A Mutual Authentication Protocol with Resynchronisation Capability for Mobile Satellite Communications

Ioana Lasc, Reiner Dojen and Tom Coffey

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

Author Note

This work was funded by the Irish Research Council for Science, Engineering and Technology (IRCSET Embark Initiative) and Science Foundation Ireland - Research Frontiers Programme (SFI RFP07CMSF 631).

Correspondence concerning this article should be addressed to Reiner Dojen, Department of Electronic and Computer Engineering, University of Limerick, Limerick, Ireland. Email: reiner.dojen@ul.ie

Abstract

Many peer-to-peer security protocols proposed for wireless communications use one-time shared secrets for authentication purposes. This paper analyses online update mechanisms for one-time shared secrets. A new type of attack against update mechanisms, which we term desynchronisation attack, is introduced. This type of attack may lead to a permanent denial of service condition. A case study demonstrates the effectiveness of desynchronisation attacks against a security protocol for mobile satellite communications.

A new mutual authentication protocol for satellite communications, incorporating a resynchronisation capability is proposed to counter the disruptive effects of desynchronisation attacks. The new protocol has a resynchronisation phase that is initiated whenever desynchronisation is suspected. Thus, the possibility of causing permanent denial of service conditions by mounting desynchronisation attacks is eliminated. A security analysis of the proposed protocol establishes its resistance against attacks such as replay attacks, dictionary attacks and desynchronisation attacks.

***Keywords**:* security protocols, authentication, wireless communications, desynchronisation attacks, denial of service

A Mutual Authentication Protocol with Resynchronisation Capability for Mobile Satellite Communications

Wireless communications are being driven by the need to provide network access for mobile or nomadic computing devices. The increase of available services using these wireless devices requires users to trust the employed communication networks with highly sensitive information. Security protocols are one of the most critical elements in enabling the secure communication and processing of information. Basic security protocols allow agents to authenticate each other, to establish fresh session keys for confidential communication and to ensure the authenticity of data and services (Dojen, Lasc, & Coffey, 2008). Building on such basic security protocols, more advanced services like non-repudiation, fairness, electronic payment and electronic contract signing are achieved. The design of such security protocols should be robust enough to resist attacks, such as replay attacks, parallel session attacks or type-flaw attacks. Additionally, the possibility of interfering with communication on physical channels by jamming needs to be considered as it can affect the security at the application layer (Gansler, & Binnendijk, 2005). Denial of Service (DoS) attacks are a common form of cross-layer attacks (Radosavac, Benammar, Baras & John, 2004), which can be achieved in two ways (Peng, Leckie, & Ramamohanarao, 2007): Firstly, an attacker can interact continuously with the targeted system to prevent service availability. In this case, the DoS condition disappears when the malicious interaction ends. Secondly, an attacker can interact with the targeted system for only a limited time to achieve a permanent denial of service condition. In this case the DoS condition remains until some corrective measures are taken.

Many peer-to-peer security protocols proposed for wireless communications use one-time shared secrets for authentication purposes (Lee, Chang, Hwang, & Chong, 2009), (Chen, Lee, & Chen, 2008), (Tseng, 2007), (Chang & Chang, 2005), (Lee, & Yeh, 2005). These secrets are used by the owning principals to prove their identity in a protocol session. Additionally, the same protocol run establishes a new instance of the shared secret that will be used in the next session. The messages of the protocol that establish the new shared secret are referred to as the update mechanism. These update mechanisms serve two purposes: Firstly, the generation of a new instance of the shared secret and, secondly, agreement on the same new shared secret by all communicating parties. Thus, update mechanisms aim to ensure synchronous storage of the shared secret by all principals at the end of each protocol run.

In this paper, online update mechanisms for shared secrets are analysed. This analysis reveals a new form of attack against security protocols employing update mechanisms. As these new attacks aim to desynchronise the communicating parties on their stored shared secret, we term them desynchronisation attacks. A successful desynchronisation attack disables the affected parties from authenticating each other in future protocol runs. Thus, by mounting a desynchronisation attack, the intruder can cause a permanent DoS condition. As an example, a case study is performed on a protocol for mobile satellite communications that aims at providing mutual authentication between a mobile user and the Network Control Centre (NCC). This case study reveals two desynchronisation attacks that exploit the weaknesses in the update mechanism employed by the protocol. Subsequently, the targeted users are denied access to the communication service and a permanent denial of service condition is reached that persists even after the jamming ends.

To counter the disruptive effects of desynchronisation attacks, a new mutual authentication protocol with resynchronisation capability is proposed that confirms synchrony of the stored shared secrets. If desynchronisation is detected a resynchronisation phase is initiated that re-establishes synchrony between the communicating parties. Thus, the possibility of affecting the security at application layer by interfering with the communication on physical channels is eliminated. A security analysis of the proposed protocol establishes its resistance against attacks such as replay attacks, dictionary attacks and desynchronisation attacks.

Online update mechanisms of shared secrets

Providing mutual authentication based on shared secrets is a common feature in security protocols (Bargh et al., 2004), where the communicating parties involved prove their identity by showing possession of the shared secrets. To avoid freshness issues, such as compromise of long-term session keys, replay attacks etc., security protocols use dynamic shared secrets, where the secrets are renewed in an online update mechanism.

Online update mechanisms have two goals: Firstly, the generation of new instance of the shared secret. Secondly, the agreement of all communicating parties on this new instance as the current shared secret. Consequently, update mechanisms aim to ensure a synchronous storage of the same shared secret by all principals at the end of each protocol run as outlined in Figure 1: When entering a protocol session, the two parties store one instance of the shared secret (the *old secret*) and at completion of the session they store the updated instance of the shared secret (the *new secret*).

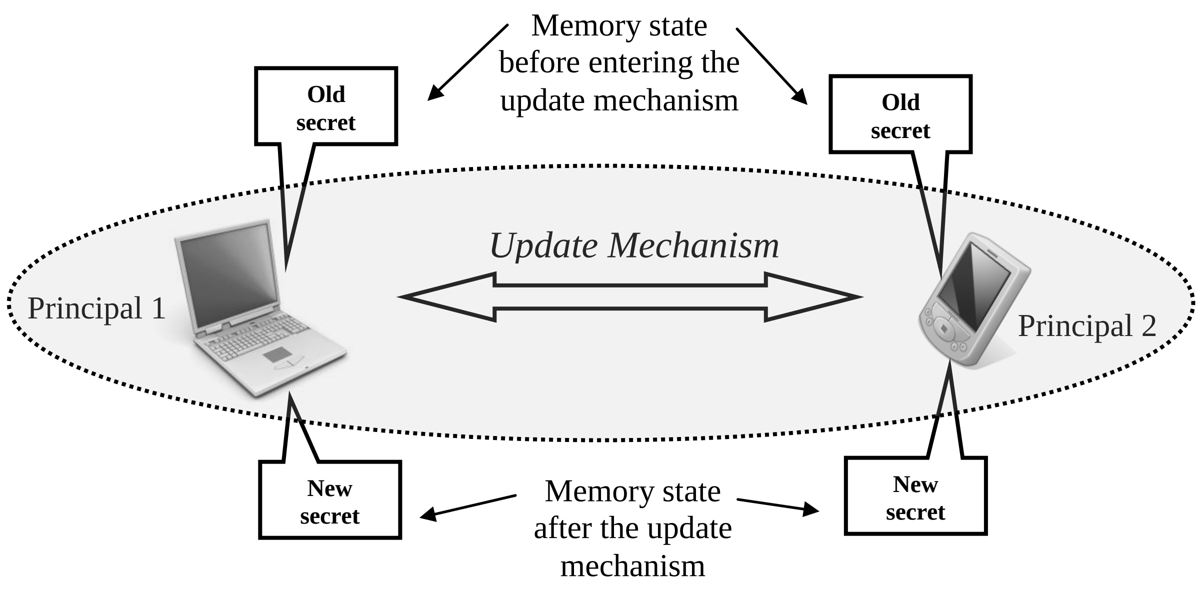
Memory state before entering the update mechanism
Old secret
Old secret
Update Mechanism
Principal 1
Principal 2
New secret
New secret
Memory state after the update mechanism

Figure 1. Synchrony of shared secrets provided by an update mechanism.

Structure of Security Protocols Employing an Update Mechanism

Security protocols use update mechanisms to establish new instances of shared secrets. A successful run of such a protocol ensures mutual authentication of both principals by proving possession of the current shared secret to each other. Further, a new shared secret is agreed upon that will be used in the subsequent protocol run.

Update mechanisms using a pre-determined chain of shared secrets can be used to minimise the computational burden on the communicating parties. Security protocols that use such online update mechanisms have the structure outlined in Figure 2: A chain of shared secrets is initially established in a one-time registration phase and can facilitate multiple (i.e. N) authentication phases, where each phase uses consecutive values from the chain. At every N^{th} authentication phase the end of the chain is reached. The update phase is then entered to establish a new chain of shared secrets.

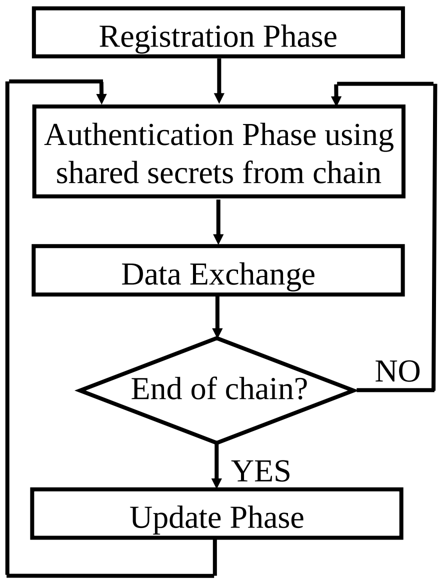

Figure 2. Structure of security protocols employing an update mechanism for shared secrets.

The details of the authentication phase are outlined in Figure 3. The user computes the new shared secret θ_{i+1} and sends its authentication request to the server using both its current shared secret θ_i and the new secret θ_{i+1}. Authenticity of the user is evaluated by the server by comparing the received θ_i against its own stored secret. If these values match, the server locally updates its stored secret from θ_i to θ_{i+1} and sends its response to the user. On successful authentication of the server's response message, the user also updates its stored current secret to θ_{i+1}. As shown, the two principals do not update their data simultaneously: the server replaces the old value (θ_i) with the new one (θ_{i+1}) before issuing the response message and the user does it after receiving the message from the server.

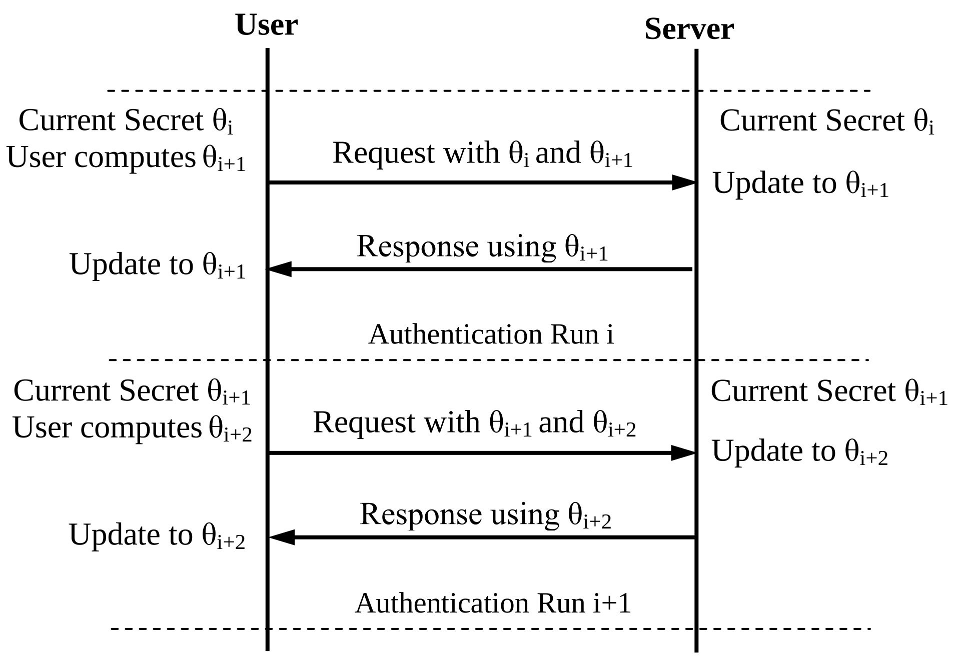
User
Server
Current Secret θ_i
User computes θ_{i+1}
Request with θ_i and θ_{i+1}
Current Secret θ_i
Update to θ_{i+1}
Response using θ_{i+1}
Update to θ_{i+1}
Authentication Run i
Current Secret θ_{i+1}
User computes θ_{i+2}
Request with θ_{i+1} and θ_{i+2}
Current Secret θ_{i+1}
Update to θ_{i+2}
Response using θ_{i+2}
Update to θ_{i+2}
Authentication Run i+1

Figure 3: Authentication phase structure.

At the Nth authentication phase the update phase is entered, where the server generates a seed for a new chain of shared secrets. Analogous to the authentication phase, the server authenticates the user on the value of the contained shared secret $\theta_{N.}$ Subsequent to successful authentication, the server generates a seed for a new chain of shared secrets and updates its current secret to the first value θ_1 of the new chain. The server also includes this seed in its response to the user. On receipt of the server's response message, the user verifies its authenticity by comparing the received θ_N against its stored instance of θ_N. If these match, the user accepts the new chain of shared secrets and updates its current secret to the first value θ_1 of the new chain.

Frequently, the pre-determined chain of shared secrets is implemented by a backward hash chain built from a *seed* and a one-way collision-resistant hash function *h*. The *seed* can be a session key or any other secret material, such as a random number. The *m*th element of the hash chain is computed with the formula $h^m(seed)$, where *m* iterates from N+1 down to 1 and $h^m()$ denotes *m* applications of the hash function *h*. Thus, in the first authentication phase the seed is hashed N+1 times to obtain the first instance of the shared secret $\theta_1 = h^{N+1}(seed)$. In general the authentication token θ_i is computed by $h^{N+1-i}(seed)$.

Desynchronisation Attacks against Update Mechanisms

Online update mechanisms for shared secrets aim to ensure generation and distribution of new instances of the shared secret between the communicating parties. As shown in the previous section, the principals executing an update mechanism do not update their data simultaneously. Instead, the local update of the shared secrets happens asynchronously as shown in Figure 4. An update mechanism is thus not an atomic operation, but a sequential process that relies on successful delivery of all outgoing messages to ensure synchrony of shared secrets after a protocol run.

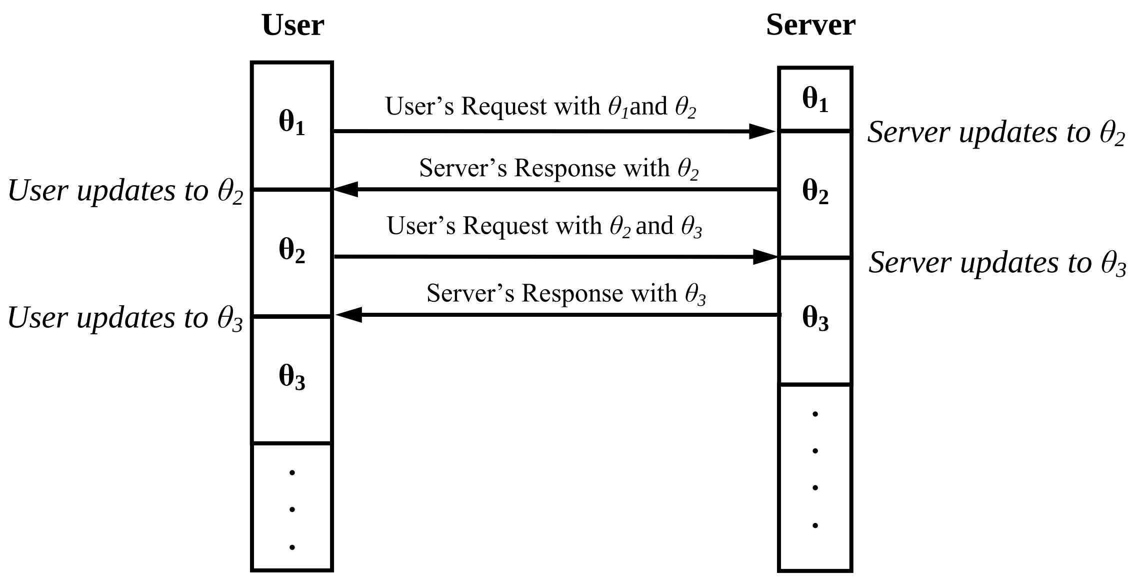
User
Server
θ1
User's Request with θ1 and θ2
θ1
Server updates to θ2
Server's Response with θ2
User updates to θ2
θ2
θ2
User's Request with θ2 and θ3
Server updates to θ3
Server's Response with θ3
User updates to θ3
θ3
θ3

Figure 4: Local update in the authentication phase.

However, the loss of messages due to accidental interference or jamming is a common threat in wireless environments (Xu, Trappe, Zhang, & Wood, 2005), for example in GSM and WLAN networks (Stahlberg, 2000), low power networks (Brodsky, & McConnell, 2009), satellite communications (Louis, & Ippolito, 2008) and Wireless Sensor Networks (Law, Hoesel, Doumen, Hartel, & Havinga, 2005). If messages that are part of the update mechanism are lost, the update mechanism may fail. As a consequence, principals may be desynchronised on their shared secrets as outlined in Figure 5. This constitutes a weakness of the update mechanism that might be exploited in a new attack. As such an attack desynchronises the principals on their stored shared secrets, we call these attacks desynchronisation attacks. A successful desynchronisation attack causes a permanent DoS condition.

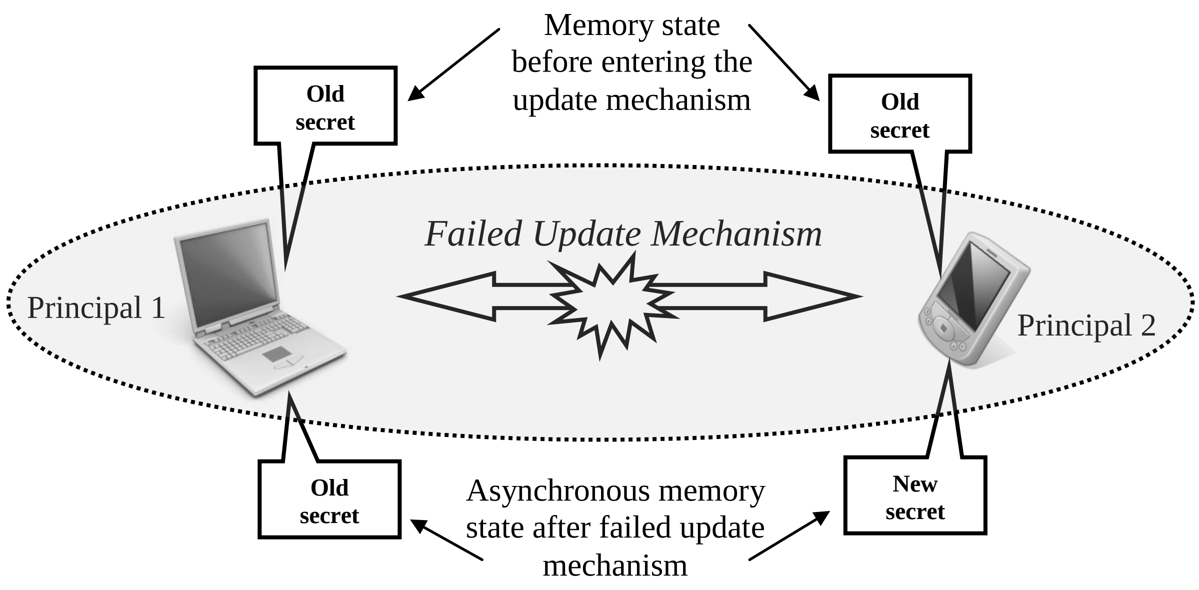
Memory state before entering the update mechanism
Old secret
Old secret
Failed Update Mechanism
Principal 1
Principal 2
Old secret
New secret
Asynchronous memory state after failed update mechanism

Figure 5. Asynchronous shared secrets after a failed update mechanism

The structure of a desynchronisation attack against an authentication phase of a protocol using an online update mechanism is outlined in Figure 6. In such an attack, the authentication phase initiates as normal: The user sends its request to the server, who in turn authenticates the request, updates its shared secret and sends its response to the user. However, if the response message is not delivered successfully to the user due to interference of an attacker, then the server and the user are desynchronised on their shared secrets: while the user still stores θ_i, the server has updated to θ_{i+1}. The user's next authentication request will fail, as it is again based upon θ_i, while the server expects a request based upon θ_{i+1}. Thus, a desynchronisation attack has been successfully mounted that causes a permanent Denial of Service condition.

Alternatively, an attacker can interfere in the same manner with the update phase. To prevent such attacks, corrective measures need to be built into an update mechanism that enable the communicating parties to recover from an asynchronous state.

Both these desynchronisation attacks are directed at individual mobile users that prevent the affected user(s) from using the satellite system even after the jamming has stopped. Any future authentication requests initiated by the affected user(s) will fail, as the server is not aware of the asynchronous state and therefore interprets the request as a replay attack using an old shared secret.

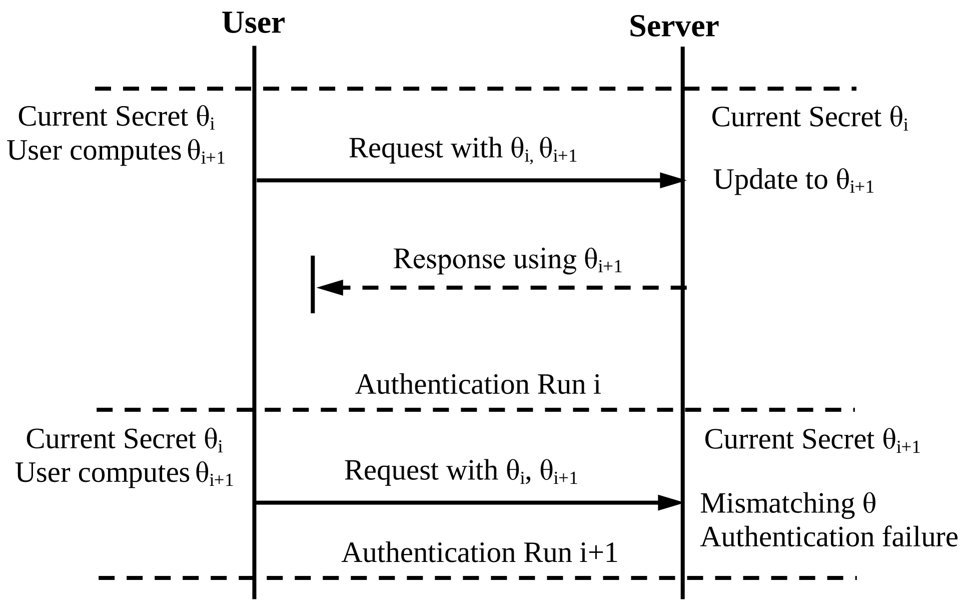
User
Server
Current Secret θ_i
User computes θ_{i+1}
Request with $\theta_{i,}$ θ_{i+1}
Current Secret θ_i
Update to θ_{i+1}
Response using θ_{i+1}
Authentication Run i
Current Secret θ_i
User computes θ_{i+1}
Request with θ_i, θ_{i+1}
Current Secret θ_{i+1}
Mismatching θ
Authentication failure
Authentication Run i+1

Figure 6. Attack structure.

Realising the Presented Desynchronisation Attacks in a Satellite Communication System

The presented desynchronisation attacks are based on jamming a single mobile user's downlink to stop messages reaching the mobile user. Such jamming can be achieved with a low-power jammer. Information on the feasibility of jamming a handset's uplink and downlink in a satellite-based communications system is presented in (Lee & Marshall, 1994), (United States General Accounting Office, 2002), (Rausch, 2006). The attack can also be realised by jamming the NCC's messages on the uplink from the gateway to the LEO. As the NCC serves multiple mobile terminals simultaneously, this affects all the mobile users executing an authentication or mobile update phase with the NCC at the time the jamming occurs. Corresponding to the previous attack, all the affected mobile users are out of synchronisation with the NCC and any of their subsequent authentication requests will be denied by the NCC. While attacking a gateway's uplink is more difficult than jamming a mobile user's downlink, the feasibility of such attacks has been presented in (Gansler & Binnendijk, 2005), (United States General Accounting Office, 2002), (Rausch, 2006).

Case Study: New Desynchronisation Attacks against an Authentication Protocol for Mobile Satellite Communications

Chang and Chang proposed a mutual authentication protocol (Chang & Chang, 2005) - hereafter referred to as the CC protocol - for mobile satellite communication systems, such as Iridium (Hartleid & Casey, 1993). These communication systems offer advanced mobility services (Comparetto & Ramirez, 1997), (Chini, Giambene, & Kota, 2010). The CC protocol was designed for the network architecture outlined in Figure 7 and aims to provide authentication between a mobile user and the Network Control

Centre (NCC) within a LEO satellite communication system. Mobile users are interconnected directly through LEO satellite links, while communication between satellites and the NCC is managed by Gateways.

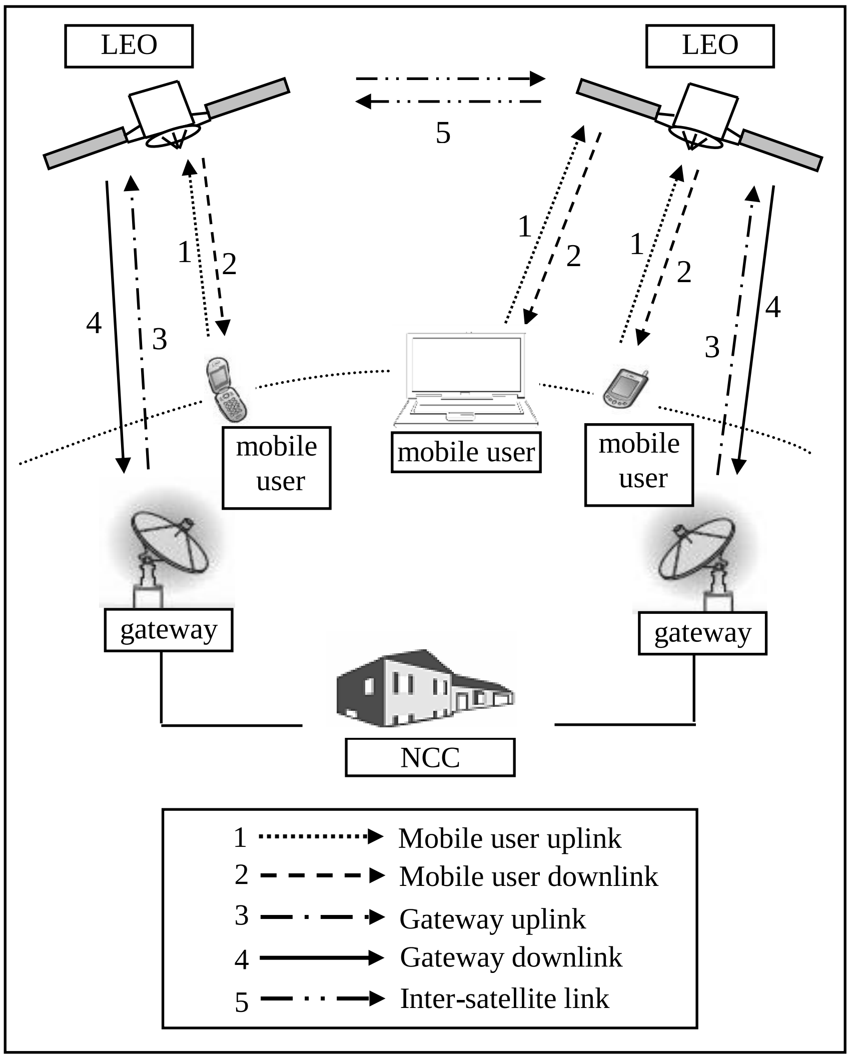

Figure 7. A LEO communications system.

The CC protocol follows the structure presented above in Figure 2 and is composed of three phases: registration, mobile authentication and mobile update. A second version for the mobile update phase was also proposed that provides for perfect forward secrecy preservation. For the analysis presented in this paper, we regard these update phases as equivalent.

- **The Registration Phase:** A new mobile user U is first registered by the gateway: U and the NCC are provided with U's secret permanent identity, U's temporary identity, the secret shared session key and the number of times (N) that the mobile user can access the service before an update phase is required.
- **The Authentication Phase**: The authentication phase is performed by U and NCC prior to any communication. By using a hash chain to create the authentication token, the same credentials/secrets are used a fixed number of times (N). Once this number has been reached, the update phase is entered.
- **The Mobile Update Phase:** On the N^{th} authentication, U and NCC enter the mobile update phase. The NCC generates and issues a new session key and a new temporary identity for the user to be used for the next N authentications.

Analysis of the CC Protocol

The notations used describe protocols throughout this paper are presented in Table 1.

U	The mobile user
LEO	Low Earth Orbit Satellite
NCC	Network Control Centre
ID_U, ID_{LEO}	User/LEO permanent identity
TID_U	User temporary identity
K	Secret key shared by U and NCC
K', TID_U'	Shared secret key/Temporary user identity newly generated by NCC
P	Authentication token
H(.)	One-way collision-resistant hash function
$\oplus$	XOR function
$\|\|$	Concatenation operator

Table 1. Notations used throughout the paper.

The CC protocol employs an update mechanism for shared secrets with a structure as discussed above (cf. Figure 2). Initially the mobile user registers with the LEO. After the registration phase, whenever the mobile user wants to access the satellite service, it emits an authentication request to the NCC using the next authentication token from a chain. If all the available authentication tokens in the chain have been used, U and NCC enter the update phase. When this phase has been completed the two parties have established a new hash chain and they can re-enter the authentication phase.

The authentication process between the authentication server NCC and the mobile user U in this protocol is based on proving possession of the current value of the authentication token P. U obtains P by hashing the concatenation of K with its permanent and temporary identities ID_U and TID_U:

$$P = H^{N-(j-1)}(K \mid\mid ID_U \mid\mid TID_U)$$

The index $N-(j-1)$ indicates the number of times that the expression $(K||ID_U||TID_U)$ is hashed to obtain the current value of the authentication token P. Thus, initially the value of P is $H^N(K||ID_U||TID_U)$ and it evolves toward $H(K||ID_U||TID_U)$ as the protocol proceeds.

While U performs the hashing of $(K||ID_U||TID_U)$ every time it enters an authentication or update phase, the NCC stores the value of $H(P)$ and updates this value after each successful completion of the authentication or update phase. In contrast, U only stores the index $N-(j-1)$ for hashing the expression $(K||ID_U||TID_U)$ and computes P when entering the authentication or update phase.

The CC protocol considers the update mechanism as an atomic operation and, thus, it is unable to handle desynchronisation of authentication tokens. However, as principals update their shared secrets asynchronously, the update mechanism is a sequential process. Consequently, the CC protocol is susceptible to desynchronisation attacks. If some messages do not reach their destination, then the authentication phase or the update phase is interrupted and the protocol reaches an undefined state as shown in Figure 8. Loss of messages can happen either by unintended interference or intentionally as a result of a jamming attack by an intruder. Jamming of messages can be achieved with a low-power jammer (Mahoney et al., 1999), (Felstead & Keightley, 1995), (Lee & Marshall, 1994).

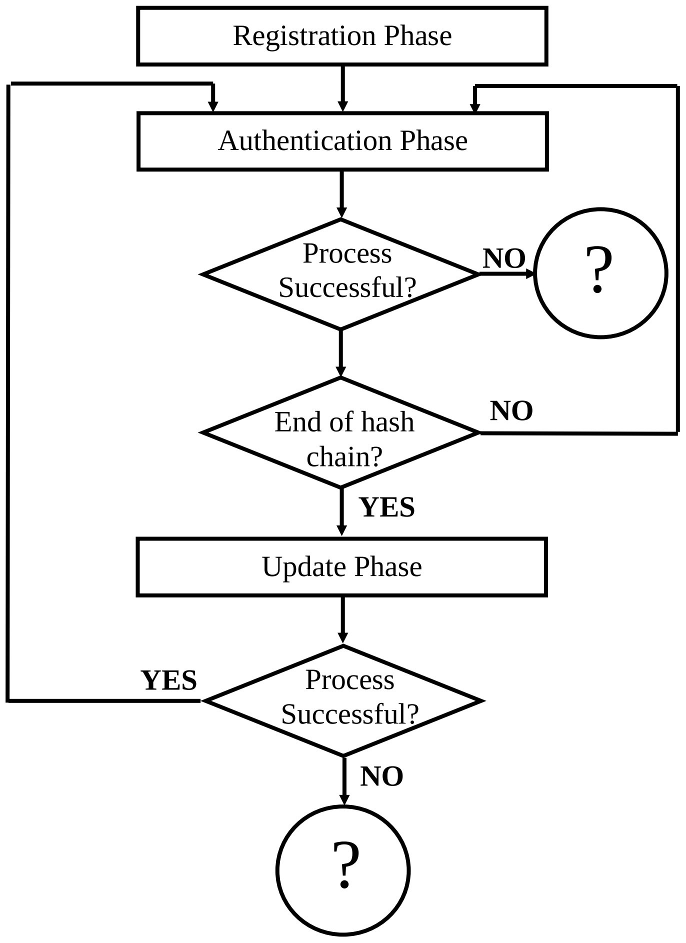

Figure 8. Reaching undefined states in the CC protocol.

In case where an undefined state is reached, the mobile user can resend the request or it can move to the next level in the chain. Both options present an opportunity to mount a desynchronisation attack. The desynchronisation attacks presented below assume that the user will resend the request. Alternatively, if the user moves to the next level in the chain, corresponding attacks can be mounted. Attempting to resynchronise by restarting the protocol from the registration phase is not feasible, as any practical registration phase is based on delivery of some hardware device (e.g. smart card).

Desynchronisation Attack on the Authentication Phase

In the authentication phase of the CC protocol, NCC and U update their shared secret information in the following way:

- The NCC updates its shared secret before sending message 4.
- U updates its shared secret after receiving message 5.

If message 5 (LEO -> U) of the authentication phase of the protocol does not reach the mobile user U, e.g. through jamming by an attacker (see Figure 9), the NCC has already updated the secret shared data, while U hasn’t updated yet. Thus, they operate asynchronously on the hash value used to authenticate each other, e.g. they are at different levels in the hash chain. Consequently, when U resends the request for authentication, this request will be rejected by the NCC as it does not match the data stored locally in its table. The NCC considers this legitimate request as an attempt by an intruder to replay an old captured request message.

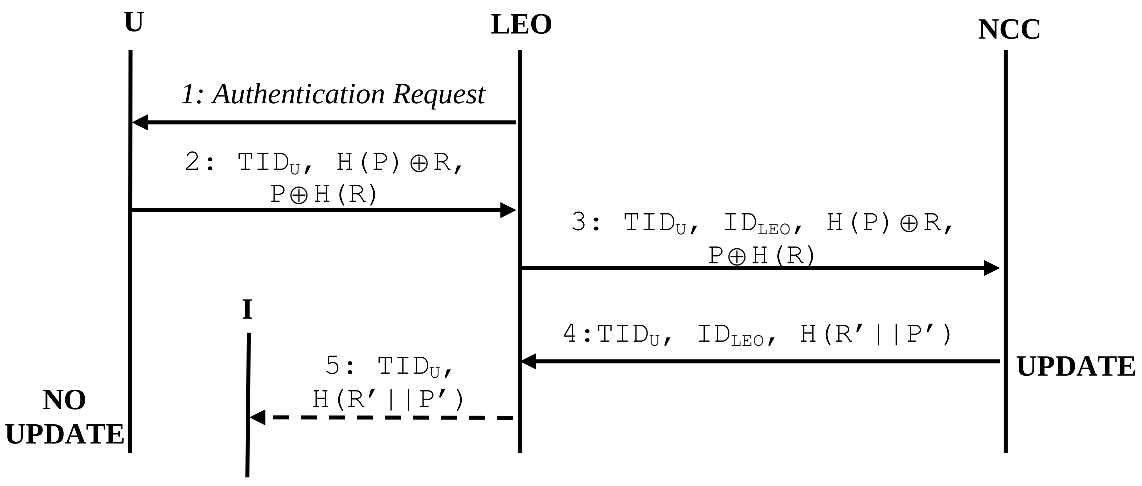
U
LEO
NCC
1: Authentication Request
2: TID_U, H(P)⊕R, P⊕H(R)
3: TID_U, ID_LEO, H(P)⊕R, P⊕H(R)
I
4:TID_U, ID_LEO, H(R'||P')
UPDATE
5: TID_U, H(R'||P')
NO UPDATE

Figure 9. A desynchronisation attack against authentication phase

Desynchronisation Attack on the Mobile Update Phase

In the mobile update phase of the CC protocol, NCC and U update their shared secret information in the following way:

- The NCC updates its shared secret before sending message 4.
- U updates its shared secret after receiving message 5.

The desynchronisation attack on the mobile update phase is outlined in Figure 10, where F1 and F2 represent the expressions $H(R'||P') \oplus K'$ and $H(R'||P') \oplus H(K'||TID_U')$ respectively.

The consequence of this attack is loss of synchronisation between U and NCC on:

- the authentication token P
- the shared secret key K
- the user's temporary identity TID_U

As a result, U and NCC operate on different hash chains. Thus, when U resends the request for authentication, this legitimate request will be rejected by the NCC.

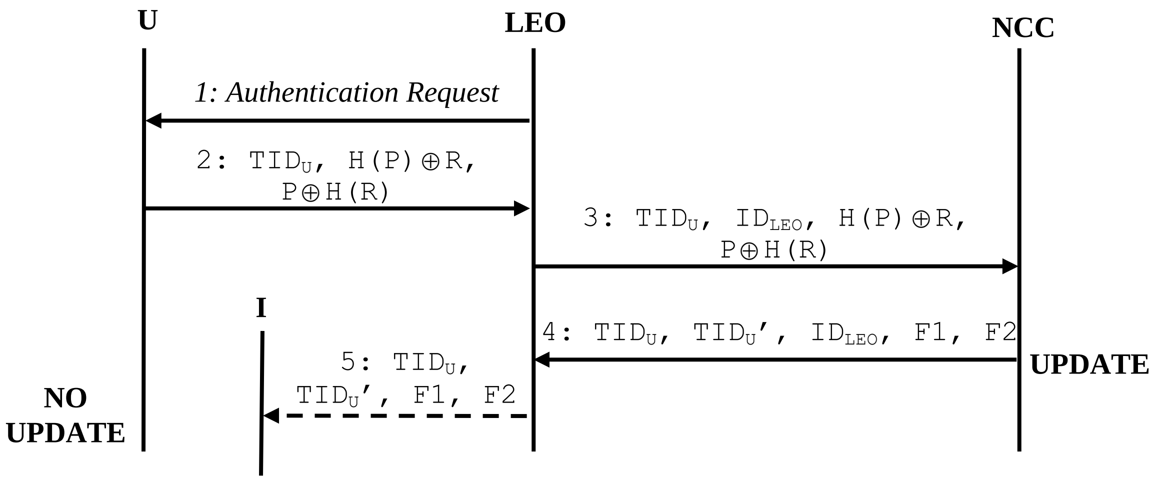
U
LEO
NCC
1: Authentication Request
2: TID_U, $H(P)\oplus R$, $P\oplus H(R)$
3: TID_U, ID_{LEO}, $H(P)\oplus R$, $P\oplus H(R)$
I
4: TID_U, TID_U', ID_{LEO}, F1, F2
UPDATE
5: TID_U, TID_U', F1, F2
NO UPDATE

Figure 10. A desynchronisation attack against mobile update phase.

A mutual Authentication Protocol with Resynchronisation Capability

The attacks presented in the previous sections highlight the possibility of affecting the security of protocols at application layer by interfering with the communication on physical channels. In this section a new security protocol for satellite communications is proposed. This protocol provides mutual authentication between a mobile user and the network control centre. Additionally the new protocol possesses a resynchronisation capability to counter the disruptive effects of desynchronisation attacks. The structure of the new protocol is shown in Figure 11.

The proposed protocol confirms synchrony between U and NCC at each authentication phase and update phase. Further, the resynchronisation phase enables the parties to recover from a desynchronised condition. Therefore, the protocol ensures that U and NCC are always able to reach a synchronous state.

Registration Phase

The proposed registration phase stores the initial shared secrets in a smart card or directly in the mobile device. In addition to any information required by the service provider the smart card contains (ID_U, TID_U, K, N) while the NCC (service provider) stores the corresponding value (ID_U, TID_U, ID_{LEO}, $H^{N+1}(K||ID_U||TID_U)$, N).

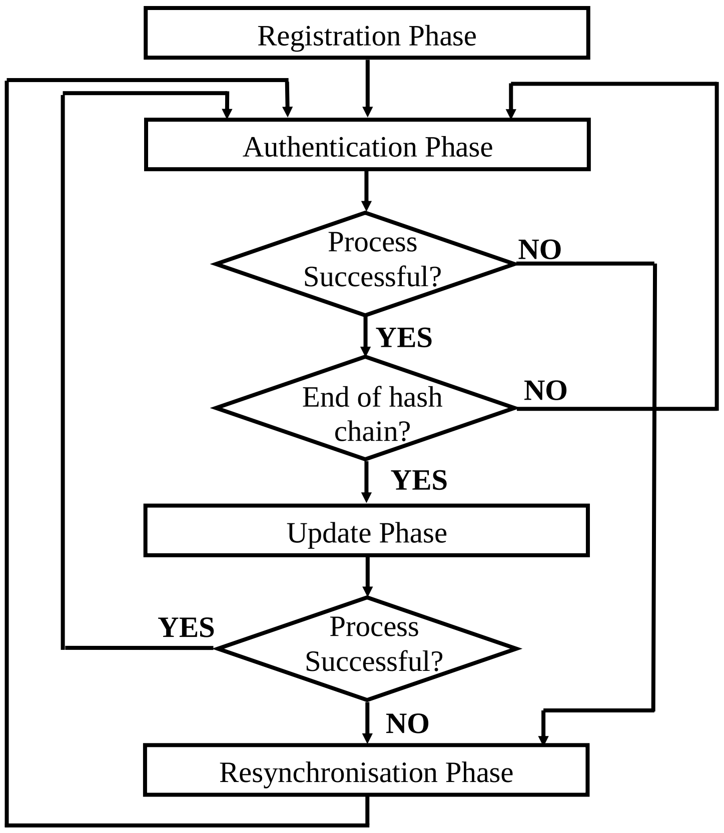

Figure 11. Structure of the mutual authentication protocol with resynchronisation capability.

Mobile Authentication Phase

The mobile authentication phase of the protocol is detailed in Figure 12. The authentication phase differentiates between legitimate authentication requests (messages 1, 2, 3, 4a, 5a) and illegitimate authentication requests (messages 1, 2, 3, 4b, 5b).

1. LEO -> U: Authentication request
2. U -> LEO: TID_U, $H(P)\oplus R$, $P\oplus H(R)$
3. LEO -> NCC: TID_U, $H(P)\oplus R$, $P\oplus H(R)$, ID_{LEO}

4.a. NCC->LEO: TID_U, ID_{LEO}, $H(R'||P')$
5.a. LEO -> U: TID_U, $H(R'||P')$

4.b. NCC -> LEO: DENY MESSAGE + RESYNC. CHALLENGE: TID_U, ID_{LEO},
$H(P_{NCC}||TID_U)$, $H(P_{NCC})\oplus R_{NCC}$
5.b. LEO -> U: DENY MESSAGE + RESYNC. CHALLENGE: TID_U,
$H(P_{NCC}||TID_U)$, $H(P_{NCC})\oplus R_{NCC}$

Figure 12. The authentication phase

After receiving the authentication request in the first message the mobile user responds with TID_U, $H(P) \oplus P$, $P \oplus H(R)$ where $P = H^{N-(j-1)}(K||ID_U||TID_U)$. When LEO receives U's response it appends its own identity ID_{LEO} to the message and forwards it to the NCC. In message 3, if ID_{LEO} is legal the NCC will lookup TID_U to retrieve $(ID_U,\ TID_U,\ ID_{LEO},\ H^{N+1-(j-1)}(K||ID_U||TID_U),\ (N-(j-1)))$. Subsequently the NCC computes the following components:

$R' = H^{N+1-(j-1)}(K||ID_U||TID_U) \oplus (H(P) \oplus R)$

$P' = H(R') \oplus (P \oplus H(R))$

$H(P')$

The NCC then checks if $H(P')$ equals $H^{N+1-(j-1)}(K||ID_U||TID_U)$. If the check holds, the NCC updates the corresponding entry in the lookup table to $(ID_U,\ TID_U,\ ID_{LEO},\ P',\ (N-j))$. In message 4 NCC sends $H(R'||P')$, ID_{LEO}, TID_U to LEO. LEO then forwards TID_U, $H(R'||P')$ to U in message 5 of the protocol. U verifies the validity of the equation: $H(R||P) = H(R'||P')$. If this proves to be true, U accepts that NCC is authentic and updates the stored secret data to $(ID_U,\ TID_U,\ K,\ (N-j))$.

On the other hand, if the NCC receives an incorrect authentication request, it responds with a resynchronisation challenge (Messages 1, 2, 3, 4b and 5b). This resynchronisation challenge contains $H(P_{NCC}||TID_U)$, $H(P_{NCC}) \oplus R_{NCC}$, where P_{NCC} is the NCC's currently stored authentication token, R_{NCC} is a newly generated random number and TID_U is the user's temporary identity.

On receiving a resynchronisation challenge, the mobile user compares the received P_{NCC} with the remaining P_U values in the chain. If a match is found, the mobile user advances to the resynchronisation phase. On the other hand, if no match is found the resynchronisation request is deemed illegitimate and the same authentication request is resent.

Mobile Update Phase

Analogous to the authentication phase, the mobile update phase in Figure 13 also distinguishes between legitimate (Messages 1, 2, 3, 4a and 5a) and illegitimate authentication tokens (Messages 1, 2, 3, 4b and 5b). On receiving message 3 the NCC generates new data TID_U' and K'. If the received authentication token is correct, the NCC sends the updated TID_U' and K' via the LEO to the user. However, to counter the disruptive effects of any jamming attacks, the NCC stores the old value $H^2(K||ID_U||TID_U)$ until the first legitimate authentication request including the new data is received from U, which proves that U has updated successfully. Meanwhile, any illegitimate authentication request is treated as an unsuccessful mobile update request.

1. LEO -> U: Authentication request
2. U -> LEO: TID_U, $H(P) \oplus R$, $P \oplus H(R)$
3. LEO -> NCC: TID_U, $H(P) \oplus R$, $P \oplus H(R)$, ID_{LEO}

4.a. NCC -> LEO: TID_U, TID_U', ID_{LEO}, $H(R' || P') \oplus K'$,
$H(R' || P') \oplus H(K' || TID_U')$

5.a. LEO -> U: TID_U, TID_U', $H(R' || P') \oplus K'$,
$H(R' || P') \oplus H(K' || TID_U')$

4.b. NCC -> LEO: DENY MESSAGE + RESYNC.CHALLENGE:
TID_U,ID_{LEO},$H(P_{NCC} || TID_U)$, $H(P_{NCC}) \oplus R_{NCC}$

5.b. LEO -> U: DENY MESSAGE + RESYNC. CHALLENGE: TID_U,
$H(P_{NCC} || TID_U)$, $H(P_{NCC}) \oplus R_{NCC}$

Figure 13. Mobile update phase

On detecting an illegitimate authentication token, the NCC assumes the mobile user is still operating on the old hash chain. Therefore, the resynchronisation challenge issued by NCC contains the last hash value from the old chain ($H^2(K||ID_U||TID_U)$). On receiving a resynchronisation challenge, the mobile user compares the received P_{NCC} with its current authentication token P_U. If these match the mobile user enters the resynchronisation phase. On the other hand, if these values do not match the resynchronisation request is deemed illegitimate and the same mobile update request is resent.

Resynchronisation Phase

The resynchronisation phase presented in Figure 14 ensures communication can continue from a safe state if synchronisation of the hash chain is lost between mobile users and the NCC. The resynchronisation phase differentiates between resynchronising the authentication phase (Messages 1, 2, 3a and 4a) and resynchronising the mobile update phase (Messages 1, 2, 3b and 4b).

1. U -> LEO: TID_U, $P_{NCC} \oplus H(R_U)$, $H(P_{NCC} || R_{NCC}) \oplus R_U$
2. LEO -> NCC: TID_U, ID_{LEO}, $P_{NCC} \oplus H(R_U)$, $H(P_{NCC} || R_{NCC}) \oplus R_U$

3\.a. NCC -> LEO: TID_U, ID_{LEO}, $H(R_U' || P_{NCC}')$
4\.a. LEO -> U: TID_U, $H(R_U' || P_{NCC}')$

3\.b. NCC -> LEO: TID_U, TID_U', ID_{LEO}, $H(R_U' || P_{NCC}') \oplus K'$,
$H(R_U' || P_{NCC}') \oplus H(K' || TID_U')$
4\.b. LEO -> U: TID_U, TID_U', $H(R_U' || P_{NCC}') \oplus K'$,
$H(R_U' || P_{NCC}') \oplus H(K' || TID_U')$

Figure 14. Resynchronisation Phase

In the resynchronisation phase, U issues the authentication message $\mathtt{TID_U}$, $\mathtt{P_{NCC} \oplus H(R_U)}$, $\mathtt{H(P_{NCC}||R_{NCC}) \oplus R}$, where $\mathtt{P_{NCC}}$ is the agreed hash value, $\mathtt{R_U}$ is a newly generated random number by U and $\mathtt{R_{NCC}}$ is the random number received from the NCC in the resynchronisation request.

On receiving this authentication message, the NCC checks that $\mathtt{R_{NCC}}$ and $\mathtt{H(P_{NCC})}$ have the correct values. If any one of these values is incorrect, the message is considered illegitimate and the NCC resends the previous resynchronisation challenge, where the random number $\mathtt{R_{NCC}}$ is newly generated. Otherwise, the resynchronisation phase continues as follows:

In case of resynchronisation of the authentication phase, the NCC computes $\mathtt{H(R_U'||P_{NCC}')}$ and sends it via the LEO to the mobile user. On receiving message 4a, U checks $\mathtt{H(R_U||P_U)}$ equals $\mathtt{H(R_U'||P_{NCC}')}$. If this is true the mobile user updates its secret data from $\mathtt{(ID_U,\ TID_U,\ K,\ m)}$ to $\mathtt{(ID_U,\ TID_U,\ K,\ i)}$, where $\mathtt{m}$ is U's level in the hash chain before resynchronisation and $\mathtt{i}$ is NCC's level in the hash chain.

In case of resynchronisation of the mobile update phase, the NCC computes $\mathtt{H(R_U'||P_{NCC}') \oplus K'}$ and $\mathtt{H(R_U'||P_{NCC}') \oplus H(K'||TID_U')}$ and sends it via the LEO to the mobile user. On receiving message 4b, U retrieves $\mathtt{K'}$ from $\mathtt{H(R_U'||P_{NCC}') \oplus K'}$ using the stored $\mathtt{H(R_U||P_{NCC})}$. U recalculates the expression $\mathtt{H(K'||TID_U')}$ to confirm integrity of the $\mathtt{TID_U'}$. If the integrity is established, U updates its table to $\mathtt{(ID_U,\ TID_U',\ K',\ N)}$.

Security Analysis of the Proposed Protocol

In the situation in which all the messages are successfully delivered, the security of the proposed protocol is equivalent to the CC protocol (Chang & Chang, 2005). The following security analysis of the proposed protocol establishes the effectiveness of the protocol when messages are lost by unintended interference or by a deliberate attack.

Triggering the Resynchronisation Process

The NCC has the responsibility of issuing the resynchronisation challenge with its current data and the resynchronisation adjustments are left to the mobile user. This approach is chosen as activating the resynchronisation process has a significant impact on the performance of the entire communication system. Further, the NCC serves multiple users and performs other complex tasks, such as call routing and call management functions and it should not be overly burdened with additional resynchronisation tasks.

Replay Attacks

Replaying an Old Resynchronisation Challenge on Behalf of NCC. An attempt by an intruder to replay an old resynchronisation challenge to mislead the mobile user into modifying its position in the hash chain will fail, as the user will search for the received P_{NCC} of the resynchronisation challenges within the remaining authentication tokens P_U in the chain. If this value is not found, then the user will ignore the resynchronisation request.

Replaying an Old Answer to a Resynchronisation Challenge on Behalf of U. In the resynchronisation phase, the mobile device sends the random number R_{NCC} in message 1, where R_{NCC} is received from NCC in the resynchronisation challenge. Replaying message 1 of the resynchronisation phase would be unsuccessful, as the NCC discards messages with incorrect R_{NCC} values.

Replaying NCC's Response in the Resynchronisation Phase. NCC's response in the resynchronisation phase contains the random number R_U submitted by mobile user U, who uses this value to establish freshness of the response. Thus, replayed messages can be identified and discarded.

Forging a Resynchronisation Challenge

An attacker may also attempt to forge the resynchronisation challenge by replacing the last component $H(P_{NCC}) \oplus R_{NCC}$ with another custom value. This will fail as the mobile user will search for the received P_{NCC} of the resynchronisation challenges within the remaining authentication tokens P_U in the chain. If this value is not found, then the user will ignore the resynchronisation request.

Dictionary Attacks

The mobile user generates a new random number R_U for message 1 in the resynchronisation challenge, rather than using the random number R used in the message 2 of the current authentication phase. This prevents a dictionary attack using the fields $P \oplus H(R)$ and $P_{NCC} \oplus H(R_U)$ of these two messages.

Desynchronisation Attacks

As detailed in section 4, there are two messages in the CC protocol vulnerable to desynchronisation attacks: the last message in the authentication phase and the last message in the mobile update phase. These vulnerabilities have been countered by introducing a resynchronisation phase. Once the jamming is stopped, this new phase will ensure that the two parties are resynchronised.

If an intruder jams either of these two messages, NCC and U are temporarily desynchronised, as the NCC advances through the hash chain, while U remains at the current level. However, on the next authentication attempt by the user, the NCC will detect desynchronisation and will issue a resynchronisation challenge, which leads to synchronised secret data shared by NCC and U.

Conclusions

The increased transmission of sensitive information over wireless networks drives the need for improved security protocols. Many peer-to-peer security protocols proposed for wireless communications employ online update mechanisms to facilitate the use of one-time shared secrets. These update mechanisms serve two purposes: Firstly, the generation of a new instance of the shared secret and, secondly, agreement on the same new shared secret by all communicating parties.

In this paper, the use of online update mechanisms for the generation and distribution of shared secrets was analysed. The presented analysis revealed a new form of attack (which we call desynchronisation attacks) against security protocols employing such update mechanisms. These new attacks desynchronise the shared secrets stored by communicating peer principals thereby causing a permanent DoS condition. A case study demonstrated the mounting of two desynchronisation attacks against a mutual authentication protocol for mobile satellite communications. As a consequence of the attacks, the targeted users are denied access to the communication service and a permanent denial of service condition is reached that persists even after the interference ends.

To counter the disruptive effects of desynchronisation attacks, a new mutual authentication protocol with resynchronisation capability was proposed. The new protocol provides mutual authentication between a mobile user and the satellite Network Control Centre. Additionally, the protocol has a resynchronisation phase that is initiated by the NCC whenever it suspects desynchronisation with a mobile user. This resynchronisation phase re-establishes synchrony between the communicating parties. Thus, the possibility of causing a permanent DoS condition by mounting a desynchronisation attack is eliminated. A security analysis of the proposed protocol established its resistance against attacks such as replay attacks, dictionary attacks and desynchronisation attacks.

Acknowledgements

This work was funded by the Irish Research Council for Science, Engineering and Technology (IRCSET Embark Initiative) and Science Foundation Ireland - Research Frontiers Programme (SFI RFP07CMSF 631).

References

Bargh, M. S., Hulsebosch, R. J., Eertink, E. H., Prasad, A, Wang, H. & Schoo, P. (2004). Fast authentication methods for handovers between IEEE 802.11 wireless LANs. *Proceedings of the 2nd ACM international workshop on Wireless mobile applications and services on WLAN hotspots, Pages: 52-60.*

Brodsky, J. & McConnell, A. (2009). Jamming and interference induced Denial of Service attacks on IEEE 802.15.4 based wireless networks. *SCADA Security Scientific Symposium, January 21-22, 2009, Miami Beach, Florida.*

Chang, YF. & Chang, CC. (2005). An efficient authentication protocol for mobile satellite communication systems. *ACM SIGOPS Operating Systems Review, Vol. 39, Issue 1 (January 2005), 70-84.*

Chen, TH., Lee, WB. & Chen HB. (2008). A self-verification authentication mechanism for mobile satellite communication systems. *Computers and Electrical Engineering, Vol. 35, Issue 1 (January 2009), 41-48.*

Chini, P., Giambene, G. & Kota, S. (2010). A survey on mobile satellite systems. *International Journal of Satellite Communicaitons and Networking, Vol. 28, Issue 1, Pages 29-57, Wiley & Sons.*

Comparetto, G. & Ramirez, R. (1997). Trends in mobile satellite technology. *IEEE Computer, Vol. 30, Issue 2, 44-52.*

Dojen, R., Lasc, I. & Coffey, T. (2008). Establishing and Fixing a Freshness Flaw in a Key-Distribution and Authentication Protocol. *IEEE 4th International Conference on Intelligent Computer Communication and Processing,* 28-30 August, Cluj-Napoca, Romania.

Felstead, E.B. & Keightley, R.J. (1995). Robustness capabilities of transponded commercial satellite communications. *Conference Record, Military Communications Conference, IEEE MILCOM 95, 7 Nov. 1995, Vol. 2, Pp. 783-787, San Diego, CA, USA.*

Gansler, J.S. & Binnendijk, H. (Ed.). (2005). *Information assurance: Trends in vulnerabilities, threats and technologies*. National Defense University, Center for technology and national security policy, Washington, D.C., ISBN 1-57906-069-2, January 2005.

Hartleid, J.E. & Casey, L. (1993). The Iridium system personal communications anytime, anyplace. *Proceedings of the Third International Mobile Satellite Conference, Pasadena (June 1993), 285-290.*

Law, YW., Hoesel, L. Doumen, J., Hartel, P. & Havinga, P. (2005). Energy-efficient link-layer jamming attacks against wireless sensor network MAC protocols. *In proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks, Pages 76-88, ISBN:1-59593-227-5, Alexandria, VA, USA.*

Lee, J.W. & Marshall, V.A. (1994). Maximum capacity prediction and anti-jamm performance analysis for commercial satellite communication systems. *Conference Record, Military Communications Conference, IEEE MILCOM 94, Oct. 1994, Pp. 506-510, Forth Monmouth, NJ.*

Lee, TF, Chang, SH., Hwang, T. & Chong, SK., (2009). Enhanced Delegation-Based Authentication Protocol for PCSs. *IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, VOL. 8, NO. 5, MAY 2009.*

Lee, WB. & Yeh, CK. (2005). A new delegation-based authentication protocol for use in portable communication systems. *IEEE Transactions on Wireless Communications, Vol. 4, No. 1, Jan. 2005.*

Louis, J. & Ippolito, Jr. (2008). *Satellite communications systems engineering: atmospheric effects, satellite link design and system performance*. Wiley Series on Wireless Communications and Mobile Computing, 2008.

Mahoney, T., Kerr, P., Felstead, B., Wells, P., Cunningham, M., Baumgartner, G. & Jeronim, L. (1999). An investigation of the military applications of commercial personal satellite-communications systems. *Military Communications Conference Proceedings 1999, MILCOMM 1999 IEEE. Volume 1, 31 Oct.-3 Nov. 1999, 112 – 116.*

Peng, T., Leckie, C. & Ramamohanarao, K. (2007). Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Computing Surveys, Volume 39, Issue 1, Article No. 3.*

Radosavac, S., Benammar, N., Baras & John S. (2004). Cross-layer attacks in wireless ad hoc networks. *Conference on Information Sciences and Systems, Princeton University*, March 17 - 19, 2004.

Rausch, H. (2006). Jamming commercial satellite communications during wartime: an empirical study. *Proceedings of the Fourth IEEE International Workshop on Information Assurance, Pp. 109-118. IEEE Computer Society, Washington, DC, USA.*

Stahlberg, M. (2000). Radio jamming attacks against two popular mobile networks. *In proceedings of the Helsinki University of Technology Seminar on Network*

Security and Mobile Security, H. Lipmaa and H. Pehu-Lehtonen, Eds. Helsinki University of Technology.

Tseng, YM. (2007). A heterogeneous-network aided public-key management scheme for mobile ad hoc networks. *International Journal of Network Management, 17: 3–15.*

United States General Accounting Office, (2002). *Critical Infrastructure Protection: Commercial Satellites Should Be ore Fully Addressed*. GAO Report to the Ranking Minority Member, Permanent Subcommittee on Investigations, Committee on Government Affairs, U.S. Senate August 2002.

Xu, W., Trappe, W., Zhang, Y. & Wood, T. (2005). The feasibility of launching and detecting jamming attacks in wireless networks. *International Symposium on Mobile Ad Hoc Networking & Computing, Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing, Urbana-Champaign, IL, USA, Pages: 46 – 57, 2005.*