Formal Verification of Security Protocols with ProVerif

Giann Spilere Nandi

Periodic Seminars

12 Fev 19

Outline

Introduction

Cryptographic Protocols

Cryptographic Protocols:

- series of steps;
- message exchanges;
- hostile environment;
- security properties.

Introduction

Cryptographic Protocols

Security Properties:

- **secrecy**;
- **authenticity**;
- integrity;
- ...

Introduction
Cryptographic Protocols

Security mechanisms that are commonly used by encryption protocols:

- Public Key Encryption
- Symmetric Encryption
- Hash Functions

Introduction

Cryptographic Protocols

The effectiveness of the protocol relies on keeping in secret the **keys**, not the **steps**.

Introduction

Formal Methods

Formal methods are **techniques** used to model complex systems as **mathematical** and **logical** entities.

Introduction

Formal Methods Applied to Cryptographic Protocols

Aim: identify possible vulnerabilities!

Introduction

Formal Methods Applied to Cryptographic Protocols

Security Protocols:

- simple execution flow;
- difficult to design non-exploitable steps;

Introduction

Formal Methods Applied to Cryptographic Protocols

Initial research on security protocols formal verification date back from the 80's.

Nowadays there are multiple automated formal verification tools.

ProVerif

Introduction to the Tool

ProVerif is based on the formal model (**Dolev-Yao model**).

ProVerif

Introduction to the Tool - Dolev-Yao model

Attackers are capable of:

- permeating themselves in **between the communication** of two participants in any process of the protocol;
- **modifying** and **copying** fragments of information sent in the network;
- replicating messages;
- forging messages;

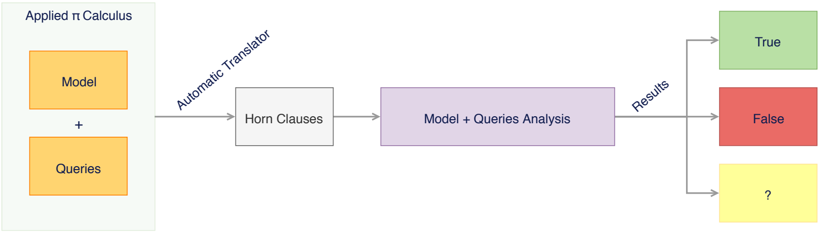
ProVerif

Introduction to the Tool - Dolev-Yao model

Attackers are capable of:

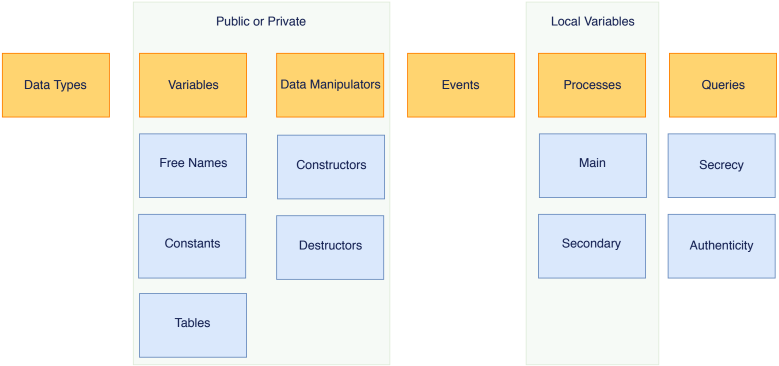
- keeping track of **all messages** sent in the network;
- actively participating as normal agents in the protocol;
- receiving responses sent to other participants

ProVerif
Introduction to the Tool



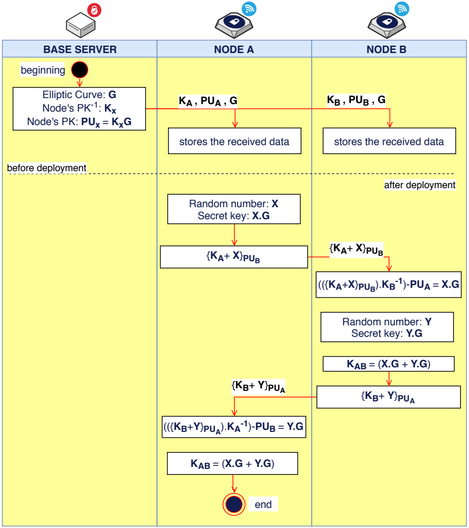
ProVerif

Code Organization



Protocol Formal Verification - Study Case

Saqib, 2016



Protocol Formal Verification - Study Case

Saqib, 2016 - Model Breakdown

Data Types

```
type skey.
type secretkey.
type id.
type pkey.
```

Protocol Formal Verification - Study Case

Saqib, 2016 - Model Breakdown


```
free C1: channel.
free C2: channel.
free CX: channel.
const GP: g [private].
table alreadyPaired(id, id).
```

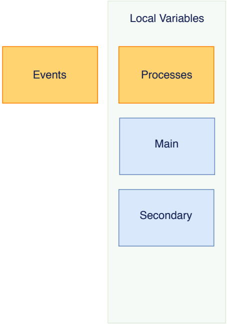
Protocol Formal Verification - Study Case

Saqib, 2016 - Model Breakdown


```
fun encryption(g,pkey): g.
reduc forall KA': skey, X': number, GENPOINT: g, KB': skey, PUA': pkey; decryption(encryption(addsSkeyPlusNumber(KA',X'), calcPublicKey(KB', GENPOINT)), KB', calcPublicKey(KA', GENPOINT)) = calcSecretKey(X', GENPOINT).
fun calcSharedKey(secretkey, secretkey): secretkey.
```

Protocol Formal Verification - Study Case

Saqib, 2016 - Model Breakdown




```
let nodeA() =
new KA: skey; new IDA: id;
let PUA = calcPublicKey(KA, GP) in
insert publicKeysA(IDA, PUA); out(CX, PUA);
!(
    get publicKeysB(IDBR1: id, PURB: pkey) in (
    get alreadyPaired(=IDA, =IDBR1) in (event eNodesAlreadyPairedA(IDA, IDBR1))
    else (
    new X: number;
    let XG = calcSecretKey(X, GP) in
    event eNodeACreatesTheSecretKey(XG);
    let KAXPUB = encryption( addsSkeyPlusNumber(KA, X), PURB) in
    event eSendKAXPUB(KAXPUB);
    out(C1, (KAXPUB, IDA, IDBR1));
    in(C2, (K2:g, IDBR2: id, IDAR: id));
    if ((IDAR = IDA) && (IDBR2 = IDBR1)) then
    let GY = decryption(K2, KA, PURB) in(
    event eDecryptedUsingKA(GY,KA);
    let XGGY = calcSharedKey(XG, GY) in
    insert alreadyPaired(IDA, IDBR1);
    event eNodeAComputesSharedKey(XGGY)))
).
```

Protocol Formal Verification - Study Case

Saqib, 2016 - Model Breakdown


```
process
      (!nodeB() | !nodeA())
```

Protocol Formal Verification - Study Case

Saqib, 2016 - Secrecy Queries

"Unauthorized agents are ***not capable*** *of deriving* ***specific*** *information..."*

Protocol Formal Verification - Study Case

Saqib, 2016 - Secrecy Queries

Queries

Secrecy

Authenticity

```
query secret KAXPUB.
query secret KBYPUA.
query attacker(new KA).
query attacker(new KB).
```

Protocol Formal Verification - Study Case

Saqib, 2016 - Authenticity Queries

Correspondence Assertions	*"if an event **e** has been executed, then **e'** has been previously executed"*

Protocol Formal Verification - Study Case

Saqib, 2016 - Authenticity

Aliveness

Weak Agreement

Non-injective Agreement

Injective Agreement

Protocol Formal Verification - Study Case

Saqib, 2016 - Authenticity Queries

Queries

Secrecy

Authenticity

```
query qXGGY: secretkey, qYGGX: secretkey, qKBYPUA: g, qKAXPUB: g, qKB:
    ↪ skey, qKA: skey, qY: number, qX: number, qGX: secretkey, qGY:
    ↪ secretkey; event(eNodeAComputesSharedKey(qXGGY)) ==> ( (inj-
    ↪ event(eNodeBComputesSharedKey(qYGGX)) ==> inj-event (
    ↪ eSendKAXPUB(qKAXPUB))) ) && qXGGY = calcSharedKey(qGX, qGY) &&
    ↪ qYGGX = calcSharedKey(qGX, qGY) && qGX = calcSecretKey(qX,GP)
    ↪ && qGY = calcSecretKey(qY,GP) && qKBYPUA = encryption(
    ↪ addsSkeyPlusNumber(qKB, qY), calcPublicKey(qKA, GP)) && qKAXPUB
    ↪  = encryption(addsSkeyPlusNumber(qKA, qX), calcPublicKey(qKB,
    ↪ GP)).
```

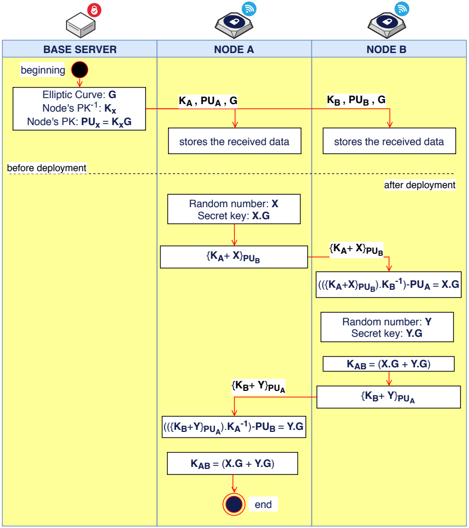
Protocol Formal Verification - Study Case

Saqib, 2016 - Model Breakdown

Did the **symmetric key** calculated by **A** contain the secret key calculated by **B**, *and* did the **symmetric key** calculated by **B** actually contain the secret key sent by **A**?

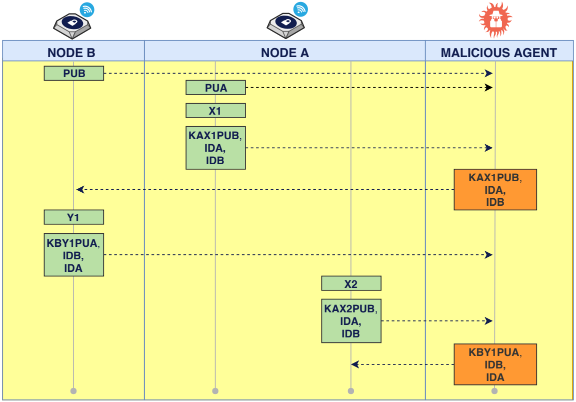
Protocol Formal Verification - Study Case

Saqib, 2016



Protocol Formal Verification - Study Case

Saqib, 2016 - Result Analysis



Conclusion

"Take Home" Messages

- It is **difficult** to design security protocols with **no vulnerabilities**;
- Formal verification techniques **help** on the process of checking if protocols guarantee certain **security properties**;
- ProVerif is one of the available tools to automate formal verification;

Conclusion

"Take Home" Messages

- ProVerif can only verify what the user provides to it.
- Find the perfect balance between the levels of **detail** and **abstraction** for your model;
- The results given by ProVerif should serve as a **tool** to improve the analysis of the protocol.

Questions?