'intégrité et d'authentifier la source de chaque paquet dans les flux de données en diffusion ou diffusion groupée. TESLA n'exige pas de confiance entre les receveurs, utilise des opérations de faible coût par paquet chez l'envoyeur et le receveur, peut tolérer tous les niveaux de pertes sans retransmission, et n'exige pas d'état par receveur chez l'envoyeur. TESLA peut protéger les receveurs contre les attaques de déni de service dans certaines circonstances. Chaque receveur doit être en gros synchronisé avec la source afin de vérifier les messages, mais par ailleurs les receveurs n'ont pas à envoyer de messages. TESLA seul ne peut prendre en charge la non répudiation de la source de données aux tiers.

Ce document d'information est destiné à aider à la rédaction de spécifications sûres et normalisables pour des protocoles fondés sur TESLA dans différents contextes.

Table des matières

1. Introduction

En diffusion groupée, un seul paquet peut atteindre des millions de receveurs. Malheureusement, cela introduit le danger qu'un attaquant peut potentiellement aussi atteindre des millions de receveurs avec un paquet malveillant. Par

l'authentification de la source, les receveurs peuvent s'assurer qu'un paquet en diffusion groupée reçu a pour origine la source correcte. À cet égard, une diffusion groupée est équivalente à une diffusion à un sur-ensemble des receveurs de diffusion groupée.

Dans une communication en envoi individuel, on peut réaliser l'authentification des données par un simple mécanisme : l'envoyeur et le receveur partagent une clé secrète pour calculer un code d'authentification de message (MAC, *Message Authentication Code*) de toutes les données communiquées. Quand un message arrive avec un MAC correct, le receveur est assuré que l'envoyeur a généré ce message. Les mécanismes standard réalisent de cette façon l'authentification de l'envoi individuel ; par exemple, TLS ou IPsec [1], [2].

L'authentification par MAC symétrique n'est pas sûre dans un schéma de diffusion. Considérons un envoyeur qui diffuse des données authentiques à des receveurs qui ne se font pas mutuellement confiance. Le MAC symétrique n'est pas sûr : chaque receveur connaît la clé de MAC et donc pourrait se faire passer pour l'envoyeur et falsifier les messages aux autres receveurs. Intuitivement, on a besoin d'un mécanisme asymétrique pour réaliser une diffusion authentifiée, de telle façon que chaque receveur puisse vérifier l'authenticité des messages qu'il reçoit, sans être capable de générer de messages authentiques. Réaliser cela de façon efficace est un problème stimulant [3].

L'approche standard pour réaliser une telle asymétrie pour l'authentification est d'utiliser un chiffrement asymétrique ; par exemple, une signature numérique. Les signatures numériques ont les propriétés d'asymétrie requises : l'envoyeur génère la signature avec sa clé privée, et tous les receveurs peuvent vérifier la signature avec la clé publique de l'envoyeur, mais un receveur avec la seule clé publique ne peut pas générer une signature numérique pour un nouveau message. Une signature numérique assure la non répudiation, une propriété plus forte que l'authentification. Cependant, les signatures numériques ont un coût élevé : elles ont de gros frais généraux de calcul aussi bien pour l'envoyeur que pour le receveur, et la plupart des signatures ont aussi aussi un coût élevé en bande passante. Comme on suppose des réglages de diffusion où l'envoyeur ne retransmet pas les paquets perdus, et que le receveur veut quand même authentifier immédiatement chaque paquet qu'il reçoit, on aurait besoin d'attacher une signature numérique à chaque message. À cause du coût élevé du chiffrement asymétrique, cette approche nous restreindrait à des flux à faible débit, et aux envoyeurs et receveurs qui ont des matériels puissants. On peut essayer d'amortir une signature numérique sur plusieurs messages. Cependant, cette approche est quand même coûteuse par rapport au chiffrement symétrique, car celui-ci est en général de 3 à 5 ordres de grandeur plus efficace que le chiffrement asymétrique. De plus, l'amortissement direct d'une signature numérique sur de multiples paquets exige la fiabilité, car le receveur a besoin de recevoir tous les paquets pour vérifier la signature. Un certain nombre de schémas suivent cette approche [4], [5], [6], [7]. Voir plus de détails dans [8].

Le présent document présente le protocole d'authentification de flux tolérante aux pertes en temps efficace (TESLA, *Timed Efficient Stream Loss-tolerant Authentication*). TESLA utilise principalement le chiffrement symétrique, et utilise la divulgation retardée pour réaliser les propriétés d'asymétrie requises. Cependant, TESLA exige des horloges plus ou moins synchronisées entre l'envoyeur et les receveurs. Voir les détails au paragraphe 3.3.1. Les schémas qui suivent une approche similaire à celle de TESLA sont [9], [10], [11].

1.1 Notation

Pour noter un indice d'une variable, on utilise le souligné entre le nom de la variable et l'indice ; par exemple, la clé K avec l'indice i est K_i, et la clé K avec l'indice i+d est K_{i+d}. Pour écrire un exposant, on utilise le signe d'omission (^) ; par exemple, la fonction F avec l'argument x exécuté i fois est F^i(x).

2. Fonctionnalités

TESLA fournit des vérifications retardées des données d'authentification et d'intégrité par paquet. L'idée clé pour fournir à la fois l'efficacité et la sécurité est une divulgation retardée des clés. La divulgation retardée des clés résulte en un délai d'authentification. En pratique, le délai est de l'ordre d'un aller-retour (RTT, *round-trip-time*).

TESLA a les propriétés suivantes :
- o Faibles frais généraux de calcul pour la génération et la vérification des informations d'authentification.
- o Faibles frais généraux de communication.
- o Mémoire tampon limitée requise pour l'envoyeur et le receveur, et donc authentification en temps utile pour chaque paquet individuel.
- o Forte robustesse à la perte de paquet.
- o S'adapte à un grand nombre de receveurs.
- o Protège les receveurs des attaques de déni de service dans certaines circonstances si il est configuré de façon appropriée.
- o Chaque receveur ne peut pas vérifier l'authenticité du message si il n'est pas à peu près synchronisé avec la source, et la

synchronisation peut avoir lieu à l'établissement de la session. Une fois la session en cours, les receveurs n'ont pas besoin d'envoyer de messages ou accusés de réception.
- La non répudiation n'est pas prise en charge ; chaque receveur peut savoir que la source d'un flux est authentique, mais ne peut pas le prouver à un tiers.

TESLA peut être utilisé à la couche réseau, à la couche transport, ou à la couche application. L'authentification retardée exige cependant la mise en mémoire tampon des paquets jusqu'à l'achèvement de l'authentification. Certaines applications intolérantes au retard peuvent vouloir traiter les paquets en parallèle de la mise en mémoire tampon tout en attendant l'authentification, tant que le retour en arrière est possible si des paquets se trouvent ultérieurement n'être pas authentifiés. Par exemple, une vidéo interactive peut exécuter des paquets qui attendent encore l'authentification, mais si ils se trouvent ensuite n'être pas authentifiés, elle pourrait arrêter la suite de l'exécution et avertir le visionneur que les x dernières ms ne sont pas authentifiées et devraient être ignorées. Cependant, dans le reste de ce document, pour faire court, on supposera que les paquets ne sont pas traités en parallèle de la mise en mémoire tampon.

2.1 Modèle de menaces et garantie de sécurité

La conception de TESLA le rend sûr contre un adversaire puissant qui a les capacités suivantes :
- Le contrôle complet du réseau. L'adversaire peut espionner, capturer, éliminer, renvoyer, retarder, et altérer les paquets.
- L'accès à un réseau rapide avec un retard négligeable.
- Les ressources de calcul de l'adversaire peuvent être très grandes mais pas illimitées. En particulier, cela signifie que l'adversaire peut effectuer des calculs efficaces, comme calculer un nombre raisonnable d'applications de fonctions pseudo aléatoires et de MAC avec un délai négligeable. Néanmoins, l'adversaire ne peut pas trouver la clé d'une fonction pseudo aléatoire (ou la distinguer d'une fonction aléatoire) avec une probabilité non négligeable.

Les propriétés de sécurité de TESLA garantissent que le receveur n'accepte jamais M_i comme un message authentique à moins que l'envoyeur envoie réellement M_i. Un schéma qui fournit cette garantie est appelé un schéma d'authentification de diffusion groupée sûr.

Parce que TESLA s'attend à ce que le receveur mette les paquets dans une mémoire tampon avant l'authentification, le receveur doit se protéger contre une potentielle attaque de déni de service (DoS) due à un flux de paquets bogués (voir le paragraphe 3.8).

2.2 Hypothèses

TESLA fait les hypothèses suivantes afin de fournir la sécurité :

1. L'envoyeur et le receveur doivent être grossièrement synchronisés. Précisément, chaque receveur doit être capable de calculer une limite supérieure au décalage de l'horloge du receveur par rapport à l'horloge de l'envoyeur. On note cette quantité D_t. (C'est-à-dire que D_t = heure de l'envoyeur - heure du receveur). On note qu'une limite supérieure de D_t peut être facilement obtenue via un simple échange de deux messages. (Un tel échange peut être porté sur tout protocole sûr d'initialisation de session. Autrement, des protocoles standard comme NTP [15] peuvent être utilisés.

2. TESLA DOIT être amorcé à l'établissement de la session par un système régulier d'authentification des données. Une option est d'utiliser pour cela un algorithme de signature numérique, et dans ce cas le receveur est obligé d'avoir une copie authentique du certificat de clé publique de l'envoyeur ou un certificat de clé racine dans le cas d'une infrastructure de clé publique (PKI, *public-key infrastructure*). Autrement, cet établissement d'initialisation peut être fait en utilisant tout protocole sûr d'initialisation de session.

3. TESLA utilise des MAC et des fonctions pseudo aléatoires (PRF, *pseudo-random function*) chiffrés. Celles-ci DOIVENT être cryptographiquement sûres. Plus de détails sur l'instanciation des MAC et PRF sont au paragraphe 3.4.

On souligne que la sécurité de TESLA NE s'appuie PAS sur une hypothèse sur le délai de propagation du réseau.

3. Protocole TESLA de base

TESLA est décrit dans plusieurs publications académiques : un livre sur la sécurité de la diffusion [12], un article de journal [13], et deux textes de conférences [7], [14]. Prière de se référer à ces publications pour des preuves approfondies de sécurité, les résultats expérimentaux, etc.

On souligne d'abord les idées principales qui sous-tendent TESLA.

3.1 Canevas du protocole

Comme on l'explique dans l'introduction, l'authentification de la diffusion exige une source d'asymétrie. TESLA utilise le temps pour l'asymétrie. On s'assure d'abord que l'envoyeur et les receveurs sont en gros synchronisés comme décrit ci-dessus. Ensuite, l'envoyeur forme une chaîne unidirectionnelle de clés, dans laquelle chaque clé dans la chaîne est associée à un intervalle de temps (disons, une seconde). Voici l'approche de base :

- L'envoyeur attache un MAC à chaque paquet. Le MAC est calculé sur le contenu du paquet. Pour chaque paquet, l'envoyeur utilise la clé courante de la chaîne unidirectionnelle comme clé de chiffrement pour calculer le MAC.

- L'envoyeur divulgue une clé de la chaîne unidirectionnelle après un délai pré-défini (par exemple, la clé utilisée dans l'intervalle de temps i est divulguée à l'intervalle de temps i+3).

- Chaque receveur reçoit le paquet. Chaque receveur connaît le programme de divulgation des clés et, comme il a une limite supérieure sur l'heure locale chez l'envoyeur, il peut vérifier que la clé utilisée pour calculer le MAC n'a pas encore été divulguée par l'envoyeur. Si ce n'est pas le cas, le receveur met alors le paquet dans sa mémoire tampon. Autrement, le paquet est éliminé à cause de l'incapacité à l'authentifier. Noter qu'on se sait pas de façon sûre si un "paquet tardif" est bogué ou simplement un paquet retardé. On élimine le paquet parce que on est incapable de l'authentifier. (Bien sûr, une mise en œuvre peut choisir de ne pas éliminer les paquets et de les utiliser non authentifiés.)

- Chaque receveur vérifie que la clé divulguée appartient à la chaîne hachée (en la comparant aux clés précédemment libérées dans la chaîne) et ensuite vérifie que le MAC est correct. Si le MAC est correct, le receveur accepte le paquet.

Noter que les chaînes unidirectionnelles ont la propriété que si des valeurs intermédiaires de la chaîne unidirectionnelle sont perdues, elles peuvent être recalculées en utilisant les valeurs suivantes de la chaîne. Même si certaines divulgations de clés sont perdues, un receveur peut récupérer les clés correspondantes et vérifier que les paquets antérieurs sont corrects.

On décrit maintenant les étapes du protocole TESLA de base dans cet ordre : établissement de l'envoyeur, amorçage du receveur, transmission par l'envoyeur des messages en diffusion authentifiés et authentification par le receveur des messages en diffusion.

3.2 Établissement de l'envoyeur

L'envoyeur divise le temps en intervalles uniformes de durée T_int. L'envoyeur alloue une clé de la chaîne unidirectionnelle à chaque intervalle de temps de la séquence.

L'envoyeur détermine la longueur N de la chaîne unidirectionnelle K_0, K_1, ..., K_N, et cette longueur limite la durée maximum de transmission avant qu'une nouvelle chaîne unidirectionnelle doive être créée. L'envoyeur prend une valeur aléatoire pour K_N. En utilisant une fonction pseudo aléatoire (PRF), f, l'envoyeur construit la fonction unidirectionnelle F : F(k) = f_k(0). Le reste de la chaîne est calculé par récurrence en utilisant K_i = F(K_{i+1}). Noter que cela nous donne K_i = F^{N-i}(K_N), de sorte que le receveur peut calculer toute valeur de la chaîne de clés à partir de K_N, même si il n'a pas de valeurs intermédiaires. La clé K_i va être utilisée pour authentifier les paquets envoyés dans l'intervalle de temps i.

Jakobsson [20] et Coppersmith et Jakobsson [21] présentent un mécanisme de mémorisation et de calcul efficace pour les chaînes unidirectionnelles. Pour une chaîne de longueur N, la mémorisation des environ log(N) éléments, et les frais généraux de calcul pour reconstruire chaque élément est aussi d'environ log(N).

L'envoyeur détermine la durée d'un intervalle de temps, T_int, et le délai de divulgation de clé, d. (T_int est mesuré en unités de temps, disons en millisecondes, et d est mesuré en nombre d'intervalles de temps. C'est-à-dire, une clé qui est utilisée pour l'intervalle de temps i va être divulguée dans l'intervalle de temps i+d). On souligne que le schéma reste sûr pour toutes valeurs de T_int et d>0. Il reste que le choix correct de T_int et d est crucial pour l'utilisation de ce schéma. Le choix est influencé par le délai estimé du réseau, la longueur de la transmission, et le délai tolérable chez le receveur. Un T_int trop court va causer l'expiration prématurée des clés. Un T_int trop long va causer un délai excessif d'authentification de certains paquets (ceux qui ont été envoyés au début d'une période). Un délai d trop court va faire que trop de paquets vont être invérifiables par le receveur. Un délai d trop long causera un délai excessif de l'authentification.

L'envoyeur estime une limite supérieure raisonnable du délai du réseau entre l'envoyeur et tout receveur comme m

millisecondes. Cela inclut tout délai attendu dans la pile (voir la Section 4, sur le placement de couche). Si l'envoyeur s'attend à envoyer un paquet toutes les n millisecondes, alors une valeur raisonnable pour T_int est max(n,m). Sur la base de T_int, une approximation pour déterminer le délai de divulgation de clé, d, est donnée au paragraphe 3.6.

La valeur ci-dessus pour T_int n'est une limite ni supérieure ni inférieure ; c'est simplement la valeur qui réduit le traitement de changement de clé au minimum sans causer de délai d'authentification supérieur à ce qui est nécessaire. Si l'application peut tolérer un délai d'authentification plus élevé, alors T_int peut être augmenté de façon appropriée. Aussi, si m (ou n) augmente durant la session, peut-être à cause d'encombrement ou d'une adhésion tardive sur un chemin déjà très encombré, T_int n'a pas besoin d'être révisé.

Finalement, l'envoyeur doit permettre à chaque receveur de synchroniser son heure avec lui. On verra les détails de la façon dont cela peut être fait au paragraphe 3.3.1. (On souligne que l'estimation du délai du réseau est une tâche distincte de celle de la synchronisation entre l'envoyeur et les receveurs.)

3.3 Receveurs de l'amorçage

Avant qu'un receveur puisse authentifier les messages avec TESLA, il doit avoir :

- Une limite supérieure, D_t, au décalage de sa propre horloge par rapport à l'horloge de l'envoyeur. (C'est-à-dire que si l'heure locale est t, l'heure actuelle chez l'envoyeur est au plus t+D_t.).

- Une clé authentifiée de la chaîne de clés unidirectionnelle. (Normalement, ce sera la dernière clé de la chaîne ; c'est-à-dire, K_0. Cette clé va être signée par l'envoyeur, et tous les receveurs vont vérifier la signature avec la clé publique du signataire.)

- Le programme de divulgation des clés suivantes :
  - T_int, la durée de l'intervalle.
  - T_0, l'heure de début de l'intervalle 0.
  - N, la longueur de la chaîne de clés unidirectionnelle.
  - d, le délai d de divulgation de clé (en nombre d'intervalles).

Le receveur peut effectuer la synchronisation et obtenir les paramètres TESLA authentifiés dans un échange de messages à deux tours, comme décrit ci-dessous. On souligne encore que la synchronisation peut être effectuée au titre du protocole d'enregistrement entre tout receveur (incluant les adhérents tardifs) et l'envoyeur, ou entre tout receveur et un contrôleur de groupe.

3.3.1 Synchronisation

Diverses approches existent pour la synchronisation [15], [16], [17], 18]. TESLA exige seulement que le receveur connaisse la limite supérieure du délai de son horloge locale par rapport à l'horloge de l'envoyeur, de sorte qu'un algorithme simple est suffisant. TESLA peut être utilisé avec une synchronisation directe, indirecte, et retardée qui sont trois options par défaut. La méthode spécifique de synchronisation fera partie de chaque instanciation de TESLA.

Pour être complet, on esquisse une méthode simple pour la synchronisation directe entre l'envoyeur et un receveur :
- Le receveur envoie un message (sync t_r) à l'envoyeur et enregistre son heure locale, t_r, au moment de l'envoi.
- À réception du message (sync t_r) l'envoyeur enregistre son heure locale, t_s, et envoie (synch, t_r,t_s) au receveur.
- À réception du (synch,t_r,t_s) le receveur règle D_t = t_s - t_r + S, où S est une limite estimée de la dérive d'horloge pour la durée de la session.

Note : en supposant que les messages sont authentiques (c'est-à-dire, que le message reçu par le receveur a bien été envoyé par l'envoyeur) et en supposant que la dérive d'horloge est au plus de S, alors en tout point de la session T_s < T_r + D_t, où T_s est l'heure actuelle chez l'envoyeur et T_r est l'heure actuelle chez le receveur. L'échange des messages sync doit être authentifié. Cela peut être fait de différentes façons ; par exemple, avec un protocole NTP sûr ou en conjonction avec un protocole d'établissement de session.

Pour la synchronisation indirecte (par exemple, la synchronisation via un contrôleur de groupe) l'envoyeur et le contrôleur s'engagent dans un protocole pour trouver la valeur D^0_t entre eux. Ensuite, chaque receveur, R, interagit avec le contrôleur de groupe (disons, quand ils s'enregistrent dans le groupe) et trouve la valeur D^R_t entre le contrôleur de groupe et R. La valeur globale de D_t au sein de R est réglée à la somme D_t = D^R_t + D^0_t.

3.4 Diffusion des messages authentifiés

Chaque clé de la chaîne de clés unidirectionnelle correspond à un intervalle de temps. Chaque fois qu'un envoyeur diffuse un message, il ajoute un MAC au message, en utilisant la clé qui correspond à l'intervalle de temps actuel. La clé reste secrète pendant les d-1 prochains intervalles, de sorte que les messages qu'un envoyeur diffuse dans l'intervalle j divulguent effectivement la clé K_j-d. On appelle d le délai de divulgation de clé.

On ne veut pas utiliser la même clé plusieurs fois dans des opérations cryptographiques différentes ; c'est-à-dire que utiliser la clé K_j pour déduire la clé précédente de la chaîne de clés unidirectionnelle K_{j-1}, et utiliser la même clé K_j comme clé pour calculer les MAC dans l'intervalle de temps j peut potentiellement conduire à une faiblesse cryptographique. En utilisant une fonction pseudo aléatoire (PRF), f', on construit la fonction unidirectionnelle F' : F'(k) = f'_k(1). On utilise F' pour déduire la clé pour calculer le MAC des messages dans chaque intervalle. L'envoyeur déduit la clé de MAC comme suit : K'_i = F'(K_i). La Figure 1 décrit la construction de la chaîne de clés unidirectionnelle et la déduction de la clé de MAC. Pour diffuser le message M_j dans l'intervalle i, l'envoyeur construit le paquet

P_j = {M_j || i || MAC(K'_i,M_j) || K_{i-d}} où || note l'enchaînement.

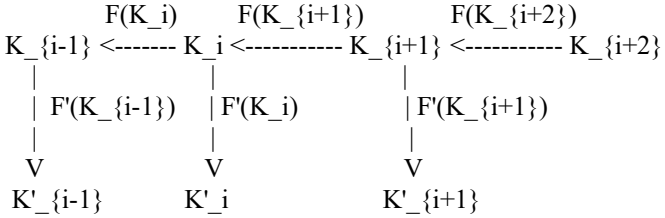

Figure 1 : Au sommet de la figure, on voit la chaîne de clés unidirectionnelle (déduite en utilisant la fonction unidirectionnelle F), et les clé de MAC déduites (en utilisant la fonction unidirectionnelle F').

3.5 Authentification chez le receveur

Une fois qu'un envoyeur a divulgué une clé, on doit supposer que toutes les parties peuvent avoir accès à cette clé. Un adversaire pourrait créer un message bogué et falsifier un MAC en utilisant la clé divulguée. Donc, chaque fois qu'un paquet arrive, le receveur doit vérifier que le MAC se fonde sur une clé sûre ; une clé sûre est celle qui est encore secrète (connue du seul envoyeur). On définit un paquet ou message sûr comme celui dont le MAC est calculé avec une clé sûre.

Si il est prouvé qu'un paquet est sûr, il va être mis en mémoire tampon, pour n'être délivré que quand sa propre clé, divulguée dans un paquet ultérieur, prouve son authenticité. Bien qu'un paquet nouvellement arrivé ne puisse être immédiatement authentifié, il peut divulguer une nouvelle clé afin que les paquets antérieurs, mis en mémoire tampon, puissent être authentifiés. Toute nouvelle clé divulguée doit être vérifiée pour déterminer si elle est authentique ; ensuite, on peut procéder à l'authentification des paquets qui ont été en attente dans la mémoire tampon.

On décrit ensuite plus en détails l'authentification TESLA chez le receveur, avec la liste des étapes dans l'ordre exact dans lequel elles devraient être suivies :

1. Vérification que le paquet est sûr : quand le receveur reçoit le paquet P_j, qui porte un indice d'intervalle i, et une clé divulguée K_{i-d}, il enregistre d'abord l'heure locale T à laquelle le paquet est arrivé. Le receveur calcule alors une limite supérieurs t_j sur l'horloge de l'envoyeur à laquelle le paquet est arrivé : t_j = T + D_t. Pour vérifier si le paquet est sûr, le receveur calcule alors le plus grand intervalle x dans lequel l'envoyeur pourrait être ; à savoir x = minimum de ((t_j - T_0) / T_int). Le receveur vérifie que x < i + d (où i est l'indice de l'intervalle) ce qui implique que l'envoyeur n'est pas encore dans l'intervalle durant lequel il divulgue la clé K_i.
   Même si le paquet est sûr, le receveur ne peut pas encore vérifier l'authenticité de ce paquet envoyé dans l'intervalle i sans la clé K_i, qui va être divulguée plus tard. À la place, il ajoute le triplet (i, M_j, MAC(K'_i, M_j)) à une mémoire tampon et en vérifie l'authenticité après avoir pris connaissance de K'_i.
   Si le paquet n'est pas sûr, le receveur considère alors que le paquet n'est pas authentifié. Il devrait éliminer les paquets non sûrs, mais, à ses propres risques, il peut choisir de les utiliser non vérifiés.

2. Vérification d'indice de nouvelle clé : Ensuite, le receveur vérifie si une clé K_v a déjà été divulguée avec le même indice v que la clé divulguée courante K_{i-d}, ou avec une plus ancienne ; c'est-à-dire, avec v ≤ i-d.

3. Essai de vérification de clé : si l'indice de la clé divulguée est nouveau, le receveur vérifie la légitimité de K_{i-d} en vérifiant, pour une clé divulguée antérieurement K_v (v<i-d) que K_v = F^{i-d-v}(K_{i-d}).

Si la vérification de clé échoue, le nouveau paquet arrivé P_j devrait être éliminé.

4. Essais de vérification de message : Si la clé divulguée est légitime, le receveur vérifie alors l'authenticité de tous les paquets sûrs antérieurs mis en mémoire tampon dans l'intervalle i-d. Pour authentifier un des paquets P_h mis en mémoire tampon contenant le message M_h protégé par un MAC qui a utilisé l'indice de clé i-d, le receveur va calculer K'_{i-d} = F'(K_{i-d}) à partir duquel il peut calculer MAC(K'_{i-d}, M_h).

Si ce MAC est égal au MAC mémorisé dans la mémoire tampon, le paquet est authentifié et peut être libéré de la mémoire tampon. Si les MAC ne s'accordent pas, le paquet P_h de la mémoire tampon devrait être éliminé.

Le receveur continue de vérifier et libérer (ou non) tous les paquets restants dans la mémoire tampon qui dépendent de la clé K_{i-d} nouvellement divulguée.

En utilisant une clé divulguée, on peut calculer toutes les clés précédemment divulguées, de sorte que même si des paquets sont perdus, on sera quand même capable de vérifier les paquets sûr mis en mémoire tampon à partir des intervalles de temps antérieurs. Donc, si i-d- v>1, le receveur peut aussi vérifier l'authenticité des paquets mémorisés des intervalles v+1 ... i-d-1.

3.6 Détermination du délai de divulgation de clé

Un paramètre important de TESLA est le délai de divulgation de clé d. Bien que le choix du délai de divulgation n'affecte pas la sécurité du système, il est un facteur important de performances. Un court délai de divulgation va causer la perte de la propriété de sûreté des paquets, de sorte que les receveurs ne vont pas être capables de les authentifier ; mais un long délai de divulgation conduit à un long délai d'authentification pour les receveurs. On recommande de déterminer le délai de divulgation de la manière suivante : dans la synchronisation directe, soit le RTT, 2m, une limite supérieure raisonnable au délai d'aller-retour entre l'envoyeur et tout receveur incluant le pire cas de délai dû à l'encombrement et le pire cas de délai de mise en mémoire tapon dans les piles des hôtes. Choisir ensuite d = max(2m / T_int) + 1. Noter que l'arrondissement du quotient à l'entier supérieur assure que $d \leq 2$. Noter aussi qu'un délai de divulgation d'un seul intervalle de temps (d=1) ne fonctionne pas. Considérons les paquets envoyés proches de la limite de l'intervalle de temps : après le délai de propagation du réseau et l'erreur de synchronisation de l'heure du receveur, un receveur ne sera pas capable d'authentifier le paquet, parce que l'envoyeur va déjà être dans le prochain intervalle de temps quand il divulguera la clé correspondante.

Mesurer le délai pour chaque receveur avant de déterminer m ne va pas prédire de façon adéquate la limite supérieure du délai pour les adhérents tardifs, ou lorsque le délai dû à l'encombrement augmente plus tard dans la session. Il peut être adéquat d'utiliser une estimation fondée sur un historique à code fixe des pires cas de délai (par exemple, les délais d'aller-retour vers tout hôte sur l'Internet intra planétaire excèdent rarement 500 ms si l'acheminement reste stable).

On souligne que la sécurité de TESLA ne repose sur aucune hypothèse sur le délai de propagation du réseau : si le délai est plus long que prévu, des paquets authentiques peuvent alors être considérés comme non authentifiés. Cependant, aucun paquet non authentique ne sera accepté comme authentique.

3.7 Protection contre le déni de service

Parce que l'authentification TESLA est retardée, les receveurs semblent vulnérables à des attaques d'inondation qui les amènent à mettre en mémoire tampon des quantités excessives de paquets, même quand il peut être prouvé qu'ils ne sont pas authentiques. Quand TESLA est déployé dans un environnement où existe une menace d'attaques d'inondation, le receveur peut prendre un certain nombre de précautions supplémentaires.

On fait d'abord la liste des simples précautions d'atténuation de DoS qui peuvent et devraient être prises par tout receveur indépendamment des autres, n'exigeant donc pas de changement du protocole ni du comportement de l'envoyeur. On spécifie précisément où ces étapes supplémentaires s'insèrent dans les étapes d'authentification de receveur déjà données au paragraphe 3.5.

- Vérification de la validité de session : avant la vérification que le paquet est sûr (étape 1) vérifier que les paquets arrivant ont une adresse IP de source et un numéro d'accès valides pour la session, qu'ils ne répètent pas un message déjà reçu dans la session, et qu'ils ne sont pas significativement plus gros que les tailles de paquet attendues dans la session.

- Vérification d'un déclassement raisonnable : avant l'essai de vérification de clé (étape 3) vérifier si l'indice de clé divulguée i-d du paquet arrivant est dans le g du plus grand indice de clé divulguée antérieur v ; donc, par exemple, i-d-

v ≤ g. g règle le seuil au delà duquel un indice de clé déclassé est suppose être malveillant plutôt que juste déclassé. Sans cet essai, un attaquant pourrait exploiter l'essai itéré de l'étape 3 pour faire que les receveurs consomment du temps de CPU extraordinaire à vérifier le long de la chaîne hachée ce qui paraît être un paquet extrêmement déclassé.

Chaque receveur peut adapter indépendamment g pour empêcher des conditions d'attaque ; par exemple, en utilisant l'algorithme suivant. Initialement, g devrait être réglé à g_max (disons, 16). Mais chaque fois d'un paquet arrivant échoue à l'essai de déclassement raisonnable ci-dessus ou à l'essai de vérification de clé (étape 3) g devrait être diminué à g_min (> 0 et normalement 1). À chaque vérification réussie de clé (étape 3) g devrait être incrémenté de 1 sauf si il est déjà à g_max. Ces précautions vont garantir que les paquets d'une attaque soutenue ne peuvent pas causer l'exécution par le receveur de plus d'une moyenne de g_min hachages à chaque fois, sauf si ils sont couplés avec des paquets authentiques. Dans ce dernier cas, les attaques sont limitées à g_max/(g_max-g_min) hachages pour chaque paquet authentique.

Pour le choix de g_max et g_min, noter qu'ils limitent l'écart moyen d'une séquence de paquets à g.max(n,m)/n paquets (voir au paragraphe 3.2 les définitions de n et m). Donc avec g = 1, un RTT de m = 100 ms, et une période inter-paquets de n = 4 ms, le reclassement va être limité à des trous de 25 paquets en moyenne. Des trous naturels plus gros s'il s'en produit devraient être inscrits comme si ils étaient des pertes.

Une plus forte protection contre le DoS exige que envoyeurs et receveurs instituent des contraintes supplémentaires sur le protocole. On mentionne plus loin trois extensions possibles au TESLA de base ; la première ajoute l'authentification de groupe, la seconde ne réutilise pas les clés durant un intervalle de temps, et la troisième déplace la mise en mémoire tampon chez l'envoyeur.

Il est important de comprendre l'applicabilité de chaque schéma, car les deux premiers schémas utilisent un peu plus de ressources (mais limitées) afin d'empêcher les attaquants de consommer des ressources illimitées. Ajouter l'authentification de groupe exige des frais généraux plus importants par paquet. Ne jamais réutiliser une clé exige que les deux extrémités traitent deux hachages par paquet (plutôt que par intervalle de temps) et l'envoyeur doit mémoriser ou re-générer une chaîne de hachage plus longue. Les mérites de chaque schéma, résumés après chaque description ci-dessous, doivent être évalués par rapport à leurs coûts additionnels.

3.7.1 Authentification de groupe supplémentaire

Ce schéma implique simplement l'ajout d'un MAC de groupe à chaque paquet. C'est-à-dire, une clé partagée K_g commune à tout le groupe est communiquée comme étape supplémentaire durant l'amorçage du receveur (paragraphe 3.3). Ensuite, durant la diffusion du message M_j (paragraphe 3.4) l'envoyeur calcule le MAC de groupe de chaque paquet MAC(K_g, P_j) qu'il ajoute à l'en-tête de paquet. Noter que le MAC de groupe couvre tout le paquet P_j ; c'est-à-dire l'enchaînement du message M_j et du matériel supplémentaire d'authentification TESLA, en utilisant la formule du paragraphe 3.4.

Immédiatement à l'arrivée du paquet, chaque receveur peut vérifier que chaque paquet vient d'un membre du groupe, en recalculant et comparant le MAC de groupe.

Noter que l'authentification de source de TESLA n'est nécessaire que quand d'autres membres du groupe ne peuvent pas être estimés pouvoir s'empêcher de tromper la source ; autrement, une plus simple authentification de groupe va être suffisante. Donc, l'authentification de groupe supplémentaire n'a de sens que dans des scénarios où les autres membres du groupe sont estimés s'interdire d'inonder le groupe, mais où il il y en a quand même qui vont essayer de mystifier la source.

3.7.2 Non réutilisation de clés

Dans TESLA comme décrit jusqu'ici, chaque clé de MAC a été utilisée de façon répétée pour tous les paquets envoyés dans un intervalle de temps. Si l'envoyeur se trouvait plutôt garantir de ne jamais utiliser plus d'une fois une clé de MAC, chaque clé divulguée pourrait assurer d'autres tâches en plus de l'authentification d'un paquet mis antérieurement en mémoire tampon. Chaque clé montrerait aussi immédiatement à chaque receveur que l'envoyeur de chaque paquet arrivant connaît la prochaine clé en remontant le long de la chaîne de hachage, qui n'est maintenant divulguée qu'une seule fois, comme dans S/KEY [22]. Donc une stratégie raisonnable de receveur serait d'éliminer tout les paquets arrivants qui divulguent une clé déjà vue. Le taux de remplissage de la mémoire tampon du receveur serait alors rythmé par chaque paquet révélé par le véritable envoyeur, empêchant les attaques d'inondation de la mémoire.

Un attaquant qui contrôle un élément de réseau ou un réseau de contournement plus rapide pourrait intercepter les messages et les remplacer par des messages différents mais avec les mêmes clés. Cependant, tant que les paquets sont seulement en mémoire tampon si ils dépassent aussi le délai de vérification de sûreté, ces paquets bogués vont échouer à la vérification

de TESLA après le délai de divulgation. On peut admettre que les receveurs pourraient être amenés à éliminer des messages authentiques qui ont été remplacés par d'autres altérés. Mais il est difficile de rattraper des messages sans compromettre un élément de réseau, et tout attaquant qui peut compromettre un élément de réseau peut éliminer de toutes façons des messages authentiques. On va maintenant décrire ce schéma plus en détails.

Pour l'envoyeur, le schéma n'est pratiquement pas différent de celui du TESLA de base. Il utilise simplement une durée d'intervalle assez courte pour assurer d'une nouvelle clé le long de la chaîne de hachage pour chaque paquet. Donc la règle d'approximation du paragraphe 3.2 pour un intervalle efficace de changement de clé T_int ne s'applique plus. À la place, T_int est simplement n, l'heure inter arrivée entre paquets en millisecondes. La règle d'approximation pour calculer d, le délai de divulgation de clé, reste celle donnée au paragraphe 3.6.

Si le taux de paquets va probablement varier, n devrait être pris comme le minimum d'heure inter départs entre deux paquets. (En fait, n n'a pas besoin d'être aussi strict ; il peut être le minimum moyen de l'heure de départ inter paquets sur toute salve de paquets attendue sur la session.)

Noter que si le taux de paquets ralentit, chaque fois qu'aucun paquet n'est envoyé dans un intervalle de changement de clé, l'indice de clé doit s'incrémenter le long de la chaîne de hachage de un pour chaque intervalle manqué. (Durant une salve, si la définition moins stricte de n ci-dessus a été utilisée, les paquets peuvent devoir partir avant leur intervalle de changement de clé. L'envoyeur peut en toute sécurité continuer de changer la clé pour chaque paquet, en utilisant des clés des futurs intervalles de clés, parce que si n a été choisi comme défini ci-dessus, de telles salves ne vont jamais durer assez longtemps pour causer la divulgation de la clé associée dans une période plus courte que le délai de divulgation ultérieur.)

Pour être absolument clair, les garanties précises que l'envoyeur s'en tient aux lignes directrices ci-dessus sont :
- ne pas réutiliser une clé de MAC,
- ne pas utiliser une clé de MAC K_i après son intervalle de temps i, et
- ne pas divulguer la clé K_i plus tôt que le délai de divulgation d * T_int après le paquet qu'il protège.

L'établissement de l'envoyeur, l'amorçage des receveurs, et les messages authentifiés en diffusion sont par ailleurs tous identiques aux descriptions, respectivement, des paragraphes 3.2, 3.3, et 3.4. Cependant, l'étape suivante doit être ajoutée aux étapes d'authentification du receveur du paragraphe 3.5 :

- Après l'étape 2, si un paquet qui arrive porte un indice de clé i-d qui a déjà été reçu, il ne devrait pas être mis en mémoire tampon.

Ce schéma simple devrait suffire contre les attaques de DoS, si ce n'était le fait que parfois un réseau déclasse des paquets sans avoir été compromis. Même sans le contrôle d'un élément de réseau, un attaquant peut exploiter de façon opportuniste de telles ouvertures pour tromper un receveur et l'amener à mettre en mémoire tampon un paquet bogué et à en éliminer un authentique un peu plus tard. Un receveur peut choisir de renoncer à une antémémoire de taille fixe et peut la gérer de façon à minimiser les chances d'élimination d'un paquet authentique. Cependant, comme de telles vulnérabilités sont rares et imprévisibles, il est plus simple de compter ces événements comme s'ajoutant au taux de perte du réseau. Comme toujours l'authentification TESLA ne va pas couvrir de paquets bogués après le délai de divulgation.

Pour résumer, éviter de réutiliser les clés a les propriétés suivantes, même dans des conditions extrêmes d'attaques d'inondation :
- Après l'authentification TESLA retardée, les paquets qui arrivent dans le délai de divulgation vont toujours être identifiés comme authentiques si ils le sont et comme non authentiques si ils ne sont pas authentiques.
- Le taux de remplissage de la mémoire tampon du receveur est rythmé par chaque paquet révélé par l'envoyeur authentique, empêchant les attaques d'inondation de la mémoire.
- Un attaquant qui contrôle un élément de réseau peut causer le taux de pertes de son choix (mais c'est vrai de toutes façons).
- Si l'attaquant n'a pas le contrôle d'un élément de réseau, le taux de perte effectif est limité par la somme du taux de pertes actuel du réseau et de son taux de reclassement.

3.7.3 Mise en mémoire tampon chez l'envoyeur

La mise en mémoire tampon des paquets peut être déplacée chez l'envoyeur ; les receveurs peuvent alors authentifier les paquets immédiatement à réception. Cette méthode est décrite dans [14].

3.8 Quelques extensions

On mentionnera deux extensions notables au schéma de base de TESLA. Une première extension permet d'avoir plusieurs chaînes d'authentification TESLA pour un seul flux, où chaque chaîne utilise un délai différent pour la divulgation des clés. Cette extension est normalement utilisée pour traiter les délais de réseaux hétérogènes au sein d'une seule transmission en diffusion groupée.

Une seconde extension permet d'avoir la plus grande partie des paquets mis en mémoire tampon du côté de l'envoyeur (plutôt que du côté du receveur). Ces deux extensions sont décrites dans [14].

L'exigence de TESLA qu'une clé soit reçu dans un paquet ultérieur pour l'authentification empêche un receveur d'authentifier la dernière partie d'un message. Donc, pour permettre l'authentification de la dernière partie d'un message ou du dernier message avant une suspension de transmission, l'envoyeur doit envoyer un message vide avec la clé.

4. Placement de couche

L'authentification TESLA peut être effectuée à toute couche de la pile de réseautage. Trois endroits naturels sont la couche réseau, la couche transport, ou la couche application. Voici quelques considérations concernant le choix de la couche:

- Effectuer TESLA dans la couche réseau présente l'avantage que les couches transport ou application ne reçoivent que des données authentifiées, ce qui peut aider à la fiabilité du protocole et atténuer les attaques de déni de service. (Bien sûr, des outils fiables de diffusion groupée fondés sur la correction d'erreur directe sont très susceptible de se prêter au déni de service à cause de paquets bogués.)

- Effectuer TESLA dans la couche transport ou application présente l'avantage que la couche réseau reste inchangée, mais cela présente l'inconvénient potentiel que les paquets ne sont obtenus par la couche application qu'après avoir été traités par la transport couche. Par conséquent, si la mise en mémoire tampon est utilisée dans le transport, cela peut alors introduire des délais supplémentaires imprévisibles en plus des délais inévitables du réseau.

- Noter que comme TESLA s'appuie sur le rythme des paquets, déployer TESLA par dessus un protocole ou couche qui met de façon agressive les paquets en mémoire tampon et cache la vraie heure d'arrivée du paquet va significativement réduire les performances de TESLA.

5. Considérations sur la sécurité

Voir dans les publications académiques sur TESLA [7], [13], [19] plusieurs analyses de la sécurité. Concernant la sécurité des mises en œuvre, le point de loin le plus délicat est la vérification des conditions de rythme. On devrait prendre le plus grand soin à s'assurer que (a) la valeur limite D_t sur le décalage d'horloge est calculé à l'établissement de la session en accord avec la spécification et que (b) le receveur enregistre l'heure d'arrivée du paquet aussitôt que possible après l'arrivée du paquet, et calcule correctement la condition de sûreté.

On devrait noter qu'un changement de programmation de la divulgation de la clé pour un flux de messages ne devrait jamais être déclaré dans le flux de messages lui-même. Cela introduirait une vulnérabilité, parce que un receveur qui ne recevrait pas la notification du changement en resterait à l'ancienne programmation de divulgation de clés.

Finalement, en commun avec tous les schémas d'authentification, si la vérification est située séparément de l'application ultime de destination (par exemple, un point d'extrémité de tunnel IPsec) un canal de confiance doit être présent entre la vérification et l'application. Par exemple, l'interface entre le vérificateur et l'application pourrait simplement supposer que les paquets reçus par l'application doivent avoir été vérifiés par le vérificateur (parce que autrement, ils auraient été éliminés). L'application est alors vulnérable à la réception de paquets qui se sont arrangés pour contourner le vérificateur.

6. Remerciements

Nous tenons à remercier de leurs retours et leur soutien Mike Luby, Mark Baugher, Mats Naslund, Dave McGrew, Ross Finlayson, Sylvie Laniepce, Lakshminath Dondeti, Russ Housley, et les réviseurs de l'IESG.

7. Références pour information

[1] [RFC2246] T. Dierks et C. Allen, "Protocole TLS version 1.0", janvier 1999. *(P.S. ; MàJ par* RFC7919)

[2] IPsec, "IP Security Protocol, IETF working group" http://www.ietf.org/html.charters/OLD/ipsec-charter.html

[3] D. Boneh, G. Durfee, and M. Franklin, "Lower bounds for multicast message authentication," dans Advances in Cryptology -- EUROCRYPT 2001 (B. Pfitzmann, ed.), Vol. 2045 de Lecture Notes in Computer Science, (Innsbruck, Austria), p. 434-450, Springer-Verlag, Berlin Germany, 2001.

[4] R. Gennaro and P. Rohatgi, "How to Sign Digital Streams", tech. rep., IBM T.J.Watson Research Center, 1997.

[5] P. Rohatgi, "A compact and fast hybrid signature scheme for multicast packet authentication", 6th ACM Conference on Computer and Communications Security , novembre 1999.

[6] C. K. Wong and S. S. Lam, "Digital signatures for flows and multicasts," dans Proc. IEEE ICNP `98, 1998.

[7] A. Perrig, R. Canetti, J. Tygar, and D. X. Song, "Efficient authentification and signing of multicast streams over lossy channels", IEEE Symposium on Security and Privacy, mai 2000.

[8] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions", Infocom '99, 1999.

[9] S. Cheung, "An efficient message authentication scheme for link state routing", 13th Annual Computer Security Applications Conference, 1997.

[10] F. Bergadano, D. Cavagnino, and B. Crispo, "Chained stream authentication," dans Selected Areas in Cryptography 2000, (Waterloo, Canada), août 2000. Une conférence décrivant ce schéma a été donnée à IBM Watson en août 1998.

[11] F. Bergadano, D. Cavalino, and B. Crispo, "Individual single source authentication on the mbone", ICME 2000, August 2000. Une conférence contenant ce travail été donnée à IBM Watson, août 1998.

[12] A. Perrig and D. Tygar, Secure Broadcast Communication in Wired and Wireless Networks Kluwer Academic Publishers, octobre 2002. ISBN 0792376501.

[13] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The tesla broadcast authentication protocol," RSA CryptoBytes, Volume 5, No. 2 Summer/Fall 2002.

[14] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast", Network and Distributed System Security Symposium, NDSS '01, p. 35-46, février 2001.

[15] [RFC1305] D. Mills, "Protocole de l'heure du réseau, version 3, spécification, mise en œuvre et analyse", STD 12, mars 1992. *(Remplacée par* RFC5905)

[16] B. Simons, J. Lundelius-Welch, and N. Lynch, "An overview of clock synchronization", Fault-Tolerant Distributed Computing (B. Simons and A. Spector, eds.), No. 448 in LNCS, p. 84-96, Springer-Verlag, Berlin Germany, 1990.

[17] D. Mills, "Improved algorithms for synchronizing computer network clocks", Proceedings of the conference on Communications architectures, protocols and applications, SIGCOMM 94, (London, England), p. 317-327, 1994.

[18] L. Lamport and P. Melliar-Smith, "Synchronizing clocks in the presence of faults", J. ACM, Volume 32, No. 1, p. 52-78, 1985.

[19] P. Broadfoot and G. Lowe, "Analysing a Stream authentication Protocol using Model Checking", Proceedings of the 7th European Symposium on Research in Computer Security (ESORICS), 2002.

[20] M. Jakobsson, "Fractal hash sequence representation and traversal", Cryptology ePrint Archive, http://eprint.iacr.org/2002/001/ , janvier 2002.

[21] D. Coppersmith and M. Jakobsson, "Almost optimal hash sequence traversal", Proceedings of the Sixth International Financial Cryptography Conference (FC '02), mars 2002.

[22] [RFC1760] N. Haller, "Système S/KEY de mot de passe à utilisation unique", février 1995. *(Information)*

Adresse des auteurs

Adrian Perrig
ECE Department
Carnegie Mellon University
Pittsburgh, PA 15218
US
mél : perrig@cmu.edu

Ran Canetti
IBM Research
30 Saw Mill River Rd
Hawthorne, NY 10532
US
mél : canetti@watson.ibm.com

Dawn Song
ECE Department
Carnegie Mellon University
Pittsburgh, PA 15218
US
mél : dawnsong@cmu.edu

J. D. Tygar
UC Berkeley - EECS & SIMS
102 South Hall 4600
Berkeley, CA 94720-4600
US
mél : doug.tygar@gmail.com

Bob Briscoe
BT Research
B54/77, BT Labs
Martlesham Heath
Ipswich, IP5 3RE
UK
mél : bob.briscoe@bt.com

Déclaration de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est actuellement fourni par la Internet Society.