Introduction à la sécurité
Protocôles de sécurité

Catalin Dima

Combinaisons de solutions cryptographiques

- Les algorithmes cryptographiques assurent une bonne qualité de la confidentialité et/ou authenticité d'une transmission, *une fois celle-ci établie !*
- Problème : établir une connexion sécurisée !
  - "IP spoofing", "mascarade", détournement de session, attaque de rejeu...
- *Protocôle de sécurité* : séquence d'actions impliquant deux ou plusieurs agents, destinée à accomplir une tâche (connexion sécurisée, échange de messages, etc.).
  - Un seul agent ne suffit pas pour avoir un protocôle...
- Les algorithmes cryptographiques peuvent être très performants, mais si le protocôle est mal conçu la sécurité des transmissions est en danger.
- On a vu un premier protocôle sur le transparent "Rassembler..."

Caractéristiques des protocôles

- But : communication sûre entre agents (utilisateurs, ordinateurs, processus...)
- Implémentent tout ou une partie des types de service de sécurité :
  - Secretisation/confidentialité.
  - Authentification.
  - Distribution des clés = envoi de clés pour la communication privée ou pour l'authentification.
  - Intégrité des messages.
  - Non-répudiation – impossibilité de nier être la source d'un message/transaction après que celle-ci a bien été intégrée au protocôle.
  - Possibilité de comptabiliser les transactions.
  - Anonymité.
- Programmes souvent de taille petite (souvent moins de 10 messages).
- Basés sur les algorithmes de cryptage, mais indépendents du choix d'algorithme (mais pas du *type* de cryptage – à clés privées ou publiques).
- Exemples : Kerberos, SSL, SSH,...

Règles de définition de protocôles

1. Tous les participants (agents) dans les protocôles doivent connaître le protocôle en avance (type de protocôle, séquence des pas à entreprendre...)
2. Le protocôle doit être défini de manière non-ambigue et complète.
3. Les agents coopèrent en s'échangeant des messages.
4. Les agents n'ont pas d'autre canal de communication que celui sur lequel le protocôle est censé à être deployé.
5. Les agents utilisent uniquement les messages reçus pour prendre des décisions sur le comportement à suivre.

Modélisation

- Participants :
  1. Agents : Alice, Bob,...
  2. Serveur(s) de confiance.
  3. Arbitres de confiance (en cas de conflit).
  4. Agents dishonnêts : Eve (oreille indiscrète, angl. *eavesdropper*), Mallory (attaquant active) – on va les noter I.
- Algorithmes cryptographiques parfaits
  - Les attaquants n'ont pas les moyens computationnels de casser les codes.
- *Session* : un déroulement complet du protocôle, impliquant tous les agents concernés.
  - Bcp. de protocôles permettent à chaque agent d'ouvrir des sessions *parallèles*.
- Modèle de base : algèbre de termes.
  - Chaque transmission de données est un "terme" construit à partir de composants primitifs (noms d'agents, messages non-chiffrés, clés...) en employant des opérations (cryptage, mais aussi juxtaposition de messages).

Composants et opérandes

- Roles : noms génériques des participants (Alice, Bob,...)
- Messages : M.
- Clés de cryptage : K, K_{Alice}, K_{AB}...
  - Algorithmes à clé publique : paire (K, K^{-1}).
  - Algorithmes symmétriques : $K = K^{-1}$.
- *Nonces* N : valeurs fraîches et imprédictibles.
  - Obtenues par des fonctions à sens unique,
  - *Étiquettes de temps.*
- Opérations de base :
  1. Juxtaposition de messages : M_1, M_2.
  2. Cryptage $\{M\}_K$.
- Décryptage – opération implicite :
  1. Si l'agent A possède la clé K^{-1}
     - sa clé privée
     - une clé symmétrique partagée avec un autre agent,
     - une clé de session reçue auparavant dans un autre message)
  2. Et reçoit un message crypté avec la clé correspondante de cryptage K,
  3. Alors il peut déchiffrer le message encapsule dans la transmission.

Modélisation

$$
\begin{aligned}
A \longrightarrow B &\quad : \quad \{N_A\}_{K_B}, \{N_A, A\}_{K_S} \\
B \longrightarrow S &\quad : \quad \{N_A, N_B\}_{K_S}, \{N_A, A\}_{K_S} \\
S \longrightarrow A &\quad : \quad \{N_A, K_{AB}\}_{K_A} \\
S \longrightarrow B &\quad : \quad \{N_B, K_{AB}\}_{K_B} \\
A \longrightarrow B &\quad : \quad \{M\}_{K_{AB}}
\end{aligned}
$$

- $A \longrightarrow B = A$ envoie à B le message...
- N_X : nonce fraîche créée par X (X peut être A ou B).
- K_X : clé publique de X (X peut être A, B, ou S).
- B peut décrypter le message qu'il a reçu car ce message est crypté par sa clé publique.
- ... mais il ne peut pas décrypter le message $\{N_A, A\}_{K_S}$.
- Donc il perçoit ce message comme une "boîte noire", et il le renvoie tel quel au S.
- K_{AB} : clé de session générée par S.
- M : message généré par A.

Exemples de protocôles

$$A \longrightarrow B \quad : \quad A, B$$
$$B \longrightarrow A \quad : \quad K_B$$
$$A \longrightarrow B \quad : \quad \{M\}_{K_B}$$

– Eventuellement assorti de l'authentification du message M, comme étant envoyé par A.
– Mais est-il correcte ?...

2e exemple : Needham-Schroeder à clés publiques

- Utilisé dans Kerberos.
- Agents : A, B et authorité de certification S.

$$
\begin{array}{lll}
M_1. & A \longrightarrow S : & A, B \\
M_2. & S \longrightarrow A : & \{K_B, B\}_{K_S^{-1}} \\
M_3. & A \longrightarrow B : & \{N_A, A\}_{K_B} \\
M_4 & B \longrightarrow S : & B, A \\
M_5 & S \longrightarrow B : & \{K_A, A\}_{K_S^{-1}} \\
M_6 & B \longrightarrow A : & \{N_A, N_B\}_{K_A} \\
M_7 & A \longrightarrow B : & \{N_B\}_{K_B}
\end{array}
$$

- M_1, M_2, M_4, M_5 : les deux agents recupèrent la clé publique de l'autre.
- M_3, M_6 et M_7 : A et B utilisent les clés obtenues pour communiquer les *nonces sécrètes* N_A et N_B.
- Nonces : défi-réponse.
- Seulement A et B peuvent obtenir N_A et N_B
- Ultérieurement, A et B peuvent utiliser N_A et/ou N_B pour *signer* des messages.

Needham-Schroeder simplifié : sans serveur

– Supposant que chacun possède déjà la clé publique de l'autre :

$$M_1. \quad A \longrightarrow B : \quad \{A, N_A\}_{K_B}$$
$$M_2. \quad B \longrightarrow A : \quad \{N_A, N_B\}_{K_A}$$
$$M_3. \quad A \longrightarrow B : \quad \{N_B\}_{K_B}$$

♦ Est-ce que les deux protocôles sont sûrs ?
  – On suppose des algorithmes de chiffrement "parfaits" !
  – L'attaquant ne peut se baser que sur ses propres déductions faites à partir des messages (chiffrés ou non) qu'il intercèpte.

Connaissance

- **Connaissance initiale** $Cn_0(A)$ pour l'agent A :
  - Ensemble d'items supposés connus par l'agent au début du protocôle.
  - Exemple : clé privée propre, clé publique d'un serveur.
- **Connaissance instantanée** $Cn_k(A)$ pour un agent A à un instant k : avant chaque échange de messages, l'émetteur devrait être en mésure de composer les messages dont il est la source
  - Soit parce qu'il possède tous les éléments constitutifs,
  - Soit parce qu'il possède des messages composés, qui sont pour lui des "boîtes noires".

$$M_{k+1} : \quad B \longrightarrow A : \quad Message$$

$$Cn_{k+1}(A) \quad = \quad Cn_k(A) \cup Decomp\Big(Cn_k(A) \cup Message)$$

- $Decomp$ = que peut X décrypter en plus, après reception du message ?
- Les nonces/timestamps apparaissant la première fois dans le message M_{k+1} font partie de sa connaissance $Cn_{k+1}(A)$.
- Construction, par l'émetteur, du message dans le k-ième pas :
  - Par rapport à la connaissance instantanée, est-ce que $Message$ est correctement construit ?

Connaissance (2)

Exemple pour Needham-Schroeder sans serveur :

$$M_1. \quad A \longrightarrow B : \quad \{A, N_A\}_{K_B}$$
$$M_2. \quad B \longrightarrow A : \quad \{N_A, N_B\}_{K_A}$$
$$M_3. \quad A \longrightarrow B : \quad \{N_B\}_{K_B}$$

- Connaissance initiale A : $Cn_0(A) = \{A, B, N_A, K_A, K_A^{-1}, K_B\}$
- Connaissance initiale B : $Cn_0(B) = \{A, B, N_B, K_B, K_B^{-1}, K_A\}$
- Après le premier pas : $Cn_1(B) = \{A, B, N_B, N_A, K_B, K_B^{-1}, K_A\}$.
  - Car B possède K_B^{-1} et peut déchiffrer $\{A, N_A\}_{K_B}$.
- Après le deuxième pas : $Cn_2(A) = \{A, B, N_B, N_A, K_A, K_A^{-1}, K_B\}$.
- Le troisième pas sert à vérifier que A répond correctement !
  - Un message différent de $\{N_B\}_{K_B}$ serait réfusé, et la session serait invalidée !
  - Réponse au challenge du message 2 !

Protocôle correctement construit !

Constructions incorrectes

- Exemple de construction incorrecte :

$$A \longrightarrow B : \quad A, B, \{A, N_A\}_{K_S}$$

$$B \longrightarrow S : \quad \{A, N_A\}_{K_S}, \{B, N_A\}_{K_S}$$

  - B ne possède pas la clé privée de S !
- Voir aussi le 3e pas du protocôle NS.

- Modélisation assez grossière :
  - Comment sait B que c'est A qui le contacte pour initier une session ?
  - En général, on devrait avoir aussi en clair l'information sur la source et la destination de chaque message.
- Hypothèses implicites sur la mémoire des agents :
  - Chaque nonce générée dans une session sera utilisée dans la seule session, et sera oubliée lors du démarrage d'une autre session.

Propriétés des protocôles : confidentialité

- Quels sont les sécrets que le protocôle ne doit pas divulguer ?
- Évidemment pas les messages chiffrés !
  - Un protocôle peut être sûr par rapport à un certain message et pas par rapport à un autre.
  - Certainement les clés de session ne devraient pas être divulguées !
- Supposons un protocôle P, dans lequel les rôles sont $A_1, \ldots, A_k$.
  - Un item M est un secret pour un rôle A_i si l'agent jouant le rôle A_i ne peut pas déduire, à partir des messages qu'il reçoit, l'item M.
  - Mais pour les attaquants ?

Attaques et attaquants

Attaques actifs : modèle de *Dolev-Yao*.

- L'attaquant possède une information complète sur le protocôle de communication employé par les agents.
- L'attaquant contrôle les liens de communication entre les agents.
- L'attaquant possède une mémoire non-bornée.
- L'attaquant peut décrypter un message chiffré seulement s'il possède la bonne clé de décryptage – il l'a reçue ou décryptée dans un message précédent.
- L'attaquant peut composer des nouvelles transmissions avec les messages chiffrés qui ont transité sur le réseau, ou avec les messages qu'il est arrivé a déchiffrer.
  - Soit il dispose de tous les éléments atomiques dans les messages
  - Soit il dispose de messages non-atomiques, qui restent pour lui comme des "boîtes noires" !
- L'attaquant peut jouer un rôle d'agent légitime.
  - Les règles de construction de la connaissance de Mallory sont les mêmes que pour Alice/Bob/Serveur etc.

Relaxations du modèle Dolev-Yao :

- L'attaquant peut casser des chiffres symmétriques s'il dispose de suffisament de temps.
- L'attaquant peut ne pas forger n'importe quel message (divers restrictions de puissance).

Types d'attaque

- Attaque de *rejeu* : Mallory mémorise un message (une partie d'un message) et le réintroduit plus tard dans le protocôle (éventuellement en tant que partie d'un autre message).
- Attaque de type *personne au milieu* : sessions paralèles où Mallory peut jouer des rôles légitimes, mais combine ("forge") des nouveaux messages.
- Attaque de type *mascarade* : Mallory prétend être Alice, et Bob ne peut pas se rendre compte de sa tromperie.
  - Notation I_A : Mallory prétend être Alice dans le protocôle.
- Attaque de type *erreur de typage* : Mallory substitue une partie du message avec un message forgé, mais qui n'est pas du bon type (e.g. identité au lieu de clé).

Attaques

- *Attaque* = enchaînement de communications entre agents + Mallory, impliquant éventuellement des sessions parallèles, (y compris des sessions dans lesquelles Mallory joue un rôle légitime), sessions dans lesquelles chaque agent ne détecte aucune anomalie.
  - Communications et des connaissances respectant les règles Dolev-Yao.
  - Anomalie : non-correspondance entre messages reçus à un instant et messages précédemment échangés dans la même session.
  - Non-exemple :

Protocôle			(Non-)attaque		
$A \longrightarrow B$	:	$\{A, B, N_A\}_{K_{AB}}$	$A \longrightarrow I_B$	:	$\{A, B, N_A\}_{K_{AB}}$
$B \longrightarrow A$	:	$\{N_A, N_B\}_{K_{AB}}$	$I_B \longrightarrow A$	:	$\{A, B, N_A\}_{K_{AB}}$

Confidentialité définie

- Un item M *est à destination d'un agent* A si, dans une attaque, l'item M est créé soit par A, soit par un autre agent B dans une session que B exécute avec A, et l'item se trouvera, dans le déroulement "sans attaque" du protocôle, dans la connaissance de A à un certain instant.
- Un item M *est confidentiel* pour Mallory pendant une attaque si
  - l'item n'est pas transmis en clair dans un message dans tout échange de messages pendant l'attaque,
  - M n'est pas à destination de Mallory, et
  - à la fin de l'attaque, M ne se trouvera pas dans la connaissance de Mallory.

Authenticité – différents types

- *Vitalité* (approx. *aliveness*) : chaque fois qu'A, termine une session complète avec (celui qu'il croît être) B, A est sûr que c'est B qui a initié/completé une session correspondante.
  - Peut ne pas être symmétrique ! (Voir attaques de reflexion)
- *Entente faible* : chaque fois qu'A, qui initie une session avec (celui qu'il croît être) B, si la session se termine correctement alors A est sûr que c'est B qui a completé une session avec A.
- *Entente non-injective* : chaque fois A initie une session avec (celui qu'il croît être) B et envoie/reçoit un ensemble d'items d, si la session se termine correctement alors A est sûr que B a bien envoyé/reçu chacun de ces items.
  - Il se peut que A ait exécuté deux sessions, tandis que B ait exécuté une seule.
- *Entente injective* : chaque fois A initie une session avec (celui qu'il croît être) B et envoie/reçoit un ensemble d'items d, si la session se termine correctement alors A est sûr que B a bien envoyé/reçu chacun de ces items, et à chaque session de A correspond bien une unique session de B et vice-versa.
- *"Récentitude"* : attribut à rajouter aux différents types d'entente.
  - E.g. *entente injective t-récente* – intervalle entre envoi et réception.

Propriétés de sécurité et attaques

- Un protocôle est **sûr par rapport à une assertion** ϕ si l'assertion ϕ est satisfaite dans tout déroulement complet du protocôle.
  - Même si tout participant démarre plusieurs sessions parallèles.
  - Y compris lors des interactions avec un Mallory !
- On doit toujours *spécifier* le but du protocôle = les assertions que le protocôle doit satisfaire !

– **Assertion** : confidentialité d'une donnée, certain type d'authenticité.

Exemple simple d'incorrectitude

- Le protocôle "simple" n'est pas sûr pour l'assertion suivante (confidentialité !) :

  A la fin du protocôle, le message M n'est connu que par A et B.

$$\begin{aligned} A &\longrightarrow B &&: \quad A, B \\ B &\longrightarrow I_A &&: \quad K_B \\ I_B &\longrightarrow A &&: \quad K_I \\ A &\longrightarrow I_B &&: \quad \{M\}_{K_I} \end{aligned}$$

- Alice considère avoir envoyé son message de manière confidentielle à Bob.
- ... mais son message est intercepté et déchiffré par Mallory !
- Éventuellement on peut compléter l'attaque de Mallory par

$$I_B \longrightarrow A \quad : \quad \{M\}_{K_A}$$

  et ainsi A et B ne se rendent même pas compte de ce qui s'est passé !

Attaques contre l'authenticité

P_1. $A \longrightarrow B$: A, B

P_2. $B \longrightarrow A$: $\{A, B\}_{K_{AB}}$

- Assertion : authentification faible (A, B) – ok.
- Assertion : authentification faible (B, A) – no !
  - Attaque très simple : Mallory initie la session !

Attaques de rejeu et parades

$$P_1. \quad A \longrightarrow B : \quad A, B$$

$$P_2. \quad B \longrightarrow A : \quad \{A, B\}_{K_{AB}}$$

- Assertion : authentification faible (A, B) – ok.
- Assertion : authentification injective (A, B) – no !
  - Problème : si Mallory recupère un message, il pourrait intervenir dans une nouvelle session et faire croire à Alice qu'il est Bob :

$$P_1'. \quad A \longrightarrow I_B : \quad A, B$$

$$P_2'. \quad I_B \longrightarrow A : \quad \{A, B\}_{K_{AB}}$$

  - Solution : rajouter des nonces/estampilles de temps :

$$P_1. \quad A \longrightarrow B : \quad A, B, N$$

$$P_2. \quad B \longrightarrow A : \quad \{A, B, N\}_{K_{AB}}$$

  - N est une valeur fraîche générée par A, que B devrait crypter avec la clé partagée.
  - Même si Mallory recupère un ancien message crypté par Bob, il ne pourra pas l'utiliser dans une nouvelle session.

Attaques de reflexion

- Modifions le protocôle d'authentification :

$$P_1. \quad A \longrightarrow B : \quad A, N$$

$$P_2. \quad B \longrightarrow A : \quad \{N\}_{K_B^{-1}}$$

- Assertion : authenticité faible (B, A) – no !
- Attaque de Mallory : reflexion

$$P_1. \quad A \longrightarrow I_B : \quad A, N$$

$$P_1' \quad I \longrightarrow B \quad I, N$$

$$P_2' \quad B \longrightarrow I \quad \{N\}_{K_B^{-1}}$$

$$P_2. \quad I_B \longrightarrow A : \quad \{N\}_{K_B^{-1}}$$

- Oublier l'identité d'un des participants n'est pas une bonne idée !
- Mais bien-sûr, authenticité faible (A, B) – ok !

Attaque contre Needham-Schroeder sans serveur

– Attaque de type *personne au milieu* :

$$
\begin{array}{llll}
M_1. & A \longrightarrow I : & \{N_1, A\}_{K_I} & \\
 & & & M'_1. \quad I_A \longrightarrow B : \quad \{N_1, A\}_{K_B} \\
 & & & M'_2. \quad B \longrightarrow I_A : \quad \{N_1, N_2\}_{K_A} \\
M_2. & I \longrightarrow A : & \{N_1, N_2\}_{K_A} & \\
M_3. & A \longrightarrow I : & \{N_2\}_{K_I} & \\
 & & & M'_3. \quad I_A \longrightarrow B : \quad \{N_2\}_{K_B}
\end{array}
$$

– Assertion : authenticité faible (B, A) – no !

– Si N_2 est une clé de chiffrement qui sera utilisée par B pour envoyer un message $Mess$, cette attaque peut engendrer une attaque contre la confidentialité de $Mess$!

– *Parade* : employer explicitement les noms dans le message de réponse :

$$
\overline{M_2}. \quad B \longrightarrow A : \quad \{N_1, N_2, B\}_{K_A}
$$

Protocôle Otway-Rees

Établissement d'une clé de session :

$$
\begin{array}{lll}
M_1. & A \longrightarrow B : & N_0, A, B, \{N_A, N_0, A, B\}_{K_{AS}} \\
M_2. & B \longrightarrow S : & N_0, A, B, \{N_A, N_0, A, B\}_{K_{AS}}, \{N_B, N_0, A, B\}_{K_{BS}} \\
M_3. & S \longrightarrow B : & N_0, \{N_A, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}} \\
M_4 & B \longrightarrow S : & N_0, \{N_A, K_{AB}\}_{K_{AS}}
\end{array}
$$

- N_0 : nonce d'identification de session.
- But : A et B doivent être convaincus de l'identité de l'autre, et qu'ils possèdent bien la clé de session qui leur permet de communiquer de manière sûre.

Attaque de type *erreur de typage* contre Otway-Rees

$$
\begin{array}{lll}
M_1. & A \longrightarrow B : & N_0, A, B, \{N_A, N_0, A, B\}_{K_{AS}} \\
M_2. & B \longrightarrow I_S : & N_0, A, B, \{N_A, N_0, A, B\}_{K_{AS}}, \{N_B, N_0, A, B\}_{K_{BS}} \\
M_3'. & I_S \longrightarrow B : & N_0, \{N_A, N_0, A, B\}_{K_{AS}}, \{N_B, N_0, A, B\}_{K_{BS}} \\
M_4 & B \longrightarrow A : & N_0, \{N_A, N_0, A, B\}_{K_{AS}}
\end{array}
$$

- Assertion : confidentialité de la clé partagée K_{AB} – no !
- Si la somme des tailles de N_0, A et B est égale à la taille de la clé K_{AB} que A et B attendent, alors les deux agents ont été dupés par Mallory d'accepter comme clé de session la juxtaposition $[N_0, A, B]$ – qui serait alors utilisée par Mallory pour déchiffrer le traffic.
- *Solution* : le type de chaque composant d'un message devrait être unique et clairement identifiable.

Attaques contre Needham-Schroeder avec serveur (2)

- Rien ne garantit la "fraîcheur" des clés que le Serveur a envoyé à Alice et Bob.
- Attaque de rejeu : si Mallory parvient à compromettre une ancienne clé, il pourrait rejouer les messages dans lesquelles le Serveur a transmis la clé respective à A et B.
- Attention ! on n'est plus dans le cadre du modèle Dolev-Yao !
  - Car on suppose que Mallory peut casser des clés !
- *Solution* : utiliser des *estampilles de temps*

$$M_2. \quad S \longrightarrow A : \quad \{K_B, B, T\}_{K_S^{-1}}$$
$$M_5. \quad S \longrightarrow B : \quad \{K_A, A, T\}_{K_S^{-1}}$$

- A et B pourraient réfuser des messages de S qui porteraient des estampilles trop anciennes.
- Cela requiert une synchronisation globale, ou un horloge de système global et sécurisé.

Règles de conception prudente

- Spécifier des conditions claires dans lesquelles un message devrait être pris en compte.
- Mentionner les noms des agents de manière explicite s'ils sont essentiels dans la compréhension du message (e.g. authentification !).
- Spécifier clairement les propriétés qu'on assume à chaque étape du protocôle.
- Spécifier clairement la raison pour laquelle le cryptage est employé (confidentialité, authentification...)
- Faire attention aux variations d'horloges (pour l'éstampillage de temps).

Utiliser des méthodes formelles pour valider la conception du protocôle !