REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Filière: Informatique

Promo: 2CS Spécialité Cyber sécurité

Rapport de projet du module Méthodes formelles pour la sécurité

Sujet : Vérification et validation d'un protocole cryptographique

Réalisé par :

- **Micipsa MERZOUK G01**
- **Walid OUTERBAH G01**
- **Sara BENTAFAT G01**
- **Imad eddine LEKOUAGHET G01**

Année Universitaire : 2023/ 2024

Table des matières

1. Introduction

La sécurité des systèmes informatiques est devenue une préoccupation majeure dans notre ère numérique. Avec la prolifération des échanges de données sensibles sur les réseaux informatiques, il est impératif de garantir que ces échanges sont sécurisés et protégés contre les attaques malveillantes. Les protocoles cryptographiques jouent un rôle central dans cette mission, en permettant aux entités de communiquer de manière sécurisée à travers des canaux non sécurisés.

Cependant, la conception et la mise en œuvre de protocoles cryptographiques présentent des défis complexes. Les erreurs de conception peuvent conduire à des vulnérabilités qui compromettent la sécurité des systèmes, exposant ainsi les données confidentielles à des risques potentiels. Pour cette raison, la vérification formelle des protocoles cryptographiques est devenue une pratique essentielle pour garantir leur robustesse et leur sécurité.

Dans ce rapport, nous nous concentrons sur la vérification formelle d'un protocole cryptographique simple à l'aide de l'outil **ProVerif**.

Notre objectif est d'évaluer la sécurité du protocole en question en vérifiant sa conformité à certaines propriétés de sécurité spécifiées. Nous cherchons à déterminer si le protocole résiste à diverses attaques potentielles, telles que l'interception des messages, la falsification de données et la divulgation d'informations confidentielles.

Nous commençons par décrire en détail le protocole cryptographique examiné, en mettant en évidence ses objectifs, ses composants et son fonctionnement. Ensuite, nous expliquons la méthodologie que nous avons suivie pour effectuer la vérification formelle à l'aide de ProVerif, un outil largement utilisé pour l'analyse de sécurité des protocoles cryptographiques. Nous décrivons les étapes de modélisation du protocole dans ProVerif et la spécification des propriétés de sécurité à vérifier.

Suite à cela, nous présentons les résultats de notre analyse de sécurité, en discutant des menaces potentielles prises en compte et des conclusions tirées de la vérification formelle. Nous évaluons également les forces et les faiblesses du protocole en termes de sécurité, en mettant en évidence les domaines où des améliorations pourraient être apportées.

2. Description du protocole:

Un protocole d'échange de clés entre le client A et le serveur B, On suppose que chaque entité dispose d'une paire de clés publique/privée, et que le client A connaît la clé publique pk(skB) du serveur B. L'objectif du protocole est que le client A partage le secret s avec le serveur B. Le protocole se déroule comme suit. À la demande d'un client A, le serveur B génère une clé symétrique fraîche k (clé de session), l'associe à son identité (clé publique pk(skB)), la signe avec sa clé secrète skB et l'encrypte en utilisant la clé publique pk(skA) de son client. Autrement dit, le serveur envoie le message aenc(sign((pk(skB),k),skB),pk(skA)). Lorsque A reçoit ce message, elle le décrypte en utilisant sa clé secrète skA, vérifie la signature numérique faite par B en utilisant sa clé publique pk(skB), et extrait la clé de session k. A utilise cette clé pour chiffrer symétriquement le secret s. La raison derrière le protocole est qu'A reçoit la signature chiffrée de manière asymétrique avec sa clé publique et donc elle devrait être la seule à pouvoir déchiffrer son contenu. De plus, la signature numérique devrait garantir que B est l'initiateur du message. Les messages envoyés sont illustrés comme suit :

$A \rightarrow B$: pk(skA)

$B \rightarrow A$: aenc(sign((pk(skB),k),skB),pk(skA))

$A \rightarrow B$: senc(s ,k)

3. Objectifs du protocole:

Ce protocole est conçu principalement pour assurer les fonctionnalitées suivantes:

1. Confidentialité : la valeur S est connue uniquement de A et B.
2. Authentification de A auprès de B : si B arrive à la fin du protocole et qu'il croit avoir partagé la clé k avec A, alors A était effectivement son interlocuteur et elle a partagé k.
3. Authentification de B auprès de A : si A arrive à la fin du protocole avec la clé partagée k, alors B a proposé k pour être utilisée par A.

4. Faille de sécurité de ce protocole:

Toutefois, le protocole présente une faille face à une attaque de l'homme du milieu (MITM attack), comme illustré ci-dessous. Si un participant malveillant, nommé I, initie une session avec B, alors I peut se faire passer pour B lors d'une session ultérieure que le client A entame avec B. À la fin du protocole, A est persuadée qu'elle partage le secret s avec B, alors qu'en réalité, elle partage s avec I.

I → B : pk(skI)

B → I : aenc(sign((pk(skB),k),skB),pk(skI))

A → B : pk(skA)

I → A : aenc(sign((pk(skB),k),skB),pk(skA))

A → B : senc(s ,k)

5. Correction:

Le protocole peut facilement être corrigé en ajoutant l'identité du client prévu :

A → B : pk(skA)

B → A : aenc(sign((pk(skA),pk(skB),k),skB),pk(skA))

A → B : senc(s ,k)

6. Vérification formelle en utilisant ProVerif:

Dans le cadre de notre étude, on va essayer d’écrire d’une façon formelle le fonctionnement du protocole précisément défini sur l’outil de vérification et de validation des protocoles cryptographique ProVerif, et on voit s’il va détecter que le protocole est vulnérable à l’attaque MAN IN THE MIDDLE.

6.1. Description de l’outil:

ProVerif est un outil permettant d'analyser automatiquement la sécurité des protocoles cryptographiques. Il prend en charge, sans s'y limiter, les primitives cryptographiques suivantes : le chiffrement symétrique et asymétrique, les

signatures numériques, les fonctions de hachage, l'engagement de bits et les preuves de connaissance nulle non interactives. ProVerif est capable de prouver des propriétés d'accessibilité, des assertions de correspondance et une équivalence observationnelle. Ces capacités sont particulièrement utiles dans le domaine de la sécurité informatique car elles permettent l'analyse des propriétés de confidentialité et d'authentification. De plus, des propriétés émergentes telles que la vie privée, la traçabilité et la vérifiabilité peuvent également être prises en compte. L'analyse des protocoles est considérée par rapport à un nombre illimité de sessions et à un espace de messages illimité. De plus, l'outil est capable de reconstruire les attaques : lorsqu'une propriété ne peut être prouvée, ProVerif tente de reconstruire une trace d'exécution qui contredit la propriété désirée.

6.2.Formalisation du protocole:

```
(* Déclaration des types de données *)
type key.
fun senc(bitstring, key): bitstring. (* Fonction de chiffrement symétrique *)
reduc forall m: bitstring, k: key; sdec(senc(m,k),k) = m. (* Propriété de déchiffrement symétrique *)

type skey. (* Clé secrète symétrique *)
type pkey. (* Clé publique asymétrique *)

fun pk(skey): pkey. (* Fonction de génération de clé publique à partir de clé secrète *)
fun aenc(bitstring, pkey): bitstring. (* Fonction de chiffrement asymétrique *)

reduc forall m: bitstring, sk: skey; adec(aenc(m,pk(sk)),sk) = m. (* Propriété de déchiffrement asymétrique *)

type sskey. (* Clé secrète pour les signatures *)
type spkey. (* Clé publique pour les signatures *)

fun spk(sskey): spkey. (* Fonction de génération de clé publique pour les signatures à partir de clé secrète *)
fun sign(bitstring, sskey): bitstring. (* Fonction de signature *)

reduc forall m: bitstring, ssk: sskey; getmess(sign(m,ssk)) = m. (* Propriété de vérification du contenu signé *)
reduc forall m: bitstring, ssk: sskey; checksign(sign(m,ssk),spk(ssk)) = m. (* Propriété de vérification de la signature *)

(* Déclaration des canaux de communication *)
free c:channel.

(* Déclaration des variables *)
free s:bitstring [private].

(* Déclaration d'une requête pour simuler une attaque *)
query attacker(s).

(* Définition du processus clientA *)
let clientA(pkA:pkey,skA:skey,pkB:spkey) =
    out(c,pkA); (* Envoi de la clé publique du client A *)
    in(c,x:bitstring); (* Réception d'un message chiffré *)
    let y = adec(x,skA) in (* Déchiffrement du message avec la clé secrète du client A *)
```

```
    let (=pkB,k:key) = checksign(y,pkB) in (* Vérification de la signature *)
    out(c,senc(s,k)). (* Chiffrement symétrique du secret s avec la clé k *)

(* Définition du processus serverB *)
let serverB(pkB:spkey,skB:sskey) =
    in(c,pkX:pkey); (* Réception de la clé publique du client A *)
    new k:key; (* Génération d'une nouvelle clé symétrique *)
    out(c,aenc(sign((pkB,k),skB),pkX)); (* Chiffrement asymétrique du message contenant la clé symétrique signée *)
    in(c,x:bitstring); (* Réception du secret chiffré *)
    let z = sdec(x,k) in (* Déchiffrement symétrique du secret avec la clé k *)

(* Définition du processus principal *)
process
    new skA:skey; (* Génération d'une nouvelle clé secrète pour le client A *)
    new skB:sskey; (* Génération d'une nouvelle clé secrète pour le serveur B *)
    let pkA = pk(skA) in out(c,pkA); (* Génération de la clé publique du client A et envoi *)
    let pkB = spk(skB) in out(c,pkB); (* Génération de la clé publique du serveur B et envoi *)
    ( (!clientA(pkA,skA,pkB)) | (!serverB(pkB,skB)) ) (* Exécution concurrente des processus clientA et serverB *)

```

Output:

```
-- Process 1-- Query not attacker(s[]) in process 1
Translating the process into Horn clauses...
Completing...
Starting query not attacker(s[])
goal reachable: attacker(s[])
RESULT not attacker(s[]) is false.

------------------------------------------------------------
Verification summary:

Query not attacker(s[]) is false.

------------------------------------------------------------
```

Interprétation du résultat d’exécution:

Ce résultat de vérification indique que la requête "not attacker(s[])" est fausse, ce qui signifie qu'il existe une exécution du processus dans laquelle l'attaquant peut obtenir le secret s. En d'autres termes, il existe un scénario dans lequel l'attaquant peut réussir son attaque et obtenir le secret s, contournant ainsi les mécanismes de sécurité du protocole. Cela suggère qu'il existe une vulnérabilité dans le protocole qui permet à l'attaquant d'intercepter le secret. Donc proVerif a pu détecter l’attaque

sur le protocole sauf qu'il na pas donné une trace de l'attaquant ou les étapes de l'attaque dans ce cas.

Correction du protocole:

En modifiant la ligne N°=42 :

```
42        out(c,aenc(sign((pkB,k),skB),pkX));
```

avec:

```
42        out(c,aenc(sign((pkX,pkB,k),skB),pkX));
```

on aura le résultat d'exécution suivant:

```
-- Process 1-- Query not attacker(s[]) in process 1
Translating the process into Horn clauses...
Completing...
Starting query not attacker(s[])
RESULT not attacker(s[]) is true.

------------------------------------------------------------
Verification summary:

Query not attacker(s[]) is true.

------------------------------------------------------------
```

Interprétation du résultat d'exécution:

Ce résultat de vérification indique que la requête "not attacker(s[])" est vraie. Cela signifie qu'après analyse, aucune exécution du processus n'a été trouvée dans laquelle l'attaquant parvient à obtenir le secret s. En d'autres termes, le protocole semble résister à une attaque de l'homme du milieu dans les conditions spécifiées. Cela suggère que le protocole est sécurisé et que les mécanismes de sécurité mis en place sont efficaces pour protéger le secret s contre les attaquants potentiels.

7. Conclusion:

La vérification formelle du protocole cryptographique à l'aide de l'outil ProVerif a fourni des résultats significatifs quant à sa robustesse face à l'attaque de l'homme du milieu. En examinant attentivement les processus impliqués dans le protocole et

en simulant divers scénarios d'attaque, nous avons pu déterminer que le protocole résiste efficacement à cette forme d'attaque. Cela suggère que les mécanismes de sécurité mis en place, tels que le chiffrement symétrique et asymétrique, ainsi que les signatures numériques, sont suffisamment solides pour protéger les communications et empêcher les attaquants d'intercepter les données sensibles. Cependant, malgré ces résultats encourageants, il est important de souligner qu'aucun système n'est totalement immunisé contre les attaques. Des recherches continues et une vigilance constante sont nécessaires pour identifier et corriger les vulnérabilités potentielles qui pourraient compromettre la sécurité du protocole. De plus, l'évolution constante des technologies et des méthodes d'attaque souligne l'importance d'une approche proactive en matière de sécurité des systèmes d'information.

En conclusion, cette étude démontre l'efficacité de la vérification formelle dans l'évaluation de la sécurité des protocoles cryptographiques. Elle souligne également l'importance d'une conception sécurisée et d'une analyse approfondie des protocoles pour garantir la confidentialité et l'intégrité des communications dans les environnements informatiques.