Automated Analysis of Some Security Mechanisms of SCEP*

Fabio Martinelli, Marinella Petrocchi, and Anna Vaccarelli

Istituto per l'Informatica e la Telematica
Consiglio Nazionale delle Ricerche
Via G.Moruzzi 1 - Pisa, Italy.
{Fabio.Martinelli, Marinella.Petrocchi, Anna.Vaccarelli}@iit.cnr.it

Abstract. The paper analyzes SCEP, the Simple Certificate Enrollment Procedure, a two-way communication protocol to manage the secure emission of digital certificates to network devices. The protocol provides a consistent method of requesting and receiving certificates from different Certification Authorities by offering an open and scalable solution for deploying certificates which can be beneficial to all network devices and IPSEC software solutions. We formally analyze SCEP through a software tool for the automatic analysis of cryptographic protocols able to discover, at a conceptual level, attacks against security procedures. Our method of survey contributes towards a better understanding of the structure and aims of a protocol both for developers, analyzers and final users.

1 Introduction

The aim of this paper is to highlight how formal methods can be useful to better define the security mechanisms and aims of security protocols and to offer a formal description of SCEP[1], the Simple Certificate Enrollment Protocol. To the best of our knowledge, this is the first attempt to give a formal description of this protocol, and we based our study on the Internet Draft [7][2].

The SCEP protocol gives specifications for digital certificate enrollment, access and revocation, for certificates and CRL queries. SCEP was developed for the distribution of digital certificates to network devices such as routers and gateways. It seems to offer a valid support for the development of Virtual Private Networks - VPNs - which are communication networks realized on a public infrastructure such as the Internet. Using a VPN involves encrypting data before sending it through the public network and decrypting it at the receiving end, in order to achieve one or more of the following goals:

* Work partially supported by Microsoft Research Europe (Cambridge); by MIUR project "MEFISTO: Metodi Formali per la Sicurezza ed il Tempo"; by MIUR project "Tecniche e Strumenti Software per l'Analisi della Sicurezza delle Comunicazioni in Applicazioni Telematiche di Interesse Economico e Sociale"; by CNR project "Strumenti, Ambienti ed Applicazioni Innovative per la Società dell'Informazione"; by CSP project "SeTAPS: Strumenti e Tecniche per l'Analisi di Protocolli di Sicurezza".

[1] SCEP is the evolution of specifications developed by Verisign Inc. and Cisco Systems and it is commercially available in both client and CA implementations.

[2] Released on May 15, 2002, it will expire on November 15, 2002. As declared by the same authors, it has to be considered as a "work in progress".

– connect users securely to their own corporate access (remote access);
– link branch offices to an enterprise network (intranet);
– extend existing infrastructure to include partners and customers (extranet).

Ideally, a VPN should behave similarly to a private network. Looking at the goals of a VPN, authentication ensures the identity of all communicating parties, may they be individual or computer resources. To correctly identify a communicating party, VPNs typically use preshared keys, but could use digital certificates, [5]. Digital certificates were born in the field of public key cryptosystems, [14]: roughly, public key cryptosystems consist of an encryption function *E*, a decryption function *D* and a pair of keys. Each pair consists of a public key k and a private key k^{-1} and each user of a public key cryptosystem holds his own pair. What is encoded with one key, can only be decoded with the other. The adjective "public" means that everyone (non only the owner of the key pair) is allowed to know the public key and so to use it. The adjective "private" means that only the user that holds that pair knows the private key which remains a closely guarded secret. These cryptographic systems may be used to guarantee both confidentiality and authentication of origin:

– Confidentiality: the sender of a message m can encrypt it with the receiver's public key k; the receiver will decrypt it with his private key k^{-1}.
– Authentication of origin: the sender encrypts message m with his private key k^{-1}; the receiver decrypts the encrypted message with the sender's public key. Authentication of origin should be guaranteed by the fact that only the sender knows the private key and thus only he could generate the encryption. These concepts are the basis for digital signature schemes, e.g. [13].

Basically, a digital certificate is an electronic document that links an identity (i.e. a person or a machine) and a public key. It is issued by a Certification Authority (CA) that can vouch for an individual identity. The way CA vouches for such links is to digitally sign the issued certificate with CA's private key. Typically, a digital certificate contains a public key, information specific to the user (a name, a company, an IP address, etc.), information specific to the Certification Authority issuer, a validity period (starting date - finishing date) and additional management information.

The paper is organized as follows. In Section 2, we present the usual scenario of security protocols and give an idea of our analysis method. In Section 3, we describe the structures of messages exchanged between parties during the phase of SCEP enrollment. Section 4 highlights the motivations for the need of some forms of correctness checks in SCEP specifications; without them, the protocol would be vulnerable to attack by adversaries. Finally, Section 5 gives some conclusions. Below, we summarize the main contributions of this paper.

1.1 Contributions

The main contributions of this paper are the following:

- (Part of) the SCEP Protocol has been formally analyzed and the results are reported in this paper. With regard to the security properties listed in the draft we did not find attacks, at least in a limited analysis scenario consisting of a finite number of participants. While this does not suffice to ensure the absolute security of the protocol in all circumstances, it does enhance the reliability of SCEP. However, we noticed a vulnerability concerning the emission of two digital certificates with the same subject name - public key binding.
- The application of automated verification tools is useful to better understand how certain mechanisms and checks ensure certain security features of communication protocols. This might be useful in revealing causes of omitted or erroneously implemented security checks. It might be also useful to render more comprehensive for those less expert some statements written in technical documents, where often it is asserted that a security check is necessary but rarely is the reason given. In order to understand the reason, we simply omit the security check in the description of the protocol, run the verification tool and wait for the result of the analysis. This methodology is particularly useful to study security protocols in a formal way by suitably changing the protocol description in order to simulate possible faults and check the relative effect (as in the common "Failure Model and Effect Assessment (FMEA)" adopted in software engineering). Our next step will be to systematically create such case studies in the security protocol analysis framework as done in [2] for safety in critical systems.

2 Analysis approach

In standard communication protocols all interactions are achieved by means of communications, i.e. parties communicate with each other by exchanging a sequence of messages on channels of the net. Talking about security communication protocols, cryptographic functions are introduced in the structures of messages in order to guarantee the satisfaction of certain security properties. Common security properties are *secrecy* (i.e. confidentiality of exchanged messages: nobody except the legitimate participants should know the content of an exchanged secret message), *message authenticity* (i.e. no alteration of the content of a message) and *entity authentication* (i.e. capability of identifying other parties during a communication). Given the sensitive nature of the information possibly exchanged in the run of a protocol (e.g. credit card numbers, passwords, current account numbers) it appears reasonable to consider the presence in the

net of third malicious parties: such intruders try to interfere with the normal execution of the protocol in order to achieve advantages for themselves. Due to the unpredictable behavior of such intruders, the design of distributed security protocols is very challenging. Several protocols that have appeared in scientific literature have been found to be flawed (e.g., see [8, 15]) and this also happens by assuming the underlying cryptographic primitives reliable. Indeed, we may have some subtle attacks depending on how messages are exchanged over the network. Several methodologies for the analysis of security protocols have been developed in recent years, e.g. [1, 3, 9, 11].

For our analysis we consider honest participants S of a protocol plus an intruder X, as a *compound system* $S|X$, i.e. a system whose components run in parallel and that can interact. Basically, our aim consists in verifying that a certain security property is satisfied by the compound system. We investigate whether, for every possible active intruder X, $S|X$ satisfies that property.

An operational language is adopted for the description of the protocols. It is a CCS process algebra style language, [12], with slight variations for the treatment of cryptographic functions, [10]. We do not need *a priori* any intruder specification and moreover we make no assumption on its behavior, X can be any term of the algebra. As the other protocol participants, X has a set of message manipulating rules that are used to model cryptographic functions such as encryption and decryption. The intruder has the capability to send and receive messages over net channels to and from other components of the system and also intercept and fake messages. He derives new messages from his initial knowledge: the set of messages we assume the intruder is in possession of at the beginning of the computation and other intercepted information obtained during the run of the protocol. Data encryption is assumed to be "opaque", i.e. a message encrypted with the public key of *i* cannot be decrypted by anyone but the person who knows the correspondent private key (unless the decryption key is compromised). An adversary can intercept and store an encrypted message and replay it later, but the structure of the message is not accessible to him, i.e. he cannot split the encrypted message unless he knows the decryption key. The intruder's knowledge grows as the computation evolves: we consider whether this knowledge satisfies, at a certain point of the computation, a predicate based on a specific security property. In the case of a positive answer, our analysis reports an attack against the security property. In other words, an attack is possible when X is able to force the execution of protocol to be performed in a different way, by cheating the honest parties. The development of the theory has lead to the implementation of a tool for the analysis of distributed systems with finite behavior, the Partial Model Checking Security Analyzer (PAMOCHSA), [4, 6].

Following, we give the standard notation with which security protocols are usually specified in literature - for information about the operational language used to describe the protocol specifications we refer to [10] - in this paper we prefer to follow a more intuitive notation.

We then consider, with a simple example, the uncertainty about protocol specifications correctness. We will show how the security properties of a protocol may be broken without implementing cryptanalysis.

2.1 Notation

We consider a set of agents able to send and receive messages. We represent the sending and reception of a message in this way: $i \quad A \mapsto B : msg$, where msg is the exchanged message, i is the i-th communication channel, on which the exchange takes place. A and B are the sender and the receiver of msg.

As already outlined, since the net is public, we have to consider the presence of a malicious agent X, that can intercept and fake messages:

$$(1) \quad X(A) \mapsto B : msg$$
$$(2) \quad A \mapsto X(B) : msg$$

Notation (1) describes intruder X that sends a message msg to party B pretending to be party A (forgery); (2) denotes: msg, originally intended for B, is actually intercepted by X (interception).

We will use the following notation throughout the paper:

$$\begin{array}{ll}
name_i, pin_i, etc & := Name\ of\ party\ i,\ password\ of\ party\ i,\ etc. \\
pk_i, pk_i^{-1} & := respectively,\ public\ and\ private\ key\ of\ party\ i \\
\{...\}_{pk_i^{-1}} & := message\ signed\ by\ party\ i \\
\{...\}_{pk_i} & := message\ encrypted\ by\ public\ key\ of\ party\ i \\
\{...\}_{KEY} & := message\ encrypted\ by\ symmetric\ key\ KEY \\
Hash\{...\} & := MD5\ fingerprint
\end{array}$$

Hash functions are a family of functions with the following main properties: i) they take as input a message of arbitrary length and produce as output a fixed length "fingerprint" of the input; ii) they are not-reversible (with high probability); iii) it is computationally infeasible to produce two messages having the same fingerprint or to produce any message having a given prespecified target fingerprint. The *Hash* function used in the SCEP implementation is MD5. We refer to the output of the MD5 *Hash* function as *MD5 fingerprint*.

2.2 A simple example

Consider the following example, where a user A wants to send to his $Bank$ an order to move money on the account of another user X. A may send the message "*move $1000 to X's account*" signed with its private key pk_A^{-1}:

$$A \mapsto Bank : \{\text{move \$1000 to X's account}\}_{pk_A^{-1}}$$

In communication networks, it is not possible to determine the origin of messages. This must be deduced by the content of the message itself. Since this message is signed with the private key of A, the $Bank$ should be sure that the message has been originated by A and the order is legitimate. Thus, the $Bank$ makes the money transfer. Suppose that X eaves-drops on this message. He can send it again to the $Bank$, i.e.:

$$X(A) \mapsto Bank : \{\text{move \$1000 to X's account}\}_{pk_A^{-1}}$$

The $Bank$ is unable to be sure of the sender of the message, and the signature of this message is still valid. Eventually, X gets \$2000. Thus, even without breaking cryptography, the protocol is attacked.

3 The SCEP Enrollment Phase

We formally describe the Simple Certificate Enrollment Procedure (SCEP), a two-way communication protocol whose goal is the secure issuance of certificates to network devices, such as routers and gateways, using existing technology. Up to today, the last document describing SCEP is an Internet Draft available on Internet at [7]. The protocol is one of the first to be adopted by numerous vendors because it offers a common method of enrolling (i.e. requesting and receiving digital certificates) from different Certification Authorities.

SCEP supports the following operations:

- CA public key distributions;
- Certificate Enrollment;
- Certificate Revocation;
- Certificate and CRL query.

In the following, we mainly focus our attention on the enrollment phase. An enrollment procedure consists of two main phases:

- the SCEP client, identified by a subject name consisting of the Fully Qualified Domain Name (i.e. alice.somewhere.com), asks for a digital certificate. It composes its certificate request and sends it to a Certification Authority Server (CA), an entity which issues certificates and whose name will be declared in the certificate issuer name field.
- The Certification Authority tests the correctness of the received request[3]; in case of positive outcome, CA issues the certificate, digitally signs it and sends it to the applicant.

The secrecy and integrity of the private key of the Certification Authority must be preserved. (This is why the CA Server should be an off-line machine. In this

[3] In subsection 3.2 it is explained how CA tests the correctness of the request.

case, communications between CA and the applicant pass through an on-line Registration Authority Server RA. In this paper, CA Server and RA Server are specified as one entity).

We will refer to SCEP client as user U. U generates its pair of asymmetric keys with a specific key usage (i.e. only for encryption, only for digital signature, or both). The public key to be certified and the key usage are conveyed to CA through the certificate enrollment request.

3.1 User Certificate Request

After the generation of the keys and having obtained the CA's certificate, necessary to retrieve CA's Public Key in order to enroll, the user generates its request using PKCS#10 and sends it to the CA exploiting PKCS#7. PKCS#10 and PKCS#7 were issued by RSA Labs and made public and modifiable as with PKCS#i (Public-Key Cryptography Standards). They are "de facto" standards: PKCS#10 describing the syntax for certification requests and PKCS#7 defining formats to represent data with the addition of cryptographic information, i.e. encrypted data or digital signatures. PKCS#7 provides different kinds of formats, like Signed Data (data plus digital signatures), Enveloped Data (encrypted data plus encrypted key by means of RSA), Degenerated Mode (for the distribution of certificates). Briefly, a PKCS#10 request can be formalized as follows:

$$PKCS\#10 := \{name_U, pk_U, pin_U, \{name_U, pk_U, pin_U\}_{pk_U^{-1}}\}$$

where $name_U$ is the Subject Name of the user U (Fully Qualified Domain Name plus IP Address), pk_U is the public key to be certified and pin_U is a secret that associates the subject name to that certificate request[4].

PKCS#10 is completed by adding the digital signature of the 3-tuple Subject Name, Public Key and Pin, using the User Private Key. This signature acts as a Proof of Possession (POP), i.e. once CA has verified the signature, it has proof that whoever originated the signature holds the corresponding private key. The key usage is specified in the PKCS#10; in our formalization the key usage is not explicitly declared, the usage is intended for both encryption and signature.

Upon composing PKCS#10, U builds the Enveloped Data[5], exploiting PKCS#7 technologies:

$$EnvelopedData := \{PKCS\#10\}_{KEY}, \{KEY\}_{pk_{CA}}$$

[4] This pin is used for certificate revocation (currently implemented as a manual process: User phones a CA Operator asking for revocation of its certificate, the operator replies asking for the challenge password, and if it coincides with the one contained in PKCS#10 request, the certificate will be revoked). The pin can also be used to authenticate the identity of User U, as explained in subsection 3.2.

[5] For the sake of readability the structures of Enveloped Data, Signed Data and Get Cert Initial message here are simplified (without however affecting the results of our analysis).

where KEY is a randomly generated symmetric key. The construction of Enveloped Data provides the encryption of KEY with the public key of the CA, pk_{CA}, so that only CA can retrieve KEY and successively obtain the PKCS#10 as a clear-text.

To complete the enrollment request, U creates Signed Data, basically consisting of:

$$SignedData := \{EnvelopedData, \{ID, Nonce\}_{pk_U^{-1}}\}$$

- The aim of the Transaction Identifier ID is to uniquely identify the transaction. ID is the MD5 fingerprint of the public key to be certified.
- $Nonce$ is a random number generated by U, and its aim is to prove the freshness of the response from the CA to the user request.

ID, $Nonce$ are sent as "authenticated attributes" of PKCS#7, signed with private key of U. Answers[6] of CA to U's enrollment request can be of three kinds:

1. SUCCESS response: CA successfully emits the requested certificate;
2. PENDING response: CA is configured to act in manual mode. Before the emission, it has to carry out some checks to verify enrollment request correctness;
3. FAILURE response: checks have produced a negative result and CA does not emit the certificate.

When User receives a response from CA containing a *pending* "status", it can enter into polling mode, i.e. it can periodically send to CA *Get Cert Initial* messages, pressing for the certificate emission. The structure of a *Get Cert Initial* message is substantially the following:

$$GetCertInitial := \{\{name_U, ID\}_{KEY}, \{KEY\}_{pk_{CA}}, \{Nonce\}_{pk_U^{-1}}\}$$

3.2 Modeling the enrollment procedure

For the sake of clarity, it is worth spending a few words on User Authentication. In protocols that use public key cryptography, the association between the public keys and the identities with which they are associated must be authenticated in a secure manner. SCEP provides two authentication methods: a manual one and one based on a pre-shared secret. In manual mode, once a certificate request has been sent to CA, U has to wait until its identity can be verified using any reliable out of band method. In [7] it is suggested that CA generates the MD5 fingerprint of the PKCS#10 retrieved from the User request and compares it with the one computed by the User itself. During this period, the state of the whole transaction is set to "PENDING".

[6] These answers contain the same ID and $Nonce$ present in User Certificate Request.

Otherwise, CA can choose to act in automatic mode: before any request takes place, CA should distribute a pre-shared secret to the User - the secret is assumed to be unique for each User (the way in which the distribution takes place is subject to the CA policy). When creating an enrollment request, the User will insert the secret in the PKCS#10 (later on we refer to this secret as pin_U). Upon receiving the request, CA should check the correspondence between pin_U and the subject name included in the PKCS#10.

Enrollment procedure with out-of-band User Authentication. Formally, the enrollment procedure with manual authentication of the User can be described as follows:

---

1 $U \mapsto CA : \{name_U, pk_U, pin_U, \{....\}_{pk_U^{-1}}\}_{KEY}, \{KEY\}_{pk_{CA}}, \{ID_U, Nonce_U\}_{pk_U^{-1}}$
2 $CA \mapsto U : \{ID_U, Nonce_U, \text{“}pending''\}_{pk_{CA}^{-1}}$
3 $U \mapsto CA : \{name_U, ID_U\}_{KEY}, \{KEY\}_{pk_{CA}}, \{Nonce1_U\}_{pk_U^{-1}}$
4 $CA \mapsto U : Hash\{name_U, pk_U, pin_U, \{....\}_{pk_U^{-1}}\}$
5 $U \mapsto CA : Comparison : ok/ko$
6 $CA \mapsto U : \{\{name_U, pk_U\}_{pk_{CA}^{-1}}\}_{KEY1}, \{KEY1\}_{pk_U}, \{Nonce1_U\}_{pk_{CA}^{-1}}$

Table 1. SCEP Enrollment Phase with out-of-band User Authentication.

1. U connects to CA and sends enveloped PKCS#10 and authenticated attributes. ID_U is the MD5 fingerprint of pk_U. $Nonce_U$ is inserted in the authenticated attributes to prevent replay attacks from the user point of view. In this procedure, every answer from CA to U has to contain the same nonce of the previous message from U to CA.
2. CA is configured to manually authenticate the end entity, so it sends a PKCS#7 message back to U containing only authenticated attributes: “status” of transaction set to *pending*, same transaction identifier and same nonce as in the user request.
3. Upon receiving a pending status, U enters into polling mode by periodically sending *Get Cert Initial* messages to CA, until he either receives the certificate or rejection or he simply times out. (Here, we suppose user successfully obtains its certificate after the sending of the first *Get Cert Initial*.)
4. Communications over channels 4 and 5 should be intended out of band, i.e. not on the network. (The analysis tool allows these two channels to be hidden from the intruder, in order to simulate a secure communication between CA and users). This reliable out of band communication could be by phone or by surface mail. In any case, CA must securely contact U and communicate the MD5 fingerprint of the PKCS#10 received in Message 1 (Message 4). Thereafter, the user can compare the fingerprint with the one computed from its original PKCS#10 (Message 5).

5. User gives a positive or negative answer to the CA, depending on the result of the comparison.
6. Upon receiving a positive answer from the User regarding the comparison of the MD5 fingerprints, CA emits the certificate. The issued certificate is formalized here with the name of the User and its public key signed by CA's private key, $\{name_U, pk_U\}_{pk_{ca}^{-1}}$. The choice in SCEP for the distribution of certificates is PKCS#7 Degenerated Mode, enveloped and followed by the same user nonce contained in the previous *Get Cert Initial*.

Enrollment Procedure with Automatic User Authentication. When a pre-shared secret scheme is used, the enrollment procedure is simply, (Tab. 2). User

$1\ U \mapsto CA : \{name_U, pk_U, pin_U, \{...\}_{pk_U^{-1}}\}_{KEY}, \{KEY\}_{pk_{CA}}, \{ID_U, Nonce_U\}_{pk_U^{-1}}$
$2\ CA \mapsto U : \{\{name_U, pk_U\}_{pk_{CA}^{-1}}\}_{KEY1}, \{KEY1\}_{pk_U}, \{Nonce_U\}_{pk_{CA}^{-1}}$

Table 2. SCEP Enrollment Phase with Automatic User Authentication.

Authentication is subject to the correspondence between pin_U and $name_U$.

4 Analysis

A formal analysis of security protocols turns out to be a useful mechanism to better understand motivations and choices for intrinsic structures of messages specified in Internet standards and drafts.
The security goals of SCEP are that no adversary can:

1. subvert the public key/identity binding from that intended;
2. discover the identity information in the enrollment requests and issued certificates;
3. cause the revocation of certificates with any non-negligible probability.

The first and second goals are met through the use of PKCS#7 and PKCS#10 encryption and digital signatures using authenticated public keys. The third goal is met through the use of a Challenge Password for revocation.

The revocation phase is not a concern to the scope of this paper but rather we consider the phase of enrollment. We formally analyze both the SCEP enrollment procedure with manual authentication of the User and the automatic procedure.

In our experiments, we consider a finite number of processes each having finite behavior. Note that, with regard to this scenario, SCEP guarantees the correct emission of certificates (i.e. Properties 1 and 2 hold).

However, we will focus our attention on particular checks suggested in [7], in order to understand some security mechanisms and the possible consequences of their absence.

In subsection 4.1 we consider the security mechanisms suggested in the Internet Draft to achieve Property 1, namely what mechanisms are involved and whether without them it would be possible to emit a certificate with a errand correspondence between the identity and the public key to be certified.

The possibility of issuing two (or more) certificates with same subject name - public key - key usage binding whose validity periods overlap is also considered. In subsection 4.2 we consider why the event should be avoided, what [7] suggests and what could happen without these suggestions.

4.1 Relevance of User Authentication.

One of the security goals of SCEP is that no intruder can force CA to emit erroneous certificates in which the public key / identity binding is subverted. Here, we perform a simple analysis about the security of SCEP with out-of-band User Authentication against an active enemy that may try to interfere. We make no assumption about the behavior of this enemy. In principle, he is able to listen to, intercept and fake communications between legitimate users and CAs. We are going to check whether CA is able to emit certificates in which the name of a legitimate user is tied to a public key provided by an enemy. This is an important question, since if CA emits such a certificate it could cause the so called "*responsibility* attack". Indeed, someone could sign something and make you responsible for that signature.

We checked whether the enrollment procedure described in Tab. 1 allows the emission of a wrong certificate like $\{name_U, pk_X\}_{pk_{ca}^{-1}}$. We checked the specifications that follow the Internet Draft (i.e. including the comparison of the PKCS#10 fingerprints). The output of the tool confirms the correct emission of the certificate.

We check another specification, modified in such a way that there is no MD5 fingerprint comparison. This specification could represent either the fact that CA does not contact the User to communicate the received fingerprint (no Message 4 in Tab. 1) or the fact that the User itself omits the comparison (no Message 5 in Tab. 1). In this case, the protocol results vulnerable to a "man in the middle" attack. The tool automatically unveils the attack.

The attack procedure consists of the following steps (see also Table 3):

1. U connects to CA as in a normal execution, but its request is intercepted by X.
2. X sends to CA a certificate request containing the name of User U.
3. CA's answer contains a pending status.
4. U enters into polling mode. Its *Get Cert Initial* message is intercepted by X.

1 $U \mapsto X(CA)$	: $\{name_U, pk_U, pin_U, \{...\}_{pk_U^{-1}}\}_{KEY}, \{KEY\}_{pk_{CA}}, \{ID_U, Nonce_U\}_{pk_U^{-1}}$
2 $X(U) \mapsto CA$	: $\{name_U, pk_X, pin_X, \{name_U, pk_X, pin_X\}_{pk_X^{-1}}\}_{KEY_X}, \{KEY_X\}_{pk_{CA}}$, $\{ID_X, Nonce_U\}_{pk_X^{-1}}$
3 $CA \mapsto U$	: $\{ID_X, Nonce_U, \text{“}pending''\}_{pk_{CA}^{-1}}$
4 $U \mapsto X(CA)$	: $\{name_U, ID_U\}_{KEY}, \{KEY\}_{pk_{CA}}, \{Nonce1_U\}_{pk_U^{-1}}$
5 $X(U) \mapsto CA$	: $\{name_U, ID_X\}_{KEY_X}, \{KEY_X\}_{pk_{CA}}, \{Nonce1_U\}_{pk_X^{-1}}$
6 $CA \mapsto U$	: $\{\{name_U, pk_X\}_{pk_{CA}^{-1}}\}_{KEY1}, \{KEY1\}_{pk_X}, \{Nonce1_U\}_{pk_{CA}^{-1}}$

Table 3. No MD5 fingerprint comparison: X is able to force the emission of a certificate where its public key is tied to the name of the User U.

5. X simulates the polling mode.
6. Something went wrong with the comparison of MD5 fingerprint. It is possible to issue the certificate related to X request.

Note 1. The particular structure of messages in SCEP helps U to discover that he actually receives a fake certificate. Upon receiving CA's answer containing the certificate (message 6), U will not be able to open the envelope, since the symmetric key $KEY1$ is encrypted with the wrong key, pk_X. In any case, the enemy could intercept message 6, so that the user does not receive the certificate. In these circumstances, the user presumably sends to CA a sequence of *GetCertInitial* messages, pressing for the certificate. The possible interceptions of *Get Cert Initial* messages by the intruder and the consequent time out interrupt of U's connection may lead U to realize that something wrong has happened.

4.2 How to avoid the issuance of two identical certificates

For User Authentication in automatic enrollment, (Tab.2), it is recommended the use of a pre-shared secret scheme. The pre-shared secret is represented in our formalization by pin_U. As an enrollment request is received by CA server, it appears reasonable to verify the right correspondence between the applicant identity and its related secret pin_U. Further, [7] encourages CAs to enforce *certificate-name uniqueness*: at any time, there will be only one pair of keys for a given subject name and key usage combination[7]. We prefer to name the property *weak uniqueness*: it is not possible to emit two (or more) valid certificates with same subject name, public key and key usage whose validity periods overlap. The *weak uniqueness* property is in contrast with another property: it is *never* possible to emit two (or more) certificates with same subject name, public key and key usage. We call the last property *strong uniqueness*.

[7] Uniqueness is not mandatory in the draft, but the draft itself claims that all current SCEP client implementations expect the property. Moreover, the authors of the draft made examples by assuming uniqueness holds. Thus, we perform our analysis assuming the property holds.

In SCEP specifications U is allowed to re-use the same request when it times out from polling for a pending request, or some errors in the network occur and the connection between U and CA goes down. In any case, the second request should not create a new transaction nor should the second request be rejected. So what should the request re-emission lead to? There are some possibilities expressed in the draft:

- If CA has already emitted the certificate and the re-emission of the request occurs less than halfway through the validity time of an existing certificate, then CA should realize that something went wrong in the first sending of the certificate. It can possibly re-send the same certificate to the applicant, without issuing another certificate.
- If CA has already emitted the certificate, and the re-emission of the request occurs more than halfway through the validity time of an existing certificate, this can be interpreted by CA as a renewal request. In that case, CA should have previously revoked the existing certificate[8].
- If CA has not yet emitted the certificate, the reception of the same request should be taken by CA as another $GetCertInitial$ message, instead of a request for a new enrollment.

Let us consider the hypothesis that CA emitted two certificates with the same subject name U, the same public key and key usage. They will differ in the serial number, say $sn1$ and $sn2$, and the respective validity periods will overlap. Suppose also that X was able to force CA to issue the last certificate, so X is conscious of its existence, while U is not. There are multiple reasons for preventing the re-transmission of the same data from creating a second certificate. The most significant reasons, according to us, are cited below:

- considering a large scale application scenario, the computational cost in generating and signing unused certificates can be high;
- a document digitally signed by U (i.e. by the public key corresponding to U's private key) could be valid longer than expected, because the not already expired certificate would validate the signature;
- U could maliciously extend his own certificate validity even when he is purposely denied the right to a new certificate; for example in a corporate environment, an employee might have access to a certain facility but only for a limited time.

Each public key to be certified is strictly connected to the Transaction Identifier ID (we remind that ID is the MD5 hash of the public key). To guarantee the *weak uniqueness* property we assume that CA records the pair $(name_U, ID_U)$.

[8] Before revoking the existing certificate, CA should presumably contact the user for confirmation, but this is not specified in [7].

We checked whether the automatic enrollment procedure in Tab. 2 allows the emission of two identical certificates whose validity periods overlap. We checked a specifications that expect the *weak uniqueness* property (i.e. a specifications including the record of the pair $(name_U, ID_U)$). The output of the tool confirms that it is not possible to emit two identical certificates.

We checked another specification, modified in such a way that CA omits to record the pair $(name_U, ID_U)$. An intruder X could eavesdrop on a legitimate certificate request and simply repeat it later. In this way, X can force CA to issue two identical certificates - apart from the serial number. The replay attack reported by the tool is shown in Tab. 4. Basically, the attack procedure consists of the following steps:

1. U connects to CA as usual. Its request is eavesdropped by X.
2. X connects to CA and repeats U certificate request.
3. CA emits a first valid certificate with serial number sn_1.
4. CA emits a second valid certificate with a different serial number, because it omits checks on crucial fields in the received requests. X is able to intercept the message.

1 $U \mapsto CA$	: $\{name_U, pk_U, pin_U, \{...\}_{pk_U^{-1}}\}_{KEY}, \{KEY\}_{pk_{CA}}, \{ID_U, Nonce_U\}_{pk_U^{-1}}$
2 $X(U) \mapsto CA$	: $\{name_U, pk_U, pin_U, \{...\}_{pk_U^{-1}}\}_{KEY}, \{KEY\}_{pk_{CA}}, \{ID_U, Nonce_U\}_{pk_U^{-1}}$
3 $CA \mapsto U$	: $\{\{name_U, pk_U, sn_1\}_{pk_{CA}^{-1}}\}_{KEY1}, \{KEY1\}_{pk_U}, \{Nonce_U\}_{pk_{CA}^{-1}}$
4 $CA \mapsto X(U)$	: $\{\{name_U, pk_U, sn_2\}_{pk_{CA}^{-1}}\}_{KEY2}, \{KEY2\}_{pk_U}, \{Nonce_U\}_{pk_{CA}^{-1}}$

Table 4. Replay Attack: if CA does not record the Transaction Identifier, the intruder is able to force the emission of a double certificate.

Note 2. The user itself may unconsciously contribute towards the emission of identical certificates. Suppose the user times out or the connection with the CA Server goes down for some reason. Under these circumstances CA could emit a first certificate but U may not receive anything because of the connection crash or time out. Consequently, U is allowed to re-issue the same request and the absence of checks by CA may lead to the emission of a double certificate.

Note 3. In the discussion above, we investigated the existence of certificates whose validity periods overlap. When automatic enrollment is used, *weak uniqueness* property is not enough to protect against replay attacks on expired certificates requests. An intruder X could eavesdrop on a legitimate certificate request. The automatic procedure will lead to the emission of a legitimate certificate, say $\{name_U, pk_U\}_{pk_{ca}^{-1}}$. The intruder may send the replay of the request once the legitimate certificate has been expired. This can cause a new certificate to be issued with the same subject name - public key binding. The existence of the last certificate may cause a document previously signed by U to be valid longer than expected. To avoid this vulnerability CAs should guarantee the *strong uniqueness* property.

5 Concluding remarks

In this paper we gave a formalization of the SCEP protocol enrollment phase and performed the analysis of part of the protocol's security properties by means of a software tool. With regard to a limited scenario (finite number of processes with a finite behavior), we did not find any attack, meaning the analyzed properties hold. When automatic enrollment is used, we found a vulnerability concerning a replay attack on expired certificates requests. To deep understand certain security mechanisms in SCEP specifications, we purposely omitted particular checks suggested in the draft in some of our experiments. As a consequence, an attack regarding the emission of certificates with the public key/identity binding subverted and an attack concerning duplicate valid certificates were automatically detected. Since the technique is general and can be applied to several cryptographic protocols, it turns out to be a valid support for the analysis of RFCs, drafts and commercial products.

Acknowledgements. We would like to thank the anonymous referees for their help in the presentation of the paper. We also thank Claud Anticoli for his valued collaboration as proof reader.

References

1. M. Abadi and A.D. Gordon. Reasoning about Cryptographic Protocols in the Spi Calculus. In *Proc. of CONCUR'97*, volume 1243 LNCS, pages 59–73. Springer, 1997.
2. T. Cichocki and J. Gorski. Formal Support for Fault Modeling and Analysis. In *Proc. of SAFECOMP'01*, volume 2187 LNCS, pages 190–199. Springer, 2001.
3. R. Focardi, R. Gorrieri, and F. Martinelli. Non Interference for the Analysis of Cryptographic Protocols. In *Proc. of ICALP'00*, volume 1853 LNCS, pages 354–372. Springer, 2000.
4. A. Giani, F. Martinelli, M. Petrocchi, and A. Vaccarelli. A Case Study with PaMoChSA: a Tool for the Automatic Analysis of Cryptographic Protocols. In *Proc. of SCI-ISAS'01*, volume 5, Orlando, July 2001.
5. R. Housley, W. Ford, W. Polk, and D. Solo. RFC 2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile, IETF, 1999. http://www.ietf.org/rfc/rfc2459.txt.
6. http://pmartinelli1.iat.cnr.it:8080/pamochsa.htm.
7. X. Liu, C. Madson, D. McGrew, and A. Nourse. Internet Draft: draft-nourse-scep-06, Cisco Systems, 2002. http://www.vpnc.org/draft-nourse-scep.
8. G. Lowe and A. W. Roscoe. Using CSP to Detect Errors in the TMN protocol. *Software Engineering*, 23(10):659–669, 1997.
9. D. Marchignoli and F. Martinelli. Automatic Verification of Cryptographic Protocols through Compositional Analysis Techniques. In *Proc. of TACAS'99*, volume 1579 LNCS, 1999.
10. F. Martinelli. Analysis of Security Protocols as Open Systems. *Theoretical Computer Science (to appear)*. A preliminary version in ICTCS, World Scientific, pages 304-315, 1998.
11. C. Meadows. Formal Verification of Cryptographic Protocols: a Survey. In *ASIACRYPT'94: Advances in Cryptology*, volume 917 LNCS, pages 135–150. Springer, 1995.
12. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
13. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Comm. of the ACM*, 21(2):120–126, 1978.
14. F.B. Schneider. *Applied Cryptography*. J. Wiley & sons, Inc, 1996.
15. V. Shmatikov and U. Stern. Efficient Finite State Analysis for Large Security Protocols. In *Proc. of CSFW'98*, pages 105–116. IEEE Computer Society Press, 1998.