A Framework for Proving the Correctness of Cryptographic Protocol Properties by Linear Temporal Logic

Abdullah Alabdulatif, Xiaoqi Ma, Lars Nolle
School of Science and Technology, Nottingham Trent University

Abstract

In this paper, a framework for cryptographic protocol analysis using linear temporal logic is proposed. The framework can be used to specify and analyse security protocols. It aims to investigate and analyse the security protocols properties that are secure or have any flaws. The framework extends the linear temporal logic by including the knowledge of participants in each status that may change over the time. It includes two main parts, the Language of Temporal Logic (LTL) and the domain knowledge. The ability of the framework is demonstrated by analysing the Needham-Schroeder public key protocol and the Andrew Secure RPC protocol as examples.

1. Introduction

Designers of protocols use the trial-and-error method to design for analysing security protocols. Therefore, without the use of formal methods for the verification of protocols errors can remain undetected[1]. One of the advantages of formal verification is that it provides a systematic way to discover weaknesses in protocols. However, formal verification is not an easy task because there are wide ranges of complicated behaviours involved in verifying security protocols. A number of methods have been proposed by researchers to formally analyse security protocols [2]. Several researchers have developed formal methods with different techniques to raise assurance level in the correctness of security protocols. The BAN logic is one of the methods used early to prove security protocols.

Burrows, Abadi and Needham developed the BAN logic method for analysing security properties of protocols. The BAN logic method is an important early attempt to examine the security of protocols. The BAN logic is a method for analysing the authentication of protocols [2,3]. However, the BAN logic is inappropriate to express the properties and processes of dynamic system as security protocols [4]. Subsequently, a number of researchers have worked to propose other formal logic for analysing the cryptographic protocols. For instance, semantics for the analysis of cryptographic protocols [5], and Syverson and van Oorschot have built a framework to unify some cryptographic protocol logics [6]. All of the proposed logics have syntax and semantic which can be used as a formal system for analysing the security protocols.

Researchers have found that time is important to express the properties of security protocols. Temporal logic is a formal logic that can be used as a method for analysing security protocols. The temporal logic can specify dynamic systems that change over time[7]. The proposed framework has been built by combining temporal and epistemic logic. It can be used to guarantee the specific knowledge of participants over time[2].

On the other side, Lei et al agreed that temporal logic is suitable to reason the properties such as safety and liveness [8]. However, they have found that there are some difficulties in using temporal logic to model security protocols. The difficulties are firstly, the time in the temporal logic is abstract, which is not appropriate to model protocols. Secondly, modelling security protocols needs to express a concrete process over a series of time that is hard to model by temporal logic. For these reason Lei et al built a framework that can express the time dependent properties[8].

The framework presented in this paper will use linear temporal logic to present the knowledge of participants over running the protocol. It analyses the knowledge in each state of the protocol to ensure participants have knowledge they should know at specific states. It describes what participants do not know and what they should not known at specific states of the protocol.

The paper is structured as follows. Section 2 will present the framework that includes two parts, the language of the logic and the domain knowledge. Section 3 will describe the steps of Needham-Schroeder public key protocol. Section 4 will show how the framework can be used to analyse the Needham-Schroeder public key protocol. Section 5 will illustrate the Andrew Secure RPC protocol steps. Section 6 will shows how we can analyse the protocol and detect the Claek-Jacob Attack by use the framework. Section 7 will present conclusions and future work.

2.The framework

This proposed framework is based on linear temporal logic. The knowledge-based framework is proposed to prove the correctness of security protocols. The language of the logic is used to write protocol steps and to represent the properties of protocols in a formal language.

2.1. The language of temporal logic

This part is representing the syntax and semantics language of the framework. The language is basically an instance of linear temporal logic.

2.1.1. Syntax of the language of temporal logic

The Language of temporal logic is composed of an alphabet, terms, formulae, axioms and deduction rules of the framework as follows:

Definition 1 (Alphabet): The alphabet of the logic language is based on symbols defined in [9], and was extended by adding the statuses and temporal operators that are appropriate to the proposed framework. The alphabet for the framework is as follows:

a. $c_1, c_2, c_3, \ldots\ldots$ (Constants).
b. $v_1, v_2, v_3, \ldots\ldots$ (Variables).
c. $f_1, f_2, \ldots\ldots$ (Function symbols).
d. $p_1, p_2, \ldots\ldots$ (Predicates).
e. $\wedge$, $\vee$, $\longrightarrow$, $\leftrightarrow$, $\neg$. (Logical connectives).
f. $\square$, $\lozenge$, o . (Temporal operators).
g. $\forall, \exists$. (Quantifiers).
h. $s_0, s_1, s_2, \ldots\ldots$ (Statuses).
i. "(" , ")"and ",". (Punctuation marks).

Definition 2 (Operation " ′ " , Next and binary relation $\preccurlyeq$): Let S be the set of statuses and $x, y \in S$ and g be time of occurrence, then the status function can be defined as follows [10]:

a. Next $\subset \boldsymbol{S^2}$: Next $(\boldsymbol{x}, \boldsymbol{y}) \leftrightarrow \boldsymbol{g(y) = g(x) + 1}$.
b. The operation "′" applies to status x then will give status x'_such that Next (x, x').
c. $\preccurlyeq\, \subset S^2 : x \preccurlyeq y \leftrightarrow g(x) \le g(y)$.

Definition 3 (Terms): Let $\boldsymbol{C}$ be the set of constant symbols, $\boldsymbol{V}$ be the set of variables and $\boldsymbol{F}$ be the set of functions. The set of terms $\boldsymbol{T}$ can be defined as follows:

a. $C, V \in T$.
b. If $f \in F$ and $t_1, t_2, \ldots, t_n \in T$, then $f(t_1, t_2, \ldots, t_n) \in T$ where $N > 0$.
c. The set of all terms is created from (a) and (b). No other string is a term.

Definition 4 (Formula): Let V be a set of variables, P be a set of predicate symbols, T be a set of terms. And S a set of statuses. The set of formulas F can be defined as follows:

a. If $t_1, t_2, \ldots\ldots, t_n \in T$, $s_i \in S$ and $p \in P$ then $p(s_i, t_1, t_2, \ldots., t_n) \in F$ where $n > 0$. It can be called atomic formula.
b. If $A \in F$ then $(\neg A) \in F$.
c. If $A \in F$ and $B \in F$ then the follows are formulas: $A \wedge B$, $A \vee B$, $A \longrightarrow B$ and $A \leftrightarrow B$.
d. If $A \in F$ and $x \in V$ then the follows are formula:
   - $\forall\, x. A$.
   - $\exists\, x. A$.
e. If $A \in F$ then the follows are formulas:
   - $\square\, A$.
   - $\lozenge\, A$.
   - $o\, A$.

To know the truth of a formula at a moment in time, a status formula is introduced by defining the set S that includes all individual statuses in the path. The set S can be defined as follows:
S: $\{s_0, s_1, s_2, \ldots\ldots, s_n\}$ where $n \in N$.

Now the definition of formula can be extended as follows:

Definition 5 (Extended Formula): Let $A \in F$ where F is the set of formulas, and $s_i, s_j \in S$ where S is the set of statuses, then the follows are formulas:

a. $A(s_i)$.
b. Next $(s_i, s_i{}')$.
c. $s_i \preccurlyeq s_j$.

In Addition, there are a number of axioms and deduction rules used in the framework which will be introduced later. The deduction rules include three kinds of rules: propositional rules, temporal rules and quantifier rules.

2.1.2. Semantics of the language of temporal logic

To give the semantics of the language of temporal logic, which is based on the Kripke structure model [11], we will firstly define the Kripke structure.

Definition 6 (Kripke Structure): Let S denote the suffix of the path $S = s_i, s_j, \ldots\ldots$ and f be a set of atomic propositions which is not empty. A Kripke structure is a four tuple $M = (S, s_i, R, L)$, where

a. S is a finite set of statuses,
b. s_i is the current status,
c. $R \subseteq S \times S$ is a transition relation, for which it holds that $\forall s \in S : \exists s' \in S : (s, s') \in R$,
d. $L : S \longrightarrow 2^f$ is labelling, a function which labels each status with atomic propositions which hold in this status.

Definition 7 (The model): Assume M is a Kripke structure, and S is a path in M. If the well-formed formula A is satisfied in the path S at specific status s_i, it can be abbreviated as $\langle S, s_i \rangle \vDash A$. The relation $\vDash$ can be define as follows:

$\langle S, s_i \rangle \vDash p(s_i)$	iff	$p \in s_i \; for \; p \in F$
$\langle S, s_i \rangle \vDash \neg A(s_i)$	iff	$\langle S, s_i \rangle \nvDash A(s_i)$
$\langle S, s_i \rangle \vDash A(s_i) \wedge B(s_i)$	iff	$\langle S, s_i \rangle \vDash A(s_i)$ and$\langle S, s_i \rangle \vDash B(s_i)$
$\langle S, s_i \rangle \vDash A(s_i) \vee B(s_i)$	iff	$\langle S, s_i \rangle \vDash A(s_i)$ or$\langle S, s_i \rangle \vDash B(s_i)$
$\langle S, s_i \rangle \vDash A(s_i) \rightarrow B(s_i)$	iff	$\langle S, s_i \rangle \nvDash A(s_i)$ or$\langle S, s_i \rangle \vDash B(s_i)$
$\langle S, s_i \rangle \vDash \Box A(s_i)$	iff	$For \; each \; s_j \; if \; s_i \leq s_j \; then \; \langle S, s_j \rangle \vDash A(s_j)$
$\langle S, s_i \rangle \vDash \Diamond A(s_i)$	iff	$there \; exists \; if \; s_i \leq s_j \; then \; \langle S, s_j \rangle \vDash A(s_j)$
$\langle S, s_i \rangle \vDash oA(s_i)$	iff	$\langle S, s_{i+1} \rangle \vDash A(s_{i+1})$

2.2. The domain knowledge

The domain knowledge defines the knowledge of agents who are participants during the running of a protocol. In the domain knowledge, three types of participants involved in the protocol are defined. The first type is a server or trusted third party. The second type is a friend agent, including all legitimate agents participating the protocol. The third type is a malicious agent or attacker, which includes agents trying to obtain information during running the protocol in an unauthorised way.

Definition 8 (Agents):

$$Agent ::= \text{Server} \; |\text{Friend} \; |\text{Attacker}$$

Agents can generate random numbers called Nonces. A Nonce should be fresh and unique for identifying a protocol session. Also, the domain knowledge has defined two types of keys. For the asymmetric cryptosystem there is a public and a private key. For a symmetric cryptosystem there is a shared session key. The time during running the protocol can be divided into statuses each of which indicates a moment of time. The agents use messages to talk over the network, where the combination of two messages and can be represented as $\langle m_1; m_2 \rangle$. In the domain knowledge there are different types of messages as defined below.

Definition 9 (Message):

$Message ::=$ Agent(A)| Nonce(N)|Key(k)| $Hash(m)$| Encrypt(m, k)|Cert(m, k)| Sign (m, k)| MPair(m1, m2).

There are actions, functions and predicates used to represent the processes and properties in the protocol. Let A and B be agents and m be message, the agent A can generate a new message using the action Generate (A, m). The agent A can send the message m to the agent B and receive message m by using the actions $send$ (A, b, m) and $Rcv(A, m)$ respectively. There are a number of functions an agent might use to help in the network to meet the cryptographic requirements. There are two kinds of functions depending on the techniques of the cryptographic protocols, which are either symmetric or asymmetric. The asymmetric functions are defined as follows:

- $Kpb(A)$: denoting the public encryption key of agent A.
- $Kpr(A)$: denoting the private encryption key of agent A.
- $Spb(A)$: denoting the public signature key of agent A.
- $Spr(A)$: denoting the private signature key of agent A.

On the other hand, the framework defines one symmetric function as follows:

- $Ksym(A, B, n)$: denoting that agents A and B share the symmetric key n. In some cases, two agents might share two or more symmetric keys, which should be distinguished from each other.

A predicate takes parameters and returns true or false. The framework defines some predicates to describe the knowledge of agents. There are a number of predicates identified as follows:

- $Know$ (s_i, A, m): denoting at status $\mathrm{s_i}$ agent A knows the message m. Either the agent A has generated the message m or received from another agent.
- $Auth(s_i, A, B, m)$: denoting at status s_i the message m has not been altered when sent to agent A from agent B.
- $Veri(s_i, A, m)$: denoting at status s_i, agent A verifies the message m.
- $Contain(s_i, m_1, m_2)$: denoting at status s_i, the message m_2 is contained within the message m_1.
- $Equal(s_i, A, B)$: denoting at status s_i, two elements of messages or agents are same as each other.

2.3. Assumption and rules

There are a number of security assumptions and rules used to prove the properties of security protocols. The security assumptions are agreed by most of researchers in the protocol verification field [12]. The rules are used to infer new knowledge at the current status. The assumptions and rules have been formulated as follows:

Assumption 1: The symmetric key must originally only be known by the two agents who share the key. No other agent or spy can know this key.

$\forall A, B, C, n. (C \neq A) \wedge (C \neq B) \longrightarrow$
$\neg \, Know \; (C, Key \; (Ksym \; (A, B, n)))$.

Assumption 2: The public key of a legitimate agent is known to all agents in the network.

$$\forall A, B. Know\left(A, Key\left(Kpr(B)\right)\right).$$

Assumption 3: Every agent knows his own private key.

$$\forall A. Know(A, key(Kpb(A))).$$

Assumption 4: The private key must not be sent over the network.

$$\forall A, B. (A \neq B) \longrightarrow \neg\, Send\ (A, B, Key(Kpr(A))).$$

Assumption 5: Over the network an attacker should not be a friend or the server.
$\forall A. (A = Attacker) \longrightarrow (A \neq Friend) \wedge (A \neq Server)$.

Assumption 6: For all keys, K^{-1} is the inverse of the key K. The equation is $K = (K^{-1})^{-1}$.

$$\forall K. K = (K^{-1})^{-1}$$

Rule 1: If an agent A knows message m and the key k, then the agent can use the key to encrypt the message. The final result is that the agent can know the encrypted messages.

$$\frac{Know\ (s_i, A, m) \wedge Know\ (s_i, A, Key(k))}{Know(s_i, A, Encrypt(m, k))}$$

Rule 2: If agent A knows the message m encrypted with the key k and knows the corresponding decryption key, then A can use the decryption key to decrypt the message. The final result is that the agent can know the content of the original message.

$$\frac{Know\left(s_i, A, Encrypt(m, k)\right) \wedge Know(s_i, A, Key(k^{-1}))}{Know(s_i, A, m)}$$

Rule 3: If agent A knows two different messages, then the agent can combine them.

$$\frac{Know(s_i, A, m_1) \wedge Know(s_i, A, m_2) \wedge s_i \preccurlyeq s_j}{know(s_j, A, \langle m_1, m_2 \rangle)}$$

Rule 4: If agent A knows that there are two messages combined together, then the agent can separate them.

$$\frac{know(s_i, A, \langle m_1, m_2 \rangle)}{Know(s_i, A, m_1) \wedge Know(s_i, A, m_2)}$$

Rule 5: If agent A sends a message m to another agent B, then agent A must know this message before he sends it.

$$\frac{Send\left(s_j, A, B, m\right) \quad s_i \preccurlyeq s_j}{Know(s_i, A, m)}$$

Rule 6: The attacker can eavesdrop all messages in the network.

$$\frac{Send(s_i, A, B, m)}{Rcv(s_i, Attacker, m)}$$

Rule 7: If agent A has received a message m then A should know the content of this message, and nobody can force agent A to delete this message.

$$\frac{Rcv(s_i, A, m) \quad s_i \preccurlyeq s_j}{\Box\, Know(s_j, A, m)}$$

Rule 8: If agent A receives a message m at moment j then there is another agent who sent this message to A before the moment j.

$$\frac{\exists\, X. Send(s_i, X, A, m)}{Rcv\left(s_j, A, m\right) \quad s_i \preccurlyeq s_j}$$

3. The Needham-Schroeder public key protocol

The Needham-Schroeder public key (NSPK) protocol is a simple protocol with just three steps and it has a known flaw. The flaw was found by Lowe in 1995 [13]. The aim of NSPK is achieving successfully established authentication between two agents A, B who are named the initiator and responder respectively. The three steps of the NSPK protocol can be represented as following:

1. $A \longrightarrow B$ $\quad \{A, N_A\}_{pub-key(B)}$
2. $B \longrightarrow A$ $\quad \{N_A, N_B\}_{pub-key(A)}$
3. $A \longrightarrow B$ $\quad \{N_B\}_{pub-key(B)}$

Note:
1.$\{A, B\}_{pub-key(c)}$:Two messages A, B are combined and encrypted by the agent C's public key.
2.N_A: It denotes the random number generated by the agent A that should be unique and unknown to other agents. It is called a nonce.

The messages of NSPK can be described as follows:
Message 1: The agent A initiates the protocol by sending to agent B an encrypted message that containing A's identity and nonce encrypted with B's public key.
Message 2: If B receives message 1, B can know N_A by decrypting the message. Then, B responds to A by sending a message encrypted with public key of A containing the nonce N_A and a sender nonce N_B which is generate by B.
Message 3: If A receives message 2, A can know N_B by decrypting the message. Then, A responds to B by sending a message encrypted with the public key of B containing nonce N_B.

After running the protocol, the agent A can be sure that he or she talks to agent B. In the same way, agent B can be sure that he or she talks to the agent

A. Lowe has shown that this is not true as explained in the next section.

4. Analysis of Needham-Schroeder public key protocol

In this section, the framework presented in the section 2 will be used to investigate the NSPK protocol as a case study. There are four steps that are guidelines for proving the correctness of security protocols [12]. This part will follow these four step to analyse the NSPK protocol.

4.1. Adjusting the framework

There are some minor differences among security protocols where each protocol has different security objectives. In this step, the framework will be adjusted slightly to the specifications of the NSPK protocol.

In the NSPK protocol there are two honest agents (A and B) and an attacker. Hence, the types of agent can be defined as follows:

$$Agents \equiv Friends \mid Attacker$$

And the set of friends is defined as:

$$Friends = \{A, B\}$$

So, the assumption of attacker can be changed according to the definition of the agents, as follows:

$$\forall Attacker. (Attacker \neq A) \wedge (Attacker \neq B)$$

4.2. Modelling the protocol

In this step, the three steps of the NSPK protocol will be converted from an informal language as written in section 3 to a formal language using the frameworks notations as follows:

$NSPK1 : Send(A, B, Encrypt(\langle A, Nonce(A)\rangle, Key\ (Kpb(B))))$.

$NSPK2 : Send(A, B, Encrypt(\langle A, Nonce(A)\rangle, Key\ (Kpb(B))) \Longrightarrow Send(B, A, Encrypt\ (\langle Nonce(B), Nonce(A)\rangle, Key(Kpb(A))))$.

$NSPK3 : Send(B, A, Encrypt\ (\langle Nonce(B), Nonce(A)\rangle, Key(Kpb(A)))) \Longrightarrow Send(A, B, Encrypt\ (Nonce(B), Key(Kpb(B))))$.

These three steps above are enough for the friends (honest users) to successfully run the protocol. On the other hand, it should not be overlooked that the attacker does not necessarily follow the protocol rules. According to the assumptions and rules in the framework, from the fake message rule and the attacker rule two additional rules can be used:

$Reply\ NSPK2\ to\ Attacker : Send(Attacker, B, Encrypt(\langle A, Nonce(A)\rangle, Key\ (Kpb(B))) \Longrightarrow Send(B, Attacker, Encrypt\ (\langle Nonce(B), Nonce(A)\rangle, Key(Kpb(Attacker))))$.

$Reply\ NSPK3\ to\ Attacker : Send(Attacker, A, Encrypt\ (\langle Nonce(B), Nonce(A)\rangle, Key(Kpb(A)))) \Longrightarrow Send(A, Attacker, Encrypt\ (Nonce(B), Key(Kpb(Attacker))))$.

4.3. Proving basic properties

There are a number of basic properties that are common among most of protocols such as knowing the content of received message. The basic properties of the protocol need to be proved. All these basic properties can be reused in proving other protocols.

In this paper we will prove two basic properties. The **lemma 1** will be proved, (Knowledge of message): $Send(A, B, m) \Longrightarrow Know(B, m)$. This lemma consists of one goal, which is $Know(B, m)$ and one antecedent, which is $Send\ (A, B, m)$. The lemma says that, if agent A sends message m to another agent B, the agent B can read the message m. The steps of proving the lemma 1 are as follows:

1. $Know(s_j, B, m)$	Consequent
2. $s_i \preccurlyeq s_j$	assumption
3. $\Box\ Know(s_i, B, m)$	1,2,always elimination rule
4. $s_e \preccurlyeq s_i$	assumption
5. $Rcv(s_e, B, m)$	3,4, rule 7
6. $s_f \preccurlyeq s_e$	assumption
7. $\exists X. Send(s_f, X, B, m)$	5,6, rule 8
8. $Send(s_f, A, B, m)$	7,Existential introduction rule
9.done	

The **lemma 2** will be proved, (Knowing encrypt message): $Know(A, Encrypt\ (m, Key(Kpb(A)))) \Longrightarrow Know(A, m)$. This lemma consists of one goal, which is $Know(A, m)$ and one antecedent, which is $Know(A, Encrypt(m, Key(Kpb(A))))$. The lemma says that if agent A know message m encrypted using A's public key, then agent A can know the content of message m. The steps of proving the lemma 2 are as follows:

1. $Know(s_j, A, m)$	Consequent
2. $i \preccurlyeq j$	assumption
3. $Know(s_j, A, Encrypt\ (m, Key(Kpb(A)))) \wedge Know(s_j, A, Key(Kpr(A)))$	1,2, rule 2
4. $Know(s_j, A, Key(Kpr(A)))$	3, conjunction introduction rule

5. $subgoal\ 2\ proved$	4, assumption3
6. $Know(s_j, A, Encrypt\ (m, Key(Kpb(A))))$	3, conjunction introduction rule
7. $done$	

4.4. Proving security properties

Proving the correctness of the NSPK protocol will be based on the nonce secrecy. If agent A and agent B have successfully completed a run of the protocol, then A should believe that B is his partner if and only if B believes that he is talking to A. So, there are two properties, which are important to prove the correctness of the NSPK protocol:

A) $Attacker\ not\ know\ A's\ Nonce$:

$\neg\ Know(Attacker, Nonce(A))$.

B) $Attacker\ not\ know\ B's\ Nonce$:

$\neg\ Know(Attacker, Nonce(B))$.

With the assumption that nonces will never be sent out over the network without encryption, the attacker does not have the opportunity to know the value of the nonces unless somebody sends the nonces encrypted by the attacker's public key.

$Attacker\ known\ the\ nonce$: $Know(Attacker, Nonce(Friends)) \longrightarrow Send(X, Attacker, Encrypt(Nonce(Friends), Key(Kpb(Attacker))))$

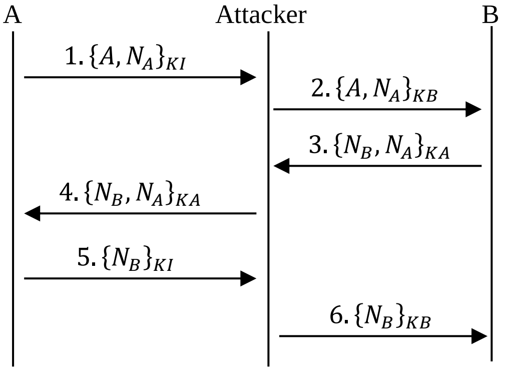

Figure1. Attack NSPK protocol

Lowe Attack: Lowe (1995) found that in step 3 of the protocol there is a potential attack. The figure 1 shows the six steps to break the protocol [13]. If we assume agent A sends B's nonce to the attacker, then by reviewing the steps of the protocol backwards we can easily find the attacker can be impersonating another principal (agent B) and illegally knows the value of B's nonce. As shown in Table1 at session 1, step 3, agent A has sent B's nonce to the attacker, before session 2 has been completed. Therefore, the attack breaks the secrecy of the nonces.

Table1. Attacking the Needham-Schroder protocol by Lowe.

Session	Step	Sent to	Message
Seession1	Step1	$A \longrightarrow I$	$\{N_A, A\}_{KA}$
Seession2	Step1	$I(A) \longrightarrow B$	$\{N_A, A\}_{KB}$
Seession2	Step2	$B \longrightarrow I(A)$	$\{N_B, N_A\}_{KA}$
Seession1	Step2	$I \longrightarrow A$	$\{N_B, N_A\}_{KA}$
Seession1	Step3	$A \longrightarrow I$	$\{N_B\}_{KI}$
Seession2	Step3	$I(A) \longrightarrow B$	$\{N_B\}_{KB}$

The scripts show the attack can know B's nonce before agent B as follows:

1. $Know(s_j, Attacker, Nonce(B))$	Consequent
2. $s_i \preccurlyeq s_j$	Assumption
3. $Know(s_j, Attacker, Encrypt\ (Nonce(B), Key(Kpb(Attacker))$	1, lemma 2
4. $Send(s_i, A, Attacker, Encrypt\ (Nonce(B), Key(Kpb(Attacker))$	2,3, lemma1
5. $s_d \preccurlyeq s_i$	assumption
6. $Send(s_d, Attacker, A, Encrypt(\ \langle Nonce(A), Nonce(B)\rangle, Key(Kpb\ (A))))$	4, 5, Reply NS3 to Attacker
7. $s_e \preccurlyeq s_d$	assumption
8. $Know(s_e, Attacker, Encrypt(\ \langle Nonce(A), Nonce(B)\rangle, Key(Kpb\ (A))))$	6,7, rule 5
9. $s_g \preccurlyeq s_e$	assumption
10. $Send(s_g, B, Attacker, Encrypt\ \langle Nonce(A), Nonce(B)\rangle, Key(Kpb\ (Attacker))))$	8,9, lemma 1
11. $s_h \preccurlyeq s_g$	assumption
12. $Send(s_h, Attacker, B, Encrypt(\langle Agent(A), Nonce(A)\rangle\ , Key(Kpb(B))))$	10,11, Reply NS2 to Attacker
13. $Know(s_h, Attacker, Encrypt(\ \langle Agent(A), Nonce(A)\rangle, Key(Kpb\ (B))))$	12, fake message
14. $Know(s_h, Attacker, \langle Agent(A), Nonce\ (A)\rangle) \wedge Know\ (s_h, Attacker, Key(Kpb\ (B)))$	13, rule 1
15. $Know\ (s_h, Attacker, Key(Kpb\ (B)))$	14, conjunction introduction rule
16. $Subgoal\ 2\ proved$	15, assumption2
17. $Know(s_h, Attacker, \langle Agent(A), Nonce\ (A)\rangle)$	14, conjunction introduction rule
18. $Know(s_h, Attacker, Encrypt(\ \langle Agent(A), Nonce\ (A)\rangle, Key(Kpb\ (Attacker))))$	15, lemma 2
19. $s_x \preccurlyeq s_h$	assumption
20. $Send(s_x, A, Attacker, Encrypt\ \langle Agent(A), Nonce(A)\rangle, key(Kpb\ (Attacker))))$	18, 19, lemma1
21. $done.$	

The framework has used to analyse the NSPK protocol. The ability of the framework demonstrated by detects the Lowe attack. Also, there are two basic properties have proved and used in the analysis of

NSPK protocol. Hence, the framework can analyse complex protocols to discover a new flaws.

5. The Andrew secure RPC protocol

The Andrew Secure RPC protocol (ASRPC) aims to authenticate handshake among two agents. This protocol purposes to provide the client A which a new session Key K'_{ab} from the server B, whereas both A and B have already had a shared session key K_{ab} [14]. The four steps of the ASRPC protocol can be represented as following:

1. $A \rightarrow B$	$A, \{N_a\}_{K_{ab}}$
2. $B \rightarrow A$	$\{N_a + 1, N_b\}_{K_{ab}}$
3. $A \rightarrow B$	$\{N_b + 1\}_{K_{ab}}$
4. $B \rightarrow A$	$\{ K'_{ab}, N'_b\}_{K_{ab}}$

The messages of ASRPC can be described as follows:

Message 1: The agent A initiates the protocol by sending to server B a message containing A's identity unencrypted and A's nonce N_a encrypted with the share session key K_{ab}.

Message 2: If B receives message 1, B can know N_a by decrypting the message. Then, B responds to A by sending a message encrypted with K_{ab} containing the nonce N_a and a new nonce N_b ,which is generated by B.

Message 3: If A receives message 2, A can know N_b by decrypting the message. Then, A responds to B after it checks and is satisfied with content of message 2 by sending a message encrypted with K_{ab} containing nonce N_B.

Message 4: If B receives message 3, B will send a new session key K'_{ab} with B's new nonce N'_b by a message encrypted with K_{ab}.

After successfully running the protocol, the client A can be sure that he or she is authenticated by server B. Also, a fresh new session key K'_{ab} can be used to exchange the data with server B. However, Burrows et al in 1989 found that the client A cannot ensure that the $K'_{ab}$is fresh [14]. In addition, Clark and Jacob proposed a typing attack in which an intruder eavesdrop the message 2 then resend it as substitutes in place of message 4. The Clark and Jacob attack is shown in next section.

6. Analysis of the Andrew Secure RPC protocol

As we have done in the section 4, The framework in section 2 will be used to analyse ASRPC protocol and find the Clark and Jacob attack. The steps of guidelines for proving the correctness of security protocols will be followed.

6.1. Adjusting the framework

In the ASRPC protocol there are two honest agents ($client\ A$ and server B) and an attacker. Hence, the type of agents can be defined as follows:

$$Agents \equiv A \mid B \mid Attacker$$

And there are three different nonces which are N_a for client A and N_b and N'_b for server B. Moreover, in this protocol there are two session keys which are K_{ab} and K'_{ab}.

6.2. Modelling the ASRPC protocol

The four steps of the ASRPC protocol will be converted from an informal language as written in section 5 to a formal language using the frameworks notations as follows:

ASRPC 1: $\mathrm{Send}(\mathrm{A}, \mathrm{B}, \langle A, Encrypt(N_a, Key(Ksym\ (A, B, n)))\rangle)$

ASRPC 2: $\mathrm{Send}(\mathrm{A}, \mathrm{B}, \langle A, Encrypt(N_a, Key(Ksym\ (A, B, n)))\rangle) \Longrightarrow \mathrm{Send}(\mathrm{B}, \mathrm{A}, \mathrm{Encrypt}\ (\langle N_a + 1\ , N_b\rangle, Key\ (Ksym\ \ (A, B, n))))$.

ASRPC 3: $\mathrm{Send}(\mathrm{B}, \mathrm{A}, \mathrm{Encrypt}\ (\langle N_a + 1\ , N_b\rangle, Key\ (Ksym\ \ (A, B, n)))) \Longrightarrow \mathrm{Send}(\mathrm{B}, \mathrm{A}, \mathrm{Encrypt}\ (\langle N_b + 1\ \rangle, Key(Ksym\ (A, B, n))))$.

ASRPC 4: $\mathrm{Send}(\mathrm{B}, \mathrm{A}, \mathrm{Encrypt}\ (\langle N_b + 1\ \rangle, Key\ (Ksym\ \ (A, B, n)))) \Longrightarrow \mathrm{Send}\ (\mathrm{B}, \mathrm{A}, \mathrm{Encrypt}\ (\langle\ (Ksym'\ (A, B, n)), N\ '\ _b\ \rangle, Key(Ksym\ (A, B, n))))$

These four steps are able to achieve the aim of protocol carrying out authentication handshake among two agents and agreement in a new session key $\boldsymbol{K'_{ab}}$. On the other hand, it should not be overlooked that the attacker does not necessarily follow the protocol rules. According to the assumptions and rules in the framework, the attacker can eavesdrop all messages and the fake message rule can be used:

Eavesdrop attacker ASRPC 2: $\mathrm{Send}(\mathrm{B}, \mathrm{A}, \mathrm{Encrypt}\ (\langle N_a + 1\ , N_b\rangle, Key\ (Ksym\ \ (A, B, n)))) \Longrightarrow\ Rcv\ (Attacker, Encrypt(\langle N_a + 1\ , N_b\rangle, Key(Ksym\ \ (A, B, n))))$

6.3. Proving basic properties

The basic properties are usually suitable to be used within various protocols. Therefore, we will use the basic properties in the section 4.3.

6.4. Proving security properties

Proving the correctness of the ASRPC Protocol will be based on executing the successful handshake and agreeing the new session key K'_{ab}. If client A and server B have successfully completed a run of the protocol, then A should believe that new key session K'_{ab} is fresh and client A will use this key for next session. So, there is a property which can be used to prove the correctness of the NSPK protocol:

$Client\ A\ know\ fresh$ new key session K'_{ab}:
$Know\ (A, Encrypt(\langle N'_b, K'_{ab}\rangle, Key(Ksym\ (A,B,n)))\ )$

We assume that the client A and server B share session key K_{ab}, and all messages are never sent out without encrypted by K_{ab} over the network. With these assumptions the attacker has no chance to know or modify the value of new session key K'_{ab}. However, the attacker can send a fake message which has same format of a message as the new session key K'_{ab} to client A. In this case, the attacker can bogus the K'_{ab} and convince the client A to accept it [1,15].

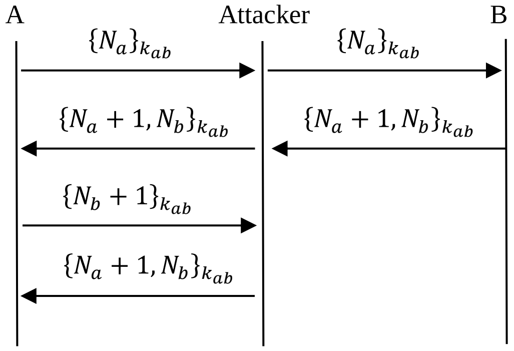
A
Attacker
B
$\{N_a\}_{k_{ab}}$
$\{N_a\}_{k_{ab}}$
$\{N_a+1, N_b\}_{k_{ab}}$
$\{N_a+1, N_b\}_{k_{ab}}$
$\{N_b+1\}_{k_{ab}}$
$\{N_a+1, N_b\}_{k_{ab}}$

Figure 2. Attack ASRPC protocol

Claek-Jacob Attack: The ASRPC protocol has a potential attack found in 1996 by Claek and Jacob. The figure 2 shows the sequence steps to attack the protocol. If we assume that the attacker is able to send a fake message and eavesdrop all messages, the attacker can impersonate server B and reply the message 2 to client A when the client A sends the message 3 to server B. At the end, client A accepts the N_a+1 to be the new session key with server B[15].

The scripts below show the attack has ability to achieve Clark-Jacob attack through impersonate server B and replying message 2.

1. $Know(s_i, A, \langle N_a+1, N_b\rangle)$	Consequent
2. $s_j \preccurlyeq s_i$	Assumption
3. $Know(s_i, A, Encrypt(\langle N_a + 1, N_b\rangle, Key\ (Ksym\ (A,B,n)))) \wedge Know\ (s_i, A, Key(Ksym(A,B,n)^{-1}))$	1, rule 2
4. $Know(s_i, A, Key(Ksym(A,B,n)$	3, conjunction introduction rule
5. $Subgoal\ 2\ proved$	4, assumption 6
6. $Know(s_i, A, Encrypt\ (\langle N_a+1, N_b\rangle, Key\ (Ksym\ (A,B,n))))$	3, conjunction introduction rule
7. $Send(s_j, Attacker, A, Encrypt\ (\langle N_a+1, N_b\rangle, Key\ (Ksym\ (A,B,n))))$	2,6, lemma 1
8. $s_k \preccurlyeq s_j$	assumption
9. $Know(s_k, Attacker, Encrypt\ (\langle N_a+1, N_b\rangle, Key\ (Ksym\ (A,B,n))))$	7,8, fake message
10. $s_n \preccurlyeq s_k$	*assumption*
11. □ Know(s_n, Attacker, Encry (⟨N_a + 1, N_b ⟩, Key(Ksym (A,B,n))))	9,10, always elimination rule
12. s_m ≼ s_n	assumption
13. $Rcv(s_m, Attacker, Encrypt\ (\langle N_a+1, N_b\rangle, Key\ (Ksym\ (A,B,n))))$	11,12, rule 7
14. $Send(s_m, B, A, Encrypt\ (\langle N_a+1, N_b\rangle, Key(Ksym\ (A,B,n))))$	13,Eavesdrop attacker ASRPC 2
15. $s_k \preccurlyeq s_m$	assumption
$Send\ (s_k, A, B, Encrypt(N_a, Key\ (Ksym\ (A,B,n))))$	14,15, ASRPC 2
done	

7. Conclusion and future work

In this paper, a framework was presented that can be used to analyse security protocols. The framework approach is linear temporal logic with statuses, which is used to prove the correctness of security protocols. The NSPK protocol and ASRPC protocol, which are well known security protocols used to prove that the framework is capable of detecting flaws. The result of the proof is that the framework detected the known flaws in these two protocols.

Future work will focus on investigating other protocols using the framework in order to identify the unknown flaws.

8. Acknowledgements

This research is supported by Saudi Arabian Cultural Bureau in London, and the Ministry of Higher Education in Saudi Arabia and Qassim University.

9. References

[1] Boyd, C.; Mathuria, A. *Protocols for Authentication and Key Establishment.*; Springer Verlag, 2003.

[2] Dixon, C.; Gago, M.C.F.; Fisher, M.; Van Der Hoek, W. Using Temporal Logics of Knowledge in the Formal Verification of Security Protocols. In Temporal Representation and Reasoning, 2004. TIME 2004. Proceedings. 11th International Symposium on; pp. 148-151.

[3] Zhang, Y.; Varadharajan, V. A Logic for Modeling the Dynamics of Beliefs in Cryptographic Protocols. In Computer Science Conference, 2001. ACSC 2001. Proceedings. 24th Australasian; pp. 215-222.

[4] Snekkenes, E. Exploring the BAN Approach to Protocol Analysis. In Research in Security and Privacy, 1991. Proceedings., 1991 IEEE Computer Society Symposium on; pp. 171-181.

[5] Syverson, P.F. Knowledge, Belief, and Semantics in the Analysis of Cryptographic Protocols. journal of Computer Security **1992**, *1*, 317-334.

[6] Syverson, P.F.; Van Oorschot, P.C. On Unifying some Cryptographic Protocol Logics. In Research in Security and Privacy, 1994. Proceedings., 1994 IEEE Computer Society Symposium on; pp. 14-28.

[7] Manna, Z.; Pnueli, A. *The Temporal Logic of Reactive and Concurrent Systems: Specification*.; Springer, 1992.

[8] Lei, X.; Xue, R.; Yu, T. A Timed Logic for Modeling and Reasoning about Security Protocols.

[9] Ebbinghaus, H.-.; Flum, J.; Thomas, W. *Mathematical Logic*, 2nd ed.; Springer-Verlag: New York; London, 1994; pp. 289.

[10] Simpson, A.K. The Proof Theory and Semantics of Intuitionistic Modal Logic. Department of Computer Science, University of Edinburgh, Edinburgh, Scotland **1994**.

[11] Shigong, L.; Lijun, W. Analysis of Cryptographic Protocols using LTL of Knowledge. In Networking and Digital Society (ICNDS), 2010 2nd International Conference on; pp. 463-466.

[12] Ma, X. A Knowledge Based Approach to Verifying Cryptographic Protocols. **2007**.

[13] Lowe, G. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. Information processing letters **1995**, *56*, 131-133.

[14] Burrows, M.; Abadi, M.; Needham, R.M. A Logic of Authentication. Proceedings of the Royal Society of London.A.Mathematical and Physical Sciences **1989**, *426*, 233-271.

[15] Clark, J.; Jacob, J. Attacking Authentication Protocols. High Integrity Systems **1996**, *1*, 465-474.