Combining ISABELLE and Timbuk for Cryptographic Protocol Verification

Frédéric OEHL & David SINCLAIR

School of Computer Applications
Dublin City University
Dublin 9
Ireland
{foehl,dsinclai}@computing.dcu.ie

Abstract

In this paper, the approaches of Paulson and of Genet and Klay to verify cryptographic protocols are summarized. Then the paper explains why and how these two techniques can be combined. By combining these methods, the strengths of each method can be exploited. Thus the verification of the properties of a cryptographic protocol, and in particular for the secrecy and authentication properties, are simplified. To illustrate the idea, the Needham-Schroeder protocol is verified using this combining approach.

1. Introduction

Cryptographic protocols are used to establish secure communication between two or more agents over an insecure network. Among the many properties of the protocol that need to be established and guaranteed are secrecy and authentication. Secrecy guarantees that an information is kept secret during protocol runs. Authentication guarantees that an agent in the network can identify the sender of a message. In the past several protocols have been used for many years before these properties have been found to be invalid. After the discovery of these security flaws, techniques have been developed for the verification of cryptographic protocols. The formal methods (c.f. [19] and [20]) are for example one of those techniques. In [1], [8], [2], [3], [11], [12], [13] and [9] detailed description of formal approaches can be found.

In [3], L. C. Paulson presents a method based on the proof by induction. In his technique, a protocol is modeled by the set of all possible traces where a trace is the list of all the messages exchanged during a protocol run. In addition an intruder or bad agent is assumed to be in the network. This intruder, based on the Dolev and Yao model [7], has access to all the traces and can decrypt messages if he has captured the appropriate decryption keys. In addition he can build and send fraudulent messages if he has the appropriate encryption keys. The properties that the protocol must satisfy are proved by induction on all possible traces that the protocol can generate. That means that after each protocol's step we verify if the properties are satisfied. This method was implemented in Isabelle[1] theorem prover and used to verify the Needham-Schroeder [3], Kerberos [5], [6], TLS [4] and some other protocols. The proofs of these protocols are available on the Isabelle's website[2].

In [12], T. Genet and F. Klay present a method for the analysis of cryptographic protocols based on tree automata, term rewriting systems (TRS) and an approximation technique. A tree automaton is used to model the networks and a term rewriting system is used to formalize the protocol's steps. Similar to the inductive approach, there is an intruder in the network (this intruder has the same abilities as the intruder in the Paulson's method). To verify the secrecy or authentication property, first a superset of all the possible configurations

[1] http://isabelle.in.tum.de/index.html
[2] http://www.cl.cam.ac.uk/Research/HVG/Isabelle/library/HOL/Auth/

of the network is computed from the network's initial configuration (all the agents in the networks want to communicate with each other) with the term rewriting system and an extended version of the approximation technique introduced in [17]. Then an automaton, that models the negation of the property that must be verified, is built. Finally, the intersection of these automata is checked. If the intersection is empty, the protocol is safe otherwise the protocol could be insecure. This method has been used to verify the Needham-Schroeder [12]. A library for OCAML[3] ([22] and [23]), Timbuk[4], has been developed to support the tree automata and the approximation technique.

In [10] we present a method which combines these two approaches. In this paper, we present a case study that use this combined approach on the good version of the Needham-Schroeder protocol.

2. Combining Isabelle and Timbuk

In this section we summarize our intuition for combining the inductive approach [3] and the approach based on tree automata and approximation [12]. First, we will see why we want to combine these approaches and then how we can combine them.

2.1. Why Combine these Two Approaches?

Our idea is to combine model-checking and theorem proving for the verification of cryptographic protocols. In this sense, combining these approaches seems to be a nice starting point.

Both approaches work on the protocol's traces and have the same model of intruder. Moreover they seem to be complementary. The inductive approach can verify several properties (secrecy, authenticity, freshness,...). But in this approach the secrecy and authenticity properties/theorems are very difficult to prove[5]. The proofs require an experienced user to inject the right lemma at the right time to make the proofs converge. This is not the case of the remaining properties, the proofs of those properties are slightly the same for all protocols.

On the other hand, with the approximation technique, a quick and semi-automatic (we only enter the TRS, the approximation function, the initial automaton) verification of the secrecy and authentication properties can be done. But with this method, if the properties are not satisfied, another technique like the inductive approach must be used to find a possible flaw. Another weakness of the approximation technique is that it can't be used to prove the remaining properties (freshness, regularity, ...), because there is no distinction between two pieces of information created in two distinct runs of a protocol.

By combining these two approaches, it should be possible to build easily a complete proof of the protocol's properties with a basic knowledge of both techniques.

2.2. How to Combine?

Our idea is to combine the approximation approach and the inductive approach to exploit the strengths of each method. So:

1. We use the approximation technique to verify the secrecy and authentication properties.

2. If these properties are satisfied, we used these results as axioms in the inductive method. Otherwise we use the inductive approach to prove the existence or the absence of a flaw.

3. We use the inductive approach to prove the remaining properties.

[3] http://caml.inria.fr/ocaml/index.html

[4] http://www.irisa.fr/lande/genet/timbuk/index.html

[5] to see what is involved look the proofs of the Needham-Schroeder protocol: http://www.cl.cam.ac.uk/Research/HVG/Isabelle/library/HOL/Auth/

With Timbuk, it is also possible to find inconsistency in protocol's steps. For example, in the Woo and Lam protocol [21] the second step is "Bob sends a Nonce to Alice" and the third is "Alice encrypts and sends her name, Bob name and his Nonce" but she can't link Bob to the Nonce. She only received a Nonce, so how could she link this nonce to Bob? Timbuk helps us to find this kind flaw.

2.3. The Prototype

We have developed a prototype that generates the input file for Timbuk from the specification file for Isabelle. This prototype is implemented in OCAML.

To summarize the behavior of this prototype, it extracts the protocol's steps from the Isabelle specification and then it uses a lexical analysis and a syntax analysis to build an abstract model of the protocol. Then it uses rewriting techniques to build the TRS and the initial automaton. The prototype also generates the approximation function used by Timbuk to compute the approximation automaton:

- Like in [12], untrustable agents are amalgamated, so three types remain: Alice, Bob and the "Rest"; Messages sent and received by the "Rest" are collapsed together, so only the messages exchanged between Alice, Bob and the "Rest" are studied.

- The normalization process is a bit different of the one in [12]: "new state is introduced to normalize a term if this last hasn't been normalized before".

The annex A (c.f. section A) shows the input file for the good version of the Needham Schroeder protocol. This the protocol will be used in this paper to illustrate the combination idea. The formalisms, used in this paper, are the same as those used in [3] and [12].

3. Needham-Schroeder Protocol

In this version, two agents , Alice and Bob, want to establish a secure communication using a public keys infrastructure (c.f. Figure 1).

	Alice initiates a protocol run, sending a nonce Na and her name A to Bob.
Message 1:	A $\Longrightarrow$ B : $\{Na, A\}_{Kb}$
	Bob responds to Alice's message with a further nonce Nb.
Message 2:	B $\Longrightarrow$ A : $\{Na, Nb, B\}_{Ka}$
	Alice proves her existence by sending Nb back to Bob.
Message 3:	A $\Longrightarrow$ B : $\{Nb\}_{Kb}$

Figure 1: Good version of the Needham-Schroeder protocol

To verify the protocol the first step is to write the Isabelle specification. In this specification, each protocol step is composed of two parts:

- the conditions to execute the step: creation of a new nonce (i.e. Nonce NAB $\sim$: used evs1: the creation by A of a nonce not used previously to communicate with B), presence of a particular message in the set of all previous steps (i.e. Says A B $\{|\ \ldots\ |\}$: set evs)

- the step itself (adding a message to the trace)

```
Theory   NS_Public = Public:

         • • •

         (*Alice initiates a protocol run, sending a fresh nonce to Bob*)
 NS1:    "[ evs1 : ns_public; Nonce NAB ∼: used evs1 ] =⇒
         Says A B {|Crypt (pubK B) {|Nonce NAB, Agent A|}|} # evs1 : ns_public"

         (*Bob responds to Alice's message with a further fresh nonce*)
 NS2:    "[ evs2 : ns_public; Nonce NBA ∼: used evs2;
         Says A' B {|Crypt (pubK B) {|Nonce NAB, Agent A|}|} : set evs2 =⇒
         Says B A {|Crypt (pubK A) {|Nonce NAB, Nonce NBA, Agent B|}|} # evs2 : ns_public"

         (*Alice proves her existence by sending NBA back to Bob.*)
 NS3:    "[ evs3 : ns_public;
         Says A B {|Crypt (pubK B) {|Nonce NAB, Agent A |}|} : set evs3;
         Says B' A {|Crypt (pubK A) {|Nonce NAB, Nonce NBA, Agent B|}|}:
         set evs3] =⇒ Says A B {|Crypt (pubK B) {|Nonce NBA|}|} # evs3 : ns_public"
  end
```

Figure 2: Inductive specification of the Needham-Schroeder protocol

Figure 2 shows the protocol specification using Isabelle syntax. Then the prototype is used to generate the Timbuk input file (c.f. section A). When this file is created, the Timbuk library is used to compute the approximated automaton.

When the computation of the approximated automaton is over, the secrecy and the authentication properties can be verified.

3.1. The Secrecy Property

The secrecy property, that must be guaranteed by the protocol, is: *"The intruder can never access the nonces created by Alice (to communicate with Bob) or Bob (to communicate with Alice)"*.

To verify this property with the approximation technique, first an automaton of the negation of this property must be built with Timbuk. This automaton is identical to the one in [12]. Figure 3 models this automaton. In this figure:

- A models Alice, B models Bob and $agt(x)$ models the agent x;
- $N(q4, q5)$ models the nonce created by $q4$ (Alice) to communicate with $q5$ (Bob);
- $U(q13, q13)$ models the union of two final states, $q13$ and $q13$.This rule is used build the automaton.

```
Automaton      Not_Secret
States         q1 q2 q4 q5 q13
Final States   q13
Transitions
               A -> q1               agt(q1) -> q4
               B -> q2               agt(q2) -> q5
               U(q13, q13) -> q13
               N(q4, q5) -> q13      N(q5, q4) -> q13
               N(q4, q4) -> q13      N(q5, q5) -> q13
```

Figure 3: The intruder has access to the nonces between Alice and Bob

Then the intersection of this automaton with the one corresponding at the approximation of the possible network configurations can be checked (still with Timbuk). For this example the intersection is empty, so the protocol satisfies the property.

3.2. The Authentication Property

The authentication property is: *"If Alice thinks that she communicates with Bob then she really speaks with Bob. And if Bob thinks that he communicates with Alice then he really talks with Alice"*.

As with for the secrecy property, an automaton of the negation of the authentication property is built [12], then the intersection with the approximation is checked. For this version of the Needham-Schroeder, the intersection is empty so the property is satisfied by the protocol.

3.3. How Approximation Results are Used?

The secrecy and the authentication properties are satisfied by the protocol. These results can be used in our inductive proof.

In the inductive proof for this protocol[6], the theorems corresponding to those properties are:

- for the secrecy:
  - Spy_not_see_NA: the Spy does not see the nonce sent in the message *NS1* if Alice and Bob are secure;
  - Spy_not_see_NB: the Spy does not see the nonce sent in the message *NS2* if Alice and Bob are secure.

With the automata technique, we proved that the intruder never catches the nonces exchanged between Alice and Bob. If the intruder can't see the nonces of Alice and Bob then he doesn't see the one sent in *NS1* and the one sent in *NS2*. So the two theorems above are true.

- for the authentication:
  - A_trusts_NS2: if Alice receives message *NS2* and has used NA to start a run, then Bob has sent message NS2;
  - B_trusts_NS3: if Bob receives message *NS3* and has used NB in NS2, then Alice has sent message NS2.

With the Genet and Klay approach, we checked that when Alice wants to establish a communication with Bob, after *NS2* she really speaks with him. We also verified that when Bob thinks that he is responding to Alice, he really speaks with Alice after *NS3*. So the theorems "A_trusts_NS2" and "B_trusts_NS3" are also true.

As all the theorems above are true, they can be added at the end of the Isabelle specification as axioms (c.f. Figure 4). The inductive proof of the other properties [3] can be completed as usual.

4. Conclusion

The Needham-Schroeder protocol used in this paper gives a good idea of how the strengths of each method are exploited. In the correct version of the protocol, the secrecy and the authentication properties are easily proved (checking the intersection of two automata) with the approximation technique. Then this result is used as axiom

[6] http://www.cl.cam.ac.uk/Research/HVG/Isabelle/library/HOL/Auth/

```
Theory              NS_Public = Public:

                    •••

axioms
                    (*If A and B aren't untrustable agents then the intruder cannot get the nonce NAB*)
Spy_not_see_NAB     "[|Says A B {|Crypt(pubK B) {|Nonce NAB, Agent A|}|} : set evs; A ~: bad;
                    B ~: bad; evs : ns_public|] ⟹ Nonce NAB ~: analz (spies evs)"

                    (*If A and B aren't untrustable agents then the intruder cannot get the nonce NBA*)
Spy_not_see_NBA     "[|Says B A {|Crypt (pubK A) {|Nonce NAB, Nonce NBA, Agent B|}|} : set evs;
                    A ~: bad; B ~: bad; evs : ns_public|] ⟹ Nonce NBA ~: analz (spies evs)"


                    (*If A receives the message NS2 then B sent this message*)
A_trusts_NS2        "[|Says A B {|Crypt(pubK B) {|Nonce NAB, Agent A|}|} : set evs;
                    Says B' A {|Crypt(pubK A) {|Nonce NAB, Nonce NBA, Agent B|}|} : set evs;
                    A ~: bad; B ~: bad; evs : ns_public|] ⟹
                    Says B A {|Crypt(pubK A) {|Nonce NAB, Nonce NBA, Agent B|}|} : set evs"

                    (*If B receives the message NS3 then A sent this message*)
B_trusts_NS3        "[|Says B A {|Crypt (pubK A) {|Nonce NAB, Nonce NBA, Agent B|}|} : set evs;
                    Says A' B {|Crypt (pubK B) {|Nonce NBA|}|} : set evs;
                    A ~: bad; B ~: bad; evs : ns_public|] ⟹
                    Says A B {|Crypt (pubK B) {|Nonce NBA|}|} : set evs"
end
```

Figure 4: New inductive specification of the Needham-Schroeder protocol

in the inductive proof and it simplifies the proof (on a 733Mhz Pentium III computer with 128Mb of memory, it takes 2 minutes to compute the approximation and few more minutes to check the intersections with Timbuk) . In the flawed version, that was not tackled for space reasons, the approximation technique is inefficient (the automaton intersections are not empty). But in few minutes (6 for the computation of the approximation plus some others to check the intersections), we know that the protocol might be flawed. So the inductive proof must be done as usual except that we have information on the context of the possible secrecy/authentication flaw. These information can be helpful to prove the existence or not of a flaw for the non-experienced users of Isabelle. In summary, the information learnt from the approximation technique can simplify the inductive proof (axioms or consciousness of the possible existence of a flaw). In the case, where the properties are not guaranteed the inductive method will find the flaw.

As already said in the introduction several methods are available for the verification of cryptographic protocols [1], [8], [2], [3], [11], [12], [13], [9].

All these techniques have been used to check protocols listed by Clark and Jacob [21] or in [1] to be validated. But at the moment few of them have been successfully used to verify an entire real protocol (SET [14], TLS [15], ...). Our concept, combining two methods to exploit their strengths, and our prototype have been also used to verify the Needham-Schroeder protocol (public key without server, shared key with server), the Woo Lam protocol and the simplified version of Otway-Rees. Our next step will be to improve our prototype (at the moment some approximation rules, that are easily detectable, have to be added by hand). Then see if with our concept we can have better results on real protocols than the other methods.

References

[1] M. Burrows and M. Abadi and R. Needham, A Logic of Authentication,DIGITAL, Systems Research Center, N 39, February 1989

[2] M. Abadi and A. D. Gordon, A calculus for Cryptographic Protocols: The Spi calculus, DIGITAL, Systems Research Center, N 149, January 1998

[3] L. C. Paulson, The Inductive Approach to Verifying Cryptographic Protocols, *Journal of Computer Security*, Volume 6, pages 85–128, 1998,http://www.cl.cam.ac.uk/users/lcp/papers/protocols.html

[4] L. C. Paulson, Inductive analysis of the Internet protocol TLS, *ACM Transactions on Information and System Security*, Volume 2, Number 3, pages 332–351,1999

[5] G. Bella and L. C. Paulson, Using Isabelle to prove properties of the Kerberos authentication system, In H. Orman and C. Meadows, editors, *Workshop on Design and Formal Verification of Security Protocols. DIMACS*, 1997

[6] G. Bella and L. C. Paulson, Kerberos Version IV: Inductive Analysis of the Secrecy Goals, *Proceedings of the 5th European Symposium on Research in Computer Security*, Springer-Verlag LNCS 1485, Louvain-la-Neuve, Belgium, J.-J. Quisquater, pages 361–375,1998

[7] D. Dolev and A. Yao, On the security of public-key protocols, *IEEE Transactions on Information Theory*, 2(29), 1983

[8] Cathrine Meadows. The NRL protocol analyser: An overview. *Journal of Logic Programming*, 26(2):113–131, February 1996

[9] Florent Jacquemard and Michael Rusinowitch and Laurent Vigneron. Compiling and Verifying Security Protocols. *Logic Programming and Automated Reasoning*, pages 131-160, 2000

[10] F. Oehl and D. Sinclair, Combining two approaches for the verification of cryptographic protocols, *Workshop Specification, Analysis and Validation for Emerging Technologies in Computational Logic (SAVE 2001)*, 2001

[11] David Monniaux, Abstracting Cryptographic Protocols with Tree Automata, *Static Analysis Symposium*, Springer-Verlag, Lecture Notes in Computer Science, pages 149-163, 1999

[12] Thomas Genet and Francis Klay, Rewriting for Cryptographic Protocol Verification, *CADE: International Conference on Automated Deduction*, 2000, http://citeseer.nj.nec.com/genet99rewriting.html

[13] Jean Goubault-Larrecq, A method for automatic cryptographic protocol verification (extended abstract), *Proc. Workshop on Formal Methods in Parallel Programming, Theory and Applications (FMPPTA'2000)*, Springer-Verlag, pages 977-984, Number 1800, Lecture Notes in Computer Science, 2000

[14] SET Working Group, SET^{TM} Specification, books 1,2 and 3, SETCO, 1996, http://www.setco.org/set_specifications.html

[15] TLS Working Group, The TLS Protocol Version 1.0, The Internet Engineering Task Force, 1996, http://www.ietf.org/html.charters/tls-charter.html

[16] D. Otway and O. Rees. Efficient and timely mutual authentication. *Operating Systems Review*, 21(1):8–10, 1987

[17] Thomas Genet, Decidable Approximations of Sets of Descendants and Sets of Normal Forms, *RTA*, pages 151-165,1998

[18] T.Genet and V. Viet Triem Tong, Reachability Analisys of Term Rewriting Systems with Timbuk, *8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2001)*, 2001

[19] S. Gritzalis, D. Spinellis, and P. Georgiadis. Security Protocols over open networks and distributed systems: Formal methods for their analysis, design, and verification. *Computer Communications*, 22(8): 695-707, May 1999.

[20] B Aziz, D. Gray, G.Hamilton, F. Oehl, J. Power, and D. Sinclair, Implementing Protocol Verification for E-Commerce, *Proceedings of the 2001 International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR 2001)*, L'Aquila, Italy, 6-12 August 2001

[21] J. Clark and J. Jacob. A Survey of Authentication Protocol Literature: Version 1.0. 1997. http://citeseer.nj.nec.com/clark97survey.html

[22] Didier Rémy and Jérôme Vouillon, Objective ML: An effective object-oriented extension to ML. *In Theory And Practice of Objects Systems*, 4(1):27–50, 1998.

[23] Xavier Leroy, Damien Doligez, Jacques Garrigue, Didier Rémy and Jérôme Vouillon, The Objective Caml system release 3.02, Documentation and user's manual, 2001

A. Timbuk input file for the Needham-Schroeder protocol

```
(* Alphabet *)

Ops mesg:3 encr:3 N:2 cons:2 A:0 B:0 S:0 o:0 suc:1 agt:1 serv:1 U:2 sharekey:2 pubkey:1 c_init:3 c_resp:3 add:1 goal:2 LHS:0 hash1:2 hash2:3 null:0

(* Variables *)

Vars x x1 y z u s a b a_18 a_17 a_16 a_15 a_14 a_13 a_12 a_11 a_10 a_9 a_8 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 b_18 b_17 b_16 b_15 b_14 b_13 b_12 b_11 b_10 b_9 b_8 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0

(* Term Rewriting System *)

TRS R

  (* Protocol steps *)
  goal(agt(a), agt(b)) -> U(LHS, mesg(agt(a), agt(b), encr(pubkey(agt(b)), agt(a), cons(N(agt(a), agt(b)), agt(a)))))
  mesg(a_4, agt(b), encr(pubkey(agt(b)), a_3, cons(N(a_1, b_1), agt(a)))) -> U(LHS, mesg(agt(b), agt(a), encr(pubkey(agt(a)), agt(b), cons(N(a_1, b_1), cons(N(agt(b), agt(a)), agt(b))))))
  mesg(a_6, agt(a), encr(pubkey(agt(a)), a_5, cons(N(agt(a), agt(b)), cons(N(a_2, b_2), agt(b))))) -> U(LHS, mesg(agt(a), agt(b), encr(pubkey(agt(b)), agt(a), N(a_2, b_2))))
  mesg(a_6, agt(a), encr(pubkey(agt(a)), a_5, cons(N(agt(a), agt(b)), cons(N(a_2, b_2), agt(b))))) -> U(LHS, c_init(agt(a), agt(b), a_5))
  mesg(a_8, agt(b), encr(pubkey(agt(b)), a_7, N(agt(b), agt(a)))) -> U(LHS, c_resp(agt(b), agt(a), a_7))


  (* Intruder abilities *)
  U(cons(x, y), z) -> U(LHS, add(x))
  U(cons(x, y), z) -> U(LHS, add(y))
  U(encr(pubkey(agt(o)), y, z), u) -> U(LHS, add(z))
  U(encr(pubkey(agt(suc(x))), y, z), u) -> U(LHS, add(z))
  U(mesg(x, y, z), u) -> U(LHS, add(z))
  add(x) -> x
  U(x, U(y, z)) -> U(U(x, y), z)
  U(U(x, y), z) -> U(x, U(y, z))
  U(x, y) -> U(y, x)

(* Initial Automaton *)
```

Automaton automat

States q0 q1 q2 q3 q4 q5 q13

Final States q13

Prior null -> q13

Transitions

o -> q0	suc(q0) -> q0	agt(q0) -> q3
A -> q1		agt(q1) -> q4
B -> q2		agt(q2) -> q5
U(q13, q13) -> q13		
goal(q4, q5) -> q13	goal(q5, q4) -> q13	goal(q4, q4) -> q13
goal(q5, q5) -> q13	goal(q3, q3) -> q13	goal(q4, q3) -> q13
goal(q3, q4) -> q13	goal(q5, q3) -> q13	goal(q3, q5) -> q13
agt(q0) -> q13	agt(q1) -> q13	agt(q2) -> q13
mesg(q13, q13, q13) -> q13	cons(q13, q13) -> q13	encr(q13, q3, q13) -> q13
pubkey(q3) -> q13	pubkey(q4) -> q13	pubkey(q5) -> q13
N(q3, q3) -> q13	N(q3, q4) -> q13	N(q3, q5) -> q13

(* Approximation Function *)

Approximation R1

States q[0–90]

Rules

(Approximation of the first step *)*

[U(LHS, mesg(agt(q2), agt(q2), encr(pubkey(agt(q2)), agt(q2), cons(N(agt(q2), agt(q2)), agt(q2))))) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q2) -> q5 N(q5, q5) -> q15 cons(q15, q5) -> q16 pubkey(q5) -> q17 encr(q17, q5, q16) -> q13 mesg(q5, q5, q13) -> q13]

[U(LHS, mesg(agt(q2), agt(q1), encr(pubkey(agt(q1)), agt(q2), cons(N(agt(q2), agt(q1)), agt(q2))))) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q1) -> q4 N(q5, q4) -> q19 cons(q19, q5) -> q20 pubkey(q4) -> q21 encr(q21, q5, q20) -> q13 mesg(q5, q4, q13) -> q13]

[U(LHS, mesg(agt(q2), agt(q0), encr(pubkey(agt(q0)), agt(q2), cons(N(agt(q2), agt(q0)), agt(q2))))) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q0) -> q3 N(q5, q3) -> q23 cons(q23, q5) -> q24 pubkey(q3) -> q25 encr(q25, q5, q24) -> q13 mesg(q5, q3, q13) -> q13]

[U(LHS, mesg(agt(q1), agt(q2), encr(pubkey(agt(q2)), agt(q1), cons(N(agt(q1), agt(q2)), agt(q1))))) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(q2) -> q5 N(q4, q5) -> q27 cons(q27, q4) -> q28 pubkey(q5) -> q17 encr(q17, q4, q28) -> q13 mesg(q4, q5, q13) -> q13]

[U(LHS, mesg(agt(q1), agt(q1), encr(pubkey(agt(q1)), agt(q1), cons(N(agt(q1), agt(q1)), agt(q1))))) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(q1) -> q4 N(q4, q4) -> q30 cons(q30, q4) -> q31 pubkey(q4) -> q21 encr(q21, q4, q31) -> q13 mesg(q4, q4, q13) -> q13]

[U(LHS, mesg(agt(q1), agt(q0), encr(pubkey(agt(q0)), agt(q1), cons(N(agt(q1), agt(q0)), agt(q1))))) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(q0) -> q3 N(q4, q3) -> q33 cons(q33, q4) -> q34 pubkey(q3) -> q25 encr(q25, q4, q34) -> q13 mesg(q4, q3, q13) -> q13]

[U(LHS, mesg(agt(q0), agt(q2), encr(pubkey(agt(q2)), agt(q0), cons(N(agt(q0), agt(q2)), agt(q0))))) -> q13] -> [LHS -> q13 agt(q0) -> q3 agt(q2) -> q5 N(q3, q5) -> q36 cons(q36, q3) -> q37 pubkey(q5) -> q17 encr(q17, q3, q37) -> q13 mesg(q3, q5, q13) -> q13]

[U(LHS, mesg(agt(q0), agt(q1), encr(pubkey(agt(q1)), agt(q0), cons(N(agt(q0), agt(q1)), agt(q0))))) -> q13] -> [LHS -> q13 agt(q0) -> q3 agt(q1) -> q4 N(q3, q4) -> q39 cons(q39, q3) -> q40 pubkey(q4) -> q21 encr(q21, q3, q40) -> q13 mesg(q3, q4, q13) -> q13]

[U(LHS, mesg(agt(q0), agt(q0), encr(pubkey(agt(q0)), agt(q0), cons(N(agt(q0), agt(q0)), agt(q0))))) -> q13] -> [LHS -> q13 agt(q0) -> q3 agt(q0) -> q3 N(q3, q3) -> q42 cons(q42, q3) -> q43 pubkey(q3) -> q25 encr(q25, q3, q43) -> q13 mesg(q3, q3, q13) -> q13]

(Approximation of the second step *)*

(* added by hand *)

[U(LHS, mesg(agt(q2), agt(q2), encr(pubkey(agt(q2)), agt(q2), cons(N(q5, q5), cons(N(agt(q2), agt(q2)), agt(q2)))))) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q2) -> q5 N(q5, q5) -> q15 cons(q15, q5) -> q16 cons(q15, q16) -> q46 pubkey(q5) -> q17 encr(q17, q5, q46) -> q13 mesg(q5, q5, q13) -> q13]

[U(LHS, mesg(agt(q2), agt(q2), encr(pubkey(agt(q2)), agt(q2), cons(N(a_1, b_1), cons(N(agt(q2), agt(q2)), agt(q2)))))) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q2) -> q5 N(q5, q5) -> q15 cons(q15, q5) -> q16 N(a_1, b_1) -> q45 cons(q45, q16) -> q46 pubkey(q5) -> q17 encr(q17, q5, q46) -> q13 mesg(q5, q5, q13) -> q13]

[U(LHS, mesg(agt(q2), agt(q1), encr(pubkey(agt(q1)), agt(q2), cons(N(a_2, b_2), cons(N(agt(q2), agt(q1)), agt(q2)))))) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q1) -> q4 N(q5, q4) -> q19 cons(q19, q5) -> q20 N(a_2, b_2) -> q48 cons(q48, q20) -> q49 pubkey(q4) -> q21 encr(q21, q5, q49) -> q13 mesg(q5, q4, q13) -> q13]

(* added by hand *)

[U(LHS, mesg(agt(q2), agt(q0), encr(pubkey(agt(q0)), agt(q2), cons(N(q5, q3), cons(N(agt(q2), agt(q0)), agt(q2)))))) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q0) -> q3 N(q5, q3) -> q23 cons(q23, q5) -> q24 cons(q23, q24) -> q52 pubkey(q3) -> q25 encr(q25, q5, q52) -> q13 mesg(q5, q3, q13) -> q13]

[U(LHS, mesg(agt(q2), agt(q0), encr(pubkey(agt(q0)), agt(q2), cons(N(a_3, b_3), cons(N(agt(q2), agt(q0)), agt(q2)))))) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q0) -> q3 N(q5, q3) -> q23 cons(q23, q5) -> q24 N(a_3, b_3) -> q51 cons(q51, q24) -> q52 pubkey(q3) -> q25 encr(q25, q5, q52) -> q13 mesg(q5, q3, q13) -> q13]

[U(LHS, mesg(agt(q1), agt(q2), encr(pubkey(agt(q2)), agt(q1), cons(N(a_4, b_4), cons(N(agt(q1), agt(q2)), agt(q1)))))) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(q2) -> q5 N(q4, q5) -> q27 cons(q27, q4) -> q28 N(a_4, b_4) -> q54 cons(q54, q28) -> q55 pubkey(q5) -> q17 encr(q17, q4, q55) -> q13 mesg(q4, q5, q13) -> q13]

(* added by hand *)

[U(LHS, mesg(agt(q1), agt(q1), encr(pubkey(agt(q1)), agt(q1), cons(N(q4, q4), cons(N(agt(q1), agt(q1)), agt(q1)))))) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(q1) -> q4 N(q4, q4) -> q30 cons(q30, q4) -> q31 cons(q30, q31) -> q58 pubkey(q4) -> q21 encr(q21, q4, q58) -> q13 mesg(q4, q4, q13) -> q13]

[U(LHS, mesg(agt(q1), agt(q1), encr(pubkey(agt(q1)), agt(q1), cons(N(a_5, b_5), cons(N(agt(q1), agt(q1)), agt(q1)))))) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(q1) -> q4 N(q4, q4) -> q30 cons(q30, q4) -> q31 N(a_5, b_5) -> q57 cons(q57, q31) -> q58 pubkey(q4) -> q21 encr(q21, q4, q58) -> q13 mesg(q4, q4, q13) -> q13]

(* added by hand *)

[U(LHS, mesg(agt(q1), agt(q0), encr(pubkey(agt(q0)), agt(q1), cons(N(q4, q3), cons(N(agt(q1), agt(q0)), agt(q1)))))) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(q0) -> q3 N(q4, q3) -> q33 cons(q33, q4) -> q34 cons(q33, q34) -> q61 pubkey(q3) -> q25 encr(q25, q4, q61) -> q13 mesg(q4, q3, q13) -> q13]

[U(LHS, mesg(agt(q1), agt(q0), encr(pubkey(agt(q0)), agt(q1), cons(N(a_6, b_6), cons(N(agt(q1), agt(q0)), agt(q1)))))) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(q0) -> q3 N(q4, q3) -> q33 cons(q33, q4) -> q34 N(a_6, b_6) -> q60 cons(q60, q34) -> q61 pubkey(q3) -> q25 encr(q25, q4, q61) -> q13 mesg(q4, q3, q13) -> q13]

[U(LHS, mesg(agt(q0), agt(q2), encr(pubkey(agt(q2)), agt(q0), cons(N(a_7, b_7), cons(N(agt(q0), agt(q2)), agt(q0)))))) -> q13] -> [LHS -> q13 agt(q0) -> q3 agt(q2) -> q5 N(q3, q5) -> q36 cons(q36, q3) -> q37 N(a_7, b_7) -> q63 cons(q63, q37) -> q64 pubkey(q5) -> q17 encr(q17, q3, q64) -> q13 mesg(q3, q5, q13) -> q13]

[U(LHS, mesg(agt(q0), agt(q1), encr(pubkey(agt(q1)), agt(q0), cons(N(a_8, b_8), cons(N(agt(q0), agt(q1)), agt(q0)))))) -> q13] -> [LHS -> q13 agt(q0) -> q3 agt(q1) -> q4 N(q3, q4) -> q39 cons(q39, q3) -> q40 N(a_8, b_8) -> q66 cons(q66, q40) -> q67 pubkey(q4) -> q21 encr(q21, q3, q67) -> q13 mesg(q3, q4, q13) -> q13]

(* added by hand *)

[U(LHS, mesg(agt(q0), agt(q0), encr(pubkey(agt(q0)), agt(q0), cons(N(q3, q3), cons(N(agt(q0), agt(q0)), agt(q0)))))) -> q13] -> [LHS -> q13 agt(q0) -> q3 agt(q0) -> q3 N(q3, q3) -> q42 cons(q42, q3) -> q43 cons(q42, q43) -> q70 pubkey(q3) -> q25 encr(q25, q3, q70) -> q13 mesg(q3, q3, q13) -> q13]

[U(LHS, mesg(agt(q0), agt(q0), encr(pubkey(agt(q0)), agt(q0), cons(N(a_9, b_9), cons(N(agt(q0), agt(q0)), agt(q0)))))) -> q13] -> [LHS -> q13 agt(q0) -> q3 agt(q0) -> q3 N(q3, q3) -> q42 cons(q42, q3) -> q43 N(a_9, b_9) -> q69 cons(q69, q43) -> q70 pubkey(q3) -> q25 encr(q25, q3, q70) -> q13 mesg(q3, q3, q13) -> q13]

(Approximation of the third step *)*

[U(LHS, mesg(agt(q2), agt(q2), encr(pubkey(agt(q2)), agt(q2), N(a_10, b_10)))) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q2) -> q5 N(a_10, b_10) -> q73 pubkey(q5) -> q17 encr(q17, q5, q73) -> q13 mesg(q5, q5, q13) -> q13]

[U(LHS, mesg(agt(q2), agt(q1), encr(pubkey(agt(q1)), agt(q2), N(a_11, b_11)))) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q1) -> q4 N(a_11, b_11) -> q75 pubkey(q4) -> q21 encr(q21, q5, q75) -> q13 mesg(q5, q4, q13) -> q13]

[U(LHS, mesg(agt(q2), agt(q0), encr(pubkey(agt(q0)), agt(q2), N(a_12, b_12)))) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q0) -> q3 N(a_12, b_12) -> q77 pubkey(q3) -> q25 encr(q25, q5, q77) -> q13 mesg(q5, q3, q13) -> q13]

[U(LHS, mesg(agt(q1), agt(q2), encr(pubkey(agt(q2)), agt(q1), N(a_13, b_13)))) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(q2) -> q5 N(a_13, b_13) -> q80 pubkey(q5) -> q17 encr(q17, q4, q80) -> q13 mesg(q4, q5, q13) -> q13]

[U(LHS, mesg(agt(q1), agt(q1), encr(pubkey(agt(q1)), agt(q1), N(a_14, b_14)))) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(q1) -> q4 N(a_14, b_14) -> q82 pubkey(q4) -> q21 encr(q21, q4, q82) -> q13 mesg(q4, q4, q13) -> q13]

[U(LHS, mesg(agt(q1), agt(q0), encr(pubkey(agt(q0)), agt(q1), N(a_15, b_15)))) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(q0) -> q3 N(a_15, b_15) -> q84 pubkey(q3) -> q25 encr(q25, q4, q84) -> q13 mesg(q4, q3, q13) -> q13]

[U(LHS, mesg(agt(q0), agt(q2), encr(pubkey(agt(q2)), agt(q0), N(a_16, b_16)))) -> q13] -> [LHS -> q13 agt(q0) -> q3 agt(q2) -> q5 N(a_16, b_16) -> q86 pubkey(q5) -> q17 encr(q17, q3, q86) -> q13 mesg(q3, q5, q13) -> q13]

[U(LHS, mesg(agt(q0), agt(q1), encr(pubkey(agt(q1)), agt(q0), N(a_17, b_17)))) -> q13] -> [LHS -> q13 agt(q0) -> q3 agt(q1) -> q4 N(a_17, b_17) -> q88 pubkey(q4) -> q21 encr(q21, q3, q88) -> q13 mesg(q3, q4, q13) -> q13]

[U(LHS, mesg(agt(q0), agt(q0), encr(pubkey(agt(q0)), agt(q0), N(a_18, b_18)))) -> q13] -> [LHS -> q13 agt(q0) -> q3 agt(q0) -> q3 N(a_18, b_18) -> q90 pubkey(q3) -> q25 encr(q25, q3, q90) -> q13 mesg(q3, q3, q13) -> q13]

(Approximation of the remaining rules *)*

[U(LHS, c_init(agt(q2),agt(q2),z)) -> q13] -> [LHS -> q13 agt(q2) -> q5 c_init(q5,q5,z) -> q13]

[U(LHS, c_init(agt(q2),agt(q1),z)) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q1) -> q4 c_init(q5,q4,z) -> q13]

[U(LHS, c_init(agt(q2),agt(q0),z)) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q0) -> q3 c_init(q5,q3,z) -> q13]

[U(LHS, c_init(agt(q1),agt(q1),z)) -> q13] -> [LHS -> q13 agt(q1) -> q4 c_init(q4,q4,z) -> q13]

[U(LHS, c_init(agt(q1),agt(q2),z)) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(q2) -> q5 c_init(q4,q5,z) -> q13]

[U(LHS, c_init(agt(q1),agt(q0),z)) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(y) -> q3 c_init(q4,q3,z) -> q13]

[U(LHS, c_init(agt(q0),agt(q2),z)) -> q13] -> [LHS -> q13 agt(q0) -> q3 agt(q2) -> q5 c_init(q3,q5,z) -> q13]

[U(LHS, c_init(agt(q0),agt(q1),z)) -> q13] -> [LHS -> q13 agt(q0) -> q3 agt(q1) -> q4 c_init(q3,q4,z) -> q13]

[U(LHS, c_init(agt(q0),agt(q0),z)) -> q13] -> [LHS -> q13 agt(q0) -> q3 c_init(q3,q3,z) -> q13]

[U(LHS, c_resp(agt(q2),agt(q2),z)) -> q13] -> [LHS -> q13 agt(q2) -> q5 c_resp(q5,q5,z) -> q13]

[U(LHS, c_resp(agt(q2),agt(q1),z)) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q1) -> q4 c_resp(q5,q4,z) -> q13]

[U(LHS, c_resp(agt(q2),agt(q0),z)) -> q13] -> [LHS -> q13 agt(q2) -> q5 agt(q0) -> q3 c_resp(q5,q3,z) -> q13]

```
[U(LHS, c_resp(agt(q1),agt(q1),z)) -> q13] -> [LHS -> q13 agt(q1) -> q4 c_resp(q4,q4,z) -> q13]
[U(LHS, c_resp(agt(q1),agt(q2),z)) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(q2) -> q5 c_resp(q4,q5,z) -> q13]
[U(LHS, c_resp(agt(q1),agt(q0),z)) -> q13] -> [LHS -> q13 agt(q1) -> q4 agt(y) -> q3 c_resp(q4,q3,z) -> q13]
[U(LHS, c_resp(agt(q0),agt(q2),z)) -> q13] -> [LHS -> q13 agt(q0) -> q3 agt(q2) -> q5 c_resp(q3,q5,z) -> q13]
[U(LHS, c_resp(agt(q0),agt(q1),z)) -> q13] -> [LHS -> q13 agt(q0) -> q3 agt(q1) -> q4 c_resp(q3,q4,z) -> q13]
[U(LHS, c_resp(agt(q0),agt(q0),z)) -> q13] -> [LHS -> q13 agt(q0) -> q3 c_resp(q3,q3,z) -> q13]
[U(LHS, add(x)) -> q13] -> [LHS -> q13 add(x) -> q13]
[U(U(x,y),y) -> q13] -> [U(x,y) -> q13]
```