Model Checking the Needham-Schroeder Protocol Using Spin

Deadline: November 6, 2022 before mid-night

1 Introduction

This section explains some basics of security protocols and what they achieve over a public network such as the Internet. Security protocols provide a secure medium for parties to communicate over the Internet. Two parties that intend to communicate securely must first ensure that their partner is the entity with whom they wish to communicate; i.e., *authentication*. In other words, each party must prove (to the other party) who they claim to be. After authentication, there should be a convention between the involved parties to exchange information in a secure fashion so that no other party can gain knowledge about the contents of the secure communication; i.e., *confidentiality*. Security protocols mainly enable mechanisms for authentication and confidentiality over the Internet. *Public key cryptography* is a method for ensuring privacy/confidentiality in the Internet. Each *participant* (a.k.a. *agent or principal*) has a *public key* that everyone knows and a *secret key* that only he/she knows. Anyone who intends to send data to an agent A should use the public key of A, denoted PK_A to encrypt information, and only agent A can decrypt the received message using its secret key, denoted SK_A.

While public key cryptography provides a simple method of secure communication, it is inefficient in terms of the number of times encryption and decryption operations must be performed. To expedite the pace of communication, imagine that two agents use public key encryption only once to establish a secret key SK_{shared} between themselves, and from that point on they use SK_{shared} for communication. This is called *symmetric key* cryptography. More specifically, consider two agents, called Alice and Bob, who would like to communicate securely. They initially use public key cryptography to communicate two random numbers. Without loss of generality (Wlog), let Alice be the initiator. Alice uses PK_{Bob} and encrypts a random number $Nonce_{Alice}$ and sends it to Bob. Then, Bob decrypts the received message using SK_{Bob} and extracts $Nonce_{Alice}$. Bob can then use a similar method to communicate back to Alice by sending $Nonce_{Alice}$ to her, indicating that Bob knows the shared random number, called *nonce*. From that point on they can use $Nonce_{Alice}$ as the shared key and communicate through symmetric key cryptography. That is, they encrypt and decrypt messages using $Nonce_{Alice}$.

Formal Notation. In order to formally express security protocols, we use a set of simple notations. For example, to indicate that Alice sends a message m to Bob, we use the notation 'Alice $\rightarrow$ Bob : m'. The unique identifier of an agent A is represented as ID_A. We use $\{m\}_{PK_A}$ to represent a message that has been encrypted with the public key of an agent A. A random number used by an agent A (e.g., Alice) only once is called a *nonce* (number used once), denoted N_A (e.g., N_{Alice}). For example, if Alice sends a message m along with her nonce to Bob (while encrypting it using Bob's public key), then we can specify this transmission as 'Alice $\rightarrow$ Bob: $\{m, N_{Alice}\}_{PK_{Bob}}$'.

2 The Needham-Schroeder Protocol

The Needham-Schroeder (NS) Protocol is a well-known algorithm for establishing a secure connection between two parties. For brevity, let A denote Alice and B represent Bob. For Alice and Bob to authenticate under the NS protocol, the following message exchanges must be performed.

1. $A \rightarrow B : \{ID_A, N_A\}_{PK_B}$
2. $B \rightarrow A : \{N_A, N_B\}_{PK_A}$
3. $A \rightarrow B : \{N_B\}_{PK_B}$

First, Alice encrypts her ID and her nonce using Bob's public key and sends them to Bob. Bob decrypts the message using his secret key and extracts Alice's ID and nonce. Second, to prove to Alice that Bob has the correct nonce, Bob encrypts Alice's nonce (along with his own nonce) using Alice's public key and sends the message to Alice. Upon its receipt, Alice decrypts the message and now knows that Bob has her nonce. She also receives Bob's nonce. Third, to prove this to Bob, she sends back Bob's nonce. Now, both of them have the private nonces and can construct a shared secret key to be used for symmetric key cryptography.

Note. Notice that, initially only Alice knows her nonce and only Bob knows his nonce. No other party knows the nonces. Then, through encryption/decryption Bob (Alice) gets to know Alice's (Bob's) nonce. This means the only way to know someone else's nonce is through decrypting a message and extracting his/her nonce.

Basic Promela Model. We would like to perform the modeling and verification of the NS protocol in an incremental fashion, where we start from a simple model that contains only Alice and Bob represented as two separate proctypes. We model the *communication network* as a channel shared by Alice and Bob. To keep the state space small, we consider a synchronous channel through which Alice and Bob can communicate. Another abstraction concern is regarding agent IDs, nonces and keys. In real-world systems, these are all multi-bit numbers. However, to keep the state space small and to simplify our model, we model them as 'mtype'. We model encrypted messages as a user-defined data type that includes three fields, all of type ''mtype'. These fields respectively represent a key, and two other pieces of data, each one of them being anything such as nonce or ID. The synchronous channel that models the network can exchange packets that have three fields: an encrypted message (i.e., payload), a message number/type, and the ID of the receiver. Notice that, the NS protocol includes three rendezvous between Alice and Bob. If they can finish these three rendezvous, then we can say that the protocol is correct. You may consider each rendezvous carrying a specific message. This way you would have three types of messages.

Deliverables of Phase 1. Your are expected to perform the following tasks at this step of the project:

1. Develop a Promela model that represents the aforementioned Basic Model. Assume that Alice is the initiator. (**15 points**)
2. Simulate your model and observe the behavior of the model as a message sequence chart. Document what you see. (**10 points**)
3. Specify an LTL property for Alice and Bob, specifying that each one of them eventually reaches the end of its proctype. (**5 points**)
4. Verify that the corresponding LTL properties hold for both Alice and Bob. Explain any counterexample you may see. (**10 points**)
5. Does weak fairness impact the result of verification? (**10 points**)

The Intruder Model. In the next step of this project, we would like to study the impact of an intruder on the authenticity and confidentiality of the NS protocol. The intruder has its own public key, ID and nonce. The intruder can non-deterministically intercept a packet and extract its three fields. Note that, in this model the intruder can only distinguish what the different fields of an encrypted message are and store them; it cannot decrypt the message components and cannot know the nonces of Alice and Bob. Then, it can non-deterministically either forward the intercepted packet to an arbitrarily picked agent, or put together a totally new packet (using the stored values) and send it to an arbitrarily selected agent. Notice that, before sending a message out, the intruder randomly decides which message type (out of the three types in the NS protocol) it would like to send out.

Deliverables of Phase 2. Your are expected to perform the following tasks in this step of the project:

1. Add a new proctype to your Promela model that captures the behaviors of the intruder. (**20 points**)
2. Simulate your model and observe the behavior of the model as a message sequence chart. Document what you see. (**10 points**)
3. Verify whether or not the corresponding LTL properties hold for both Alice and Bob. Explain any counterexample you may see. (**10 points**)
4. Does weak fairness impact the result of verification? (**10 points**)