Formalization of Blind Signature using the Inductive Method

N. S. Miramirkhani and R. Jalili

Abstract— **Verification of a security protocol can be done using model checking or theorem proving approach. Unlike model checking, theorem proving methods can prove security properties of a protocol under an unlimited number of agents participating in the protocol and interleaving an infinite number of protocol sessions. But these methods usually suffer from not being completely automatic. However, the Inductive Method is a pure theorem proving approach that has been successful in analyzing many classic to real world protocols such as SET. This method has been evolved and extended during years to be able to verify more complicated protocols such as e-commerce, smart cards, non-repudiation and multicast based protocols. Blind signature is an advanced cryptographic primitive vastly used in protocols with a major concern on anonymity such as e-voting protocols. Using the great potential of the Inductive Method in formalizing new concepts, we have extended the Inductive Method to make it support modeling this concept. Moreover, we have proved that our model meets the desired requirements and preserves the properties of already defined elements of the Inductive Method.**

Index Terms— **Formal Verification, Inductive Method, Blind Signature, Theorem Proving**

I. Introduction

THE two main approaches to the formal verification of systems and especially security protocols, are based on model checking and theorem proving . These two approaches originate from the two different insights into the proofs in a formal system. Each formal system has a syntactic and semantic structure and a deduction system.

In general, a model checking problem is a problem of verifying that a formula Φ holds in a model M: M |= Φ, where M represents the protocol model and is usually finite, and Φ is the desired security property expressed in a temporal logic. Model checking techniques are based on the exhaustive state traversal of the model, and are often automatic and always terminate. The model in these techniques is often restricted to a finite state transition system, but generally the problem of model checking is not limited to finite transition systems.

This work was supported in part by Amnafzar Gostar Sharif Co.

N. S. Miramirkhani is a Senior Penetration Tester at Consulting Group of Amnafzar Gostar Sharif Co, Tehran, Iran. (e-mail: miramirkhani@amnafzar.com).

R. Jalili is an Associate Professor in the Department of Computer Engineering at Sharif University of Technology, Tehran, Iran. (e-mail: jalili@sharif.edu).

A theorem proving problem, on the other hand, is a problem of checking the validity of a formula Φ, that is, Φ must hold in every model. The logic is often much more expressive than propositional temporal logic often used in model checking, and the properties over infinite domains can be easily expressed. A theorem proving problem is solved by finding a proof of the formula (which is then called a theorem) in a proof system. Since most of the formal systems are undecidable, theorem proving tools are not automatic and they usually need the interaction of an expert user.

The Inductive Method is one of the notable works in the verification of systems relying on theorem proving. This method supported by the interactive theorem prover Isabelle/HOL is used to verify the security properties of different classic [3], [4] and real world protocols such as SET [12]-[14]. Its main underlying idea is that simple mathematical induction suffices to model security protocols and reason about their goals.

During the last decade, this method has been evolved and extended in different ways to be able to deal with different security protocols and more advanced goals. Since blind signature is an advanced cryptographic concept which is used in different security protocols such as e-voting protocols to preserve privacy of agents, we have found the opportunity of extending the Inductive Method to support this concept. Moreover, we prove a series of theorems which show that our model meet the desired requirements and expected behaviors and preserves the properties of already defined components of the Inductive Method.

Our idea in this paper is to briefly review the extensions to the Inductive Method and to motivate our augmentation of the Inductive Method with a blinding signature extension. Then we will have a review on the key concepts of the method used in the following sections. After that, we will see our contributions in the extension of the inductive method towards supporting formalization of the blind signature schema. We will also prove that our model is compatible with the threat model and already defined concepts of the method.

Isabelle/HOL is a free interactive theorem prover released under the revised BSD license. Proofs of this study are mechanized by Isabelle/HOL, but for the sake of simplicity, theorems, lemmas and definitions are presented in a simplified syntax.

II. RELATED WORK

The foundations of the Inductive Method are due to Paulson [1]. He applied the method to some classical protocols such as Needham-Schroeder, Otway-Rees and Yahalom [3], [4]. Since then, the inductive method has been extended in various aspects in order to be able to model other security protocols.

The method was first extended with the treatment of timestamps which allows modeling and verifying security properties of timestamp-based protocols such as the BAN Kerberos, Kerberos IV and V [5]-[7].

Later a third event to model message reception was introduced. Since the event of message reception [8] is not forced to occur, this models a network that is entirely controlled by an active Spy who can intercept certain messages and prevent their delivery. Moreover, the reception event allows the formalization of any agent's knowledge [9]. Introducing message reception provided a more realistic formalization of security goals and a higher level of formal expressiveness. Using this concept, protocol model for BAN Kerberos, Kerberos IV and V was updated and expressed in a simpler form.

By formalizing smartcards and their interactions with card owners, e-commerce protocols such as Shoup-Rubin could be verified [10].

After that, the threat model of the inductive model was changed to support formalization of accountability protocols like Zhou-Gollmann's non-repudiation protocol and Abadi's certified e-mail protocol [11]. Finally, by maturing this method, the e-commerce protocol SET was verified [12]-[14].

Recently, another extension to the Inductive Method has been introduced that makes it enable to reason about the multicast-based security protocols.

The above researches confirm the great potential of the inductive method to formalizing new concepts. As blind signature is an advanced cryptographic primitives vastly used in protocols with a major concern on anonymity, we decided to provide its formalization in the Inductive Method.

III. INDUCTIVE METHOD

The Inductive Method is a pure theorem proving method which uses simple mathematical induction to model security protocols and reason about their goals. A key concept here is the trace, a list of network events occurring while an unbounded population of agents is running a protocol.

Traces are defined inductively and so is the set of all traces admissible under a specific protocol. This set represents the formal protocol model. Proofs can be carried out by induction on a generic trace of the model, with mechanical support offered by the theorem prover Isabelle [2]. They establish trace properties representing goals of the underlying protocol.

In the following subsections, we will review briefly some key concepts of the method used in our model or theorems:

A. The Structure of Cryptographic Keys

The free type *key* is introduced to formally represent all types of cryptographic keys. The function *invkey* is defined over the set of keys and leaves a key unaltered in case it is symmetric; otherwise, it turns the key into its corresponding asymmetric half.

In the asymmetric-cryptography setting, each agent is endowed with the asymmetric key pair public and private keys. These keys are correspondingly modeled by the *priK* and *pubK* functions from the set of agents to the set of keys.

The set *symKeys* formalizes symmetric keys. According to its definition, a key is an asymmetric key if and only if it has the same decryption and encryption keys [1]:

$$\text{Set key}: \ \text{symKeys} = \{\text{key } k.\ K = \text{invKey}(k)\} \tag{1}$$

B. Agents and Threat Model

A security protocol is a sequences of steps designed to achieve certain goals at the time of its completion. Broadly speaking, security protocols aim to provide secure communication through an unsecure network. The network may contain a *Spy* who can intercept all messages and prevent their delivery. He can also tamper with them by decomposing concatenated messages and opening up ciphers sealed with keys he knows. Then, he can use the learnt message components to form new messages at will by concatenation and encryption. He can also collude with some agents and knows their long term secrets. However the capabilities of the *Spy* are specified in the threat model in which the protocol is studied.

Steps of a protocol are executed by the agents who run the protocol. Three types of agents are defined in the Inductive method: *Server*, *Spy* and *Friend i*. *Server* models the trusted third party required in most symmetric cryptography protocols, *Spy* indicates the adversary in the environment and *Friend i* formalizes an unlimited number of friendly agents indexed by natural numbers *i*.

The threat model of the Inductive Method is an extension of Dolev-Yao's threat model. The *Spy* in this threat model, in addition to the capabilities of a Dolev-Yao's adversary can collude with a set of compromised agents. The set of compromised agents is specified by the set *bad*. Definition of the *bad* set is as follows:

$$\text{Set agent}: \text{bad} = \{\text{Spy}\} \cup \{\text{agent } a.\ a \neq \text{Server}\} \tag{2}$$

Since the *Spy* knows his own long-term secrets, he belongs to this set. In addition, Trusted servers are always assumed to be uncompromised, namely not to belong to the *bad* set.

C. Messages

Running each protocol step requires the transmission of a message, possibly encrypted, between the peers. Messages may include various components such as cryptographic keys, timestamps and ciphers.

In the Inductive Method, seven types of messages are defined: agent names, guessable numbers (modeling

timestamps), random numbers (modeling nonces), cryptographic keys, hashed messages, concatenated messages and ciphers. While the first four message types are the basic types, the three later types are recursive constructors.

An important point about encrypted messages is that if two ciphers are equal, then the corresponding clear texts would be equal too and the corresponding encryption keys are also the same. Message types are specified as follow:

```
Type msg = Agent  agent
         | Number natural
         | Nonce  natural
         | Key    key
         | Hash   msg
         | {msg, msg}
         | Crypt  key msg
```
(3)

D. Message Operators

The three operators *parts*, *synth*, and *analz* are introduced to manage sets of messages and implementing the *Spy*'s capabilities. The operator *parts* extracts all components from a given set of messages by decomposition and decryption. The operator *synth* formalizes message creation from known components, and the *analz* operator formalizes the act of breaking up messages into their components by decomposition and decryption where the decryption key is available. Inductive definition of the *analz* operator is defined as follows:

```
Inductive Set : analz :: Set msg => Set msg
 for H :: Set msg
    X ∈ H ==> X ∈ analz H
 | {|X,Y|} ∈ analz H ==> X ∈ analz H
 | {|X,Y|} ∈ analz H ==> Y ∈ analz H
 | Decrypt : [|Crypt K X ∈ analz H; Key (invKey K) ∈ analz H|]
          ==> X ∈ analz H
```
(4)

It can be observed that the *Decrypt* rule can obtain the clear text of a cipher in the message set H only if its decryption key belongs to H. On the other hand, based on this definition, encryption is assumed to be perfect.

E. Traces and Events

A history of the network traffic can be represented by the list of the events that occurred throughout that history. Such a list is called a trace. The set of all possible traces of unbounded length is a formal model of the protocol.

The Inductive Method allows for three types of events formalizing the acts of sending, noting, or receiving a message (Says, Notes, and Gets).

F. Agent's Knowledge of Messages

Agent's knowledge of messages is formalized by the agent's static knowledge, namely what he knows before any protocol session takes place. The later, formalizes the agent's dynamic knowledge, namely what he obtains from inspecting protocol traces.

Agent's initial (static) knowledge consists of all the messages, long-term secrets and keys the agent knows before engaging in any sessions of the protocol. For example, a friendly agent knows all public keys and only its own private key. The *Spy* knows all compromised agents' private keys (he is assumed to collude with compromised agents), and all public keys.

Agent's dynamic knowledge of messages contains messages that he acquire from a protocol trace. So the set of all message components that the *Spy* can derive from the observation of the traffic in the trace evs can be modeled by *analz*(*knows Spy evs*).

At the beginning of the protocol when the trace is empty, all the agents only know their own initial knowledge. During the protocol execution, the agents may dynamically expand their knowledge based on the events occurred in the network. For example, any messages ever sent by anyone and any messages ever noted by a compromised agent belong to the *Spy*'s dynamic knowledge but other agents only know what they send to anyone, what they receive and what they note in a trace.

G. The Used Operator

The *used* operator formalizes the set of all message components that either belong to some agent's initial knowledge or appeared on some trace. So freshness of a message *m* in a trace *evs* can be formalized by not belonging *m* to the set of *used evs.*

IV. BLIND SIGNATURE

Blind signature requires that a signer be able to sign a document without knowing its contents. This intuitively corresponds to signing a document with the eyes closed. In security protocols that anonymity of the sender is a major concern, blind signature can be employed. Two such examples are e-voting and electronic cash. To illustrate the concept, we look at the voting scheme in more detail.

In an e-voting protocol, on one hand, the voting center will be concerned of duplicate or fraudulent votes, and on the other hand, the voter will be concerned about the anonymity of his vote. A solution can be to have a trusted Election Authority that authenticates the owner of a vote, namely it is responsible for checking validity and non-duplicity of votes. Then the voting center accepts any votes with the signature of the Election Authority.

The concern of the voter can be solved using blind signatures. The Election Authority should be able to sign on a blinded vote sent by the voter and the voter should be able to extract a valid signature on the original unblinded vote.

Precisely speaking, any blind signature schema is a security

the one who requests for signing a message. For example a simplified version of the FOO protocol which is an e-voting protocol based on blind signature is as follows [15]:

1) $V \rightarrow A : V, \sigma V(\chi(v, b))$
2) $A \rightarrow V : \sigma A(\chi(v, b))$
3) $V: \sigma A(v)$

In the first step, a voter V selects a vote and blinds it by a blinding function χ and a random blinding factor b. then he sings the blinded message and sends it beside his id to the authority A.

In the second step, A investigates whether V has the right to vote, it has not voted yet and his signature is valid. If all of these conditions hold, A signs the blinded message and sends it back to the voter.

Finally, V unblinds the received message and obtains the vote signed by A.

V. FORMAL MODEL OF BLIND SIGNATURE

The inductive method previously allowed only three types of keys: asymmetric keys, symmetric keys and session keys. We formalize blind signature as a symmetric encryption that the symmetric key is not shared with any other agents, namely it is exclusive. In other words, only the owner of the blinding factor can blind and unblind a message using his blinding factor. Another issue that should be addressed is that blinding factor is assumed to be unique.

In order to model blind signature, we define the function *blindFac* from the set of agents to the set of keys (5) that maps each blinding factor into its owner. To address the uniqueness property of the blinding factors, we specify that the *blindFac* function is injective (5).

DEF1: Function blindFac : "agent => Key"
blindFac is injective (5)

Using Axiom 1, we define that blinding factor is a symmetric key, so it has the same encryption and decryption keys and all the theorems and lemmas related to the symmetric keys hold for blinding factors too. Consequently based on the perfectness of encryption, no one can reveal the contents of a blinded message.

Axiom 1 : blindFac X $\in$ symKeys (6)

The only issue which remains is to specify that blinding factor is exclusive. We will meet this requirement in the next section where we want to define the agents' static knowledge.

Based on this formalization, the message M blinded by the agent A's blinding factor can be modeled by *Crypt*(*blindFac* A)M. Then the blinded message can be signed by the *priK* function mentioned before.

VI. STATIC KNOWLEDGE OF AGENTS

As mentioned in the previous section, blinding factor of an agent is not shared with any other agents. We redefine the *initState* function which formalizes the static knowledge of agents, in such a way to address this property.

Moreover, threat model of the Inductive model allows the *Spy* to collude with the set of compromised agents. To remain consistent with this model, compromised agents should reveal their blinding factors to the *Spy* since the beginning of protocol execution. So the *Spy*'s initial knowledge should contain the blinding factor of *bad* agents.

Concluding above requirements, The *Spy* knows the blinding factors of compromised agents, private keys of bad agents and all the public keys. Friendly agents and the *Server* knows its own blinding factor, his private key and all the public keys.

DEF2: Function initialState:
initState_Server: Key (All pubK $\cup$ priK server $\cup$ blindFac Server)
| (7)
initState_Friend i: Key (All pubK $\cup$ priK Friend i $\cup$ blindFac (Friend i))
|
initState_Spy: Key (All pubK $\cup$ priK bad $\cup$ blindFac bad)

Now we move on to prove some lemmas that are used in the following sections. Lemma1 states that the Spy knows the blinding factor of all the bad agents. This lemma can be easily derived from DEF2.

lemma 1: A $\in$ bad ==> Key (blindFac A) $\in$ initialState Spy (8)

Lemma 2 states that each agent knows his own blinding factor before any protocol session takes place. If the agent is a friendly agent or the *Server*, it can be easily proved using DEF2, else if the agent is the *Spy*: he knows blinding factors of *bad* agents and according to the definition of *bad* set, the *Spy* is a member of this set, so he knows his key.

lemma 2: Key (blindFac A) $\in$ initialState A (9)

VII. THEOREMS OF THE USED FUNCTION

The *used* function, formalizes the set of all message components that are used in a trace. At the beginning of protocol execution, the trace is empty; therefore the set of used messages is equal to the set of *initState* of all the agents. So we expect that blinding factor of each agent belongs to the set of used messages (10).

lemma 3: " Key (blindFac A) $\in$ used evs" (10)

The lemma can be proved using (11), (12). Since the blinding factor of agent A belongs to the set of A's *initState* message components (11) and the set of his *initState* messages

is a subset of *used* messages (12), blinding factor is a member of used messages.

lemma 4 : Key (blindFac A) ∈ parts(initState A) (11)

lemma 5: initState A ⊆ used evs (12)

VIII. COMPATIBILITY WITH THE INDUCTIVE METHOD KEYS

Blinding factor is introduced as a new key type so it is necessary to prove that the set of *blindFac* keys are completely disjoint from any other keys previously modeled in the Inductive Method. In particular, we should prove there are not any equal blinding factor and asymmetric key. Formally speaking we should prove that:

Theorem 1: blindFac A ≠ priK C (13)

The proof can be done using an axiom in the theory of asymmetric keys which states that *priK* ≠ *pubK*, the definition of *invkey* and *symKeys*, and the axiom1.

According to the definition of the *invkey* function, in case of asymmetric keys, it turns the key into its corresponding asymmetric half. Using the axiom in the theory of asymmetric keys, we can conclude that *invkey* of an asymmetric key is not equal to that key. But the basic property of the key K as a symmetric key is that *invkey* (k) = k. Therefore the range of symmetric keys and asymmetric keys are disjoint. According to Axiom1, blinding factor is a symmetric key, consequently we can prove that its range is disjoint from asymmetric keys and it differs from any public and private keys.

Having proved theorem 1, the following theorem can be proved easily and theorems related to the *priK* can also be proved similarly.

Theorem 2: pubK A ∉ Range (blindFac AA) (14)

Another key type that is formalized in the inductive method is the session key. A session key is a fresh symmetric key that some protocols distribute it to their peers. By the word fresh, it means that the key has not been used in the current trace of the protocol. Because the blinding factor is also a symmetric key, it is essential to prove that the range of blinding factors and session keys are disjoint. The following theorem states it formally:

Theorem 3: Key K ∉ used evs ==> K ∉ range blindFac (15)

This theorem is proved using lemma3 which states that a blinding factor is in the set of used messages for any trace of the protocol so it is also used for the trace *evs* and it differs from session keys.

IX. COMPATIBILITY WITH THE THREAT MODEL

In this section we provide some theorems to show that the formalization of the blind signature is compatible with the threat model and satisfies the properties and behaviors expected.

The function *analz* is particularly important in formalizing the threat model of the inductive method. It represents a potentially unlimited series of decomposing concatenated messages and decryptions by available keys and as already mentioned *analz*(*knows Spy* evs) represents the set of all message components that the *Spy* can derive from the network traffic.

As the *Spy* knows the blinding factor of compromised agents, it can be proved that he can obtain components of the observed messages blinded by the compromised agents.

Formally speaking, if A is a compromised agent, and the message blinded by his blinding factor belongs to the set of message components analyzed by the *Spy* from the trace evs, then the unblinded message is also available to the Spy :

Theorem 4: [| Crypt (blindFac A) X ∈ analz (knows Spy evs) ; A ∈ bad |] ==> X ∈ analz (knows Spy evs) (16)

In order to prove the above theorem, we should first prove simpler theorems and lemmas. The theorem 5 is a simpler version of theorem4 which states that if the *Spy* has the blinding factor, he can decrypt the blinded message. Specifically, if the blinding factor and the message blinded by it belong to the set of messages H, then the *Spy* can recursively uses the operator *analz* to obtain the unblinded message X:

Theorem 5 : [| Crypt (blindFac A) X ∈ analz H; Key(blindFac A) ∈ analz H |] ==> X ∈ analz H (17)

According to the definition of the *analz* operator, ciphers can be decrypted if and only if the decryption key is available. If we prove that the decryption key of a blinding factor is equal to it, based on the assumptions of theorem 5, we can conclude that the decryption key is available and so prove theorem5. The issue is formally stated in the following lemma:

lemma 6: invKey (blindFac A) = blindFac A (18)

The proof of lemma6 is straightforward using axiom1 and the definition of symmetric keys.

We observe that the *Spy*'s knowledge is omitted in the theorem4.In the next session we will investigate the Spy's knowledge in more detail and find some lemmas which link between this theorem and the *Spy*'s knowledge.

X. AGENT'S DYNAMIC KNOWLEDGE

The function *knows* recursively represents how the knowledge of each agent (the *Spy* included) is expanded during the protocol execution, namely formalizes the dynamic knowledge of agents.

Prior to the protocol execution, the trace of events is empty

and the knowledge of an agent is equal to its initial knowledge. During the protocol execution, based on the events occurred in the network and the agent who causes the event to occur, dynamic knowledge of agents maybe added. It can be proved by induction that the set of agent's initial knowledge in a trace is a subset of his dynamic knowledge on that trace:

$$\text{lemma 7: initialState A} \subseteq \text{Knows A evs} \tag{19}$$

By introducing the blinding factor and consequently extending the initial knowledge of agents, we proved in lemma2 that the blinding factor of an agent belongs to his initial knowledge. So using lemma6 and lemma2, it is proved that the blinding factor of an agent belongs also to his dynamic knowledge for any trace evs:

$$\text{Theorem 6: Key (blindFac A)} \in \text{knows A evs} \tag{20}$$

Similarly we expect that the *Spy* knows the compromised agents' blinding factors in any trace of evs. This requirement is expressed formally in the following lemma:

$$\begin{aligned}&\text{lemma 8: A} \in \text{bad}\\ &\quad \Longrightarrow \text{Key (blindFac A)} \in \text{knows Spy evs}\end{aligned} \tag{21}$$

The lemma can be proved by induction on the length of the trace evs: For the base case when the trace is empty, according to the definition of the function *knows*, dynamic knowledge of an agents is equal to his initial knowledge and based on lemma1, if the agent A is a compromised agent, then the *Spy* knows the A's blinding factor in any trace evs.

For the inductive steps, if the trace has been extended by any event, then the set of *Spy*'s dynamic knowledge is a super set of his previous knowledge set. Then using the base case, compromised agents' blinding factor also belongs to this expanded knowledge set of the *Spy*.

Now, let's move on to prove the theorem4. If we replace the message set H by the term *knows Spy* evs, a variant of the theorem5 which is closer to the theorem4 can be proved easily. In addition based on the lemma8, blinding factor of the compromised agent A belongs to the knowledge set of the *Spy*, namely the set H. Consequently, theorem4 is proved using the variant of theorem5.

XI. DERIVING SIGNED MESSAGE FROM SIGNED BLINDED MESSAGE

We have not addressed yet the key property of the blinding signature: using the blinding factor, it is possible to derive the signed message from the signed blinded message.

If the agent who derives the signed message is the owner of the blinding factor, the property can be modeled implicitly in formalizing the protocol steps. In this case, when the agent receives the signed blinded message, he can send the signed message in the following steps of the protocol.

In the case of formalizing this property in our threat model, the *Spy* should be able to derive the signed messages from the signed blinded message. More precisely, if the blinding factor r is available for the *Spy*, he can unblind the message X blinded by r and signed by the key K and obtain the message X signed by K. formally speaking, using r it is possible to derive *Crypt* K X from *Crypt* K (*Crypt* r X).

In our present model, the Spy can only unblind the blinded message and it is not possible to formalize this capability of him. So it is required to add some assumptions to our model, these assumptions are specified in the following axiom:

$$\begin{aligned}&\text{Axiom2 : [| Crypt (priK A) (Crypt (blindFac V) X)} \in \text{analz H;}\\ &\qquad \text{Key(blindFac V)} \in \text{analz H |]}\\ &\quad \Longrightarrow \text{Crypt (priK A) X} \in \text{analz H}\end{aligned} \tag{22}$$

Similar to theorem4, we expect that the *Spy* could obtain the signed message from a signed blinded message, if the message has been blinded by a compromised agent. The following states this issue:

$$\begin{aligned}&\text{Theorem 7: [| B} \in \text{bad;}\\ &\text{Crypt (priK A) (Crypt (blindFac B) X)} \in \text{analz (knows Spy evs)|]}\\ &\quad \Longrightarrow \text{Crypt (priK A) X} \in \text{analz (knows Spy evs)}\end{aligned} \tag{23}$$

As the *Spy* knows the blinding factor of compromised agents, it can be proved that he can obtain components of the observed messages blinded by the compromised agents. The theorem can be proved, using lemma8 which proved that blinding factor of compromised agent is available to the *Spy*.

XII. CONCLUSION

In this paper, we have extended the inductive method by introducing the formalization of Blind Signature, an advanced cryptographic primitive used mostly in e-voting protocols. Blind signature is modeled in the Inductive Method, as an exclusive unique symmetric key that provides the capability of obtaining a signed message from a signed blinded message. Then we showed backward compatibility of our model by already defined keys and threat model and proved that the formalization meets the expected requirements. The model in this study enables us to specify and verify protocols which use blind signature.

REFERENCES

[1] L. C. Paulson, "Proving Properties of Security Protocols by Induction, " Proc. of 10th Computer Security Foundations Workshop, USA, 1997.
[2] L. C. Paulson, "Isabelle A Generic Theorem Prover," Vol. 828, Lecture Notes in Computer Science. Springer, 1994.
[3] L. C. Paulson, "Mechanized Proofs for a Recursive Authentication Protocol," Proc. of 10th IEEE Computer Security Foundations Workshop, 1997, pp. 84-95.
[4] L. C. Paulson, "Relations between Secrets: two Formal Analyses of the Yahalom Protocol" Journal of Computer Security, 2000.

[5] G. Bella and L. C. Paulson, "Mechanising BAN Kerberos by the Inductive Method," Proc. of the International Conference on Computer-Aided Verification, Vol. 1427 of Lecture Notes in Computer Science, Springer , 1998, pp. 416-427.
[6] G. Bella and L. C. Paulson, "Kerberos Version IV: Inductive Analysis of the Secrecy Goals, " Proc. of the 5th European Symposium on Research in Computer Security, Vol. 1485, Springer, 1998. pp. 361-375.
[7] G. Bella and L. C. Paulson, "Using Isabelle to prove Properties of the Kerberos Authentication System, " Proc. of Workshop on Design and Formal Verification of Security Protocols, 1997.
[8] G. Bella, "Message reception in the inductive approach," Technical Report 460, Computer Laboratory, University of Cambridge, 1999.
[9] G. Bella, "Modelling agents' knowledge inductively," Proc. of the 7th Security Protocols Workshop,Springer, 2000, pp. 85–94.
[10] G. Bella, "Mechanising a protocol for smartcards," Proc. of the 1st International Conference on Research in Smartcards, Springer, 2001, pp. 19–33.
[11] G. Bella, L.C. Paulson, "Mechanical Proofs about a Non-repudiation Protocol," Proc. of Theorem Proving in Higher Order Logics, Vol. 2152, Springer, 2001, pp. 91-104.
[12] G. Bella, F. Massacci, L. C. Paulson, "Verifying the SET registration protocols," Journal of Selected Areas in Communications, Vol. 21, No. 1, 2003, pp. 77–87.
[13] G. Bella, F. Massacci, L. C. Paulson, "An Overview of the Verification of SET," Journal of Information Security, Vol. 4, no. 1, pp. 17–28, 2005.
[14] G. Bella, F. Massacci, L. C. Paulson, "Verifying the SET Purchase Protocols," Journal of Automated Reasoning, Vol. 36, No.1, 2006, pp. 5–37.
[15] A. Fujioka, T. Okamoto, and K. Ohta, "A Practical Secret Voting Scheme for Large Scale Elections," Proc. Advances in Cryptology, Springer, Berlin, 1992, pp. 244-251.