

The Needham-Schroeder Public-Key Protocol –How to Break It

Version 1.0

Jens Chr. Godskesen
IT University of Copenhagen

©2005 Jens Chr. Godskesen

March 10, 2005

1 Introduction

In this note we model the Needham-Schroeder Public-Key protocol using CCS and subsequently we analyze the model using MWB. We show, following the strategy in [6], that the protocol is insecure in that an intruder can impersonate another user of the protocol.

2 What is a security protocol?

A *protocol* in our context is a set of rules of how to communicate in order to obtain a certain goal. For instance, HTTP (the Hypertext Transfer Protocol) is a protocol for communicating hypertext over the Internet.

If the goal is to enforce security the communicating protocols are often denoted security protocols. As an example HTTPS (the Secure Hypertext Transfer Protocol) is a communication protocol designed to transfer encrypted information between computers over the Internet. HTTPS is based on the secure socket layer protocol SSL developed as a web based e-commerce security protocol that is now named Transport Layer Security (TLS) [2]

In this note we study a particularly famous security protocol: the Needham-Schroeder Public-Key Protocol (NSPK) [9]. It belongs to the family of *authentication* protocols, a specific branch of security protocols where the goal is to establish assurance of the identity between two communicating parties, making sure that they are actually talking to each other.

The protocol NSPK is notoriously famous because it actually appeared and was in use for about 17 years before it was eventually broken [6]. For instance the client/server network authentication protocol Kerberos [5] is based on NSPK and Kerberos is used in numerous commercial products. To illustrate the subtleness of the attack breaking the protocol it was incorrectly proven correct in 1989 [1].

3 The Needham-Schroeder Public-Key Protocol

As mentioned above the purpose of NSPK is to establish mutual authentication between two parties. In order to do so the protocol uses *public key encryption* [3].

Encryption is used to protect a message m from being read by others it may be encrypted using a parametric key K in which case we write:

$$\{m\}_K$$

If the key is *symmetric* we may decrypt with the same key, hence

$$m = \{\{m\}_K\}_K$$

Instead of any two parties sharing a common symmetric key it may be advantageous to work with *asymmetric* keys.

A pair of asymmetric keys consists of a *private* and a *public* key, K^- and K^+ respectively. The two keys are each others inverses, ie.

$$m = \{\{m\}_{K^-}\}_{K^+} = \{\{m\}_{K^+}\}_{K^-}$$

The idea is that the owner of a private key distributes its inverse public key to all interesting parties. Then any message encrypted by the public key can only be decrypted by the owner holding the private key (and the private key can be used by its owner to sign messages).

In this note we study a simplified version of NSPK. The version studied is similar to the one presented in [6]. The protocol can be described as follows using standard notation for security protocol definition:

$$\begin{aligned} Alice \longrightarrow Bob & : \{n_A, K_A^+\}_{K_B^+} \\ Bob \longrightarrow Alice & : \{n_A, n_B\}_{K_A^+} \\ Alice \longrightarrow Bob & : \{n_B\}_{K_B^+} \end{aligned}$$

Intuitively the meaning of the protocol is: Alice is the *initiator* of the protocol, she wants to establish a session with the *responder* Bob. Alice starts sending a message to Bob, the message is encrypted by Bob's public key that Alice somehow has obtained. Hence the message can only be decrypted by Bob. The message contains a randomly generated *nonce*, n_A , for this session of the protocol only, and also it contains the public key of Alice.

When Bob receives the message he decrypts it using his private key and obtains the nonce n_A and Alice's public key K_A^+ . Bob then returns n_A together with a new randomly generated nonce n_B to Alice all encrypted using Alice's public key.

When Alice receives the second message she may be assured that the message came from Bob, since only he could decrypt the first message and hence obtain the nonce n_A that is now returned to Alice. Alice then returns to Bob the nonce n_B encrypted with Bob's public key.

When Bob receives the third message he may be convinced that the message came from Alice since only she could be able to obtain the returned nonce n_B .

It then seems that Alice and Bob are confident to be talking to each other. So why is the protocol incorrect? We'll see.

4 Modelling NSPK

We now model the NSPK protocol in CCS, or to be more precise we model it using an extension of CCS which allows for receiving and sending values.

The idea is to extend CCS by letting output actions now become on the form:

$$'act\langle a_1, a_2, \dots, a_k \rangle . P$$

intuitively meaning that along the channel act the values a_1, a_2, \dots, a_k can be sent after which the process behaves as P .

Dually CCS is extended with input actions on the form:

$$act(x_1, x_2, \dots, x_k) . Q$$

Intuitively $act(x_1, x_2, \dots, x_k)$ is a binder of the free occurrences of the variables x_1, x_2, \dots, x_k in Q . Hence, when the process on channel act receives k values, say a_1, a_2, \dots, a_k , then it becomes

$$Q[a_1/x_1, a_2/x_2, \dots, a_k/x_k],$$

i.e. it becomes Q where all the variables x_1, x_2, \dots, x_k are substituted by a_1, a_2, \dots, a_k respectively.

That is, if we take the parallel composition

$$'act\langle a_1, a_2, \dots, a_k \rangle . P \mid act(x_1, x_2, \dots, x_k) . Q$$

then there is an internal transition to

$$P \mid Q[a_1/x_1, a_2/x_2, \dots, a_k/x_k]$$

Actually, what above has been referred to as values or variables are formally both just names and name binders respectively, but we shall not go into a detailed discussion about that here. The interested reader is referred to literature about the π -calculus, see e.g. [7, 8].

4.1 Initiator and Responder

Using the extension of CCS presented above we may model the initiator Alice and the responder Bob by:

```
agent Alice(a, kB, kA, i, c) =
    (^ n) ('i. 'a<kB, n, kA>. a(x, y, z). [x=kA][y=n]'c. 'a<kB, z>. 0)
agent Bob(a, kB, r, c) =
    (^ n) (a(x, y, z). [x=kB]'r. 'a<z, y, n>. a(x, y). [x=kB][y=n]'c. 0)
agent Sys(iAB, rBA, cAB, cBA) = (^ a, kA, kB) ( Alice<a, kB, kA, iAB, cAB>
    | Bob<a, kB, rBA, cBA> )
```

The idea is that Alice and Bob communicate over their shared channel a and each message send over a is supposed to contain some parameters.

First Alice signals by i that she is about to start initiating a session run with Bob. Then she send along a the message $\langle kB, n, kA \rangle$ meaning kB , the public key of Bob, and that it contains a new nonce n and the public key of Alice. Notice that n is restricted so no one else but Alice knows about it yet.

Then Alice expects to receive three values on a . The first is checked to be Alice's public key, meaning that the message is encrypted by that key. The next value is checked to be the nonce she previously send to Bob. If both checks succeed then she signals by c that the run of the session is completed and the third value received is send to Bob encrypted under his public key.

The encoding of Bob follows the same ideas as the encoding of Alice. First he receives three values, the first telling as to whether the message is encrypted using his public key. If that is the case Bob signals by r to be running a session and then he returns y and a new n both encrypted using z . If Bob then receives n encrypted using his own public key he signals to have completed the protocol run succesfully by c .

In the parallel composition we are only interested in observing the events signalling the initiation, running and completion of the protocl.

If we step through the (deterministic) run of the execution of the system of Alice and Bob we get (omitting some of the details):

```

MWB>step Sys(iAB,rBA,cAB,cBA)
* Valid responses are:
  a number N >= 0 to select the Nth commitment,
  <CR> to select commitment 0,
  q to quit.
0: |>'iAB. (^~v,~v6,~v7)((^~v8)'~v<~v7,~v8,~v6>.~v(x,y,z).[x=~v6]... (A)
Step>0
0: |>t. (^~v,~v6,~v7,~v8)(~v(x,y,z).[x=~v6][y=~v8]'cAB.'~v<~v7,z>.0 ... (B)
Step>0
0: |>'rBA. (^~v,~v6,~v7,~v8)(~v(x,y,z).[x=~v6][y=~v8]'cAB.'~v<~v7,z>.0 ...
Step>0
0: |>t. (^~v,~v6,~v7)'cAB.'~v<~v6,~v7>.0 | ~v(x,y).[x=~v6][y=~v7]'cBA.0)
Step>0
0: |>'cAB. (^~v,~v6,~v7)'~v<~v6,~v7>.0 | ~v(x,y).[x=~v6][y=~v7]'cBA.0)
Step>0
0: |>t.'cBA.0
Step>0
0: |>'cBA.0
Step>0
No commitments for 0
Quitting.

```

We indeed observe that Alice starts the protocol with Bob, that Bob engages in a run with Alice, and finally that first as expected Alice commits to be running a session with Bob and thereafter that Bob commits to be engaged in a session run with Alice.

Notice the restricted name $v8$ in the line marked by A above, it is the MWB internal representation of Alice's nonce n . When the nonce is send to Bob he becomes aware of it and that is why $v8$ in the linie marked by B belongs to the outer most restriction. The restriction that first belonged to Alice is said to be *extruded* to contain also Bob. As an example, when a locally restricted name is send to another process as e.g. in

$$(\hat{a})((\hat{n})(a \langle n \rangle . P \mid a(x) . Q))$$

then the internal transition results in

$$(\hat{a}, n)(P \mid Q[n/x])$$

5 Adding more parties

Clearly there may be more than two parties involved in running the NSPK protocol and it may be that some party may both take the role as initiator and responder. We therefore add to the specification from above a new agent Eve which may act both as initiator and responder, and in so doing we introduce the general agents of an initiator and a respond.

```

agent Initiator(a,kX,kY,i,c) =
    (^ n)('i.'a<kY,n,kX>.a(x,y,z).[x=kX][y=n]'c.'a<kY,z>.0)
agent Responder(a,k,r,c) =
    (^ n)(a(x,y,z).[x=k]'r.'a<z,y,n>.a(x,y).[x=k][y=n]'c.0)
agent Alice(ab,ae,kA,kB,kE,iAB,iAE,cAB,cAE) =
    Initiator<ab,kA,kB,iAB,cAB> + Initiator<ae,kA,kE,iAE,cAE>
agent Bob(ab,eb,kB,rBA,rBE,cBA,cBE) =
    Responder(ab,kB,rBA,cBA) + Responder(eb,kB,rBE,cBE)
agent Eve(ae,eb,kE,kB,iEB,rEA,cEB,cEA) =
    Initiator<eb,kE,kB,iEB,cEB> + Responder(ae,kE,rEA,cEA)
agent Sys(iAB,iAE,iEB,rBA,rBE,rEA,cAB,cBA,cAE,cEA,cBE,cEB) =
    (^ ab,ae,eb,kA,kB,kE)( Alice<ab,ae,kA,kB,kE,iAB,iAE,cAB,cAE>
        | Bob<ab,eb,kB,rBA,rBE,cBA,cBE>
        | Eve<ae,eb,kE,kB,iEB,rEA,cEB,cEA>)

```

An *Initiator* with public key kX initiates a session with a responder on channel a , this party is assumed to have public key kY . Encrypted with this key it sends along a fresh nonce n and its own public key. If in response on the same channel it receives encrypted with its own public key the nonce n and another nonce z the successful completion of the authentication is signalled and z is returned encrypted by kY .

If a *Responder* on channel a receives a message with nonce y and key z encrypted with its public key k then it signals to be running a session with the party with whom it shares the communication channel. On the channel it returns a new fresh nonce n together with the nonce y it just received, all encrypted with the key z . If it then on the same channel receives its nonce back encrypted by its public key then successful completion of the authentication is signalled.

The involved parties Alice, Bob, and Eve are then defined using the general agents taking care that they pairwise communicate on distinct channels.

Notice although that the roles of the parties are either or, i.e. Alice initiates with either Bob or Eve, Bob responds to either Alice or Eve, and Eve either responds to Alice or initiates with Bob.

Exercise How to model that parties can take up several roles simultaneously? Will that cause problems?

6 The Attack

When several parties are running the protocol we don't know a priori that they are all honest. Within the security protocol community a dishonest party is often referred to as an *intruder*.

In [6] they show a so called "man in the middle attack" on the NSPK protocol where Alice initiates a run with one party that appears to be an intruder. The intruder then initiates a run with Bob in a tricky fashion such that Bob eventually believes he is running a session with Alice.

Letting Eve now be the intruder the attack goes as follows:

$$\begin{aligned}
 Alice &\longrightarrow Eve && : \{n_A, K_A^+\}_{K_E^+} \\
 Eve(Alice) &\longrightarrow Bob && : \{n_A, K_A^+\}_{K_B^+} \\
 Bob &\longrightarrow Eve(Alice) && : \{n_A, n_B\}_{K_A^+} \\
 Eve &\longrightarrow Alice && : \{n_A, n_B\}_{K_A^+} \\
 Alice &\longrightarrow Eve && : \{n_B\}_{K_E^+} \\
 Eve &\longrightarrow Bob && : \{n_B\}_{K_B^+}
 \end{aligned}$$

First Alice following the rules of the protocol initiates a run with Eve. Eve then decrypts the messages and impersonating Alice forwards it to Bob encrypted using Bob's public key. Bob then decrypts the message, finds the public key of Alice and therefore believes the messages came from Alice¹. According to the protocol Bob then returns Alice's nonce together with a new nonce both encrypted by the public key of Alice to the intruder Eve which he mistakenly believes is Alice. Eve cannot decrypt the message returned from Bob, but she can simply forward it to Alice. Alice then decrypts the message and finds a nonce she believes belongs to Eve, according to the protocol she returns the nonce to Eve encrypted by Eve's public key. Upon reception of that message Eve decrypts the message, retrieves Bob's nonce which then is sent to Bob making him believe he is talking to Alice, since only she was supposed to be able to read Bob's nonce.

Notice how the attack is an interleaving of two runs of the NSPK protocol.

Exercise Explain the two interleavings of the NSPK protocol constituting the attack.

7 Modelling the intruder

We next provide a model containing the interference shown above, it is presented along the lines used by the model in Section 5 by devising a special purpose agent for the intruder Eve, naturally Eve is neither a genuine initiator or responder.

¹We assume the public key also can be used for identification of its owner, otherwise the message simply could contain an identification (say IP address) from the sender.

```

agent Initiator(a,kX,kY,i,c) =
  (^ n)('i.'a<kY,n,kX>.a(x,y,z).[x=kX][y=n]'c.'a<kY,z>.0)
agent Responder(a,k,r,c) =
  (^ n)(a(x,y,z).[x=k]'r.'a<z,y,n>.a(x,y).[x=k][y=n]'c.0)
agent Alice(ae,kA,kE,iAE,cAE) = Initiator(ae,kA,kE,iAE,cAE)
agent Bob(eb,kB,rBA,cBA) = Responder(eb,kB,rBA,cBA)
agent Eve(ae,eb,kE,kB) =
  ae(x,y,z).[x=kE]'eb<kB,y,z>.eb(x,y,z).
  'ae<x,y,z>.ae(x,y).[x=kE]'eb<kB,y>.0
agent Sys(iAE,rBA,cBA,cAE) =
  (^ ae,eb,kA,kB,kE)( Alice<ae,kA,kE,iAE,cAE>
    | Bob<eb,kB,rBA,cBA>
    | Eve<ae,eb,kE,kB>)

```

The attack shown above is indeed possible in that the system satisfies the following HML formula:

```

MWB>prove Sys(iAE,rBA,cBA,cAE)
  <'iAE><t><t><'rBA><t><t><'cAE><t><t><'cBA>TT
Model Prover says: YES!
(21 inferences)

```

8 How to make a general intruder?

Above we explicitly constructed an intruder for a specific known attack on the NSPK protocol. In principle for a new protocol we don't know beforehand whether there is an attack on the protocol and if there is we certainly don't know how a successful attack could be established.

Instead of inventing a number of potential attacks on a given protocol Dolev and Yao put forward in their seminal paper [4] the approach to define certain realistic capabilities that any intruder may have. For instance, an intruder may be beyond initiating and responding to runs of the protocol:

- listen to any message being part of a protocol run,
- replay any message seen possibly changing unencrypted parts of the message,
- steal messages and hide them from other parties,
- decrypt messages encrypted by the keys he knows, and thereby probably learn new nonces and keys, and
- send messages based on the knowledge (of nonces and keys) obtained so far.

Then based upon these capabilities a general intruder for a protocol could be defined, i.e. general in the sense that any possible attack on the protocol should be somehow composed of the intruders capabilities.

This now leaves us with the question: How to define in our extension of CCS a general intruder that could find the attack himself?

First we may assume that all parties share and communicate over the same channel, say a . By convenience, to make the model below simpler and more compact, we let a channel always take a fixed number of parameters so in the model below a always takes three parameters, hence when only two parameters are needed we duplicate the second parameter.

To make the model even simpler we assume that the intruder doesn't discard messages.

Below is the model of the general specifications of an initiator and a responder:

```

agent Initiator(a,kX,kY,i,ic) =
  (^ n)('i.'a<kY,n,kX>.Initiator1<a,kX,kY,i,ic,n>)
agent Initiator1(a,kX,kY,i,ic,n) =
  a(x,y,z).([x=kX]([y=n]'ic.'a<kY,z,z>.0
    + 'a<x,y,z>.Initiator1<a,kX,kY,i,ic,n>)
    + 'a<x,y,z>.Initiator1<a,kX,kY,i,ic,n>)

agent Responder(a,k,k1,k2,r1,r2,c1,c2) =
  a(x,y,z).([x=k]Responder1<a,k,k1,k2,r1,r2,c1,c2,y,z>
    + 'a<x,y,z>.Responder<a,k,k1,k2,r1,r2,c1,c2>)
agent Responder1(a,k,k1,k2,r1,r2,c1,c2,y,z) =
  [z=k1]'r1.Responder2<a,k,c1,y,z>
  + [z=k2]'r2.Responder2<a,k,c2,y,z>
agent Responder2(a,k,c,y,z) = (^ n)('a<z,y,n>.Responder3<a,k,c,n>)
agent Responder3(a,k,c,n) =
  a(x,y,z).([x=k]([y=n]'c.0 + 'a<x,y,z>.Responder3<a,k,c,n>)
    + 'a<x,y,z>.Responder3<a,k,c,n>)

```

and next we define the actual system composed of Alice, Bob, and Eve:

```

agent Alice(a,kA,kB,kE,iAB,iAE,icAB,icAE) =
  Initiator<a,kA,kB,iAB,icAB> + Initiator<a,kA,kE,iAE,icAE>
agent Bob(a,kB,kA,kE,rAB,rEB,cAB,cEB) =
  Responder(a,kB,kA,kE,rAB,rEB,cAB,cEB)
agent Eve(a,kE,kA,kB) = (a(x,y,z).([x=kE]Eve1<a,kE,kA,kB,y,z>
  + 'a<x,y,z>.Eve<a,kE,kA,kB>)
  + (^ n)('a<kB,n,kE>.Eve<a,kE,kA,kB>))
agent Eve1(a,kE,kA,kB,n,k) = 'a<kA,n,k>.Eve<a,kE,kA,kB>
  + 'a<kB,n,k>.Eve<a,kE,kA,kB>

agent Sys(iAB,iAE,icAB,icAE,rAB,rEB,cAB,cEB) =
  (^ a,kA,kB,kE) ( Alice<a,kA,kB,kE,iAB,iAE,icAB,icAE>
    | Bob<a,kB,kA,kE,rAB,rEB,cAB,cEB>
    | Eve<a,kE,kA,kB> )

```

The initiator declaration is almost as before, however now because all parties share a common channel we must also take care to return any message on the channel that is not encrypted under the key of the initiator and likewise we return a message if the second parameter does not match the expected nonce. Notice, that the initiator has now a repetitive behaviour.

The model of the responder also returns messages that are not encrypted by its public key. Moreover, the responder makes a match on the received key (the third element in

the first message received) in order to find out who (supposedly) send the message ² and thereby making the proper signalling of responding and committing respectively to a session run. In the previous model such a check wasn't needed because there was a unique channel between any two communicating parties. The second message received is checked for mismatch and returned if one is found. Like the initiator also the responder has repetitive behaviour.

The intruder Eve is allowed to receive any message encrypted by her own public key. If that happens she may send the contents of the message to any other party, i.e. in this case either Alice or Bob. If the message is not encrypted by Eve's public key it is returned on the common channel. Eve may also initiate a protocol run by sending a first message of the protocol to any receiver, i.e. in this case only Bob. Eve, as the two processes above, can continue its behavior indefinitely.

To demonstrate the appropriateness of this general model we can check that the attack shown above is still possible:

```
MWB>prove Sys(iAB,iAE,icAB,icAE,rAB,rEB,cAB,cEB)
          <'iAE><t><t><'rAB><t><t><'icAE><t><t><'cAB>TT
Model Prover says: YES!
(83 inferences)
```

Exercise One of the internal transitions in the attack demonstrated by the attack above can be eliminated. Show and explain the property corresponding to that attack.

Of course it is satisfactory that the known attack can be carried out in the general model also, but for a new protocol as touched upon above we don't know a priori what exact sequence of behaviour reveals an attack. Therefore we must do something more cleverly, say for instance define a more general property that is satisfied by models if there is an attack on the protocol it models. Also, since the property above reveals an attack it should be the case that if a model satisfy that property then it would also satisfy our more general property.

Intuitively we would say that there has been an attack on the NSPK protocol if there is an execution of the protocol where a receiver B ends committing to be running a session with an initiator A without ever A prior in that same execution has committed to be running a session with B.

We may write that property as a kind of weak liveness property where eventually the signalling of A committing to be running a session with B, i.e. $\langle 'cAB \rangle TT$, holds but only along a path where no event $'icAB$, i.e. is carried out.

In MWB we may check for the general model above that:

²It is a feature of the model that the third element in the first message is always a key so if the responder is cleverly defined to check against the values of keys from all other parties it for sure will find a match and therefore never has to return the message on the channel due to a mismatch on the third element of a received message.

```

MWB>prove Sys(iAB,iAE,icAB,icAE,rAB,rEB,cAB,cEB)
      mu X.( <'cAB>TT |
              ( <t>X | <'iAB>X | <'iAE>X | <'icAE>X
                | <'rAB>X | <'rEB>X | <'cAB>X | <'cEB>X ))
Model Prover says: YES!
(31483 inferences)

```

revealing that there indeed is a possible attack on the NSPK protocol as it has been modelled above.³

Notice, in the property above that the event `'icAB` is not allowed to be used for ensuring progress, this is precisely the means to ensure that Alice never along that run has committed to be running a session with Bob.

Exercise How can the property above be simplified?

Exercise Explain why any model satisfying

$$\langle 'iAE \rangle \langle t \rangle \langle t \rangle \langle 'rAB \rangle \langle t \rangle \langle t \rangle \langle 'icAE \rangle \langle t \rangle \langle t \rangle \langle 'cAB \rangle TT$$

also satisfy the property above.

9 A Corrected Version of NSPK

Having focused so much on the failure of the NSPK protocol, it would of course be of great interest to improve it and make it correct. In [6] they devise a remedy correcting the protocol from the attack they show. The corrected version of the NSPK protocol they define by:

$$\begin{aligned}
 Alice \longrightarrow Bob & : \{n_A, K_A^+\}_{K_B^+} \\
 Bob \longrightarrow Alice & : \{n_A, n_B, K_B^+\}_{K_A^+} \\
 Alice \longrightarrow Bob & : \{n_B\}_{K_B^+}
 \end{aligned}$$

Exercise Explain why the attack found above cannot be carried out any longer.

10 Exercises

1. Do the exercises above
2. Modify the general model of the intruder such that it is being allowed to intercept messages. Does this influence on the attack on the NSPK protocol? Check using MWB.

³Warning, it may take a while to run this check.

3. Modify the general model of the intruder such that it is allowed to act also as initiator and receiver. Does this influence on the attack on the NSPK protocol? Check using MWB.
4. How to slightly change the weak liveness property above to define that there is an execution of the protocol where Bob ends committing to be running a session with Alice without ever Alice prior in that same execution has nor initiated and neither committed to be running a session with Bob.
5. Can you come up with other formulas describing possible attack on the NSPK protocol?
6. Model the corrected version of the NSPK protocol and show that it doesn't satisfy the formulas for attacks.

References

- [1] Michael Burrows, Martín Abadi, and Roger M. Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, 1990.
- [2] T. Dierks and C. Allen. The tls protocol, version 1.0. Technical Report RFC 2246, IETF, January 1999.
- [3] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transaction on Information Theory*, 22:644–654, 1976.
- [4] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Trans. on Information and Theory*, 29(2):198–208, 1983.
- [5] J. Kohl and C. Neumann. The kerberos network authentication service (v5). Technical Report RFC 1510, IETF, January 1993.
- [6] G. Lowe. Breaking and fixing the needham-schroeder public-key protocol using *fd*. *Software - Concepts and Tools*, 17:93–102, '1996.
- [7] Robin Milner. *Communicating and Mobile Systems: The π -Calculus*. Cambridge University Press, 1999.
- [8] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part I/II. 100:1–77, September 1992.
- [9] Roger Needham and Michael Schroeder. Using encryption for authentication in large networks of computers. *Communication of the ACM*, 21(2):120–126, '1978.