Design Requirements to Counter Parallel Session Attacks in Security Protocols

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

***Abstract* - This work is concerned with the possible exploitation of weaknesses in security protocols by attackers using parallel session attacks and discovering ways of eliminating these weaknesses. A new analysis is presented on the reasons why security protocols, with certain weaknesses in their design, are vulnerable to parallel session attacks. Building on this analysis a new set of design requirements is proposed, whose aim is to eliminate these vulnerabilities. The proposed set of design requirements is evaluated by applying them to a range of security protocols with known weaknesses as well as protocols known to be free of these weaknesses. The results of the evaluation indicate that the set of design requirements are effective as: protocols with known weaknesses violate some of the rules, while protocols without weaknesses do not violate any of the rules.**

Keywords—security protocols; design requirements; parallel session attacks

I. INTRODUCTION

Security protocols are used to protect the integrity and confidentiality of information send through insecure communication networks. A security protocol should enforce the data exchange between honest participants, while the dishonest ones should be denied any benefit of it. However, if a security protocol contains a flaw, some of its goals may not be achieved. An attacker can exploit such a flaw to gain an advantage in the communication, e.g. to impersonate another legitimate party. Many security protocols have been found to contain weaknesses, which have been subsequently exploited by attacks [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] such as replay and parallel session.

In a parallel session attack, the attacker can successfully subvert the security goals of the protocol by utilising two or more protocol runs executed concurrently. Messages from one protocol run are used to form spoofed messages in another run. The attacker may also manipulate the protocol participants in multiple roles such as initiator and responder.

In this paper we investigate the reasons why parallel session attacks can be successfully deployed against protocols. We examine the structure of protocol message exchanges between communicating participants and we develop a finite set of vulnerabilities that are exploitable by these attacks. We demonstrate possible parallel session attacks for these vulnerabilities, and we propose a new set of protocol design requirements whose aim is to eliminate all vulnerabilities identified.

The remainder of this paper is organized as follows: Section II introduces the definitions used. Section III presents an analysis on the reasons why parallel session attacks succeed against security protocols. A new set of design requirements, to avoid protocol weaknesses that are exploitable by parallel session attacks for a broad spectrum of security protocols, is then proposed. Section III presents an analysis on the reasons why parallel session attacks succeed against security protocols. A new set of design requirements in the form of design rules is then proposed. In sections IV the effectiveness of the proposed design requirements is evaluated by way of an empirical study on a set of protocols with known vulnerabilities and their amended versions. The proposed design rules are applied in detail to one protocol to establish its conformance, while the conformance of a range of protocols is presented in summary form. Section V concludes the paper.

II. DEFINITIONS

This section introduces the definitions used through this paper.

Definition 1. A protocol P is a set of ordered steps {S1, S2,….,Sz}, z>=1, executed in any run of P. A protocol step Sr is defined as: Sr: $A \rightarrow B$: 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 Sr and B is the recipient.

Definition 2. An initiation step is any step in a protocol, where any principal that is not a trusted third party (denoted through this paper as TTP) is the sender.

Definition 3. A response step Sp is the first step after an initiation step So, where the recipient of Sp is the sender of So.

Definition 4. A cryptographic transformation c is either a cryptographic expression ({x}k) or a hashed expression (H(x)).

Definition 5. A signed statement is a cryptographic expression where the signing key is a private key.

Definition 6. Two expressions x and y are symmetric if both contain components of the same type and in the same order (e.g. expression1: (Na, A,) and expression2: (Nb, B), where first component of expression1 *Na* and expression2 *Nb* are of type *nonce* and the second component of expression1 *A* and expression2 *B* are of type *principal*).

Definition 7. 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 8. The generator of an expression x is the sender of the step Sr, in which x appears for the first time.

Definition 9. Two messages x and y are principal value type equivalent 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 and y_i are identical*
- *if x_i is a TTP then y_i is also a TTP*
- *if x_i is the generator of x then y_i is the generator of y*
- *if x_i is the intended recipient of x then y_i is the intended recipient of y*

Definition 10. A cryptographic transformation c is a parent cryptographic transformation if it is not contained (i.e. included) in any other cryptographic transformation.

Definition 11. A receiver bound cryptographic transformation 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 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 12. A challenge-response nonce handshake is a message exchange where fresh data x_A is a challenge generated by a principal (A) acting as verifier and then delivered as part of a cryptographic expression to a principal (B) acting as prover. The prover authenticates itself to the verifier by means of the received fresh data x_A.

III. Types of Weaknesses, Attacks and Proposed Design Rules

In this section we analyse the reasons why parallel session attacks succeed over a broad spectrum of security protocols. The methodology adopted for this analysis is to characterise the general circumstances under which parallel session attacks may exist by examining the structure of message exchanges of security protocols with known vulnerabilities. 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.

The work is presented under three categories: (A) message symmetry; (B) signed statements; (C) challenge-response handshakes. For each category, the types of weaknesses are identified, possible parallel session attacks are summarised and the appropriate design rules to avoid these weaknesses are constructed.

The notation used in this section, when describing potential attacks, is as follows:

- *I* - denotes an intruder/attacker/dishonest principal,
- *I(A), I(B), I(TTP)* - denotes an intruder I impersonating principal A, B or TTP,
- *i.Sr. A → I(B): c* – denotes that in step Sr of run *i* principal A sends to B expression *c*, where intruder I impersonating principal B (I(B)) intercepts this message.

A. Message Symmetry - Possible Parallel Session Attacks

In the case of two cryptographic transformations denoted as c1 and c2 that are symmetric and principal value type equivalent, c2 can be used instead of expression c1, or c1 can be reconstructed based on expression c2 in a parallel protocol run.

1) Mountable attacks

For this category mountable attacks are shown in Fig. 1 to Fig. 4, which consider the possible situations of c1 and c2 being exchanged directly or indirectly (via a TTP) between two principals A and B. As the expression c1' required in step Sq of second run is symmetric and principal type value equivalent with c2, the receiving principal cannot distinguish the replayed c2 from a genuine expression c1'.

i.Sq. A → B: c1
i.Sr. B → I(A): c2
ii. Sq. I(B) → A: c2
ii. Sr. A → I(B): c3

Fig. 1. Potenial attack type 1

i.Sq. A → I(TTP): c1
i.Sr. B → I(TTP): c2
ii. Sq. I(B) → TTP: c2
ii. Sr. I(A) → TTP: c1

Fig. 2. Potenial attack type 2

i.Sq. A → I(TTP): c1
i.Sr. TTP→ I(B): c2
ii. Sq. I(B) → TTP: c2
ii. Sr. TTP→ A: c3

Fig. 3. Potenial attack type 3

i.Sq. TTP → A: c1
i.Sr. TTP→ I(B): c2
ii. Sq. I(TTP) → B: c2

Fig. 4. Potenial attack type 4

2) Proposed design rule for preventing message symmetry

In order to prevent message symmetry vulnerabilities exploitable by parallel session attacks, the design rule described in Table I should be obeyed.

TABLE I. Proposed Design Rules for Preventing Symmetry

R1	*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.*

B. Signed Statements - Possible Parallel Session Attacks

In the case of a signed statement, that is not receiver bound and does not include appropriate identities of principals, an intruder can impersonate a legitimate principal using a parallel protocol run.

1) Mountable attacks

For this category mountable attacks are shown in Fig. 5 to Fig. 8. These attack scenarios consider the following situations where:

- A signed statement{x}KaPriv is a parent cryptographic expression (Fig. 5);
- A signed statement {x}KaPriv is contained by a parent cryptographic expression {y}KbPub that is encrypted with public key of a principal B, KbPub (Fig. 6);
- A signed statement {x}KaPriv is contained by a parent cryptographic expression {y}Kab that is encrypted with symmetric key Kab shared by two principals A and B (Fig. 7);
- A signed statement is for public key distribution (Fig. 8).

i.Sr. A → I(B): {x}KaPriv
ii. Sr. I(A) → C: [x}KaPriv

Fig. 5. Potenial attack type 1

i.Sr. A → I: {{x}KaPriv,…}KiPub
ii. Sr. I(A) →C: {{x}KaPriv,…}KcPub

Fig. 6. Potenial attack type 2

i.Sq. C → I: {Kai,…}KiPub
ii. Sq. I(C) → B: {Kai,…}KbPub
i.Sr. A → I: {{x}KaPriv,…}Kai
ii. Sr. I(A) → B: {{x}KaPriv,…}Kai

Fig. 7. Potenial attack type 3

i.Sq. A → I(TTP): $request_B$
ii. Sq. I(A) → TTP: $request_I$
ii. Sr. TTP →I(A): {KiPub,z}KttpPriv
i.Sr. I(TTP) → A: {KiPub,z}KttpPriv
i.Su. A → I(B): {w}KiPub

Fig. 8. Potenial attack type 4

Fig. 5 shows how an intruder I can impersonate principal A if component x is not receiver bound: I replays message {x}KaPriv obtained in step *i.Sr* of run *i* as valid signed message of *ii.Sr* in a parallel run *ii* with any principal C. C has no way to establish that message{x}KaPriv was originally intended for B.

Fig. 6 shows how a dishonest principal I can impersonate principal A if component y does not contain at least one receiver bound cryptographic transformation: I can use the subcomponents of {y}KiPub obtained in step *i.Sr* of run *i* to compute {y'}KcPub as a valid expression of *ii.Sr* in a parallel run with principal C. The attacker removes the outer encryption of {y}KiPub, 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'}KcPub was intended for I. Further, as {y'}KcPub contains signed message {x}KaPriv, C believes that the message was sent by A.

Fig. 7 shows how a dishonest principal I can impersonate A if component y does not contain at least one receiver bound cryptographic transformation. Assume key Kab is generated either by A or TTP and is transmitted as part of a cryptographic expression {z}KbPub in step Sq prior to Sr. Further, {z}KbPub is free of any receiver bound constituent signed by C. The attacker I can use the subcomponents of {z}KiPub obtained in step *i.Sq* of run *i* to compute {z'}KbPub as a valid expression of *ii.Sr* in a parallel run with B. This is accomplished by removing the outer encryption of {z}KiPub, replacing any data indicating identity of receiver I with data indicating B and re-encrypting z', which includes symmetric key Kai, 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'}KbPub is a legitimate message from C and accepts the contained key Kai as a session key for communication with A. In a similar process attacker I can create component {y'}Kai that contains {x}KaPriv as required for step *ii.Sr*. Principal B has no way to detect that {y'}Kai was originally intended for I, as the message is not receiver bound. Further, as {y'}Kai contains signed message {x}KaPriv, B believes that the message was sent by principal A.

Fig. 8 shows how a legitimate but dishonest principal I can mislead other principals, to bind its public key (KiPub) to the identity of another principal if a signed statement ({KbPub,z}KttpPriv) distributing a public key (KbPub) does not include identity of key's owner (B). In run *i* principal I intercepts message *i.Sp* intended for TTP and starts a second run *ii*. In this second run I pretends to be principal A initiating a session with I by sending a request for I's public key to TTP. According to the specification of the exchange, TTP sends the signed statement containing public key of I in step *ii.Sr*. As this signed statement does not contain the identity of the owner of the public key, attacker I can replay it as a valid signed statement containing principal B's key in step *i.Sr* of first run. Principal A cannot distinguish the replayed *ii.Sr* containing I's public key from a genuine response containing B' 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 KiPub and I can successfully impersonates B to A.

2) Proposed design rules for signed statements

In order to prevent vulnerabilities exploitable by parallel session attacks, the design rules for signed statements described in Table II should be obeyed.

C. Challenge Response Handshakes - Possible Parallel Session Attacks

In the case of a challenge-response authentication protocol P with principals A acting as a verifier and B as a prover, if the cryptographic transformations of the challenge-response handshakes do not include appropriate identities of principals, an intruder can impersonate a legitimate principal using a parallel run.

We use the classification of a challenge-response handshake as given in [11]: *POSH* (*Public Out Secret Home*), *SOPH* (*Secret Out Public Home*) and *SOSH* (*Secret Out Secret Home*). Additionally, we define a *direct challenge-response handshake* as a handshake where the messages are exchanged directly between principals without involvement of a trusted third party TTP. In an *indirect challenge-response handshake*, the messages are exchanged indirectly between principals with the assistance of a TTP.

TABLE II. PROPOSED DESIGN RULES FOR SIGNED STATEMENTS

R2.1	*If a signed message {x}KaPriv is a parent cryptographic expression, then {x}KaPriv should be receiver bound.*
R2.2	*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.*
R2.3	*If a signed message {x}KaPriv of a step Sr of protocol P is contained in a parent cryptographic expression {y}Kab that fulfils the following conditions:* • *the symmetric key Kab is generated by a principal C belonging to {TTP, A} and is transmitted by C in a cryptographic expression {z}KbPub encrypted with B's public key in step Sq prior to Sr* • *{z}KbPub in step Sq is free of any receiver bound messages signed by C* *then component y should contain at least one receiver bound cryptographic transformation.*
R2.4	*Any signed statement intended for public key distribution should include the identity of the public key owner.*

1) Mountable attacks

For this category mountable attacks are shown in Fig. 9 to Fig 19. These figures consider both direct and indirect challenge-response handshakes for symmetric key encryption, while only direct handshakes need to be considered for asymmetric encryption.

Fig. 9 shows how an intruder I can impersonate principal B, if the cryptographic expression transmitted as part of a *direct POSH challenge-response handshake using symmetric keys,* does not contain the identity of either prover or verifier: In run *i* the intruder intercepts message *i.So* intended for prover B and starts a parallel run *ii* of the handshake with A. In this second run attacker I pretends to be B and uses the same data x_A. Principal A sends the response to the challenge in step *ii.Sp*. As the contained cryptographic expression $\{F(x_A),\ldots\}Kab$ does not contain the identity of either verifier or prover, I can replay this message in step *i.Sp* of the first run. A cannot distinguish the replayed *ii.Sp* sent by the intruder from a genuine response *i.Sp* sent by B. Consequently, A believes it has established a session with B in run *i* and that B has established a session with it in run *ii*, even though B is in fact absent - the intruder I has successfully impersonated principal B to A.

i.So. A → I(B): x_A
ii. So. I(B) → A: x_A
ii. Sp. A → I(B): $\{F(x_A),\ldots\}Kab$
i.Sp. I(B) → A: $\{F(x_A), \ldots\}Kab$

Fig. 9. Potential attack type 1

i.So. A → I(B): x_A
ii. So. I(A) → C: x_A
ii. Si. C → TTP: *request*
ii. Sp. TTP → I(A): $\{F(x_A), \ldots\}Kattp$
i.Si. omitted
i.Sp. I(TTP) → A: $\{F(x_A), \ldots\}Kattp$

Fig. 10. Potenial attack type 2

Fig. 10 shows how an intruder I can impersonate principal A, if the cryptographic expression transmitted as part of an *indirect POSH challenge-response handshake using symmetric keys* (where the messages are exchanged indirectly between principals A and B via a TTP), does not contain the identity of prover B: In run *i* the intruder intercepts message *i.So* intended for prover B and starts a parallel run *ii* of the handshake with a principal C (in run *ii*, C is the prover*)*. In this second run attacker I pretends to be principal A and uses the same data x_A. Principal C sends a request to TTP in step *ii.Si* and TTP responses to challenge x_A in step *ii.Sp*, according with the specification of the exchange. As the contained cryptographic expression $\{F(x_A),\ldots\}Kattp$ does not contain the identity of prover C, I can replay this message in step *i.Sp* of the first run. A cannot distinguish the replayed *ii.Sp* sent by the intruder from a genuine response *i.Sp* sent by TTP. Consequently, A believes it has established a session with B in run *i*, even though B is in fact absent.

Fig. 11 shows how an intruder I can impersonate principal B, if the cryptographic expression transmitted as part of a direct SOPH challenge-response handshake using symmetric keys, does not contain the identity of either prover or verifier: In run *i* the intruder intercepts the message *i.So* intended for prover B and starts parallel run *ii* of the handshake with A. In this second run I pretends to be B and uses the same cryptographic expression {xA,…}Kab. 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 *ii.Sp*, which contains the decrypted data F(xA). Attacker I intercepts step *ii.Sp* and uses the contained data as correct response in step *i.Sp* of the first run. Consequently, A believes it has established a session with B in run *i* and that B has established a session with it in run *ii*, even though B is in fact absent - the intruder I has successfully impersonated principal B to A.

i.So. A → I(B): $\{x_A, \ldots\}Kab$
ii. So. I(B) → A: $\{x_A, \ldots\}Kab$
ii. Sp. A → I(B): $F(x_A)$
i.Sp. I(B) → A: $F(x_A)$

Fig. 11. Potential attack type 3

i.So. A → I(B): $\{x_A, \ldots\}Kattp$
ii. So. I(A) → C: $\{x_A, \ldots\}Kattp$
ii. Si. C → TTP: $\{x_A,\ldots\}Kattp$
ii. Sp. TTP → I(A): $F(x_A)$
i.Si. omitted
i.Sp. I(TTP) → A: $F(x_A)$

Fig. 12. Potenial attack type 4

Fig. 12 shows how an intruder I can impersonate principal A, if the cryptographic expression transmitted as part of an *indirect SOPH challenge-response handshake using symmetric keys,* does not contain the identity of prover B: In run *i* the intruder intercepts the message *i.So* intended for prover B and starts parallel run *ii* of the handshake with a principal C. In this second run I pretends to be A and uses the same cryptographic expression $\{x_A,\ldots\}Kattp$. 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 *ii.Sp*, which contains the decrypted data $F(x_A)$. I intercepts step *ii.Sp* and uses the contained data as the correct response in step *i.Sp* of the first run. Consequently, A believes it has established a session with B in run *i*, even though B is in fact absent.

i.So. A → I(B): $\{x_A, \ldots\}$Kab
ii. So. I(B) → A: $\{x_A, \ldots\}$Kab
ii. Sp. A → I(B): $\{F(x_A), \ldots\}$Kab
i.Sp. I(B) → A: $\{F(x_A), \ldots\}$Kab

Fig. 13. Potential attack type 5

i.So. A → I(B): $\{x_A, \ldots\}$Kattp
ii. So. I(A) → C: $\{x_A, \ldots\}$Kattp
ii. Si. C → TTP: $\{x_A, \ldots\}$Kattp
ii. Sp. TTP → I(A): $\{F(x_A),\ldots\}$Kattp
i.Si. omitted
i.Sp. I(TTP) → A: $\{F(x_A),\ldots\}$Kattp

Fig. 14. Potenial attack type 6

Fig. 13 shows how an intruder I can impersonate principal B, if both cryptographic expressions transmitted as part of a *direct SOSH challenge-response handshake using symmetric keys,* do not contain the identity of either prover or verifier: In run *i* the intruder intercepts message *i.So* intended for B and starts a second run *ii* of the handshake with A. In this second run I pretends to be B and uses the same cryptographic expression $\{x_A,\ldots\}$Kab. As $\{x_A,\ldots\}$Kab does not contain any identity of verifier or prover, principal A accepts the message and sends the response to the challenge in step *ii.Sp*. Again, as the contained cryptographic expression $\{F(x_A), \ldots\}$Kab does not contain the identity of either verifier or prover, I can replay this message in step *i.Sp* of the first run. A cannot distinguish the replayed *ii.Sp* sent by the intruder from a genuine response *i.Sp* sent by B. Consequently, A believes it has established a session with B in run *i* and that B has established a session with it in run *ii*, even though B is in fact absent - the intruder I has successfully impersonated principal B to A.

Fig. 14 shows how an intruder I can impersonate principal A, if both cryptographic expressions transmitted as part of an *indirect SOSH challenge-response handshake using symmetric keys,* do not contain the identity of prover B: In run *i* the intruder intercepts message *i.So* intended for prover B and starts a second run *ii* of the handshake with a principal C. In this second run I pretends to be A and uses the same cryptographic expression $\{x_A,\ldots\}$Kattp. As $\{x_A,\ldots\}$Kattp 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 *ii.Sp*. Again, as the contained cryptographic expression $\{F(x_A),\ldots\}$Kattp does not contain the identity of the prover, I can replay this message in step *i.Sp* of the first run. A cannot distinguish the replayed *ii.Sp* sent by the intruder from a genuine response *i.Sp* sent by TTP. Consequently, A believes it has established a session with B in run *i*, even though B is in fact absent.

Fig. 15 shows how an intruder I can impersonate principal B, if the cryptographic expression transmitted as part of a *direct POSH challenge-response handshake using private keys,* does not contain the recipient's identity (verifier A): In run *i* the intruder intercepts message *i.So* intended for B and starts a parallel run *ii* of the handshake with B. In this second run attacker I acts as an legitimate principal and uses the same data x_A. Principal B accepts this message and responds with step *ii.Sp*, which contains $\{F(x_A),\ldots\}$KbPriv. As $\{F(x_A),\ldots\}$KbPriv is free of any data containing recipient's identity, the attacker can replay it as a valid response message *i.Sp* in the first run. Consequently, A believes it has established a session with B in run *i* even though B does not participate in run *i*, thus the intruder I has successfully impersonated principal B to A.

i.So. A → I(B): x_A
ii. So. I → B: x_A
ii. Sp. B → I: $\{F(x_A), \ldots\}$KbPriv
i.Sp. I(B) → A: $\{F(x_A), \ldots\}$KbPriv

Fig. 15. Potential attack type 7

i.So. A → I(B): $\{x_A, \ldots\}$KaPriv
ii. So. I→ B: $\{x_A, \ldots\}$KiPriv
ii. Sp. B → I: $\{F(x_A), \ldots\}$KbPriv
i.Sp. I(B) → A: $\{F(x_A), \ldots\}$KbPriv

Fig. 16. Potenial attack type 8a

Fig. 16 shows how an intruder I can impersonate principal B, if the cryptographic expression $\{F(x_A),\ldots\}$KbPriv in step Sp of a *direct SOSH challenge-response handshake using private keys,* does not contain the recipient's identity (verifier A): In run *i* the intruder intercepts message *i.So* intended for principal B and starts a parallel run *ii* of the handshake with B. In this second run attacker I acts as an legitimate principal and uses the same data x_A encrypted with private key KiPriv. B accepts this message and responds with step *ii.Sp*, which contains $\{F(x_A),\ldots\}$KbPriv. As $\{F(x_A),\ldots\}$KbPriv is free of any data containing recipient's identity, the attacker can replay it as a valid response message *i.Sp* in the first run. Consequently, A believes it has established a session with B in run *i* even though B does not participate in run *i*, thus the intruder I has successfully impersonated principal B to A.

i.So. A → I(B): $\{x_A, \ldots\}$KaPriv
ii. So. I(A)→ C: $\{x_A, \ldots\}$KaPriv
ii. Sp. C → I(A): $\{F(x_A),\ldots\}$KcPriv

Fig. 17. Potenial attack type 8b

i.So. A → I(B): $\{x_A, \ldots\}$KbPub
ii. So. I → B: $\{x_A, \ldots\}$KbPub
ii. Sp. B → I: $F(x_A)$
i.Sp. I(B) → A: $F(x_A)$

Fig. 18. Potential attack type 9

Similarly, Fig. 17 shows how an intruder I can impersonate principal A, if the cryptographic expression $\{x_A,\ldots\}$KaPriv in step So of a *direct SOSH challenge-response handshake using private keys,* does not contain the recipient's identity (prover B): In run *i* the intruder intercepts message *i.So* intended for prover B and starts a parallel run *ii* of the handshake with a principal C. In this second run I pretends to be A and uses the same cryptographic expression $\{x_A,\ldots\}$KaPriv. As this cryptographic expression does not contain the prover's identity, C cannot distinguish the replayed *i.So* sent by the intruder, from a genuine *ii.So* sent by A. C responds with *ii.Sp* and believes it has established a session with A in run *ii*, even though A does not participate in run *ii*, thus the intruder I has successfully impersonated principal A to C in *ii*.

Fig. 18 shows how an intruder I can impersonate principal B, if the cryptographic expression transmitted as part of a *direct SOPH challenge-response handshake using public keys,* does not contain the verifier's identity: In run *i* the intruder intercepts message *i.So* intended for B and starts a parallel run *ii* of the handshake with prover B. In this second run attacker I acts as an legitimate verifier and uses the same cryptographic expression $\{x_A,\ldots\}$KbPub. As this cryptographic expression does not contain the verifier's identity, B accepts it as a genuine message from I and responds with step *ii.Sp*. Attacker I uses the contained data as the correct response in step *i.Sp* of the first run. Consequently, A believes it has established a session with B in run *i*, even though B does not participate in run *i*, thus the intruder I has successfully impersonated principal B to A.

i.So. A → I(B): $\{x_A, \ldots\}KbPub$
ii. So. I → B: $\{x_A, \ldots\}KbPub$
ii. Sp. B → I: $\{F(x_A), \ldots\}KiPub$
i.Sp. I(B) → A: $\{F(x_A), \ldots\}KaPub$

Fig. 19. Potential attack type 10

Fig. 19 shows how an intruder I can impersonate principal B, if both cryptographic expressions transmitted as part of a *direct SOSH challenge-response handshake using public keys,* do not contain the identity of their sender (verifier A for the first cryptographic expression and prover B for the second): In run *i* the intruder intercepts message *i.So* intended for B and starts a second run *ii* of the handshake with prover B. In this second run attacker I acts as a legitimate verifier, but uses the same cryptographic expression $\{x_A,\ldots\}KbPub$. 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 *ii.Sp*, which contains $F(x_A)$ encrypted with I's public key. Attacker I extracts $F(x_A)$ and creates cryptographic expression $\{F(x_A),\ldots\}KaPub$ as a suitable message for *i.Sp*. Consequently, A believes it has established a session with B in run *i*, even though B does not participate in the exchange of run *i,* thus - the attacker I has successfully impersonated principal B to A.

2) Proposed design rules for challenge-response hanshakes construction

In order to prevent vulnerabilities exploitable by parallel session attacks, the design rules for challenge-response handshakes construction described in Table III should be obeyed.

IV. Evaluation of Design Requirements

This evaluation seeks to establish the conformance of a range of protocols to the proposed design requirements introduced in tables I, II and III. The design requirements are considered effective if: (i) protocols with known weaknesses violate some of the rules, (ii) protocols without weaknesses do not violate any of the rules.

The protocols chosen for this study incorporate protocols from differing categories such as: authentication, key management, secrecy, etc, with known vulnerability to parallel session attacks and their published amended versions. One authentication protocol is examined in detail. The lack of conformance of this protocol to the proposed design requirements is established. The known vulnerability of this protocol is identified as it violates one of the proposed design rules. The protocol weakness is then fixed by applying the appropriate design rule. Additionally, the conformance or lack of conformance of a range of protocols with the proposed design requirements is presented in summary form.

TABLE III. Proposed Design Rules for Handshakes Construction

R3.1	*At least one of the cryptographic expressions, belonging to one of the direct POSH, SOPH or SOSH challenge-response handshake using symmetric encryption should contain the identity of the verifier or prover.*
R3.2	*At least one of the cryptographic expressions, belonging to one of the indirect POSH, SOPH or SOSH challenge-response handshake using symmetric encryption should contain the prover's identity (B).*
R3.3	*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.*
R3.4	*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).*
R3.5	*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.*
R3.6	*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).*

A. Applying Design Rules to HCH Authentication & Key Agreement Protocol

The proposed set of design rules is now applied to the HCH id-based protocol for mobile client-server environment on elliptic curve cryptography [20], which has a known weakness that can be exploited by a parallel session attack. The protocol consists of three phases: system initialisation, client registration and mutual authentication with key agreement. In the *system initialisation phase* the server S generates parameters of the system: the elliptic curve equation E, the base point P with the order n over E, three has funtions $H_1()$, $H_2()$, $H_3()$, the message authentication code $MAC_k()$, the master key x and the public key PsPub. S keeps master key x in private and publishes the others parameters. In the *client registration phase* a client C sends its identity ID_C to the server S and receives back its private key KcPriv. The *mutual authentication phase* is executed whenever the client C wants to access the services on the server S. The steps of this phase are outlined in Fig. 20. In first step the client C chooses a random number Nc and computes messages: M={Nc}P, M'={Nc}KcPriv and $k=H_2(ID_C,Tc,M,M')$, where Tc is the current timestamp of the user. After receiving M_1, server S checks the validity of ID_C, freshness of Tc and integrity of $MAC_K(ID_C,Tc,M)$ with the key k. In the case of positive results, S chooses a random number Ns and computes messages: W={Ns}P, K_S={Ns}M

and the session key $sk=H_3(ID_C,Tc,Ts,M,W,K_S)$, where Ts is the current timestamp of the server.

1. $C \rightarrow S$: $M_1=(ID_C,Tc, M, MAC_K(ID_C, Tc,M))$
2. $S \rightarrow C$: $M_2=(ID_C,Ts, W, MAC_K(ID_C, Ts,W))$

Fig. 20. Mutual authentication phase of HCH protocol

On investigating the message exchanges of the protocol to establish if they are in accordance with all of the defined design rules (R1, R2.1-R2.4, R3.1-R3.6), it can be seen that the design rule for preventing message symmetry (R1) is not respected. The client C's login request M_1 (step 1) is symmetric to server S's response M_2 (step 2): the two cryptographic messages $MAC_K(ID_C,Tc,M)$ (part of M_1) and $MAC_K(ID_C,Ts,W)$ (M_2) are symmetric and principal value type equivalent.

As a consequence, a parallel session attack can be mounted against the authentication phase of the HCH protocol, where an attacker can forge login messages to impersonate a legitimate user.

Mounting a parallel session attack on the authentication phase of HCH protocol is detailed in Fig. 21, where an intruder I without knowing client C's password masquerades as a legal user by creating a valid login message from the eavesdropped communication between S and C. In first run *i*, intruder I intercepts and blocks response message M_2 and starts a second run *ii* impersonating client C, by sending fabricated login request $M^*_1=(ID_C, T^*c=Ts, M^*=W, MAC^*_K(ID_C,Tc,M)=MAC_K(ID_C,Ts,W))$ to server S. Note that the integrity check succeeds because $MAC^*_K(ID_C,Tc,M)$ is actually computed by S with the correct k. The server S cannot distinguish the replayed message M^*_1 sent by the attacker from a genuine message *ii.1* sent by a honest user C. Hence, S computes M^*_2 and sends the message *ii.2* in response to *ii.1*, as specified in the protocol.

i.1. $C \rightarrow S$: $M_1=(ID_C,Tc, M, MAC_K(ID_C, Tc,M))$
i.2. $S \rightarrow I(C)$: $M_2=(ID_C,Ts, W, MAC_K(ID_C, Ts,W))$
ii.1. $I(C) \rightarrow S$: $M^*_1= M_2$
ii.2. $S \rightarrow I(C)$: $M^*_2= (ID_C,T's, W', MAC_K(ID_C, T's,W'))$

Fig. 21. The attack on authentication phase of HCH protocol

Although the session key *sk* computed by S is unknown to the attacker I, the above attack reveals a weakness in the HCH protocol that can be employed to carry out other subtle (or unintended) damages to its application systems. For example, suppose that the services provided by S have to be charged, and S will begin to charge C once C has successfully logged into server S. Since attacker I can impersonate C to login S, it will fool S into wrongly charging C. Further, such an impersonation attack can distort the system logs and therefore this attack can be practical and indeed rather damaging.

1) Fixing protocol weakness

This protocol weakness can be fixed by following design rule R1, where the symmetrical structure of the cryptographic messages $MAC_K(ID_C,Tc,M)$ (part of M_1) and $MAC_K(ID_C,Ts,W)$ (part of M_2) should be broken. Here we propose that the sender's identity ID_S to be included in the composition of the second MAC in message M_2.The corrected protocol steps are given in Fig. 22.

1. $C \rightarrow S$: $M_1=(ID_C,Tc, M, MAC_K(ID_C, Tc,M))$
2. $S \rightarrow C$: $M_2=(ID_C,Ts, W, \mathbf{MAC_K(ID_S, ID_C, Ts,W)})$

Fig. 22. Application of rule R1 to authentication phase of HCH protocol

The amended protocol is no longer susceptible to the attack illustrated in Fig. 21, since if the attacker I sends $MAC_K(ID_S,ID_C,Ts,W)$ back to S (in *ii.1*) the attack will be detected as soon as the server checks the equality $MAC_K(ID_S,ID_C,Ts,W)?= MAC_K(ID_C,Ts,W)$. Thus, the above described parallel session attack will no longer be successful against the patched protocol.

B. 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 proposed design rules defined to overcome parallel session attacks. These protocols include those with known weaknesses and those that are known to be secure.

TABLE IV. EVALUATION OF DESIGN RULES – EMPIRICAL RESULTS

Analysed Protocol	Published PS Attacks	Design Rule Violation
NS PK,1978	1995[12]	R3.6
Lowe's fix NS PK,1995 [12]	No attack	None
AS RPC,1989	1996[13]	R3.1
BAN mod AS RPC,1990	1996[13]	R3.1
BAN conc.AS RPC,1990	1996[13]	R3.1
Lowe AS RPC, 1996 [13]	No attack	None
CCITT X.509(3), 1987	1990[1]	R2.1; R3.4
BAN CCITTX.509(3), 1990[1]	No attack	None
BAN simplif. Yahalom, 1990	1994[14]	R1
Paulson's Yahalom, 2001[7]	No attack	None
Neumann Stubb., 1993	1995 [15]	R1; R3.1
SPLICE/AS, 1991	1995[16]	R2.4
DS PK, 1981	1996[17]	R2.2
AN fix DS PK, 1996 [17]	No attack	None
KSL, 1992	1996[13]	R1; R3.1
AKKA_v3, 1996	1997[3]	R2.2
WLM Auth., 1994	1997 [18]	R1
W-M Frog, 1990	1997 [19]	R1
SSH PK, 1996	1997 [3]	R2.3
Abadi fix SSH PK, 1997 [3]	No attack	None
PKMv2 IEEE802.16, 2005	2006 [10]	R3.1
Lowe W-M Frog, 1997	2008 [5]	R1; R3.1
KZ Auth., 2006	2008 [4]	R1
YRY Auth., 2004	2009 [6]	R1
LKY Auth., 2005	2007[2]	R1
NKPW mod. LKY, 2007[2]	No Attack	None
JDC mod. LKY, 2013 [8]	No attack	None
HCH Auth.& KA, 2012	2013 [9]	R1

The results of this conformance evaluation are summarized in Table IV. The second column of this table

enumerates previously published parallel session attacks on the analysed protocols and the years when the attacks were published, while the third column indicates the violated design rules.

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

V. CONCLUSION

This research work investigated why parallel session attacks can be successfully mounted against security protocols that have weaknesses in their message structure. The circumstances under which these attacks can be mounted were detailed. Based on the presented analysis a new set of design requirements that ensure resistance to parallel session attacks was proposed and presented in tables I, II and III.

An empirical study on evaluating the effectiveness of the proposed design requirements was undertaken to establish if a wide range of protocols with known weaknesses and their corrected versions observe the rules.

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 requirements are effective as:

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

Finally, if the designer of a security protocol ensures that the presented guidelines are observed, confidence that the protocol is not vulnerable to parallel session attacks is enhanced.

ACKNOWLEDGMENT

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

REFERENCES

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

[2] Nam, J., Kim, S., Park, S., Won, D.: "Security Analysis of a Nonce-Based User Authentication Scheme Using Smart Cards". IEICE Transactions Fundamentals, vol. E90-A, no. 1, pp.299-302, 2007.

[3] Abadi, M., "Explicit Communication Revisited: Two New Attacks on Authentication Protocols", IEEE Transactions on Software Engineering, vol. 23, no. 3, pp.185-186, March 1997.

[4] Xu, J., Zhu, W., Feng, D., "Improvement of a Fingerprint-Based Remote User Authentication Scheme", International Journal of Security and its Applications, vol. 2, no. 3, July, 2008.

[5] Dojen, R., Jurcut, A., Coffey, T., Györödi, C.: "On Establishing and Fixing a Parallel Session Attack in a Security Protocol", In: Intelligent Distributed Computing, Systems and Applications, Springer Berlin / Heidelberg, vol. 162, pp. 239-244, 2008.

[6] Hsiang, H.C., Shih, W.K.: "Weaknesses and improvements of the Yoon-Ryu-Yoo remote user authentication scheme using smart cards". Computer Communications, vol. 32, pp. 649-652, 2009.

[7] Paulson, L., "Relations between secrets: Two formal analyses of the yahalom protocol", Journal of Computer Security, vol. 9, no. 3, pp.197–216, 2001.

[8] Jurcut, A., Coffey, T., Dojen, R.: "Establishing and Fixing Security Protocols Weaknesses using a Logic-based Verification Tool", Journal of Communication, vol. 8, no. 11, ISSN 1796-2021, pp. 795-806, November 2013.

[9] Wang, D., Ma, C.: "Cryptanalysis of a remote user authentication scheme for mobile client–server environment based on ECC". Information Fusion, vol. 14 , pp. 498–503, 2013.

[10] Xu, S., Huang, C.: "Attacks on PKM Protocols of IEEE 802.16 and Its Later Versions", In: Proceedings of 3th International Symposium on Wireless Communication System (ISWCS), pp. 185-189, IEEE Press, Spain, 2006.

[11] Gordon, A., Jeffrey, A., "Types and effects for asymmetric cryptographic protocols", Journal of Computer Security, vpl. 12, no.3/4, pp. 435–484, 2004.

[12] Lowe, G., "An attack on the Needham-Schroeder public key authentication protocol", Information Processing Letters, vol. 56, no. 3, pp. 131–136, Nov 1995.

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

[14] Syverson, P., "A taxonomy of replay attacks", In: Proceedings of the Computer Security Foundations Workshop (CSFW97), pp. 187–191, June 1994.

[15] Hwang, T., Lee, N., Li, C., Ko, M., Chen, Y., "Two attacks on neumann-stubblebine authentication protocols", Information Processing Letters, vol. 53, pp. 103 – 107, 1995.

[16] Hwang, T., Chen, Y., "On the security of splice/as : The authentication system in wide internet", Information Processing Letters, vol. 53, pp. 97-101, 1995.

[17] Abadi, M., Needham, R., "Prudent engineering practice for cryptographic protocols", IEEE Transactions on Software Engineering, vol. 22, no.1, pp. 6-15, 1996.

[18] Clark, J., Jacob, J., "A survey of authentication protocol literature: Version 1.0", November 1997.

[19] Lowe, G., "Casper: A compiler for the analysis of security protocols," In: Proceedings of the 10th Computer Security Foundations Workshop, 1997.

[20] 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, vol.13, no.3, pp. 223–230, 2012.