Consiglio Nazionale delle Ricerche

Infrastrutture a chiave pubblica e protocolli di sicurezza

M. Petrocchi

IAT B4-04/2001

Technical Report

Aprile 2001

Istituto per le Applicazioni Telematiche

Infrastrutture a chiave pubblica e protocolli di sicurezza

Marinella Petrocchi

Istituto per le Applicazioni Telematiche - CNR
Via G.Moruzzi 1, 56124 PISA

17 Aprile 2001

Parole chiave: Firma digitale, PKI, Protocollo di Sicurezza, Verifica Assistita con Calcolatore.

Indice

1 Introduzione

Contestualmente all'utilizzo della rete per la trasmissione di dati e documenti, negli ultimi anni è cresciuta notevolmente l'esigenza di garantire a questi determinate proprietà di sicurezza. È infatti possibile che i messaggi, viaggiando da un nodo all'altro della rete senza alcuna forma di protezione, siano intercettati e subiscano modifiche prima di giungere a destinazione. I protocolli di comunicazione sicura devono tendere essenzialmente a soddisfare i seguenti attributi:

- confidenzialità: assicurazione che l'informazione non possa essere letta mentre transita in rete dal nodo mittente al nodo destinatario;
- integrità: assicurazione che l'informazione giunga a destinazione nella forma esatta in cui è stata inviata;
- non ripudio: prevenzione dalla possibilità che il mittente ripudi l'informazione inviata;
- autenticità: certezza sull'identità del mittente dell'informazione.

Principalmente gli strumenti utilizzati per garantire il soddisfacimento di queste proprietà sono basati su algoritmi crittografici. Tuttavia, pur assumendo affidabile il sistema crittografico di base, un gran numero di protocolli apparsi in letteratura e di standard internazionali per comunicazioni remote sono stati trovati difettosi, anche dopo anni dalla pubblicazione [1] e dopo essere stati ritenuti precedentemente corretti [2]. Data la oggettiva difficoltà nella progettazione dei protocolli, sorge la necessità di avere a disposizione strumenti che analizzino e verifichino, almeno ad un livello concettuale, la loro correttezza con un elevato grado di automazione.

Questa presentazione è strutturata nel seguente modo: nel primo paragrafo si espongono nozioni base di crittografia e si definiscono i concetti di firma digitale, certificato digitale ed infrastruttura a chiave pubblica (PKI); nel secondo paragrafo, si descrivono alcuni aspetti pratici legati alla gestione di una PKI. Il terzo paragrafo è dedicato al problema della corretta progettazione dei protocolli di sicurezza: si descrivono PaMoChSA , un tool per l'analisi dei protocolli, e la sua interfaccia grafica; si analizza inoltre, per mezzo di PaMoChSA , una particolare procedura di rilascio di un certificato digitale. Nel quarto paragrafo si presentano gli studi preliminari per la progettazione di un compilatore per l'analisi dei protocolli.

2 La crittografia, la firma digitale e le PKI

2.1 Gli algoritmi crittografici

La crittografia è la disciplina che individua algoritmi capaci di trasformare in modo reversibile un messaggio in modo tale che il suo significato sia comprensibile ad un gruppo ristretto di persone, se non ad una soltanto. Tali algoritmi non sono di recente scoperta (risalgono infatti agli anni 70), ma il loro utilizzo si è imposto sul panorama informatico mondiale per l'inarrestabile sviluppo degli scambi telematici in campo di Pubblica Amministrazione, Commercio Elettronico, Posta Elettronica, solo per citare alcune applicazioni tipiche in cui si richiedono transazioni elettroniche sicure.
Esistono due grandi categorie di algoritmi crittografici: algoritmi a chiave privata (o simmetrici) ed algoritmi a chiave pubblica (o asimmetrici) (vedi [15, 16]); in entrambi i casi il sistema consiste di una funzione di cifratura *E*, una funzione di decifratura *D*, ed una coppia di chiavi: mentre negli algoritmi a chiave simmetrica le due chiavi sono in realtà una sola, utilizzata sia per cifrare che per decifrare e che deve essere mantenuta in possesso segreto ed esclusivo del mittente e del destinatario, negli algoritmi asimmetrici ogni coppia consiste di una chiave pubblica k e di una privata k^{-1}, tali che $k^{-1^{-1}} = k$. La chiave privata è nota ad una sola persona, mentre ognuno può conoscere ed utilizzare la corrispondente chiave pubblica. Dati una informazione m ed una chiave k, *E* rende la cifratura dell'informazione con la chiave, in genere rappresentata in letteratura con la notazione $\{m\}_k$, tale che

$$E(m,k) = \{m\}_k$$

e che

$$D\,(E(m,k),k^{-1}) = m$$

Ogni coppia di chiavi viene generata in modo tale da soddisfare determinate proprietà:

- nell'ambito di una coppia, quello che una chiave cifra, l'altra decifra;
- con la sola conoscenza del messaggio cifrato $\{m\}_k$ non è possibile ottenere alcuna valida informazione su m o su k;
- con la sola conoscenza della chiave pubblica non è possibile ottenere alcuna informazione valida sulla corrispondente chiave privata.

La crittografia simmetrica richiede, per comunicazioni sicure entro un gruppo di n persone, l'utilizzo di circa n^2 chiavi, una per ciascuna coppia mittente/destinatario. Al contrario, l'utilizzo di algoritmi a chiave pubblica prevede, per lo stesso numero

di persone, la gestione di n chiavi pubbliche. Ovviamente, ciascuno avrà cura di conservare con attenzione la propria chiave privata. Parallelamente al problema della gestione delle chiavi, l'esistenza della chiave pubblica elimina anche il problema dello scambio della chiave segreta tra due utenti, grosso limite degli algoritmi simmetrici. Per contro, l'utilizzo di crittografia asimmetrica richiede tempi elevati per cifrare blocchi di informazione di grandi dimensioni (MByte ed oltre), e si preferisce in questo caso utilizzare algoritmi simmetrici. A tale proposito, uno dei molteplici utilizzi della cifratura con chiave asimmetrica consiste nello scambio sicuro della chiave simmetrica tra mittente e destinatario.

L'utilizzo di algoritmi asimmetrici permette di soddisfare le seguenti proprietà:

- confidenzialità ed integrità: il mittente cifra il messaggio con la chiave pubblica del destinatario, nota a tutti, e solo il destinatario, possessore della corrispondente chiave privata, è in grado di decifrarlo.

- autenticità, non ripudio, integrità: il mittente cifra il messaggio con la sua chiave privata; in questo modo, ognuno ha la possibilità di decifrare il messaggio, utilizzando la chiave pubblica del mittente stesso, ma questo non ha importanza, visto che non è richiesta la proprietà di confidenzialità. In particolare però, il destinatario ha garanzia, decifrando con una certa chiave pubblica, del fatto che il messaggio è stato originato dal possessore della chiave privata corrispondente; inoltre, il mittente non può arbitrariamente negare la paternità di quel particolare messaggio.

2.2 La firma digitale

La firma digitale viene apposta utilizzando algoritmi asimmetrici (vedi [16]). Essa serve a garantire l'autenticità di un messaggio, non necessariamente tutelandone la segretezza. Poiché i documenti da firmare possono avere lunghezza arbitraria, e data la lentezza delle operazioni coinvolte, si evita in genere di applicare l'algoritmo di firma a tutto il documento: dal testo originale si ricava, tramite particolari funzioni dette funzioni *hash*, un riassunto, detto *digest*. La firma digitale è il risultato della cifratura del *digest* di un documento con la chiave privata.

I passi da compiere per l'apposizione della firma sono i seguenti:

- calcolare il digest del messaggio;

- cifrare il digest con la chiave privata del mittente;

- aggiungere il messaggio originale alla firma.

Il ricevente verifica la correttezza della firma con i seguenti passi:

- separa il messaggio originale dalla firma;
- applica la chiave pubblica del mittente alla firma, recuperando il digest;
- calcola il digest del messaggio originale;
- confronta i due digest ottenuti: se sono uguali, il messaggio può essere accettato, altrimenti è rifiutato.

Se i due digest sono uguali, significa che il messaggio non ha subito modifiche durante la trasmissione (è garantita l'integrità); inoltre, l'origine del messaggio è certa (autenticità e non ripudio).

2.3 Le infrastrutture a chiave pubblica

Trattando con algoritmi asimmetrici, appare chiaro come uno dei punti più critici dell'intero sistema sia la corrispondenza tra una chiave pubblica ed il suo possessore. Questa corrispondenza viene dichiarata nel cosiddetto "certificato digitale", rilasciato da un ente apposito, noto come "Ente Certificatore" in Italia e come "Trusted Third Authority" in Europa; nel seguito ci riferiremo all'ente denominandolo "Autorita' di Certificazione" (CA). Il certificato è rilasciato dalla CA sotto previa autorizzazione di una Autorità di Registrazione (LRA), il cui compito è accertare l'identità della persona che fa richiesta della certificazione di una chiave pubblica.
Una PKI (Public Key Infrastructure - Infrastruttura a Chiave Pubblica) è una infrastruttura dedita al rilascio dei certificati digitali secondo lo standard X.509. Una PKI fornisce la minima sicurezza per lo scambio di dati attraverso il canale informatico. Tra gli elementi di una PKI vi sono:

- la Policy di sicurezza: stabilisce i principi su cui si basa l'organizzazione e fornisce informazioni sui livelli di sicurezza che si intende raggiungere;
- il CPS (Certificate Practice Statement): un documento contenente le operazioni procedurali che permettono di raggiungere la sicurezza referenziata nella policy. Tali documenti contengono le procedure applicate per l'emissione di certificati e per la revoca, spiegano come sono emesse le chiavi, come sono registrati i certificati, dove sono archiviati e come sono resi noti all'utente;
- un sistema di CA;
- un sistema di LRA.

Un cammino di certificazione è una sequenza di nodi tra l'entità richiedente il certificato e una CA radice. Le CA di un PKI possono essere organizzate in una struttura gerarchica con una sola CA radice oppure le CA possono stabilire di fidarsi l'una dell'altra (ovvero ognuna stabilisce di riconoscere validi i certificati emessi dalle altre).

Oltre a farsi garante dell'identità di un richiedente ed a rilasciare e rendere pubblico un certificato, una PKI ha svariati compiti, tra i quali dichiarare la propria politica di sicurezza, procedere alla revoca o alla sospensione dei certificati, assicurare la corretta manutenzione del sistema di certificazione. Anche un' Autorità di Certificazione ha la sua coppia di chiavi, pubblica e privata. Attraverso la chiave privata, la CA appone la sua firma su ogni certificato digitale rilasciato. Quando il possessore presenta telematicamente il certificato ad un altro utente, con l'intento di identificarsi per dare inizio ad una serie di operazioni, quest'ultimo è avvertito del fatto che il certificato è stato emesso da una determinata CA; a quel punto può decidere se fidarsi di quella particolare CA e conseguentemente della buona fede dell'interlocutore.

Il formato X.509 prevede i seguenti campi obbligatori:

- versione: contiene il numero di versione dello standard X.509 con cui è generato il certificato;
- numero seriale: è un numero intero assegnato dalla CA che emette il certificato;
- identificatore dell'algoritmo di firma: indica con quale algoritmo la CA firma il certificato;
- identificatore dell'emettitore: identifica la CA che rilascia il certificato, attraverso il paese e l'organizzazione di appartenenza ed un indirizzo e-mail;
- periodo di validità del certificato;
- identificatore del soggetto certificato: identifica il soggetto di cui viene certificata la chiave pubblica, attraverso paese, organizzazione, nome, cognome ed indirizzo e-mail;
- identificatore della chiave pubblica del soggetto: contiene la chiave pubblica certificata;

È prevista la presenza di possibili campi opzionali, contenenti informazioni sull' organizzazione gerarchica dei certificati e sulla politica di distribuzione delle

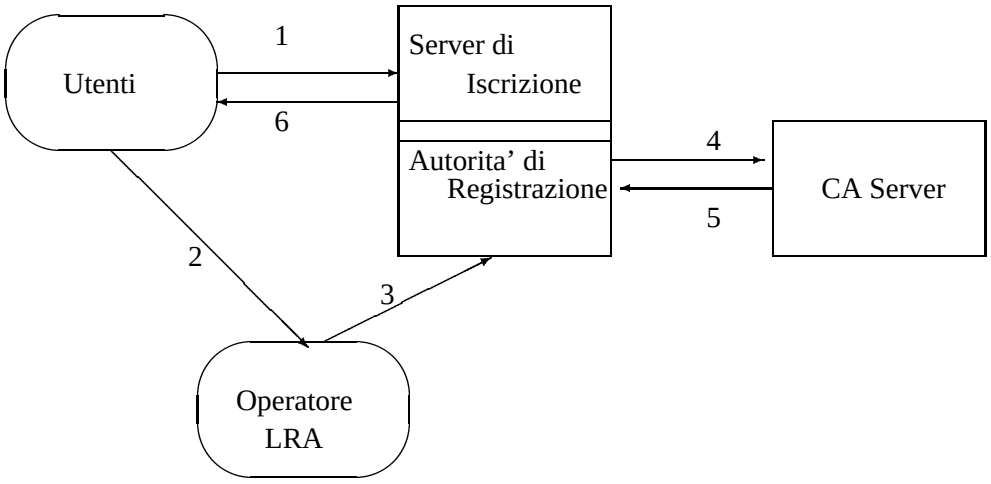

Figure 1: Rappresentazione grafica della PKI-RA.

liste di revoca dei certificati. Altre estensioni standard possibili riguardano la politica che regola l'emissione dei certificati e l'utilizzo per il quale il certificato viene emesso, ovvero se la chiave pubblica certificata può essere utilizzata per firma digitale, cifratura di dati, firma di altri certificati, non ripudio, ecc.

3 Gestione di una Public Key Infrastructure

Lo studio teorico delle infrastrutture a chiave pubblica ha trovato applicazione in casi pratici, nell'ambito di sperimentazioni avviate dall'Istituto per le Applicazioni Telematiche. L'istituto svolge per la comunità Internet italiana il ruolo di Registration Authority (RA) per la registrazione dei nomi a dominio sotto il TLD .it; nell'ambito di un progetto di posta elettronica sicura è stata istituita la PKI-RA (Public Key Infrastructure - Registration Authority). Il principale scopo di questa infrastruttura è quello di consentire agli Internet Service Provider (ISP) di comunicare via e-mail con la RA in modo sicuro (vedi [17]). Nel seguito, si descrivono le modalità di richiesta e rilascio di un certificato, descrivendo l'organizzazione della PKI realizzata per la sperimentazione. Attualmente, le richieste dei certificati devono essere fatte con il Browser Netscape, v. 4.72.

Le entità coinvolte sono le seguenti:

- un utente che richiede un certificato;

- un server (detto di iscrizione) a cui l'utente si collega per riempire un documento elettronico con i suoi dati personali e per stampare un modulo da riempire a mano;
- un operatore locale LRA con il compito di verificare che i dati inseriti nella form elettronica corrispondano ai dati dei documenti cartacei consegnati personalmente dall'utente; l'operatore interagisce sia con l'utente che col server RA;
- un server che funge da Autorità di Registrazione (RA), a cui si connette l'operatore LRA per approvare le richieste di certificato;
- l' Autorità di Certificazione vera e propria (CA), che è una macchina non in linea, il cui compito e' il rilascio dei certificati. Le interazioni tra CA ed RA server sono svolte da un gestore di CA, con l'ausilio di mezzi removibili. Il gestore di CA conosce la password che protegge la chiave privata della CA. La chiave privata e' il cuore di tutta la struttura; la compromissione di quest'ultima infatti consente ad un intruso di generare certificati fasulli, in cui la corrispondenza tra una chiave pubblica ed una persona e' assolutamente arbritaria.

3.1 Procedura di rilascio di un certificato digitale

L'utente si collega al server di iscrizione, (passo 1 in figura 1), provvede a stampare un modulo da compilare e presentare, assieme ad un documento d'identità, all'operatore LRA (passo 2), compila infine il modulo elettronico di richiesta del certificato e lo invia; tra i dati da inserire, vi è anche un pin (sequenza casuale a discrezione dell'utente, di almeno 10 caratteri), e viene inviato anche il cosiddetto SPKAC di Netscape, che consiste nella chiave pubblica dell'utente (generata assieme alla chiave privata prima dell'invio della richiesta) unita ad un'altra sequenza casuale, chiamata $nonce$, entrambe firmate con la chiave privata dell'utente. La creazione e l'invio di SPKAC sono attuati automaticamente dal browser, così come l'invio della chiave pubblica da certificare. Lo SPKAC firmato agisce da "Prova di Possesso" (POP); attesta che l'utente, il quale sta chiedendo di legare il suo nome ad una certa chiave pubblica, è realmente in possesso della corrispondente chiave privata. Come anticipato, il richiedente si presenta (passo 2) allo sportello di registrazione, e fornisce all'operatore LRA un documento valido ed il modulo cartaceo. L'operatore si connette al server sicuro RA, (passo 3), e seleziona, nella lista delle richieste pendenti, la form elettronica inviata precedentemente dal richiedente; è sua cura a questo punto verificare la correttezza dei dati inseriti nella form, confrontandoli con i dati su carta. Se i dati corrispondono,

l'operatore firma con la sua chiave privata la richiesta, approvandola implicitamente; automaticamente, dalla lista delle richieste pendenti la richiesta passa alle approvate. La workstation CA è situata fisicamente in luogo sicuro, accessibile esclusivamente ad un operatore fidato, il quale tramite floppy disk esporta dal Server RA le richieste approvate e le importa nel Server CA (passo 4). Dopo avere verificato che la richiesta importata contenga la firma dell'operatore LRA, l'operatore controlla la correttezza dello SPKAC, applicandovi l'algoritmo di verifica della firma, per mezzo della chiave pubblica del richiedente, per ottenere la chiave pubblica stessa e la nonce; se i controlli hanno buon esito, si emette il certificato. Per autorizzare l'emissione, l'operatore deve essere a conoscenza della password che protegge la chiave privata della CA. Il certificato emesso viene esportato, sempre tramite floppy disk, al Server RA, (passo 5) e infine viene mandata una e-mail al richiedente, con le istruzioni per l'ottenimento del certificato dal server di iscrizione (passo 6).

Per connettersi al server d'iscrizione non sono richiesti particolari privilegi, ogni generico utente può fare richiesta di un certificato. Per connettersi invece ai server RA e CA viene richiesta dai server stessi la presentazione di un particolare certificato, detto di staff, rilasciato esclusivamente agli operatori, e l'inserimento di una password, che protegge la chiave privata relativa al certificato presentato. Il possesso di un certificato di staff e la conoscenza di detta password consentono quindi agli operatori LRA di approvare le richieste, agli operatori CA di emettere certificati (fermo restando la necessità per questi ultimi di conoscere la password che protegge la chiave privata della CA). Ovviamente, per raggiungere fisicamente la CA, l'operatore deve essere in possesso delle chiavi del locale sicuro dove si trova la workstation. È necessario, inoltre, che l'operatore di CA esegua, prima di qualsiasi operazione, un backup dello stato attuale, per contenere rischi di perdita di informazione dovuti ad eventuali errori umani.

3.2 Gestione della lista di revoca dei certificati

Una corretta manutenzione della PKI-RA, come di una qualsiasi altra infrastruttura a chiave pubblica, prevede la gestione e l'aggiornamento delle CRLs (Certificate Revocation Lists), le liste di revoca dei certificati. Normalmente, un certificato digitale non ha un tempo di vita illimitato e scade dopo un determinato periodo (l'informazione sulla data di scadenza è presente in un apposito campo del certificato stesso). Ciò non toglie che, in determinate situazioni, si debba procedere alla revoca di un certificato prima del suo decorso naturale. Le ragioni sono molteplici, ad esempio la compromissione della chiave privata del possessore del certificato (per impossibilità di accedervi o per furto). Nel caso si debba procedere alla re-

voca, le operazioni da compiere sono le seguenti:

1. l'operatore di CA accede all'omonimo server, autentificandosi presentando il suo certificato di staff;

2. accede alla cartella $ValidCertificates$, seleziona il certificato da revocare e procede alla revoca;

3. procede all'emissione di una nuova CRL (aggiornata con l'inserimento del certificato revocato) e la esporta per mezzo di floppy disk nel Server RA, in modo che gli utenti possano scaricarla dal server di iscrizione.

Ovviamente, anche per compiere le operazioni di revoca di un certificato ed emissione di una nuova CRL occorre l'inserimento, su richiesta del CA Server, dell'opportuna password per la chiave segreta della CA. Il rinnovo della CRL avviene all'interno del CA Server; infatti, la CRL emessa viene convalidata dalla CA stessa firmando la lista con la sua chiave privata, che di fatto è stata generata e risiede fisicamente dentro la workstation (da qui il bisogno di accedere alla chiave privata inserendo la password).

4 I protocolli di sicurezza

Il corretto design dei protocolli di sicurezza presenta risvolti alquanto delicati: molti protocolli, creduti inattaccabili per lunghi periodi, sono in seguito stati trovati difettosi, pur assumendo affidabile e robusto il sistema crittografico di base. Gli attacchi si rivelano frequentemente molto sottili: si supponga, a titolo di esempio, che un utente A voglia comprare in rete un coniglio di peluche per il nipote. Per fare questo, comunica al venditore V cosa vuole comprare ed il numero della sua carta di credito, criptato con la chiave pubblica di V, in modo che solo questi possa decifrarlo, con la corrispondente chiave privata. L'intero messaggio è inoltre firmato con la chiave privata di A. Nella notazione usata in letteratura per descrivere un protocollo, l'invio della richiesta di compera è :

$$A \mapsto V \quad : \quad \{1\_peluche, \{numero\_carta\}_{pk_V}\}_{pk_A^{-1}}$$

Nelle reti di comunicazioni non è possibile stabilire l'origine di un messaggio: essendo il messaggio interamente firmato con la chiave privata di A, il venditore può ritenere che il messaggio sia stato generato proprio da A. Durante il percorso compiuto dal messaggio tra i nodi della rete, un intruso X può di fatto intercettarlo. Il numero di carta di credito rimane al sicuro, in quanto X non conosce il modo

per decifrare quella parte, ma può rispedire, nella sua interezza, l'intero messaggio più e più volte a V:

$$X \mapsto V \quad : \quad \{1\_peluche, \{numero\_carta\}_{pk_V}\}_{pk_A^{-1}}$$

La firma è ancora valida e V, formalmente, non ha motivo per ritenere ingiustificati i successivi ordini. Ovviamente, la conseguenza di questa azione non consona da parte di X non ha conseguenze drammatiche, ma comporta comunque un disservizio: A può trovarsi nelle circostanze di ricevere una quantità di merce notevolmente diversa da quella desiderata. Una specifica del genere quindi, pur garantendo la segretezza di informazioni preziose, quale il numero di carta di credito, può essere sfuttata da una terza parte per causare inconvenienti. Una soluzione possibile è quella di cifrare assieme al numero di carta anche la merce desiderata, assieme a data e giorno di spedizione:

$$A \mapsto V \quad : \quad \{\{1\_peluche, numero\_carta, marca\_temporale\}_{pk_V}\}_{pk_A^{-1}}$$

Per l'analisi, l'implementazione e l'utilizzo di un protocollo di sicurezza occorre quindi considerare, modellandola opportunamente, la presenza di un possibile nemico, capace di interferire con la normale esecuzione del protocollo.

4.1 Il tool PaMoChSA

Un modello diffuso in letteratura per la caratterizzazione dell'intruso è quello proposto da Dolev e Yao ([8]), dove si suppone che le comunicazioni viaggianti in rete siano sotto il completo controllo di un nemico, con facoltà di intercettare, falsificare, origliare ed aumentare la propria conoscenza durante lo svolgimento del protocollo.

In alcuni lavori di ricerca ([3, 4, 5, 6]) le proprietà di sicurezza di un sistema sono state specificate in termini del comportamento del sistema stesso, per ogni possibile ambiente di esecuzione. In altre parole, i protocolli di sicurezza sono stati trattati come *sistemi aperti*, ovvero sistemi che hanno qualche componente il cui comportamento non è specificato. Appare conveniente non fissare in anticipo il modo di operare di un intruso: assunzioni ingiustificate possono condurre a risultati di verifica errati. In formule:

$$\text{Per ogni possibile intruso} X \text{ si ha } S|X \models \phi \tag{1}$$

dove S rappresenta il sistema in esame, X l'intruso, $|$ è l'operatore di composizione parallela, ϕ è una formula logica che descrive una proprietà di sicurezza, e $\models$

è la relazione di verità. La formula si legge in questo modo: "la proprietà ϕ è soddisfatta per qualunque X interagente con S " La quantificazione universale in (1) rappresenta un limite soprattutto per analisi strumentali automatizzate. In [6] è stato presentato un nuovo approccio: per testare che $S|X \models \phi$ si controlla che X soddisfi una formula ϕ_S dipendente dal comportamento di S: X cerca di scoprire una qualche informazione; la formula ϕ_S è utilizzata per stabilire o meno le possibilità per X di ottenere l'informazione.

Il tool PaMoChSA , Partial Model Checking based Security Analyzer, implementa le precedenti idee. PaMoChSA è scritto nel linguaggio funzionale *ocaml* ,che è possibile ottenere in rete all'indirizzo [12], ed è implementato su una macchina Pentium III, con sistema operativo Linux Red Hat 6.2.

Lo scopo del tool è analizzare la possibilità che la conoscenza dell'intruso X soddisfi, durante l'esecuzione di un protocollo dato, una formula p che può codificare un segreto noto inizialmente solo ai partecipanti al protocollo.

Il tool necessita di tre ingressi:

1. la descrizione del protocollo;
2. la conoscenza iniziale dell'intruso;
3. la formula p sulla conoscenza dell'intruso, che esprime la proprietà di sicurezza che deve essere testata; tale formula può assumere la forma base di un singolo messaggio m o può essere logicamente composta attraverso gli operatori logici *and, or, not.*

In uscita il tool rende la traccia di un possibile attacco o un messaggio che segnala la correttezza del protocollo (fig. 2).

4.2 L'interfaccia grafica di PaMoChSA

Sin dalla sua prima implementazione, PaMoChSA è stato in grado di analizzare svariati protocolli di comunicazione, trovandovi gli stessi errori rilevati dai tool presenti sul mercato, come FDR ([9, 1]). Uno dei suoi principali limiti era però quello di funzionare a "linea di comando" dalla shell tipica di Linux, entrando nell'ambiente di lavoro *ocaml* ; una tale struttura ne limitava le applicazioni possibili e la funzionalità, negandone l'utilizzo ad operatori con limitate (o nulle) conoscenze del linguaggio *ocaml* . È nata quindi l'esigenza di implementare una interfaccia grafica "user friendly" a supporto di PaMoChSA .

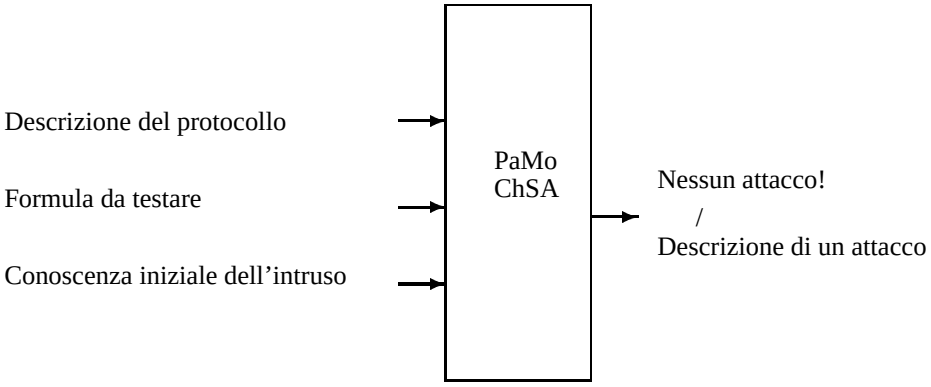

Figure 2: Ingressi ed uscita di PaMoChSA

Lo sviluppo dell'interfaccia (anch'essa scritta in *ocaml*) ha richiesto inizialmente una scelta del pacchetto grafico da utilizzare e lo studio delle funzioni del pacchetto stesso. Testando gli strumenti di interfaccia grafica suggeriti all'indirizzo [10], la scelta è ricaduta sul pacchetto *mlgtk* (vedi [11]), che integra il kit grafico GTK all'ambiente di lavoro *ocaml*. L'interfaccia (mostrata in figura 3) è modulare, ovvero non è necessario modificare il suo codice a seguito di eventuali modifiche al codice di PaMoChSA .

L'interfaccia è dotata di classici menù a tendina. Il menù "File" è composto dai seguenti comandi:

$File$
$Load\_experiment$
$Load\_spec$
$Load\_intruder\_knowledge$
$Load\_formula$
$Load\_hide\_channels$
$Quit$

In principio, l'utente carica la terna di ingressi necessari a PaMoChSA . Per fare questo, si può agire sequenzialmente attraverso il secondo, il terzo ed il quarto comando; alla selezione di ciascuno di questi appare il pannello tipico di esplorazione risorse e si possono selezionare separatamente un file di descrizione del protocollo (*Load_spec*), un file di conoscenza iniziale (*Load_intruder_knowledge*) ed un file contenente la formula da testare (*Load_formula*); la descrizione della terna così selezionata appare negli spazi "Current Spec Name", "Intruder Knowledge" e "Formula" sottostanti ai menu. Una caratteristica di PaMoChSA è quella

Figure 3: Interfaccia grafica di PaMoChSA

di negare all'intruso la possibilità di partecipare a determinati passi del protocollo: poiché ogni passo di un protocollo è rappresentato dallo scambio di un determinato messaggio, da un mittente ad un destinatario, su un particolare canale di comunicazione, per ottenere la caratteristica sopra citata si è scelto di rendere a priori inaccessibili alcuni canali di comunicazione all'intruso, in modo tale che su quelli non possa attivare le sue facoltà di ascolto ed intercettazione. I canali nascosti relativi a ciascuna specifica si caricano tramite il comando *Load_hide_channels.*

Un'alternativa per la selezione degli input è quella di caricare contemporaneamente la desiderata combinazione: col comando *Load_experiment* si può scegliere tra i files cosiddetti *di esperimento*; un file di esperimento contiene la formula, la conoscenza iniziale, l'elenco dei canali nascosti e la specifica, ciascuno racchiuso

tra speciali identificatori:

$$
\begin{array}{l}
< FORM > \\
\quad Formula \\
< END\_FORM > \\
< KNOWLEDGE > \\
\quad Conoscenza \\
< END\_KNOWLEDGE > \\
< HIDE > \\
\quad Elenco \quad canali \quad nascosti \\
< END\_HIDE > \\
< SPEC > \\
\quad Specifica \\
< END\_SPEC >
\end{array}
$$

Dopo il caricamento dei dati iniziali, si lancia l'elaborazione (bottone "Run") e si ha la possibilità di monitorare il tempo trascorso dall'inizio dell'analisi e lo stato di avanzamento dell'indagine (bottone "Update").
Alla fine della computazione viene visualizzato il risultato, che consiste nella descrizione di un attacco riuscito, oppure in un messaggio che avverte della correttezza del protocollo.
Nel caso in cui si dia inizio all'elaborazione non avendo immesso dati d'ingresso sintatticamente corretti, o qualora questi siano mancanti, assolutamente o parzialmente, tali errori sono segnalati con opportuni messaggi.
Dopo il caricamento degli ingressi si possono settare specifiche proprietà caratterizzanti l'elaborazione:

- possibilità per l'intruso di accedere a tutti i canali di comunicazione descritti nella specifica (ovvero ignorare il precedente settaggio dei canali nascosti);
- possibilità per l'intruso di generare una particolare forma di nuovi messaggi casuali durante lo svolgimento del protocollo;
- possibilità di immagazzinamento in memoria dello stato d'avanzamento dell'indagine, con registrazione degli stati visitati dall'inizio della elaborazione.

Opzioni accessorie (menu Show) consentono infine di visualizzare la specifica e la formula correntemente in analisi.

Nella successiva sezione si presenta con un esempio l'utilizzo del tool e della sua interfaccia grafica.

4.3 Procedura di rilascio certificati: analisi tramite PaMoChSA

Con l'aiuto di PaMoChSA è stata analizzata la procedura di richiesta e rilascio di certificati digitali da parte di una Certification Authority. In particolare, è stato analizzata una CA implementata tramite il software *open source* OpenCA ([18]), in pratica una interfaccia grafica per i servizi di CA offerti da OpenSSL ([19]) . CA basate su questo software sono realmente distribuite.

Con riferimento al paragrafo 2, la descrizione formale del protocollo per il rilascio dei certificati è la seguente:

$$
\begin{array}{lllll}
1 & U \mapsto ES & : & \{name_U, pk_U, pin_U, \{pk_U, n_U\}_{pk_U^{-1}}\}_{pk_{c1}} \\
2 & U \mapsto LRA & : & \{\{name_U\}_{pk_{Gov}^{-1}}, pin_U\}_{pk_{c2}} \\
3.1 & LRA \mapsto RA & : & \{name_U, pin_U\} \\
3.2 & RA \mapsto LRA & : & \{\{name_U, pk_U, pin_U, \{pk_U, n_U\}_{pk_U^{-1}}\} \\
3.3 & LRA \mapsto RA & : & \{\{name_U, pk_U, pin_U, \{pk_U, n_U\}_{pk_U^{-1}}\}_{pk_{LRA}^{-1}}\} \\
4 & RA \mapsto CA & : & \{name_U, pk_U, pin_U, \{pk_U, n_U\}_{pk_U^{-1}}\} \\
5 & CA \mapsto RA & : & \{name_U, pk_U\}_{pk_{ca}^{-1}} \\
6 & ES \mapsto U & : & \{name_U, pk_U\}_{pk_{ca}^{-1}}
\end{array}
$$

avendo indicato con U un utente che richiede il certificato, con ES il server d'iscrizione, con LRA l'operatore RA che approva le richieste, con RA il server di Registration Authority, con CA la workstation che implementa l'Autorità di Certificazione. $name_U$, pk_U, pk_U^{-1} e pin_U sono, rispettivamente, il nome, la chiave pubblica, la chiave privata ed il pin dell'utente e pk_{ca}^{-1} è la chiave privata della CA, che con la sua firma convalida l'emissione dei certificati. Si è scelto di rappresentare il documento di riconoscimento del richiedente con la quantità $\{name_U\}_{pk_{Gov}^{-1}}$, ovvero il suo nome firmato con la chiave privata del governo. I primi due passi, la connessione al server di iscrizione da parte del richiedente, ed il riconoscimento di questi da parte dell'operatore LRA (azione modellata formalmente con l'invio del documento e del pin), sono comunicazioni protette: i messaggi scambiati sono criptati con apposite chiavi pubbliche, pk_{c1} e pk_{c2}, le cui corrispondenti chiavi private sono in possesso esclusivo del server di iscrizione e dell'operatore di RA. Il certificato digitale viene rappresentato, in notazione formale, dalla quantità $\{name_U, pk_U\}_{pk_{ca}^{-1}}$, ovvero dal nome e dalla chiave pubblica del richiedente firmati dalla chiave privata della CA.

In generale, una Autorità di Certificazione deve poter soddisfare due importanti proprietà:

1. la CA non deve emettere certificati in cui il nome di un legittimo utente sia legato alla chiave pubblica di un altro (l'intruso);

2. la CA non deve emettere certificati in cui la chiave pubblica di un legittimo utente sia legata al nome di un altro.

Il fallimento della prima proprietà può causare il cosiddetto "attacco di responsabilita", ovvero la possibilità che qualcuno firmi documenti e renda responsabile per la firma un'altra persona; il fallimento della seconda può causare l' "attacco di credito", ovvero qualcuno può pretendere di avere originato un documento firmato in realtà da un'altra persona.
Formalmente, la notazione per due certificati dalle caratteristiche precedenti è la seguente:

1. $\{name_U, pk_X\}_{pk_{ca}^{-1}}$

2. $\{name_X, pk_U\}_{pk_{ca}^{-1}}$

avendo indicato con $name_X$ e pk_X il nome e la chiave pubblica dell'intruso.

Con il supporto di PaMoChSA è stata analizzata la possibilità di generare erroneamente questo tipo di certificati, ed è stato mostrato come la perdita di particolari informazioni registrate sul server RA possa condurre all'emissione, da parte della CA, di certificati indesiderati. In particolare, è stata riscontrata l'esistenza di due diverse tipologie di attacco:

- l'intruso conosce la sequenza casuale (pin) che l'utente legittimo ha inserito nelle form cartacea ed elettronica di richiesta del certificato: tale circostanza può comportare la generazione di un certificato di tipo 1;

- l'intruso conosce lo SPKAC dell'utente $\{pk_U, n_U\}_{pk_U^{-1}}$: tale circostanza può comportare la generazione di un certificato di tipo 2.

L'intruso può ottenere tali informazioni tramite attacco diretto al server RA ,il quale, a differenza della workstation CA, è una macchina più vulnerabile poiché in rete. Nel seguito si riporta la descrizione del secondo tipo di attacco.

Per l'analisi, il tool richiede la tripla di ingressi:

- (**Descrizione del protocollo**) OpenCA_SPKAC_noto.

- (**Formula**) $\{name_X, pk_U\}_{pk_{ca}^{-1}}$.

- (**Conoscenza iniziale dell'intruso**)

  $name_X, pin_X, pk_X, \{name_X\}_{pk_{Gov}^{-1}}, pk_U, name_U, pk_{c1}$

Il trucco descrittivo utilizzato nella specifica del protocollo per far conoscere all'intruso lo SPKAC del richiedente è quello di spedirlo su un canale accessibile a chiunque, in modo tale che anche l'intruso abbia la possibilità di riceverlo ed aggiungerlo alla sua conoscenza iniziale. L'inserimento nella conoscenza iniziale dell'intruso della chiave pubblica pk_{c1} significa assumere che lo stesso è in grado di connettersi in modo sicuro al server di iscrizione per richiedere un certificato.

La procedura di attacco consiste nei seguenti passi:

1. l'utente U si collega al server di iscrizione e richiede un certificato;
2. l'intruso X, venuto a conoscenza dello SPKAC di U, si collega al server di iscrizione e fa una richiesta di certificato inserendo il suo nome $name_X$, una sequenza casuale pin_X, la chiave pubblica dell'utente U, pk_U, e lo SPKAC dell'utente U;
3. X contatta l'operatore LRA per certificare che i dati inseriti nella form elettronica, tra cui il pin, sono corretti e presenta il suo documento di riconoscimento;
4. l'operatore si connette al server RA ed approva la richiesta corrispondente al nome $name_X$ ed alla chiave pk_U;
5. la richiesta di certificato è importata nel server CA dal server RA;
6. l'operatore CA, una volta controllato che la richiesta inserita nella sezione "pendenti" sia correttamente firmata dall'operatore LRA, applica la chiave pubblica dell'utente U allo SPKAC per verificarne la firma. Se il controllo ha buon esito, emette il certificato $\{name_X, pk_U\}_{pk_{ca}^{-1}}$ e lo importa nel server RA;
7. X si connette al server di iscrizione per scaricare il certificato.

In figura 4 si riporta la descrizione formale dell'attacco ed il risultato dell'elaborazione cosi come appare attraverso l'interfaccia grafica di PaMoChSA .

$$
\begin{array}{lllll}
1 & U \mapsto ES & : & \{name_U, pk_U, pin_U, \{pk_U, n_U\}_{pk_U^{-1}}\}_{pk_{c1}} \\
2 & X \mapsto ES & : & \{name_X, pk_U, pin_X, \{pk_U, n_U\}_{pk_U^{-1}}\}_{pk_{c1}} \\
3 & X \mapsto LRA & : & \{\{name_X\}_{pk_{Gov}^{-1}}, pin_X\}_{pk_{c2}} \\
4 & LRA \mapsto RA & : & \{name_X, pin_X\} \\
5 & RA \mapsto CA & : & \{name_X, pk_U, pin_X, \{pk_U, n_U\}_{pk_U^{-1}}\} \\
6 & CA \mapsto RA & : & \{name_X, pk_U\}_{pk_{ca}^{-1}} \\
7 & ES \mapsto U & : & \{name_X, pk_U\}_{pk_{ca}^{-1}}
\end{array}
$$

Figure 4: Attacco di credito: l'intruso è capace di forzare l'emissione di un certificato in cui la sua chiave pubblica è legata al nome dell'utente U.

4.4 Una nota sull'attacco SPKAC

L'attacco illustrato nella precedente sezione funziona perché l'informazione contenuta nello SPKAC non viene interamente sfruttata nei controlli. L'architettura software analizzata si basa su OpenCA (ver 0.20) e OpenSSL (ver 0.94); nella relativa documentazione è esplicitamente dichiarato che la sequenza casuale (nonce) n_U contenuta nello SPKAC non viene utilizzata ai fini di un controllo sullo stesso per cui, nella descrizione formale della nostra specifica, abbiamo omesso di verificare la replica della nonce. Al contrario, inserendo una registrazione degli SPKAC contenuti nelle varie richieste di certificato, al passo 6 l'operatore CA può rendersi conto che la nonce utilizzata da X è la stessa presente nello SPKAC della richiesta di U e può fermare le operazioni di rilascio. Da queste considerazioni segue che, eseguendo correttamente i controlli su SPKAC, una singola CA non emette certificati errati. Al contrario, nell'ambito di PKI più complesse, anche il controllo sulla nonce può fallire. Si consideri infatti una struttura gerarchica, costituita da una CA radice che ha rilasciato i certificati per due sub-CA, le quali operano direttamente con gli utenti; si supponga che un utente A abbia scoperto lo SPKAC di un utente B che ha precedentemente ottenuto un certificato dalla prima delle due sub-CA: A può fare alla seconda sub-CA una richiesta di certificato contenente il suo nome e lo SPKAC di B. In questo caso, il controllo sulla nonce si rivela inutile, infatti la seconda CA riceve lo SPKAC per la prima volta.

Un tentativo per risolvere questo problema è l'inserzione, nella struttura dello SPKAC, di una informazione relativa alla CA che deve emettere il certificato, così come previsto dallo standard PKCS#10, istituito da RSA Laboratories (vedi [14]) e che fa parte degli standard per la crittografia a chiave pubblica (Public - Key Cryptography Standards). PKCS#10 è uno standard "de-facto", e descrive la sintassi per le richieste di certificati; tali richieste prevedono, tra l'altro, un campo informazione, costituito da:

- il nome del soggetto la cui chiave pubblica deve essere certificata;
- il cosiddetto "SubjectPublicKeyInfo", contenente informazioni sulla chiave pubblica da certificare;
- un set di attributi che danno informazioni addizionali riguardo al soggetto.

Il campo informazione è firmato con la chiave privata del soggetto.

Una tale struttura contiene quindi il nome del soggetto, per cui non ha alcun senso che un intruso attui la richiesta come nel passo 2 della sezione 4.3:

$$X \mapsto ES \quad : \quad \{name_X, pk_U, pin_X, \{name_U, pk_U, attributes\}_{pk_U^{-1}}\}_{pk_{c1}}$$

Infatti l'operatore RA ha la possibilità di verificare che il nome del soggetto richiedente è diverso da quello contenuto nella struttura PKCS#10. L'analisi effettuata con PaMoChSA ha condotto alla modifica della versione di OpenCA utilizzata fino alla scoperta dell'attacco, ammettendo anche richieste di certificazione che adottano lo standard PKCS#10.

5 Un compilatore automatico per l'analisi dei protocolli

Tra gli ingressi necessari a PaMoChSA , uno è rappresentato dalla specifica del protocollo da analizzare. Il linguaggio utilizzato per scrivere tale specifica si chiama Crypto-CCS ([7]), ed è sostanzialmente una variante dell'algebra di processi CCS di R. Milner; tale linguaggio è più concreto rispetto alla sequenza di messaggi comunemente utilizzata in letteratura per la descrizione dei protocolli. Crypto-CCS descrive qualsiasi interazione tra i partecipanti al protocollo per mezzo di comunicazioni (spedire e ricevere messaggi). La stesura della specifica in Crypto-CCS richiede la conoscenza del linguaggio stesso e soprattutto tempi non trascurabili; inoltre, anche persone esperte possono commettere errori che portano alla stesura di una specifica diversa da quella prevista.

Per aggiungere funzionalità e maggiore automazione a PaMoChSA abbiamo avvertito l'esigenza di progettare un compilatore automatico per mappare la descrizione dei protocolli comunemente nota in letteratura nella specifica d'ingresso in Crypto-CCS richiesta dal tool. Sostanzialmente, l'utente potrà scrivere un file di input descrivente il protocollo in una notazione astratta derivante da quella letteraria ed il compilatore tradurrà tale file in codice Crypto-CCS.

Se da un lato la notazione letteraria è più semplice da utilizzare, dall'altro però molti controlli non sono esplicitamente dichiarati. Si consideri a titolo di esempio il seguente scambio di messaggi tra due partecipanti A e B:

$$\begin{array}{lll} 1 \quad A \mapsto B & : & \{name_A, n_A\}_{pk_A^{-1}} \\ 2 \quad B \mapsto A & : & \{name_B, n_A, n_B\}_{pk_B^{-1}} \\ 3 \quad A \mapsto B & : & \{n_B\} \end{array}$$

Nel passo 1 del protocollo A invia a B il suo nome ed una sequenza casuale da lui inventata, *n_A*, entrambe firmate con la sua chiave privata; nel passo 2 B invia ad

A il suo nome, *n_A* e la sua sequenza casuale *n_B*, anch'esse firmate con la chiave privata di B; infine A rimanda a B *n_B*. Il buon senso suggerisce, leggendo la specifica in questa notazione, che i due partecipanti, per completare positivamente il protocollo, siano in grado:

- di verificare le firme presenti nei messaggi scambiati al passo 1 e 2, utilizzando opportune chiavi pubbliche per la decifratura e l'ottenimento delle singole parti di un messaggio (quali n_A, $name_A$, ecc.);
- di associare a nomi uguali lo stesso oggetto (ovvero, la nonce n_A che B spedisce nel passo 2 deve essere proprio la stessa da lui ricevuta nel passo 1, ecc.).

La specifica scritta in Crypto-CCS prevede la presenza di verifiche del genere, poiché al momento della stesura sono esplicitamente inseriti a mano tutti i controlli suggeriti dal buon senso; questa esplicitazione deve essere resa automatica dal compilatore: prevedendo di prendere in ingresso un file contenente i tre passi descritti, il file d'uscita deve contenere non solo comandi che descrivono la spedizione e la ricezione di quantità il cui nome è n_A, n_B, ecc., bensì le stesse azioni devono essere subordinate a verifiche di fattibilità da parte dei partecipanti.

La decifratura di un messaggio cifrato da parte di un partecipante sarà soggetta al possesso di una opportuna chiave di cifratura (e lo stesso varrà per la cifratura); alla ricezione di un qualsiasi messaggio, a questo verrà assegnato un particolare identificatore e verranno fatti dei controlli su ricezioni e spedizioni precedenti, per cercare corrispondenza tra eventuali passate occorrenze di tutto il messaggio o di una sua parte. In Crypto-CCS inoltre i messaggi sono "tipati", ovvero a ciascuno viene associato un particolare tipo tramite l'operatore ":". Poiché in notazione letteraria non è prevista l'associazione del tipo, il traduttore dovrà essere in grado di dedurlo automaticamente, associando per esempio ad un nome il tipo "Name", ad una chiave il tipo "Key", ecc.

In molti protocolli, infine, è previsto che un agente riceva messaggi cifrati da non decifrare, ma semplicemente da redirigere ad una terza parte in un successivo passo del protocollo. Per trattare questi casi, si è deciso di adottare la notazione % utilizzata dal compilatore Casper (vedi [13]): nel file di input questo particolare tipo di messaggio sarà della forma m%v ed il partecipante A, al momento della sua ricezione, non dovrà preoccuparsi di possedere una determinata chiave per la decifratura, ma si limiterà ad immagazzinare m nell'identificatore v. In un successivo passo del protocollo, A spedirà ad una terza parte il messaggio, utilizzando la notazione contraria v%m: questo vuol dire che il mittente A spedirà il contenuto

di v, e sarà compito della terza parte verificare che questo coincida proprio con m.

Al momento attuale, la progettazione del compilatore è stata completata a livello concettuale. Il compilatore è sviluppato in ambiente di lavoro *ocaml* e la fase implementativa ha finora previsto lo sviluppo di funzioni per l'inferenza automatica del tipo dei messaggi e per la ricerca di occorrenze precedenti di un dato messaggio (o di una parte di esso) in particolari insiemi ordinati.

6 Conclusioni

Notevole è l'interesse sviluppatosi negli ultimi tempi per tematiche riguardanti la sicurezza delle comunicazioni in rete. L'Istituto per le Applicazioni Telematiche del CNR svolge attività di ricerca su aspetti teorici relativi alla firma digitale ed alle PKI, ed ha sviluppato infrastrutture a chiave pubblica sia per sperimentazioni interne all'istituto sia per il rilascio di certificati digitali nell'ambito di un progetto di posta elettronica sicura tra l'Istituto stesso e gli Internet Service Provider.

Tematiche specifiche interne all'Istituto riguardano lo studio dei metodi formali per l'analisi dei protocolli di sicurezza. In particolare, è stato sviluppato un tool, PaMoChSA , che, tramite la sua interfaccia grafica, offre la possibilità di verificare agevolmente la correttezza di svariati protocolli di comunicazione. Allo stato attuale, il tool è in grado di analizzare proprietà di segretezza e alcune forme di autenticazione. In questo rapporto, è stata presentata l'applicazione di PaMoChSA all'analisi della procedura di rilascio di certificati di una Autorità di Certificazione; in particolare la CA analizzata si basa sul software OpenCA, ma la stessa tecnica può essere utilizzata per l'analisi di altre CA e di protocolli che sfruttano algoritmi crittografici. L'analisi ha mostrato come la conoscenza di particolari informazioni dia ad un intruso la possibilità di forzare la CA ad emettere certificati indesiderati.

È in corso d'implementazione un compilatore per la traduzione automatica delle specifiche dei protocolli dalla notazione comune in letteratura al particolare codice richiesto in ingresso a PaMoChSA ; la funzionalità del tool migliorerà notevolmente a compilatore ultimato.

Bibliografia

[1] G. Lowe. Breaking and fixing the Needham Schroeder public-key protocol using FDR. In *Proceedings of Tools and Algorithms for the Construction and the Analisys of*

Systems, volume 1055 di *Lecture Notes in Computer Science*, pages 147–166. Springer Verlag, 1996.

[2] G. Lowe, B. Roscoe. Using CSP to detect errors in the TMN Protocol. In *IEEE Transactions on Software Engineering*, 23(10), pagg. 659-669, 1997.

[3] M. Abadi, A.D. Gordon. Reasoning about Cryptographic Protocols in the Spi Calculus. In *CONCUR '97: Concurrency Theory, 8th International Conference*, volume 1243 di *Lecture Notes in Computer Science*, pagg. 59-73, 1997.

[4] R. Focardi, R. Gorrieri. A Classification of Security Properties. In *Journal of Computer Security*, 3(1), pagg. 5-33, 1995.

[5] R. Focardi, R. Gorrieri, F. Martinelli. Non Interference for the Analysis of Cryptographic Protocols. In *Proceedings of 27th International Colloquium in Automata, Languages and Programming*, volume 1853 di *Lectures Notes in Computer Science*, pagg. 354-372, 2000.

[6] F. Martinelli. Partial Model Checking and Theorem Proving for Ensuring Security Properties. In *Proceedings of 11th Computer Security Foundations Workshop*, pagg. 44-52. IEEE Computer Society Press, 1998.

[7] F. Martinelli. Languages for Description and Analysis of Authentication Protocols. In *Proceedings of 6th Italian Conference on Theoretical Computer Science*, pagg. 304-315. World Scientific, 1998.

[8] D. Dolev and A. Yao. On the security of public key protocols. In *IEEE Transactions on Information Theory*, 29(12):198–208, 1983.

[9] P. Gardiner, D. Jackson, B. Roscoe. Security modelling in CSP and FDR: Deliverable Boundle 3. Technical Report, Formal Systems (Europe) Ltd, Giugno 1996.

[10] http://caml.inria.fr/hump.html#graph.

[11] http://cristal.inria.fr/ cuoq/mlgtk.html

[12] http://pauillac.inria.fr/ocaml/

[13] G. Lowe. Casper: a Compiler for the Analysis of Security Protocols. In *Proceedings of the 10th Computer Security Foundations Workshop*, pagg 18-30, IEEE Computer Society Press, 1997. http://www.mcs.le.ac.uk/ glowe/Security/Casper/

[14] http://www.rsasecurity.com/rsalabs/pkcs/pkcs-10/index.html

[15] F. B. Schneider. Applied Cryptography, J. Wiley & sons, Inc., 1996.

[16] A. Giani, A. Vaccarelli. Note introduttive alla crittografia e alla firma digitale. IAT-CNR, Technical Report, IAT-B4-2000-009, 2000.

[17] http://pki-ra.iat.cnr.it/

[18] http://www.openca.org/

[19] http://www.openssl.org/