On the Detection of Desynchronisation Attacks against Security Protocols that use Dynamic Shared Secrets

Ioana Lasc
Department of Electronic & Computer Engineering,
University of Limerick,
Limerick, Ireland
ioana.lasc@ul.ie

Reiner Dojen (corresponding author)
Department of Electronic & Computer Engineering,
University of Limerick,
Limerick, Ireland
reiner.dojen@ul.ie
Phone: +353-61-213442
Fax: +353-61-338176

Tom Coffey
Department of Electronic & Computer Engineering,
University of Limerick,
Limerick, Ireland
tom.coffey@ul.ie

Abstract— Many peer-to-peer security protocols in mobile communications utilise shared secrets. Synchronous storage of shared secrets is imperative for the successful operation of security protocols, as asynchronous storage of shared secrets may lead to service unavailability. Hence, update mechanisms must not only guarantee the secrecy of shared secrets, but also their synchrony.

This paper addresses synchronisation weaknesses in security protocols for wireless communications. It is demonstrated that a wide range of protocols contain such weaknesses. A new class of attack, called suppress-and-desynchronise attack, is introduced that exploit these weaknesses. These new attacks desynchronise the shared secrets of principals by suppressing messages, resulting in a permanent denial of service condition.

A verification system to model update mechanisms for shared secrets is introduced. Based on this verification system detection rules are developed that are able to detect synchronisation weaknesses that can be exploited by suppress-and-desynchronise attacks. Application of the detection rules to three security protocols results in the detection of hitherto unknown weaknesses. Consequently, these security protocols are susceptible to suppress-and-desynchronise attacks and details of mounting the attacks are presented. Finally, amendments to one of these protocols are proposed and application of the introduced formal system establishes the immunity of the amended protocol against suppress-and-desynchronise attacks.

Keywords - Authentication, Protocol verification, Wireless communication, Mobile communication systems, Denial of Service

1. Introduction

The security of electronic networks and information systems is nowadays a critical issue for the growth of information and communication technologies. This is particularly the case in the wireless environment, where the data is broadcasted over open airwaves. As these networks are often trusted with highly sensitive information, the security of both the infrastructure itself and the information that runs through it must be guaranteed. Security protocols are used to provide such protection by offering services such as authentication, key establishment, confidentiality, integrity and non-repudiation. Such security protocols need to be able to withstand threats such as replay attacks, type-flaw attacks and denial of service attacks.

In addition to security requirements, security protocols for wireless communications have to consider the restricted computational abilities and the limited power resources of the wireless/mobile devices [1], [2]. Thus, many peer-to-peer security protocols in the mobile environment utilise shared secrets to minimise the computational burden on mobile devices [3-8]. However, there are security concerns raised by the long-term storage of shared secrets [9]. Therefore, modern security protocols utilise dynamic shared secrets, which are updated to new values in each session by an underlying update mechanism [10-15]. For most applications an off-line update via smart cards or other security tokens is not feasible. Therefore, an online update mechanism is employed, where the new shared secret value is established through message exchanges between the involved principals.

Mutual authentication based on such dynamic shared secrets is performed by proving ownership of the current instance of the shared secret [7], [13]. Additionally, these dynamic shared secrets may also be used:

- as fresh components to protect against replay attacks and in the generation of session keys [3], [8], [12], [14].
- to act as non-related aliases to mobile users while roaming in foreign domains with the purpose of preserving user privacy [15], [16].
- in the creation of evidence of service access to provide non-repudiation in billing protocols [5], [17].

Online update mechanisms for dynamic shared secrets must not only ensure the secrecy of the shared secrets, but must also guarantee their synchrony.

1.1 Original Contribution of this Work

In this paper we reveal a new weakness in the update mechanisms of current security protocols that utilise dynamic shared secrets. It will be demonstrated that this weakness is inherent in a wide range of protocols such as the Aziz-Diffie (AD) [3], Hwang-Yang-Shiu (HYS) [12], Chang-Chang (CC) [13], Tseng [14] and Chen-Lee-Chen (CLC) [15] protocols.

Further, a new form of attack – termed in this paper as suppress-and-desynchronise attack (SD attack) – that exploits this weakness is presented. In an SD attack, the intruder interferes with the delivery or integrity of messages to cause failure of the update mechanism. In many cases, the SD attack is mounted by suppressing a single message between the communicating parties. As a result, the attacked security protocol is compromised and further communication between the involved parties is no longer possible. Thus, if the security protocol is not designed to deal with the possibility of a failed update mechanism, a permanent denial of service condition is reached.

This paper also proposes a new verification system that is able to model update mechanisms for shared secrets. Further, the verification system is able to detect update mechanisms that are susceptible to SD attacks.

To demonstrate the effectiveness of the verification system, it is used to establish the presence of hitherto unknown weaknesses in the AD [3], HYS [12] and Tseng [14] security protocols. It is demonstrated how an attacker can exploit the detected weaknesses by mounting SD attacks against these protocols. In each case, executing the SD attack results in a permanent DoS condition. Finally, amendments to the AD protocol are proposed and the formal system is used to prove the immunity of the amended AD protocol against SD attacks.

2. New Attacks against Security Protocols Implementing Update Mechanisms

Providing mutual authentication based on shared secrets is a common feature in security protocols. The communicating parties involved in a protocol session prove their identity by showing possession of the shared secrets as follows: Two principals initially establish a shared secret θ and store it in their memory. During one protocol run (we use the terms “protocol run”, “protocol session” and “protocol iteration” synonymously) the communicating parties challenge each other to prove possession of the shared secret θ by sending a message based on θ. If both principals are able to formulate the expected messages, mutual authentication is successful. If either principal is incapable of producing the correct message, mutual authentication fails.

Authentication based on static shared secrets, which are not updated online during a session, implies their long term usage. However, there are potential vulnerabilities associated with the long term storage of static shared secrets [9], such as disclosure of past and current session keys and identity disclosure [16]. Dynamic shared secrets, which are updated online, can be used to avoid potential vulnerabilities associated with static shared secrets [13], [15].

2.1 Authentication Based on Dynamic Shared Secrets

Security protocols that implement dynamic shared secrets employ an underlying online mechanism to update these shared secrets to new values. Thus, a sequence of shared secrets θ_1 to θ_n is used, where a successful run of a protocol ensures mutual authentication of both principals by proving possession of the current shared secret θ_i (see Figure 1).

The initial shared secret θ_1 is usually established in an offline process. In the first protocol run, principals mutually authenticate each other using θ_1. At the same time, the update phase of the protocol provides the principals with θ_2, the next instance of the shared secret. In general, the protocol run that uses θ_i also establishes the next shared secret θ_{i+1}. This new shared secret θ_{i+1} will be used in the subsequent protocol run, which also updates to the next shared secret θ_{i+2} and so on. Subsequent shared secrets can be either unrelated, i.e. be generated randomly [15], or can be created from the same seed [17]. In the latter case there is a functional relationship between consecutive values and perfect forward secrecy needs to be addressed [13], to avoid a situation where an attacker can predict future shared secret values.

An authentication process based on dynamic shared secrets requires the synchrony of the shared secret instances: both parties should store the same instance of the shared secret at the end of the protocol run and use the same instance of the secret in the subsequent protocol run to authenticate each other. Thus, in a successful online update mechanism the storage of the shared secrets by the two principals evolves synchronously after each protocol run.

2.2 General Structure of Suppress-and-Desynchronise Attacks

For the successful operation of security protocols utilising dynamic shared secrets the synchronous storage of these shared secrets is required. If an update mechanism for dynamic shared secrets fails, then the principals store a different instance of the shared secret. As will be demonstrated below, an attacker with the capabilities of the Dolev-Yao attacker model [18] can force such a condition.

For example, consider the situation detailed in Figure 2: In protocol run i the update mechanism fails, where only Principal 2 has updated to the new shared secret θ_{i+1} and the two principals end the protocol run with different values of the shared secret. The next authentication attempt by Principal 1 will be based upon the shared secret θ_i as this is Principal 1's current shared secret. However, as Principal 2 has updated to the new secret θ_{i+1}, Principal 1's request will be rejected as it does not contain the expected shared secret instance θ_{i+1}. Consequently, mutual authentication will fail and the requested service will be denied. Further, if the security protocol is not designed to deal with the possibility of a failed update mechanism, a permanent denial of service (DoS) condition is reached.

2.3 Mounting a Suppress-and-Desynchronise Attack against a Mutual Authentication Protocol for Mobile Satellite Communications

This section demonstrates the mounting of a Suppress-and-Desynchronise Attack (SD attack) against the CLC mutual authentication protocol for mobile satellite communications. The exploited weakness has been identified by the authors of this paper and an improved protocol has been proposed [10]. A similar attack is demonstrated against the CC protocol in [11].

2.3.1 The CLC Mutual Authentication Protocol

The CLC mutual authentication protocol [15] is designed to provide mutual authentication and key establishment between a mobile user (U) and the network control centre (NCC) of a satellite network within a LEO satellite communication system. The CLC protocol has three phases: initialisation phase, mobile user registration phase and mobile user authentication phase.

In the initialisation phase the NCC creates its pair of private/public keys. Further, during the registration phase the following shared secrets are established between U and NCC: U's temporary identity T_{ID} and the session key sk, which is computed using T_{ID}. These credentials (T_{ID}, sk) are used once to perform the first authentication phase by U and NCC. Subsequently, the authentication phase is performed each time prior to communication between U and NCC with the following goals:

- The NCC generates a new value for U's temporary identity (T_{IDnew}) and sends it to U
- Both U and NCC derive the new session key sk_{new} from U's new temporary identity T_{IDnew}.

2.3.2 Mounting a SD Attack against the CLC Protocol

In these attacks it is assumed that the attacker has the capability of the Dolev-Yao attacker model [18]. Such an attacker can mount a SD attack by preventing the last message of the authentication phase from reaching U and thus cause failure of the update mechanism for U as outlined in Figure 3.

As discussed in [10], this attack can be mounted by using a low-power jammer to suppress delivery of message 4. This results in a situation where the NCC has already updated its shared secret to T_{IDnew} and implicitly the new session key sk_{new} computed on T_{IDnew}, whereas

U still stores the old shared secrets T_{ID}. Hence, while NCC has authenticated U, the opposite is not true: U has not yet authenticated NCC. Thus, eventually U will resend the same authentication request. However, at this point in time the shared secrets of U and NCC are desynchronised: U attempts to authenticate with T_{ID} and the obsolete sk, while NCC expects the new shared secrets T_{IDnew} and sk_{new}. Consequently, this authentication request will fail. Furthermore, as the CLC protocol is not designed to deal with failed update mechanisms the satellite service becomes permanently unavailable as the user is not enabled to re-authenticate.

3. A New Verification System to Model Shared Secret Update Mechanisms

The formal verification of security protocols is an imperative step in the design of security protocols [2],[19], which aims at proving that the verified protocol meets its security goals and demonstrate the absence of mountable attacks against the protocol. The approach presented in this work is aiming at demonstrating the absence of mountable SD attacks. In this section we introduce a new formal verification system to model update mechanisms for dynamic shared secrets, which expresses the actions of principals, their storing strategies for shared secrets and the principals' roles in the different types of update mechanisms. The verification system can be used to establish the resistance of security protocols against SD attacks. Table 1 summarises the notations introduced in this section.

3.1 A Classification of Update Mechanisms for Dynamic Shared Secrets in Authentication Protocols

In general, authentication protocols that use dynamic shared secrets start with an initialisation or registration phase to establish the initial shared secrets. Subsequently, they loop through an authentication or update phase followed by a data exchange phase. While the details of the update mechanisms vary greatly, some common structures emerge. We classify the update mechanism into the following two categories:

- Update Mechanism A (UM-A): Establishes a new and unpredictable secret during each authentication/update session.
- Update Mechanism B (UM-B): Establishes a set of N pre-determined shared secrets. After N iterations a new set of shared secrets is established. The N values within each set are either:
  - derived through a formula from the same seed or
  - pre-calculated random and stored values

3.1.1 Update Mechanism UM-A

In a protocol that uses Update Mechanism A (for example [3], [12], [14], [15]) at any given time only a single instance of the shared secret is generated (cf. Figure 4). During the initialisation/registration phase both communicating principals establish the initial instance θ_1 of the shared secret. In every protocol run authentication is based upon the current value θ_i of the shared secret and a new instance of the shared secret (θ_{i+1}) is generated and distributed. Both principals update their current shared secret to θ_{i+1} for use in the next protocol run.

3.1.2 Update Mechanism UM-B

The second update mechanism (UM-B) works with sets $\theta_{i,j}$ of N pre-determined shared secrets (for example [13], [17]). Such a set (also called a chain) is used for N consecutive authentications, followed by the establishment of a new set (cf. Figure 5). During the initialisation/registration phase both communicating principals establish the initial set $\theta_{1,j}$ of shared secrets. The values within each set can either be pre-calculated random values or derived from the same seed by some formula. In each protocol run authentication is based upon the current value $\theta_{i,j}$ of the shared secret and both principals update to the new shared secret $\theta_{i,j+1}$. After N protocol runs, all values in the current set have been used and both principals enter an update phase that establishes a new set $\theta_{i+1,j}$ of shared secrets.

3.2 Principal Classification by Roles, Visibility Scopes and Operating Values

The principals are classified by their roles in the update mechanism and by their visibility scope. In an update mechanism, principals take on either the role of initiator or closer. Additionally, one principal takes also the role as generator. These roles are defined as follows:

- *Initiator* (α) – The principal that issues the first message in the update mechanism
- *Closer* (ω) – The principal that issues the last message in a two-message update mechanism, or the principal that receives the message in a one-message update mechanism.
- *Generator* (Γ) – The principal that generates the new secret value(s)

The visibility scope formally expresses the amount of shared secret instances that a principal is able to access either by storing them or through computation. Principals can be classified by their visibility scopes as follows:

- A *Unary* principal is designed to only access the current shared secret.
- A *Binary* principal is designed to access both the current shared secret and the previous or subsequent one.
- An *n-ary* principal is designed to access a chain of n shared secrets.

For example, Figure 6 shows the visibility scope of two principals in a UM-A, where both, the Initiator (α) and the Closer (ω), store only a single instance of the shared secret θ. Thus, both principals have at any given time only access to a single instance of the shared secret and, therefore, they are both Unary principals.

In a UM-B at least one principal can access multiple instances of θ as depicted in Figure 7. The principal able to access multiple instances can either be the Closer (as shown in the figure), the Initiator or both principals. In the depicted example, the Initiator (α) is a Unary principal, while the Closer (ω) is an n-ary principal.

For some protocols using a UM-B the Generator (Γ) is designed as a Unary principal. In these cases, even if the principal initially generated N instances of θ, it is implemented to be unable to access all of these instances at any given time. Instead, the principal computes or stores only one instance of θ corresponding to the protocol session.

Two visibility scopes are called *disjunctive* if they share no common value of the shared secret θ.

A principal's *operating values* of the shared secret θ are those values that the principal is actively using in one protocol run to either prove its identity when issuing a message or to validate incoming messages against. A principal's operating values are always a subset (less or equal) of its visibility scope.

Principals can be classified by the number of operating values as follows:

- A *Monostable* principal has only a single operating value. Thus, the principal operates only with the current secret value used in the update mechanism. A generic Monostable principal is denoted M. The notation can also be used to indicate the visibility scope, where M^1 indicates a Unary visibility scope, M^2 indicates a Binary visibility scope and M^n indicates an n-ary visibility scope.

- A *Bistable* principal uses two operating values. Thus, the principal operates with both the current shared secret and the previous or the subsequent one. A Bistable principal is denoted B. A Bistable principal can only have a Binary visibility scope (B^2) or an n-ary visibility scope (B^n).
- An *N-Stable* principal uses more than two operating values – in general, all remaining values in the visibility scope are used as operating values. An N-Stable principal is denoted N and can only have an n-ary visibility scope (N^n).

3.3 The Asynchronous Interval

In the verification system a protocol session regards the update mechanism as an atomic unit. However, this update mechanism is a sequential process involving five steps:

1. One principal (A) updates the shared secret first from θ_i to θ_{i+1}.
2. A computes the message containing the new operating value θ_{i+1}.
3. A sends the message to the other principal (B).
4. B receives the message from A.
5. On successful authentication of A, B updates its shared secret to θ_{i+1}.

The time interval from step 2 until step 5 is called the *Asynchronous Interval*, as during this time one principal has updated the shared secret while the other is still storing the old value.

3.4 Initiator’s Evaluation of the Authentication Results

Update mechanisms for dynamic shared secrets can be employed by security protocols as a stand-alone authentication phase or as an embedded process in an exchange of multiple messages. The verification system proposed in this section identifies and models only the messages that constitute the update mechanism. In peer-to-peer authentication systems two messages are sufficient to provide the update mechanism (cf. the Two General’s Problem [21]), therefore the verification systems only needs to model update mechanisms with up to two messages. This section outlines how the Initiator evaluates the results of the authentication session.

3.4.1 Two-Message Update Mechanisms

In a two-message update mechanism the Initiator (α) receives a notification of the Closer’s authentication acceptance. If the Initiator receives an “access denied” message or when the legitimate authentication request times out, the Initiator assumes the authentication session has failed. If the Initiator receives an “access granted” message, the authentication session has been successful.

3.4.2 One-Message Update Mechanisms

In this case the Closer (ω) is a silent principal that doesn’t issue any message as part of the update mechanism. Hence, the Initiator (α) establishes the outcome of the authentication session in the subsequent data exchange phase. If the Initiator’s attempt to enter data exchange with its current session key is rejected by the Closer, the Initiator assumes the authentication session has failed. Otherwise, the Initiator knows the authentication session has been successful.

3.5 Strategy of Operating Value Usage

The principals may use any of the following strategies to update their operating value in case of failed authentication sessions.

3.5.1 Strategy of Operating Value Usage in Two-Message Update Mechanism

After a failed authentication session an Initiator can issue the next authentication request according to the following strategies:

- A *Repeater* (Ξ) will continue to issue requests with the current operating value (which hasn't been updated due to authentication failure) until the authentication session is successful.
- An *Increaser* (Ψ) will always use the next operating value (θ_{i+1}) in a subsequent authentication attempt. Thus, an Increaser will never resend a request with the current operating value (θ_i). It is assumed that the Increaser tries to authenticate only once with the next operating value (θ_{i+1}) after a failed session. If this second authentication request also fails, then the Increaser will keep its current operating value (θ_{i+1}).
- *Differentiator* (Δ) – It can be Binary Bistable (B^2) or n-ary Bistable (B^n) or n-ary N-Stable (N^n). A Differentiator will resend its request according to any one of the following types:
    - Issues next value – forward differentiator: Δ^+
    - Issues previous value – backward differentiator: $^-\Delta$
    - Issues arbitrary value in the chain – random differentiator: $^-\Delta^+$ - if it is N^n, depending on what it just received in the resynchronisation challenge
    - Issues all of the above: Δ – used as a wildcard for all three types of differentiator. This is used later to minimise configuration tables in section 4.

After a successful authentication session, where the current operating value has been updated, an Initiator always acts as a Repeater.

The Closer only updates its operating value in response to incoming requests from the Initiator and updates the shared secret according to the following strategies:

- A *Repeater* (Ξ) issues an "access granted" message and updates its operating value if the request is accepted. Alternatively, on receiving an invalid request, it issues an "access denied" message and keeps the current operating value.
- A *Differentiator* (Δ) differentiates between its responses as a function of the incoming messages. The same types as for the Initiator are possible.

3.5.2 Strategy of Operating Value Usage in One-Message Update Mechanism

An Initiator can issue the next authentication request according to the following strategies:

- An *Unconditional Increaser* (Θ) will always use the next operating value in a subsequent authentication attempt, even in case of previously failed authentication sessions.
- A *Differentiator* (Δ) can act differently based on the result of authentication sessions. Further, a Differentiator can potentially use any operating value in its visibility scope. The same types as for the two message update mechanism are possible.

The Closer updates its operating value analogous to the two-message update mechanism, but does not issue any messages.

3.6 Update Mechanism Session Identification

In some cases it is necessary to identify a single session or a range of sessions. A single session is identified by σ (p), while a range of sessions is identified by σ (p,q). The notation p identifies session number "p", while the pair p,q expresses: each session from p to q.

3.7 Reset Phases in Protocols using Update Mechanisms

In addition to the update mechanism, some protocols also include a reset phase that re-initialises the shared secret of the communicating parties [5], [6], [7], [20]. In such a phase, the parties authenticate each other using additional secrets and thus, this phase can be performed even by communicating parties with different operating values. In the verification system the presence of such a reset phase is indicated by the notation σ.ρ, while the absence of such a phase is denoted σ.Φ.

3.8 Impact of the SD attack

The SD attack against a security scheme has the following consequences:

- One principal is unaware of the other principal's status on the update event
- The update mechanism finishes asynchronously and the visibility scope of the principals becomes disjunctive.

The party that is prevented by the attack from updating to the new instance of the shared secret is called the desynchronised party and is denoted by δ, e.g. δα indicates that the initiator α has been prevented from updating to the new operating value.

3.9 Direction of SD attacks

The following symbols are required to model the direction of an attack:

- $\overset{\longrightarrow\!|}{\dashleftarrow\dashleftarrow}$ symbolises an attack that stops the first message in a two-message update mechanism
- $\overset{\longrightarrow}{|\!\longleftarrow}$ symbolises an attack that stops the second message in a two-message update mechanism
- $\longrightarrow\!|$ symbolises an attack that stops the message in a one-message update mechanism

4. Detection Rules of the Verification System

This section introduces a set of formal rules to detect weak update mechanisms that are susceptible to SD attacks. The formal rules are introduced firstly for one-message update mechanisms and secondly for two-message update mechanisms. Each rule establishes the presence of a weakness in an update mechanism and indicates the corresponding type of attack that inflicts asynchronous operating values for the communicating parties. These rules were developed by generating a complete set of all possible configurations of combinations of Initiators and Closers with all their possible characteristics and then analyzing these configurations to evaluate the presence of SD attacks in them. As the set is exhaustive, the resulting verification system is complete and therefore it is able to detect all possible SD attacks.

4.1 Update Mechanisms with Change of Generator Role

In some UM-Bs the role of generator changes over time. While the role of Generator is played by one party during the N-1 sessions, it is played by the opposite party in the Nth session. In these cases, the update mechanism must be expressed twice: once for each principal as generator. Consequently, two detection rules are applicable: one to model the update mechanism performed by the first N-1 sessions and the second to model the Nth session.

4.2 One-Message Update Mechanisms

One-message update mechanisms can exist as stand-alone update phases [5], [7] or as an underlying process in other protocols [17]. Two distinct cases are possible for one-message update mechanisms:

1. The role of the Generator is played by the Initiator (Γ.α).
2. The role of Generator is played by the Closer (Γ.ω).

4.2.1 The Closer as a Generator – $\omega.\Gamma$

In a one-message update mechanism the Closer can only be a Generator if it is a UM-B. As the Initiator is not the Generator, it requires a chain of N instances of shared secrets to be delivered to it in advance. This enforces that the Initiator is an n-ary principal. Additionally, the presented rules assume that the chain is updated through a two-message update mechanism, where ω responds to α's challenge computed with the Nth operating value. Thus, these rules concern only the first N-1 iterations of the update mechanism (σ (1, N-1)).

When a protocol implements a reset phase to update the chain, all N instances of the shared secret within a chain may be used in the one-message update mechanism. In this case, the number of sessions that these rules apply to is then increased by one and σ (1, N-1) changes to σ (1, N). Table 2 shows the complete set of possible configurations for a one-message update mechanism where the Closer is the Generator. Each configuration is labelled as follows:

- I for impossible configurations
- N for configurations that make no sense in practice
- A for configuration vulnerable to a suppress attack
- S for safe configurations

As can be seen, there are two configurations that are marked as vulnerable to attack: configurations number 20 and 34. In configuration number 20, the Initiator (α) is an n-ary Monostable principal (M^n) and the Closer (ω) is a Unary Monostable principal (M^1). The Initiator acts as an Unconditional Increaser (Θ) and the Closer is a Repeater (Ξ). Thus, each time α enters a new session of the protocol during the lifetime of a chain, it sends a request to ω using its operating value θ_i from the chain. Simultaneously, as an Unconditional Increaser, α also updates its operating value to θ_{i+1}. On receiving the request, ω validates θ_i against its own operating value and if they match, α is authenticated and ω also updates its operating value to θ_{i+1}. As ω only updates on reception of messages from α, the asynchronous interval of this update mechanism begins with computation of α's request and ends on successful authentication of α by ω after reception of α's request. With the assumption that no reset phase is present, an attacker can desynchronise α and ω in the first N-1 iterations in each chain by preventing α's request from reaching ω. As now α's operating value is updated to θ_{i+1} while ω's operating value is still θ_i, any future authentication request from α will fail and a permanent DoS condition is reached.

The rule "DR1) $\alpha.M^n \wedge \alpha.\Theta \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \longrightarrow\!| \equiv \delta\omega$" detects the described SD attack against any of the first N-1 iterations (σ (1, N-1)) of an update mechanism using configuration number 20. This rule only applies to protocols that do not implement a reset phase ($\sigma.\Phi$). By stopping the first message ($\longrightarrow\!|$), the attacker prevents ω from updating to the new operating value and thereby desynchronises ω ($\delta\omega$). As a consequence, α and ω have different operating values.

Correspondingly, the detection rule based upon unsafe configuration number 34 is: "DR2) $\alpha.M^n \wedge \alpha.\Theta \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \longrightarrow\!| \equiv \delta\omega$". The detection rules for one-message update mechanisms with the Initiator as Generator are summarised in Table 3.

4.2.2 The Initiator as a Generator – $\alpha.\Gamma$

Analogous to the previous section, a complete set of configurations was generated and analysed for one-message update mechanisms with the Initiator as Generator. Table 4 summarises the configurations found vulnerable to desynchronisation attacks.

Configuration number 1 in Table 4 refers to a one-message UM-A, where both the Initiator (α) and the Closer (ω) are Unary Monostable (M^1) principals. Here the Initiator acts as an Unconditional Increaser (Θ) and the Closer as a Repeater (Ξ). On each authentication request α sends its current operating value θ_i and the next proposed operating value θ_{i+1}. As an unconditional increaser, α also updates its operating value to θ_{i+1} before sending the authentication request. On receiving the request, ω authenticates α on θ_i. On successful authentication ω updates its operating value to θ_{i+1}.

As α is the generator, θ_{i+1} is unknown to ω until it receives α's request. Thus, the asynchronous interval in this mechanism is between computation of the message issued by α and ω's successful authentication of α at reception of α's request. With the assumption that no reset phase is present, an attacker can desynchronise α and ω by preventing α's request from reaching ω. Now α's operating value is updated to θ_{i+1}, while ω's operating value is still θ_i. Thus, any future authentication request from α will fail and a permanent DoS condition is reached.

The rule "DR3) $\alpha.M^1 \wedge \alpha.\Theta \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi \wedge \sigma.\Phi \wedge \longrightarrow\!| \equiv \delta\omega$" detects the described SD attack against an UM-A using configuration number 1. Rule DR3 only applies if the protocol does not implement a reset phase ($\sigma.\Phi$). By stopping α's message ($\longrightarrow\!|$) from reaching ω, an attacker prevents ω from updating to the new operating value. The Closer becomes the desynchronised principal ($\delta\omega$). Table 5 summarises the detection rules obtained from the configurations in Table 4. Depending on the chain update policy, rules may be extended to N sessions (cf. The Closer as a Generator – $\omega.\Gamma$).

4.3 Two-Message Update Mechanisms

Two message update mechanisms are the most common form and are usually implemented as standalone protocol phases [8], [20].

4.3.1 The Closer as a Generator – $\omega.\Gamma$

This scenario considers the types of update mechanisms in which the Closer (ω) is the principal that generates the new instances of θ. Thus, the Initiator (α) is provided with future instances of θ by ω at receipt of message two in the update mechanism. In this scenario α cannot be a Unary Monostable Increaser ($\alpha.M^1 \wedge \alpha.\Psi$) as it has no knowledge of future θ instances. From the complete set of configurations for this scenario, the vulnerable configurations are summarized in Table 6.

Configuration number 4 in Table 6 refers to a UM-B, where the Initiator (α) is an n-ary Monostable and acts as an Increaser (Ψ). The Closer (ω) is a Unary Monostable (M^1) principal, a Repeater (Ξ) and the Generator (Γ). Even though the Closer has initially generated N instances of θ, it is implemented to be unable to access all of these instances at any given time. Instead, the principal computes or stores only one instance of θ corresponding to the protocol session.

The Initiator α sends the first message of the update mechanism, which contains its operating value θ_i. On receipt of this message ω validates θ_i against its own operating value. If they match, then ω generates a new instance θ_{i+1} and sends its response to α. This response contains both θ_i and θ_{i+1}, where θ_i is used to authenticate ω and θ_{i+1} indicates the new operating value to which α should update to. When α receives the response, it will assess ω's identity by comparison of the received θ_i against its stored value. On successful authentication of ω, α updates its operating value to θ_{i+1}.

As ω is a Unary Monostable principal, θ_{i+1} is unknown to ω until it receives α's request. Thus, the asynchronous interval in this mechanism is between computation of the message issued by α and ω's successful authentication of α at reception of α's request. With the assumption that no reset phase is present, an attacker can desynchronise α and ω by preventing α's request from reaching ω. As the Initiator is an Increaser, when entering the next authentication session α's operating value is θ_{i+1} while ω's operating value is still θ_i, any future authentication request from α will fail and a permanent DoS condition is reached.

The rule "DR12) $\alpha.M^n \wedge \alpha.\Psi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\ (1, N\text{-}1) \wedge \sigma.\Phi \wedge \substack{\rightarrow\!| \\ \dashleftarrow} \equiv \delta\omega$" detects a desynchronisation attack against an UM-B using configuration number 4 of Table 6. Rule DR12 only applies to the first N-1 iterations (σ (1, N-1)) if no reset phase is present ($\sigma.\Phi$). By stopping α's request ($\substack{\rightarrow\!| \\ \dashleftarrow}$) from reaching ω, an attacker prevents ω from updating to the new operating value. The Closer becomes the desynchronised principal ($\delta\omega$). Against the setup modelled by rule DR12, the attack ($\substack{\rightarrow\!| \\ \dashleftarrow}$) has an impact only against the first N-1 sessions, before the principals update to a new chain of secrets.

Against the Nth run (σ (N)), which additionally provides an update to a new chain, only an attack that stops the second message ($\substack{\rightarrow \\ |\!\leftarrow}$) desynchronises the principals. This attack prevents α from updating to the first instance of θ or the seed of the new chain and therefore α becomes a desynchronised principal ($\delta\alpha$). The following rule detects this weakness in an update mechanism: "DR13) $\alpha.M^n \wedge \alpha.\Psi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\ (N) \wedge \sigma.\Phi \wedge \substack{\rightarrow \\ |\!\leftarrow} \equiv \delta\alpha$". Table 7 summarises the detection rules obtained from the configurations in Table 6. These rules allow a change of generator in the Nth iteration. If the role of generator does not change, then rules limited to N-1 sessions may be extended to N sessions by replacing σ(1, N-1) with σ(1, N).

4.3.2 The Initiator as a Generator – $\alpha.\Gamma$

When the Initiator (α) is the Generator (Γ), then instances of θ are delivered in the form of authentication requests to the Closer (ω) and the confirmation of ω's update event is presented to α in the second message. From the complete set of configurations for this scenario, the vulnerable configurations are summarized in Table 8.

Configuration number 1 in table 8 models a UM-B, where the Initiator (α) and the Closer (ω) are Unary Monostable (M^1) principals and Repeaters (Ξ). Each time α enters a new session of the protocol during the lifetime of a chain, it sends a request to ω using its operating value θ_i. On receiving the request, ω validates θ_i against its own operating value and if they match α is authenticated and ω updates its operating value to θ_{i+1}. As this is a two-message update mechanism, α only updates its operating value on receipt of the response from ω. Hence, the asynchronous interval in this update mechanism begins with ω's successful authentication of α (ω updates to θ_{i+1}) and it ends with α's successful authentication of ω.

With the assumption that no reset phase is present, an attacker can mount a SD attack. Stopping the second message from reaching α will desynchronise α, while ω has already updated to θ_{i+1}, α's operating value is still θ_i. Thus, any future authentication requests from α will fail and a permanent DoS condition is reached.

The rule “DR18) $\alpha.M^1 \wedge \alpha.\Xi \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{|\!\!\longleftarrow} \equiv \delta\alpha$” detects desynchronisation attacks against update mechanisms using configuration number 1 of Table 8. This rule only applies if no reset phase is present. By stopping the second message ($\overset{\longrightarrow}{|\!\!\longleftarrow}$) in the update mechanism, α is prevented from updating and it therefore becomes a desynchronised principal (δα). Table 9 summarises the detection rules obtained from the configurations in Table 8. If the role of generator does not change, rules limited to N-1 sessions may be extended to N sessions by replacing σ(1, N-1) with σ(1, N).

5. Applying the New Verification System to Wireless Protocols

This section demonstrates the application of the new verification system by applying it against the following protocols:

- Authentication scheme for mobile satellite communications (HYS protocol) [12].
- Public-key management scheme for mobile ad hoc networks (Tseng protocol) [14].
- Mutual authentication protocol for wireless communications (AD protocol) [3].

The performed verifications reveal hitherto unknown weaknesses in the HYS, Tseng and AD security protocols. In these verifications the update mechanisms of the protocols are formalised and the introduced detection rules are applied to detect the presence of weaknesses in the protocol. In addition, it is demonstrated how an attacker can cause a permanent DoS condition by mounting SD attacks against these protocols. Also, amendments to the AD protocol are proposed and application of the verification system to the amended AD protocol establishes its immunity against SD attacks.

5.1 Guidelines to Update Mechanism Formalisation and Application of Detection Rules

In order to use the proposed verification system and apply the detection rules, the update mechanism under consideration must first be formalized. The following steps are required to extract an update mechanism from a security protocol and model it using our formal verification system:

1) Identification of the main protocol phases, such as initialisation phase, authentication phase, key update phase etc. Attention must be given to potential hidden phases - the evolution of shared secrets must be observed throughout the flow of the entire security protocol.
2) For each phase identified in 1) identify:
    a) initialisation point for each new instance of any secret,
    b) first time usage point for each secret used for authentication purposes,
    c) re-instantiation points, where any secret data is updated.
3) Identification of update mechanism(s) inside the security scheme - a security scheme can implement:
    - One update mechanism as a standalone protocol phase.
    - One update mechanism embedded in a protocol phase comprised of more than two messages.
    - Two or more sequential update mechanisms, where one UM completes before commencement of the other.
    - Two or more interleaving update mechanisms, where subsequent UMs commence before previous UMs have completed.
4) For each update mechanism identified in 3) isolate the update mechanism by removing relaying principals.
5) For each isolated update mechanism in 4):

a) Identify update mechanism as of type UM-A or UM-B

b) Identification of θ.

c) Identification of number of messages.

d) Establishment of existence of a reset phase among the identified protocol phases in 1).

e) Identification of the Generator of θ.

f) Identification of α and ω's visibility scopes and number of used operating values.

g) Identify Initiator and closer as Repeater, Increaser or Differentiator. Where the behaviour cannot be identified from the description of the protocol, all cases should be considered.

6) For any identified UM-B:

a) Isolation of the round of the update mechanism that ensures generation and update to a new chain of secrets (identified by the verification system as the Nth phase of a UM-B)

b) Establishment of Generator in the Nth session, which may be different than the principal that played this role throughout the initial N-1 sessions

- When the same principal is the Generator in all N rounds, the same configuration is used for all N rounds.
- If a different principal plays the role of Generator, the Nth session is modelled as a UM-A with a different configuration than the first N-1 sessions.

7) The detection rules are applied

- If a detection rule matches, a SD attack is possible against the protocol.
- If no detection rule matches, the protocol is safe against SD attacks.

5.2 Analysis of the HYS Protocol

The HYS mutual authentication protocol [12] is designed to provide mutual authentication and key establishment between a mobile user (MS) and the network control centre (NCC) of a satellite network within a LEO satellite communication system. In this protocol the LEO satellite has the functionality of forwarding messages from mobile users to the NCC and vice versa. The HYS protocol has two phases: mobile user registration phase and mobile user authentication phase. During the registration phase, the gateway assigns the new mobile user (MS) a permanent identity U_{ID}, a temporary identity T_{ID} and a secret key Kmd. Before communicating, MS and NCC must execute the authentication phase, which performs mutual authentication and the update to a new shared secret as outlined in Figure 8.

5.2.1 Applying Detection Rules for SD attacks to the HYS Protocol

The Registration Phase of the HYS protocol is executed once when the mobile user enters the system for the first time. The authentication phase is a two-message update mechanism UM-A, as at any given time only a single instance of the shared secret is generated. The update mechanism is performed by MS as an Initiator and the NCC as the Closer. The LEO only relays the shared secret θ formed by the pair (T_{ID}, Kmd) that is refreshed with each session. Therefore, the messages are regarded as directly exchanged by MS and NCC. The NCC plays the role of the Generator. Both parties operate with one operating value (Monostable principals) and both parties have a unary visibility scope. The registration phase, which provides the initially synchronous θ for both parties, is executed only once. Subsequently, the authentication phase is performed repetitively and it is based upon θ. The protocol does not implement a reset phase for θ.

As the Initiator (MS) is Unary Monostable and not the Generator, it has only one option in case of an occurring time-out: resending the authentication request to the NCC with its current θ. Thus, the Initiator is a Repeater.

The following configuration is obtained for the UM-A update mechanism executed by MS (α) and NCC (ω): “$\alpha.M^1 \wedge \alpha.\Xi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$”. As no reset phase is presence, rule DR7 ($\alpha.M^1 \wedge \alpha.\Xi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma.\Phi \wedge \underset{\mid\!\longleftarrow}{\longrightarrow} \equiv \delta\alpha$) matches. Consequently, a desynchronisation attack mounted against this protocol will cause the MS to desynchronise and lead to a permanent DoS condition for MS.

5.2.2 A Desynchronisation Attack against the HYS Protocol

The detected weakness can be exploited by intercepting the response message from the NCC to MS. This attack against the HYS protocol desynchronises the Initiator: Initially, MS issues an authentication request based on its current θ_i constituted by the pair (T_{ID}, K_{md}). After successfully authenticating MS, the NCC updates to a new $\theta_{i+1} =$ (T_{ID}’, K_{md}’). Further, NCC responds with the second message that aims to provide MS with the new θ_{i+1}. The attacker prevents NCC’s response message from reaching MS (cf. Figure 9). Consequently, the NCC updates to the new θ_{i+1}, while MS does not. As MS acts as a Repeater, it will use its current operating value θ_i in future authentication requests. However, as NCC has updated to θ_{i+1}, the received θ_i is obsolete and the authentication request is denied.

5.3 Analysis of the Tseng Protocol

The Tseng protocol [14] was proposed to provide nodes within a mobile ad hoc network with certificates. The protocol implements two certificate distribution phases built on the same principle: the first phase enables a cluster head (CH) to obtain a certificate by using the Wide-Covered Heterogeneous Network (WCN-AS) as outlined in Figure 10. The second phase enables a general node (GN) to obtain a certificate from the corresponding server (S) through its cluster head (CH).

5.3.1 Applying Detection Rules for SD attacks to the Tseng Protocol

In the first phase a Cluster Head (CH) without certificate in the network performs the ticket acquisition phase once based on an initial password PW_{HID}. In subsequent certificate acquisition operations, both CH and WCN-AS replace this password with a nonce R_{HID}. This scheme does not implement a reset phase for its shared secret $\theta = R_{HID}$.

The first phase is a two-message update mechanism UM-A, as at any given time only a single instance of the shared secret is generated. In this phase, the CH is the Initiator and the WCN-AS is the Closer. The Initiator is also the Generator of R_{HID}. Only one instance of this nonce is generated by CH and stored by WCN-AS, therefore both the Initiator and the Closer are Unary Monostable principals. As the Initiator’s behaviour in case of time-out is not specified by the protocol, both Repeater and Increaser are considered. The following UM-A configuration is obtained for the Tseng protocol: “$\alpha.M^1 \wedge \alpha.\Xi \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi$”. As no reset phase is present, rules DR18 ($\alpha.M^1 \wedge \alpha.\Xi \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi \wedge \sigma.\Phi \wedge \underset{\mid\!\longleftarrow}{\longrightarrow} \equiv \delta\alpha$) and DR19 ($\alpha.M^1 \wedge \alpha.\Psi \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi \wedge \sigma.\Phi \wedge \underset{\dashleftarrow}{\longrightarrow\!\mid} \equiv \delta\omega$) do match. Consequently, there are two effective desynchronisation attacks against this phase: The first attack desynchronises the CH, while the second attack desynchronises WCN-AS. In both cases a permanent DoS condition is reached.

The second phase proposed for certificate acquisition for a general node uses the same update mechanism with the distinction that the Initiator and the Closer are played by GN and S and the other two parties (CH and WCN-AS) just relay θ. Thus, detection rules DR18 and DR19 do match and the corresponding attacks can be mounted.

5.3.2 Desynchronisation Attacks against the Tseng Protocol

In each phase of the Tseng protocol a weakness has been revealed by the above analysis. As the type of the Initiator is not clearly identified by the protocol, the following considers two cases: firstly, the Initiator as a Repeater and secondly, the Initiator as an Increaser. Under the assumption that the Initiator acts as a Repeater, a desynchronisation attack can be mounted by intercepting the response message as shown in Figure 11.

Initially, the Cluster Head's certificate request is based on CH's current $\theta_i = PW_{HID}$ and the subsequent $\theta_{i+1} = R_{HID}$. On successful authentication of CH, WCN-AS updates its operating value to θ_{i+1} and sends its response message to CH. The attacker prevents WCN-AS's response message from reaching CH (cf. Figure 11). As CH does not receive the response to its certificate request, it resends the same request based on θ_i. However, as WCN-AS has updated to θ_{i+1}, the received θ_i is obsolete and the authentication request is denied.

Alternatively, CH may behave as an Increaser, that is, after a time-out it will update its operating value to θ_{i+1}. In this case, a desynchronisation attack can be mounted by intercepting the request message (cf. Figure 12). As WCN-AS does not receive the request, it will not update its operating value to θ_{i+1}. Consequently, when CH enters the following session with the updated operating, the request is rejected as the used θ_{i+1} is inconsistent with WCN-AS's current operating value θ_i.

Corresponding desynchronisation attacks can be mounted against the second certificate distribution phase of the Tseng protocol with the distinction that the Initiator and the Closer are played by GN and S. In this phase, CH and WCN-AS just relay messages containing θ.

5.4 Analysis of the AD Protocol

The AD protocol [3] is proposed to ensure mutual authentication between a mobile user (Mobile) and the base station (Base) in a wireless environment. The protocol consists of two phases:

- The Authentication Phase is performed once and it produces a session key stored by the communicating parties.
- The Key Change Phase is subsequently performed for mutual authentication and session key renewal (cf. Figure 13).

5.4.1 Applying Detection Rules for SD attacks to the AD Protocol

The Authentication phase is executed once by Mobile and Base. At completion of this three message exchange both parties store the initial instance of their shared secret $\theta_1 = {}^{new}RN^{M} \oplus {}^{new}RN^{B} = RN_1^M \oplus RN_1^B$. The notations symbolize the first random numbers generated by the Mobile (RN_1^M) and by the Base (RN_1^B). Following this phase Data Exchange may be entered by the two parties using the freshly obtained session key θ_1. Subsequently, the parties further authenticate by execution of Key Change phase. This phase implements a two-message update mechanism that aims to ensure new synchronous instances of each party's contribution to the session key: RN_i^M and RN_i^B. The Initiator (played by the Mobile) is a Unary Monostable and it behaves as a Repeater. The Generator (played by the Base) is also a Unary Monostable Repeater.

The following configuration is obtained for this UM-A update mechanism: $\alpha.M^1 \wedge \alpha.\Xi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$. As no reset phase is available for this protocol rule DR7 ($\alpha.M^1 \wedge \alpha.\Xi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{\mid\!\!\longleftarrow} \equiv \delta\alpha$) matches. Consequently, a desynchronisation attack mounted against this protocol will cause the Mobile to desynchronise, leading to a permanent DoS condition.

5.4.2 Desynchronisation Attacks against the AD Protocol

During the Key Change phase the Mobile and the Base authenticate each other based on the internal stored RN_i^M and RN_i^B. Figure 14 presents the SD attack against the AD protocol detected by the above analysis. The attacker prevents the second message from reaching the Mobile. As a consequence, the Base updates its stored data to θi+1 = ${}^{new}RN^{M} \oplus {}^{new}RN^{B}$ $= RN_{i+1}^M \oplus RN_{i+1}^B$ while the Mobile does not update to θ_{i+1} and continues to operate with θ_i = $RN_i^M \oplus RN_i^B$. As the Base and the Mobile now store different values for θ, any subsequent attempt to authenticate each other or to enter Data Exchange will fail.

5.5 Proposed Amendments to the AD Protocol

To eliminate the weakness causing a permanent DoS condition from the AD protocol we propose a new Key Change phase as outlined in Figure 15.

The Initiator (Mobile) remains a Unary Monostable principal (M^1) and a Repeater (Ξ): Message 1 is identical to the AD Protocol. The Closer (Base) becomes a Binary Bistable principal (B^2) and a Backward Differentiator (-Δ). Thus, the Base stores its current and its previous instances of RN^M and is capable of detecting any request from a desynchronised Mobile.

If the Base receives a request based on the expected random number (RN_i^M) it responds by issuing its own current and subsequent random numbers (RN_i^B and RN_{i+1}^B) in message 2.a and updates to θ_{i+1} = $RN_{i+1}^M \oplus RN_{i+1}^B$. However, if the Mobile's request is based on the previous pair of random numbers (RN_{i-1}^M and RN_i^M) the Base issues the resynchronisation challenge (Message 2.b) containing its current instance of the Mobile's random number (RN_i^M). If the Base receives requests containing any other random numbers the request is deemed a replay and ignored.

On receiving message 2.a Mobile updates to θ_{i+1} = $RN_{i+1}^M \oplus RN_{i+1}^B$ and session key renewal is successfully completed. The Mobile only reacts to a resynchronisation challenge (message 2.b) containing RN_i^M, if the most recently sent request was a failed attempt to update to RN_i^M. If the resynchronisation challenge is accepted, the Mobile responds with message 1 containing the random number indicated in the resynchronized challenge (RN_i^M) and the Mobile's subsequent random number (RN_{i+1}^M). The introduction of the resynchronisation challenge into the new Key Change phase does not prolong the life-time of the session key built on the obsolete pair of random numbers (RN_i^M and RN_i^B).

5.5.1 Analysis of the Amended AD Protocol

The amended version of the AD protocol proposed in Figure 15 implements a two-message update mechanism. In the new Key Change phase, the Initiator remains a Unary Monostable Repeater ($\alpha.M^1 \wedge \alpha.\Xi$). The Closer is still the Generator (ω.Γ), but changes its visibility scope to Binary and the operating values policy to Bistable ($\omega.B^2$). Further, the Closer now behaves as a Backward Differentiator (ω.-Δ).

The obtained configuration ($\alpha.M^1 \wedge \alpha.\Xi \wedge \omega.B^2 \wedge \omega.\Gamma \wedge \omega.\text{-}\Delta$) has no corresponding attack detection rule. Therefore, the proposed Key Change phase is safe against SD attacks.

6. Conclusions

This paper addressed synchronisation issues in security protocols for wireless communications that utilise online update mechanisms. The paper revealed a new weakness in the design of update mechanisms of current security protocols that utilise dynamic shared secrets. We have shown that this weakness is present in a wide range of protocols for wireless communication. Further, a new attack, called suppress-and-desynchronise attack (SD attack), that exploits the exposed design weaknesses was presented. The impact of a successful SD attack on these protocols is that a permanent denial of service condition is reached, preventing further communication between the involved parties.

A formal verification system that is able to model update mechanisms for dynamic shared secrets was developed. This system models update mechanisms by expressing the actions of principals, their storing strategies for shared secrets and the principals' roles in the different types of update mechanisms. As part of the verification system a set of rules for the detection of update mechanisms that are susceptible to suppress-and-desynchronise attacks was proposed. As these rules were derived from exhaustive configuration sets, the resulting verification system is complete and therefore it is able to detect all possible SD attacks.

The effectiveness of the verification system was confirmed by utilising it in the analysis of several security protocols for wireless communications. Each analysed protocol was briefly introduced and then formally modelled in the verification system. The analysis of the protocols was performed by applying the detection rules to the formalised protocols. The results of the presented verifications revealed hitherto unknown weaknesses in the HYS, Tseng and AD protocols and details of suppress-and-desynchronise attacks that exploit these weaknesses were presented. In each case, mounting a suppress-and-desynchronise attack resulted in a permanent DoS condition for the involved communicating parties. An amended version of the AD protocol was proposed and application of the verification system established the immunity of the amended AD protocol against the identified suppress-and-desynchronise attacks.

The hitherto unnoticed presence of the revealed weaknesses in the analysed protocols highlights the importance of using formal methods to analyse the design of security protocols. The authors anticipate that future work will lead to the integration of the presented SD attack detection system into existing state-space or logic-based verification systems.

7. Acknowledgement

This work was funded by the Irish Research Council for Science, Engineering and Technology (IRCSET Embark Initiative) and Science Foundation Ireland - Research Frontiers Programme (11/RFP.1/CMS/3340).

8. References

[1] S. Ravi , A. Raghunathan, P. Kocher, S. Hattangady, "Security in Embedded Systems: Design Challenges," ACM Transactions on Embedded Computing Systems, vol.3, no. 3, pp. 461–491, August 2004.

[2] T. Coffey, R. Dojen, and T. Flanagan, "Formal Verification: An Imperative Step in the Design of Security Protocols", Computer Networks Journal, Elsevier Science, Volume 43, Number 5, 5 December 2003, pp.601-618.

[3] Aziz, and W. Diffie, “Privacy and Authentication for Wireless Local Area Networks”, IEEE Personal Communications, pp. 25-31, First Quarter 1994

[4] M. S. Bargh, R. J. Hulsebosch, E. H. Eertink, A. Prasad, H. Wang, P. Schoo, “Fast authentication methods for handovers between IEEE 802.11 wireless LANs”, in ACM Proc. of Wireless Mobile Applications and Services on WLAN Hotspots, Philadelphia, 1 October 2004, pp. 51-60.

[5] WB. Lee and CK. Yeh, “A new delegation-based authentication protocol for use in portable communication systems”, IEEE Transactions on Wireless Communications, vol. 4, no. 1, pp. 57-64, Jan. 2005.

[6] J. Nan and W. Jian, “Anonymous authentication protocol for multi-services in wireless environments”, The Journal of China Universities of Posts and Telecommunications, vol. 15, no.4, 69–74, December 2008.

[7] TF. Lee, SH. Chang, T. Hwang, and SK. Chong, “Enhanced Delegation-Based Authentication Protocol for PCSs”, IEEE Trans. on Wireless Communications, vol. 8, no. 5, May 2009.

[8] M. Li and K. Sandrasegaran, “A Proxy Based Authentication Localisation Scheme for Handover between Non Trust-Associated Domains”, SIGMOBILE Mobile Computing and Communications Review, vol. 13, no. 4. pp. 47-58, October 2009.

[9] K. Shim, “The risk of compromising secret information”, in ICICS 2002, LNCS 2513, R. Deng et al. Ed., pp. 122–133, 2002, Springer-Verlag Berlin.

[10] Lasc, R. Dojen, T. Coffey, “Countering jamming attacks against an authentication and key agreement protocol for mobile satellite communications”, Elsevier Computers & Electrical Engineering, vol. 37, issue 2, pp. 160-168, March 2011

[11] Lasc, R. Dojen, T. Coffey, “A Mutual Authentication Protocol with Resynchronisation Capability for Mobile Satellite Communications”, International Journal of Information Security and Privacy, vol. 5, no.1, January 2011, pp. 33-49.

[12] MS. Hwang, CC. Yang, CY. Shiu, “An authentication scheme for mobile satellite communication systems”, ACM SIGOPS Operating Systems Review, vol. 37, no. 4, pp. 42-47, October 2003.

[13] YF. Chang and CC. Chang - “An efficient authentication protocol for mobile satellite communication systems”, ACM SIGOPS Operating Systems Review, vol. 39, no. 1, pp. 70-84, January 2005.

[14] YM. Tseng, “A heterogeneous-network aided public-key management scheme for mobile ad hoc networks”, International Journal of Network Management; vol. 17, pp. 3–15, 2007.

[15] TZ. Chen, WB. Lee and HB. Chen, “A self-verification authentication mechanism for mobile satellite communication systems”, Computers and Electrical Engineering, vol. 35, no. 1, pp. 41-48, January 2009.

[16] H. Chen, Y. Xiao, X. Hong, F. Hu and J. Xi, “A survey of anonymity in wireless communication systems”, Security and Communication Networks, vol. 2, pp. 427–444, 2009.

[17] YM. Tseng, CC. Yang and JH. Su - “Authentication and Billing Protocols for the Integration of WLAN and 3G Networks”, Wireless Personal Communications, vol. 29, pp. 351–366, 2004 Kluwer Academic Publishers.

[18] D. Dolev and A. Yao, “On the security of public key protocols”, IEEE Transactions on Information Theory, vol. 29, no.2, pp. 198-208, 1983.

[19] R. Dojen and T. Coffey, “Layered Proving Trees: A Novel Approach to the Automation of Logic-Based Security Protocol Verification”, ACM Transactions on Information and System Security (TISSEC), Volume 8, Issue 3, August 2005, pp. 287-311

[20] Y. Jiang, C. Lin and X. Shen, “Mutual Authentication and Key Exchange Protocols for Roaming Services in Wireless Mobile Networks”, IEEE Transaction on Wireless Communications, vol. 5, no. 9, 2006

[21] J. Gray, “Notes on Data Base Operating Systems”, Proceedings of Operating Systems, Springer Verlag, pp.393-481, 1978

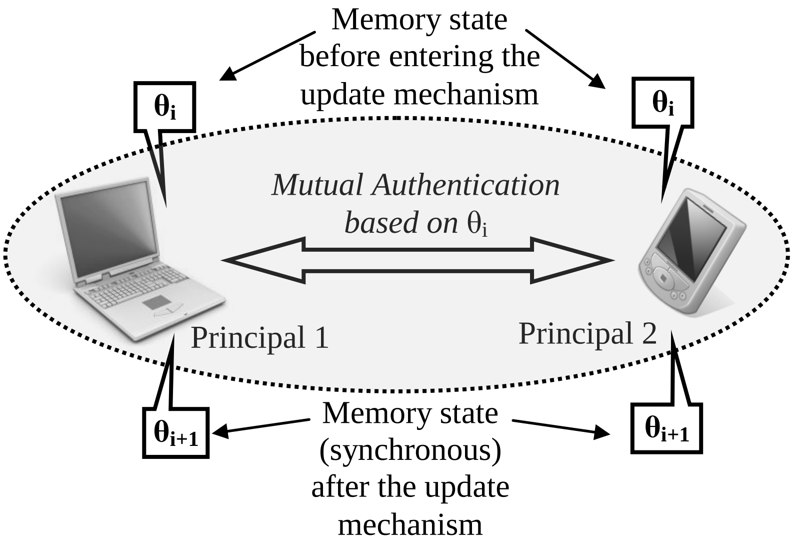
Memory state before entering the update mechanism
θ_i
θ_i
Mutual Authentication based on θ_i
Principal 1
Principal 2
θ_{i+1}
Memory state (synchronous) after the update mechanism
θ_{i+1}

Fig. 1. Synchronous shared secrets provided by an update mechanism.

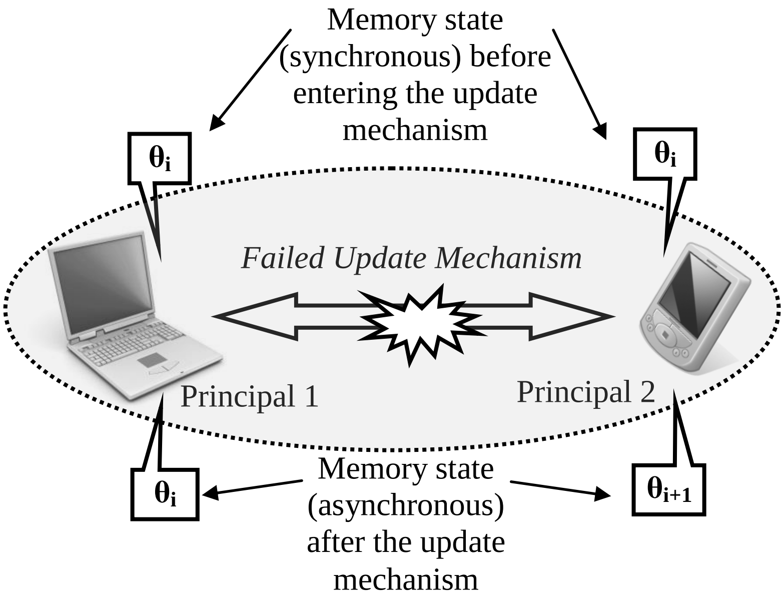
Memory state (synchronous) before entering the update mechanism
θ_i
θ_i
Failed Update Mechanism
Principal 1
Principal 2
θ_i
Memory state (asynchronous) after the update mechanism
θ_{i+1}

Fig. 2. Asynchronous evolution of shared secrets in consecutive authentication sessions.

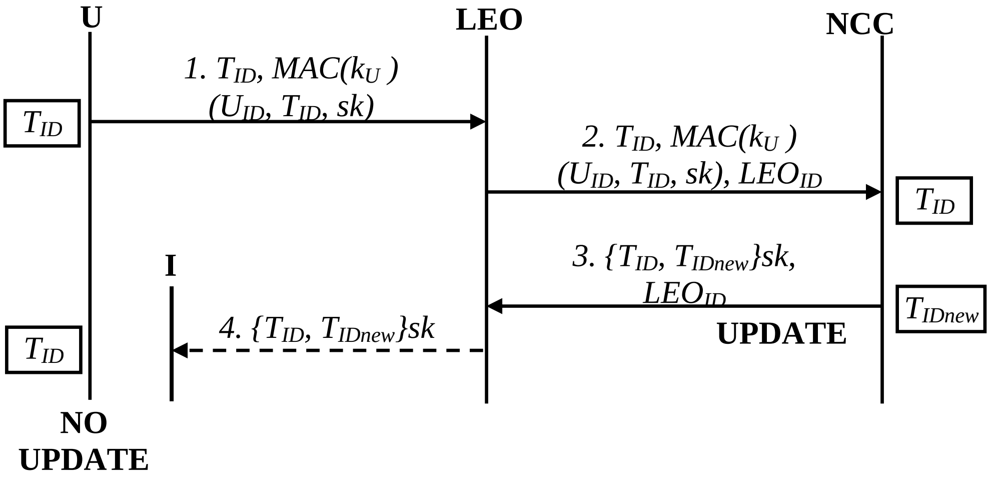
U
LEO
NCC
1. T_{ID}, $MAC(k_U)$ (U_{ID}, T_{ID}, sk)
T_{ID}
2. T_{ID}, $MAC(k_U)$ (U_{ID}, T_{ID}, sk), LEO_{ID}
T_{ID}
I
3. $\{T_{ID}, T_{IDnew}\}sk$, LEO_{ID}
T_{IDnew}
UPDATE
4. $\{T_{ID}, T_{IDnew}\}sk$
T_{ID}
NO UPDATE

Fig. 3. SD attack against Authentication Phase in CLC Protocol.

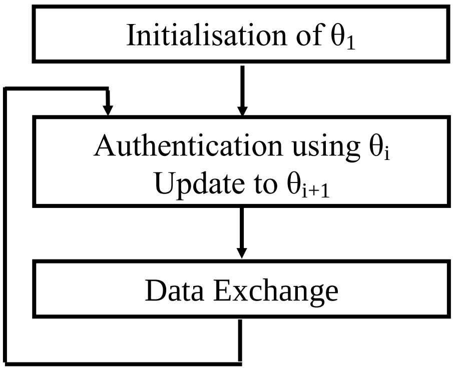

Fig. 4. Details of Update Mechanism UM-A.

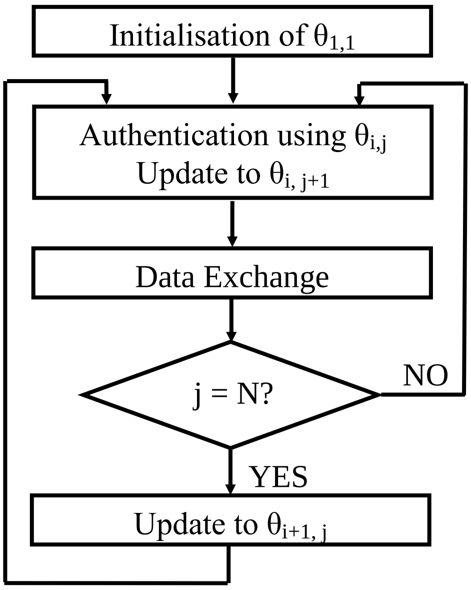
Initialisation of $\theta_{1,1}$
Authentication using $\theta_{i,j}$
Update to $\theta_{i, j+1}$
Data Exchange
j = N?
NO
YES
Update to $\theta_{i+1, j}$

Fig. 5. Details of Update Mechanism UM-B.

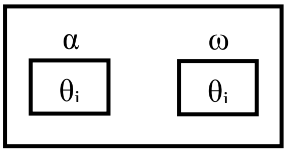

Fig. 6. Visibility scope for principals in UM1.

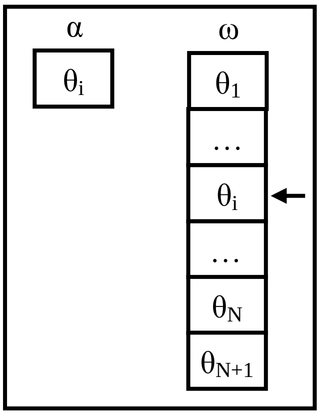

Fig. 7. Visibility scope for principals in UM2

1. *MS −> LEO:* T_{ID}, $K_{md}(U_{ID}, T_{ID})$
2. *LEO −> NCC:* T_{ID}, LEO_{ID}, $K_{md}(U_{ID}, T_{ID})$
3. *NCC −> LEO:* T_{ID}, LEO_{ID}, $K_{md}(T_{ID}, T_{ID}', K_{md}')$
4. *LEO −> MS:* T_{ID}, $K_{md}(T_{ID}, T_{ID}', K_{md}')$

Fig. 8. HYS Mobile User Authentication Phase

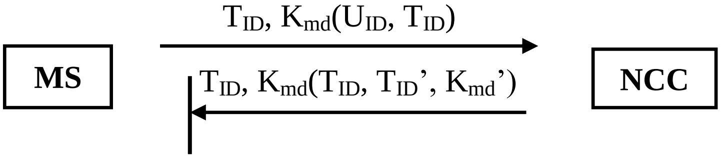
T_{ID}, $K_{md}(U_{ID}, T_{ID})$
MS
T_{ID}, $K_{md}(T_{ID}, T_{ID}', K_{md}')$
NCC

Fig. 9. Desynchronisation attack against the HYS protocol.

1. *CH −>WCN-AS:* HID, PK_{HID}, $E_{PKc}(HID||R_{HID})$,H(PW_{HID}, R_{HID}, HID, PK_{HID})
2. *WCN-AS −>CH*: HID, $Cert_{HID}$, PK_C, $Cert_C$, $Chain_{C–CA}$

Fig. 10. Tseng Protocol Phase One

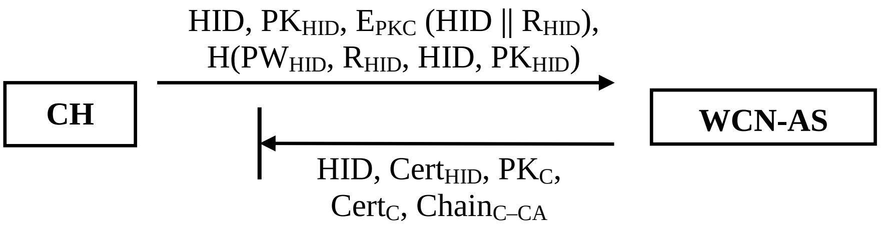
HID, PK_{HID}, E_{PKC} (HID || R_{HID}),
H(PW_{HID}, R_{HID}, HID, PK_{HID})
CH
WCN-AS
HID, $Cert_{HID}$, PK_C,
$Cert_C$, $Chain_{C–CA}$

Fig. 11. Desynchronisation Attack against the Tseng Protocol with CH as Repeater.

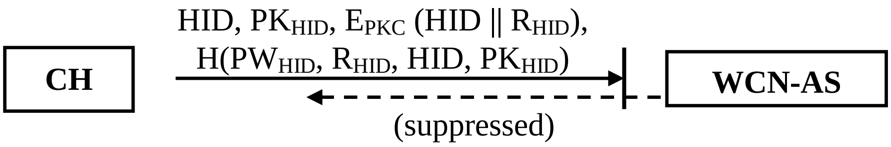
HID, PK_{HID}, E_{PKC} (HID || R_{HID}),
H(PW_{HID}, R_{HID}, HID, PK_{HID})
CH
WCN-AS
(suppressed)

Fig. 12. Desynchronisation Attack against the Tseng Protocol with CH as Increaser.

1. Mobile −> Base: Signed(Priv_Mobile, { E(Pub_Base, RN_i^M), E(Pub_Base, RN_{i+1}^M) })

2. Base−> Mobile: Signed(Priv_Base, { E(Pub_Mobile, RN_i^B), E(Pub_Mobile, RN_{i+1}^B) })

Fig. 13. AD protocol key change phase

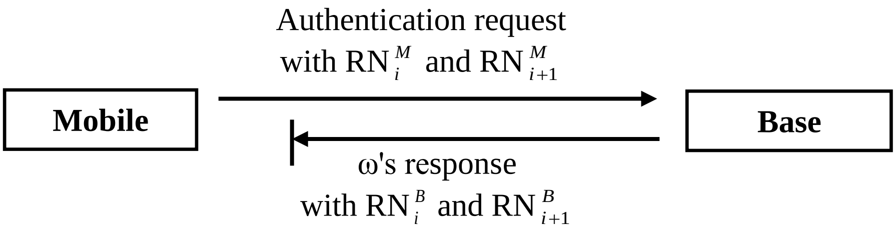
Authentication request
with RN_i^M and RN_{i+1}^M
Mobile
Base
ω's response
with RN_i^B and RN_{i+1}^B

Fig. 14. Desynchronisation attack against AD Protocol

1. Mobile −> Base: Signed(Priv_Mobile, { E(Pub_Base, RN_i^M), E(Pub_Base, RN_{i+1}^M) })

2.a. Base−> Mobile: Signed(Priv_Base, { E(Pub_Mobile, RN_i^B), E(Pub_Mobile, RN_{i+1}^B) })

2.b. Base −> Mobile: Signed(Priv_Base, {E(Pub_Mobile, RN_i^M)})

Fig. 15. The proposed Key Change phase.

Symbol/Term	Meaning
UM-A	Update mechanism in which at any given time only a single instance of the shared secret is generated.
UM-B	Update mechanism that works with sets of N pre-determined shared secrets.
α	Initiator
ω	Closer
Γ	Generator
Visibility Scope	Accessible shared secret instances – principals can be classified as Unary, Binary, n-ary
Operating Values	Shared secret instances used in a session
M	Monostable*
B	Bistable*
N	N-Stable*
Asynchronous interval	Period during update mechanism when principals have asynchronous shared secret instances.
Ξ	Repeater
Ψ	Increaser
Δ	Differentiator
Δ^{+}	Forward Differentiator
${}^{-}\Delta$	Backward Differentiator
${}^{-}\Delta^{+}$	Random Differentiator
Θ	Unconditional Increaser
σ(p)	Single Session Identification
σ(p,q)	Session Range Identification
σ.ρ	Presence of Reset Phase
σ.Φ	Absence of Reset Phase
P.property1 ∧ P.property2	The principal P has the properties: property1 AND property2

*Optionally, arity is indicated by superscript 1: Unary, 2: Binary, n: n-ary

Table 1. Verification System Symbols and Terms

No.	Configuration	Configuration Label
1	$\alpha.M^1 \wedge \alpha.\Xi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
2	$\alpha.M^1 \wedge \alpha.\Theta \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
3	$\alpha.M^1 \wedge \alpha.\Delta \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
4	$\alpha.M^1 \wedge \alpha.\Xi \wedge \omega.M^2 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
5	$\alpha.M^1 \wedge \alpha.\Theta \wedge \omega.M^2 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
6	$\alpha.M^1 \wedge \alpha.\Delta \wedge \omega.M^2 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
7	$\alpha.M^1 \wedge \alpha.\Xi \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	I
8	$\alpha.M^1 \wedge \alpha.\Theta \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	I
9	$\alpha.M^1 \wedge \alpha.\Delta \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	I
10	$\alpha.M^2 \wedge \alpha.\Xi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
11	$\alpha.M^2 \wedge \alpha.\Theta \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
12	$\alpha.B^2 \wedge \alpha.\Delta \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
13	$\alpha.M^2 \wedge \alpha.\Xi \wedge \omega.M^2 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
14	$\alpha.M^2 \wedge \alpha.\Theta \wedge \omega.M^2 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
15	$\alpha.B^2 \wedge \alpha.\Delta \wedge \omega.M^2 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
16	$\alpha.M^2 \wedge \alpha.\Xi \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	I
17	$\alpha.M^2 \wedge \alpha.\Theta \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	I
18	$\alpha.B^2 \wedge \alpha.\Delta \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	I
19	$\alpha.M^n \wedge \alpha.\Xi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
20	$\alpha.M^n \wedge \alpha.\Theta \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	A
21	$\alpha.B^n \wedge \alpha.\Delta^+ \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	N
22	$\alpha.B^n \wedge \alpha.{}^-\Delta \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	S
23	$\alpha.B^n \wedge \alpha.{}^-\Delta^+ \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
24	$\alpha.N^n \wedge \alpha.\Delta^+ \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	N
25	$\alpha.N^n \wedge \alpha.{}^-\Delta \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	S
26	$\alpha.N^n \wedge \alpha.{}^-\Delta^+ \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	S
27	$\alpha.M^n \wedge \alpha.\Xi \wedge \omega.M^2 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
28	$\alpha.M^n \wedge \alpha.\Theta \wedge \omega.M^2 \wedge \omega.\Xi \wedge \omega.\Gamma$	N
29	$\alpha.B^n \wedge \alpha.\Delta^+ \wedge \omega.M^2 \wedge \omega.\Xi \wedge \omega.\Gamma$	N
30	$\alpha.B^n \wedge \alpha.{}^-\Delta \wedge \omega.M^2 \wedge \omega.\Xi \wedge \omega.\Gamma$	N
31	$\alpha.B^n \wedge \alpha.{}^-\Delta^+ \wedge \omega.M^2 \wedge \omega.\Xi \wedge \omega.\Gamma$	I
32	$\alpha.N^n \wedge \alpha.\Delta \wedge \omega.M^2 \wedge \omega.\Xi \wedge \omega.\Gamma$	N
33	$\alpha.M^n \wedge \alpha.\Xi \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	I
34	$\alpha.M^n \wedge \alpha.\Theta \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	A
35	$\alpha.B^n \wedge \alpha.\Delta^+ \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	N
36	$\alpha.B^n \wedge \alpha.{}^-\Delta \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	S
37	$\alpha.B^n \wedge \alpha.{}^-\Delta^+ \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	I
38	$\alpha.N^n \wedge \alpha.\Delta^+ \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	N
39	$\alpha.N^n \wedge \alpha.{}^-\Delta \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	N
40	$\alpha.N^n \wedge \alpha.{}^-\Delta^+ \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	S

Table 2. One Message Update Mechanism Configurations with $\omega.\Gamma$

Config.	Rule Label	Detection Rule
20	DR1	$\alpha.M^n \wedge \alpha.\Theta \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \longrightarrow\!\!\mid\ \equiv \delta\omega$
34	DR2	$\alpha.M^n \wedge \alpha.\Theta \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \longrightarrow\!\!\mid\ \equiv \delta\omega$

Table 3. Desynchronisation attack detection rules for one-message update mechanism with ω.Γ

No.	Configuration	Configuration Label
1	$\alpha.M^1 \wedge \alpha.\Theta \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi$	A
2	$\alpha.M^1 \wedge \alpha.\Theta \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi$	A
3	$\alpha.\ M^n \wedge \alpha.\Theta \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi$	A
4	$\alpha.M^n \wedge \alpha.\Theta \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi$	A

Table 4. Unsafe one-message update mechanisms with α.Γ.

Config.	Rule Label	Detection Rule
1	DR3	$\alpha.M^1 \wedge \alpha.\Theta \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi \wedge \sigma.\Phi \wedge \longrightarrow\!\!\mid\ \equiv \delta\omega$
2	DR4	$\alpha.M^1 \wedge \alpha.\Theta \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \longrightarrow\!\!\mid\ \equiv \delta\omega$
3	DR5	$\alpha.\ M^n \wedge \alpha.\Theta \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \longrightarrow\!\!\mid\ \equiv \delta\omega$
4	DR6	$\alpha.M^n \wedge \alpha.\Theta \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \longrightarrow\!\!\mid\ \equiv \delta\omega$

Table 5. Desynchronisation attack detection rules for one-message update mechanism with α.Γ

No.	Configuration	Configuration Label
1	$\alpha.M^1 \wedge \alpha.\Xi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	A
2	$\alpha.M^1 \wedge \alpha.\Xi \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	A
3	$\alpha.M^n \wedge \alpha.\Xi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	A
4	$\alpha.\ M^n \wedge \alpha.\Psi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma$	A
5	$\alpha.\ M^n \wedge \alpha.\Xi \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	A
6	$\alpha.M^n \wedge \alpha.\Psi \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma$	A

Table 6. Unsafe two-message update mechanisms with ω.Γ.

Config.	Rule Label	Detection Rule
1	DR7	$\alpha.M^1 \wedge \alpha.\Xi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{\mid\!\!\longleftarrow} \equiv \delta\alpha$
2	DR8	$\alpha.M^1 \wedge \alpha.\Xi \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{\mid\!\!\longleftarrow} \equiv \delta\alpha$
	DR9	$\alpha.M^1 \wedge \alpha.\Xi \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\,(N) \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{\mid\!\!\longleftarrow} \equiv \delta\alpha$
3	DR10	$\alpha.M^n \wedge \alpha.\Xi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\,(1,\ N\text{-}1) \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{\mid\!\!\longleftarrow} \equiv \delta\alpha$
	DR11	$\alpha.M^n \wedge \alpha.\Xi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\,(N) \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{\mid\!\!\longleftarrow} \equiv \delta\alpha$
4	DR12	$\alpha.M^n \wedge \alpha.\Psi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \overset{\longrightarrow\!\!\mid}{\dashleftarrow} \equiv \delta\omega$
	DR13	$\alpha.M^n \wedge \alpha.\Psi \wedge \omega.M^1 \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\,(N) \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{\mid\!\!\longleftarrow} \equiv \delta\alpha$
5	DR14	$\alpha.M^n \wedge \alpha.\Xi \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{\mid\!\!\longleftarrow} \equiv \delta\alpha$
	DR15	$\alpha.M^n \wedge \alpha.\Xi \wedge \omega.\ M^n \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\,(N) \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{\mid\!\!\longleftarrow} \equiv \delta\alpha$
6	DR16	$\alpha.M^n \wedge \alpha.\Psi \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \overset{\longrightarrow\!\!\mid}{\dashleftarrow} \equiv \delta\omega$
	DR17	$\alpha.M^n \wedge \alpha.\Psi \wedge \omega.M^n \wedge \omega.\Xi \wedge \omega.\Gamma \wedge \sigma\,(N) \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{\mid\!\!\longleftarrow} \equiv \delta\alpha$

Table 7. Desynchronisation attack detection rule for two-message update mechanism with ω.Γ.

No.	Configuration	Configuration Label
1	$\alpha.M^1 \wedge \alpha.\Xi \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi$	A
2	$\alpha.M^1 \wedge \alpha.\Psi \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi$	A
3	$\alpha.M^1 \wedge \alpha.\Xi \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi$	A
4	$\alpha.M^1 \wedge \alpha.\Psi \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi$	A
5	$\alpha.M^n \wedge \alpha.\Xi \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi$	A
6	$\alpha.M^n \wedge \alpha.\Psi \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi$	A
7	$\alpha.M^n \wedge \alpha.\Xi \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi$	A
8	$\alpha.M^n \wedge \alpha.\Psi \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi$	A

Table 8. Unsafe two-message update mechanisms with α.Γ.

Config.	Rule Label	Detection Rule
1	DR18	$\alpha.M^1 \wedge \alpha.\Xi \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{|\!\!\longleftarrow} \equiv \delta\alpha$
2	DR19	$\alpha.M^1 \wedge \alpha.\Psi \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi \wedge \sigma.\Phi \wedge \overset{\longrightarrow\!|}{\dashleftarrow} \equiv \delta\omega$
3	DR20	$\alpha.M^1 \wedge \alpha.\Xi \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{|\!\!\longleftarrow} \equiv \delta\alpha$
	DR21	$\alpha.M^1 \wedge \alpha.\Xi \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi \wedge \sigma\,(N) \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{|\!\!\longleftarrow} \equiv \delta\omega$
4	DR22	$\alpha.M^1 \wedge \alpha.\Psi \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi \wedge \sigma\,(1, N\text{-}1) \wedge \overset{\longrightarrow\!|}{\dashleftarrow} \wedge \sigma.\Phi \equiv \delta\omega$
	DR23	$\alpha.M^1 \wedge \alpha.\Psi \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi \wedge \sigma\,(N) \wedge \sigma.\Phi \wedge \overset{\longrightarrow\!|}{\dashleftarrow} \equiv \delta\omega$
5	DR24	$\alpha.M^n \wedge \alpha.\Xi \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi \wedge \sigma\,(1, N\text{-}1) \wedge \overset{\longrightarrow}{|\!\!\longleftarrow} \wedge \sigma.\Phi \equiv \delta\alpha$
	DR25	$\alpha.M^n \wedge \alpha.\Xi \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi \wedge \sigma\,(N) \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{|\!\!\longleftarrow} \equiv \delta\alpha$
6	DR26	$\alpha.M^n \wedge \alpha.\Psi \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \overset{\longrightarrow\!|}{\dashleftarrow} \equiv \delta\omega$
	DR27	$\alpha.M^n \wedge \alpha.\Psi \wedge \alpha.\Gamma \wedge \omega.M^1 \wedge \omega.\Xi \wedge \sigma\,(N) \wedge \sigma.\Phi \wedge \overset{\longrightarrow\!|}{\dashleftarrow} \equiv \delta\omega$
7	DR28	$\alpha.M^n \wedge \alpha.\Xi \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{|\!\!\longleftarrow} \equiv \delta\alpha$
	DR29	$\alpha.M^n \wedge \alpha.\Xi \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi \wedge \sigma\,(N) \wedge \sigma.\Phi \wedge \overset{\longrightarrow}{|\!\!\longleftarrow} \equiv \delta\alpha$
8	DR30	$\alpha.M^n \wedge \alpha.\Psi \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi \wedge \sigma\,(1, N\text{-}1) \wedge \sigma.\Phi \wedge \overset{\longrightarrow\!|}{\dashleftarrow} \equiv \delta\omega$
	DR31	$\alpha.M^n \wedge \alpha.\Psi \wedge \alpha.\Gamma \wedge \omega.M^n \wedge \omega.\Xi \wedge \sigma\,(N) \wedge \sigma.\Phi \wedge \overset{\longrightarrow\!|}{\dashleftarrow} \equiv \delta\omega$

Table 9. Desynchronisation attack detection rules for two-message update mechanisms with α.Γ