# Analysis of a Security Protocol in $\mu$CRL

Jun Pang

Centrum voor Wiskunde en Informatica
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
Jun.Pang@cwi.nl

**Abstract.** In this paper, we present how the process-algebraic language $\mu$CRL can be used to specify security protocols and discuss the analysis process using the $\mu$CRL toolset and CADP. To illustrate the feasibility of our approach, we analyzed the Needham-Schroeder public-key protocol and reproduced the error found by Gavin Lowe [7]. Two more definitions of authentication are also studied. We give some remarks on our approach and discuss some possible directions for future work.

## 1 Introduction

The security of communication between computers has become a hot research issue. A variety of security protocols based on cryptographic primitives are used to establish secure communication over insecure open networks. Unfortunately, security protocols often contain serious errors. Formal methods are mathematically based techniques for specifying and verifying software and hardware systems. Their mathematical underpinning allows formal methods to analyze systems in a more precise and non-ambiguous fashion. This makes it possible to use formal description and verification to obtain assurance that a protocol cannot be attacked by an intruder.

$\mu$CRL provide a notation for the specification and analysis of distributed systems in an algebraic fashion. This language extends the process algebra ACP [5] with equational *abstract data types*. The $\mu$CRL toolset [1], together with the CÆSAR ALDÉBARAN DEVELOPMENT PACKAGE (CADP) [4], which acts as a back-end for the $\mu$CRL toolset, features visualization, simulation, state space generation, model checking, theorem proving and state bit hashing capabilities. It has been successfully applied in the analysis of a wide range of protocols and distributed systems (see `http://www.cwi.nl/~mcrl/`).

In this paper, we present how the process-algebraic language $\mu$CRL can be used to specify security protocols and discuss the analysis process. The behavior of agents of a security protocol can be modeled by processes. The abstract data types in $\mu$CRL can be used to abstract from the complex cryptographic primitives and to model the knowledge databases of agents. We define some *security actions* to indicate the critical points in the protocol and these *security actions* can contain the information relevant to the properties we want to verify. To illustrate the feasibility of our approach, we analyzed the Needham-Schroeder public-key protocol, and reproduced a known error. Our approach resembles the

method used by Leduc and Germeau [6]. The $\mu$CRL toolset was used to generate the state spaces from specifications, and CADP was used to model check the requirements.

**Related work** FDR, a model checker for CSP, is used to analyze the Needham-Schroeder public-key protocol in [7]. The agents of the protocol and an intruder are modeled as processes. FDR takes an implementation and a specification of the protocol as input, and checks whether the implementation refines the specification. A security error was discovered. They adapted the protocol to remove this error and detected no further attacks.

In [10], a finite state exploration tool, Mur$\phi$, is used to analyze several security protocols. A process is modeled by a set of related rules. The parallel composition of two processes is modeled by a simple union of the rules of the two processes. The correctness properties can be specified as invariants in Mur$\phi$. The Needham-Schroeder public-key protocol was analyzed. Mur$\phi$ was able to reproduce the error described in [7].

How LOTOS can be used to specify security protocols is presented in [6]. Security properties can be modeled as safety properties and checked automatically by a model-based verification tool. This technique is illustrated on a concrete registration protocol. An error is found and corrected.

## 2 The Needham-Schroeder public-key protocol

The Needham-Schroeder public-key protocol [11] aims to provide mutual authentication between an initiator $A$ and a responder $B$, after which some session involving the exchange of messages can take place. Both the initiator and the responder want to be assured of the identity of the other. The formal presentation of this protocol can be decomposed into several parts. We present a simplified form of the protocol, which can be described in three steps: In step 1, the initiator $A$ seeks to establish a connection with the responder $B$ by selecting a nonce $N_a$, and sending it along with its identity to $B$, both encrypted with $B$'s public key $K_b$. When $B$ receives this message, it decrypts the message to obtain the knowledge of $N_a$. It then returns the nonce $N_a$ with a new nonce $N_b$ to $A$. Both nonces are encrypted with $A$'s public key. When $A$ receives this message, it decrypts it and concludes that it is talking to $B$, since only $B$ should be able to decrypt $A$'s initial message containing nonce $N_a$; $B$ is authenticated. $A$ returns the nonce $N_b$ to $B$, encrypted with $B$'s public key. In the same fashion, $A$ is authenticated after step 3.

## 3 Analysis process

Several approaches have been developed for analyzing security protocols. We take the *explicit intruder method* [3] as the basis of our analysis approach and adopt it with $\mu$CRL and its toolset. The whole process may have this sequence of

steps: specify the protocol; model the intruder; state the correctness properties and verify the protocol.

### 3.1 Algebraic specification

Each $\mu$CRL specification has two parts, one defines the data types and the other gives the specification of behavior. The data part expresses which kinds of data are used and the operations on them. By this, we can abstract away the complex cryptographic primitives, such as encryption and decryption. In $\mu$CRL, we model the knowledge database for agents as a set.

To verify the correctness of this protocol, normally we need to put the agents into a hostile environment by adding an intruder into the protocol. In $\mu$CRL, we can model an intruder as a process which can mimic attacks of a real-world intruder. We refer to a general set of modeling assumptions with wider applicability as the *Dolev-Yao model* [2]. The responder and initiator are linked together by communication channels. These channels are insecure, meaning that they can be eavesdropped by an intruder. Both of the responder and initiator are defined as a $\mu$CRL process. Due to space limitation, the specification in $\mu$CRL can be found in the full version of this paper [12].

### 3.2 Stating the correctness properties

Model checking is an automatic technique to determine whether a state transition system satisfies certain requirements. A requirement should be expressed as a temporal logic formula first. A model checker searches the reachable states of a labeled transition system to determine whether this formula holds. The temporal logic used by Evaluator[1] is called *regular alternation-free $\mu$-calculus*. The syntax of this logic is given in [9].

Similar to what Leduc and Germeau did [6], we need to define some *security actions* (e.g. actions *I_running*, *I_commit*) for agents to determine the critical points in the specification. These actions can be parameterized with any data relevant to the properties we want to check. The parameters of these actions play an important role when model checking the properties. By this, we can abstract away from the details of communication and only focus on these actions. The correctness requirements of a security protocol are always related to authentication and rely on the fact that the intruder does not know some secret. In this paper, we study three kinds of authentication, and show that the Needham-Schroeder public-key protocol and its repaired version can or cannot guarantee secure communication at different level. The fact that the intruder does not know some secret can be characterized as safety properties.

### 3.3 Requirements of the protocol

We catch three kinds of definition for authentication from [8] and formulate them in the *regular alternation-free $\mu$-calculus*.

[1] A model checker among the CÆSAR ALDÉBARAN DEVELOPMENT PACKAGE.

The mostly investigated definition of authentication is: whenever $A$ completes a run apparently with $B$, then $B$ has recently been running the protocol apparently with $A$.

A3: [(¬ R_running(A,B))* · I_commit(A,B)] False

A3 claims that the responder is correctly authenticated. It says that if an execution sequence does not contain an action *R_running*($A$,$B$), then in the resulting state an initiator $A$ cannot believe that it is talking with a responder $B$.

A4: [(¬ I_running(A,B))* · R_commit(A,B)] False

In the same way, A4 claims that the initiator is correctly authenticated.

A stronger definition insists not only that the two agents agree on each other's state, but also a one-one relationship, meaning that the two agents agree upon all data values used in the run.

A1: [(¬ I_commit(A,B,$N_a$,$N_b$))* · I_commit(A,B,$N_a$,$N_b$)]
$\mu$ X · (<T>T ∧ [¬ R_commit(A,B,$N_a$,$N_b$)] X)

It states that after an action *I_commit*($A$,$B$,$N_a$,$N_b$), the reachability of an action *R_commit*($A$,$B$,$N_a$,$N_b$) is inevitable.

A2: [(¬ I_commit(A,B,$N_a$,$N_b$))* · R_commit(A,B,$N_a$,$N_b$)] False

A2 claims that the initiator is correctly authenticated.

A weaker definition is: whenever an agent $A$ completes a run apparently with $B$, then agent $B$ has recently been running the protocol. Note that agent $B$ may run the protocol with some other agent and never have heard of $A$.

A5: [(¬ R_running(B))* · I_commit(A,B)] False
A6: [(¬ I_running(A))* · R_commit(A,B)] False

### 3.4 Verification results

In our verification, we were able to discover the protocol error described in [7]. Property A3 was proved as *false* by the model checker. After fixing the protocol as Gavin Lowe did in [7], the repaired protocol was shown to satisfy the properties A3 and A4. Table 1 summarizes the verification result on all the properties listed in Section 3.3. It shows that both the protocol and its repaired version can satisfy weaker authentication (A5 and A6). The protocol satisfies neither A1 nor A2, while the repaired version satisfies A2 but not A1.

## 4 Conclusion and future work

This paper presents a formal approach to analyze security protocols in $\mu$CRL. We took the Needham-Schroeder public-key protocol as a case study. The known error can be reproduced by model checking. Furthermore, we also can study another two definitions of authentication.

Compared with works using general purpose verification methods to analyze security protocols, our approach has achieved some success. Encouraged by the work presented in this paper, we can list some further directions of research.

| | A1 | A2 | A3 | A4 | A5 | A6 |
|---|---|---|---|---|---|---|
| NS-PKP | False | False | False | True | True | True |
| Repaired NS-PKP | False | True | True | True | True | True |

**Table 1.** Verification result on the properties

1. Try to analyze other security protocols, and hope to discover new errors;
2. Apply our approach to more complicated e-commerce protocols, where security plays an important role, e.g. the electronic payment protocols;
3. Combine the approach with the design of new security protocols;
4. Applying techniques from process algebra and theorem proving for the formal analysis of security protocols.

## References


1. S.C.C. Blom, W.J. Fokkink, J.F. Groote, I.Z. van Langevelde, B. Lisser and J.C. van de Pol. $\mu$CRL: A toolset for analysing algebraic specification. In *Proc. CAV'2001*, *LNCS* 2102, Springer-Verlag, pp. 250–254.
2. D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory 29(2)* (1983).
3. N. Durgin and J. Mitchell. Analysis of security protocols. In *Calculational System Design* (1999), ISO Press, pp. 369–395.
4. J.-C. Fernandez, H. Garavel, A. Kerbrat, L. Mounier, R. Mateescu and M. Sighireanu. CADP – a protocol validation and verification toolbox. In *Proc. CAV'1997*, *LNCS* 1102, Springer-Verlag, pp. 437–440.
5. W.J. Fokkink. *Introduction to Process Algebra.* Texts in Theoretical Computer Science. Springer-Verlag, 2000.
6. G. Leduc and F. Germeau. Verification of security protocols using Lotos-method and application. *Computer Communications 23* (2000), 1089–1103.
7. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR. In *2nd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems* (1996), Springer-Verlag, pp. 147–166.
8. G. Lowe. Some new attacks upon security protocols. In *9th IEEE Computer Security Foundations Workshop* (1996), IEEE Press, pp. 162–169.
9. R. Mateescu and M. Sighireanu. Efficient on-the-fly model-cheching for regular alternation-free mu-calculus. In *Proc. FMICS'2000*, pp. 65–86.
10. J. Mitchell, M. Mitchell and U. Stern. Automated analysis of cryptographic protocols using Mur$\phi$. In *Proc. of IEEE Symposium on Security and Privacy* (1997), pp. 141–151.
11. R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM 21* (1978), 120–126.
12. J. Pang. Analysis of a security protocol in $\mu$CRL. Tech. Rep. SEN-R0201, CWI, Amsterdam, 2002.