COMPUTERS & SECURITY 45 (2014) 255–273

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Design guidelines for security protocols to prevent replay & parallel session attacks

Anca D. Jurcut, Tom Coffey, Reiner Dojen *

Department of Electronic & Computer Engineering, University of Limerick, Limerick, Ireland

ARTICLE INFO

Article history:
Received 26 September 2013
Received in revised form
28 February 2014
Accepted 27 May 2014
Available online 12 June 2014

Keywords:
Security protocols
Design guidelines
Attack detection
Replay attacks
Parallel session attacks
Freshness of messages
Symmetry of messages
Signed messages
Challenge-response handshake

ABSTRACT

This work is concerned with the design of security protocols. These protocols are susceptible to intruder attacks and their security compromised if weaknesses in the protocols' design are evident. In this paper a new analysis is presented on the reasons why security protocols are vulnerable to replay and parallel session attack and based on this analysis a new set of design guidelines to ensure resistance to these attacks is proposed. The guidelines are general purpose so as to encompass a wide spectrum of security protocols.

Further, an empirical study on the effectiveness of the proposed guidelines is carried out on a set of protocols, incorporating those that are known to be vulnerable to replay or parallel session attacks as well as some amended versions that are known to be free of these weaknesses. The goal of this study is to establish conformance of the set of protocols with the proposed design guidelines. The results of the study show that any protocol following the design guidelines can be considered free of weaknesses exploitable by replay or parallel session attacks. On the other hand, if non-conformance of a protocol with the design guidelines is determined, then the protocol is vulnerable to replay or parallel session attacks.

1. Introduction

Cryptographic protocols play an important role in today's communications environment, where they are used to provide a wide variety of security services, such as: key distribution, data confidentiality, authentication and non-repudiation.

The design of provably secure protocols is complex and prone to error, where the main difficulty is to address the vast possibilities of an adversary to gain information (Dojen and Coffey, 2005). These protocols can be vulnerable to a host of subtle attacks that compromise the services they provide; so designing them to be impervious to such attacks has proved to be extremely challenging. Many published security protocols have subsequently been found to contain security weaknesses that are exploitable by attacks. The public key authentication of Needham and Schroeder (Needham and Schroeder, 1978), for example, was considered secure for over a decade. Other protocol weaknesses exploitable by attacks can be found in: (Denning and Sacco, 1981; Burrows et al., 1990; Syverson, June 1994; Lowe, 1996; Abadi, March 1997; Heather et al., 2000). The difficulty of designing security protocols that are free of mountable attacks continues today, as highlighted by many

* *Corresponding author.*
E-mail addresses: anca.jurcut@ul.ie (A.D. Jurcut), tom.coffey@ul.ie (T. Coffey), reiner.dojen@ul.ie (R. Dojen).
http://dx.doi.org/10.1016/j.cose.2014.05.010
0167-4048/

recently found instances of replay and parallel session attacks:

- attacks found in 2003 (Coffey et al., December 2003) on several authentication and key agreement protocols for mobile communications (Beller et al., 1993; Carlsen, 1994; Mu & Varadharajan, 1996)
- attacks found in 2006 (Xu & Huang, September, 2006) and 2008 (Altaf et al., 2008) on privacy and key management IEEE Std. 802.16e-2005 protocol, (IEEE Std. 802.16e/D12, 2005)
- attacks (Nam et al., Jan 2007) found in 2007 on an authentication protocol introduced in 2005 (Lee et al., 2005)
- attack found in 2008 (Espelid et al., 2008) on Norway national security for e-commerce protocol BankID introduced in 2006 (The Norwegian Banks' Payment and Clearing Centre, 2006)
- attack found in 2008 (Xu et al., July, 2008) on a fingerprint-based authentication scheme with smart cards introduced in 2006 (Khan and Zhang, 2006)
- attacks found in 2008 (Dojen et al., 2008b) on a Key Management Protocol for Wireless Sensor Networks (Shen et al., 2008)
- attack found in 2008 (Dojen et al., 2008c)on a key distribution protocol (Lowe, 1997)
- attack found in 2009 (Hsiang and Shih, 2009) on a remote user authentication scheme using smart cards introduced in 2005 (Yoon and Yoo, 2005)
- attacks found in 2009 (Dojen et al., 2009) on an end-to-end authentication and secrecy protocol introduced in 2003 (Lee et al., 2003)
- attacks found in 2012 (Yoo et al., 2012) on an authentication scheme introduced in 2009 (Das, 2009) and its derivatives (Nyang and Lee, 2009; Huang et al., October 2010; Chen and Shih, 2010; Khan and Alghathbar, 2010)
- attacks found in 2013 (Wang and Ma, 2013) on ID-based scheme for mobile client–server environment introduced in 2012 (He et al., 2012)
- attacks found in 2013 (Fu and Guo, 2013) on lightweight RFID mutual authentication protocol introduced in 2011 (Jin et al., 2011)
- attack found in 2013 (Zhuang et al., July 2013) on RAPP ultra-lightweight RFID protocol introduced in 2012 (Tian et al., 2012)

1.1. *Original contribution of this work*

The research work presented in this paper is concerned with the design of security protocols, in particular the prevention of design weaknesses that may be subsequently exploited by replay or parallel session attacks.

A new analysis on the reasons why freshness and parallel session attacks against security protocols succeed is presented. This analysis discovers the vulnerabilities in the structure of the protocol message exchanges that can be exploited by these attacks. Specifically, the analysis seeks answers to the questions:

- Why are freshness and parallel session attacks successful?
- Can the reasons be represented by a finite set of data patterns representing the message exchanges?
- Is it possible to develop a finite set of protocol design rules or guidelines that will prevent the effectiveness of these attacks?

A new comprehensive set of guidelines for security protocols to prevent design weaknesses that are exploitable by replay or parallel session attacks is proposed. The guidelines are general purpose so as to encompass a wide spectrum of security protocols. The effectiveness of the guidelines is also established in a presented case study which shows that: (i) protocols with known weaknesses violate some of the guidelines, (ii) protocols without weaknesses do not violate any guidelines.

These guidelines are intended to be used at the design stage of security protocols. Any protocol following these guidelines can be considered to be free of any weaknesses exploitable by replay or parallel session attacks. On the other hand, if non-conformance of a protocol with the design guidelines is established, then the protocol is vulnerable to replay or parallel session attacks.

1.2. *Paper structure*

The remainder of this paper has the following structure. Section 2 gives an overview of related work and Section 3 outlines the analysed methodology used in this work. Section 4 introduces the language and definitions used through the paper. Each defined weakness type is then separately analysed and corresponding design guidelines are proposed: Section 5 addresses the issue of freshness of messages, Section 6 addresses the issue of symmetry of messages, Section 7 addresses the issue of signed messages and Section 8 addresses the issue of challenge-response handshake construction. In Section 9, the effectiveness of the proposed guidelines is evaluated by way of an empirical study on a set of protocols with known weaknesses and those that are known to be secure. The conformance of one protocol (and its amended version) is analysed in detail and the conformance of a range of protocols is presented in summary form. Finally, Section 10 concludes the paper. A summary of the proposed design guidelines is included in Appendix A.

2. Related work

The design of reliable and trustworthy security protocols has been addressed by a series of publications over the past two decades. Bird et al. (Bird et al., 1992) introduced in 1992 a two-way authentication protocol using symmetric key cryptography and gave a set of considerations to avoid weaknesses in the design of these types of protocols. This work was extended in 1993 (Bird et al., June 1993) in the form of a methodology to systematically build a family of cryptographic two-way authentication protocols that are resistant to a number of attacks. In order to protect protocol messages from being vulnerable to replay attacks, Carlsen (Carlsen, June 1994) provided a list (list included: protocol identifier, step identifier, message subcomponents identifier, primitive types of data items and protocol run identifier) of information that should be attached to cyphertexts. Gong and Syverson (Gong &

Syverson, September 1995) presented the notion of fail-stop protocols over a restrictive class of protocol design rules that avoid replay attacks under certain conditions. However, the authors indicate that some protocols may have other requirements which conflict with those of fail-stop protocols.

In 1996 Abadi and Needham (Abadi and Needham, 1996) proposed a set of basic principles for strengthening the design of security protocols. They addressed two major issues in particular: i) the messages involved in a protocol together with their content and ii) the dependent trust relationship of the protocol participants. Anderson and Needham (Anderson and Needham, 1995) extended this work by incorporating principles to avoid protocol design weaknesses, when using public and private key encryption. However, these sets of principles do not guarantee protocol correctness as they only act as "rules of thumb" for protocol designers (Anderson and Needham, 1995).

Aura (Aura, June 1997) suggested recommendations to avoid replay attacks, which include: type-tagging messages with unique cryptographic functions and how to produce unique session keys without assuming mutual trust between the principals. These recommendations do not guarantee protocol security, but are specific ways for improving the robustness of the designed protocol. Malladi et al. (Malladi et al., 2002) in 2002 also put forward a recommendation to avoid replay attacks by associating with every protocol run a session-id generated by all protocol participants. The authors present a proof that this association guarantees that no replay attack can be elaborated, but did not provide a mechanism on how the participants agree the session-id.

Lasc, Dojen and Coffey (Lasc et al., 2013) identified desynchronisation attacks in 2013 on a group of protocols that use dynamic shared secrets update mechanisms for wireless communications. The authors presented design guidelines to prevent such weaknesses and a formal system to model update mechanisms for shared secrets.

The work on designing new reliable security protocols is on-going to-date, as is the identification and fixing of design weaknesses in existing protocols (Kumar et al., 2011; Ahirwal & Sonwanshi, February 2012; Yoo et al., 2013). The remainder of this paper is concerned with reasoning why replay and parallel session attacks are successful and how protocols can be designed so that they are not vulnerable to these attacks.

3. Analysis methodology

The goal of this analysis is to establish why replay and parallel session attacks succeed on a broad spectrum of security protocols, to propose a set of protocol design guidelines to counter these attacks and to establish the effectiveness of the proposed guidelines.

3.1. *Methodology*

The methodology adopted for this analysis is to characterise the general circumstances under which replay and parallel session attacks may exist by examining the structure of message exchanges of a large set of security protocols with known vulnerabilities. The set of published security protocols incorporates those found to be vulnerable to replay and parallel session attacks and also the published protocol fixes for the prevention of these attacks. Our investigation takes into account: (i) the knowledge of the principals involved, (ii) the role of the messages in the protocol, (iii) the way messages are transmitted and (iv) the content of messages.

On completing the examination of the structure of message exchanges for the considered set of security protocols, a finite set of message exchange patterns is derived and grouped into four categories. These categories are as follows:

1. freshness of messages
2. symmetry of messages
3. signed messages
4. challenge-response handshake construction

For each of these pattern categories a new set of design guidelines addressing problems leading to replay and parallel session attacks is elaborated in the following sections.

4. Language and definitions

This section introduces the language, considerations, definitions and predicates used through this paper. The following notation is used to classify the variables and constants of the language:

- **P:** protocol
- **w, x, y, z:** data objects (atomic or composite)
- **A, B, C:** legitimate principals
- **TTP:** trusted third party principal
- **I(A)/I(B):** intruder masquerading as principal A/B
- **ENT(P):** set of all principals involved in a protocol P
- **x** : data component generated by principal A
- **k:** any cryptographic key, where concrete keys are denoted: K_{AB} for a symmetric key shared by A and B, K_{APub} for a public key of A and K_{APriv} for a private key of A
- T_A: timestamp of principal A
- **TS(P):** set of all timestamps of P
- N_A: nonce generated by principal A
- **{x}k:** cryptographic expression: data x encrypted with key k
- **F(x):** generic function applied to data x
- **H(x):** one way hash function applied to data x
- S_q, S_r: steps of protocol P
- **m(S_r):** the set of all data transmitted in step S_r
- **s(S_r):** the sender of step S_r
- **r(S_r):** the recipient/receiver of step S_r
- **CT(P):** set of all cryptographic transformations of P
- **CRH(P,A,B):** set of all challenge-response handshakes in a protocol P, where principal A is acting as the verifier and B as the prover

The language includes the classical logical connectives of conjunction ($\wedge$), disjunction ($\vee$), negation ($\neg$) and material implication ($\rightarrow$). Further, $\in$ indicates set membership, $\setminus$ denotes set exclusion and {} denotes the empty set. The symbol comma (,) denotes concatenation, $=$ denotes equality and $\neq$ denotes inequality, while the symbols $\forall$ and $\exists$ denote universal and existential quantification respectively. In a universal quantification $\forall x|PR(x):Z$ the predicate $PR(x)$ defines the

domain of the quantified variable x. If the predicate *PR(x)* indicates a set membership, the shorthand $\forall x \in S{:}Z$ is used. Existential quantification uses the same notation. However, definition of the domain can be omitted for existential quantifications. Individual values, such as principals, cryptographic keys, nonces or arbitrary binary values form the atoms of the language. Valid data objects (also called components) are defined recursively as follows: Any atom φ is a valid data object. If b and c are valid data objects, then *b*,*c*; (*b*,*c*); {*b*}c; *F(b)* and *H(b)* are also valid data objects.

A formula *a* of the language can be defined recursively as follows:

$$a ::= PR(x) | \neg b | b \wedge c \mid b \vee c \mid b \rightarrow c \mid y \in S \mid S \setminus y \mid S \setminus S_1 \mid b = c \mid b \neq c \mid \forall x | PR(x){:}\ c \mid \exists x | PR(x){:}\ c$$

where PR is a predicate, x is a data object that satisfies the requirements of PR in arity and type, b and c are formulae of the language and S and S_1 are sets of equal types.

All communications considered are intended to be between principals A, B and potentially a trusted third party TTP. It assumed that an intruder has full control over the communications environment, i.e. that the intruder can intercept, modify and replay messages or start new executions of the protocol. The intruder can be an outside agent or a dishonest legitimate principal of the system. An ideal cryptographic environment is assumed, where ciphertext can only be decrypted with the required key and keys are only possessed by their legitimate owners.

Definition 0. A principal is an entity that can be authenticated by a computer system or network.

Definition 1. A protocol P is a set of ordered steps $\{S_1, S_2, \ldots, S_z\}$, $z \geq 1$, executed in any run of P. A protocol step S_r is defined as: S_r: A → B: m, where A and B are principals involved in P and m is the message transmitted. A is the sender of message m of step S_r ($A = s(S_r)$) and B is the recipient ($B = r(S_r)$).

Definition 2. An initiation step is any step in a protocol, where any principal that is not a TTP is the sender. The set of all initiation steps is denoted by IS(P).

Definition 3. A response step S_p is the first step after an initiation step S_o, where the recipient of S_p is the sender of S_o. The set of all response steps of all principals is denoted by RS(P).

Definition 4. A protocol run is a single execution of the ordered set of steps $\{S_1, S_2, \ldots, S_z\}$ of a protocol P. When multiple runs of a protocol are considered, the number of the run is indicated by an integer superscript, e.g. run a of protocol P is denoted P^a. A step S_n of protocol run P^a is denoted S_n^a.

Definition 5. A cryptographic transformation c is either a cryptographic expression ({x}k) or a hashed expression (H(x)). $CT(S_n)$ denotes the set of all cryptographic transformations transmitted in a step Sn and CT(P) the set of all cryptographic transformations of a protocol P.

Definition 6. A principal A recognises a component x as fresh (*Fresh*(*A*,*x*)) if x is a timestamp (under the assumption of synchronized clocks) or x is a function of a component w_A freshly generated and sent by A in a previous initiation step of the same protocol run.

Definition 7. A signed statement is a cryptographic expression where the signing key is a private key.

Definition 8. Two expressions x and y are symmetric if both contain components of the same type and in the same order.

Definition 9. Two cryptographic expressions {x}k1, {y}k2 are symmetric if x, y are symmetric and keys k1, k2 are either:

- identical
- both symmetric and are shared with a TTP
- two different public keys
- two different private keys

Definition 10. The generator of an expression x is the sender of the step S_r, in which x appears for the first time.

Definition 11. The intended recipient of a message x of a protocol P is a principal who is the recipient of x and is able to decrypt or verify the content of x.

Definition 12. Two messages x and y are principal value type equivalent (*Pvte*(*x*,*y*))if for each subcomponent x_i at position i of x that is of type principal there is a corresponding subcomponent y_i at the same position i of y that is also of type principal and at least one of the following also holds:

- if x_i is a trusted third party (TTP) then y_i is also a trusted third party (TTP)
- if x_i is the generator of x then y_i is the generator of y
- if x_i is not the generator of component x then y_i is not the generator of y
- if x_i is the intended recipient of x then y_i is the intended recipient of y
- if x_i is not the intended recipient of x then y_i is not the intended recipient of y

Definition 13. A cryptographic transformation c is a parent cryptographic transformation (*Π*(c)) if it is not contained in any other cryptographic transformation.

Definition 14. Two cryptographic transformation c1, c2 are travelling in opposite direction (↑↓(*c1*,*c2*)) if they are exchanged directly or indirectly between two principals.

Definition 15. A receiver bound cryptographic transformation (*RBound*(*B*,*c*)) is either a:

- cryptographic expression that contains at least one component that identifies the recipient or
- cryptographic expression that is encrypted with the public key of the recipient or
- cryptographic expression that is encrypted with a symmetric key shared with the recipient and which contains some secret data only known by both principals or

- hashed expression that contains some secret data only known by both principals and at least one component that identifies the recipient.

Definition 16. A challenge-response nonce handshake is a message exchange where fresh data x_A is a challenge generated by a principal acting as verifier and then delivered as part of a cryptographic expression to a principal acting as prover. The prover authenticates itself to the verifier by means of the received fresh data x_A. The set of all challenge-response handshakes in a protocol P having principal A as a verifier and B as a prover is denoted by CRH(P,A,B).

The following predicates are defined to evaluate properties of message exchanges and their components as well as principals:

- C(x,y,S_r): component x contains component y at step S_r
- Gen(A, x, S_r): Principal A generated x in step S_r – x cannot appear prior to step S_r. By convention, the generator is the sender of the step S_r, in which x appears for the first time
- KMaterial(x): x is key material, i.e. object x or some of its components are used in the generation of a key
- Symmetric(x,y): objects x and y are symmetric (see *Definition 8*)
- Π(c): c is parent cryptographic transformation (see *Definition 13*)
- Fresh(A, x): x is fresh for principal A (A recognise component x as fresh)
- ↑↓(*c1*,*c2*): cryptographic transformations c1 and c2 are travelling in opposite directions (see *Definition 14*)
- RBound(B, x): x is receiver bound (see *Definition 15*)

5. Analysing problems related to freshness of messages

Consider two principals A and B involved in a protocol P ($A,B \in ENT(P)$). In an arbitrary step S_r of the protocol P, A sends a message containing a cryptographic expression {x}k to B ($\exists S_r \in P, S_r{:}A \rightarrow B{:}\{x\}k$). If this cryptographic expression does not have at least one component fresh for recipient B ($\neg\exists y|C(\{x\}k,y,S_r) \wedge Fresh(y,B)$) then the message containing {x}k can be replayed. An intruder (I) can mislead the recipient (B) to accept old and possibly compromised data as a component of the cryptographic expression from message containing {x}k. Further, if message containing {x}k is used for authentication, then the intruder can get authenticated by substituting a previously recorded message (from A to B) for {x}k. In the following cases freshness-based attacks can be mounted:

Case 1. Absence of synchronised clocks.

Assume $\exists S_q^a \in P^a$ such that q<r, $s(S_q^a) = B$ and $r(S_q^a) = C$. If $\forall\{x\}k \in m(S_r^a), \forall y|C(\{x\}k, y, S_r^a) \wedge \neg Fresh(y,B)$ then an intruder I can replay message S_r^a of run a instead of message S_r^b in a second run b. Principal B has no way to detect the replay, as {x}k does not contain any component which B recognizes as being fresh. The attack is outlined in Fig. 1.

S^a_q. B→ C: y
S^a_r. A → B: {x}k
S^b_q. B→ C: z
S^b_r. I(A) → B: {x}k

Fig. 1 – Replay attack in Case 1.

Case 2. Synchronized clock environment.

When timestamps are used by a protocol P, if $\forall\{x\}k \in m(S_r^a), \forall t \in TS(P) : \neg C(\{x\}k, t, S_r^a)$, an intruder I can replay message S_r^a instead of S_r^b. Principal B has no way to detect the replay, as the cryptographic expressions {x}k of S_r^a does not contain any fresh component.

Case 3. Exchange of key material.

Assume $\exists S_q^a \in P^a$ such that q<r, $s(S_q^a) = B$ and $r(S_q^a) = C$. If $\forall\{x\}k \in m(S_r^a), \exists KMaterial(w)|C(\{x\}k, w, S_r^a) \wedge \forall y|C(\{x\}k, y, S_r^a) \wedge \neg Fresh(y,B)$ then an intruder I can replay message S_r^a of run a instead of message S_r^b in a second run b. Principal B has no way to detect the replay, as {x}k does not contain any component which B recognizes as being fresh. Thus, B considers the replayed S_r^a a legitimate message S_r^b and accepts the contained old key material w. The attack is outlined in Fig. 2.

Flaws corresponding to all cases in this section have been revealed on security protocols such as Andrew Secure RPC protocol (Burrows et al., 1990), Needham Schroeder symmetric key protocol (Denning and Sacco, 1981), Neumann Stubblebine protocol (Hwang et al., 1995), Hwang and Chen modified SPLICE/AS protocol (Clark and Jacob, 1995), TMN protocol (Lowe and Roscoe, 1997) and Kao-Chow v1.protocol (Dojen et al., 2008a).

5.1. *Proposed guidelines to ensure message freshness*

In order to prevent freshness related issues exploitable by replay attacks, the following guidelines should be obeyed:

(GL1.1) In the absence of synchronized clocks, each cryptographic expression that is part of a response step in a protocol should contain a fresh component generated by the recipient in a previous step of the protocol.

$$\forall\{x\}k : \{x\}k \in m(S_p) \wedge S_p \in RS(P)\, \exists x_B | C(\{x\}k, x_B, S_p) \wedge \exists o : o < p \wedge Gen(B, x_B, S_o) \wedge B = r(S_p)$$

(GL1.2) If timestamps are used (with the assumption of synchronized clocks), then each cryptographic expression of a protocol should contain a timestamp.

$$\forall\{x\}k \in m(P) | \exists t \in TS(P) \wedge C(\{x\}k, t, S_r)$$

S^a_q. B→ C: y
S^a_r. A → B: {x,w}k
S^b_q. B→ C: z
S^b_r. I(A) → B: {x,w}k

Fig. 2 – Replay attack in Case 3.

S^a_q. A → B: c1
S^a_r. B → I(A): c2
S^b_q. I(B) → A: c2
S^b_r. A → I(B): c3

Fig. 3 – Parallel session attack in Case 1.

(GL1.3) A component w used in the generation of a key should be sent at least once as part of a cryptographic expression containing at least one component which the recipient recognizes as being fresh.

$$\forall KMaterial(w)\exists\{x\}k\in m(S_r)|C(\{x\}k,(x_B,w),S_r)\wedge B=r(S_r)\wedge Fresh(B,x_B)$$

6. Analysing problems related to symmetry of messages

Consider two cryptographic transformations c1 and c2 that are exchanged directly or indirectly ($\uparrow\downarrow(c1,c2)$) between two principals $A,B\in ENT(P)$. If c1 and c2 are symmetric ($Symmetric(c1,c2)$) and principal value type equivalent messages ($Pvte(c1,c2)$), then expression c2 can be used instead of expression c1 or c1 can be reconstructed based on expression c2 in a parallel protocol run. In the following cases symmetry-based attacks can be mounted:

Case 1. Expressions c1 and c2 are exchanged directly in steps S_q and S_r: Assume $\exists S^a_q,S^a_r\in P^a$ such that:$C(m(S^a_q),c1,S^a_q)$ and $C(m(S^a_r),c2,S^a_r)$. If $\uparrow\downarrow(c1,c2)$ then expression c2 obtained in step S^a_r of run a is used in step S^b_q of a parallel run b as outlined in Fig. 3. As the expression c1' required in step S^b_q is symmetric and principal value type equivalent with c2 ($Symmetric(c1',c2)\wedge Pvte(c1',c2)$), the receiving principal cannot distinguish the replayed c2 from a genuine expression c1'.

Case 2. Expressions c1 and c2 are exchanged indirectly (via a TTP) in steps S_q and S_r: Sender of S_q creates c1 and sender of S_r creates c2.

Assume $\exists S^a_q,S^a_r\in P^a$ such that: $Gen(A,c1,S^a_q)$ and $Gen(B,c2,S^a_r)$. If $\uparrow\downarrow(c1,c2)$, essentially, the same rationale as in case 1 applies: As outlined in Fig. 4, expression c2 of first run is used as c1' in the second run (alternatively, c1 of first run can also be used as c2' in second run).

Case 3. Expressions c1 and c2 are exchanged indirectly (via TTP) in steps S_q and S_r: Sender of S_q creates expression c1 and TTP creates expression c2.

Assume $\exists S^a_q,S^a_r\in P^a$ such that: $Gen(A,c1,S^a_q)$ and $Gen(TTP,c2,S^a_r)$. As outlined in Fig. 5, expression c2 of the run a

S^a_q. A → TTP: c1
S^a_r. B → I(TTP): c2
S^b_q. I(B) → TTP: c2
S^b_r. A → TTP: c3

Fig. 4 – Parallel session attack in Case 2.

S^a_q. A → I(TTP): c1
S^a_r. TTP→ I(B): c2
S^b_q. I(B) → TTP: c2
S^b_r. TTP→ A: c3

Fig. 5 – Parallel session attack in Case 3.

is used as expression c1' in the parallel run b. As $Symmetric(c1,c2)\wedge Pvte(c1,c2)$, TTP cannot distinguish the replayed c2 from a genuine expression c1'.

Case 4. Expressions c1 and c2 are exchanged indirectly (via TTP) in steps S_q and S_r: TTP creates both expressions c1, c2.

Assume $\exists S^a_q,S^a_r\in P^a$ such that: $Gen(TTP,c1,S^a_q)$ and $Gen(TTP,c2,S^a_r)$. As outlined in Fig. 6, an attacker can replay expression c2 obtained in run S^a_q as c1' in a parallel run b. As $Symmetric(c1,c2)\wedge Pvte(c1,c2)$, the receiver of S^b_q cannot distinguish the replayed c2 from a genuine expression c1'.

Flaws corresponding to all cases in this section have been revealed on security protocols such as BAN simplified version of the Yahalom protocol (Syverson, June 1994), Wide-Mouthed Frog protocol (Burrows et al., 1990), Lowe's modified version of the Wide-Mouthed Frog protocol (Dojen et al., 2008c), KSL protocol (Lowe, 1996) and Lee et al. Nonce-based user authentication scheme (Nam et al., Jan 2007).

6.1. *Proposed guidelines to prevent message symmetry*

In order to prevent symmetry related issues exploitable by parallel session attacks, the following guideline should be obeyed:

(GL2) If two principals exchange a pair of cryptographic transformations c1 and c2 directly or indirectly via a TTP, then c1 and c2 should not at the same time be symmetric and principal value type equivalent.

$$\forall c1,c2\in CT(P)|\uparrow\downarrow(c1,c2)\wedge\neg(Symmetric(c1,c2)\wedge Pvte(c1,c2))$$

7. Analysing problems related to signed messages

Consider a protocol P, where in step S_r a signed statement $\{x\}K_{APriv}$ is transmitted. If this signed statement is not receiver bound ($\neg RBound(B,\{x\}K_{APriv})$) and does not include appropriate identities of principals, then an intruder can use the signed statement $\{x\}K_{APriv}$ in a parallel run to impersonate a legitimate principal. In the following cases signed message-based attacks can be mounted:

S^a_q. TTP →A: c1
S^a_r. TTP→ I(B): c2
S^b_q. I(TTP) → B: c2

Fig. 6 – Parallel session attack in Case 4.

S^a_r. A → I(B): {x}K_{APriv}
S^b_r. I(A) → C: {x}K_{APriv}

Fig. 7 – Parallel session attack in Case 1.

Case 1. {x}K_{APriv} is a parent cryptographic expression:

Assume $\exists S_r \in P$ such that $A = s(S_r)$ and $B = r(S_r)$. If $\{x\}K_{APriv} \in S_r | \Pi(\{x\}K_{APriv}) \wedge \neg RBound(B,\{x\}K_{APriv})$, an intruder I can impersonate principal A as outlined in Fig. 7. I replays message {x}K_{APriv} obtained in step S^a_r of run *a* as valid signed message of S^b_r in a parallel run *b* with any principal C. C has no way to establish that message {x}K_{APriv} was originally intended for B.

Case 2. {x}K_{APriv} is a contained by a parent cryptographic expression {y}K_{BPub} that is encrypted with B's public key:

Assume $\exists S_r \in P$ such that $A = s(S_r)$ and $B = r(S_r)$. If $\exists \{y\}K_{BPub} \in m(S_r)$ such that $C(\{y\}K_{BPub},\{x\}K_{APriv},S_r) \wedge \Pi(\{y\}_{KBPub})$ and further $\forall c | C(y,c,S_r) \wedge \neg RBound(B,c)$ (i.e. component y does not contain any receiver bound cryptographic transformation), a dishonest principal I can impersonate A as outlined in Fig. 8: I can use the subcomponents of {y}K_{IPub} obtained in step S^a_r of run *a* to compute {y'}K_{CPub} as a valid expression of S^b_r in a parallel run *b* with principal C. The attacker removes the outer encryption of {y}K_{IPub}, replaces any data indicating I's identity with data indicating C's identity and re-encrypts the required data y' with the public key of C. As y is free of any receiver bound cryptographic transformation, all data indicating I's identity in y exist outside of cryptographic transformations and can be modified by I. Principal C is unable to detect that the content of {y'}K_{CPub} was intended for I. Further, as {y'}K_{CPub} contains signed message {x}K_{APriv}, C believes that the message was sent by principal A.

Case 3. {x}K_{APriv} is a contained by a parent cryptographic expression {y}K_{AB} that is encrypted with symmetric key by A and B:

Assume $\exists S_r \in P$ such that $A = s(S_r), B = r(S_r)$ and $\exists \{y\}K_{AB} \in m(S_r)$ such that $C(\{y\}K_{AB},\{x\}K_{APriv},S_r) \wedge \Pi(\{y\}K_{AB})$. Further, key K_{AB} is generated either by A or TTP (this generating principal is subsequently identified by C) and is transmitted as part of a cryptographic expression {z}K_{BPub} in step S_q prior to S_r ($\exists q < r | Gen(C,\{z\}K_{BPub},S_q) \wedge C(z,K_{AB},S_q)$). Moreover, $\forall \{w\}K_{CPriv} | C(z,\{w\}K_{CPriv},S_q) \wedge \neg RBound(B,\{w\}K_{CPriv})$. If $\forall c | C(y,c,S_r) \wedge \neg RBound(B,c)$, a dishonest principal I can impersonate A as outlined in Fig. 9: I can use the subcomponents of {z}K_{IPub} obtained in step S^a_q of run *a* to compute {z'}K_{BPub} as a valid expression of S^b_r in a parallel run b with principal B. This is accomplished by removing the outer encryption of {z}K_{IPub}, replacing any data indicating identity of receiver I with data indicating B and re-encrypting z', which includes symmetric key K_{AI}, with the public key of B. As z is free of any receiver bound constituents all data indicating receiver's identity exists outside of cryptographic transformations and can be modified by I. Consequently, B believes {z'}K_{BPub} is a legitimate

S^a_r. A → I: {...,{x}K_{APriv},...}K_{IPub}
S^b_r. I(A) → C: {...,{x}K_{APriv},...}K_{CPub}

Fig. 8 – Parallel session attack in Case 2.

S^a_q. C → I: {...,K_{AI},...}K_{IPub}
S^b_q. I(C) → B: {...,K_{AI},...}K_{BPub}
S^a_r. A → I: {...,{x}K_{APriv},...}K_{AI}
S^b_r.I(A) → B: {...,{x}K_{APriv},...} K_{AI}

Fig. 9 – Parallel session attack in Case 3.

message from C and accepts the contained key K_{AI} as a session key for communication with A. In a similar process attacker I can create component {y'} K_{AI} that contains {x}K_{APriv} as required for step S^b_r. B has no way to detect that {y'} K_{AI} was originally intended for I, as the message is not receiver bound. Further, as {y'} K_{AI} contains signed message {x}K_{APriv}, B believes that the message was sent by principal A.

Case 4. Signed statements for public key distribution:

Assume $\exists S_q \in P$ such that $A = s(S_q), TTP = r(S_q)$ and $\exists request_B \in m(S_q)$ (i.e. principal A sends a request for B's public key to a TTP). Further, $\exists S_r \in P$ such that $A = r(S_r)$ and $\{K_{BPub},x\}K_{TTPPriv} \in m(S_r) \wedge \{K_{BPub},x\}K_{TTPPriv}) \in DC(P)$ (i.e. TTP responds to A's request by emitting a signed statement containing B's public key). If $\neg C(\{K_{BPub},x\}K_{TTPPriv},B,S_r)$ then it is not possible to bind the public key to its owner (B). Thus, a legitimate, but dishonest principal I can mislead other principals to bind its own public key K_{IPub} to any other principal as outlined in Fig. 10. In run *a* principal I intercepts message S^a_q intended for TTP and starts a second run *b*. In this second run I pretends to be A by sending a request for I's public key to TTP. According to the specification of the exchange, TTP sends the signed statement in step S^b_r. As this signed statement does not contain the identity of the owner of the public key, I can replay it as a valid signed statement containing principal B's key in step S^a_r of the first run. Principal A cannot distinguish the replayed S^b_r containing I's public key from a genuine response containing the B's public key and will assume the contained public key belongs to B. Thus, any message that A wants to send in confidence to B will be encrypted with the wrong public key K_{IPub} and I can successfully impersonates B to A.

Flaws corresponding to all cases in this section have been revealed on security protocols such as Denning Sacco Public Key protocol (Abadi and Needham, 1996), CCITT X.509 (3) protocol (Burrows et al., 1990), SSH and version three of AKA protocols (Abadi, March 1997), the SSL protocol (Abadi and Needham, 1996) and the SPLICE/AS protocol (Yamaguchi et al., November 1991).

7.1. *Proposed guidelines for signed messages*

In order to prevent signed messages related issues exploitable by parallel session attacks, the following guidelines should be obeyed:

S^a_q. A → I(TTP): $request_B$
S^b_q. I(A) → TTP: $request_I$
S^b_r. TTP → I(A): {K_{IPub},z}$K_{TTPPriv}$
S^a_r. I(TTP) → A: {K_{IPub},z}$K_{TTPPriv}$
S^a_u. A → I(B): {w}K_{IPub}

Fig. 10 – Parallel session attack in Case 4.

(GL3.1) If a signed message $\{x\}K_{APriv}$ is a parent cryptographic expression, then $\{x\}K_{APriv}$ should be receiver bound.

$$\exists S_r \in P : s(S_r) = A \wedge r(S_r) = B \wedge \forall \{x\}K_{APriv} \in m(S_r) | \Pi(\{x\}K_{APriv}) : RBound(B, \{x\}K_{APriv})$$

(GL3.2) If a signed message $\{x\}K_{APriv}$ of a step S_r is contained by a parent cryptographic expression $\{y\}K_{BPub}$ encrypted with the public key of recipient B of S_r, then component y should contain at least one receiver bound cryptographic transformation.

$$\exists S_r \in P : s(S_r) = A \wedge r(S_r) = B \wedge \forall \{x\}K_{APriv} \in m(S_r), \exists \{y\}K_{BPub} \in m(S_r) | C(\{y\}K_{BPub}, \{x\}K_{APriv}, S_r) \wedge \Pi(\{y\}K_{BPub}), (\forall c | C(y, c, S_r) \wedge RBound(B, c))$$

(GL3.3) If a signed message $\{x\}K_{APriv}$ of a step S_r of protocol P is contained in a parent cryptographic expression $\{y\}K_{AB}$ that fulfils the following conditions:

- the symmetric key K_{AB} is generated by a principal C belonging to {TTP, A} and is transmitted by C in a cryptographic expression $\{z\}K_{BPub}$ encrypted with B's public key in step S_q prior to S_r
- $\{z\}K_{BPub}$ in step S_q is free of any receiver bound messages signed by C

then component y should contain at least one receiver bound cryptographic transformation.

$$\exists S_r \in P : s(S_r) = A \wedge r(S_r) = B \wedge \forall \{x\}K_{APriv} \in m(S_r), \exists \{y\}K_{AB} \in m(S_r) | C(\{y\}K_{AB}, \{x\}K_{APriv}, S_r) \wedge \Pi(\{y\}K_{AB}) \wedge (\exists q < r | Gen(C, \{z\}K_{BPub}, S_q) \wedge C(z, K_{AB}, S_q) \wedge (\forall \{w\}K_{CPriv} | C(z, \{w\}K_{CPriv}, S_q) \wedge \neg RBound(B, \{w\}K_{CPriv}))) : (\forall c | C(y, c, S_r) \wedge RBound(B, c))$$

(GL3.4) Any signed statement intended for public key distribution should include the identity of the public key owner.

$$\exists S_r \in P : \forall \{K_{BPub}, x\}K_{TTPPriv} \in m(S_r) | C(\{K_{BPub}, x\}K_{TTPPriv}, B, S_r)$$

8. Analysing problems related to challenge-response handshakes construction

This section discusses issues related to challenge-response handshakes in security protocols. To facilitate the presented analysis a new classification for challenge-response handshakes and the criteria upon which they are based is introduced.

8.1. Classifying challenge-response handshakes

The following criteria are used to classify challenge-response handshakes:

1) The number of steps in the handshake.
2) Whether or not a trusted third party (TTP) is involved in the handshake.
3) The messages structure of the handshake.

8.1.1. Categories by number of steps

A *two-way challenge-response handshake* contains two steps: firstly the verifier sends out the nonce and then the prover returns it. Steps 3 and 4 of the DJCG Lowe's modified version of the Wide-Mouthed Frog protocol (Dojen et al., 2008c) are an example of a two-way challenge-response handshake:

$B \rightarrow A : \{N_B\}K_{AB}$
$A \rightarrow B : \{succ(N_B)\}K_{AB}$

A *three-way challenge-response handshake* contains three steps: firstly the verifier sends out the nonce, secondly, the prover returns it including its own nonce and finally, the verifier returns the prover's nonce. Steps 5, 6 and 7 of the Lowe's fixed version of Needham–Schroeder Public-Key protocol (Lowe, Nov 1995) are an example of a three-way challenge-response handshake:

$A \rightarrow B : \{N_A, A\}K_{BPub}$
$B \rightarrow A : \{N_A, N_B, B\}KK_{APub}$
$A \rightarrow B : \{N_B\}K_{BPub}$

The above three-way challenge-response handshake can be divided into two different two-way challenge-response handshakes as follows:

The first two-way challenge-response handshake:

$A \rightarrow B : \{N_A, A\}K_{BPub}$
$B \rightarrow A : \{N_A, N_B, B\}K_{APub}$

The second two-way challenge-response handshake:

$B \rightarrow A : \{N_A, N_B, B\}K_{BPub}$
$A \rightarrow B : \{N_B\}K_{BPub}$

A *four-way challenge-response handshake* contains four steps: First steps are the same as in a three-way challenge-response, followed by a "nonce update" by one principal. Steps 1, 2, 3 and 4 of the BAN modified Andrew Secure RPC protocol (Burrows et al., 1990) are an example of four-way challenge-response handshake:

$A \rightarrow B : \{N_A\}K_{AB}$
$B \rightarrow A : \{succN_A, N_B\}K_{AB}$
$A \rightarrow B : \{succN_B\}K_{AB}$
$B \rightarrow A : \{K'_{AB}, N'_B, N_A\}K_{AB}$

As in the case of a three-way handshake, the above four-way challenge-response handshake can be divided into three different two-way handshakes.

8.1.2. Categories by TTP involvement

In a *direct challenge-response handshake* the messages are exchanged directly between principals without involvement of a trusted third party (TTP). All handshake examples given in 8.1.1 are direct challenge-response handshakes.

In an *indirect challenge-response handshake*, the messages are exchanged indirectly between principals with the assistance of a TTP. This is needed in cases where the principals do not possess a shared key. Paulson's strengthened version of the Yahalom protocol (Paulson, 2001) is an example of an indirect challenge-response handshake:

$A \rightarrow B : A, N_A$
$B \rightarrow TTP : \{A, N_A\}K_{BTTP}$
$TTP \rightarrow A : \{B,\ K_{AB}, N_A\}K_{ATTP}$

8.1.3. *Categories by structure of handshake's messages*

Categories by message structure are defined in accordance with that given in (Gordon and Jeffrey, 2004):

- *POSH* (*Public Out Secret Home*) *challenge-response handshake*: The fresh data goes out in the clear and returns encrypted. An example of a POSH challenge-response handshake can be found in the repeated authentication part of Lowe's modified KSL protocol (Lowe, 1996):

$A \rightarrow B : N'_A$
$B \rightarrow A : \{N'_A, B\}K_{AB}$

- *SOPH* (*Secret Out Public Home*) *challenge-response handshake*: The fresh data goes out encrypted and returns in the clear or as subcomponent of a hashed expression. An example of a SOPH challenge-response handshake can be found in the Lee et al. authentication protocol (Lee et al., 2005):

$U_i \rightarrow TTP : (ID_i, C_2) = ID_i, H(ID_i \oplus x) \oplus NU_i$
$TTP \rightarrow U_i : (V_1, C_3) = (H(C_2, N_{Ui}), H(ID_i \oplus x) \oplus N_{TTP})$

- *SOSH* (*Secret Out Secret Home*) *challenge-response handshake*: The fresh data goes out encrypted and returns encrypted. An example of a SOSH can be identified in steps 3 and 4 of the DJCG Lowe's modified version of the Wide-Mouthed Frog protocol (Dojen et al., 2008c):

$B \rightarrow A : \{N_B\}K_{AB}$
$A \rightarrow B : \{succ(N_B)\}K_{AB}$

8.2. *Challenge-response handshake scenarios*

Consider a protocol P where principals A (verifier) and B (prover) are involved in a challenge-response handshake ($A,B \in ENT(P)$). If the cryptographic transformations of the handshake do not include appropriate identities of principals, then an intruder can use the cryptographic transformations in a parallel run to impersonate a legitimate principal. In the following cases mountable attacks are considered for both direct and indirect challenge-response handshakes for symmetric key encryption, while for asymmetric encryption only direct handshakes need to be considered:

Case 1 a). Assume a direct POSH challenge-response handshake using symmetric key K_{AB} shared by principals A and B. The verifier A sends out freshly generated data x_A in the clear (S_o. $A \rightarrow B$: x_A) and receives it back encrypted in a subsequent step (S_p. $B \rightarrow A$:$\{F(x_A),...\}K_{AB}$). If $\neg(C(\{F(x_A),...\}K_{AB},A,S_p) \vee C(\{F(x_A),...\}K_{AB},B,S_p))$ (i.e. cryptographic expression $\{F(x_A), ...\}K_{AB}$ does not contain the identity of either prover or verifier), then an intruder I can impersonate principal B as outlined in Fig. 11.

S^a_o. $A \rightarrow I(B)$: x_A
S^b_o. $I(B) \rightarrow A$: x_A
S^b_p. $A \rightarrow I(B)$: $\{F(x_A), ...\}K_{AB}$
S^a_p. $I(B) \rightarrow A$: $\{F(x_A), ...\}K_{AB}$

Fig. 11 – Parallel session attack in Case 1.a.

In run *a* the intruder intercepts message S^a_o intended for prover B and starts a parallel run *b* of the handshake with A. In run *b* I pretends to be B and uses the same data x_A. Principal A sends the response to the challenge in step S^b_p. As the contained cryptographic expression $\{F(x_A),...\}K_{AB}$ does not contain the identity of either verifier or prover, I can replay this message in step S^a_p of run *a*. Principal A cannot distinguish the replayed S^b_p sent by the intruder from a genuine response S^a_p sent by B. Consequently, A believes it has established a session with B in run *a* and that B has established a session with it in run *b*, even though B is in fact absent – the intruder I has successfully impersonated principal B to A.

Case 1 b). Assume an indirect POSH challenge-response handshake using symmetric keys, where the messages are exchanged indirectly between principals A and B via a TTP. The verifier A sends out a freshly generated data x_A in the clear (S_o. $A \rightarrow B$: x_A). Since B does not possess a shared key with A, the prover B makes a request, which includes data x_A together with A's identity, to a trusted third party TTP in a subsequent step (S_i. $B \rightarrow TTP$: request). TTP extracts data x_A and sends it back to the verifier A, encrypted with a symmetric key shared with A (S_p. $TTP \rightarrow A$:$\{F(x_A),...\}K_{ATTP}$). If $\neg C(\{F(x_A),...\}K_{ATTP},B,S_p)$ (i.e. cryptographic expression $\{F(x_A),...\}K_{ATTP}$ does not contain the identity of prover B), then intruder I can impersonate principal B as outlined in Fig. 12. In run *a* the intruder intercepts message S^a_o intended for prover B and starts a parallel run *b* of the handshake with a principal C (in run *b*, C is the prover). In this second run I pretends to be A and uses the same data x_A. Principal C sends a request to TTP in step S^b_i and TTP responses to challenge x_A in step S^b_p, according with the specification of the exchange. As the contained cryptographic expression $\{F(x_A),...\}K_{ATTP}$ does not contain the identity of prover C, attacker I can replay this message in step S^a_p of the first *a*. Principal A cannot distinguish the replayed S^b_p sent by the intruder from a genuine response S^a_p sent by TTP. Consequently, A believes it has established a session with B in run *a*, even though B is in fact absent.

Case 2 a). Assume a direct SOPH challenge-response handshake using symmetric key K_{AB} shared by principals A and B. Verifier A sends out encrypted freshly generated data x_A (S_o. $A \rightarrow B$: $\{x_A, ...\}K_{AB}$) and receives it in a subsequent step as cleartext or hashed text (S_p. $B \rightarrow A$: $F(x_A)$). If $\neg(C(\{x_A,...\}K_{AB},A,S_o) \vee C(\{x_A,...\}K_{AB},B,S_o))$ (i.e. cryptographic expression $\{x_A, ...\}K_{AB}$ does not contain the identity of either verifier or prover), then an intruder I can impersonate principal B as outlined in Fig. 13. In run *a* the intruder intercepts the message S^a_o intended for prover B and starts parallel run *b* of the handshake with A. In this second run I pretends to be B and uses the same cryptographic expression $\{x_A, ...\}K_{AB}$. As this cryptographic expression does not contain the identity of either sender or recipient, A cannot distinguish it from a genuine request and will respond with step S^b_p, which contains the decrypted data $F(x_A)$. Attacker I intercepts step S^b_p and uses the contained data as correct response in step S^a_p of the run *a*. Consequently, A believes it has established a session with B in run *a* and that B has established a session with it in run *b*,

S^a_o. A → I(B): x_A
S^b_o. I(A) → C: x_A
S^b_i. C → TTP: *request*
S^b_p. TTP → I(A): {F(x_A), ...}K_{ATTP}
S^a_i. omitted
S^a_p. I(TTP) → A: {F(x_A), ...} K_{ATTP}

Fig. 12 – Parallel session attack in Case 1.b.

even though B is in fact absent – the intruder I has successfully impersonated principal B to A.

Case 2 b). Assume an indirect SOPH challenge-response handshake using symmetric keys, where the messages are exchanged indirectly between principals A and B via a TTP. The verifier A sends out freshly generated data x_A encrypted with a symmetric key K_{ATTP}, it shares with TTP (S_o. A → B: {x_A, ...}K_{ATTP}) and receives it in a subsequent step as cleartext or hashed text (S_p. TTP → A: F(x_A)). If ¬C({x_A,...}K_{ATTP},B,S_o) (i.e. cryptographic expression {x_A,...} K_{ATTP} does not contain the identity of prover B), then an intruder I can impersonate verifier A as outlined in Fig. 14. In run *a* the intruder intercepts the message S^a_o intended for prover B and starts parallel run *b* of the handshake with a principal C. In this second run attacker I pretends to be A and uses the same cryptographic expression {x_A,...}K_{ATTP}. As this cryptographic expression does not contain the identity of the prover, TTP cannot distinguish it from a genuine request and will respond with step S^b_p, which contains the decrypted data F(x_A). Attacker I intercepts step S^b_p and uses the contained data as the correct response in step S^a_p of run *a*. Consequently, A believes it has established a session with B in run *a*, even though B is in fact absent.

Case 3 a). Assume a direct SOSH challenge-response handshake using symmetric key K_{AB} shared by principals A and B. Verifier A sends out encrypted freshly generated data x_A (S_o. A → B:{x_A,...}K_{AB}) and receives it back encrypted in a subsequent step (S_p. B → A:{F(x_A),...}K_{AB}). If ¬C({x_A,...}K_{AB},A,S_o)∧¬C({x_A,...}K_{AB},B,S_o))∧ ¬C({F(x_A),...}K_{AB},A,S_p)∧¬C({F(x_A),...}K_{AB},B,S_p) (i.e. both cryptographic expressions {x_A,...}K_{AB} and F(x_A),...}K_{AB} do not contain the identity of either verifier or prover), then an intruder I can impersonate prover B as outlined in Fig. 15. In run *a* the intruder intercepts message S^a_o intended for B and starts a second run *b* of the handshake with A. In this second run attacker I pretends to be B and uses the same cryptographic expression {x_A,...}K_{AB}. As {x_A,...}K_{AB} does not contain any identity of verifier or prover, principal A accepts the message and sends the response to the challenge in step S^b_p. Again, as the contained cryptographic expression {F(x_A),...}K_{AB} does not contain the identity of either verifier or prover, attacker I can replay this message in step S^a_p of run *a*. Principal A cannot distinguish the replayed S^b_p sent by the intruder from a genuine response S^a_p sent by B. Consequently, A believes it

S^a_o. A → I(B): {x_A, ...}K_{AB}
S^b_o. I(B) → A: {x_A, ...} K_{AB}
S^b_p. A → I(B): F(x_A)
S^a_p. I(B) → A: F(x_A)

Fig. 13 – Parallel session attack in Case 2.a.

S^a_o. A → I(B): {x_A, ...}K_{ATTP}
S^b_o. I(A) → C: {x_A, ...} K_{ATTP}
S^b_i. C → TTP: {x_A, ...} K_{ATTP}
S^b_p. TTP → I(A): F(x_A)
S^a_i. omitted
S^a_p. I(TTP) → A: F(x_A)

Fig. 14 – Parallel session attack in Case 2.b.

has established a session with B in run *a* and that B has established a session with it in run *b*, even though B is in fact absent – the intruder I has successfully impersonated principal B to A.

Case 3 b). Assume an indirect SOSH challenge-response handshake using symmetric keys, where the messages are exchanged indirectly between principals A and B via a TTP. The verifier A sends out freshly generated data x_A encrypted with a symmetric key K_{ATTP} shared with TTP (S_o. A → B: {x_A, ...}K_{ATTP}) and receives it encrypted with the key shared with TTP in a subsequent step (S_p.TTP → A:{F(x_A),...}K_{ATTP}). If ¬C({x_A,...}K_{ATTP},B,S_o)∧¬C({F(x_A),...}K_{ATTP},B,S_p) (i.e. both cryptographic expressions {x_A, ...}K_{ATTP} and {F(x_A),...}K_{ATTP} do not contain the identity of prover B), then intruder I can impersonate verifier A as outlined in Fig. 16. In run *a* the intruder intercepts message S^a_o intended for prover B and starts a parallel run *b* of the handshake with a principal C. In run *b* attacker I pretends to be A and uses the same cryptographic expression {x_A, ...}K_{ATTP}. As {x_A, ...}K_{ATTP} does not contain the identity of the prover, TTP cannot distinguish it from a genuine request and sends the response to the challenge in step S^b_p. Again, as the contained cryptographic expression {F(x_A),...}K_{ATTP} does not contain the identity of the prover, attacker I can replay this message in step S^a_p of run *a*. Principal A cannot distinguish the replayed S^b_p sent by the intruder from a genuine response S^a_p sent by TTP. Consequently, A believes it has established a session with B in run *a*, even though B is in fact absent.

Case 4. Assume a direct POSH challenge-response handshake using private keys. Verifier A sends out freshly generated data x_A in clear (S_o. A → B: x_A) and receives it back encrypted in a subsequent step (S_p. B → A: {F(x_A),...}K_{BPriv}). If ¬C({F(x_A),...}K_{BPriv},A,S_p) (i.e. cryptographic expression {F(x_A),...}K_{BPriv} does not contain the identity of verifier), then an intruder I can impersonate prover B as outlined in Fig. 17. In run *a* the intruder intercepts message S^a_o intended for B and starts a parallel run *b* of the handshake with principal B. In run *b* attacker I acts as a legitimate principal and uses the same data x_A. Principal B accepts this message and responds with step S^b_p, which contains {F(x_A),...}K_{BPriv}. As {F(x_A),...}K_{BPriv} is free of any data containing identity of verifier A, the

S^a_o. A → I(B): {x_A, ...}K_{AB}
S^b_o. I(B) → A: {x_A, ...}K_{AB}
S^b_p. A → I(B): {F(x_A), ...}K_{AB}
S^a_p. I(B) → A: {F(x_A), ...}K_{AB}

Fig. 15 – Parallel session attack in Case 3.a.

S^a_o. A → I(B): $\{x_A, ...\}K_{ATTP}$
S^b_o. I(A) → C: $\{x_A, ...\}K_{ATTP}$
S^b_i. C → TTP: $\{x_A, ...\}K_{ATTP}$
S^b_p. TTP → I(A): $\{F(x_A),...\}K_{ATTP}$
S^a_i. omitted
S^a_p. I(TTP) → A: $\{F(x_A),...\}K_{ATTP}$

Fig. 16 – Parallel session attack in Case 3.b.

attacker can replay it as a valid response message S^a_p in run *a*. Consequently, A believes it has established a session with B in run *a* even though B does not participate in run *a*, thus the intruder I has successfully impersonated principal B to A.

Case 5. Assume a direct SOSH challenge-response handshake using private keys. Verifier A sends out freshly generated encrypted data x_A (S_o. A → B:$\{x_A,...\}K_{APriv}$) and receives it back encrypted in a subsequent step (S_p. B → A:$\{F(x_A), ...\}K_{BPriv}$).

a)If $\neg C(\{F(x_A),...\}K_{BPriv},A,S_p)$ then intruder I can impersonate prover B as outlined in Fig. 18. In run *a* the intruder intercepts message S^a_o intended for principal B and starts a parallel run *b* of the handshake with B. In run *b* attacker I acts as a legitimate principal and uses the same data x_A encrypted with his private key K_{IPriv}. Principal B accepts this message and responds with step S^b_p, which contains $\{F(x_A), ...\}K_{BPriv}$. As $\{F(x_A), ...\}K_{BPriv}$ is free of any data containing identity of verifier A, the attacker can replay it as a valid response message S^a_p in run *a*. Consequently, A believes it has established a session with B in run *a* even though B does not participate in run *a*, thus the intruder I has successfully impersonated principal B to A.

b) If $\neg C(\{x_A,...\}K_{APriv},B,S_o)$, then intruder I can impersonate verifier A as outlined in Fig. 19. In run *a* the intruder intercepts message S^a_o intended for prover B and starts a parallel run *b* of the handshake with a principal C. In run *b* attacker I pretends to be A and uses the same cryptographic expression $\{x_A,...\}K_{APriv}$. As this cryptographic expression does not contain the prover's identity, C cannot distinguish the replayed S^a_o sent by the intruder, from a genuine S^b_o sent by A. Principal C responds with S^b_p and believes it has established a session with A in run *b*, even though A does not participate in run *b*, thus the intruder I has successfully impersonated principal A to C in *b*.

Case 6. Assume a direct SOPH challenge-response handshake using public keys. Verifier A sends out freshly generated encrypted data x_A (S_o. A → B:$\{x_A,...\}K_{BPub}$) and receives it back in clear or hashed in a subsequent step (S_p. B → A: $F(x_A)$). If $\neg C(\{x_A,...\}K_{BPub},A,S_o)$, then intruder I can impersonate prover B as outlined in Fig. 20. In run *a*, the intruder intercepts message S^a_o intended for B and starts a parallel run *b* of the

S^a_o. A → I(B): x_A
S^b_o. I → B: x_A
S^b_p. B → I: $\{F(x_A), ...\}K_{BPriv}$
S^a_p. I(B) → A: $\{F(x_A), ...\}K_{BPriv}$

Fig. 17 – Parallel session attack in Case 4.

S^a_o. A → I(B): $\{x_A, ...\}K_{APriv}$
S^b_o. I→ B: $\{x_A, ...\}K_{IPriv}$
S^b_p. B → I: $\{F(x_A), ...\}K_{BPriv}$
S^a_p. I(B) → A: $\{F(x_A), ...\}K_{BPriv}$

Fig. 18 – Parallel session attack in Case 5a.

S^a_o. A → I(B): $\{x_A, ...\}K_{APriv}$
S^b_o. I(A)→ C: $\{x_A, ...\}K_{APriv}$
S^b_p. C → I(A): $\{F(x_A), ...\}K_{CPriv}$

Fig. 19 – Parallel session attack in Case 5b

handshake with prover B. In run *b* attacker I acts as a legitimate verifier and uses the same cryptographic expression $\{x_A,...\}K_{BPub}$. As this cryptographic expression does not contain the verifier's identity, B accepts it as a genuine message from I and responds with step S^b_p. Attacker I uses the contained data as the correct response in step S^a_p of run *a*. Consequently, A believes it has established a session with B in run *a*, even though B does not participate in run *a*, thus the intruder I has successfully impersonated principal B to A.

Case 7. Assume a direct SOSH challenge-response handshake using public keys. Verifier A sends out freshly generated encrypted data x_A (S_o. A → B:$\{x_A, ...\}K_{BPub}$) and receives it back encrypted in a subsequent step (S_p. B → A: $\{F(x_A),...\}K_{APub}$). If$\neg C(\{x_A,...\}K_{BPub},A,S_o)\wedge\neg C(\{F(x_A),...\}K_{APub},B,S_p)$, then an intruder I can impersonate prover B as outlined in Fig. 21. In run *a*, the intruder intercepts message S^a_o intended for B and starts a parallel run *b* of the handshake with prover B. In run *b* attacker I acts as a legitimate verifier, but uses the same cryptographic expression $\{x_A,...\}K_{BPub}$. As this cryptographic expression does not contain the verifier's identity, prover B accepts it as a genuine message from I and responds with step S^b_p, which contains $F(x_A)$ encrypted with I's public key. Attacker I extracts $F(x_A)$ and creates cryptographic expression $\{F(x_A),...\}K_{APub}$ as a suitable message for S^a_p. Consequently, A believes it has established a session with B in run *a*, even though B does not participate in the exchange of run *a*, thus - attacker I has successfully impersonated principal B to A.

Flaws corresponding to all cases in this section have been revealed on security protocols such as BAN concrete Andrew Secure RPC protocol (Lowe, 1996), Lowe's modified version of the Wide-Mouthed Frog protocol (Dojen et al., 2008c), KSL protocol (Lowe, 1996), Neumann Stubblebine protocol (Hwang et al., 1995), SPLICE protocol (Hwang and Chen, 1995; Clark and Jacob, 1995), Needham-Shroeder public key protocol (Lowe, Nov 1995) and CCITT X.509(3) protocol (Burrows et al., 1990), BAN modified Andrew Secure RPC (Burrows et al., 1990), Andrew Secure RPC (Burrows et al., 1990).

S^a_o. A → I(B): $\{x_A, ...\}K_{BPub}$
S^b_o. I → B: $\{x_A, ...\}K_{BPub}$
S^b_p. B → I: $F(x_A)$
S^a_p. I(B) → A: $F(x_A)$

Fig. 20 – Parallel session attack in Case 6.

S^a_o. A → I(B): $\{x_A, \ldots\}K_{BPub}$
S^b_o. I → B: $\{x_A, \ldots\}K_{BPub}$
S^b_p. B → I: $\{F(x_A), \ldots\}K_{IPub}$
S^a_p. I(B) → A: $\{F(x_A), \ldots\}K_{APub}$

Fig. 21 – Parallel session attack in Case 7.

8.3. *Proposed guidelines for challenge-response handshakes construction*

As was previously shown in this section, three-way and four-way challenge response handshakes can be disunited into multiple two-way handshakes. Therefore, the proposed design guidelines for two-way challenge-response handshakes are also applicable to three-way and four–way challenge-response handshakes. In order to prevent challenge-response handshakes construction related issues exploitable by parallel session attacks, the following guidelines should be obeyed:

(GL4.1.D) At least one of the cryptographic expressions, belonging to a direct POSH challenge-response handshake using symmetric encryption should contain the identity of the verifier or prover.

$$\forall y \in CRH(P,A,B) : \exists \{x\}K_{AB}, S_p | C(y, \{x\}K_{AB}, S_p) : (C(\{x\}K_{AB}, A, S_p) \vee C(\{x\}K_{AB}, B, S_p))$$

(GL4.1.I) At least one of the cryptographic expressions, belonging to an indirect POSH challenge-response handshake using symmetric encryption should contain the prover's identity (B).

$$\forall y \in CRH(P,A,B) : \exists \{x\}K_{ATTP}, S_p | C(y, \{x\}K_{ATTP}, S_p) : C(\{x\}K_{ATTP}, B, S_p)$$

(GL4.2.D) At least one of the cryptographic expressions, belonging to a direct SOPH challenge-response handshake using symmetric encryption should contain the identity of the verifier or prover.

$$\forall y \in CRH(P,A,B) : \exists \{x\}K_{AB}, S_o | C(y, \{x\}K_{AB}, S_o) : (C(\{x\}K_{AB}, A, S_o) \vee C(\{x\}K_{AB}, B, S_o))$$

(GL4.2.I) At least one of the cryptographic expressions, belonging to an indirect SOPH challenge-response handshake, using symmetric encryption, should contain the prover's identity (B).

$$\forall y \in CRH(P,A,B) : \exists \{x\}K_{ATTP}, S_o | C(y, \{x\}K_{ATTP}, S_o) : C(\{x\}K_{ATTP}, B, S_o)$$

(GL4.3.D) At least one of the cryptographic expressions, belonging to a direct SOSH challenge-response handshake using symmetric encryption, should contain the identity of the verifier or prover.

$$\forall y \in CRH(P,A,B) : \exists \{x\}K_{AB}, S_r, r \in \{o,p\} | C(y, \{x\}K_{AB}, S_r) : (C(\{x\}K_{AB}, A, S_r) \vee C(\{x\}K_{AB}, B, S_r))$$

(GL4.3.I) At least one of the cryptographic expressions (emitted by the verifier or by the prover), belonging to an indirect SOSH challenge-response handshake using symmetric encryption should contain the prover's identity (B).

$$\forall y \in CRH(P,A,B) : \exists \{x\}K_{ATTP}, S_r, r \in \{o,p\} | C(y, \{x\}K_{ATTP}, S_r) : C(\{x\}K_{ATTP}, B, S_r)$$

(GL4.4) At least one of the cryptographic expressions, belonging to a POSH challenge-response handshake using encryption with private keys should contain the verifier's (recipient's) identity.

$$\forall y \in CRH(P,A,B) : \exists \{x\}K_{BPriv}, S_p | C(y, \{x\}K_{BPriv}, S_p) : C(\{x\}K_{BPriv}, A, S_p)$$

(GL4.5) All the cryptographic expressions, belonging to a SOSH challenge-response handshake using encryption with private keys should contain the recipient's identity (i.e. the cryptographic expressions emitted by the verifier should contain the prover's identity and the cryptographic expressions emitted by the prover should contain the verifier's identity).

$$\forall y \in CRH(P,A,B) : (\forall \{x\}K_{APriv}, S_o | C(y, \{x\}K_{APriv}, S_o) : C(\{x\}K_{APriv}, B, S_o)) \wedge (\forall \{z\}K_{BPriv}, S_p | C(y, \{z\}K_{BPriv}, S_p) : C(\{z\}K_{BPriv}, A, S_p))$$

(GL4.6) At least one of the cryptographic expressions, belonging to a SOPH challenge-response handshake using encryption with public keys should contain the verifier's (sender's) identity.

$$\forall y \in CRH(P,A,B) : \exists \{x\}K_{BPub}, S_o | C(y, K_{BPub}, S_o) : C(\{x\}K_{BPub}, A, S_o)$$

(GL4.7) At least one of the cryptographic expressions, belonging to a SOSH challenge-response handshake using encryption with public keys should contain the sender's identity (i.e. the cryptographic expressions emitted by the verifier should contain the verifier's identity or the cryptographic expressions emitted by the prover should contain the prover's identity).

$$\forall y \in CRH(P,A,B) : (\exists \{x\}K_{BPub}, S_o | C(y, \{x\}K_{BPub}, S_o) : C(\{x\}K_{BPub}, A, S_o)) \vee (\exists \{z\}K_{APub}, S_p | C(y, \{z\}K_{APub}, S_p) : C(\{z\}K_{APub}, B, S_p))$$

9. Evaluation of proposed guidelines

This study seeks to evaluate the effectiveness of the proposed design guideline by evaluating whether security protocols observe the guidelines. This evaluation considers a wide range of protocols, incorporating those with known weaknesses and their published amended versions. The guidelines are considered effective if:

- Protocols with known weaknesses violate some of the guidelines.

- Protocols without weaknesses do not violate any guidelines.

In the remainder of this study the conformance of protocols to the proposed guidelines is established by examining if the message exchanges are in accordance with the four defined categories, namely: (i) freshness of messages, (ii) symmetry of messages, (iii) signed messages and (iv) challenge-response handshake construction. The conformance of the LKY protocol (and its amended version) is analysed in detail and the conformance of a range of protocols is presented in summary form.

9.1. *Evaluating conformance of LKY protocol*

In this section the proposed design guidelines are applied to the mutual authentication protocol of Lee, Kim and Yoo (LKY) (Lee et al., 2005) and its amended version (Jurcut et al., November 2013). We show which of the proposed guidelines are not adhered to by the protocol, we describe both replay and parallel session attacks on the protocol, and we show that the amended version of the protocol (Jurcut et al., November 2013) conforms to the design guidelines.

9.1.1. *LKY protocol*

LKY protocol (Lee et al., 2005) is a nonce-based mutual authentication scheme using smart cards. The protocol employs the same authentication structure for both the remote user and the system. It 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, referred throughout this paper as the authentication session, are carried out whenever a user wants to gain access to the server. Fig. 22 outlines the authentication session of the scheme.

Assume k denotes the only secret key maintained by the system TTP and ID_i is an identifier of the remote user U_i. In the registration phase, U_i registers his identifier ID_i and password PW_i to the system in a secure channel. The system computes $R_i = H(ID_i \oplus k) \oplus PW_i$, where $\oplus$ denotes the bit-wise exclusive-OR operator, stores the hash-algorithm H() and the value R_i into the memory of a smart card, and issues the smart card to U_i. In the login phase, U_i 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 $C_2 = C_1 \oplus N_{Ui}$, where N_{Ui} is a random nonce generated by user U_i
- Send the message (ID_i, C_2) to the system.

In the verification phase, the system and the smart card execute the following operations to achieve mutual authentication:

1) The system checks the validity of ID_i and then computes:

$$C_1 = H(ID_i \oplus k),$$

$$N_{Ui} = C_2 \oplus C_1,$$

1. $U_i \rightarrow TTP$: $(ID_i, C_2) = ID_i, H(ID_i \oplus k) \oplus N_{Ui}$
2. $TTP \rightarrow U_i$: $(V_1, C_3) = H(C_2, N_{Ui}), H(ID_i \oplus k) \oplus N_{TTP}$
3. $U_i \rightarrow TTP$: $(V_2) = H(C_3, N_{TTP})$

Fig. 22 – Authentication session of LKY scheme.

$$V_1 = H(C_2, N_{Ui}), \text{ and}$$

$$C_3 = C_1 \oplus NTTP,$$

where N_{TTP} is a random nonce generated by the system TTP.

2) The system sends the message (V_1, C_3) to U_i.
3) Upon receiving the message (V_1, C_3), U_i verifies whether $V_1 = H(C_2, N_{Ui})$. If equal, U_i believes that the system is authenticated. Then the smart card computes:

$$N_{TTP} = C_3 \oplus C_1 \text{ and}$$

$$V_2 = H(C_3, NTTP).$$

4) The smart card sends the message (V_2) to the system.
5) The system verifies whether $V_2 = H(C_3, N_{TTP})$. If equal, the system believes that U_i is authenticated.

9.1.2. *Evaluating guidelines conformance*

Analysing the message exchanges of the LKY protocol, to establish if they are in accordance with all the defined design guidelines, reveals that the protocol (Fig. 22) does not observe the guidelines for freshness of messages (GL1.1–GL1.3). Additionally, the LKY protocol does not observe the proposed guidelines for symmetry of messages (GL2). It is thus vulnerable to replay and parallel session attacks. Consequently, the LKY protocol fails to provide a sound mutual authentication service, as an attacker without knowing any secret information can impersonate a remote user U_i to cheat the system. Fixing the protocol to eliminate these design weaknesses is outlined below.

9.1.2.1. a. Replay attack on LKY protocol. On verifying the conformance of the message exchanges with the design guidelines for freshness it can be seen that the second cryptographic message of the protocol $C_3 = H(ID_i \oplus k) \oplus N_{TTP}$ transmitted in response step 2 does not contain any fresh component for the recipient U_i. Therefore U_i has no assurance that the nonce N_{TTP} is fresh and is not replayed from a previous run of the protocol. Hence, a replay attack can be mounted, where an attacker can substitute a previously recorded message 2 from TTP to U_i and mislead user U_i into accepting an old and possibly compromised nonce (N'_{TTP}) As a consequence the attacker can impersonate a target remote user U_i.

The replay attack can be carried out as depicted in Fig. 23. The attack assumes that a remote user U_i initiates the protocol with the system, by sending message S_1^a (ID_i, C_2), consisting of its identity concatenated with the component $H(ID_i \oplus k) \oplus N_{Ui}$ to the server TTP. An attacker I intercepts message S_1^a intended for TTP and initiates a new session (denoted b) with the server TTP, by sending message S_1^b, where N'_I is a random number generated by attacker I. After receiving the messages

S_1^a and S_1^b, the server TTP generates and sends messages S_2^a and S_2^b according to the specification of the exchange. After the attacker intercepts and blocks the messages S_2^a and S_2^b, it sends the fabricated message $S_2^{a'}$ to U_i, according to the specification exchange of step 2. The attacker computes $S_2^{a'}$, by replaying first component of the message S_2^a, (V_1) together with the second component of message S_2^b (C_3'). Upon receiving the message $S_2^{a'}$ in run a, user U_i computes and sends the message S_3^a, according to the specification of the exchange. Note that U_i computes the message S_3^a (i.e. the response of the nonce challenge emitted as the second component of message $S_2^{a'}$), because it successfully verified the message $V_1 = H(C_2, N_{Ui})$, which is the corresponding response to the challenge sent as part of message $S^a{}_1$in run a. The attacker can replay S_3^a (V_2^i) as message S_3^b in parallel run b, to finish its session and pass the system's authentication.

As a result, although U_i authenticates the system in the first session a, after the attacker's session b, the system TTP mistakenly believes that the attacker is an 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 U_i to cheat the system.

9.1.2.2. b. Parallel session attack on LKY protocol. Analysing the message exchanges to establish if they are in accordance with the defined guidelines for symmetry of messages reveals that two pairs ((C_2,C_3) and (V_1,V_2)) of cryptographic messages exchanges between the user and server are symmetric and principal value type equivalent.

(1) $C_2 = H(ID_i \oplus k) \oplus N_{Ui}$ (step 1) and $C_3 = H(ID_i \oplus k) \oplus N_{TTP}$ (step 2)
(2) $V_1 = H(C_2,N_{Ui})$ (step 2) and $V_2 = H(C_3,N_{TTP})$ (step 3).

Thus guideline (GL2) is violated. As a consequence, a parallel session attack can be mounted against the LKY protocol, where an attacker can forge login messages to impersonate a legitimate user.

Mounting a parallel session attack on the LKY protocol is detailed in Fig. 24, where it is assumed that attacker I wants to gain access to the server masquerading as a legitimate user U_i.

The attacker, impersonating user U_i, launches the attack by choosing a random number N_I and sending message S_1^a (ID_i,C_2') as a login request message to the server TTP. Note that from the server's point of view, N_I is indistinguishable from nonce N_{Ui} of an honest execution, since both are random numbers. The server TTP sends the message $S^a{}_2$ to the attacker masquerading as U_i, according to the specification of the exchange. After receiving message in step 2, the attacker starts a parallel session b, posing again as user U_i and replaying the challenge C_3 sent by the server in the original session S_2^a. The server TTP cannot distinguish the replayed response C_3 sent by

$S^a{}_1$. $U_I \rightarrow TTP$: $(ID_I,C_2) = ID_I, H(ID_i \oplus k) \oplus N_{Ui}$
$S^b{}_1$. $I(U_i) \rightarrow TTP$: $(ID_I,C'_2) = ID_I, H(ID_i \oplus k) \oplus N'_I$
$S^a{}_2$. $TTP \rightarrow I(Ui)$: $(V1,C_3) = H(C_2,N_{Ui}), H(ID_i \oplus k) \oplus N_{TTP}$
$S^b{}_2$. $TTP \rightarrow I(U_i)$: $(V'_1,C'_3) = H(C'_2,N'_I), H(ID_i \oplus k) \oplus N'_{TTP}$
$S^a{}_2'$. $I(TTP) \rightarrow U_i$: $(V_1,C'_3) = H(C_2,N_{Ui}), H(ID_i \oplus k) \oplus N'_{TTP}$
$S^a{}_3$. $U_I \rightarrow I(TTP)$: $(V'_2) = H(C'_3, N'_{TTP})$
$S^b{}_3$. $I(U_i) \rightarrow TTP$: $(V'_2) = H(C'_3, N'_{TTP})$

Fig. 23 – Replay Attack on LKY Protocol.

the attacker from a genuine message S_1^b sent by a honest user U_i. Hence, TTP computes V_1 and sends the message S_2^b (V_1, C_3') in response to S_1^b, as specified in the protocol. The component V_1 sent as part of message S_2^b in the parallel session, is exactly what the attacker needs in order to gain access permission in the original session a. 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 U_i to cheat the system.

9.1.3. Conformance of the amended LKY protocol

As shown in the previous sections the LKY scheme (Lee et al., 2005) does not follow the proposed set of guidelines. We now demonstrate that the amended protocol (Jurcut et al., November 2013) is in accordance with the design guidelines, making it resistant to replay and parallel session attacks.

Following the proposed design guideline for freshness of messages (guideline (GL1.1)), the cryptographic message $C_3 = H(ID_i \oplus k) \oplus N_{TTP}$ (step 2) should include a component which the recipient of step 2, recognizes as fresh. This can be achieved by including nonce N_{Ui}, previously generated by user U_i in step 1, in the content of C_3, as presented in Fig. 25. Thus, the cryptographic message that contains the nonce N_{TTP} generated by the server can be identified by user U_i as fresh, i.e. as belonging to the current protocol run. Consequently, any attempt by an attacker to replay the message C_3' of step 2 will fail, as U_i can identify the replay through the incorrect value of N_{Ui}.

In order to prevent a potential parallel session attack the symmetrical structure of the hashed expressions $V_1 = H(C_2,N_{Ui})$ and $V_2 = H(C_3,N_{TTP})$, transmitted in steps 2 and 3 and the symmetrical structure of the cryptographic messages $C_2 = H(ID_i \oplus k) \oplus N_{Ui}$ and $C_3 = H(ID_i \oplus k) \oplus N_{TTP}$ exchanged in steps 1 and 2 of the protocol should be broken. Following the design guideline for symmetry of messages (guideline (GL2)) and the proposed solutions for breaking the symmetry of two cryptographic and/or hashed expressions, the identity of user U_i is added to the cryptographic message V_2 in step 3. Further, the solution proposed for freshness of messages also breaks the symmetrical structure of cryptographic messages C_2 and C_3. As the cryptographic messages $V_1 = H(C_2,N_{Ui})$ and $V_2' = H(C_3',N_2,U_i)$ and respectively, $C_2 = H(ID_i \oplus k) \oplus N_{Ui}$ and $C_3' = H(ID_i \oplus k) \oplus N_{TTP} \oplus N_{Ui}$ are now no longer symmetric, the attack depicted in Fig. 24 is prevented

9.2. Results of evaluation study

In this study the conformance of a range of protocols is established by examining if the message exchanges are in accordance with the four defined categories of the proposed guidelines, namely: (i) freshness of messages, (ii) symmetry of messages, (iii) signed messages and (iv) challenge-response handshake construction. These protocols include those with known weaknesses and those that are known to be secure.

$S^a{}_1$. $I(U_i) \rightarrow TTP$: $(ID_i,C'_2) = ID_I, H(ID_i \oplus k) \oplus N_I$
$S^a{}_2$. $TTP \rightarrow I(U_i)$: $(V'_1,C_3) = H(C'_2,N_I), H(ID_i \oplus k) \oplus N_{TTP}$
$S^b{}_1$. $I(U_i) \rightarrow TTP$: $(ID_I, C_3) = ID_I, H(ID_i \oplus k) \oplus N_{TTP}$
$S^b{}_2$. $TTP \rightarrow I(U_i)$: $(V_1,C'_3) = H(C_2,N_{TTP}), H(ID_i \oplus k) \oplus N'_{TTP}$
$S^a{}_3$. $I(U_i) \rightarrow TTP$: $(V_1) = H(C_2, N_{TTP})$
$S^b{}_3$. dropped

Fig. 24 – A parallel session attack on LKY protocol.

1. $U_i \rightarrow TTP$: $(ID_i, C_2) = ID_i, H(ID_i \oplus k) \oplus N_{Ui}$
2. $TTP \rightarrow U_i$: $(V_1, C'_3) = H(C_2, N_{Ui}), H(ID_i \oplus k) \oplus N_{TTP} \oplus N_{Ui}$
3. $U_i \rightarrow TTP$: $(V'_2) = H(C'_3, N_{TTP}, U_i)$

Fig. 25 – Proposed amended version for LKY protocol.

The results of this conformance evaluation are summarised in Table 1. The second column of this table enumerates previously published replay (R) or parallel session (P) attacks on the analysed protocols and the year when the attack was published, while the third column indicates the violated design guidelines and related attacks.

This study shows that for all the protocols evaluated those with known replay or parallel session attacks violate at least one of the proposed guidelines. The set of guidelines is able to detect all design weaknesses exploitable by the published replay and parallel session attacks in the chosen set of security protocols. Also, none of the proposed guidelines are violated for protocols which are known to be secure against replay or parallel session attacks. Thus, the proposed design guidelines can be considered effective.

10. Conclusion

This research work investigated why replay and parallel session attacks can be successfully deployed against security protocols thus compromising their security. The analysis

Table 1 – Empirical results of proposed design guidelines.

Analysed Protocol	Published Attacks	Violated guidelines
1. NS PK, 1978 (Needham and Schroeder, 1978)	R, 1981(Denning and Sacco, 1981); P, 1995(Lowe, Nov 1995)	GL1.1(R); GL1.2(R); GL4.7(P)
2. Lowe's fix NS PK, 1995 (Lowe, Nov 1995)	R, 1981 (Denning and Sacco, 1981)	GL1.1(R); GL1.2(R)
3. AS RPC, 1989 (Satyanarayanan, 1989)	R,1990(Burrows et al., 1990); P,1996(Lowe, 1996)	GL1.2(R); GL4.3.D(P)
4. BAN mod AS RPC, 1990(Burrows et al., 1990)	P, 1996(Lowe, 1996)	GL4.3.D(P)
5. BAN concrete AS RPC, 1990(Burrows et al., 1990)	P, 1996(Lowe, 1996)	GL4.1.D(P)
6. Lowe AS RPC, 1996(Lowe, 1996)	No attack	None
7.CCITT X.509(3), 1987 (CCITT, 1987)	P, 1990(Burrows et al., 1990)	GL3.1(P); GL4.5(P)
8.BAN CCITT X.509(3), 1990(Burrows et al., 1990)	No attack	None
9.BAN simplif. Yahalom, 1990(Burrows et al., 1990)	P, 1994(Syverson, June 1994)	GL2(P)
10.Paulson's Yahalom, 2001(Paulson, 2001)	No attack	None
11. Neumann Stubb., 1993 (Neumann & Stubblebine, April 1993)	P, 1995 (Hwang et al., 1995)	GL2(P); GL4.1.D(P)
12.SPLICE/AS, 1991(Yamaguchi et al., November 1991)	R, 1995 (Clark and Jacob, 1995); P,1995(Hwang and Chen, 1995)	GL1.3(R); GL3.4(P)
13.HC mod SPLICE/AS, 1995(Hwang and Chen, 1995)	R, 1995 (Clark and Jacob, 1995)	GL1.3(R)
14. Kao Chow Auth.v.1, 1995 (Kao and Chow, 1995)	R, 1995(Kao and Chow, 1995); 2008(Dojen et al., 2008a)	GL1.3(R); GL2(P)
15. DS PK, 1981 (Denning and Sacco, 1981)	P, 1996(Abadi and Needham, 1996)	GL3.2(P)
16. AN fix DS PK, 1996 (Abadi and Needham, 1996)	No attack	None
17. KSL, 1992 (Kehne et al., 1992)	P, 1996(Lowe, 1996)	GL2(P); GL4.1.D(P)
18. AKKA_v3, 1996 (Safford et al., July 1996)	P, 1997(Abadi, March 1997)	GL 1.2(R); GL 3.2(P)
19.WLM Auth., 1994 (Woo and Lam, 1994)	P, 1997 (Clark & Jacob, November 1997)	GL2(P)
20. W-M Frog, 1990 (Burrows et al., 1990)	P, 1997 (Lowe, 1997)	GL2(P)
21.SSH PK, 1996 (Ylönen, June 1996)	P, 1997 (Abadi, March 1997)	GL3.3 (P)
22. Abadi fix SSH PK, 1997 (Abadi, March 1997)	No attack	None
23. TMN, 1989 (Tatebayashi et al., 1989)	R, 1997 (Lowe and Roscoe, 1997)	GL1.1(R); GL1.2(R); GL1.3(R)
24. V. 2 of TMN, 1997 (Lowe and Roscoe, 1997)	R, 1997(Lowe and Roscoe, 1997)	GL1.1(R); GL1.2(R); GL1.3 (R)
25. V. 3 of TMN, 1997 (Lowe and Roscoe, 1997)	R, 1997(Lowe and Roscoe, 1997)	GL1.1(R); GL1.2(R); GL1.3 (R)
26. PKM Auth. IEEE 802.16, 2004 (Johnston and Walker, 2004)	R, 2006(Xu & Huang, September, 2006); R, 2006 (Xu & Huang, September, 2006)	GL1.1(R); GL1.2(R)
27. PKMv2 IEEE 802.16, 2005(IEEE Std. 802.16e/D12, 2005)	P, 2006 (Xu & Huang, September, 2006)	GL4.1.D (P)
28. Lowe W-M Frog, 1997 (Lowe, 1997)	P, 2008 (Dojen et al., 2008c)	GL2(P); GL4.3.D(P)
29. KZ Auth., 2006 (Khan and Zhang, 2006)	P, 2008 (Xu et al., July, 2008)	GL2 (P)
30.CBKM - IC, 2008 (Shen et al., 2008)	R, 2008(Dojen et al., 2008b)	GL1.1(R); GL1.2(R); GL2(P)
31.DZC fix CBKM-IC, 2008 (Dojen et al., 2008b)	No attack	None
32.CBKM-RC, 2008 (Shen et al., 2008)	R, 2008(Dojen et al., 2008b)	GL1.1(R); GL1.2(R)
33. YRY Auth., 2004 (Yoon et al., 2004)	P, 2009 (Hsiang and Shih, 2009)	GL2 (P)
34. HCH Auth.& KA, 2012 (He et al., 2012)	P, 2013 (Wang and Ma, 2013)	GL2 (P)
35. Jin et al.RFID Auth., 2011 (Jin et al., 2011)	R, 2013 (Fu and Guo, 2013)	GL1.1 (R)
36. LKY Auth., 2005 (Lee et al., 2005)	R, 2013(Jurcut et al., November 2013); P, 2007(Nam et al., Jan 2007)	GL1.1(R); GL2 (P)
37. NKPW mod. LKY, 2007 (Nam et al., Jan 2007)	R, 2013(Jurcut et al., November 2013)	GL1.1(R)
38. JDC mod. LKY, 2013 (Jurcut et al., November 2013)	No attack	None

discovered the vulnerabilities in the structure of the protocol message exchanges that can be exploited by these attacks and also characterised the general circumstances under which these attacks can be mounted. Based on this analysis a new set of design guidelines that ensure resistance to replay and parallel session attacks was proposed. The proposed set incorporates guidelines for each message exchange pattern related to freshness of messages, symmetry of messages, signed messages and challenge-response handshake construction. These guidelines are general purpose so as to encompass a wide-variety of protocols.

An empirical study on evaluating the effectiveness of the proposed design guidelines was undertaken to establish if a wide-range of protocols with known weaknesses and their corrected versions observe the guidelines. The evaluation of the conformance of one protocol - a nonce-based mutual authentication scheme using smart cards – was presented in detail, showing how the protocol did not conform to the guidelines. Further, a mountable replay attack and a parallel session attack on this protocol were presented, as well as an amended version of the protocol. The amended protocol conforms to the design guidelines making it resistant to replay and parallel session attacks. The results of the empirical evaluation of a wide-range of protocols, incorporating those with known weaknesses and their published amended versions, showed that the proposed design guidelines are effective as:

- the protocols with known weaknesses violated some of the guidelines,
- the protocols without weaknesses did not violate any of the guidelines.

Finally, as future work, it is anticipated that the set of design guidelines presented in this paper will be formalised and incorporated into a logic theory for attack detection in security protocols, and implemented into an automated security protocol verification tool.

Acknowledgement

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

Appendix A. Design guidelines for preventing replay and parallel session attacks

(GL1.1)	In the absence of synchronized clocks, each cryptographic expression that is part of a response step in a protocol should contain a fresh component generated by the recipient in a previous step of the protocol. $\forall \{x\}k : \{x\}k \in m(S_p) \wedge S_p \in RS(P) \exists x_B \mid C(\{x\}k, x_B, S_p) \wedge \exists o : o < p \wedge Gen(B, x_B, S_o) \wedge B = r(S_p)$
(GL1.2)	If timestamps are used (with the assumption of synchronized clocks), then each cryptographic expression of a protocol should contain a timestamp. $\forall \{x\}k \in m(P) \mid \exists t \in TS(P) \wedge C(\{x\}k,t,S_r)$
(GL1.3)	A component w used in the generation of a key should be sent at least once as part of a cryptographic expression containing at least one component which the recipient recognizes as being fresh. $\forall KMaterial(w) \exists \{x\}k \in m(S_r) \mid C(\{x\}k, (x_B, w), S_r) \wedge B = r(S_r) \wedge Fresh(B, x_B)$
(GL2)	If two principals exchange a pair of cryptographic transformations c1 and c2 directly or indirectly via a TTP, then c1 and c2 should not be symmetric and at the same time not principal value type equivalent. $\forall c1,c2 \in CT(P) \mid \uparrow \downarrow (c1,c2) \wedge (\neg Symmetric(c1,c2) \vee \neg Pvte(c1,c2))$
(GL3.1)	If a signed message $\{x\}K_{APriv}$ is a parent cryptographic expression, then $\{x\}K_{APriv}$ should be receiver bound. $\exists S_r \in P : s(S_r) = A \wedge r(S_r) = B \wedge \forall \{x\}K_{APriv} \in m(S_r) \mid \Pi(\{x\}K_{APriv}), RBound(B, \{x\}K_{APriv})$
(GL3.2)	If a signed message {x}KaPriv of a step Sr is contained by a parent cryptographic expression {y}KbPub encrypted with the public key of recipient B of Sr, then component y should contain at least one receiver bound cryptographic transformation. $\exists S_r \in P : s(S_r) = A \wedge r(S_r) = B \wedge \forall \{x\}K_{APriv} \in m(S_r), \exists \{y\}K_{BPub} \in m(S_r) \mid C(\{y\}_{KBPub}, \{x\}K_{APriv}, S_r) \wedge \Pi(\{y\}_{KBPub}), (\forall c \mid C(y, c, S_r) \wedge RBound(B, c))$
(GL3.3)	If a signed message $\{x\}K_{APriv}$ of a step S_r of protocol P is contained in a parent cryptographic expression $\{y\}K_{AB}$ that fulfils the following conditions: • the symmetric key K_{AB} is generated by a principal C belonging to {TTP, A} and is transmitted by C in a cryptographic expression $\{z\}K_{BPub}$ encrypted with B's public key in step S_q prior to S_r • $\{z\}K_{BPub}$ in step S_q is free of any receiver bound messages signed by C then component y should contain at least one receiver bound cryptographic transformation. $\exists S_r \in P : s(S_r) = A \wedge r(S_r) = B \wedge \forall \{x\}K_{APriv} \in m(S_r), \exists \{y\}K_{AB} \in m(S_r) \mid C(\{y\}K_{AB}, \{x\}K_{APriv}, S_r) \wedge \Pi(\{y\}K_{AB}) \wedge (\exists q < r \mid Gen(C, \{z\}K_{BPub}, S_q) \wedge C(z, K_{AB}, S_q) \wedge (\forall \{w\}K_{CPriv} \mid C(z, \{w\}K_{CPriv}, S_q) \wedge \neg RBound(B, \{w\}K_{CPriv}))), (\forall c \mid C(y, c, S_r) \wedge RBound(B, c))$

(GL3.4)	Any signed statement intended for public key distribution should include the identity of the public key owner. $\exists S_r \in P: \forall \{K_{BPub}, x\} K_{TTPPriv} \in m(S_r) \mid C(\{K_{BPub}, x\} K_{TTPPriv}, B, S_r)$
(GL4.1.D)	At least one of the cryptographic expressions, belonging to a direct POSH challenge-response handshake using symmetric encryption should contain the identity of the verifier or prover. $\forall y \in CRH(P, A, B) : \exists \{x\} K_{AB}, S_p \mid C(y, \{x\} K_{AB}, S_p) : (C(\{x\} K_{AB}, A, S_p) \vee C(\{x\} K_{AB}, B, S_p))$
(GL4.1.I)	At least one of the cryptographic expressions, belonging to an indirect POSH challenge-response handshake using symmetric encryption should contain the prover's identity (B). $\forall y \in CRH(P, A, B) : \exists \{x\} K_{ATTP}, S_p \mid C(y, \{x\} K_{ATTP}, S_p) : C(\{x\} K_{ATTP}, B, S_p)$
(GL4.2.D)	At least one of the cryptographic expressions, belonging to a direct SOPH challenge-response handshake using symmetric encryption should contain the identity of the verifier or prover. $\forall y \in CRH(P, A, B) : \exists \{x\} K_{AB}, S_o \mid C(y, \{x\} K_{AB}, S_o) : (C(\{x\} K_{AB}, A, S_o) \vee C(\{x\} K_{AB}, B, S_o))$
(GL4.2.I)	At least one of the cryptographic expressions, belonging to an indirect SOPH challenge-response handshake, using symmetric encryption, should contain the prover's identity (B). $\forall y \in CRH(P,A,B): \exists \{x\} K_{ATTP}, S_o \mid C(y, \{x\} K_{ATTP}, S_o): C(\{x\} K_{ATTP}, B, S_o)$
(GL4.3.D)	At least one of the cryptographic expressions, belonging to a direct SOSH challenge-response handshake using symmetric encryption, should contain the identity of the verifier or prover. $\forall y \in CRH(P,A,B): \exists \{x\} K_{AB}, S_r, r \in \{o,p\} \mid C(y, \{x\} K_{AB}, S_r): (C(\{x\} K_{AB}, A, S_r) \vee C(\{x\} K_{AB}, B, S_r))$
(GL4.3.I)	At least one of the cryptographic expressions (emitted by the verifier or by the prover), belonging to an indirect SOSH challenge-response handshake using symmetric encryption should contain the prover's identity (B). $\forall y \in CRH(P, A, B) : \exists \{x\} K_{ATTP}, S_r, r \in \{o, p\} \mid C(y, \{x\} K_{ATTP}, S_r) : C(\{x\} K_{ATTP}, B, S_r)$
(GL4.4)	At least one of the cryptographic expressions, belonging to a POSH challenge-response handshake using encryption with private keys should contain the verifier's (recipient's) identity. $\forall y \in CRH(P,A,B): \exists \{x\} K_{BPriv}, S_p \mid C(y, \{x\} K_{BPriv}, S_p): C(\{x\} K_{BPriv}, A, S_p)$
(GL4.5)	All the cryptographic expressions, belonging to a SOSH challenge-response handshake using encryption with private keys should contain the recipient's identity (i.e. the cryptographic expressions emitted by the verifier should contain the prover's identity and the cryptographic expressions emitted by the prover should contain the verifier's identity). $\forall y \in CRH(P,A,B): (\forall \{x\} K_{APriv}, S_o \mid C(y, \{x\} K_{APriv}, S_o): C(\{x\} K_{APriv}, B, S_o)) \wedge (\forall \{z\} K_{BPriv}, S_p \mid C(y, \{z\} K_{BPriv}, S_p): C(\{z\} K_{BPriv}, A, S_p))$
(GL4.6)	At least one of the cryptographic expressions, belonging to a SOPH challenge-response handshake using encryption with public keys should contain the verifier's (sender's) identity. $\forall y \in CRH(P,A,B): \exists \{x\} K_{BPub}, S_o \mid C(y, K_{BPub}, S_o): C(\{x\} K_{BPub}, A, S_o)$
(GL4.7)	At least one of the cryptographic expressions, belonging to a SOSH challenge-response handshake using encryption with public keys should contain the sender's identity (i.e. the cryptographic expressions emitted by the verifier should contain the verifier's identity or the cryptographic expressions emitted by the prover should contain the prover's identity). $\forall y \in CRH(P,A,B): (\exists \{x\} K_{BPub}, S_o \mid C(y, \{x\} K_{BPub}, S_o): C(\{x\} K_{BPub}, A, S_o)) \vee (\exists \{z\} K_{APub}, S_p \mid C(y, \{z\} K_{APub}, S_p): C(\{z\} K_{APub}, B, S_p))\ (\exists \{z\} K_{APub}, S_p \mid C(y, \{z\} K_{APub}, S_p): C(\{z\} K_{APub}, B, S_p))$

REFERENCES

Abadi M, Needham R. Prudent engineering practice for cryptographic protocols. IEEE Transactions on Software Engineering 1996;22:6–15.

Abadi M. Explicit communication Revisited: two new attacks on authentication protocols. IEEE Transactions on Software Engineering March 1997;23(3):185–6.

Ahirwal RR, Sonwanshi SS. An efficient and secure ID-based remote user authentication scheme using smart card. International Journal of Applied Information Systems (IJAIS) February 2012;1(6). ISSN: 2249-0868. Foundation of Computer Science FCS, New York, USA.

Altaf A, Javed M, Ahmed A. Security Enhancements for Privacy and Key Management Protocol in IEEE 802.16e-2005. Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing; 2008. pp. 335–9.

Anderson R, Needham R. Robustness principles for public key protocols. In: Proceedings of CRYPTO; 1995. pp. 236–47.

Aura T. Strategies against replay attacks. In: Proceedings of the 10th IEEE Computer Society Foundations Workshop. Rockport, MA: IEEE Computer Society Press; June 1997. pp. 59–68.

Beller MJ, Chang L-F, Yacobi Y. Privacy and authentication on a portable communications system. IEEE Journal on Selected Areas in Communications 1993;11(6):821–9.

Bird R, Gopal I, Herzberg A, Janson P, Kutten S, Molva R, et al. Systematic design of two-party authentication protocols. In: Advances in Cryptology: Crypto'91, Lecture Notes in Computer Science 576. Berlin: Springer-Verlag; 1992. pp. 44–61.

Bird R, Gopal I, Herzberg A, Janson P, Kutten S, Molva R, et al. Systematic design of a family of attack-resistant authentication protocols. IEEE Journal on Selected Areas in Communications June 1993;11(5):679–93.

Burrows M, Abadi M, Needham R. A logic of authentication. ACM Transactions on Computer Systems TOCS 1990;8(1):18–36.

Carlsen U. Optimal Privacy and Authentication on a portable communications system. ACM Operating System Reviews 1994:16–23.
Carlsen U. Cryptographic protocol flaws. In: Proc. IEEE Computer Security Foundations Workshop VII. IEEE Computer Press; June 1994. pp. 192–200.
CCITT. The directory authentification framework. Draft Recommendation X.509, Version 7; 1987.
Chen TH, Shih WK. A robust mutual authentication protocol for wireless sensor networks. ETRI Journal 2010;32(5):704–12.
Clark J, Jacob J. On the security of recent protocols. Information Processing Letters 1995;56:151–5.
Clark J, Jacob J. A survey of authentication protocol literature: Version 1.0; November 1997.
Coffey T, Dojen R, Flanagan T. Formal verification: an Imperative step in the design of security protocols. Comput Networks Journal December 2003;43(5):601–18. Elsevier Science.
Das ML. Two-factor user authentication in wireless sensor networks. IEEE Trans on Wireless Communications 2009;8(3):1086–90.
Denning D, Sacco G. Timestamps in key distributed protocols. Communications of the ACM 1981;24(8):533–5.
Dojen R, Coffey T. Layered proving trees: a novel approach to the automation of logic-based security protocol verification. ACM Transactions on Information Systems Security (TISSEC) 2005;8(3):287–311.
Dojen R, Lasc I, Coffey T. Establishing and fixing a freshness flaw in a key-distribution and authentication protocol. IEEE International Conference on Intelligent Computer Communication and Processing, August 2008a. pp. 185–92.
Dojen R, Pasca V, Coffey T. Impersonation attacks on a Mobile security protocol for end-to-end communications. In: Security and privacy in Mobile information and Communication systems, Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Tele-communications Engineering (LNICST), vol. 17; 2009. pp. 278–87.
Dojen R, Zhang F, Coffey T. On the formal verification of a cluster based key management protocol for wireless sensor networks. 27th IEEE International Performance Computing and Communications Conference (IPCCC08) – Workshop of Information and Data Assurance (WIDA08), Austin, Texas, USA, 2008b. pp. 499–506.
Dojen R, Jurcut A, Coffey T, Györödi C. On establishing and fixing a parallel session attack in a security protocol. Intelligent distributed computing, systems and applications, Vol. 162. Springer Berlin/Heidelberg; 2008c. pp. 239–44.
Espelid Y, Netland L, Klingsheim A, Hole K. A proof of concept attack against Norwegian internet banking systems. In: Proc. of the 12th International Conference on Financial cryptography and data security (FC08), Cozumel, Mexico; January 28–31, 2008. pp. 197–201.
Fu X, Guo Y. A lightweight RFID mutual authentication protocol with ownership transfer. Advances in wireless sensor networks, Communications in Computer and Information Science, vol. 334; 2013. pp. 68–74.
Gong L, Syverson P. Fail-stop protocols: an approach to designing secure protocols. In: 5th International Working Conference on Dependable Computing for Critical Applications; September 1995. pp. 44–55.
Gordon A, Jeffrey A. Types and effects for asymmetric cryptographic protocols. Journal of Computer Security 2004;12(3/4):435–84.
He D, Chen J, Hu J. An id-based client authentication with key agreement protocol for mobile client–server environment on ECC with provable security. Information Fusion 2012;13(3):223–30.
Heather J, Lowe G, Schneider S. How to prevent type flaw attacks on security protocols. IEEE Computer Society; 2000. pp. 255–68.
Hsiang HC, Shih WK. Weaknesses and improvements of the Yoon-Ryu-Yoo remote user authentication scheme using smart cards. Computer Communications 2009;32:649–52.
Huang HF, Chang YF, Liu CH. Enhancement of two-factor user authentication in wireless sensor networks. In: Proceedings of the 6th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP'10); October 2010. pp. 27–30.
Hwang T, Chen Y. On the security of splice/as : the authentication system in wide internet. Information Processing Letters 1995;53:97–101.
Hwang T, Lee N, Li C, Ko M, Chen Y. Two attacks on neumann-stubblebine authentication protocols. Information Processing Letters 1995;53:103–7.
IEEE Std. 802.16e/D12. IEEE Standard for local and metropolitan area networks, part 16: Air interface for fixed and mobile broadband wireless access systems. IEEE Press; 2005.
Jin Y, Sun H, Xin W, Luo S, Chen Z. Lightweight RFID mutual authentication protocol against feasible problems. In: Qing S, Susilo W, Wang G, Liu D, editors. ICICS 2011. LNCS, vol. 7043. Heidelberg: Springer; 2011. pp. 69–77.
Johnston D, Walker J. Overview of IEEE 802.16 Security. IEEE Security & Privacy 2004;2(3).
Jurcut A, Coffey T, Dojen R. Establishing and fixing security protocols weaknesses using a logic-based verification tool. Journal of Communications November 2013;8(11). ISSN: 1796-2021:795–806.
Kao I, Chow R. An efficient and secure authentication protocol using uncertified keys. Operating Systems Review 1995;3(29):14–21.
Kehne A, Schönwälder J, Langendörfer H. Multiple authentications with a nonce-based protocol using generalized timestamps. In: Proc. ICCC '92, Genua; 1992.
Khan MK, Alghathbar K. Cryptanalysis and security improvements of two-factor user authentication in wireless sensor networks. Sensors 2010;10(3):2450–9.
Khan MK, Zhang J. An efficient and practical fingerprint-based remote user authentication scheme with smart cards. In: ISPEC 2006, Lecture Notes in Computer Science 3903; 2006. pp. 260–8.
Kumar M, Gupa MK, Kumari S. An Improved smart card based remote user authentication scheme with session key agreement during the verification phase. Journal of Applied Computer Science & Mathematics 2011;5(11). Suceava.
Lasc Ioana, Dojen Reiner, Coffey Tom. On the detection of desynchronisation attacks against security protocols that use dynamic shared secrets. Computers & Security 2013;32:115–29.
Lee CC, Yang CC, Hwang MS. A new privacy and authentication protocol for end-to-end mobile users. International Journal of Communication Systems 2003;16(9):799–808.
Lee S, Kim H, Yoo K. Efficient nonce-based remote user authentication scheme using smart cards. Applied Mathematics and Computation 2005;167(1):355–61.
Lowe G. Some new attacks upon security protocols. In: IEEE Computer Society Press, editor. Proceedings of the Computer Security Foundations Workshop VIII; 1996. pp. 162–9.
Lowe G. Casper: a compiler for the analysis of security protocols. In: Proceedings of the 10th Computer Security Foundations Workshop; 1997.
Lowe G, Roscoe AW. Using CSP to detect errors in the TMN protocol. IEEE Transactions on Software Engineering 1997;10(23):659–69.
Lowe G. An attack on the Needham-Schroeder public key authentication protocol. Information Processing Letters Nov 1995;56(3):131–6.
Malladi S, Alves-Foss J, Heckendorn R. On preventing replay attacks on security protocols. In: Proceedings International Conference on Security and Management, ICSM'02; 2002. pp. 77–83.

Mu Y, Varadharajan V. On the design of security protocols for Mobile communications. In: Information security and Privacy, Lecture Notes in Computer Science. Springer-Verlag; 1996. pp. 134–45.
Nam J, Kim S, Park S, Won D. Security analysis of a nonce-based user authentication scheme using smart cards. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences Jan 2007;E90–A(1):299–302.
Needham R, Schroeder M. Using encryption for authentication in large networks of computers. Communications of the ACM 1978;21(12):993–9.
Neumann B, Stubblebine S. A note on the use of timestamps as nonces. Operating Systems Review April 1993;2(27):10–4.
Nyang D, Lee M. Improvement of Das's two-factor authentication protocol in wireless sensor networks. Cryptology ePrint Archive 2009/631; 2009., http://eprint.iacr.org/2009/631.pdf.
Paulson L. Relations between secrets: two formal analyses of the Yahalom protocol. Journal of Computer Security 2001;3(9):197–216.
Safford D, Schales D, Hess D. Texas A&M University Anarchistic Key Authorization (AKA). In: Proc. Sixth Usenix Security Symp; July 1996. pp. 179–85.
Satyanarayanan M. Integrating security in a large distributed system. ACM Transactions on Computer Systems 1989;3(7):247–80.
Shen L, Feng H, Qiu Y, Ding H. A new kind of cluster-based key management protocol in wireless sensor network. In: Proceedings of the IEEE Conference of Networking, Sensing and Control; 2008.
Syverson P. A taxonomy of replay attacks. In: Proceedings of the Computer Security Foundations Workshop (CSFW97); June 1994. pp. 187–91.
Tatebayashi M, Matsuzaki N, Newman D. Key distribution protocol for digital mobile communication systems. In: Advance in Cryptology – CRYPTO '89, volume 435 of LNCS. Springer-Verlag; 1989. pp. 324–33.
The Norwegian Banks' Payment and Clearing Centre: BankID FOI white paper. (Release 2.0.0); 2006 (in Norwegian).
Tian Y, Chen G, Li J. A new ultralightweight RFID authentication protocol with permutation. IEEE Communications Letters 2012;16(5):702–5.
Wang D, Ma C. Cryptanalysis of a remote user authentication scheme for mobile client–server environment based on ECC. Information Fusion 2013;14:498–503.
Woo T, Lam S. A lesson on authentication protocol design. ACM Operating Systems Review 1994;28(3):24–37.
Xu S, Huang C. Attacks on PKM protocols of IEEE 802.16 and its Later versions. Columbia: Computer Science and Engineering Department, University of South Carolina; September, 2006.
Xu J, Zhu W-T, Feng D-G. Improvement of a fingerprint-based remote user authentication scheme. International Journal of Security and its Applications July, 2008;2(3).
Yamaguchi S, Okayama K, Miyahara H. The design and implementation of an authentication system for the wide area distributed environment. IEICE Transactions on Information Systems November 1991;E74(11):3902–9.
Ylönen T. SSH transport layer protocol; June 1996. Internet draft, at, ftp://ds.internic.net/internet-drafts/draft-ietf-tls-ssh-00.txt.
Yoo S, Park KY, Kim J. A security-performance balanced user authentication scheme for wireless sensor networks. International Journal of Distributed Sensor Networks 2012;2012. Article ID 382810, 11 pages.
Yoo S, Lee H, Kim J. A performance and usability aware secure two-factor user authentication scheme for wireless sensor networks. International Journal of Distributed Sensor Networks 2013;2013.
Yoon E, Yoo K. More efficient and secure remote user authentication scheme using smart cards. In: Proc. of 11th International Conf. on Parallel and Distributed Systems, vol. 2; 2005. pp. 73–7.
Yoon E, Ryu E, Yoo K. Further improvement of an efficient password based remote user authentication scheme using smart cards. IEEE Transactions on Consumer Electronics 2004;50:612–4.
Zhuang X, Wang Z, Chang C, Zhu Y. Security analysis of a new ultra-lightweight RFID protocol and its improvement. Journal of Information Hiding and Multimedia Signal Processing July 2013;4(3). ISSN: 2073-4212. Ubiquitous International.

Anca D. Jurcut received title of Bachelor of Mathematics and Computer Science from West University of Timisoara, Romania (2007) and 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, Formal Verification of Cryptographic Protocols, Logics Based Verification Tehniques, Data Security and Network Security, Software Engineering, 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 is currently employed as Lecturer in the Department of Electronic and Computer Engineering at the University of Limerick, Ireland. Received title of Diplom-Ingenieur (FH) (1999) from University of Applied Sciences Osnabrück, Germany, Master of Engineering (2000) from University of Limerick and Doctor of Philosophy (2004) from University of Limerick. His research interests include design and verification of security protocols, design and implementations of verification techniques for security protocols, data and network Security, automated theorem proving and artificial intelligence.