Spécification et Vérification de protocoles cryptographiques

Steve Kremer

Laboratoire Spécification et Vérification
ENS Cachan

Troisième partie III

Adversaires actifs

Adversaire actif

Un adversaire actif peut

- intercepter des messages (et rajouter ses messages à sa connaissance)
- envoyer des messages qui sont déductibles à partir de ses connaissances
- initier des nouvelles sessions

Arbres d'exécution :

- à branchement infini (la taille des messages n'est pas bornée)
- à profondeur infinie (le nombre de sessions n'est pas borné)

Indécidabilité

Décider si un protocole préserve un secret est indécidable en général.

Définition (Problème de correspondance de Post)

Soit Σ un alphabet fini.
Données : Une séquence de paires $\langle u_i, v_i \rangle_{1 \leq i \leq n}$ $u_i, v_i \in \Sigma^*$, $n \in \mathbb{N}$
Question : Est-ce qu'il existe $k, i_1, \ldots, i_k \in \mathbb{N}$ tel que $u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$?

Indécidabilité

Décider si un protocole préserve un secret est indécidable en général.

Définition (Problème de correspondance de Post)

Soit Σ un alphabet fini.
Données : Une séquence de paires $\langle u_i, v_i \rangle_{1 \leq i \leq n}$ $u_i, v_i \in \Sigma^*$, $n \in \mathbb{N}$
Question : Est-ce qu'il existe $k, i_1, \ldots, i_k \in \mathbb{N}$ tel que $u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$?

Exemple

u_1	u_2	u_3	u_4	v_1	v_2	v_3	v_4
aba	bbb	aab	bb	a	aaa	$abab$	$babba$

Une solution : 1431
$u_1 \cdot u_4 \cdot u_3 \cdot u_1 = aba \cdot bb \cdot aab \cdot aba = a \cdot babba \cdot abab \cdot a = v_1 \cdot v_4 \cdot v_3 \cdot v_1$

Par contre pas de solution pour $\langle u_1, v_1 \rangle, \langle u_2, v_2 \rangle, \langle u_3, v_3 \rangle$

Indécidabilité

Décider si un protocole préserve un secret est indécidable en général.

Définition (Problème de correspondance de Post)

Soit Σ un alphabet fini.
Données : Une séquence de paires $\langle u_i, v_i \rangle_{1 \leq i \leq n}$ $u_i, v_i \in \Sigma^*$, $n \in \mathbb{N}$
Question : Est-ce qu'il existe $k, i_1, \ldots, i_k \in \mathbb{N}$ tel que $u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$?

Exemple

u_1	u_2	u_3	u_4	v_1	v_2	v_3	v_4
aba	bbb	aab	bb	a	aaa	$abab$	$babba$

Une solution : 1431
$u_1 \cdot u_4 \cdot u_3 \cdot u_1 = aba \cdot bb \cdot aab \cdot aba = a \cdot babba \cdot abab \cdot a = v_1 \cdot v_4 \cdot v_3 \cdot v_1$

Par contre pas de solution pour $\langle u_1, v_1 \rangle, \langle u_2, v_2 \rangle, \langle u_3, v_3 \rangle$

Théorème (Indécidabilité du problème de Post)

Le problème de Post est indécidable.

Indécidabilité (2)

On construit un protocole tel que si on peut décider la préservation du secret alors on peut décider le problème de Post.

$$A : \quad send(\{\langle u_i, v_i \rangle\}_{K_{ab}}) \qquad (1 \leq i \leq n)$$

$$B : \quad receive(\{\langle x, y \rangle\}_{K_{ab}})$$
$$send(\langle \{\langle x \cdot u_i, y \cdot v_i \rangle\}_{K_{ab}}, \{s\}_{\langle \{\langle x \cdot u_i, x \cdot u_i \rangle\}_{K_{ab}} \rangle}) \qquad (1 \leq i \leq n)$$

On suppose que K_{AB} est une clé secrète, partagée entre A et B.

L'attaquant peut déduire le secret s ssi l'attaquant peut résoudre le problème de correspondance de Post.

Les clauses de Horn

Une clause de Horn est une formule logique de la forme

$$\frac{L_1, \ldots, L_n}{L} \quad (\equiv \neg L_1 \vee \ldots \vee \neg L_n \vee L)$$

Formalisme simple et homogène pour

- modéliser les capacités de l'intrus
- modéliser les règles du protocole
- vérifier un nombre non borné de sessions

Ce formalisme est utilisé comme représentation intermédiaire (traduction à partir d'un langage de plus haut niveau) dans l'outil ProVerif [Blanchet2001]

`http ://www.di.ens.fr/~blanchet/crypto.html`

Syntaxe de la représentation des protocoles

T	::=		terme
	$\mid$	x	variable x
	$\mid$	$a[T_1, \ldots, T_n]$	nom a
	$\mid$	$f(T_1, \ldots, T_k)$	application du symbole $f \in \Sigma$ $(ar(f) = k)$
F	::=		faits
	$\mid$	$p(M_1, \ldots, M_n)$	application du prédicat p
R	::=		règle
	$\mid$	$F_1 \wedge \ldots \wedge F_n \rightarrow F$	implication

Modélisation des primitives cryptographiques

Les primitives cryptographiques sont représentées par des fonctions.

Exemple :
Le chiffrement symétrique d'un message m par une clé k est représenté par une fonction d'arité 2 encrypt(m, k).

Soit Σ la signature contenant l'ensemble des fonctions. On peut partitioner cet ensemble en constructeurs et destructeurs.

- Les constructeurs sont les fonctions qui apparaissent explicitement dans les termes
- Les destructeurs manipulent les termes
  Un destructeur g est défini par une équation $g(T_1, \ldots, T_k) = T$ où $T_1, \ldots, T_k, T$ contiennent uniquement des constructeurs et des variables

Exemples de modélisation de primitives cryptographiques

Chiffrement symétrique

Le chiffrement symétrique est modélisé par un constructeur encrypt(m, k) et un destructeur decrypt$(\text{encrypt}(m, k)), k) = m$

Signature numérique

La signature numérique est modélisé par deux constructeurs sign(m, sk) et pk(sk) et un destructeur getmsg$(\text{sign}(m, sk), \text{pk}(sk)) = m$

Fonction de hashage

Une fonction de hashage est modélisée par un constructeur h(m)

Les capacités de l'intrus par des clauses de Horn

- Introduction du prédicat $I(m)$ pour modéliser la connaissance de l'intrus
- $I(m)$ est vrai si et seulement si l'intrus connaît le message m

Exemple de modélisation des capacités de l'intrus

$$\frac{I(m), I(n)}{I(\text{pair}(m,n))}(\text{pair}) \qquad \frac{I(\text{pair}(m,n))}{I(m)}(\text{unpairL}) \qquad \frac{I(\text{pair}(m,n))}{I(n)}(\text{unpairR})$$

$$\frac{I(m), I(pubk)}{I(\text{enc}(m,pubk))}(\text{encrypt}) \qquad \frac{I(enc(m,pk(x))), I(x)}{I(m)}(\text{decrypt})$$

Capacités de l'intrus par des clauses de Horn (2)

Soit f un constructeur d'arité n
f est modélisé par la règle

$$\frac{I(x_1), \ldots, I(x_n)}{I(f(x_1, \ldots, x_n))}$$

Soit g un destructeur défini par l'équation $g(T_1, \ldots, T_n) = T$
g est modélisé par la règle

$$\frac{I(T_1), \ldots, I(T_n)}{I(T)}$$

Remarque :
Notons que le destructeur g n'apparaît jamais explicitement dans les règles !

Connaissance initiale de l'intrus

Généralement on donne à l'intrus une connaissance initiale
Le fait que l'intrus connaisse initialement le terme clos (sans variable) T est modélisé par la règle

$$\rightarrow I(T) \quad \text{(noté simplement } I(T)\text{)}$$

Exemple :
L'intrus connait la clé publique correspondant à la clé secrète $sA[]$ de A :

$$I(pk(sA[]))$$

Les règles de protocoles par clauses de Horn

Le protocole de Needham-Schroeder modélisé par des clauses de Horn

$$\begin{array}{lcll} A & \longrightarrow & B & : \{N_a, A\}_{\mathrm{pub}(B)} \\ B & \longrightarrow & A & : \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A & \longrightarrow & B & : \{N_b\}_{\mathrm{pub}(B)} \end{array}$$

$$\frac{I(pk(x))}{I(enc((Na[pk(x)], pk(sA[])), pk(x)))}$$

$$\frac{I(encrypt((x, y), pk(sB[])))}{I(encrypt((x, Nb[x, y]), y))}$$

$$\frac{I(pk(x)), I(encrypt((Na[pk(x)], y), pk(sA[])))}{I(encrypt(y, pk(x)))}$$

Modélisation du premier message

Nous supposons que l'intrus choisit avec qui A exécute le protocole

Les règles de protocoles par clauses de Horn

Le protocole de Needham-Schroeder modélisé par des clauses de Horn

$$\begin{array}{lcll} A & \longrightarrow & B & : \{N_a, A\}_{\mathrm{pub}(B)} \\ B & \longrightarrow & A & : \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A & \longrightarrow & B & : \{N_b\}_{\mathrm{pub}(B)} \end{array}$$

$$\frac{I(pk(x))}{I(enc((Na[pk(x)], pk(sA[])), pk(x)))}$$

$$\frac{I(encrypt((x, y), pk(sB[])))}{I(encrypt((x, Nb[x, y]), y))}$$

$$\frac{I(pk(x)), I(encrypt((Na[pk(x)], y), pk(sA[])))}{I(encrypt(y, pk(x)))}$$

Modélisation des valeurs fraîches

Les valeurs fraîches sont modélisées comme des fonctions des paramètres du protocole.

Les règles de protocoles par clauses de Horn

Le protocole de Needham-Schroeder modélisé par des clauses de Horn

$$\begin{array}{lcll} A & \longrightarrow & B & : \{N_a, A\}_{\mathrm{pub}(B)} \\ B & \longrightarrow & A & : \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A & \longrightarrow & B & : \{N_b\}_{\mathrm{pub}(B)} \end{array}$$

$$\frac{I(pk(x))}{I(enc((Na[pk(x)], pk(sA[])), pk(x)))}$$

$$\frac{I(encrypt((x, y), pk(sB[])))}{I(encrypt((x, Nb[x, y]), y))}$$

$$\frac{I(pk(x)), I(encrypt((Na[pk(x)], y), pk(sA[])))}{I(encrypt(y, pk(x)))}$$

L'intrus contrôle le réseau

On suppose que l'intrus a un contrôle total sur le réseau : tout message transite via l'intrus

Les règles de protocoles par clauses de Horn

Le protocole de Needham-Schroeder modélisé par des clauses de Horn

$$\begin{array}{lcll} A & \longrightarrow & B & : \{N_a, A\}_{\mathrm{pub}(B)} \\ B & \longrightarrow & A & : \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A & \longrightarrow & B & : \{N_b\}_{\mathrm{pub}(B)} \end{array}$$

$$\frac{I(pk(x))}{I(enc((Na[pk(x)], pk(sA[])), pk(x)))}$$

$$\frac{I(encrypt((x, y), pk(sB[])))}{I(encrypt((x, Nb[x, y]), y))}$$

$$\frac{I(pk(x)), I(encrypt((Na[pk(x)], y), pk(sA[])))}{I(encrypt(y, pk(x)))}$$

L'intrus contrôle le réseau

On suppose que l'intrus a un contrôle total sur le réseau : tout message transite via l'intrus

Les règles de protocoles par clauses de Horn

Le protocole de Needham-Schroeder modélisé par des clauses de Horn

$$\begin{array}{lcll} A & \longrightarrow & B & : \{N_a, A\}_{\mathrm{pub}(B)} \\ B & \longrightarrow & A & : \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A & \longrightarrow & B & : \{N_b\}_{\mathrm{pub}(B)} \end{array}$$

$$\frac{I(pk(x))}{I(enc((Na[pk(x)], pk(sA[])), pk(x)))}$$

$$\frac{I(encrypt((x, y), pk(sB[])))}{I(encrypt((x, Nb[x, y]), y))}$$

$$\frac{I(pk(x)), I(encrypt((Na[pk(x)], y), pk(sA[])))}{I(encrypt(y, pk(x)))}$$

L'intrus contrôle le réseau

On suppose que l'intrus a un contrôle total sur le réseau : tout message transite via l'intrus

Approximations

Ce modèle contient des imprécisions

- les valeurs fraiches sont modélisées par des noms qui sont fonction des messages reçus précédement
  Si l'intrus envoie les même messages les même noms "frais" seront utilisés
- Une étape du protocole peut être exécutée plusieurs fois si les étapes précédentes ont été exécutées au moins une fois
  Exemple :
  1. L'intrus envoie à A le message M_1
  2. A répond par M_2
  3. L'intrus envoie à A le message M_3
  4. A répond par M_4
  5. L'intrus envoie à A le message M_3' (sans exécuter les 2 premières étapes)
  6. A répond par M_4'

Approximations correctes

Les approximations peuvent donner lieu à des fausses attaques

En pratique sur les protocoles étudiés, les fausses attaques sont très rares.

Les approximations sont correctes

Si on prouve la correction d'un protocole dans le modèle des clauses de Horn, le protocole est aussi correct dans un modèle plus précis.

La dérivabilité

Définition [Implication entre règles]
$(H_1 \rightarrow C_1) \Rightarrow (H_2 \rightarrow C_2)$ ssi il existe une substitution σ telle que $C_1\sigma = C_2$ et $H_1\sigma = H_2$ (H_1 et H_2 sont des ensembles d'hypothèses)

Définition [Dérivabilité]
Soit F un fait clos (i.e. qui ne contient pas de variable) et B un ensemble de règles. F est dérivable de B ssi il existe un arbre fini tel que

1. tous les nœuds (sauf la racine) sont étiquettés par une règle $R \in B$
2. les arcs sont étiquettés par des faits
3. si un arbre contient un nœud étiquetté par une règle R avec un arc entrant, étiquetté F_0 et n arcs sortants étiquettés $F_1, \ldots, F_n$ alors $R \Rightarrow \{F_1, \ldots, F_n\} \rightarrow F_0$
4. La racine a un arc sortant étiquetté par F

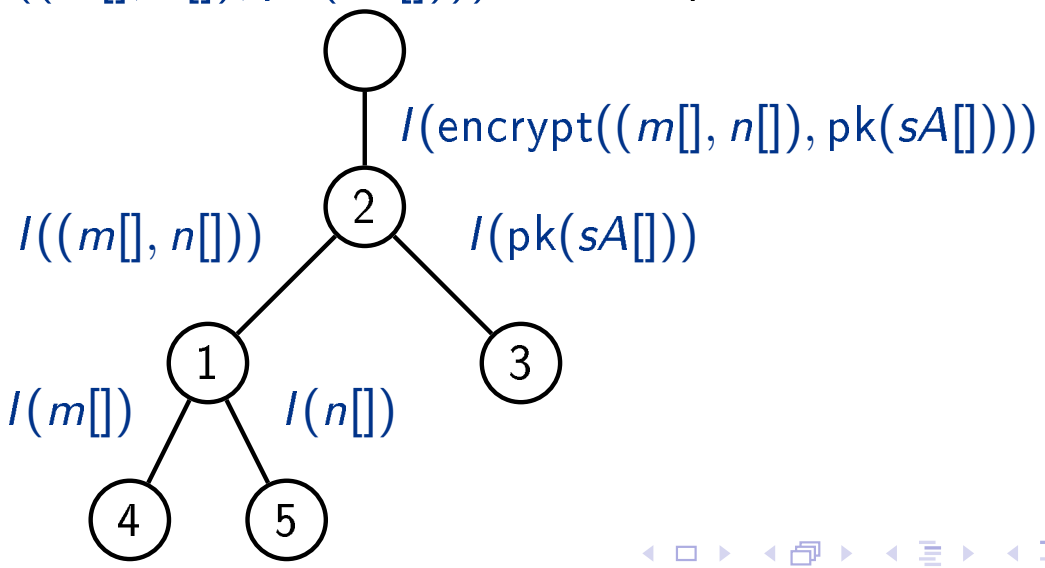
La propriété de secret

Une des propriétés de sécurité les plus basiques est le secret
On dit qu'un terme clos S est secret si on ne peut pas dériver $I(S)$ des règles modélisant le protocole et les capacités de l'intrus.

Exemple : Supposons qu'on ait les règles

$$I(x) \wedge I(y) \rightarrow I((x,y)) \quad (1) \qquad I(m[]) \quad (4)$$
$$I(x) \wedge I(y) \rightarrow I(\text{encrypt}(x,y)) \quad (2) \qquad I(n[]) \quad (5)$$
$$I(\text{pk}(sA[])) \quad (3)$$

On peut dériver $I(\text{encrypt}((m[], n[]), \text{pk}(sA[])))$ de la façon suivante :



Automatisation ?

Nous avons un ensemble de règles (des clauses de Horn) et nous demandons si un fait F peut être dérivé de ces règles

Ce problème correspond exactement au problème résolu par Prolog

Mais : les algorithmes utilisés dans Prolog ne terminent pas pour des règles typiques apparaissant dans la modélisation de protocoles cryptographiques

Dans [Blanchet2001], Bruno Blanchet présente un nouvel algorithme de résolution qui "guide" la résolution et qui est adapté aux protocoles cryptographiques

Définitions préliminaires ...

Combinaison de règles simplifiée

Soit $R = H \rightarrow C$ et $R' = H' \rightarrow C'$ deux règles. Si $C \in H'$ Alors

$$R \circ R' = H \cup (H' \setminus C) \rightarrow C'$$

Définition [Combinaison de règles]

Soit $R = H \rightarrow C$ et $R' = H' \rightarrow C'$ deux règles. Supposons qu'il existe un fait $F_0 \in H'$, tel que F_0 et C sont unifiables et σ est l'unificateur le plus général pour C et F_0. Alors

$$R \circ_{F_0} R' = (H \cup (H' \setminus F_0))\sigma \rightarrow C'\sigma$$

Exemple :

$R = I(\mathrm{pk}(x)) \rightarrow I(\mathrm{encrypt}(sign(msg[], skA[]), \mathrm{pk}(x)))$

$R' = I(\mathrm{encrypt}(m, \mathrm{pk}(sk))) \wedge I(sk) \rightarrow I(m)$

Prenons $F_0 = I(\mathrm{encrypt}(m, \mathrm{pk}(sk))$. Alors

$$R \circ_{F_0} R' = I(\mathrm{pk}(x)) \wedge I(x) \rightarrow I(sign(msg[], skA[]))$$

avec $\sigma = \{sk = x, m = sign(msg[], skA[])\}$

Pour guider l'algorithme

Soit S un ensemble fini de faits. On dit que $F \in_r S$ ssi il existe une substitution σ de variables par d'autres variables tel que $F\sigma \in S$.

Dans l'algorithme S servira pour guider le choix de combinaison de règles : on ne combine pas R et R' par $R \circ_{F_0} R'$ si $F_0 \in_r S$

Typiquement, on choisit $S = \{I(x)\}$ pour éviter la situation suivante. Soit les règles

$$I(x) \rightarrow I(pk(x)) \quad (1)$$
$$I(pk(x)) \wedge I(y) \rightarrow \text{encrypt}(y, \text{pk}(x)) \quad (2)$$

Si on applique la combinaison $(2) \circ_{I(x)} (1)$ on obtient

$$I(pk(x)) \wedge I(y) \rightarrow I(\text{pk}(\text{encrypt}(y, \text{pk}(x)))) \quad (3)$$

On peut alors appliquer la combinaison $(3) \circ_{I(x)} (1)$ et on obtient

$$I(pk(x)) \wedge I(y) \rightarrow I(\text{pk}(\text{pk}(\text{encrypt}(y, \text{pk}(x))))) \quad (4)$$

On voit que ces combinaisons successives ne terminent pas. De façon similaire si on choisit $I(y) \in S$ comme F_0 on boucle sur le chiffrement

Algorithme de résolution : phase 1

$$\text{Soit } add(R,B) = \begin{cases} B & si\ \exists R' \in B, R' \Rightarrow R \\ \{R\} \cup \{R' \in B \mid R \not\Rightarrow R'\} & sinon \end{cases}$$

Soit B_0 l'ensemble de règles décrivant le protocole et l'intrus

1. Pour tout $R \in B_0$ $B = add(R, B)$
2. Soit $R \in B$, $R = H \rightarrow C$ et $R' \in B$, $R' = H' \rightarrow C'$. Supposons qu'il existe $F_0 \in H'$ tel que
   (a) $R \circ_{F_0} R'$ est défini
   (b) $\forall F \in H, F \in_r S$
   (c) $F_0 \notin_r S$

   Alors $B = add(R \circ_{F_0} R', B)$
   Exécuter l'étape 2. jusqu'à atteindre un point fixe
3. $B' = \{(H \rightarrow C) \in B \mid \forall F \in H, F \in_r S\}$

Après l'exécution de la phase 1, on a qu'un fait clos F peut être dérivé de B' ssi F peut être dérivé de B_0.

Algorithme de résolution : phase 2

derivablerec(R, B'')

1. derivablerec$(R, B'') = \emptyset$
   si $\exists R' \in B''.R' \Rightarrow R$ `# boucle : backtrack`
2. sinon, derivablerec$(\emptyset \rightarrow C, B'') = \{C\}$ `# preuve de C`
3. sinon, derivablerec$(R, B'') = \cup\{derivablerec(R' \circ_{F_0} R, \{R\} \cup B'') \mid$
   $R' \in B', F_o$ est tel que $R' \circ_{F_0} R$ est défini$\}$

derivable(F) = derivablerec$(\{F\} \rightarrow F, \emptyset\})$

Intuitivement,

- l'hypothèse de R contient les faits qu'on essaie à un moment donné
- la conclusion de R contient le fait qu'on essaie de dériver
- l'ensemble B'' est l'ensemble des règles déjà rencontrées

On a que : F est dérivable de B_0 ssi $F \in$ derivable(F)

Remarques sur l'algorithme

Le point fixe de la première phase peut ne pas terminer

En pratique, sur presque tous les exemples de protocoles, cette première phase termine !

Il est possible de montrer que si $S = \{I(x)\}$ et que si F est un fait clos, alors derivable(F) termine

L'outil ProVerif implémentant cet algorithme contient de nombreuses extensions et optimisations ...

Conclusion et perspectives

La vérification de protocoles cryptographiques est un domaine ayant des applications directes à des problèmes concrets

Il y a des questions théoriques intéressantes : complexité, algorithmes de vérification, ...

On n'a parlé que de la propriété de secret (en terme de déduction) : il existe des notions de secret plus fortes ; d'autres propriétés sont parfois nécessaires (authentification, anonymat, etc.)

On peut modéliser des propriétés algébriques de certaines primitives cryptographiques : nouvelles questions de decidabilité et de complexité

...