Establishing and Fixing Security Protocols Weaknesses Using a Logic-based Verification Tool

Anca D. Jurcut, Tom Coffey, and Reiner Dojen
University of Limerick, Limerick, Ireland
Email: anca.jurcut@ul.ie; tom.coffey@ ul.ie; reiner.dojen@ ul.ie

***Abstract*—**Formal verification aims at providing a rigid and thorough means of evaluating the correctness of security protocols and also establishing that the protocols are free of weaknesses that can be exploited by attacks. This paper discusses the process of formal verification using a logic–based verification tool. The verification tool with attack detection capabilities is introduced, and the verification process is demonstrated by way of a case study on two published security protocols that provide mutual authentication using smart cards. The performed verification reveals new weaknesses in the protocols that can be exploited by a replay attack and a parallel session attack. The impact of these attacks is that an attacker is able to masquerade as a legitimate remote user to cheat the system. The reasoning why these attacks are possible is detailed and an amended protocol, resistant to these attacks is proposed. Formal verification of the amended protocol provides confidence in the correctness and effectiveness of the proposed modifications.

***Index Terms*—**freshness, replay attacks, parallel session attacks, weaknesses, attack detection, formal verification, logic-based verification tool, smart card, mutual authentication

I. Introduction

Security is a major issue in electronic communications. Cryptographic protocols are used to provide security services, such as confidentiality, authentication and non-repudiation. The design of provably correct security protocols is complex and highly prone to error. The main difficulty in the development of security protocols is to address the vast possibilities of an adversary to gain information [1].

Frequently, informal and intuitive techniques are used to verify such protocols. However, the absence of formal verification of these protocols can lead to weaknesses and security errors remaining undetected. Many published security protocols have subsequently been found to contain security weaknesses [2]-[6], which can be exploited by various attacks on the protocol such as replay and parallel session.

A replay attack occurs when a message recorded in a previous run of the protocol is replayed by the intruder as a message in the current run of the protocol. In a parallel session attack the attacker starts new runs of the protocol using knowledge gathered from previous runs. The attacker mixes and matches pieces of session running in parallel, to achieve advantages, which were not intended by the security protocol designer.

A remote user authentication scheme is a two-party protocol whereby an authentication server in a distributed system confirms the identity of a remote individual logging on to the server over an un-trusted, open network. Several schemes using smart cards have been proposed recently for remote user authentication, including: Yang-Shieh's scheme [7], Hwang-Li's scheme [8], Lu-Cao's scheme [9], Hwang-Lee-Tang's scheme [10] and Chien-Jan-Tseng's scheme [11]. The scheme proposed by Lee, Kim and Yoo [12] is a nonce-based mutual authentication scheme, in order to prevent the possibility of mounting parallel session and reflection attacks.

This paper is concerned with formal verification and its use in the design of security protocols. Section II of the paper introduces formal verification of security protocols. In section III a new CDVT/AD logic-based verification tool is introduced. Section IV describes the mutual authentication scheme of Lee, Kim and Yoo [12] and its amended version [2]. These schemes are formally verified in section V. New weaknesses in these schemes, exploitable by a replay attack and a parallel session attack, are detailed. Additionally the reasons why these attacks are possible are explained in section VI. An amended version of the Lee, Kim and Yoo scheme is proposed in section VII and its formal verification is presented in section VIII. Finally, section IX concludes this paper.

II. Formal Verification

Formal verification of security protocols is concerned with proving that the goals of the protocols are established and demonstrating the presence of any weaknesses that may be exploitable by mountable attacks. Formal verification is an essential part of the design process [13], as it:

- Provides a systematic way to detect design flaws;
- Identifies the exact cryptographic properties a protocol aims to satisfy;
- Identifies the assumptions and the environment under which these properties hold;
- Removes ambiguity in the specifications of the protocol.

Formal verification techniques can be categorized in two main classes: deductive reasoning [1], [14]-[18] and

Manuscript received August 13, 2013; revised October 25, 2013.
This work was supported by Science Foundation Ireland under Grant No. 11/RFP.1/CMS 3340.
Corresponding author email: anca.jurcut@ul.ie; tom.coffey@ul.ie; reiner.dojen@ ul.ie.
doi:10.12720/jcm.8.11.795-805

state exploration methods [19]-[22]. Existing deductive reasoning methods include theorem proving and logic-based techniques. Deductive techniques are based on theories that represent the protocol faithfully comprising sets of axioms and deduction rules. Analysis of the protocol in that theory entails proving one or more theorems. State exploration methods take a quite different approach, which is more akin to simulation and testing. In state exploration, a protocol is characterized as the set of all its possible traces. Given the security protocol specification as input, the verification method explores as many execution paths of the protocol as possible, checking at each reachable state if some conditions hold.

A. Logic-based Verification

Formal verification using logic-based techniques has been shown to be effective and a number of weaknesses in protocols previously considered secure have been discovered [13], [23], [24]. Logics have an advantage in that they are usually decidable, efficiently computable, and thus can be completely automated, as has been shown by Dojen and Coffey's automated GNY logic [1] and the CDVT logic based verification tool [25].

Logic theories can be used to reason about the safety, security and authenticity properties of security protocols. The objective of the conventional logical analysis is to verify whether the desired goals of the protocols can be derived from the initial assumptions and protocol steps using a deductive reasoning process. Although different conventional logics may use distinct notations and involve different axioms, postulates or rules, these logics follow the same steps for the application of deductive reasoning process, as described in [13]:

- Formalization of the protocol messages;
- Specification of the initial assumptions;
- Specification of the protocol goals;
- Application of the logical postulates.

Manual application of formal logics theories to prove the correctness of security protocols can be difficult and error prone [13], mainly because logic-based techniques require a high level of skill to use, relying on the ability and experience of a user to generate the formal proof of the protocol. Automation of the verification process minimizes the risk of faulty proofs and simplifies the verification process for the protocol verifier. Details of logic specific implementation issues for automation can be found in [1].

III. CDVT/AD - A Logic-based Verification Tool with Attack Detection

The CDVT/AD verification tool (cf. Fig. 1) is a new automated system that implements a modal logic of knowledge and an attack detection theory. The tool uses a proving engine based on Layered Proving Trees concept [1]. The implemented tool can analyze the evolution of both knowledge and belief during a protocol execution and therefore is useful in addressing issues of both security and trust. Additionally, the verification tool has the capability of detecting protocol design weaknesses that can be exploited by replay or parallel session attacks. This attack detection facility incorporates detection rules that are classified into five main categories addressing problems related to: (1) message freshness, (2) message symmetries, (3) handshake construction, (4) signed statements and (5) certificates. The resulting automated system, as shown in Fig. 1, enables both attack detection analysis and conventional logic-based protocol verification from a single protocol specification.

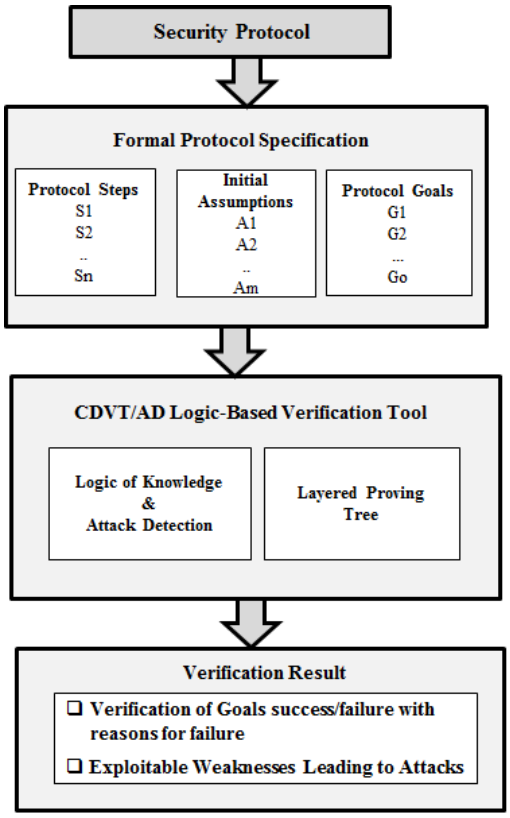


Figure 1. The CDVT/AD verification tool.

The verification tool applies the axioms and rules of the implemented logic theory in an attempt to derive the protocol goals as a logical consequence of the initial assumptions and the protocol steps. If such a derivation exists, the verification is successful and the verified protocol can be considered secure within the scope of the logic. Additionally, the verification tool triggers an attack detection rule violation if the prerequisites of the rule can be derived from the formal specification. If the verification fails (i.e. goals are not proven or an attack detection rule is triggered), investigation of the verification process can point to missing assumptions or weaknesses in the protocol. In this case the protocol should be re-designed and re-verified.

A. The Language of the CDVT/AD tool

The CDVT/AD verification tool uses a parser to read in the protocol specification from a text file. Table I summarizes the atomic units of the textual grammar.

TABLE I. ATOMIC UNITS OF TEXTUAL GRAMMAR

Textual Grammar	Regular Expression
Principal	[AB-EIJLMOQRSU-Z][A-Za-z_0-9_]*
Trusted Principal	TTP[A-Za-z0-9_]*
Sym. Key	K[a-z][a-zA-Z0-9_]*
Public Key	K[a-z][A-Za-z0-9_]*Pub
Private Key	K[a-z][A-Za-z0-9_]*Priv
Nonce	N[a-z][A-Za-z0-9_]*
Timestamp	TS[a-z][A-Za-z0-9_]*
Function	F[A-Za-z0-9_]*
Hash	H[A-Za-z0-9_]*
Binary Data	[a-z][A-Za-z0-9_]*

Composite data components are constructed according to Table II, where elements follow the regular expressions as given in Table I and "Data" represents an arbitrary data element (either atomic unit or composite data).

TABLE II. COMPOSITE DATA CONSTRUCTION

Composite Data	Textual Representation
Concatenation	Data,Data
Group Element	(Data)
Symmetric Encryption	{Data}Data
Public Key Encryption	{Data}KPub
Private Key Encryption	{Data}KPriv
Function of Data	F(Data)
Hash of Data	H(Data)
Key Material of Data	KMaterial(Data)

Statements are defined according to the rules presented in Table III, where elements follow the regular expressions as given in Table I, "Data" is either an atomic data unit or a composite data as defined in Table II, "i" indicates the indexed discrete time and "Statement" represents an arbitrary statement. "Operator" can be any of: "send," "receive" or "possess," while "Trans_Operator" are the transmission operators and can be any of the following: "send to" or "receive from". The purpose of these transmission operators is to be used for the construction of a specific type of statement expressing reception from a principal or emission to a principal.

TABLE III. STATEMENT CONSTRUCTION

Principal Operator at[i] Data
Principal Trans_Operator Principal at[i] Data
Principal know at[i] Statement
Principal believe at[i] Statement
Principal know at[i] NOT (Statement)
Principal believe at[i] NOT (Statement)
(Statement)
NOT(Statement)
(Statement AND Statement)
(Statement IMPLY Statement)

Each line of the textual specification file is preceded by a label. Assumptions are labeled "An," protocol steps are labeled "Sn" and protocol goals are labeled "Gn," where n numbers each group sequentially. Every line must be closed with a semicolon (';') and comments are introduced by a double forward slash ('//' – C++ style comments).

The inference rules provided are the standard rules of natural deduction. The axioms of the logic of knowledge express the fundamental properties of public-key cryptographic protocols such as the ability of a principal to encrypt/decrypt based on knowledge of a cryptographic key, while the axioms in the case of the attack detection logic theory enable reasoning about message characteristics in cryptographic protocols. The axioms also reflect the underlying assumptions of the logics, which are as follows:

- The communication environment is reliable, but hostile. That is, message loss or modification can only occur as consequence of hostile intervention.
- The cryptosystem is ideal. That is, the encryption and decryption functions are completely non-invertible without knowledge of the appropriate cryptographic key and are invertible with knowledge of the appropriate cryptographic key. The cryptosystem is collision-free so that it is not possible to create the same ciphertext from two different pieces of plaintext.
- A public key used by the system is considered valid if it has not exceeded its validity period and only its rightful owner knows the corresponding secret key.
- If a piece of data is encrypted/decrypted, then the entity which performed the encryption/decryption must know that data (the data can be plaintext or ciphertext).

IV. LKY NONCE-BASED MUTUAL AUTHENTICATION SCHEME AND ITS DERIVATIVE

In 2005, Lee, Kim, and Yoo (LKY) [12] proposed a nonce-based mutual authentication scheme using smart cards. The scheme employs the same authentication structure in both the remote user and the system. The scheme consists of three phases: the registration phase, the login phase, and the verification phase. The registration phase is performed only once when a new user registers itself with the server. The login and verification phases are carried out whenever a user wants to gain access to the server.

In 2007 Nam, Kim, Park and Won (NKPW) [2] claimed that the LKY scheme is vulnerable to a parallel session attack, in which an intruder who is not registered with the server is able to gain access to the server. The authors proposed a fix in order to prevent the claimed parallel session attack.

A. Lee, Kim, and Yoo Scheme

A description for each of the registration, login and verification phases of the LKY is as follows:

1) Registration phase

Let x be the only secret key maintained by the system (denoted AS) and h() be a one-way cryptographic hash function. Assume a remote user U_i registers his identifier ID_i and password PW_i to the system in a secure channel. The system computes $R_i = h(ID_i \oplus x) \oplus PWi$, where $\oplus$

denotes the bit-wise exclusive-OR operator, stores h() and R_i into the memory of a smart card, and issues the smart card to U_i.

2) Login phase

When U_i wants to log into the system, he inserts his smart card into the terminal and enters his identifier ID_i and password PW_i. The smart card then performs the following operations:

- Compute $C_1 = R_i \oplus PW_i$ *and* $C_2 = C_1 \oplus N_1$, *where* N_1 *is a random nonce.*
- Send the message $M_1 = (ID_i, C_2)$ *to the system.*

3) Verification phase

After the authentication request message M_1 is received, the system and the smart card execute the following operations to achieve mutual authentication.

a) The system checks the validity of ID_i and then computes:
   $C_1 = h(ID_i \oplus x)$,
   $N_1 = C_2 \oplus C_1$,
   $V_1 = h(C_2, N_1)$, and
   $C_3 = C_1 \oplus N_2$, where N_2 is a random nonce.
b) The system sends the message $M_2 = (V_1, C_3)$ to U_i.
c) Upon receiving the message M_2, U_i verifies whether $V_1 = h(C_2, N_1)$. If equal, U_i believes that the system is authenticated. Then the smart card computes:
   $N_2 = C_3 \oplus C_1$ and
   $V_2 = h(C_3, N_2)$.
d) The smart card sends the message $M_3 = (V_2)$ to the system.
e) The system verifies whether $V_2 = h(C_3, N_2)$. If equal, the system believes that U_i is authenticated.

Fig. 2 outlines the authentication session of the scheme.

1. U_i ->AS: $M_1 = (ID_i, C_2)$
2. AS->U_i: $M_2=(V_1, C_3)=(h(C_2, N_1), h(ID_i \oplus x) \oplus N_2)$
3. U_i ->AS: $M_3 = (V_2) = h(C_3, N_2)$

Figure 2. Authentication session of LKY scheme.

B. NKPW Modified LKY Scheme

Nam, Kim, Park and Won (NKPW) [2] identified a weakness in the authentication session of the LKY scheme relating to symmetric structure of the messages exchanged between the user and the server. In order to address this problem, the authors proposed a fix, where the sender's identity is to be included as part of the hash input in computing V_1 and V_2. Thus, V_1 was changed from $V_1=h(C_2, N_1)$ to $V_1=h(AS, C_2, N_1)$ and V_2 was changed from $V_2=h(C_3, N_2)$ to $V_2=h(U_i, C_3, N_2)$. The steps of the authentication session for NKPW modified LKY scheme are shown in Fig. 3.

1. U_i ->AS: $M_1 = (ID_i, C_2)$
2. AS->U_i: $M_2 = (V_1, C_3) = (h(AS, C_2, N_1), h(ID_i \oplus x) \oplus N_2)$
3. U_i ->AS: $M_3 = (V_2) = h(U_i, C_3, N_2)$

Figure 3. Authentication session of NKPW modified LKY scheme.

V. Verification of the LKY and NKPW Mutual Authentication Schemes

In this section the CDVT/AD verification tool is used to establish the correctness of the authentication session of the LKY and the NKPW modified LKY authentication schemes. In addition, any vulnerability in the design of the verified protocol that may be exploited by replay or parallel session attacks will be highlighted by the verification tool.

Prior to verification, the protocol must be formalized, i.e. translated into the language of the tool. A formalized protocol consists of three components:

- Initial assumptions (conditions that hold before the protocol starts);
- Protocol steps (the messages exchanged between the principals);
- Protocol goals (conditions that are expected to hold if the protocol terminates successfully).

A. Formalisation of LKY Scheme

The following notations are used when translating the LKY authentication scheme(s) into the language of the CDVT/AD tool:

user U_i: principal A
system AS: TTP
identifier ID_i: identifier of A
nonce N_1: nonce generated by principal A, Na
nonce N_2: nonce generated by server TTP, Nttp
data ⊕ data: {data}data
expression C_1: H({A}datax)
expression C_2: {Na}H({A}datax)
expression C_3: {Nttp}H({A}datax)
expression V_1: H({Na}H({A}datax), Na)
expression V_2: H({Nttp}H({A}datax), Nttp)

The description of the authentication session of LKY scheme (Fig. 2), using the above presented notations is as follows:

1. A ->TTP: A, {Na}H ({A}datax)
2. TTP->A: H({Na}H({A}datax),Na), {Nttp}H({A}datax)
3. A ->TTP: H({Nttp}H({A}datax), Nttp)

1) Initial assumptions

Initial assumptions are statements defining what each principal possesses and knows at the beginning of a protocol run. The following specifies the initial assumptions of the LKY scheme:

A1: A possess at[0] H({A}datax);
A2: A know at[0] TTP possess at[0] H({A}datax);
A3: A possess at[0] Na;
A4: A know at[0] NOT(Zero possess at[0] Na);
A5: TTP possess at[0] H({A}datax);
A6: TTP know at[0] A possess at[0] H({A}datax);
A7: TTP possess at[0] Nttp;
A8: TTP know at[0] NOT(Zero possess at[0] Nttp);

Statements A1-A4 define the initial assumptions for principal A before a protocol run with TTP (i.e. at time t0). Assumption A1 states that A possesses symmetric key "H({A}datax)". A2 specifies that A is aware of the fact that TTP possesses "H({A}datax)". A3 specifies that

A possesses the nonce Na and assumption A4 states that A knows that nonce Na is fresh for the current run of the protocol. Statements A5-A8 define the initial assumptions of TTP's possessions and knowledge before the start of the protocol run. A5 states that TTP possesses key "H({A}datax)". A6 specifies that TTP is aware of the fact that principal A possesses "H({A}datax)". A7 expresses the fact that TTP possesses the nonce Nttp and A8 states that TTP knows that Nttp is fresh for the current run of the protocol.

2) LKY scheme steps

The LKY scheme steps are formalised as follows:

S1: TTP receivefrom A at [1] A,{Na}H({A}datax);

S2: A receivefrom TTP at [2] H({Na}H({A}datax),Na), {Nttp}H({A}datax);

S3: TTP receivefrom A at [3] H({Nttp}H({A}datax), Nttp);

3) LKY scheme goals

The objective of the LKY scheme is the mutual authentication of the user and the system (i.e. the authentication of A to TTP and of TTP to A). The formalized goals of the LKY scheme are as follows:

G1: A know at [2] TTP send at [2] H({Na}H({A}datax),Na);

G2: A know at [2] NOT(Zero send at [0] H({Na}H({A}datax), Na));

G3: A know at [2] TTP send at[2] {Nttp}H({A}datax);

G4: A know at [2] NOT (Zero send at [0] {Nttp}H({A}datax));

G5: TTP know at [3] A send at [3] H({Nttp}H({A}datax),Nttp);

G6: TTP know at[3] NOT(Zero send at [0] H({Nttp}H({A}datax),Nttp));

Goals G1-G4 relate to authentication of TTP to A. G1 states that A knows at step 2 that TTP is the source of message component H({Na}H({A}datax),Na), which is the reply to A's nonce challenge. G2 states that A knows that this message component has been created during the current protocol run. G3 states that A knows at step 2 that TTP is the sender of the message component {Nttp}H({A}datax) and G4 states A knows that this message component has been created during the current protocol run. Goals G5-G6 are the corresponding goals regarding authentication of A to TTP. G5 states that TTP knows at step 3 that A is the source of message component H({Nttp}H({A}datax), Nttp), i.e. of the reply to TTP's nonce challenge. G6 states that TTP knows that this message component has been created during the current protocol run.

4) LKY scheme verification results

The results of the automated verification of the LKY scheme are presented in Fig. 4 - Fig. 6. The verification results of the LKY scheme shown in Fig. 5, indicate that a number of the security goals (represented by "(4)," "(5)" and "(6)") are not satisfied. Browsing the verification tool allows further investigation of the reasons for the failed goals. For example, Fig. 5 details the failed verification of goal G3, where A's inability to establish the freshness of nonce Nttp in step 2 is the reason for the failure.

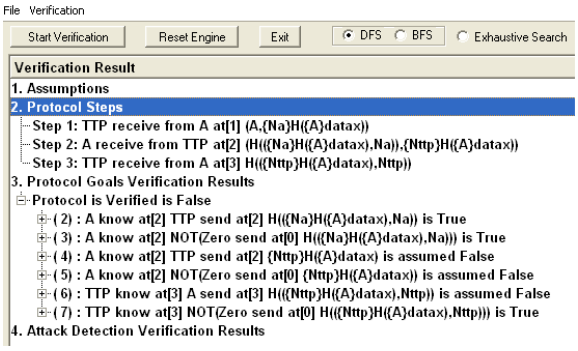


Figure 4. Verification results of LKY scheme

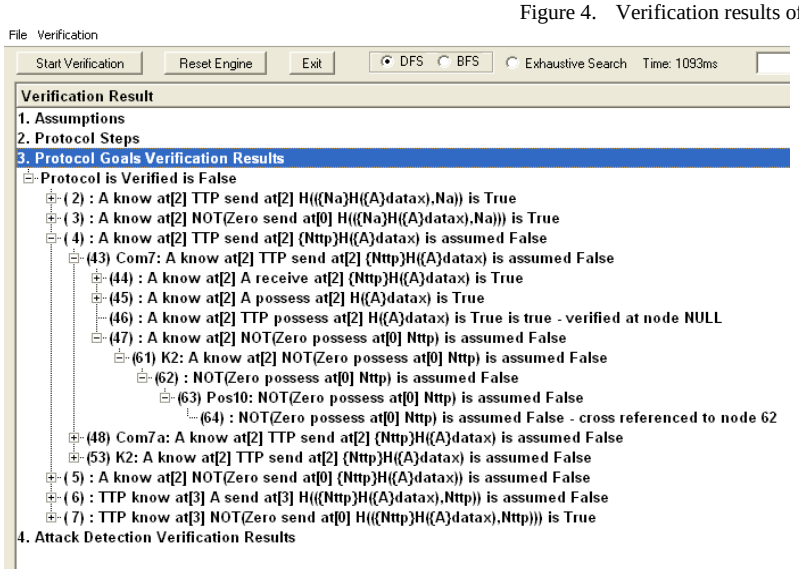


Figure 5. Investigating the reason why a security goal failed verification

Investigating the verification results for failed protocol goals in this fashion reveals that the protocol suffers from the following weaknesses:

- A's inability to establish that TTP is the source of message component {Nttp}H({A}datax) in step 2 (goal G3) prevents the authentication of TTP to A.

- A's inability to establish that message component {Nttp}H({A}datax) of step 2 (goal G4) is fresh.
- TTP's inability to establish that A is the source of message component H({Nttp}H({A}datax),Nttp) in step 3 (goal G5) prevents the authentication of A to TTP.

Thus, neither authentication of A to TTP nor authentication of TTP to A is achieved by the LKY scheme.

Additionally, three weaknesses in the design of LKY scheme, identifying freshness and parallel session vulnerabilities are revealed. It can be seen from Fig. 6 that three attack detection rules of the verification tool (one freshness rule and two symmetry rules) are triggered. The result obtained with respect to the freshness rules is that the cryptographic expression in step 2 {Nttp}H({A}datax) is not freshness protected. This implies that {Nttp}H({A}datax) does not contain data which the receiver A recognizes as being fresh (i.e. a nonce previously generated by A in the same protocol run). The results derived for the symmetry rules reveal the following two weaknesses in the LKY scheme:

- The cryptographic expressions {Na}H ({A}datax) of step 1 and {Nttp}H({A}datax) of step 2 are symmetric;
- The hashed expressions H({Na}H({A}datax), Na) of step 2 and H({Nttp}H({A}datax,A), Nttp) of step 3 are symmetric.

B. Formalisation of NKPW Modified LKY Scheme

The description of the authentication session of scheme, using the above presented re-denotations is as follows:

1. A ->TTP: A, {Na}H ({A}datax)
2. TTP->A: H({Na}H({A}datax),Na,TTP), {Nttp}H({A}datax)
3. A ->TTP: H({Nttp}H({A}datax,A), Nttp,A)

1) Initial assumptions

The formalized initial assumptions are the same as that of the LKY scheme.

2) Steps

The steps of the scheme are formalised as follows:

S1: TTP receivefrom A at [1] A,{Na}H({A}datax);

S2: A receivefrom TTP at [2] H({Na}H({A}datax),Na,TTP), {Nttp}H({A}datax);

S3: TTP receivefrom A at [3] H({Nttp}H({A}datax,A),Nttp,A);

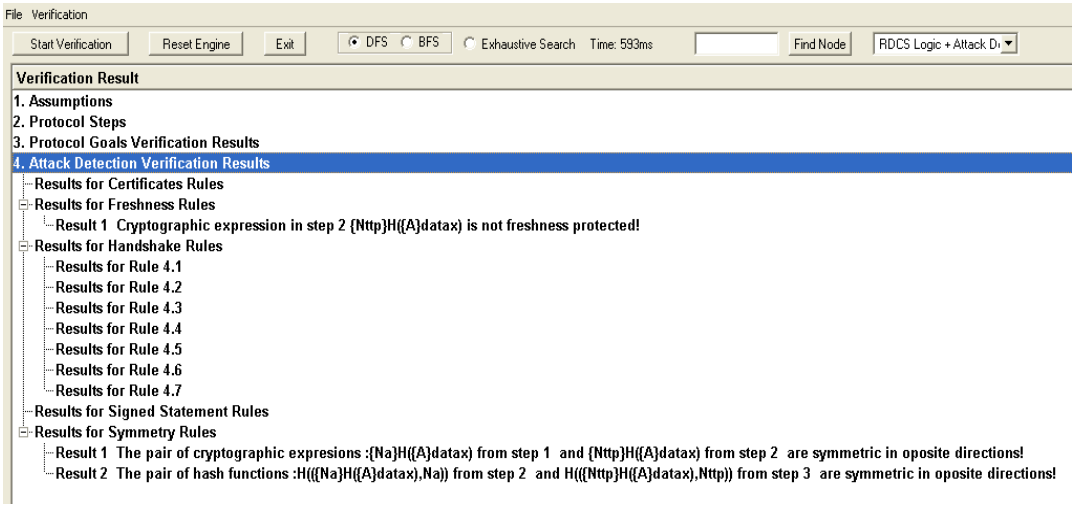


Figure 6. LKY scheme attack detection verification results.

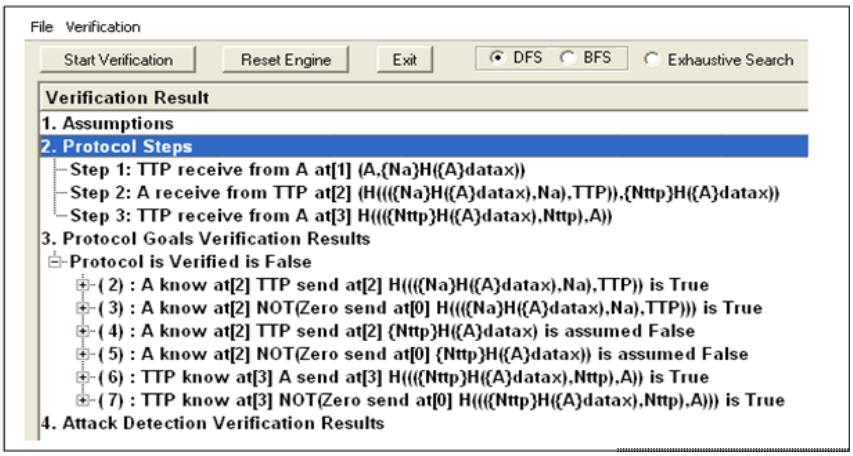


Figure 7. Verification results of NKPW scheme

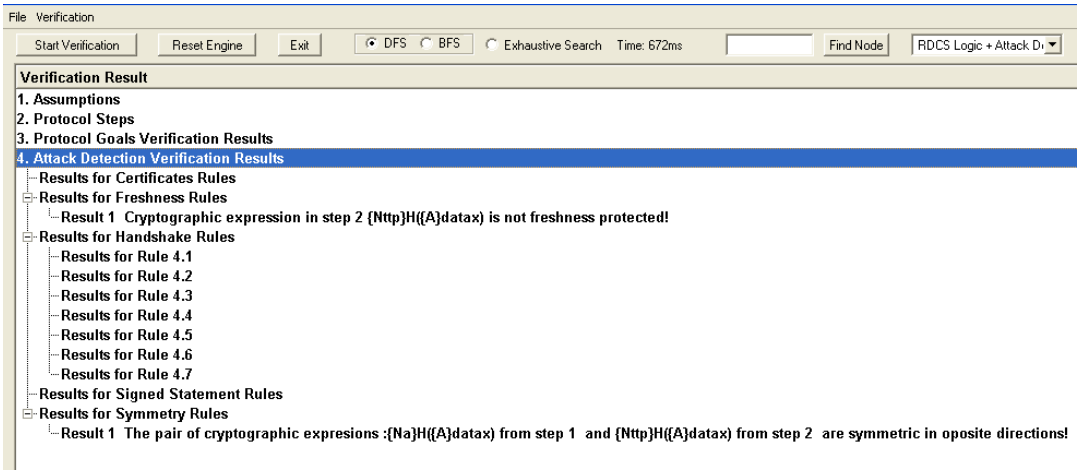


Figure 8. NKPW scheme attack detection verification results.

3) Goals

The formalized goals are similar to the LKY scheme.

G1: A know at [2] TTP send at [2] H({Na}H({A}datax), Na, TTP);

G2: A know at [2] NOT(Zero send at [0] H({Na}H({A}datax), Na, TTP));

G3: A know at [2] TTP send at [2] {Nttp}H({A}datax);

G4: A know at [2] NOT (Zero send at [0] {Nttp}H({A}datax));

G5: TTP know at [3] A send at [3] H({Nttp}H({A}datax), Nttp, A);

G6: TTP know at [3] NOT(Zero send at [0] H({Nttp}H({A}datax), Nttp, A));

4) NKPW modified LKY scheme verification results

The verification results are presented in Fig. 7 and Fig. 8. The results show that the NKPW modified LKY scheme continues to suffer from two of the three previously found weaknesses on the LKY scheme:

- The cryptographic expression in step 2 {Nttp}H({A}datax) is not freshness protected (i.e. does not contain anything which the receiver A recognizes it as being fresh)
- The cryptographic expressions {Na}H ({A}datax) of step 1 and {Nttp}H({A}datax) of step 2 are symmetric

Hence, the modified scheme is still susceptible to a possible replay attack which will be discussed in the following section.

VI. Discussion on Design Weaknesses Leading to Replay and Parallel Session Attacks on LKY and NKPW Modified LKY Schemes

The two types of weaknesses revealed above in the design of LKY and NKPW modified LKY schemes can be exploited by an attacker in a new replay attack and a parallel session attack.

Both LKY and NKPW modified LKY schemes are vulnerable to a replay attack due to the fact that the cryptographic expression in step 2 {Nttp}H({A}datax) does not contain any component which the receiver A recognizes as being fresh. The impact of this attack is that an attacker, without knowing any secret of a remote user, can masquerade as a legitimate remote user and can obtain the valid authentication message from any normal session between the remote user and the system TTP.

The LKY scheme is vulnerable to a parallel session attack due to the symmetrical structure of (1) the pair of cryptographic expressions {Na}H({A}datax) and {Nttp}H({A}datax) and (2) the pair of hashed expressions H({Na}H({A}datax),Na) and H({Nttp}H({A}datax), Nttp). The impact of this attack is that an intruder is able to masquerade as a legitimate remote user and fool the server into accepting a login request even from a user who is not registered with the system.

A. A New Replay Attack on the LKY Scheme

Our verification of the LKY scheme, outlined above, reveals that a remote user A cannot tell whether data received from the server TTP is fresh. As a consequence, a replay attack can be carried out against the LKY protocol, where an attacker can impersonate a target remote user A. Assuming I(A) denotes the attacker I impersonating A, the attack can be carried out as follows:

i.1. A -> TTP: A, {Na}H ({A}datax)
ii.1 I(A) ->TTP: A, {Ni}H ({A}datax)
i.2. TTP -> I(A): H({Na}H({A}datax),Na), {Nttp}H({A}datax)
ii.2. TTP -> I(A): H({Ni}H({A}datax),Ni), {Nttp_1}H({A}datax)
i.2'. I(TTP) -> A: H({Na}H({A}datax),Na), {Nttp_1}H({A}datax)
i.3. A ->I(TTP): H({Nttp_1}H({A}datax,A), Nttp_1)
ii.3. I(A)->TTP: H({Nttp_1}H({A}datax,A),Nttp_1)

Figure 9. New attack on LKY scheme.

The replay attack (detailed in Fig. 9) assumes that a remote user A initiates the protocol with the system, by sending message i.1 consisting of its identity concatenated with the component {Na}H({A}datax) to the server TTP. An attacker I intercepts the message i.1 intended for TTP and initiates a new session (denoted ii) with the server TTP, by sending message ii.1, where Ni is

a random number generated by attacker I. After receiving the message i.1 and ii.1, the server TTP generates and sends messages i.2 and ii.2 according to the specification of the exchange in the two running sessions i and ii. After the attacker intercepts and blocks the messages i.2 and ii.2, it sends the fabricated message i.2' to A, according to the specification exchange of step 2. The attacker computes i.2', by replaying first component of the message i.2 together with the second component of message ii.2. Upon receiving the message i.2', principal A computes and sends the message i.3, according to the specification of the exchange. Note that A computes the message i.3 (i.e. the response of the nonce challenge emitted as the second component of message ii.2'), since it successfully verifies the message H({Na}H({A}datax), Na), which is the corresponding response to its challenge sent as part of the message i.1. The attacker can replay i.3 as message ii.3 to finish its session and pass the system's authentication.

As a result, although A authenticates the system in the first session i, after the attacker's session ii, the system TTP mistakenly believes that the attacker is the honest user. Hence, LKY scheme fails to provide the mutual authentication service, since the attacker without knowing any secret information can impersonate a remote user A to cheat the system.

This replay attack can also be mounted on the NKPW modified LKY scheme.

1) Reasoning on replay attack

This replay attack can be mounted due to the weakness in the message components transmitted in step 2 of the authentication session of the scheme. The cryptographic expression {Nttp}H({A}datax) does not contain any component which the receiver A can recognize as being fresh, since Nttp is a new nonce generated by the sender (server TTP) in step 2. Therefore principal A has no assurance that the nonce Nttp is fresh and is not replayed from a previous run of the protocol. Hence, an intruder can substitute a previously recorded message 2 from TTP to A ({Nttp_1}H({A}datax)) and mislead principal A to accept an old and possibly compromised nonce (Nttp_1).

B. A Parallel Session Attack on the LKY Scheme

Our verification of the LKY scheme, outlined above, reveals that two pairs of cryptographic messages are symmetric. As a consequence a parallel session attack that can be mounted against the LKY protocol, where an attacker can forge login messages to impersonate a legitimate user.

Assume that an intruder I wants to gain access to the server masquerading as a legitimate user A. The corresponding attack scenario is outlined in Fig. 10.

i.1. I(A)-> TTP: A, {Ni}H ({A}datax)
i.2. TTP -> I(A): H({Ni}H({A}datax),Ni), {Nttp}H({A}datax)
ii.1 I(A) -> TTP: A, {Nttp}H({A}datax)
ii.2. TTP -> I(A): H({Nttp}H({A}datax),Nttp), {Nttp_1}H({A}datax)
i.3. I(A) ->TTP: H({Nttp}H({A}datax),Nttp)
ii.3. dropped

Figure 10. A parallel session attack on LKY scheme.

The intruder, impersonating user A, launches the attack by choosing a random number Ni and sending message i.1 as a login request message to the server TTP. Note that from the server's point of view, Ni is indistinguishable from nonce Na of an honest execution, since both are random numbers. The server TTP sends the message i.2 to the intruder masquerading as A, according to the specification of the exchange. After receiving message i.2, the attacker starts a parallel session ii, posing again as user A and replaying the challenge {Nttp}H({A}datax) sent out by the server in the original session i.2. The server TTP cannot distinguish the replayed response {Nttp}H({A}datax) sent by the intruder from a genuine message ii.1 sent by a honest user A. Hence, TTP computes H({Nttp}H({A}datax),Nttp) and sends the message ii.2 in response to ii.1, as specified in the protocol. The component H({Nttp}H({A}datax), Nttp) sent as part of message ii.2 in the parallel session is exactly what the intruder needs in order to gain access permission in the original session i. Now that the attacker has obtained access to the server, it drops the parallel session with TTP. Hence, the attacker without knowing any secret information can impersonate a remote user A to cheat the system.

1) Reasoning on parallel session attack

The above parallel session attack can be mounted due to the symmetrical structure of two pairs of cryptographic messages:

- The pair of cryptographic expressions exchanged in steps 1 and 2 ({Na}H({A}datax),{Nttp}H({A}datax));
- The pair of hashed expressions exchanged in steps 2 and 3 (H({Na}H({A}datax),Na), H({Nttp}H({A}datax)).

Hence, an intruder I can impersonate an honest user A in a parallel run, where the cryptographic expression {Nttp}H({A}datax) obtained in step 2 of the first protocol run (run i) is used in step 1 of the parallel run (run ii) and respectively, the hashed expression H({Nttp}H({A}datax), Nttp) obtained in step 2 of the parallel protocol run is used in step 3 of the first run. As the cryptographic expression required in step 1 of the parallel run is symmetric with {Nttp}H({A}datax) of step 2 of the first run and the hashed expression required in step 3 of the first run is symmetric with H({Nttp}H({A}datax), Nttp) of step 2 of the parallel run, the server TTP cannot distinguish the replayed messages in step 1 of the parallel run and step 3 of the first run, sent by the intruder, from the legitimate messages sent by A.

VII. AMENDING THE LKY SCHEME

As shown in the previous sections, neither the LKY scheme [12] nor the modified version [2] can be considered secure. We now present an amended version of the LKY scheme to overcome the described weaknesses that can be exploited by replay and parallel session attacks.

In order to prevent a potential replay attack the cryptographic messages transmitted in the scheme needs to be freshness protected, when necessary. The cryptographic message {Nttp}H({A}datax) in step 2 should include a component which the recipient of step 2, recognizes as fresh. This can be achieved by including nonce Na, previously generated by A in step 1 in the content of the second message of step 2, as presented in Fig. 11. Thus, the cryptographic expression that contains the nonce Nttp generated by the server can be identified by user A as fresh, i.e. as belonging to the current protocol run. Consequently, any attempt by an intruder to replay second message of step 2 will fail, as A can identify the replay through the incorrect value of Na. In order to prevent a potential parallel session attack the symmetrical structure of the hashed expressions transmitted in steps 2 and 3 and the symmetrical structure of the pair of cryptographic expressions exchanged in steps 1 and 2 of the scheme should be broken.

1. A ->TTP: A, {Na}H ({A}datax)
2. TTP->A: H({Na}H({A}datax),Na), {Nttp, Na}H({A}datax)
3. A ->TTP: H({Nttp, Na}H({A}datax,A), Nttp,A)

Figure 11. Amended version proposed for LKY scheme.

Fig. 12 outlines our proposed amended version in the original notations for the LKY nonce-based mutual authentication scheme using smart cards:

1. U_i ->AS: $M_1 = (ID_i, C_2)$
2. AS->U_i:$M_2=(V_1, C'_3)=(h(C_2, N_1), h(ID_i \oplus x) \oplus N_2 \oplus N_1)$
3. U_i ->AS: $M_3 = (V'_2) = h(C'_3, N_2, U_i)$

Figure 12. Proposed amended version for LKY scheme.

VIII. Verification of the Proposed Amended Version of the LKY Scheme

A. Initial Assumptions

The initial assumptions are identical with the ones for the LKY scheme.

B. Amended Scheme Steps

Having changed the steps of the scheme, these changes need to be reflected in the formalisation of the steps:

S1: TTP receivefrom A at [1] A,{Na}H({A}datax);
S2: A receivefrom TTP at [2]
H({Na}H({A}datax),Na), {Nttp, Na}H({A}datax);
S3: TTP receivefrom A at [3]
H({Nttp, Na}H({A}datax,A),Nttp,A);

C. Amended Scheme Goals

The modifications of the steps are also reflected in the corresponding goals:

G1: A know at [2] TTP send at [2]
H({Na}H({A}datax), Na);
G2: A know at [2] NOT(Zero send at [0]
H({Na}H({A}datax), Na));
G3: A know at [2] TTP send at [2]
{Nttp, Na}H({A}datax);
G4: A know at [2] NOT (Zero send at [0]
{Nttp, Na}H({A}datax));
G5: TTP know at [3] A send at [3]
H({Nttp, Na}H({A}datax), Nttp, A);
G6: TTP know at [3] NOT(Zero send at [0]
H({Nttp, Na}H({A}datax), Nttp, A));

D. Results of the Verification

The results of the automated verification for the amended version of the scheme are shown in Fig. 13 and Fig. 14. As can be seen, the outcome for the attack detection verification is free of any message indicating a weakness in the protocol's design that can be exploited by mountable replay or parallel session attacks. Additionally, all goals are verified successfully. This provides confidence in the correctness and effectiveness of the amended scheme.

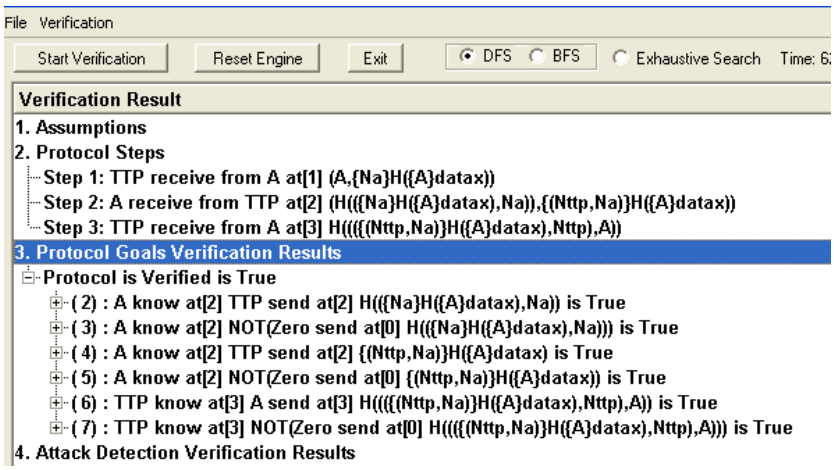


Figure 13. Proposed amended LKY scheme verification results.

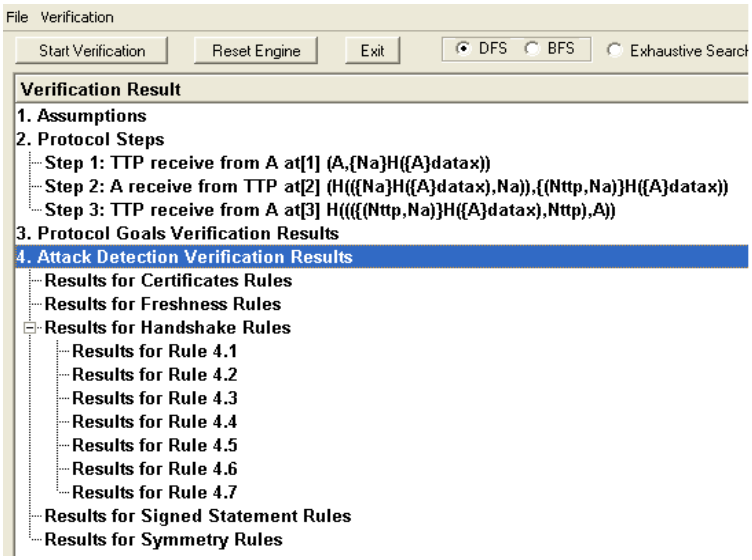


Figure 14. Proposed amended LKY scheme attack detection verification results.

IX. Conclusion

In this paper the process of formal verification of cryptographic security protocols using a modal logic was discussed. A new (automated) logic-based verification tool with attack detection capabilities for replay and parallel session attacks was introduced. The tool was used to verify the LKY nonce-based mutual authentication scheme using smart cards and NKPW amended version of the scheme.

The performed verification revealed new weaknesses in these authentication schemes that can be exploited by mountable replay and parallel session attacks. The impact of these attacks is that an attacker is able to masquerade as a legitimate remote user to cheat the system. The reasons why these attacks are possible were detailed and an amended protocol, resistant to these attacks was proposed. Formal verification of the amended protocol provides confidence in the correctness and effectiveness of the proposed modifications.

Acknowledgment

This work was funded by Science Foundation Ireland - Research Frontiers Programme (11/RFP.1/CMS 3340).

References

[1] R. Dojen and T. Coffey, "Layered proving trees: A novel approach to the automation of logic-based security protocol verification," *ACM Transactions on Information and System Security*, vol. 8, no. 3, pp. 287-311, 2005.

[2] J. Nam, S. Kim, S. Park, and D. Won, "Security analysis of a nonce-based user authentication scheme using smart cards," *IEICE Transactions Fundamentals*, vol. E90-A, no. 1, pp. 299-302, Jan 2007.

[3] X. Fu and Y. Guo, "A lightweight RFID mutual authentication protocol with ownership transfer," *Advances in Wireless Sensor Networks Communications, Computer and Information Science*, vol. 334, pp. 68-74, 2013.

[4] R. Dojen, V. Pasca, and T. Coffey, "Impersonation attacks on a mobile security protocol for end-to-end communications," *Security and Privacy in Mobile Information and Communication Systems, Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Tele-communications Engineering*, vol. 17, pp. 278-287, September 2009.

[5] R. Dojen, A. Jurcut, T. Coffey, and C. Győrödi, "On establishing and fixing a parallel session attack in a security protocol," *Intelligent Distributed Computing, Systems and Applications*, Springer Berlin / Heidelberg, vol. 162, September 2008, pp. 239-244.

[6] G. Lowe, "Some new attacks upon security protocols," in *Proc. 9th IEEE Computer Security*, Foundations Workshop, 1996, pp. 162–169.

[7] W. H. Yang and S. P. Shieh, "Password authentication schemes with smart cards," *Computer and Security*, vol. 18, no. 8, pp. 727-733, 1999.

[8] M. S. Hwang and L. H. Li, "A new remote user authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 1, pp. 28-30, 2000.

[9] R. X. Lu and Z. F. Cao, "Efficient remote user authentication scheme using smart card," *Computer Networks*, vol. 49, no. 4, pp. 535-540, 2005.

[10] M. S. Hwang, C. C. Lee, and Y. L. Tang, "A simple remote user authentication scheme," *Mathematical and Computer Modelling*, vol. 36, no. 1-2, pp. 103-107, 2002.

[11] H. Y. Chien, J. K. Jan, and Y. M. Tseng, "An efficient and practical solution to remote authentication: Smart card," *Computer and Security*, vol. 21, no. 4, pp. 372-375, 2002.

[12] S. W. Lee, H. S. Kim, and K. Y. Yoo, "Efficient nonce-based remote user authentication scheme using smart cards," *Applied Mathematics and Computation*, vol. 167, no. 1, pp. 355-361, 2005.

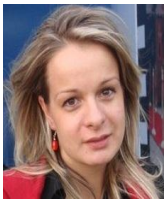
[13] T. Coffey, R. Dojen, and T. Flanagan, "Formal verification: An imperative step in the design of security protocols," *Computer Networks*, Elsevier Science, vol. 43, no. 5, pp. 601-618, 2003.

[14] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Transactions on Computer Systems*, vol. 8, no. 1, pp. 18-36, 1990.

[15] L. Paulson, "The inductive approach to verifying cryptographic protocols," *Journal of Computer Security*, vol. 6, no. 1, 1998.

[16] E. Cohen, "First-order verification of cryptographic protocols," *Journal of Computer Security*, vol. 11, no. 2, 2003.

[17] T. Coffey and P. Saidha, "A Logic for verifying public key cryptographic protocols," *IEE Journal Proceedings-Computers and Digital Techniques*, vol. 144, no. 1, pp. 28-32, 1997.
[18] A. Datta, A. Derek, J. Mitchell, and A. Roy, "Protocol composition logic (PCL)," *Electronic Notes in Theoretical Computer Science*, vol. 172, pp. 311-358, 2007.
[19] G. Lowe, "Casper: A compiler for the analysis of security protocols," *Journal of Computer Security*, vol. 6, pp. 53–84, 1998.
[20] D. Basin, S. Mödersheim, and L. Vigano, "OFMC: A symbolic model checker for security protocols," *International Journal of* Information Security, vol. 4, no. 3, pp. 181–208, 2005.
[21] C. Meadows and P. Syverson, "Formalizing GDOI group key management requirements in NPATRL," in *Proc. 8th ACM Conference on Computer and Communications Security*, Nov 2001, pp. 235-244.
[22] D. Song, S. Berezin, and A. Perrig, "Athena: A novel approach to efficient automatic security protocol analysis," *Journal of Computer Security*, vol. 9, pp. 47–74, 2001.
[23] V. Kessler and G. Wedel, "AUTLOG - an advanced logic of authentication," in *Proc. Computer Security Foundations Workshop VII*, *IEEE Computer Society*, June 1994, pp. 90-99.
[24] S. Brackin, "Automatically detecting most vulnerabilities in cryptographic protocols," in *DARPA Information Survivability Conference and Exposition*, vol. 1, Hilton Head, South Carolina, January 2000, pp. 222–236.
[25] R. Dojen, I. Lasc, and T. Coffey, "Establishing and fixing a freshness flaw in a key-distribution and authentication protocol," in *Proc. IEEE International Conference on Intelligent Computer Communication and Processing*, Cluj-Napoca, Romania, 2008, pp. 185-192.



Anca D. Jurcut received a bachelor of mathematics and computer science from West University of Timisoara, Romania (2007) and a PhD from University of Limerick, Ireland (2013). She is currently working as a postdoctoral researcher in the Department of Electronic and Computer Engineering at the University of Limerick, Ireland. Her research interests are: cryptographic protocols analysis, logics-based verification techniques, data and network security, software engineering, and engineering mathematics.

Tom Coffey is professor of electronic and computer engineering at the University of Limerick, Ireland, where he is also director and founder of Data Communication Security Laboratory. He is a Chartered Engineer; he holds Master of Science (1978) and Doctor of Philosophy (1994) degrees from City University (London) and University of Ulster (Ireland) respectively. His research work encompasses: encryption systems, verifiably secure cryptographic protocols for open hostile environments, formal verification of security protocols using logic-based and state space-based techniques, generation of modal-logics of knowledge and belief, automated proving systems for security protocol verification.

Reiner Dojen received title of Dipl.-Ing.(FH) from University of Applied Sciences Osnabrück, Germany (1999), MEng from University of Limerick (2000) and PhD from University of Limerick (2004). He is currently employed as Lecturer in the Department of Electronic and Computer Engineering at the University of Limerick, Ireland. Research interests are: cryptography, security protocols, data and network security, automated theorem peorem P TDMEP interests are: roving and artificial intelligence.