INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Security

SUBMITTED BY

Hugo LE BELGUET 5ISS-A2
Yacine BENCHEHIDA
Assia NGUYEN
Agathe LIEVRE
Leila MENECEUR

Academic Year: 2021-2022
January 14, 2022

RTK, you're not lost

Notre projet consiste à mettre en place une solution de localisation RTK, le but est de récupérer la charge utile d'un ballon météorologique. Ce ballon va être envoyé à 32km d'altitude avec des appareils de mesure pour faire des relevées. La particularité du RTK (Real Time Kinetic) est que, contrairement à un GPS classique, ça offre une précision au cm près. C'est rendu possible par une correction de la position entre une base (immobile) au sol et un rover (mobile) faisant partie de la charge utile du ballon.

On a donc deux types de télécommunication qui s'opèrent, la première entre la base est le rover en LoRa et la seconde entre la base/rover et les satellites. La base et le rover calculent leurs positions chacun indépendamment l'un de l'autre, ensuite le rover envoie sa position à la base, qui va la corriger et la renvoyer au rover.

Dans notre cas, on pourrait imaginer qu'un attaquant voudrait récupérer la charge utile avant nous pour en exploiter les données. Pour se faire il n'aurait pas d'accès physique car le rover se situe dans un ballon à plusieurs kilomètres du sol, et la base est en permanence avec l'équipe ou dans un lieu sécurisé. L'attaquant n'aura donc accès au médium qu'en écoutant les communications.

Notre projet ne prévoit absolument aucune mesure visant à la sécurité des communications car ça ne présente pas réellement d'intérêt, en effet il est peu probable qu'un attaquant veuille voler des données météorologiques destinées à la recherche. Donc toutes les données de localisation circulent en clair et sont accessible par n'importe qui en écoute à la bonne fréquence.

Ainsi une personne mal intentionnée n'aurait qu'à lancer une attaque de type « man in the middle » afin de récupérer la position du rover, la rediriger vers sa propre base pirate pour obtenir la correction et envoyer une fausse position à la base légitime en parallèle de cette manière l'équipe serait envoyé au mauvais endroit et il pourrait récupérer le paquet sans difficulté.

Pour remédier à cela, on peut imaginer mettre en place un chiffrement de la liaison radio à l'aide d'une clé de chiffrement publique et privée. On pourrait pour cela, par exemple utiliser l'algorithme d'ElGamal. Il permet un chiffrement entre la base et le rover directement sur un canal public ce qui ne dénature pas le canal et permet d'assurer l'intégrité et la confidentialité des données. Les clés sont générées de manière asynchrone. On a privilégié ce mode car on peut imaginer un cas d'utilisation avec plusieurs rover dans ce sens le mode asynchrone permet une plus grande flexibilité.

Sans rentrer dans les détails voici le fonctionnement de cet algorithme.

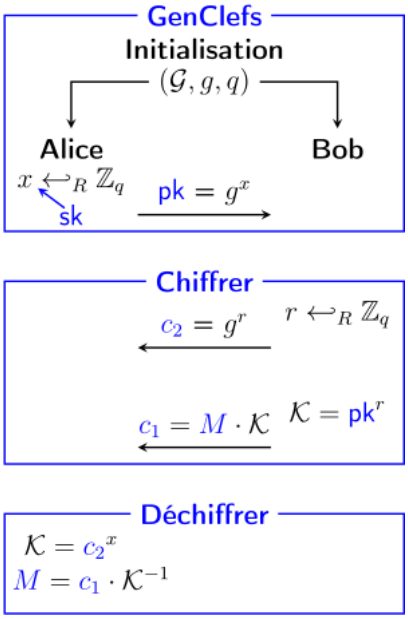

Figure 1 : pseudo-code algorithme ElGamal

Voici les différentes étapes :

- *On génère un entier premier A très grand*
- *On sélectionne un entier x compris dans [1 ;A-1]*
- *On sélectionne G, une racine génératrice de A*
- *On calcul la clé publique :* $y = G^x modulo(A)$ *et la privée : x*
- *Pour encrypter : on sélectionne R un entier compris dans [1 ;A-1]*
- *Pour* $c_1 = (G^R) modulo(A)$
- *Pour* $c_2 = (M \times y^R) modulo(A)$
- *Pour décrypter :* $M = (c_1 (c_2^x)^{-1}) modulo(A)$

On va maintenant étudier cette contre-mesure sur ProVerif. Voici le code que nous avons mis en place avec à chaque étape une explication.

```
type publickey.  -> clé publique

type privatekey.  -> clé privée

free publickeyy: publickey.  -> on précise ce qui est connu de tous

free privatekeyy: privatekey [private]. -> on définit ce qui est privé

free c : channel.  -> channel en clair


fun aenc1(bitstring,publickey) : bitstring.  -> encryption asynchrone de c1

fun aenc2(bitstring,publickey) : bitstring.  -> encryption asynchrone de c2

reduc forall message1:bitstring,pk1:publickey,z1:privatekey;
dec1(aenc1(message1,pk1),z1) = message1. -> déclaration des variables de
décryption de la transmission 1

reduc forall message2:bitstring,pk2:publickey,z2:privatekey;
dec2(aenc2(message2,pk2),z2) = message2. -> déclaration des variables de la
décryption de la transmission 2


query attacker(privatekeyy). -> on définit ce que l'attaquant essaye de
connaitre


process

new c1:bitstring; -> création variable (messages à transmettre donc
coordonnées GPS)

new c2:bitstring; -> création variable (messages à transmettre donc
coordonnées GPS)

out (c,aenc1(c1,publickeyy)); -> On envoie les deux messages en crypté sur
le channel c

out (c,aenc2(c2,publickeyy)); -> On envoie les deux messages en crypté sur
le channel c

in (c,z:privatekey) [precise]; -> on déclare une variable z qui correspond
à la clé privée de la personne qui reçoit le message

out (c,dec1(aenc1(c1,publickeyy),z)); -> On décrypte le message reçu

out (c,dec2(aenc2(c2,publickeyy),z)) -> On décrypte le message reçu
```

Avec ce code on met en place un chiffrement asynchrone ElGamal avec une paire de clé publique et privée pour le serveur et le client. On veut ensuite prouver que si un attaquant se met sur cette liaison qui passe en clair, il ne pourra pas récupérer les clés privées et donc ne pourra pas déchiffrer le message.

Le résultat du test le prouve :

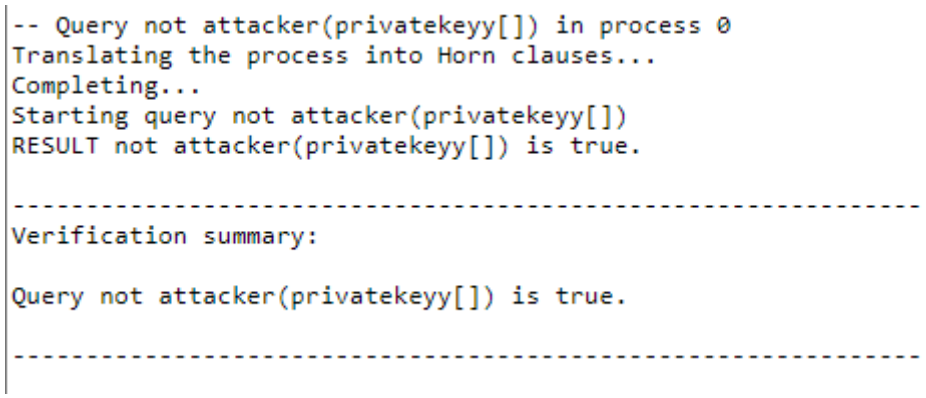

Figure 2 : résultat ProVerif