

Università degli Studi di Bologna

FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA IN INGEGNERIA INFORMATICA
CALCOLATORI ELETTRONICI II



**Sistemi per la sicurezza nella gestione di
documenti informatici:
un'applicazione in ambito universitario**

Tesi di Laurea di:
MIRKO TEDALDI

Relatore:
Chiar.mo Prof. Ing. MAURELIO BOARI

Correlatori:
Chiar.ma Prof.ssa Ing. ANNA CIAMPOLINI
Chiar.mo Prof. Ing. ROBERTO LASCHI
Dott.ssa Ing. REBECCA MONTANARI

Anno Accademico 1998-99

Parole Chiave:

Crittografia
Documento Informatico
Firma Digitale
Smart Card
Sicurezza

INDICE

INTRODUZIONE	1
CAPITOLO 1. LA CRITTOGRAFIA A CHIAVE PUBBLICA	5
1.1 INTRODUZIONE.....	5
1.2 CRITTOGRAFIA A CHIAVE SIMMETRICA.....	7
1.3 SISTEMI A CRITTOGRAFIA A CHIAVE PUBBLICA.....	9
1.4 LA FIRMA DIGITALE.....	11
1.5 GLI ALGORITMI	15
1.5.1 RSA.....	15
1.5.2 SHA-1.....	17
CAPITOLO 2. INFRASTRUTTURE A CHIAVE PUBBLICA	19
2.1 CERTIFICATI ELETTRONICI.....	19
2.2 LO STANDARD X.509 PER I CERTIFICATI.....	21
2.3 LA NECESSITÀ DI COPPIE DI CHIAVI DISTINTE PER CIFRATURA E FIRMA	24
2.4 INFRASTRUTTURA A CHIAVE PUBBLICA (PKI)	26
2.5 REQUISITI DI UNA PKI.....	29
2.6 CICLO DI VITA DEI CERTIFICATI	31
2.7 LA GESTIONE DELLE CHIAVI	32
2.8 LA REVOCA DI UN CERTIFICATO	33
2.9 I CAMMINI DI CERTIFICAZIONE	37
2.10 PRODOTTI COMMERCIALI PER LE PKI.....	40
CAPITOLO 3. IL DOCUMENTO INFORMATICO IN ITALIA	43
3.1 LA “LEGGE BASSANINI”	43
3.2 L’ALLEGATO TECNICO DELL’AIPA	45
3.2.1 <i>Le regole tecniche di base</i>	46
3.2.2 <i>Regole tecniche per la certificazione delle chiavi</i>	48
3.2.3 <i>Regole per la validazione temporale</i>	50
3.2.4 <i>Regole tecniche per le pubbliche amministrazioni</i>	51
CAPITOLO 4. SMART CARD.....	53
4.1 INTRODUZIONE: PERCHÉ LA SCELTA DELLE SMART CARD.....	53
4.2 UNA PANORAMICA: DALLA NASCITA AI POSSIBILI SCENARI FUTURI.....	54
4.3 CHE COS’È UNA SMART CARD.....	56
4.4 CARTE CON E SENZA CONTATTO	58
4.5 STANDARD	59
4.5.1 <i>Lo standard ISO 7816</i>	59
4.5.2 <i>Standard per la comunicazione</i>	61
4.6 LE SMART CARD PER APPLICAZIONI CRITTOGRAFICHE	64
4.6.1 <i>Active RSA</i>	64
4.6.2 <i>Key Generation</i>	65
4.6.3 <i>Capacità di memorizzazione</i>	66

4.7	PARTICOLARI SMART CARD.....	67
4.7.1	<i>Smart card biometriche</i>	67
4.7.2	<i>Java Card</i>	68
CAPITOLO 5. LE SPECIFICHE DEL PROGETTO.....		71
5.1	INTRODUZIONE: LE ENTITÀ DEL SISTEMA.....	71
5.2	INIZIALIZZAZIONE DEI DOCENTI.....	73
5.3	INIZIALIZZAZIONE DELL'APPLICAZIONE SERVER.....	77
5.4	VERBALIZZAZIONE DEGLI ESAMI.....	78
5.5	IL PROTOCOLLO DI COMUNICAZIONE.....	82
5.5.1	<i>Fallimento della verifica della firma del docente</i>	83
5.5.2	<i>Fallimento delle verifica delle capability del docente</i>	85
5.5.3	<i>Fallimento della verifica della firma del server</i>	85
5.5.4	<i>Problemi dovuti alla rete di interconnessione</i>	86
5.5.5	<i>Fault Tolerance dei sistemi di memorizzazione</i>	88
5.5.6	<i>Recovery dei verbali non trasmessi</i>	88
CAPITOLO 6. IMPLEMENTAZIONE		91
6.1	INTRODUZIONE.....	91
6.2	LA PROGETTAZIONE DELLA PKI.....	91
6.2.1	<i>Aggiunta dell'utente alla Directory</i>	93
6.2.2	<i>Abilitazione dell'utente</i>	94
6.2.3	<i>Inizializzazione dell'utente</i>	95
6.3	IL SERVIZIO DI MARCATURA TEMPORALE (TIMESTAMPING).....	97
6.3.1	<i>Introduzione al timestamping</i>	97
6.3.2	<i>L'implementazione della marcatura temporale con Entrust/Timestamp</i>	99
6.4	SMART CARD.....	102
6.5	ENTRUSTFILE TOOLKIT.....	103
6.5.1	<i>La scelta del toolkit di sviluppo</i>	103
6.5.2	<i>Le funzionalità di EntrustFile</i>	103
6.6	REALIZZAZIONE DEI MECCANISMI DI SICUREZZA: LA CLASSE UTENTEENTRUST.....	105
6.7	LE CLASSI PER LA COMUNICAZIONE.....	110
6.8	L'APPLICAZIONE CLIENT.....	110
6.9	L'APPLICAZIONE SERVER.....	115
6.10	LE STRUTTURE DATI.....	117
6.10.1	<i>Record d'esame</i>	117
6.10.2	<i>Contenuto floppy disk</i>	117
6.10.3	<i>Il file delle capability</i>	118
CAPITOLO 7. RISULTATI		121
7.1	PRESTAZIONI DELLE OPERAZIONI CRITTOGRAFICHE.....	121
7.2	LE STRUTTURE DATI.....	124
7.2.1	<i>I profili utenti</i>	124
7.2.2	<i>Dimensioni dei verbali</i>	126
7.2.3	<i>Il file delle capability</i>	127
7.3	I TEST DELL'APPLICAZIONE.....	128

CAPITOLO 8. CONCLUSIONI	135
APPENDICE A. ENTRUST	137
APPENDICE B. ENTRUSTFILE TOOLKIT	147
BIBLIOGRAFIA	151

INTRODUZIONE

La diffusione capillare delle reti di calcolatori, e la conseguente apertura derivante dalla loro interconnessione, soprattutto grazie all'espansione di Internet, sta portando mutamenti profondi nella società dove ormai in tutti gli ambienti vengono sempre più spesso utilizzate queste nuove tecnologie di comunicazione per lo scambio di informazioni. Uno degli effetti di questo fenomeno sociale e tecnologico è la progressiva sostituzione del tradizionale documento cartaceo con quello informatico.

Indubbiamente l'utilizzo del documento informatico comporta innumerevoli vantaggi: la possibilità di scambiare informazioni in tempi rapidissimi e senza vincoli geografici (ad esempio, utilizzando la posta elettronica), l'incremento nella velocità di trattamento dell'informazione, la praticità nell'archiviazione. D'altra parte, un documento informatico, che viaggia attraverso una rete insicura come Internet, può subire alterazioni, contraffazioni, utilizzi illeciti; pertanto, l'utilizzo del documento informatico non può prescindere dalla progettazione di infrastrutture di sicurezza che siano in grado di garantire l'integrità, l'autenticità, la riservatezza e il non ripudio dei dati.

La crittografia a chiave pubblica offre la possibilità di realizzare questi servizi di sicurezza; in particolar modo, l'utilizzo della firma digitale, al pari della firma autografa, consente di autenticare un documento informatico, di preservarne l'integrità e di evitarne un eventuale ripudio.

La firma digitale è sicuramente uno degli argomenti di maggior attualità in campo informatico e organizzativo; a tale proposito basta osservare l'attenzione che ad essa è stata dedicata dai legislatori italiani e stranieri: infatti, sia a livello italiano che a

livello europeo, si stanno definendo le norme tecnico-giuridiche per dare validità legale ai documenti informatici firmati digitalmente; da questo punto di vista l'Italia è all'avanguardia grazie alla "legge Bassanini" (la legge 15 marzo 1997, n.59).

La legalizzazione del documento informatico apre nuove potenzialità applicative soprattutto nell'ambito del commercio elettronico, delle transazioni finanziarie e, in particolar modo, della Pubblica Amministrazione, nella quale si ha la possibilità di snellire il sistema burocratico.

Come specificato nella "legge Bassanini", l'utilizzo della firma digitale richiede l'esistenza di Autorità di Certificazione che, emettendo i cosiddetti certificati elettronici, garantiscono la validità delle chiavi pubbliche dei loro membri. Più in generale, l'utilizzo di tecniche crittografiche a chiave pubblica richiede la realizzazione di infrastrutture a chiave pubblica (PKI), il cui compito è quello di gestire il ciclo di vita delle chiavi crittografiche e dei certificati elettronici, garantendo in questo modo l'autenticità nella comunicazione.

Inoltre, la "legge Bassanini", tramite l'allegato tecnico definito dall'AIPA (Autorità per l'Informatica nella Pubblica Amministrazione), definisce precise norme per quanto riguarda il processo di generazione della firma digitale al fine di garantire l'integrità e la non violabilità della chiave privata di firma. Per soddisfare i requisiti dell'AIPA nasce l'esigenza di ricorrere alla tecnologia delle "smart card", che, se dotate di particolari caratteristiche, possono essere utilizzate come dispositivi di firma.

Lo scopo di questa tesi è quello di sperimentare l'utilizzo della firma digitale all'interno dell'organizzazione universitaria. A tal fine è stata individuata, come possibile applicazione pilota, la verbalizzazione degli esami con l'intento di alleggerire e accelerare l'intero processo, dalla compilazione del verbale da parte del docente alla verifica e registrazione da parte delle segreterie di

facoltà. L'obiettivo è realizzare un sistema sicuro per la gestione di documenti informatici che consenta ai docenti di redigere, firmare digitalmente e trasmettere alla segreteria i verbali d'esame attraverso l'utilizzo delle smart card.

Inizialmente è stata svolta un'analisi della "legge Bassanini" e dell'allegato tecnico dell'AIPA per capire quali fossero le norme tecniche da soddisfare per poter dare validità legale al documento informatico. Questa analisi ha evidenziato la necessità dell'utilizzo delle smart card come dispositivo atto alla generazione, conservazione e utilizzo delle chiavi necessarie nel processo di firma digitale. A tal fine è stato compiuto uno studio sulle smart card per comprendere lo stato dell'arte di tale tecnologia e il loro utilizzo per applicazioni crittografiche.

Successivamente si è affrontato il problema di come realizzare l'infrastruttura a chiave pubblica. Tra i vari prodotti disponibili sul mercato si è scelto di utilizzare Entrust, sicuramente uno dei migliori per la completezza dei servizi forniti.

Una volta a disposizione delle conoscenze e degli strumenti necessari, si è passati a definire le specifiche del progetto e alla realizzazione dell'applicazione.

La tesi è strutturata come segue:

Il primo capitolo introduce la crittografia a chiave pubblica, mostrando quali sono le principali problematiche nell'ambito della sicurezza e le possibili soluzioni offerte da questa tecnologia, soffermandosi in particolare sulla firma digitale.

Il capitolo due illustra le infrastrutture a chiave pubblica: i requisiti di progettazione, il ciclo di vita dei certificati elettronici, la gestione delle chiavi crittografiche, i prodotti commerciali disponibili per la realizzazione.

Nel terzo capitolo si analizza la "legge Bassanini" e le norme tecniche dell'AIPA, allo scopo di capire in quale contesto

legislativo si inserisce il documento informatico e quali implicazioni tecnologiche e organizzative comporta l'utilizzo della firma digitale in Italia.

Il quarto capitolo propone uno studio sulla tecnologia delle smart card: si fa una panoramica sulla loro storia e sui possibili scenari futuri, si considerano quali sono gli standard attualmente adottati e si esamina come questa tecnologia possano integrarsi con l'utilizzo della firma digitale.

Il quinto capitolo fornisce le specifiche del progetto, mentre il sesto capitolo descrive la realizzazione dell'applicazione distribuita per la verbalizzazione degli esami.

Il capitolo sette analizza i risultati ottenuti nelle prove sperimentali, mentre l'ottavo capitolo riporta le conclusioni del lavoro svolto e vengono evidenziate le linee di sviluppo per il lavoro futuro.

Infine vengono fornite due appendici dove si analizza in dettaglio il prodotto utilizzato per la realizzazione della PKI e il toolkit di sviluppo usato in fase di programmazione.

Capitolo 1. LA CRITTOGRAFIA A CHIAVE PUBBLICA

1.1 Introduzione

La crittografia [RSA98] [S96] è lo studio della codifica e della decodifica dei dati. Il termine *crittografia* viene dalle parole greche *kryptos* che significa *nascosto*, e *graphia*, che significa *scrittura*.

La crittografia riveste un ruolo preminente nell'ambito dei meccanismi di sicurezza, essendo in grado di fornire la maggior parte dei servizi contemplati nell'architettura di sicurezza stabilita dall'ISO (International Organization for Standardization); pertanto i sistemi crittografici rivestono un'importanza fondamentale nella soluzione di molte problematiche di sicurezza connesse con la protezione di informazioni.

Per inquadrare il problema è opportuno rifarsi al modello di riferimento proposto dall'ISO che individua cinque classi di funzionalità necessarie per garantire il desiderato livello di sicurezza. Queste sono:

1. *confidenzialità*: con tale funzionalità si vuole impedire di rilevare informazioni riservate ad entità non autorizzate alla conoscenza di tali informazioni; la confidenzialità è ottenuta tramite tecniche crittografiche di cifratura dei dati.
2. *integrità dei dati*: i servizi di integrità dei dati proteggono contro gli attacchi attivi¹ finalizzati ad alterare illegittimamente il valore di un dato; l'alterazione di un messaggio può

¹ Si definisce attacco passivo un attacco con cui un intrusore intercetta semplicemente un flusso di informazioni, senza apportare alcun genere di modifiche; con un attacco attivo, invece, l'intrusore introduce alterazioni al flusso di dati che possono comprendere modifica del contenuto informativo, eliminazione di tutto o solo di alcune parti del messaggio o trattenimento del messaggio per poi replicarlo in tempi successivi.

comprendere la cancellazione, la modifica, o il cambiamento dell'ordine dei dati

3. *autenticazione*: i servizi di autenticazione garantiscono l'accertamento dell'identità; quando a qualcuno (o a qualcosa) si attribuisce una certa identità, i servizi di autenticazione forniscono lo strumento con cui verificare la correttezza di tale affermazione di identità; i servizi di autenticazione si suddividono in:
 - servizi di autenticazione dell'entità: in questo caso si autentica l'identità presentata ad un'entità remota che partecipa ad una sessione di comunicazione (le password sono un esempio tipico di strumenti atti ad ottenere autenticazione dell'entità).
 - servizi di autenticazione dell'origine dei dati: in questo caso si autentica l'identità che l'originatore di un messaggio rivendica di avere;
4. *controllo degli accessi*: una volta che l'autenticazione è avvenuta, è possibile eseguire un controllo degli accessi in modo tale da verificare che vengano utilizzate solo quelle risorse o servizi ai quali si è autorizzati.
5. *non ripudio*: tali servizi hanno la finalità di impedire che un'entità riesca successivamente a rinnegare con successo la partecipazione a transazione o comunicazione a cui in realtà ha preso parte; i servizi di non ripudio non prevengono il ripudio di una comunicazione o di una transazione, forniscono, invece, gli elementi per dimostrare in caso di contenzioso l'evidenza dei fatti. Se è vero che autenticazione ed integrità dei dati sono elementi essenziali nell'implementazione di un servizio di non ripudio senza, il non ripudio, tuttavia, va oltre le problematiche di autenticazione ed integrità: è, infatti, lo strumento con cui dimostrare ad una terza parte e dopo l'accadimento del fatto che una comunicazione o transazione è stata originata o avviata da

una certa entità o consegnata ad una certa entità. Si pensi ad esempio al caso di una persona, Alice, che invia un ordine di acquisto di beni a Bob; la paternità dell'ordine di acquisto viene attribuita con assoluta certezza ad Alice se si adotta un servizio corretto di non ripudio; in caso di contenzioso Bob può dimostrare con assoluta certezza che l'ordine di acquisto è stato effettuato da Alice. Nei documenti cartacei, quali contratti, ordini, bonifici, è la firma autografa ad essere utilizzata per garantire il servizio di non ripudio, nei documenti elettronici è, invece, la tecnica crittografica di firma digitale.

Le principali tecniche crittografiche, cifratura e firma digitale, costituiscono i componenti basilari nell'implementazione di tutti i servizi di sicurezza sopra descritti. Alla base delle tecniche crittografiche si colloca l'algoritmo crittografico che fornisce le funzioni di trasformazione di un testo da testo in chiaro a testo cifrato. Gli algoritmi crittografici si suddividono in due classi distinte: algoritmi a chiave simmetrica e algoritmi a chiave pubblica. Fino a poco più di vent'anni fa erano conosciuti solo gli algoritmi a chiave simmetrica; nel 1976 Diffie ed Hellman [DH76] presentarono un protocollo per lo scambio di una chiave segreta attraverso un canale insicuro; tale meccanismo era stato inteso essenzialmente per risolvere il problema dell'avvio di un normale sistema di cifratura a chiavi simmetriche ma in realtà ha posto le basi della crittografia a chiave pubblica.

1.2 Crittografia a chiave simmetrica

Gli algoritmi simmetrici utilizzano un singola chiave matematica sia per la cifratura che per la decifratura dei dati. Un messaggio sicuro viene cifrato per il destinatario utilizzando una chiave nota solamente al mittente ed al destinatario. Il principio

matematico su cui si basa l'algoritmo è che cifrando un testo in chiaro con un certa chiave e decifrando il testo cifrato con la medesima chiave si ottiene nuovamente il testo in chiaro (vedi figura 1.1):

$$D_k(E_k(P)) = P$$

con P = testo in chiaro, E = operazione di cifratura,
 D = operazione di decifratura, K = chiave

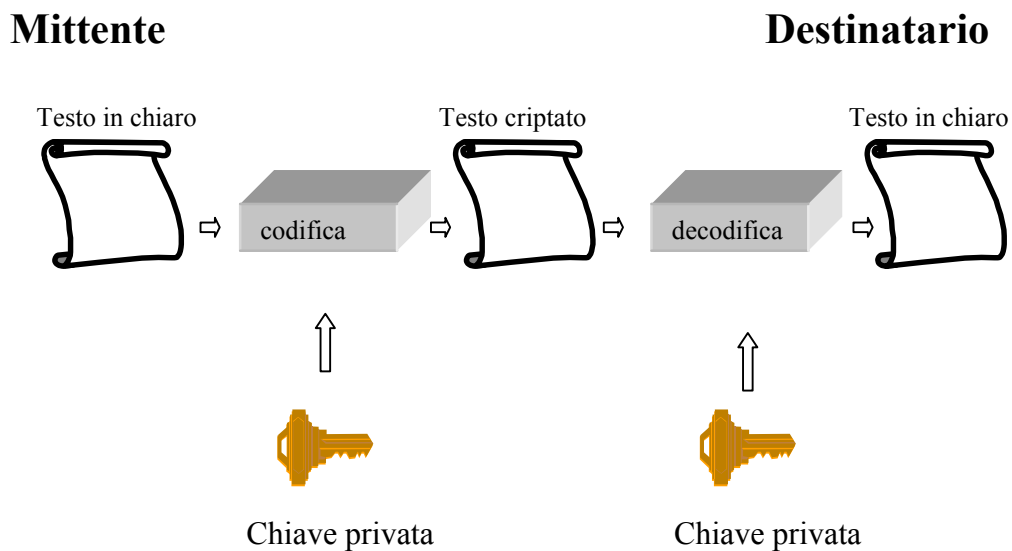


Figura 1.1: Schema di funzionamento di un algoritmo simmetrico

Il principale problema implementativo di algoritmi a chiave simmetrica è dovuto alla difficoltà di distribuire le chiavi in modo sicuro e su larga scala. Infatti, l'intera sicurezza dell'algoritmo si basa sulla segretezza della chiave che deve essere conosciuta solo dai due interlocutori e pertanto si pone il problema di come poter distribuire tale chiave in maniera sicura (non si potranno utilizzare canali intrinsecamente insicuri come, ad esempio, Internet). Inoltre, se si assume che una chiave distinta è richiesta per ogni coppia di

interlocutori, il numero di chiavi da distribuire aumenta drasticamente all'aumentare delle parti in gioco; se, infatti, n sono le parti coinvolte, il numero di chiavi da distribuire è pari a $n*(n-1)/2$.

A questo va aggiunto che un protocollo di comunicazione che utilizza crittografia a chiave simmetrica non mette a disposizione che garantiscano l'integrità e l'autenticazione dell'origine dei dati. Infatti, un intrusore che riesce a conoscere la chiave segreta di due interlocutori, può poi, intercettando i messaggi che si scambiano, alterarne il contenuto, inficiandone così l'integrità. Inoltre, il destinatario non è più in grado di distinguere se i messaggi provengono dal mittente originale o dall'intrusore e quindi non ha più nessuna garanzia di autenticazione.

Gli algoritmi simmetrici più diffusi sono CAST, DES, Triple DES, RC2, RC4, RC5, IDEA.

1.3 Sistemi a crittografia a chiave pubblica

Gli algoritmi a chiave pubblica (altrimenti detti asimmetrici), sono progettati in modo tale che la chiave di cifratura risulta differente dalla chiave di decifratura. Quello che viene cifrato con la chiave di cifratura, detta chiave pubblica, può essere decifrato solo con l'altra chiave di decifratura, detta chiave privata (vedi figura 1.2). La conoscenza della chiave pubblica non permette di determinare quella privata (si tratta di un problema computazionalmente complesso). Questo è il motivo per cui la chiave di cifratura può essere resa pubblica senza compromettere la sicurezza del sistema.

$$C = E(K_{pub}, P)$$

$$P = D(K_{priv}, C)$$

P = testo in chiaro, C = testo cifrato

E = operazione di cifratura, K_{pub} = chiave pubblica del destinatario

D = operazione di decifratura, K_{priv} = chiave privata del destinatario

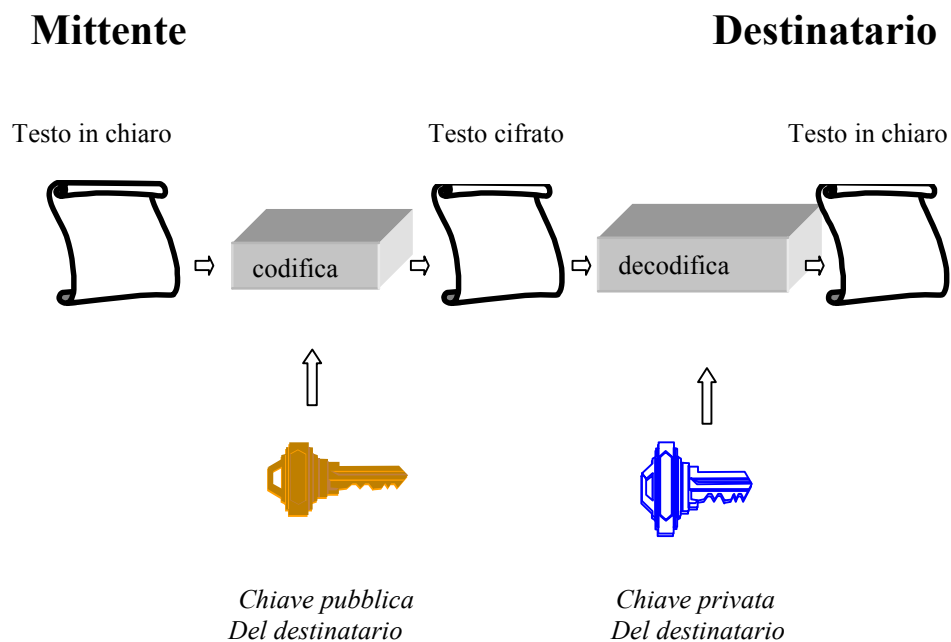


Figura 1.2: Schema di funzionamento di un algoritmo a chiave pubblica

Con tali algoritmi, a differenza di quelli simmetrici, non è più necessario prevedere un canale sicuro per la trasmissione della chiave, in quanto, essendo la chiave di decifratura distinta da quella di cifratura, è possibile distribuire quest'ultima in maniera non riservata tramite dei server pubblici. Se, poi, n sono gli utenti coinvolti, n è anche il numero di chiavi da distribuire e non $n*(n-1)/2$ come nel caso degli algoritmi simmetrici.

Gli algoritmi asimmetrici garantiscono la confidenzialità nella comunicazione. Infatti, un messaggio cifrato con la chiave pubblica del destinatario fa sì che solo quest'ultimo sia in grado di decifrare tale messaggio, in quanto è l'unico che possiede la corrispondente chiave privata.

Inoltre invertendo l'utilizzo delle chiavi, ossia cifrando con la chiave privata del mittente e decifrando con la chiave pubblica del mittente, è possibile garantire l'autenticazione. È su tale principio che si basa la firma digitale.

1.4 La firma digitale

Fino a pochissimo tempo fa la firma autografa era considerata l'unico strumento adeguato per attribuire con certezza la paternità di un messaggio: un documento originale, stampato su carta e firmato in maniera autografa, è ritenuto, infatti, legalmente valido. Tale validità legale è attribuita ritenendo che:

- sia possibile rilevare alterazioni o abrasioni della carta che ne indichino una contraffazione.
- sia possibile ricondurre con certezza (quantomeno "legale") la firma apposta in calce ad un documento alla persona la cui firma è apposta, e alla manifestazione della sua volontà di sottoscrivere, accettare o richiedere quanto esposto nel documento stesso.

La certezza dell'autenticità della firma, come è ben evidente a chiunque abbia falsificato la firma di un conoscente, è assai lontana dalla realtà. Nonostante questo, la legge tutela questa forma di manifestazione della volontà e la considera talmente valida da richiedere che certi documenti, di cui si ipotizza una eventuale futura contestazione, debbano essere conservati per lunghi periodi di tempo.

La firma digitale (*digital signature*), invece, è, oggi, la tecnologia con cui possono essere effettivamente soddisfatti tutti i requisiti richiesti per dare validità legale ad un documento elettronico firmato digitalmente; garantisce i servizi di integrità, autenticazione e non ripudio.

Vediamo quali sono le proprietà richieste al processo di apposizione di una firma digitale (proprietà che nel caso della firma autografa si assume siano garantite):

- autenticità della firma: la firma deve assicurare il destinatario del documento che il mittente ha deliberatamente sottoscritto il contenuto del documento.
- non falsificabili: la firma è la prova che solo il firmatario e nessun altro ha apposto la firma sul documento.
- non riusabilità: la firma fa parte integrante del documento non deve essere utilizzabile su un altro documento.
- non alterabilità: una volta firmato, il documento non deve poter essere alterato.
- non contestabilità: il firmatario non può rinnegare con successo la paternità dei documenti firmati; la firma attesta la volontà del firmatario di sottoscrivere quanto contenuto nel documento.

La firma digitale viene realizzata tramite tecniche crittografiche a chiave pubblica insieme all'utilizzo di particolari funzioni matematiche, chiamate funzioni hash unidirezionali. Il processo di firma digitale passa attraverso tre fasi:

1. Generazione dell'impronta digitale
2. Generazione della firma
3. Apposizione della firma

Nella prima fase viene applicata al documento in chiaro una funzione di hash appositamente studiata che produce una stringa binaria di lunghezza costante e piccola, normalmente 128 o 160 bit,

12

chiamata *digest message* o *digital fingerprint* ossia impronta digitale. Queste funzioni devono avere due proprietà:

- unidirezionalità, ossia dato x è facile calcolare $f(x)$, ma data $f(x)$ è computazionalmente difficile risalire a x .
- prive di collisioni (*collision-free*), ossia a due testi diversi deve essere computazionalmente impossibile che corrisponda la medesima impronta. ($x \neq y$ allora $f(x) \neq f(y)$)

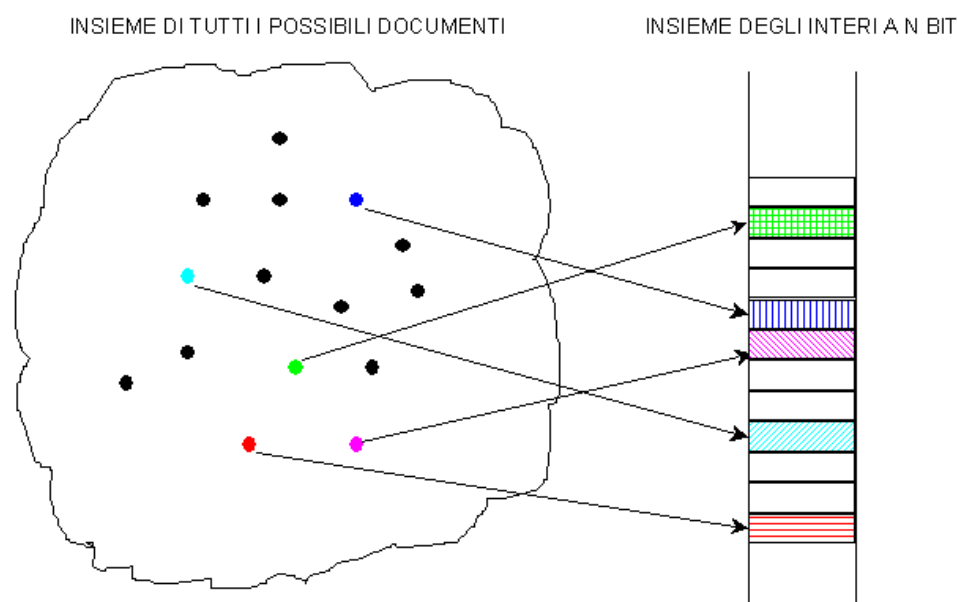


Figura 1.3: Schema di funzionamento di una funzione hash

L'utilità dell'uso delle funzioni hash consente di evitare che per la generazione della firma sia necessario applicare l'algoritmo di cifratura, che è intrinsecamente inefficiente, all'intero testo che può essere molto lungo. Inoltre consente l'autenticazione, da parte di una terza parte fidata, della sottoscrizione di un documento senza che questa venga a conoscenza del suo contenuto¹.

La seconda fase, la generazione della firma, consiste semplicemente nella cifratura con la propria chiave privata dell'impronta digitale generata in precedenza. In questo modo la

firma risulta legata, da un lato (attraverso la chiave privata usata per la generazione) al soggetto sottoscrittore, e dall'altro (per il tramite dell'impronta) al testo sottoscritto. In realtà l'operazione di cifratura viene effettuata, anziché sulla sola impronta, su una struttura di dati che la contiene insieme con altre informazioni utili, quali ad esempio l'indicazione della funzione hash usata per la sua generazione. Sebbene tali informazioni possano essere fornite separatamente rispetto alla firma, la loro inclusione nell'operazione di codifica ne garantisce l'autenticità.

Nell'ultima fase, la firma digitale generata precedentemente viene aggiunta in una posizione predefinita, normalmente alla fine, al testo del documento. Di solito, insieme con la firma vera e propria, viene allegato al documento anche il valore dell'impronta digitale ed eventualmente il certificato² da cui è possibile recuperare il valore della chiave pubblica.

La firma digitale così generata viene aggiunta in una posizione predefinita, normalmente alla fine, al testo del documento. Normalmente, insieme con la firma vera e propria, viene allegato al documento anche il valore dell'impronta digitale ed eventualmente il certificato da cui è possibile recuperare il valore della chiave pubblica.

L'operazione di verifica della firma digitale viene effettuata ricalcolando, con la medesima funzione di hash usata nella fase di firma, il valore dell'impronta digitale (hash "fresco"); poi si decifra con la chiave pubblica del firmatario la firma (hash firmato) e si controllando che il valore così ottenuto coincida con hash "fresco".

Gli algoritmi per la firma digitale più utilizzati sono RSA e DSA [S90], mentre per le funzioni hash abbiamo SHA-1, RIPEMD-160 [DBP96], MD2 [RFC1319], MD4 [RFC1320], MD5 [RFC1321].

¹ Un tipica situazione in cui si sfrutta tale situazione è nei sistemi di validazione temporale (timestamp) che vedremo nel capitolo 6.

² Vedremo nel capitolo seguente che cosa si intende esattamente.

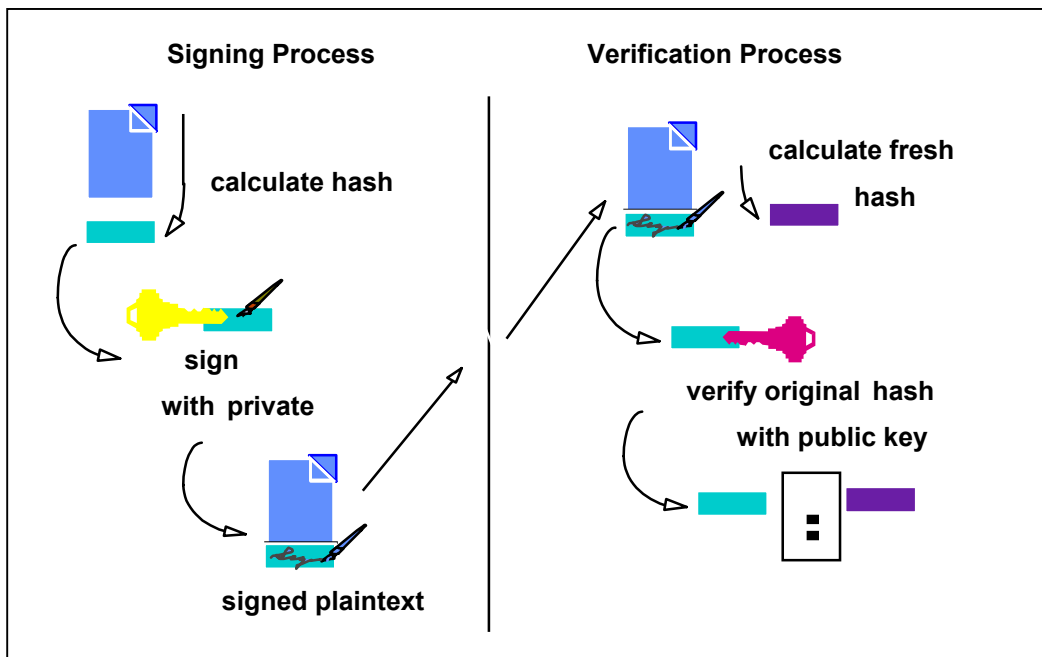


Figura 1.4: Generazione e verifica della firma

1.5 Gli algoritmi

Vediamo di seguito i due principali algoritmi che vengono utilizzati per generare e verificare la firma digitale. Sono gli algoritmi che saranno utilizzati nella realizzazione della nostra applicazione.

1.5.1 RSA

Proposto nel 1978 da Rivest, Shamir e Adleman [RSA78], da cui il nome, è il primo sistema di crittografia a chiavi pubbliche basato sui concetti proposti da Diffie ed Hellman ed è anche quello attualmente più diffuso ed utilizzato. Il metodo si basa sulla

fattorizzazione di interi di grandi dimensioni e per utilizzarlo ciascun interlocutore deve compiere le seguenti azioni:

1. Scegliere due grandi interi p e q che siano primi; il loro prodotto corrisponde al valore di N chiamato modulo, in quanto utilizzato nel calcolo del modulo nelle operazioni di codifica e decodifica.
2. Scegliere un intero e , che sia primo rispetto a $T = (p-1) * (q-1)$ detta funzione di Eulero, ossia tale che tra T ed e non ci siano fattori comuni eccetto 1. Si ottiene così la chiave pubblica di codifica $K_p = (e, N)$.
3. Calcolare l'intero d per il quale risulta $e * d \bmod T = 1$, ossia trovare l'intero d per cui $(e*d-1)$ sia divisibile per T . Si ottiene così la chiave segreta di decodifica $K_s = (d, N)$.

Il messaggio cifrato X corrispondente ad un messaggio M si ottiene dalla relazione:

$$X = M^e \bmod N$$

mentre la decodifica avviene secondo la relazione:

$$X^d \bmod N = (M^e \bmod N)^d \bmod N = M^{ed} \bmod N = M.$$

La sicurezza dello RSA è affidata alla difficoltà di determinare i fattori primi di un intero quando questo è molto grande, difficoltà che aumenta in modo esponenziale al crescere del numero di bit usati per la chiave. Infatti se un intrusore riuscisse a sapere che N è fattorizzato in p e q , allora potrebbe calcolare T e conoscendo d risalire a e tramite l'equazione $e*d \bmod T = 1$.

Va osservato che tale algoritmo può essere utilizzato sia per cifrare un documento sia per firmarlo digitalmente. Per far ciò è sufficiente invertire la sequenza di utilizzo delle chiavi: il mittente firma il documento con la propria chiave privata e poi il destinatario verifica la tale firma con la chiave pubblica del mittente. Ciò è possibile in virtù della seguente proprietà dell'aritmetica modulare:

$$(M^e \bmod N)^d \bmod N = (M^d \bmod N)^e \bmod N = M^{ed} \bmod N = M.$$

La lunghezza della chiave dipende dal grado di sicurezza richiesto; tipici valori sono 512, 1024 o 2048 bit.

Esempio:

$$p=3 \text{ e } q=11 \rightarrow N=p*q=33$$

$$T=(p-1)*(q-1)=20 \rightarrow e=7$$

$e*d \bmod T = 1 \rightarrow$ bisogna trovare un q intero tale che $T*x+1$ sia un multiplo di $e \rightarrow x=1 \rightarrow d = (T*x+1)/e = 3$

chiave privata = (7 , 33) chiave pubblica = (3 , 33)

$$M = 15 \rightarrow X = (15)^3 \bmod 33 = 9$$

$$(9)^7 \bmod 33 = 15 = M$$

1.5.2 SHA-1

La funzione hash SHA (Secure Hash Algorithm) produce un digest message di 160 bit per ogni messaggio di lunghezza inferiore a 2^{64} bit. Vediamo le principali fasi algoritmiche [S96]:

1. Il messaggio viene completato con dei bit per ottenere multipli di 512 bit (padding): all'inizio del messaggio si mette un 1 seguito da tanti zeri quanti sono necessari affinché il messaggio sia più corto di 64 bit di un multiplo di 512 bit, mentre alla fine del messaggio si mettono 64 bit che danno la lunghezza del messaggio prima del padding.
2. Vengono inizializzate 5 variabili di 32 bit l'una (A, B, C, D, E).
3. L'algoritmo processa blocchi di 512 bit alla volta; ogni blocco viene processato attraverso 4 funzioni non lineari, ognuna applicata 20 volte. Senza entrare nei dettagli implementativi di queste funzioni che fondamentalmente eseguono operazioni

logiche (and, or, not, xor) e operazioni di shifting, abbiamo come risultato finale che ogni blocco di 512 bit processato modifica le cinque variabili iniziali. Una volta processati tutti i blocchi che costituiscono il messaggio, concatenando tali variabili otteniamo un hash a $5 \times 32 = 160$ bit del messaggio.

Tuttora non esistono attacchi crittografici contro SHA-1; inoltre, producendo un hash a 160 bit, risulta più robusto ad attacchi di forza - bruta rispetto a MD4 e MD5 che producono un message digest di 128 bit.

Capitolo 2. INFRASTRUTTURE A CHIAVE PUBBLICA

2.1 Certificati elettronici

Nella tecnologia di crittografia a chiave pubblica sia in fase di cifratura sia in fase di verifica di una firma digitale occorre ritrovare la chiave pubblica o del destinatario di un messaggio o del firmatario del messaggio firmato. In entrambi i casi il valore delle chiavi pubbliche non è confidenziale; la criticità del reperimento delle chiavi sta nel garantire non la confidenzialità, ma l'autenticità delle chiavi pubbliche, ossia sta nell'assicurare che una certa chiave pubblica appartenga effettivamente all'interlocutore per cui si vuole cifrare o di cui si deve verificare la firma. Se, infatti, una terza parte prelevasse la chiave pubblica del destinatario sostituendola con la propria, il contenuto dei messaggi cifrati sarebbe disvelato e le firme digitali potrebbero essere falsificate.

La distribuzione delle chiavi pubbliche è, pertanto, il problema cruciale della tecnologia a chiave pubblica. In un dominio con un numero limitato di utenti si potrebbe anche ricorrere ad un meccanismo manuale di distribuzione delle chiavi: due interlocutori che abbiano una relazione di conoscenza già stabilita, potrebbero, ad esempio, scambiarsi reciprocamente le chiavi attraverso floppy disk. Meccanismi di distribuzione manuale diventano, tuttavia, assolutamente inadeguati ed impraticabili in dominio scalabile dove non c'è alcuna diretta conoscenza prestabilita tra gli interlocutori.

Il problema della distribuzione delle chiavi pubbliche è risolto tramite l'impiego dei certificati elettronici. I certificati a chiave pubblica costituiscono, infatti, lo strumento affidabile e

sicuro attraverso cui rispondere ad esigenze di scalabilità; attraverso i certificati elettronici le chiavi pubbliche vengono distribuite e rese note agli utenti finali con garanzia di autenticità e integrità. L'utilizzo dei certificati elettronici presuppone, tuttavia, l'esistenza di una Autorità di Certificazione (Certification Authority o CA) che li emetta e li gestisca.

Ogni certificato è una struttura dati costituita da una parte dati contenente al minimo:

- informazioni che identificano univocamente il possessore di una chiave pubblica (ad esempio il nome);
- il valore della chiave pubblica;
- il periodo di validità temporale del certificato;
- la firma digitale della autorità di certificazione con cui si assicura autenticità della chiave ed integrità delle informazioni contenute nel certificato;

La semplicità del meccanismo di distribuzione delle chiavi è diretta conseguenza delle caratteristiche stesse dei certificati: i certificati, infatti, possono essere distribuiti senza dover necessariamente ricorrere ai tipici servizi di sicurezza di confidenzialità, integrità, e autenticazione delle comunicazioni.

Per le proprietà della crittografia a chiave pubblica non c'è infatti alcun bisogno di garantire la riservatezza del valore della chiave pubblica; durante il processo di distribuzione, poi, non ci sono requisiti di autenticazione ed integrità dal momento che il certificato è per sua costituzione una struttura già protetta (la firma digitale dell'autorità di certificazione sul certificato fornisce, infatti, sia autenticazione sia integrità).

Se, quindi, un intrusore tentasse, durante la pubblicazione del certificato, di alterarne il contenuto, la manomissione sarebbe immediatamente rilevata in fase di verifica della firma sul certificato; il processo di verifica fallirebbe e l'utente finale sarebbe

avvertito della non integrità della chiave pubblica contenuta nel certificato.

Le caratteristiche stesse, quindi, del certificato permettono di distribuire i certificati a chiave pubblica anche mediante canali non sicuri (file server insicuri o sistemi di directory o protocolli di comunicazione intrinsecamente insicuri.)

2.2 Lo standard X.509 per i certificati

Lo standard ormai diffusamente riconosciuto di definizione del formato dei certificati è quello descritto nello standard X.509 ISO/IEC/ITU [RFC2459] (Visa e MasterCard hanno ad esempio adottato le specifiche X.509 come base per la definizione dello standard per il commercio elettronico SET, Secure Electronic Transaction).

Tale standard è giunto alla terza versione (X.509v3), dopo che le precedenti due versioni si erano dimostrate insufficienti a risolvere certe problematiche. Vediamo da quali campi è composto un certificato [CUR96]:

- **Version:** indica la versione del formato del certificato (1, 2 o 3)
- **Serial number:** è un codice numerico che identifica il certificato tra tutti i certificati emessi dall'Autorità di Certificazione
- **Signature Algorithm:** specifica l'algoritmo utilizzato dalla CA per firmare il certificato; è data dalla coppia funzione hash – algoritmo a chiave pubblica
- **Issuer X.500 Name:** è il nome della CA; è fornito secondo lo standard di naming X.500
- **Validity Period:** specifica la data e l'ora di inizio validità e di fine validità del certificato
- **Subject X.500 Name:** nome del possessore del certificato

- **Subject Public Key Information:** contiene il valore della chiave pubblica del possessore del certificato e l'algoritmo con cui tale chiave viene usata
- **Issuer Unique Identifier:** è una stringa di bit aggiuntivi, opzionale, usata nel caso in cui nella struttura ad albero ci siano due CA
- **Subject Unique Identifier:** è una stringa di bit aggiuntivi, opzionale, usata nel caso di omonimia di due membri in una stessa CA
- **CA Signature:** è la firma digitale della CA sul certificato

Nella versione 3, rispetto alle precedenti, è stato aggiunto un ulteriore campo (**standard extension**); questo campo è suddiviso in tre sottoinsiemi: l'identificatore del tipo di estensione, un indicatore di criticità e il valore effettivo dell'estensione; l'indicatore di criticità facilita l'interoperabilità tra sistemi che non utilizzano certificati con determinate estensioni e sistemi che, invece, interpretano tutte le estensioni definite a livello di standard; l'indicatore di criticità è semplicemente un flag che stabilisce se l'estensione è critica o non critica; se l'estensione non è critica, significa che il sistema che deve elaborare il certificato può eventualmente ignorare l'estensione in questione se non è in grado di interpretarla.

Le estensioni standard definite nei certificati X.509v3 si suddividono in quattro gruppi:

- estensioni contenenti informazioni sulla chiave pubblica che specificano il tipo di utilizzo per cui è stata emessa una certa chiave.
- estensioni contenenti informazioni aggiuntive relative all'Autorità di Certificazione e all'utente possessore del certificato (ad esempio nomi alternativi per la CA e per l'utente).

- estensioni contenenti informazioni sulle politiche di emissione e sulle finalità di utilizzo dei certificati (le estensioni “*certificate policy*”³ e “*policy mapping*”⁴).
- estensioni contenenti informazioni sui vincoli di spazio dei nomi o di politica da imporre durante il ritrovamento o la verifica di un certificato appartenente ad un dominio di fiducia esterno.

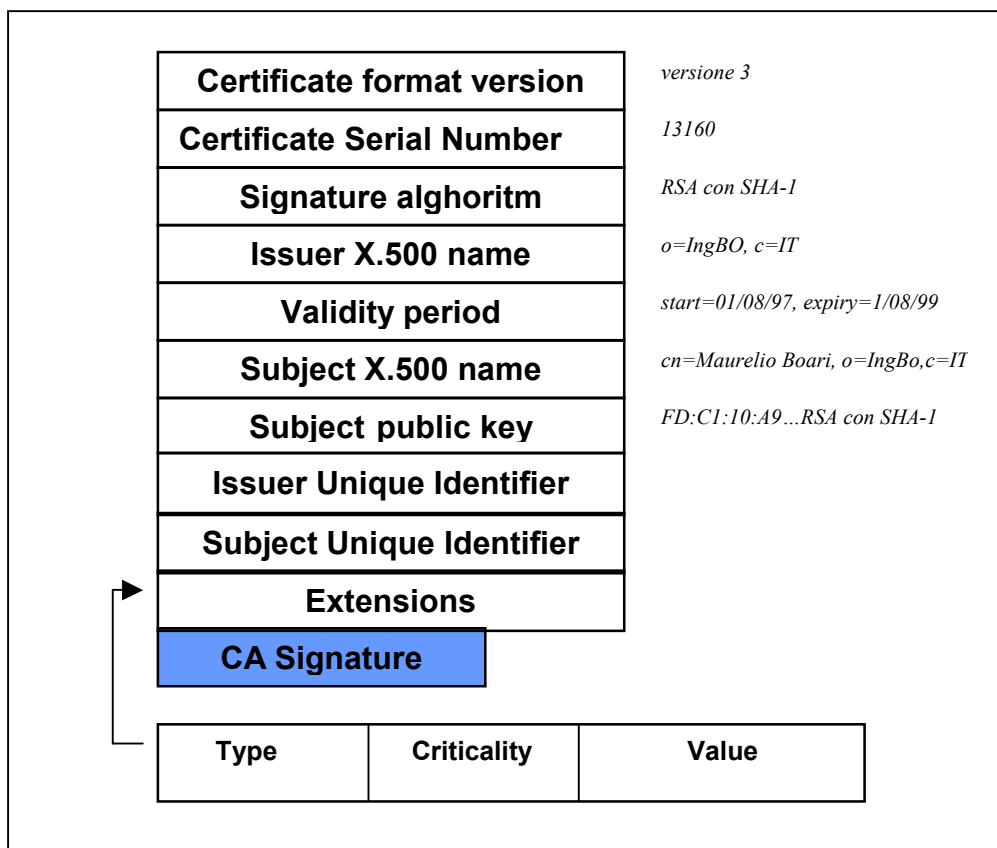


Figura 2.1: Esempio di certificato X.509v3

³ Specificano la politica sotto cui un certificato è stato emesso

⁴ Fornisce il meccanismo con cui controllare la compatibilità tra le politiche differenti di autorità di certificazione quando un'autorità di certificazione crea una relazione di fiducia con un'altra autorità.

2.3 La necessità di coppie di chiavi distinte per cifratura e firma

L'algoritmo a chiave pubblica più utilizzato, RSA, ha la proprietà di permettere sia operazioni di cifratura sia operazioni di firma digitale con la stessa coppia di chiavi crittografiche; tuttavia un uso doppio della coppia di chiavi è sconsigliato. I motivi sono deducibili dall'esame dei requisiti di gestione che sono differenti nel caso in cui le chiavi siano utilizzate per la cifratura o per la firma digitale.

Per la coppia di chiavi di firma digitale si devono, infatti, rispettare i seguenti requisiti [F94]:

- la chiave privata di firma deve essere custodita durante l'intero periodo di validità in modo tale che solo il suo legittimo possessore possa averne accesso; tale requisito è un presupposto fondamentale per fornire supporto a servizi di non ripudio (vedremo nel terzo capitolo come la legge italiana imponga che la chiave di firma sia custodita all'interno dello stesso dispositivo in cui sarà utilizzata.)
- una chiave privata di firma non deve essere sottoposta a backup per proteggersi contro perdite accidentali della chiave privata; se la chiave di firma, infatti, viene accidentalmente persa, una nuova coppia di chiavi di firma può essere facilmente generata, mentre sottoporre a backup la chiave privata di firma contrasterebbe con il primo requisito.
- una chiave privata di firma non necessita di essere archiviata; l'archiviazione contrasterebbe con il primo requisito; una chiave privata di firma deve essere distrutta in modo sicuro allo scadere della sua validità; se, infatti, si potesse risalire al suo valore anche dopo la cessazione del suo utilizzo, tale chiave privata potrebbe ugualmente essere utilizzata per falsificare la firma su

vecchi documenti (per evitare ciò, si può ricorrere a servizi di validazione temporale.)

- una chiave pubblica di firma necessita di essere archiviata; tale chiave potrebbe essere utilizzata per la verifica di firme apposte su documenti in tempi successivi alla scadenza della validità della coppia di chiavi di firma.

Per la coppia di chiavi di cifratura, invece, si devono rispettare i seguenti requisiti, in contrasto con i precedenti:

- la chiave privata, utilizzata nelle operazioni di decifratura, necessita di essere sottoposta a backup o archiviata; infatti, se una chiave privata venisse persa (ad esempio a causa della corruzione del dispositivo di protezione della chiave o per dimenticanza da parte del legittimo possessore del segreto utilizzato per sbloccare l'accesso al dispositivo di protezione) e non ci fossero meccanismi di recupero della chiave (*key recovery*), sarebbe inaccettabile perdere irrimediabilmente l'accesso ai dati cifrati con quella chiave.
- la chiave pubblica utilizzata per la cifratura dei dati non necessita generalmente di archiviazione; l'algoritmo utilizzato determina se un backup della chiave pubblica è necessario o meno (se si utilizza un meccanismo di trasporto delle chiavi basato su RSA, il backup non è necessario, può diventarlo se si utilizza, invece, l'algoritmo Diffie-Hellmann per lo scambio delle chiavi).
- la chiave privata non deve essere distrutta allo scadere della sua validità di impiego per poter continuare a decifrare in futuro i documenti cifrati col la corrispondente chiave pubblica.

Alla luce delle considerazioni elencate è evidente, quindi, che, se si impiegasse un'unica coppia di chiavi sia per la cifratura

sia per la firma, sarebbe impossibile garantire il soddisfacimento contemporaneo di tutti i requisiti.

Inoltre si tenga presente che:

- le realizzazioni a chiave pubblica utilizzate a supporto della cifratura sono tipicamente soggette a limiti di esportazioni molto più restrittivi rispetto a quelli previsti per la firma digitale
- le chiavi di firma e di cifratura possono avere validità temporali differenti per motivi di politica organizzativi/gestionali e di sicurezza.
- non tutti gli algoritmi a chiave pubblica hanno la proprietà dell'algoritmo RSA di effettuare sia cifratura sia firma con la stessa coppia di chiavi.

2.4 Infrastruttura a chiave pubblica (PKI)

Le infrastrutture a chiave pubblica (Public Key Infrastructure) forniscono il supporto necessario affinché la tecnologia di crittografia a chiave pubblica sia utilizzabile su larga scala. Le infrastrutture offrono servizi relativi alla gestione delle chiavi e dei certificati e delle politiche di sicurezza. Le autorità di certificazione e la problematica di gestione dei certificati elettronici costituiscono, infatti, il cuore delle infrastrutture a chiave pubblica.

Una infrastruttura a chiave pubblica introduce il concetto di *third-party trust*, ossia di quella situazione che si verifica quando due generiche entità si fidano implicitamente l'una dell'altra senza che abbiano precedentemente stabilito una personale relazione di fiducia. Questo è possibile perché entrambe le entità condividono una relazione di fiducia con una terza parte comune.

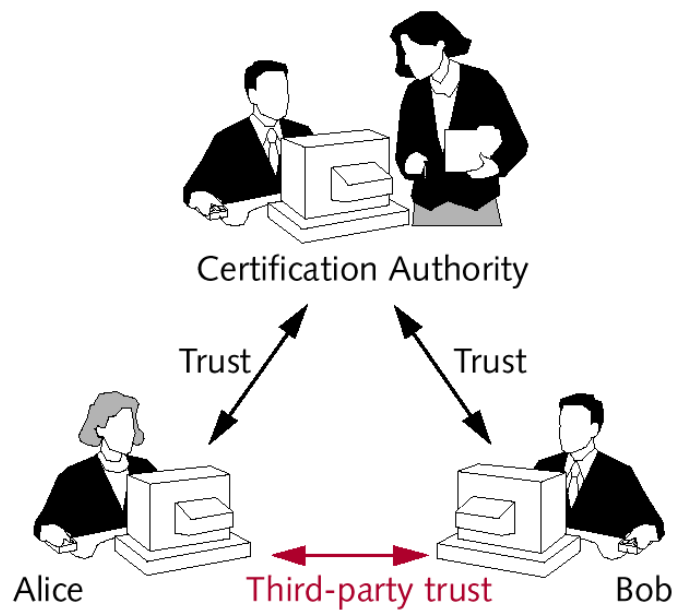


Figura 2.2: *Third-party trust*

Third-party trust è un requisito fondamentale per qualsiasi implementazione su larga scala che utilizzi crittografia a chiave pubblica e in una PKI viene realizzata attraverso l'Autorità di Certificazione [CUR95].

Vediamo in dettaglio quali sono i componenti fondamentali di una PKI sono [L97] (vedi figura 2.3):

Autorità di Registrazione (RA)

L'accertamento dell'identità dell'utente richiedente un certificato elettronico, deve precedere l'effettiva emissione del certificato; è indispensabile procedere a tale verifica dato che con l'emissione di un certificato elettronico si rende pubblicamente valida l'associazione tra una certa chiave pubblica e una certa entità. Una volta attestata la validità dell'identità dell'utente attraverso una serie di procedure definite nell'ambito di una precisa politica di sicurezza (ad esempio, il controllo della carta di

identità), l'autorità di registrazione ha il compito di abilitare l'utente come appartenente ad uno specifico dominio di fiducia ; la funzionalità di autorità di registrazione può essere espletata dall'autorità di certificazione stessa oppure delegata ad altre entità.

Autorità di Certificazione (CA)

Costituisce il cuore di una PKI; la sua principale funzione consiste nel creare i certificati elettronici per quegli utenti precedentemente abilitati nella fase di registrazione al dominio di fiducia di cui la CA è garante; un'Autorità di Certificazione non si deve limitare esclusivamente alla generazione dei certificati, ma deve poterne gestire l'intero ciclo di vita. Il ciclo di vita comprende le fasi di generazione, aggiornamento (nel caso in cui il certificato stia per perdere validità temporale), sostituzione (nel caso di scadenza della validità temporale) e revoca nel caso in cui le condizioni di emissione del certificato non siano più valide. Un ulteriore compito della CA è stabilire relazioni di fiducia con altre CA.

Sistema di Directory

Contiene i certificati a chiave pubblica, reperibili dagli utenti qualora sia necessario, e le liste dei certificati revocati (CRL). Tale sistema costituisce un elemento fondamentale per la distribuzione su larga scala delle chiavi pubbliche utilizzate nella cifratura e nella firma dei dati. Il sistema di Directory si basa su un particolare protocollo di comunicazione LDAP (Lightweight Directory Access Protocol) [RFC1777] che utilizza il TCP/IP.

PKI Database

Oltre alla Directory, di solito, c'è un'altra entità dedicata alla memorizzazione delle chiavi: è il database gestito esclusivamente dalla CA nel quale viene fatto un backup delle

chiavi e vengono archiviate le chiavi scadute. Questo database, a differenza della Directory, è privato ed è accessibile solo dalla CA.

Utenti finali

Gli utenti finali (*end-entity*) sono dotati del software in grado di interagire con la PKI in tutte le fasi in cui sia richiesta un'interazione tra le applicazioni client e la CA o la directory (ad esempio un'interazione fondamentale interviene nella fase di inizializzazione dell'utente, fase nella quale vengono creati i relativi certificati di cifratura e di firma).

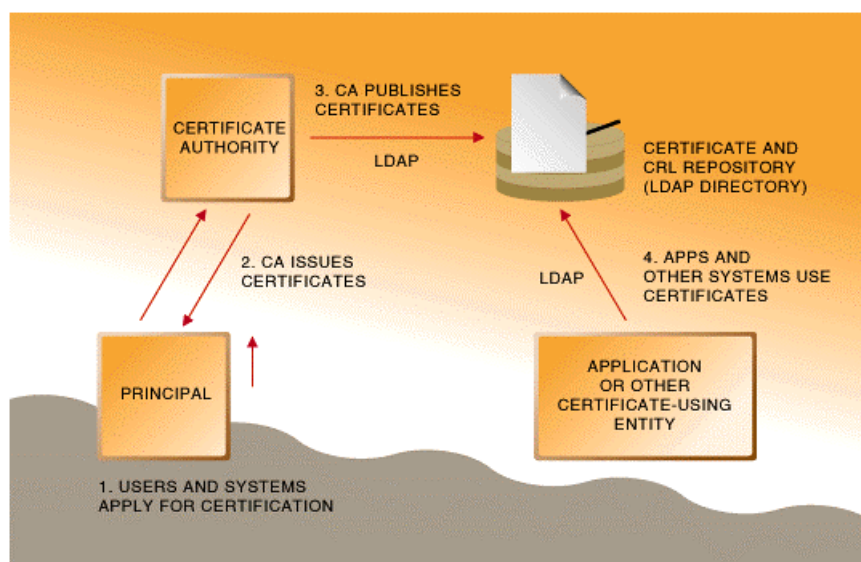


Figura 2.3: Architettura di una PKI

2.5 Requisiti di una PKI

L'implementazione di una infrastruttura a chiave pubblica deve tener conto di una serie di requisiti progettuali che si possono sintetizzare nei seguenti:

- scalabilità.

- supporto per applicazioni multiple: a beneficio degli utenti finali in termini di convenienza, sicurezza ed economia, una stessa infrastruttura deve garantire il supporto per molteplici applicazioni (posta elettronica sicura, applicazioni Web, trasferimento di file sicuro); pertanto, il modello di gestione della sicurezza deve essere consistente e uniforme per tutte le applicazioni.
- interoperabilità tra infrastrutture differenti: non è certo una soluzione praticabile quella di realizzare un'unica infrastruttura rispondente alle necessità di sicurezza di un dominio di utenti su scala globale; è, quindi, evidente che si debba ricorrere a domini di sicurezza distinti, ognuno amministrato da un'infrastruttura specifica. Tuttavia, l'interoperabilità di tali infrastrutture distinte deve essere assicurata ed è richiesta per garantire il raggiungimento di un buon livello di scalabilità.
- supporto per una molteplicità di politiche: cammini di certificazione⁵ considerati appropriati per un'applicazione, possono non essere considerati altrettanto validi per un'altra applicazione; è, quindi, necessario implementare meccanismi che permettano da un lato di attribuire politiche differenti ai vari cammini di certificazione dall'altro di associare ad ogni applicazione una politica di sicurezza specifica.
- conformità agli standard: una vera interoperabilità tra PKI distinte è ottenibile soltanto con l'adozione di standard che definiscono i protocolli funzionali e di comunicazione relativi ai componenti costitutivi di una infrastruttura a chiave pubblica.

⁵ I cammini di certificazione sono i cammini attraverso cui è possibile verificare l'integrità dei certificati elettronici degli attori di una comunicazione sicura appartenenti a domini di sicurezza distinti; verranno presi in considerazione a fine di questo capitolo.

2.6 Ciclo di vita dei certificati

L'emissione effettiva di un certificato elettronico da parte di un'Autorità di Certificazione a favore di un utente finale deve essere preceduta da una fase di registrazione dell'utente richiedente il certificato elettronico. Attraverso il processo di registrazione l'utente richiedente un servizio di certificazione si identifica presso l'autorità preposta al servizio di registrazione; le credenziali che in questa fase l'utente deve sottoporre all'Autorità di Registrazione dipendono fortemente dalle modalità e procedure di registrazione definite nell'ambito di una politica di sicurezza.

Il processo di registrazione stabilisce una relazione iniziale tra utente finale ed Autorità di Certificazione; l'utente finale, una volta attestata l'autenticità della sua identità, viene registrato nel dominio di fiducia gestito dalla CA. L'obiettivo, di primaria importanza del processo di registrazione è, quindi, quello di garantire che la chiave pubblica, di cui un certo utente finale richiede la certificazione, appartenga effettivamente al nome del richiedente.

Terminata la fase di registrazione, l'utente può richiedere l'emissione di un certificato elettronico. La procedura di generazione di un certificato elettronico consiste dei seguenti passi:

- l'utente finale sottopone all'autorità di certificazione le informazioni da certificare.
- l'Autorità di Certificazione può verificare l'accuratezza delle informazioni presentate in accordo a politiche e standard applicabili.
- l'Autorità di Certificazione firma le informazioni generando il certificato e lo pubblica sul sistema di Directory; di solito, la CA archivia una copia del certificato sul suo database privato; ogni operazione di generazione di certificati elettronici viene registrata su un archivio di registrazione dati.

Ogni certificato elettronico generato ha una validità temporale limitata al cui termine va sostituito; il periodo di validità di un certificato, in assenza di compromissioni o usi illeciti, garantisce l'utente che deve utilizzare tale certificato che la chiave pubblica possa essere utilizzata per lo scopo per cui è stata generata e che l'associazione tra la chiave pubblica e le altre informazioni contenute nel certificato sia ancora valida.

2.7 La gestione delle chiavi

Alla gestione del ciclo di vita dei certificati è strettamente collegata la gestione delle coppie di chiavi (*key management*); tale gestione comprende le funzioni di generazione, di backup, recupero ed aggiornamento delle chiavi.

In generale, le chiavi sono generate dagli stessi utenti finali che ne hanno bisogno (*key generation*); successivamente, gli utenti possono chiedere alla CA la certificazione di tali chiavi. La generazione e la successiva certificazione costituiscono la fase di inizializzazione dell'utente.

La necessità di mantenere copie delle chiavi (*key backup*⁶) nasce dal fatto che l'eventuale perdita delle chiavi comporta l'impossibilità di decifrare i messaggi cifrati e di verificare i messaggi firmati. Spesso le organizzazioni richiedono un backup delle chiavi dei loro membri per evitare che l'onere di mantenimento delle copie sia esclusivamente di questi ultimi. Tali organizzazioni possono avere la necessità di verificare documenti digitalmente firmati da loro membri anche oltre la validità

⁶ Per applicazioni più "delicate", sono richieste per motivi legali procedure di backup più complesse. Si parla in tal caso di *key escrow*.

temporale di tali chiavi e non possono fidarsi unicamente alle eventuali copie di backup fatte dal possessore della chiave⁷.

Inoltre c'è sempre la possibilità che un utente si dimentichi la password di protezione delle sue chiavi; pertanto deve essere possibile recuperare le proprie chiavi e di impostare una nuova password (*key recovery*).

Un altro problema nella gestione delle chiavi è l'aggiornamento (*key update*). Spesso le organizzazioni impongono ai loro membri un aggiornamento a intervalli regolari delle loro chiavi. Questo può comportare una gran mole di lavoro, in particolar modo per organizzazione con un gran numero di partecipanti. Infatti, l'aggiornamento delle chiavi comporta una riemissione del certificato elettronico corrispondente. A questo si aggiunga il fatto che, non raramente, capita che le chiavi dell'intera organizzazione scadono lo stesso giorno.

2.8 La revoca di un certificato

L'operazione di revoca di un certificato costituisce una fase della gestione del ciclo di vita dei certificati di alta criticità. Il certificato elettronico deve essere revocato in presenza delle seguenti condizioni:

- compromissione rilevata o semplicemente sospettata della chiave privata corrispondente alla chiave pubblica contenuta nel certificato.
- cambiamento di una qualsiasi delle informazioni contenute nel certificato elettronico o delle condizioni iniziali di registrazione.

⁷ Come osservato in precedenza, normalmente un utente dispone di due copie di chiavi: una per la firma, una per la cifratura. In tal caso, sono normalmente oggetto di backup solo le chiavi di cifratura.

La revoca del certificato elettronico è effettuata dall'Autorità di Certificazione e generalmente viene avviata su richiesta dello stesso utente finale; in questo caso, data la criticità e le implicazioni dell'operazione di revoca, è indispensabile predisporre, un sistema di autenticazione della richiesta di revoca.

La difficoltà legata al processo di revoca sta nel garantire una corretta notifica su larga scala dell'avvenuta revoca di un particolare certificato; ogniqualvolta un utente finale intende utilizzare un certificato o per cifrare dati o per verificare una firma, deve essere informato sullo stato di quel certificato.

Il meccanismo più comunemente utilizzato per la notifica su larga scala di avvenute revoche fa uso delle cosiddette liste di revoca dei certificati (Certificate Revocation List o CRL); la gestione di tali liste è delegata alla CA; nell'ambito del dominio amministrato, ogni CA pubblica periodicamente una struttura dati contenente l'elenco dei certificati revocati, ossia una lista, firmata digitalmente dall'Autorità di Certificazione, che riporta i certificati revocati, la data temporale in cui è avvenuta la revoca ed eventualmente il motivo della revoca; i motivi per cui anche la CRL deve essere firmata digitalmente sono analoghi a quelli descritti nel caso dei certificati elettronici (si veda figura 2.4).

La CRL, al pari dei certificati, deve essere pubblicata in modo che sia consultabile da qualunque utente. La frequenza di pubblicazione di una CRL dipende fortemente dalla politica di sicurezza definita all'interno dell'organizzazione; una determinata politica potrebbe richiedere la pubblicazione di una nuova CRL ogniqualvolta si richiede una revoca; tale procedura risulterebbe, tuttavia, molto dispendiosa dal punto di vista amministrativo. Generalmente, la CA rilascia liste di revoca dei certificati su base periodica, con intervalli di periodicità definibili a livello di politica di sicurezza.

Issuer Name
CRL Issue Time/Date
Certificate serial Number
Revocation Time/Date
Certificate serial Number
Revocation Time/Date
Certificate serial Number
Revocation Time/Date
CA Signature

Figura 2.4: Formato delle CRL secondo lo standard X.509

Le liste di revoca giocano un ruolo di fondamentale importanza in tutti i processi crittografici, in quanto devono essere consultate sia in fase di cifratura sia in fase di verifica di una firma. Infatti, prima di cifrare un messaggio, va verificato che la chiave pubblica che si deve usare per la cifratura appartenga ad un certificato valido. Analogo discorso vale per la verifica della firma, poiché una firma può essere ritenuta valida solo se il certificato del firmatario, oltre ad essere autentico e integro, non è stato revocato.

La dimensione di una CRL è di fondamentale importanza nelle prestazioni di una PKI. Per ovviare ad un possibile overhead dovuto alle eccessive dimensione della lista di revoca sono stati introdotti i cosiddetti punti di distribuzione delle CRL che consentono di partizionare in modo arbitrario l'intera CRL: ogni partizione è associata ad un punto di distribuzione. In questo modo la massima dimensione che può raggiungere una singola partizione è controllata dall'Autorità di Certificazione.

Il metodo di distribuzione della notifica di revoca attraverso CRL distribuite periodicamente dalla CA è detto di tipo *pull*; il nome del metodo deriva dal fatto che, quando necessario, sono i sistemi degli utenti finali a reperire le CRL da un sistema di distribuzione.

Il problema, tuttavia, legato al metodo di distribuzione delle notifiche delle revoche tramite CRL, è quello della latenza introdotta. Esistono transazioni in cui è necessario garantire revoca in tempo reale e che non possono tollerare finestre temporali in cui le CRL non sono perfettamente allineate; il metodo sopra descritto presenta proprio la limitazione che la latenza introdotte nelle notifiche delle revoche dipende dalla periodicità di emissione delle CRL.

Una possibile soluzione potrebbe essere l'emissione forzata di una CRL fuori dalla sequenza degli intervalli temporali prestabilita da parte della CA; ma se viene avviato un attacco attivo finalizzato ad impedire la pubblicazione di tale lista di revoca fuori sequenza, non c'è un meccanismo affidabile di distribuzione delle CRL che permetta di rilevare con assoluta certezza la mancata pubblicazione di tale CRL.

Esistono metodi alternativi di distribuzione delle CRL mirati a risolvere il problema della revoca in tempo reale. Un primo approccio è quello che utilizza un metodo di tipo *push* in cui è l'Autorità di Certificazione ad inviare in modalità "broadcast" le CRL non appena sopravviene una nuova revoca. Se, da un lato, questo metodo garantisce tempi di distribuzione delle CRL estremamente veloci, tuttavia, dall'altro lato presenta alcuni svantaggi che lo rendono non diffusamente impiegato. Innanzitutto tale metodo richiede una modalità di distribuzione sicura che garantisca che le CRL effettivamente raggiungano i sistemi degli utenti finali previsti; in secondo luogo, esso può dar luogo ad un incremento considerevole di traffico sulla rete; in terzo luogo tale

metodo non presenta né uno standard, né una proposta di definizione che ne permetta un'implementazione diffusa.

Un altro metodo di notifica delle revoche, alternativo a quelli precedenti, consiste nell'effettuare una transazione *online* di verifica dello stato del certificato (tale protocollo, definito a livello di Internet draft [MAM99], è conosciuto con il nome di OCSP, "Online Certificate Status Protocol"); il sistema PKI mette a disposizione un server online dedicato a questo servizio che garantisca disponibilità ed accessibilità continua del servizio.

2.9 I cammini di certificazione

Se si potesse disporre di un'unica Autorità di Certificazione su scala globale, il problema della distribuzione, del reperimento e della verifica della validità delle chiavi pubbliche non sussisterebbe; tuttavia una tale soluzione non è praticabile per motivi di scalabilità, flessibilità e sicurezza. Diventa, quindi, inevitabile ricorrere ad un modello costituito da Autorità di Certificazione multiple tra loro concatenate secondo differenti modelli organizzativi, detti anche modelli di fiducia.

In uno scenario costituito da una molteplicità di Autorità di Certificazione su larga scala, strutturate secondo un certo modello organizzativo, non è pensabile che ogni utente abbia diretta conoscenza delle chiavi pubbliche di ogni potenziale interlocutore, sotto forma di certificato elettronico, o delle chiavi pubbliche delle corrispondenti autorità di certificazione competenti. Occorre, quindi, disporre di un meccanismo corretto di ritrovamento dei certificati elettronici degli interlocutori appartenenti a domini di sicurezza esterni. Il modello generale su cui si basano tutti i sistemi di distribuzione su larga scala delle chiavi pubbliche sotto forma di

certificati elettronici, utilizza le cosiddette catene di certificazione, altrimenti conosciute come cammini di certificazione.

Il problema del ritrovamento di un cammino di certificazione consiste sostanzialmente nel trovare, se esiste, un cammino di certificazione che permetta di verificare l'autenticità del certificato elettronico di uno specifico utente remoto a partire da un insieme di chiavi pubbliche, assunte come radici del cammino, di cui si ha diretta e sicura conoscenza; la risoluzione del problema, quindi, assume per date certe condizioni iniziali: tali condizioni iniziali si identificano nelle chiavi di specifiche autorità di certificazione.

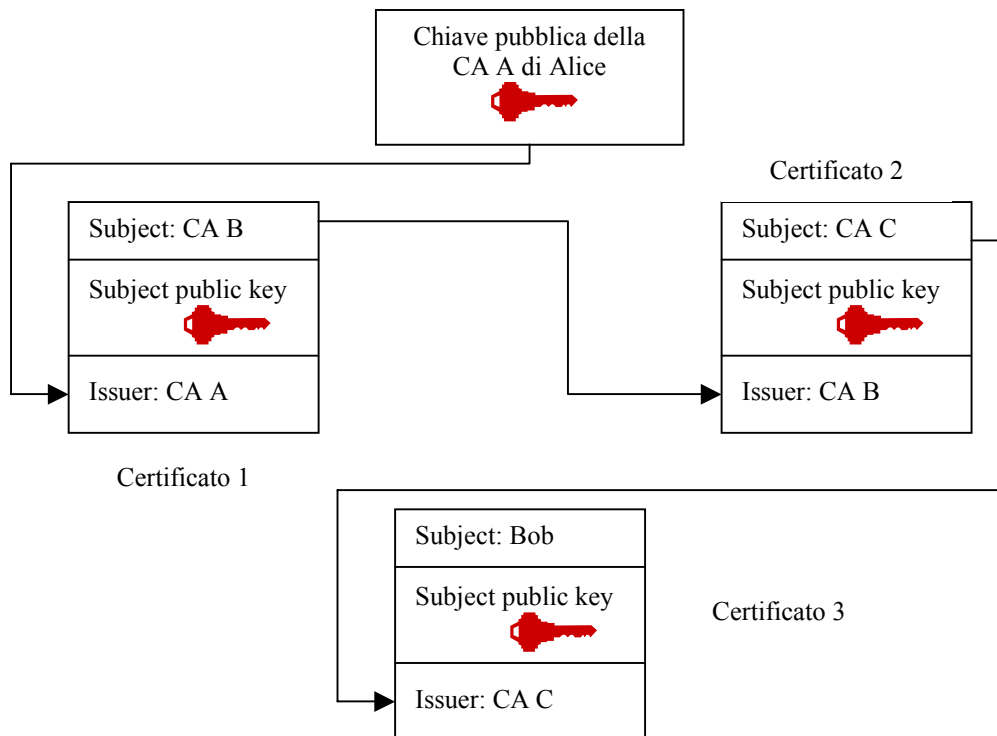


Figura 2.5: Esempio di catena di certificati

In figura 2.5 viene riportato un esempio di catena di certificati; la presenza del certificato 1 garantisce Alice

dell'autenticità della chiave pubblica dell'autorità di certificazione B; Alice può, quindi, utilizzare la chiave pubblica di B per verificare l'autenticità del certificato 2; la presenza del certificato 2 garantisce ora Alice dell'autenticità della chiave pubblica dell'autorità di certificazione C; ora Alice può utilizzare la chiave pubblica dell'autorità di certificazione C per verificare l'autenticità della chiave pubblica di Bob; se fosse mancato il certificato 2, Alice non avrebbe potuto verificare l'autenticità del certificato elettronico di Bob e conseguentemente avviare con Bob una comunicazione sicura.

Naturalmente i cammini di certificazione dipendono dal modello di fiducia, ossia dal modo in cui le diverse CA sono concatenate. Un tipico modello è quello a struttura gerarchica in cui le CA sono collegate tra loro in una struttura ad albero in cui esiste una CA radice (*CA root*) che crea canali affidabili tra le varie CA subordinate. Questi canali permettono agli utenti che non sono certificati dalla stessa CA di trovare una CA comune, in modo da stabilire una relazione di fiducia. Nella seguente figura è mostrato un modello gerarchico in cui la CA A è la *CA root* mentre le B e C sono le CA subordinate. Membri delle CA B e CA C possono fidarsi gli uni degli altri grazie alla CA A. (Si parla di *cross-certification*).

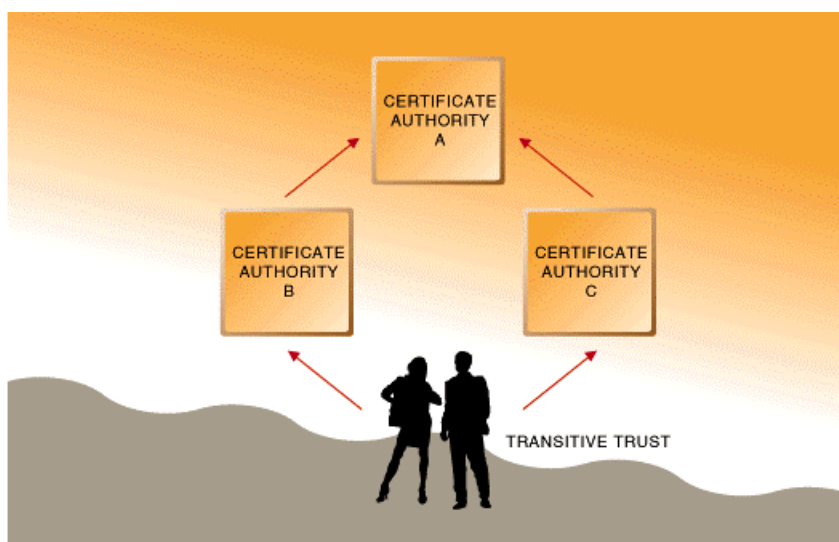


Figura 2.6 : Gerarchia tra CA.

2.10 Prodotti commerciali per le PKI

A conclusione di questo capitolo sulle infrastrutture a chiave pubbliche facciamo una breve panoramica dei principali prodotti commerciali che sono attualmente disponibili sul mercato per la costruzione e la gestione di una PKI.

Entrust (<http://www.entrust.com>)

Sviluppato dall'azienda canadese Entrust Technologies Limited, si presenta come leader dei tool di sviluppo per PKI, fornendo alcuni particolari servizi che ne esaltano la completezza del prodotto (ad esempio, timestamping, la gestione della doppia coppia di chiavi, una per la cifratura, una per la firma, l'aggiornamento automatico delle chiavi, il key recovery). Consente l'integrazione con smart card tramite il protocollo PKCS#11 della RSA e mette a disposizione numerosi plug-in per prodotti commerciali quali Netscape e Microsoft Office, Eudora,

Exchange,. Ha il protocollo LDAP integrato per la gestione del registro dei certificati e una buona gestione delle CRL.

iD2 (<http://www.id2.se>)

È prodotto dalla società svedese iD2 Technologies che realizza sistemi PKI, principalmente pensati per ambienti nei quali l'utilizzo delle smart card è fondamentale; è in grado di supportare i due principali standard attuali per le smart card, PC/SC e PKCS#11.

UniCERT - Baltimore (<http://www.baltimore.com>)

Baltimore è un'azienda irlandese che produce un sistema PKI simile per completezza delle funzionalità offerte a Entrust ma ancora sprovvisto del servizio di timestamping. Supporta attualmente solo il PKCS#11 per le smart card ma non ancora il PC/SC.

Altri prodotti per PKI sono:

- **GTE/CyberTrust** (<http://www.cybertrust.com>)
- **Verisign** (<http://www.verisign.it>)
- **Xcert** (<http://www.xcert.com>)
- **SDP - Entegrity Solution** (<http://www.entegrity.com>)
- **Secude** (<http://www.secude.com>)

Capitolo 3. IL DOCUMENTO INFORMATICO IN ITALIA

3.1 La “legge Bassanini”

Da poco più di due anni, con la legge 15 marzo 1997, n.59 (articolo 15, comma 2), ribattezzata "legge Bassanini" dal nome del ministro che l'ha presentata, l'Italia ha introdotto per la prima volta il concetto di documento informatico. Tale legge prevede un apposito regolamento attuativo che è stato emanato con il Decreto del Presidente della Repubblica n.513 del 10 novembre 1997: "Regolamento recante criteri e modalità per la formazione, l'archiviazione e la trasmissione di documenti con strumenti informatici e telematici.". L'articolo 2 del DPR 513/97 contiene una vera e propria "rivoluzione informatica" poiché, per la prima volta, si dà validità legale ai documenti informatici e alle transazioni basate su essi, equiparandoli, di fatto, ai tradizionali documenti cartacei:

Art. 2. - Il documento informatico da chiunque formato, l'archiviazione su supporto informatico e la trasmissione con strumenti telematici, sono validi e rilevanti a tutti gli effetti di legge se conformi alle disposizioni del presente regolamento.

Nell'articolo successivo si prevede la stesura di un allegato tecnico per definire gli standard da utilizzare per la firma digitale dei documenti informatici; tale allegato tecnico è stato redatto dall'AIPA (Autorità per l'Informatica nella Pubblica Amministrazione) ed è stato presentato a fine del 1998 ed è stato convertito in Decreto del Presidente del Consiglio dei Ministri in

data 8 febbraio del 1999. Da tale data il documento informatico ha pari diritto legislativo di quello cartaceo.

Nel primo articolo del DPR 513/97 vengono fornite la definizioni di documento informatico (*rappresentazione informatica di atti, fatti o dati giuridicamente rilevanti*) e di firma digitale (*risultato della procedura informatica basata su un sistema di chiavi asimmetriche a coppia, una pubblica e una privata, che consente al sottoscrittore tramite la chiave privata e al destinatario tramite la chiave pubblica, rispettivamente, di rendere manifesta e di verificare la provenienza e l'integrità di un documento informatico o di un insieme di documenti informatici*). Come si può vedere, AIPA basa tutto il suo allegato tecnico su un sistema di chiavi asimmetriche, ossia su un sistema a infrastruttura a chiave pubblica, descritto nel precedente capitolo. Infatti sempre nel medesimo primo articolo si possono leggere le definizioni di chiave privata, chiave pubblica, certificazione, certificatore, validità del certificato, revoca del certificato, sospensione del certificato, validazione temporale; tutte queste definizioni sono conformi a quanto visto nei primi due capitoli.

Tra gli altri articoli del DPR 513/97 ricordiamo l'articolo 4 in cui si afferma che il documento informatico, munito dei requisiti espressi in tale decreto, "*soddisfa il requisito legale della forma scritta*"; negli articoli 5 e 6 si specifica che i documenti informatici e i loro duplicati "*hanno piena efficacia se ad essi è apposta o associata la firma digitale di colui che li spedisce o rilascia*".

L'articolo 7 consente al titolare della coppia di chiavi di poter ottenere il deposito in forma segreta della chiave privata presso un notaio, mentre il successivo articolo obbliga il titolare a sottoporre alla procedura di certificazione la propria chiave pubblica e delega l'AIPA alla gestione dell'elenco pubblico dei certificatori autorizzati.

Nell'articolo 10 si afferma espressamente che “a ciascun documento informatico può essere apposta un firma digitale” e “l'apposizione della firma digitale integra e sostituisce, ad ogni fine previsto dalla normativa vigente, l'apposizione di sigilli, punzoni, timbri, contrassegni e marchi di qualsiasi genere.”

Negli altri articoli del decreto, tra le altre cose, si dà validità legale ai contratti stipulati con strumenti informatici mediante l'uso della firma digitale (art. 11), si equipara la trasmissione del documento informatico per via telematica alla notificazione per mezzo della posta (art. 12), si consente il trasferimento elettronico dei pagamenti (art. 14), è possibile l'archiviazione in supporto informatico, anziché cartaceo, di tutti quei documenti per cui ci sia l'obbligo di tenuta (art. 15), si dà la possibilità ai notai di autenticare la firma digitale (art. 16).

3.2 L'allegato tecnico dell'AIPA

L'allegato dell'AIPA fornisce le regole tecniche da soddisfare per chiunque voglia utilizzare documenti informatici. È suddiviso in quattro parti: nella prima parte si forniscono le varie definizioni utilizzate nell'allegato, si specificano gli algoritmi utilizzabili, si forniscono le caratteristiche generali delle chiavi, le modalità per la loro generazione e conservazione, si definisce il formato dei certificati elettronici, si fissano le regole per la generazione e la verifica della firma.

Nella seconda parte vengono fornite le regole tecniche per la certificazione delle chiavi, vale a dire le regole che devono soddisfare i certificatori. Nella terza parte ci si occupa dei sistemi per la validazione temporale, mentre nell'ultima parte vengono date specifiche regole per le pubbliche amministrazioni.

3.2.1 Le regole tecniche di base

Nel primo articolo AIPA riprende le definizioni contenute nella legge “Bassanini”; in particolare AIPA introduce un nuovo concetto, quello di dispositivo di firma, inteso come “l’apparato elettronico, programmabile solo all’origine, facente parte del sistema di validazione, in grado almeno di conservare in modo protetto le chiavi private e generare al suo interno firme digitali.” Vedremo come questa definizione insieme ad altri articoli ci condizionerà nel nostro progetto.

Vediamo di seguito alcune norme tecniche da rispettare nella gestione di documenti informatici. Per gli algoritmi utilizzati per la generazione e la verifica delle firma AIPA impone l’utilizzo del RSA (Rivest Shamir Adleman) oppure del DSA (Digital Signature Algorithm), mentre come funzioni hash si possono usare gli algoritmi RIPEMD-160 o SHA-1 (art 2 e 3).

Le chiavi e i correlati servizi si distinguono in:

- Chiave di sottoscrizione, destinate alla generazione e verifica delle firme apposte ai documenti (vengono anche chiamate semplicemente chiavi di firma.)
- Chiavi di certificazione, destinate alla generazione e verifica delle firme apposte ai certificati e alle loro liste di revoca (CRL) o sospensione (CSL).
- Chiavi di marcatura temporale, destinate alla generazione e verifica delle marche temporali.

Ogni coppia di chiave può essere attribuita ad un solo titolare e non è consentito un loro utilizzo per funzioni diverse da quelle previste dalla tipologia di appartenenza. Devono essere lunghe almeno 1024 bit.

Gli articoli 5-8 trattano della generazione e della conservazione delle chiavi. La generazione delle chiavi deve essere

effettuata mediante apparati e procedure in grado di assicurare l'unicità e la robustezza della coppia generata e la segretezza della chiave privata, il tutto in rapporto allo stato delle conoscenze scientifiche e tecnologiche. Questi dispositivi devono soddisfare i livelli di sicurezza E3 e di robustezza HIGH definiti dall'ITSEC [ITSEC].

Le chiavi di certificazione e marcatura temporale possono essere generate esclusivamente dal responsabile del servizio che utilizzerà tali chiavi, mentre le chiavi di sottoscrizione possono essere generate, oltre che dal certificatore, anche dal titolare. Se le chiavi di sottoscrizione vengono generate autonomamente dal titolare, tale generazione deve avvenire all'interno del dispositivo (art. 6 comma 3). Questo impone che il dispositivo di firma sia in grado di generare autonomamente la coppia di chiavi. (Vedremo in seguito l'importanza di tale aspetto.) Se, invece, le chiavi di sottoscrizione sono generate dal certificatore, bisogna garantire il trasferimento della chiave privata nel dispositivo di firma in condizioni di massima sicurezza e l'impossibilità di recupero di qualsiasi informazione prodotta durante l'esecuzione di tale procedura.

Le chiavi private vanno custodite all'interno del dispositivo di firma e ne è vietata la duplicazione. Il titolare deve conservare con la massima diligenza il dispositivo di firma e provvedere immediatamente alla richiesta di revoca in caso perda o comprometta il dispositivo.

Gli articoli 9-11 si occupano delle modalità di generazione e verifica della firma. In particolare, viene sottolineato il fatto che, prima di poter procedere alla generazione della firma, il dispositivo di firma debba procedere all'identificazione del titolare e che l'intero processo di generazione debba avvenire all'interno del dispositivo per prevenire possibili intercettazioni del valore della chiave privata. Inoltre ogni messaggio firmato deve sempre essere

accompagnato dal corrispondente certificato elettronico da utilizzare in fase di verifica.

I certificati e le relative liste di revoca devono essere conformi alla norma ISO/IEC 9594-8:1995 con le estensioni definite nella Variante 1, ovvero alla specifica pubblica PKCS#6 e PKCS#9 e successive modificazioni o integrazioni [PKCS6] [PKCS9]. L'accesso al registro dei certificati (Directory) deve essere compatibile col protocollo LDAP (Lightweight Directory Access Protocol) [RFC1777].

I certificati devono contenere almeno le seguenti informazioni:

- numero di serie del certificato
- ragione o denominazione sociale del certificatore
- codice identificativo del titolare presso il certificatore
- nome cognome e data di nascita ovvero ragione o denominazione sociale del titolare
- valore della chiave pubblica
- algoritmi di generazione e verifica utilizzabili
- inizio e fine del periodo di validità delle chiavi
- algoritmo di sottoscrizione del certificato
- tipologia della chiave

3.2.2 Regole tecniche per la certificazione delle chiavi

L'AIPA gestisce direttamente l'intera struttura di certificazione, mantenendo un elenco pubblico dei certificatori, elenco sottoscritto dalla stessa AIPA. Ai certificatori è data la possibilità di definire accordi di certificazione fra loro.

Per ottenere la certificazione di una chiave pubblica, il titolare deve prima essere registrato dal certificatore. La richiesta di

registrazione deve essere redatta per iscritto e conservata dal certificatore per almeno 10 anni. Al momento della certificazione il certificatore deve obbligatoriamente verificare l'identità del docente e deve attribuire a ciascun titolare registrato un codice identificativo (*serial number*) di cui garantisce l'univocità nell'ambito dei propri utenti.

Dopo la registrazione il certificatore pubblica il certificato nel suo personale registro e tale pubblicazione deve essere validata da una marca temporale, che va conservata fino alla scadenza della validità delle chiavi. Il certificato e la relativa marca temporale devono essere inviati al titolare. Per ciascun certificato emesso il certificatore deve fornire al titolare un codice riservato da utilizzare in caso di emergenza per l'autenticazione dell'eventuale richiesta di revoca del sistema.

Un certificato può essere revocato su richiesta del titolare, del certificatore o di un terzo interessato. La revoca comporta la cessazione anticipata della validità del certificato e la sua pubblicazione nella CRL gestita dal certificatore. La revoca diventa definitiva a partire dal momento della pubblicazione della CRL; tale pubblicazione deve essere immediata nel caso in cui la causa della revoca sia la possibile compromissione della chiave privata.

Analoghe procedure avvengono per la sospensione dei certificati con la differenza che la revoca è definitiva, mentre la sospensione no. Il registro dei certificati (Directory) deve, quindi, contenere:

- I certificati emessi dal certificatore
- La lista dei certificati revocati
- La lista dei certificati sospesi

Si possono suddividere le liste dei certificati in più liste distinte e si può replicare il registro dei certificati su più siti, purché venga garantita la consistenza e l'integrità delle copie.

Il certificatore deve provvedere almeno 90 giorni prima della scadenza alla sostituzione delle sue chiavi di certificazione, generando una nuova coppia di chiavi. Deve generare due certificati contenenti la nuova chiave pubblica (uno firmato con la vecchia chiave privata e uno con la nuova) che devono essere trasmessi all'AIPA.

3.2.3 Regole per la validazione temporale

L'AIPA definisce che una "evidenza informatica" (sequenza di simboli binari) è sottoposta a validazione temporale quando viene generata una marca temporale e ad essa viene applicata. La marca temporale deve essere generata da un apposito sistema elettronico sicuro.

La marca temporale deve contenere almeno:

- identificativo dell'emittente
- numero di serie della marca temporale
- algoritmo di sottoscrizione della marca temporale
- identificativo del certificato relativo alla chiave di verifica della marca
- data ed ora di generazione della marca
- identificatore dell'algoritmo di hash utilizzato per generare l'impronta dell'evidenza informatica sottoposta a validazione temporale
- valore dell'impronta dell'evidenza informatica

La data e l'ora contenute nella marca devono essere specificate con riferimento al Tempo Universale Coordinato (UTC); inoltre l'ora della marca deve corrispondere con una

differenza non superiore ad un minuto al momento della generazione.

Ogni coppia di chiave utilizzata per la validazione temporale deve essere univocamente associata ad un sistema di validazione temporale. In più, per limitare il numero di marche temporali generate con la medesima coppia, le chiavi di marcatura temporale debbono essere sostituite dopo non più di un mese di utilizzazione, indipendentemente dalla durata del loro periodo di validità e senza revocare il corrispondente certificato.

I sistemi di validazione temporale devono produrre un registro operativo in cui registrare tutti gli eventi significativi. Inoltre, tutte le marche temporali emesse devono essere conservate in un apposito archivio digitale fino alla scadenza della chiave pubblica della coppia utilizzata per la loro generazione.

3.2.4 Regole tecniche per le pubbliche amministrazioni

L'AIPA consente alle pubbliche amministrazioni di provvedere autonomamente alla certificazione delle chiavi pubbliche dei propri organi e uffici, nell'attività amministrativa di loro competenza, sempre nel rispetto delle regole tecniche e di sicurezza descritte precedentemente. A tal fine possono avvalersi dei servizi offerti da certificatori inclusi nell'elenco pubblico gestito dall'AIPA, rispettando le norme vigenti per l'aggiudicazione dei contratti pubblici.

Capitolo 4. SMART CARD

4.1 Introduzione: perché la scelta delle smart card

Nel capitolo precedente abbiamo preso in considerazione il Decreto del Presidente della Repubblica e il relativo allegato tecnico dell'AIPA che danno validità al documento informatico e alla firma digitale grazie all'utilizzo della crittografia a chiave asimmetrica. Tale tecnologia ha come punto nevralgico la generazione e la conservazione della chiave privata di firma (definita nell'allegato tecnico "chiave privata di sottoscrizione".) La stessa AIPA impone che la generazione della coppia di chiavi di firma avvenga all'interno del dispositivo di firma del titolare e che tale dispositivo sia adeguatamente protetto contro rischi di interferenze e intercettazioni (art. 6-7). In più, i comma 3 e 4 dell'articolo 10 impongono che la generazione della firma debba avvenire all'interno del dispositivo di firma così che non sia possibile l'intercettazione del valore della chiave privata utilizzata; il dispositivo di firma, prima della generazione della firma, deve provvedere all'identificazione del titolare.

È in tale contesto normativo che si inseriscono le smart card quali candidati ideali alla memorizzazione e all'utilizzo delle chiavi di firma. Infatti le smart card realizzano perfettamente quel connubio tra possibilità di memorizzazione dei dati e capacità di elaborazione, il tutto in un ambito in cui la sicurezza gioca un ruolo fondamentale. Pertanto nel nostro progetto, che nel limite del possibile vuole essere il più fedele alle normative dell'AIPA, verranno utilizzate le smart card come dispositivo di firma.

L'obiettivo di questo capitolo sarà quello di illustrare la tecnologia delle smart card; faremo una breve panoramica sulla

storia e gli attuali utilizzi delle smart card per poi concentrarci sul loro utilizzo nella progettazione di un ambiente sicuro per la gestione dei documenti informatici.

4.2 Una panoramica: dalla nascita ai possibili scenari futuri

Nel gennaio del 1974, Ronald Moreno, un inventore autodidatta francese, progettò un sistema di pagamenti rivoluzionario. La sua invenzione consisteva in una applicazione a valore elettronico precaricato, installata su un dispositivo “portatile”. Questo dispositivo doveva essere costituito da un componente di silicio montato su una struttura maneggevole in modo da facilitarne l’uso e quindi la diffusione: poteva, ad esempio, essere montato su carte di credito, orologi da polso, chiavi o anelli [SCM98].

Qualche mese dopo, Moreno propose la sua invenzione ad alcune banche francesi. Il prototipo dimostrativo era costituito da un circuito integrato inserito su un anello e da un dispositivo elettronico per simulare la transazione. L’idea alla base era di dare la possibilità del possessore di tale di anello di “precaricarci” un valore monetario e poi di poterlo spendere presso i punti di vendita che disponessero dell’appropriato equipaggiamento per il pagamento elettronico.

Nel mese di settembre dello stesso anno, il chip venne inserito su una carta invece che su un anello: nacque così la prima smart card⁸. Questa invenzione inizialmente ebbe uno scarso impatto sul mercato e solo alla fine degli anni ’70 iniziarono a interessarsi alcune grandi compagnie come Bull CP8 e sede europea della Motorola.

Da allora fino arrivare ai giorni nostri sono stati compiuti numerosi studi sulle smart card e sono state applicati nei più svariati ambiti: nella telefonia dalle carte telefoniche alle diffusissime carte dei cellulari GSM; nella televisione con le carte per l'accesso alle pay TV; in ambito bancario con le carte di credito; in ambito sanitario con l'intento ambizioso di realizzare una carta medica (*health card*) in grado di contenere informazioni utili in caso di cure mediche d'emergenza (gruppo sanguigno, cartella clinica, radiografie); altri studi e sperimentazioni riguardano l'utilizzo delle carte nell'ambito dei trasporti, come strumento per controllare gli accessi in certi luoghi, come possibilità di identificare le persone.

Attualmente sono in circolazione in tutto il mondo circa un miliardo di smart card e si stima che tale cifra raddoppierà con l'inizio del nuovo millennio. In particolare, nel prossimo decennio si prevede una vertiginosa crescita delle smart card come oggetto di identificazione personale in grado in futuro di sostituire l'attuale carta di identità e come dispositivo per il commercio elettronico che introdurrà il concetto di moneta elettronica. A conclusione di questo scenario futuro, riportiamo di seguito una tabella che dà una stima dell'impiego delle smart card.

⁸ La patria delle smart card può considerarsi a buon diritto la Francia; infatti originariamente le smart card si chiamavano "carte a memoire", cioè carte con memoria.

Applicazioni (cifre in milioni)	1995	1998	2000
Carte telefoniche	420	970	1334
Carte GSM	15	47	61
Carte biomediche	14	122	157
Carte bancarie	17	113	185
Carte di identificazione	-	-	-
Trasporti pubblici	-	-	-
Pay TV	7	62	110
Controllo d'accesso	1	4	10
City card	11	44	143
Totale	485	1362	2000

Fonte: Philips Communication Systems [PCS99]

Tabella 4.1: Il mercato delle smart card entro l'anno 2000

4.3 Che cos'è una smart card

Una smart card (o Integrated Circuit Card - ICC), simile in dimensioni ad una normale carta di credito, è costituita da un particolare circuito integrato, la cui funzione è quella di memorizzare informazioni. A differenza delle carte a banda magnetica, le smart card dispongono di tutte le informazioni e di tutte le funzionalità nella carta stessa e pertanto non hanno bisogno di accedere a un database remoto ad ogni transazione. Inoltre, pur essendo economicamente più costose, forniscono una capacità di memorizzazione assai superiore e offrono una maggiore sicurezza.

A seconda delle caratteristiche del chip, possiamo distinguere due tipi base di smart card [SCO99]: le Memory Card che sono prive di un'unità di elaborazione e vengono utilizzate unicamente per memorizzare dati localmente e le Microprocessor

Card che sono dotate sia di una memoria sia di una CPU in grado di compiere elaborazione dei dati memorizzati. Le memory card, non disponendo di un processore, possono elaborare i dati solo secondo operazioni predefinite. Inoltre hanno una capacità di memorizzazione non molto elevata, di solito da 1 a 4 KB. Sono attualmente le carte più diffuse e rappresentano una alternativa più sicura rispetto alle carte a banda magnetica per molte applicazioni, prima fra tutte quella delle carte telefoniche prepagate.

Le microprocessor card, chiamate anche intelligent card o chip card, oltre ad offrire una maggiore capacità di memorizzazione (da 8 a 16 KB), dispongono di un processore, attualmente con parallelismo a 8 bit (anche se le nuove generazioni di carte avranno un parallelismo a 16 o addirittura a 32 bit), che consente una qualsiasi elaborazione dei dati. Possiamo paragonare tali carte, per potenza di elaborazione, allo storico computer IBM-XT. Queste carte sono utilizzate specialmente in applicazioni crittografiche e sarà su queste che soffermeremo la nostra analisi.

Per completezza, esiste un terzo tipo di smart card, tecnologicamente diverso dai due precedenti: sono le Optical Memory Card. Sono delle carte con un pezzo di CD incorporato nella parte superiore. Hanno una grande capacità di memorizzazione (oltre 4 MB di dati), ma i dati una volta scritti non possono più essere cambiati o rimossi. Attualmente non dispongono di processore, anche se è previsto per il futuro. Questo tipo di carte è ideale per archiviare dati, anche di grandi dimensioni, come ad esempio i file medici.

Riportiamo di seguito una tabella comparativa tra le carte a banda magnetica e i tre tipi di smart card.

	Capacità dati	Processore	Costo carta	Costo lettore e connessione
Carte a banda magnetica	140 byte	Nessuno	\$0.20 - \$0.75	\$750
Memory card	1 Kbyte	Nessuno	\$1 - \$2.50	\$500
Processor card	8 Kbyte	CPU a 8 bit (16 – 32 bit)	\$7 - \$15	\$500
Optical memory card	4.9 Mbyte	Nessuno	\$7 - \$12	\$3,500 - \$4,000

Fonte: Gartner Group

Tabella 4.2: Tabella comparativa tra i vari tipi di carte

4.4 Carte con e senza contatto

Le smart card devono poter comunicare con dispositivi che permettono di accedere ed eventualmente modificare le informazioni memorizzate. Per far questo, le carte hanno due possibilità a seconda che si tratti di contactless card o carte con contatti.

Le prime comunicano usando le frequenze radio tramite degli opportuni ripetitori; questo permette di avere più carte in un unico contenitore, ogni carta esegue una diversa transazione senza che ci siano interferenze tra i diversi segnali. Al momento le contactless card sono poco sperimentate (un'interessante implementazione è in corso a Seoul in Corea [CSCS98]).

Le carte con contatti, invece, stabiliscono la connessione grazie al contatto metallo-metallo che si crea tra il chip della carta e un apposito lettore chiamato Interface Device (IFD). Il lettore comunica col computer tramite porta seriale o PCMCIA ed è alimentato da un trasformatore esterno oppure è dotato di una porta PS/2 per prendere l'alimentazione dalla tastiera. In questo modo, il

lettore può fornire l'alimentazione alla carta e stabilire con essa una connessione per lo scambio di dati.

4.5 Standard

La necessità di stabilire degli standard nasce dal bisogno di garantire portabilità alle proprie applicazioni sviluppate con smart card. Assicurarsi che carte e i dispositivi di lettura siano uniformati a certi standard consente un'interoperabilità tra i produttori di carte e i produttori di dispositivi e facilita la diffusione stessa di tale tecnologia.

Purtroppo la standardizzazione delle smart card con processore è ancora lontana e per ora si è riusciti solo a uniformare alcuni aspetti. Di seguito ci occuperemo brevemente dello standard ISO 7816 che è riuscito a definire delle caratteristiche meccaniche ed elettriche uguali per tutte le smart card. Poi considereremo quali sono gli standard de facto per quel che riguarda le smart card per applicazioni crittografiche.

4.5.1 Lo standard ISO 7816

Lo standard ISO 7816 (parte 1-6) definisce le caratteristiche fisiche ed elettriche, universalmente accettate per le smart card con processore. Tale standard garantisce una matrice minima di interoperabilità tra i vari produttori, limitandosi semplicemente a definire la dimensione e la posizione dei contatti, i segnali elettronici e i protocolli di trasmissione e le operazioni di basso livello di lettura e scrittura sulla carta [ATS95].

La seguente figura mostra le caratteristiche fisiche di una smart card come definito nella parte 1 dell'ISO 7816 [CD98].

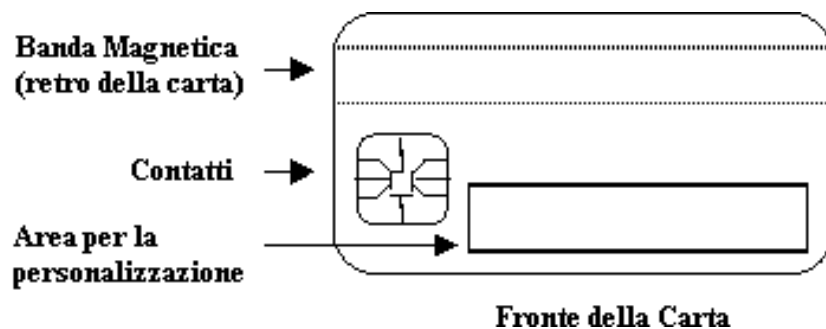


Figura 4.1: Fronte di una smart card (caratteristiche fisiche)

L'area per la personalizzazione (embossing area) è riservata all'incisione col laser o alla scrittura in rilievo dei dati personali del possessore: nome, cognome, numero della carta o altri dettagli personali rilevanti per la particolare applicazione in cui la carta è impiegata.

La successiva figura, in base alla parte 2 dell'ISO 7816, mostra l'interfaccia di comunicazione della carta, che è costituita da 8 contatti elettrici: supply voltage, reset, clock, ground, programming voltage, input/output e 2 contatti opzionali.

Per ciò che riguarda le altre parti dello standard ISO 7816, ricordiamo semplicemente che le smart card dialogano col mondo esterno usando un tipo di pacchetti denominati APDU (Application Data Unit) secondo un modello master/slave, in cui la carta svolge sempre un ruolo passivo in ricezione di un comando a cui inviare poi una risposta (si parla di *command APDU* e *response APDU*.)

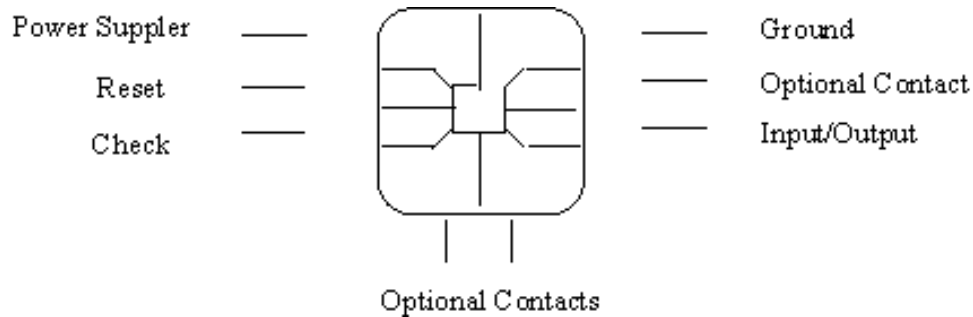


Figura 4.2: Contatti elettrici di una smart card

4.5.2 Standard per la comunicazione

I lettori costituiscono il mezzo mediante il quale un'applicazione può comunicare con una carta. Tale comunicazione avviene secondo un preciso protocollo. Purtroppo, il gruppo ISO (International Standard Organization) non è ancora riuscito a sviluppare uno standard di comunicazione. Di conseguenza, ogni produttore fornisce un diverso protocollo per dialogare con la smart card e pertanto, per poter inviare un comando alla carta da un'applicazione, bisogna trovare il comando che la carta supporta, incapsularlo in un pacchetto secondo lo standard ISO e infine avvolgere questo comando nel particolare pacchetto richiesto dal lettore in uso.

Se consideriamo applicazioni crittografiche, possiamo individuare tre standard di fatto:

- PKCS #11 (che tra poco sarà rimpiazzato dal PKCS #15), sviluppato dalla RSA Laboratories

- PC/SC (sigla per Interoperability Specification for ICCs and Personal Computer Systems), sviluppato dalla Microsoft insieme ad altre compagnie⁹.
- OpenCard Framework (OCF), sviluppato dall'IBM, Oracle, Sun e Netscape.

Essendo il mercato delle smart card ancora molto giovane e quindi in rapida evoluzione, è molto difficile fare un raffronto tra i vari standard. Di seguito, riportiamo alcune considerazioni che sono un lavoro di analisi e sintesi di varie pubblicazioni [K98], [MFAQ98], [OFC98], [PCSC97], [PKCS11], [PCKS15], [RFAQ99] :

- Prima di tutto va detto che PKCS#11 non fornisce una architettura completa, come PC/SC e OCF, ma unicamente uno standard API che consente di interfacciarsi alle funzioni crittografiche della carta; pertanto non dispone di funzioni di gestione del file system della smart card (es.: scrivi file, leggi file, crea directory). PKCS#11 gestisce la carte come degli oggetti (*cryptographic token*) con metodi ad alto livello, non effettuando quindi una netta distinzione tra lettore e smart card. Questo fa sì che si possano verificare problemi di interoperabilità qualora i produttori di smart card e di lettore siano differenti. Inoltre si rischia di perdere la portabilità tra smart card diverse, anche se tutte supportano il PKCS#11. Infatti ogni carta, compatibile con tale standard, possiede un proprio driver che mappa le chiamate PKCS#11 in chiamate di basso livello, secondo lo standard ISO 7816. Tali funzioni PKCS#11, essendo di alto livello, possono non funzionare più su un altro tipo di carta con un file system formattato

⁹ Le compagnie che stanno sviluppando PC/SC sono: Bull CP8, Gemplus, Schlumberger, Siemens-Nixdorf, Hewlett Packard, Toshiba, Verifone, IBM, Sun, Microsoft.

diversamente; in altre parole, ogni carta richiede un diverso driver.

- A causa di tale di incompletezza, RSA ha deciso di produrre un nuovo standard che si chiamerà PKCS#15. Per ora è disponibile solo un draft di queste specifiche. Da tale draft si evince che il nuovo protocollo, denominato "Information Format for Cryptographic Tokens", oltre a mantenere una piena compatibilità con lo standard PKCS#11, non sarà più semplicemente un'interfaccia API, ma diventerà una architettura che garantirà interoperabilità tra piattaforme diverse, tra produttori di smart card diversi e tra applicazioni software diverse.
- Fondamentalmente PC/SC e OpenCard presentano un'architettura molto simile e si pongono, quindi, sul mercato come due standard alternativi. La principale differenza tra PC/SC e OCF sta nella filosofia di progettazione delle due architetture: OCF si distingue per un'implementazione completamente object-oriented. Per ora non c'è compatibilità tra i due standard, anche se OpenCard ha annunciato che presto le sue specifiche saranno completamente compatibili con quelle del PC/SC. Inoltre, mentre OpenCard può girare su qualsiasi sistema operativo in quanto è stato scritto tutto in Java, PC/SC è compatibile solo con sistemi Win32 (Windows NT o Windows 95-98).
- Infatti, la filosofia della Microsoft con lo standard PC/SC è di proporre una forte integrazione con il suo nuovo sistema operativo, Windows 2000. In questa integrazione va segnalato il cosiddetto "*Public Key Interactive Logon*": la Microsoft ha annunciato che il nuovo sistema operativo prevederà il supporto completo per PKI, consentendo l'autenticazione dell'utente non più semplicemente tramite la coppia username - password, ma mediante smart card. La smart card, inserita in un apposito

lettore e abilitata dal suo PIN, fornirà al processo di autenticazione il certificato elettronico del possessore di carta. Il sistema verificherà l'autenticità di tale certificato verificando la firma dell'autorità di certificazione, e in caso positivo potrà dedurre la validità della chiave pubblica e quindi verificare la corrispondenza con la chiave privata, anch'essa contenuta nella smart card.

- Per quanto riguarda le API per lo sviluppo di applicazioni crittografiche, il PC/SC propone le CryptoAPI, mentre PKCS#11 le Cryptoki, che attualmente sono supportate dai principali tool di sviluppo per PKI, mentre ancora scarsa è la diffusione delle API Microsoft¹⁰.

4.6 Le smart card per applicazioni crittografiche

Una smart card, se utilizzata per applicazioni crittografiche, deve disporre di alcune proprietà aggiuntive perché possa soddisfare le specifiche contenute nel regolamento dell'AIPA. È proprio l'esistenza di carte con tali caratteristiche che ci porta a scegliere le smart card come dispositivo di firma per la nostra applicazione.

4.6.1 Active RSA

Una smart card con la proprietà dell'*active RSA* è una particolare carta a microprocessore che ha, incluso nel suo sistema operativo interno, il software (eventualmente anche in parte cablato nell'hardware) per implementare il suddetto algoritmo a chiave

pubblica. Tale algoritmo viene applicato al digest, cioè al risultato ottenuto dal calcolo del hash sul documento, utilizzando la chiave privata per firmare l'hash e ottenere la cosiddetta fingerprint (impronta digitale o hash firmato). Se la carta è di tipo RSA attivo, in tale operazione la chiave privata non esce mai dalla carta, nemmeno a runtime, e questo evita possibili *sniffing*¹¹ da parte di un virus residente in memoria.

Va osservato che l'operazione di hash non avviene all'interno della carta come per la cifratura; questo perché il processore della carta ha una bassa capacità di elaborazione e si troverebbe a calcolare il digest di documenti anche di dimensioni considerevoli (alcuni Mbyte). In più, tali documenti dovrebbero essere trasferite da lettore alla carta e usando ad esempio lettori seriali, la cui velocità massima è di 115 Kbps, si avrebbe un trasferimento eccessivamente lento.

Le smart card con *active RSA* consentono di soddisfare l'articolo 10 del regolamento dell'AIPA che impone che l'operazione di firma avvenga all'interno del dispositivo di firma.

4.6.2 Key Generation

Le smart card che hanno la *key generation* sono in grado di generare autonomamente, all'interno della carta, la coppia di chiavi utilizzata dal RSA. In tal modo si evita, come per l'*active RSA*, che le chiavi siano in un certo momento fuori dalla memoria della carta. Questo permette alla carta di non avere condizionamenti esterni mentre genera la coppia di chiavi ed evita quindi tutti quei

¹⁰ Entrust, il tool PKI utilizzato nel nostro progetto, supporta unicamente lo standard PKCS#11. Al momento sono pochi i tool che, invece, supportano PC/SC. Tra essi citiamo ID2 prodotto dalla omonima azienda svedese.

¹¹ Sniffing: possibilità di "spiare" ciò che è stato temporaneamente copiato sulla memoria principale (RAM) durante un'elaborazione da parte della CPU.

problemi che si possono verificare se la *key generation* avvenisse, invece, in un computer, eventualmente collegato in rete (es.: attacco di un virus residente in memoria). Inoltre la carta dispone di un algoritmo "certificato" pseudo-random per la generazione di numeri casuali che garantisce, nel limite del possibile, l'univocità e l'equiprobabilità della coppia generata.

La *key generation* consente alle smart card che ne sono equipaggiate di poter soddisfare ai requisiti richiesti dall'AIPA in merito alla generazione e conservazione della coppia di chiavi di firma (articoli 5-8.)

4.6.3 Capacità di memorizzazione

Una smart card possiede tre tipi di memorie:

- ROM (Read-Only-Memory), ossia una memoria non volatile a sola lettura, in cui è memorizzato il sistema operativo della carta; inoltre, se la carta è *active RSA*, nella memoria vi è stato anche cablato l'omonimo algoritmo.
- RAM (Random-Access-Memory), ossia una memoria volatile, utilizzata dal microprocessore durante l'elaborazione *runtime* per scrivere dati temporanei.
- EEPROM (Electrically-Erasable-Programmable-ROM), ossia una memoria non volatile programmabile, in cui è contenuto il file system della carta. Tale memoria funge da deposito di memorizzazione dei dati all'interno della carta ed è indubbiamente delle tre memorie quella che qualifica di più la smart card, in quanto ROM e RAM incidono solo sulla velocità di elaborazione del processore, ma non sulla capacità di memorizzazione dei dati. È nella EEPROM che vengono memorizzati le chiavi e certificati elettronici. In particolare avremo bisogno almeno di memorizzare le due chiavi private

(firma e cifratura) e i relativi certificati elettronici e il certificato CA. Inoltre potrebbe essere necessario memorizzare altri certificati quali il certificato SSL per l'accesso a siti Web). Poiché ogni certificato occupa una dimensione di circa 1-1,5 Kbytes, mentre la chiave RSA è almeno di 1024 bit (per applicazioni a massima sicurezza anche a 2048 bit), è necessario o che tale memoria sia almeno di 8 Kbytes.

Una smart card di prestazioni ordinarie potrebbe avere 16 KB di ROM, 8 KB (o 16 KB) di EEPROM e 256 byte (o 512 byte) di RAM.

4.7 Particolari smart card

A conclusione di questo studio di ricerca sulle smart card ci soffermiamo brevemente su due "particolari" tipi di carte che stanno ultimamente affermandosi e che probabilmente occuperanno una fetta importante del mercato delle smart card.

4.7.1 Smart card biometriche

Sono particolari smart card che come autenticazione non utilizzano il PIN (Personal Identification Number), cioè un codice numerico posseduto dal proprietario della carta, ma utilizzano le impronte digitali [ABC99]. Infatti, oltre al lettore di smart card, c'è un dispositivo in grado di rilevare e confrontare le impronte digitali. Queste carte attualmente sono quelle più sicure, perché non permettono il prestito della carta e non presentano problemi in caso di smarrimento o furto, garantendo in questo modo un'associazione univoca tra proprietario e carta e conseguentemente tra proprietario

e coppia di chiave crittografiche. (Potranno essere utilizzate anche per il *Public Key Interactive Logon*.)

Con queste carte l'identificazione del titolare (come richiesto dell'AIPA, articolo 10 comma 4) è molto più sicura che con l'utilizzo di un codice numerico o password.

4.7.2 Java Card

Una Java Card della Sun Microsystem [CD98] [JC99] è una particolare smart card dotata di una Java Virtual Machine (JVM), implementata nel sistema operativo della carta, che consente l'esecuzione sul chip del linguaggio Java. Il vantaggio di tali carte consiste nel sfruttare Java, come linguaggio *platform-independent*. Infatti, al momento non esiste un linguaggio standard per smart card. Le software house sviluppano il loro codice con diversi linguaggi (es: C, C++, assembler) e poi lo compilano in un linguaggio macchina strettamente legato al chip della carta. Tutto ciò pone forti limitazioni sulla portabilità e flessibilità delle applicazioni. Un programma scritto in Java, invece, girerà su qualsiasi altra piattaforma Java compatibile, vale a dire che un applicazione Java girerà in una qualsiasi smart card che contenga una JVM.

Per la programmazione, la Sun ha sviluppato le Java Card API, particolari API scritte apposta per processori a 8 bit che racchiudono in sé tutte le proprietà principali del linguaggio Java con alcune importanti eccezioni (es: senza multitreading, nessun supporto per il garbage collector e alcune differenze nei tipi primitivi di dati.)

Va infine notata la differenza tra le Java Card e OpenCard Framework: entrambi sono scritti in Java, ma OCF gira sul lettore o

sulla workstation e fornisce delle API per la comunicazione tra carta e lettore.

Capitolo 5. LE SPECIFICHE DEL PROGETTO

5.1 Introduzione: le entità del sistema

In questo capitolo descriveremo l'intera struttura del sistema di compilazione e trasmissione dei verbali d'esame: considereremo le varie entità che fanno parte del sistema e i protocolli mediante i quali esse interagiscono. Naturalmente, l'ossatura del nostro sistema sarà un'architettura PKI (Public Key Infrastructure) descritta nel secondo capitolo, mentre per tutto quello che riguarderà i protocolli di comunicazione tra i vari componenti della PKI seguiremo lo standard Internet "Certificate Management Protocols" [RFC2510].

Riconsiderando i componenti di un'infrastruttura a chiave pubblica, abbiamo visto che possiamo distinguere concettualmente tre tipi di entità: le *end entities* (EE), la *registration Authority* (RA), la *Certificate Authority* (CA).

Con il termine *end entities* (utenti finali) intendiamo tutti i componenti del sistema che interagiscono con esso attivamente, usufruendo dei vari sistemi messi a disposizione da un'infrastruttura a chiave pubblica. Le *end entity* non necessariamente devono essere degli "utenti umani" che utilizzano applicazioni software di interfaccia con la PKI ma possono essere delle applicazioni autonome. Nel nostro sistema avremo delle *end entity* rappresentate dai docenti che, accedendo a dedicate postazioni client e utilizzando una specifica interfaccia, potranno verbalizzare gli esami in maniera digitale. In più, ci sarà un'applicazione server che riceverà i verbali compilati e firmati dai docenti e provvederà alla loro verifica e memorizzazione.

L'Autorità di Registrazione è l'entità che deve inizialmente accertare l'identità dell'utente che richiede un certificato elettronico in base a una certa politica di sicurezza che verrà in seguito descritta. Pertanto nel nostro sistema la RA avrà il compito di registrare i vari docenti e fornire loro l'autorizzazione iniziale e le relative carte d'accesso.

L'Autorità di Certificazione gestisce l'intero ciclo di vita dei certificati elettronici assegnati ai vari utenti che precedentemente si erano registrati presso l'Autorità di Registrazione, dalla creazione alla revoca, passando attraverso fasi di aggiornamento e sostituzione. Sarà sempre la CA che avrà l'onere di stabilire relazioni di fiducia con altre CA, qualora diventi necessario per la nostra applicazione un'interazione col "mondo esterno".

A queste tre entità se ne affianca una quarta: il *timestamp server*, ossia un sistema di validazione temporale che è in grado di fornire ai documenti una data e un ora autenticata. Vedremo in dettaglio il suo funzionamento nel successivo capitolo quando ci occuperemo dell'implementazione del sistema,

A queste tre entità si affiancheranno due sistemi di immagazzinamento dei dati: il sistema di Directory, che avrà il compito di memorizzare e rendere pubblici i vari certificati elettronici e di gestire la Certificate Revocation List (CRL) e il database che conterrà i vari verbali d'esame firmati dai docenti e sarà gestito dal server.

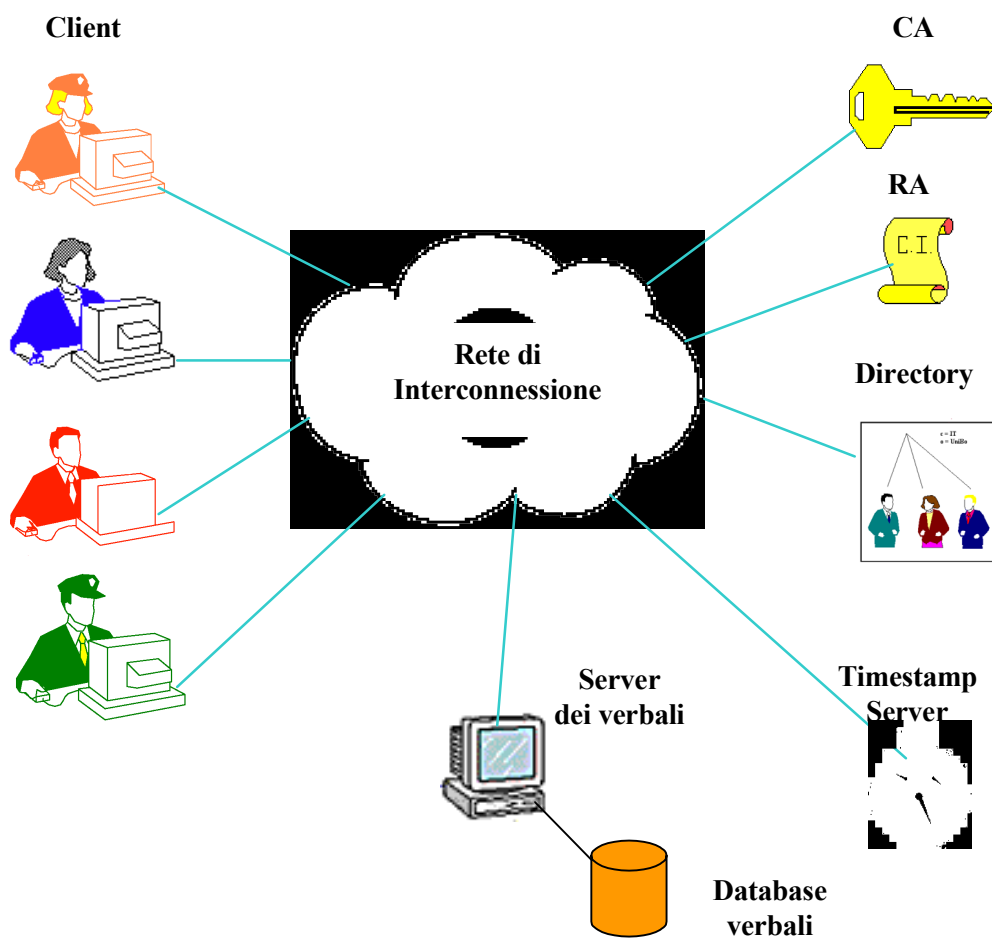


Figura 5.1: Schema dell'organizzazione del sistema complessivo

5.2 Inizializzazione dei docenti

La fase di inizializzazione dei docenti è quella fase iniziale tramite la quale i vari docenti, accedendo a una postazione client, si identificano per la prima volta alla PKI e grazie alla quale, in caso di esito favorevole, potranno successivamente essere abilitati a svolgere le varie operazioni di firma e cifratura all'interno dell'organizzazione universitaria. Questa fase è così articolata:

- 1. Registrazione iniziale:** i docenti, come qualsiasi end entity, devono presentarsi all'Autorità di Registrazione e, una volta identificati, vengono registrati dalla RA come professori universitari. Al docente viene dato, in supporto cartaceo, un valore segreto iniziale (Initial Authentication Key) relativo ad un certo valore di riferimento (Reference Value), in maniera analoga a quanto fa una banca con i suoi clienti quando fornisce il PIN (Personal Identification Number) per l'utilizzo di una carta bancomat. Questa coppia di valori serve all'utente per disporre dell'autorizzazione necessaria nella successiva fase di certificazione con la CA. Inoltre la RA fornisce a ciascun docente un dispositivo di memorizzazione per la coppia di chiavi (per esempio, una smart card) e un floppy disk, il cui utilizzo sarà spiegato in seguito. Va osservato che questa fase viene svolta completamente in modalità "out-of-band", ossia senza ricorrere ad alcun tipo di comunicazione che utilizzi una struttura di reti di calcolatori. La RA potrebbe coincidere con l'ufficio personale dell'Ateneo: alla presa di servizio di un nuovo docente si provvede anche alla sua registrazione iniziale.
- 2. Key Generation:** l'utente, inserita la smart card nell'apposito dispositivo di lettura, tramite l'interfaccia grafica dell'applicazione client può generare la sua coppia di chiavi. La generazione avviene completamente all'interno della smart card, grazie al chip che ivi è incorporato, in modo che la chiave privata non esca mai dalla carta. Si noti che il nostro sistema prevede la possibilità che l'utente possa svolgere operazioni sia di firma sia di cifratura; pertanto verranno generate due coppie di chiavi, a cui corrisponderanno due certificati elettronici.¹²

¹² In realtà, in conformità con le diverse necessità di gestione delle chiavi solo la coppia di chiavi di firma viene generata all'interno della smart card, mentre la coppia di chiavi di cifratura viene generata dalla PKI e successivamente trasferita nella carta.

- 3. Richiesta di certificazione:** a questo punto l'applicazione genera un messaggio di richiesta di certificazione per la CA. Tale richiesta può essere autenticata grazie all'utilizzo dell'associazione Reference Value-Initial Authentication Key (RV-IAK), che garantisce che l'utente era stato registrato precedentemente dalla RA. Il messaggio con la chiave pubblica dell'utente è, quindi, spedito alla CA in modalità "on-line" in virtù del fatto la coppia numerica (RV-IAK) ha creato un canale di comunicazione autenticato tra end-entity e la CA. Va osservato che tale canale, finita la fase di inizializzazione, non potrà più essere utilizzato.¹³
- 4. Verifica della richiesta di certificazione:** la CA alla ricezione del messaggio ne verifica l'autenticità tramite la coppia RV-IAK e in caso di successo provvede a generare un profilo per l'utente e la creazione di due certificati elettronici, uno per la chiave pubblica di firma, l'altro per la chiave pubblica di cifratura. Questi certificati, oltre alla rispettiva chiave pubblica, contengono il nome e il cognome del docente, il nome della CA di appartenenza, la validità temporale delle chiavi e altre informazioni in conformità con lo standard X.509v3 [RFC2459]; sono memorizzati nel relativo sistema di directory¹⁴, e firmati con la chiave privata della CA, sono spediti insieme al certificato della stessa CA alla postazione client che ha richiesto la certificazione.
- 5. Conferma della certificazione:** il docente riceve dalla CA il messaggio di risposta alla richiesta di certificazione; l'applicazione client verifica i certificati tramite la chiave pubblica della CA e in caso di successo memorizza i certificati elettronici ricevuti nella smart card. Ovviamente il docente deve

¹³ Per creare questo canale autenticato, Entrust utilizza un suo protocollo SEP (Secure Exchange Protocol) che utilizza la crittografia a chiave simmetrica.

¹⁴ Come vedremo in seguito, Entrust memorizza nella directory solo i certificati di cifratura e non anche quelli di verifica.

già possedere in maniera fidata la chiave pubblica della CA; per far questo la CA nella fase di registrazione iniziale deve fornire in modalità “out-of-band” l’impronta digitale (*fingerprint*) della sua chiave pubblica¹⁵.

Riportiamo di seguito il diagramma di flusso dei dati dell’inizializzazione del docente corredato dal relativo dizionario dei dati.

Notazione	Descrizione
RV	Reference Value: è un valore di riferimento generato dalla RA per la prima identificazione dell’utente.
IAK	Initial Authentication Key: è il codice segreto relativo al RV che consente a una end-entity di autenticarsi quando accede per la prima volta alla CA.
Kpub	Sono le due chiavi pubbliche: una di firma, l’altra di cifratura.
Kpriv	Sono le due chiavi private: una di firma, l’altra di cifratura.
Certificato CA	E’ il certificato elettronico X.509v3 dell’Autorità di certificazione
Certificato docente	Sono i due certificati elettronici X.509v3 corrispondenti alle due chiavi pubbliche.

Tabella 5.1: Dizionario di Dati per l’inizializzazione del docente

¹⁵ Entrust non richiede la verifica della chiave pubblica della CA in quanto la sua validità è già garantita dall’utilizzo del protocollo SEP.

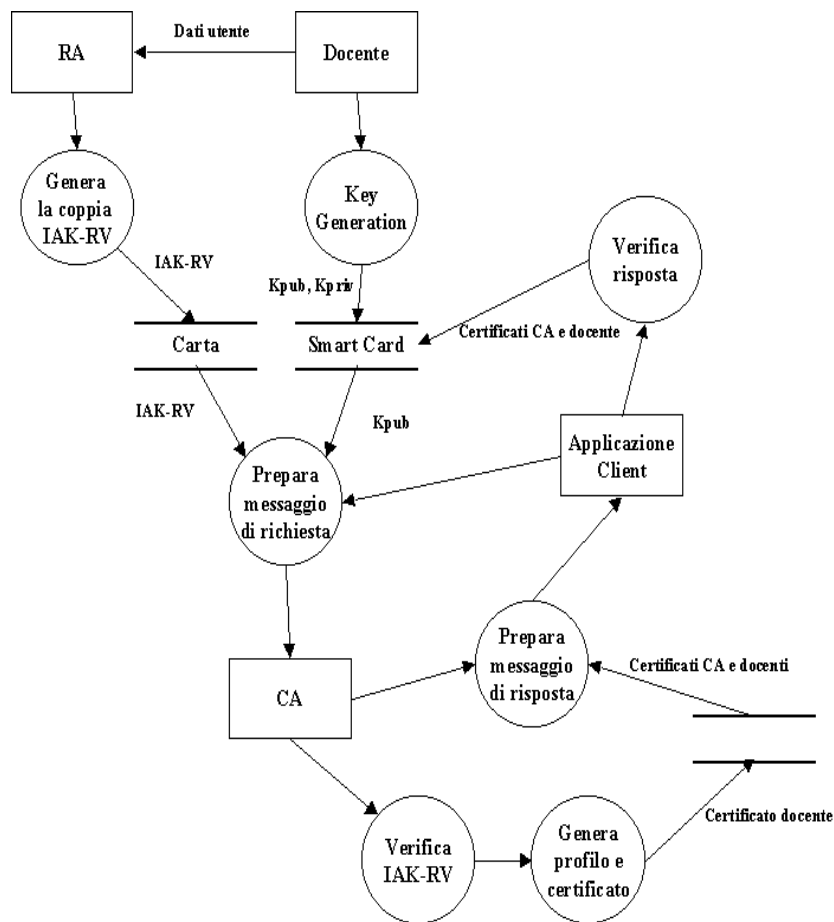


Figura 5.2: Data Flow Diagram per l’inizializzazione del docente

5.3 Inizializzazione dell’applicazione server

Anche l’applicazione server, residente in un dedicato elaboratore, da noi inseguito denominato server, essendo anch’essa un end-entity, avrà una fase di inizializzazione simile a quella dei docenti. La principale differenza deriva dal fatto che l’applicazione, una volta inizializzata, non richiederà più l’intervento umano, se non per motivi di manutenzione. Pertanto la fase di inizializzazione

si tradurrà semplicemente nel generare un profilo per tale applicazione, che sarà anch'essa registrata dalla CA. Ovviamente a differenza dei client utilizzati dai docenti, l'applicazione server non disporrà di una smart card per la memorizzazione delle due coppie di chiavi, le quali saranno semplicemente memorizzate localmente nella postazione server. Va da sé che l'elaboratore server dovrà essere custodito in un ambiente intrinsecamente sicuro in modo da evitare un'eventuale manomissione o copia di tali chiavi. Un analogo discorso andrà fatto per il sistema che ospita l'intera struttura PKI (RA, CA e directory.)

Nella postazione server sono anche presenti le cosiddette *capability* dei docenti, ossia è definita una matrice degli accessi in virtù della quale si stabilisce per ogni corso d'esame quale docente risulti titolare (e quindi, se abbia potere di firma dei verbali.) Queste *capability* sono fornite dalle Segreterie di Facoltà in base ai carichi didattici stabiliti per ogni anno accademico dai diversi Consigli dei Corsi di laurea.

5.4 Verbalizzazione degli esami

Una volta terminata la fase di inizializzazione dell'applicazione server, ogni docente, dopo avere eseguito la propria fase di inizializzazione, è in grado, mediante l'interfaccia grafica del client, di poter compiere tutte quelle operazioni necessarie per la verbalizzazione digitale di un esame. Si suppone che ogni verbale sia relativo ad un unico studente esaminato. L'intera operazione attraversa le seguenti fasi:

1. Il docente accede attraverso una qualsiasi postazione client, carica l'applicazione "client" che consente la verbalizzazione dell'esame. Per utilizzare tale applicazione, il docente deve

inserire la propria smart card e il floppy disk nei rispettivi lettori. L'applicazione chiede il codice di protezione della carta (PIN), che garantisce l'autenticazione del possesso della smart card da parte del docente. Se il PIN è corretto, allora, tramite le informazioni contenute nel certificato elettronico del docente (memorizzato nella carta), viene verificato che il possessore della carta sia effettivamente un docente; per far ciò si utilizzano le estensioni del certificato secondo lo standard X.509v3.

2. Se l'accesso viene autenticato, l'applicazione, mediante l'interfaccia grafica, consente al docente la compilazione del verbale per gli esami di cui ha diritto: il docente inserisce i vari dati richiesti dalla schermata (anagrafica dello studente, voto, quesiti, ecc.). In tale fase di compilazione, l'applicazione può prelevare dal floppy disk tutte quelle informazioni che alleggeriscono il docente dall'onere della digitazione (es: numero del verbale, codice esame, codice docente, ecc.) Al termine della compilazione il docente avvia la procedura di firma del verbale: l'applicazione calcola mediante un'opportuna funzione hash il *digest message* del documento contenente i dati del verbale appena compilato e passa tale *digest message* alla smart card, la quale, pilotata dall'applicazione, effettua l'operazione di firma del *digest* tramite la chiave privata di firma del docente contenuta nella carta. Il *digest message* firmato, che rappresenta la firma del verbale, viene restituito all'applicazione. Poiché un verbale deve essere firmato da due docenti, l'applicazione provvede a ripetere la prima fase di autenticazione per il secondo docente, il quale, una volta autenticato, firma il verbale, anche lui utilizzando la sua smart card. A questo punto il messaggio è costituito dal verbale in chiaro (in quanto il verbale, essendo un documento pubblico,

non ha bisogno di essere crittografato) e dalle firme elettroniche dei due docenti.

3. Il messaggio, prima di essere inoltrato al server, viene spedito a un gestore di servizi di marcatura temporale (*timestamp server*) che marca con la data e l'ora di trasmissione il verbale: al documento viene apposta una terza firma, quella del *timestamp server*, che consente al verbale di avere un riferimento temporale autentificato.
4. Finito il processo di validazione temporale, il *timestamp server* restituisce il messaggio al client la quale provvede a spedirlo al server tramite un appropriato protocollo di trasporto.
5. Il server, a meno di problemi occorsi in fase di trasmissione sulla rete, riceve il verbale firmato e passa a verificarne la firma. Per fare ciò, utilizza la chiave pubblica di verifica del docente contenuta nel relativo certificato (facente parte anch'esso del messaggio inviato dalla postazione client¹⁶); tale chiave pubblica deve essere verificata in quanto potrebbe non essere più valida (es: un docente ha smarrito la smart card oppure è stato trasferito; in entrambi i casi il suo certificato elettronico è stato, pertanto, revocato); per far questo, l'applicazione deve connettersi con la Directory, nel quale sono contenuti tutti i certificati validi e quelli invece revocati (CRL). Oltre alla verifica della firma, bisogna verificare le *capability* di colui che ha firmato il documento, ossia bisogna verificare che il mittente, oltre ad avere la qualifica di docente, stia firmando un verbale per un esame per il quale ha effettivamente il diritto di docenza. A questo scopo l'applicazione server consulta il database locale contenente le *capability* dei docenti e ne verifica la legittimità di tale firma. Se la verifica del documento va a buon fine e quindi il verbale non è stato alterato, l'applicazione server comunica

¹⁶ Il regolamento dell'AIPA richiede espressamente che il certificato di verifica della firma sia incluso nel messaggio firmato, nonostante il server avrebbe la possibilità di disporre già di una copia di tutti i certificati dei docenti.

all'applicazione client il buon esito della verbalizzazione, mandando un messaggio d'acknowledge firmato con la propria chiave privata di firma. Il verbale firmato è infine memorizzato in un apposito database.

6. L'applicazione cliente riceve il messaggio d'acknowledge firmato dal server, ne verifica la firma e, in caso di successo, ne deduce la buona riuscita dell'intera operazione.
7. Se invece l'applicazione server constata un errore nel processo di verifica del verbale firmato, non ritiene valido tale documento; pertanto non effettua nessuna sua archiviazione, ma manda un messaggio di notifica di errore al client (anch'esso firmato). Il client, ricevendo tale notifica di errore, ha l'onere di rispedire nuovamente il verbale firmato. Quando descriveremo nel dettaglio il protocollo di comunicazione vedremo come gestire tali situazioni.

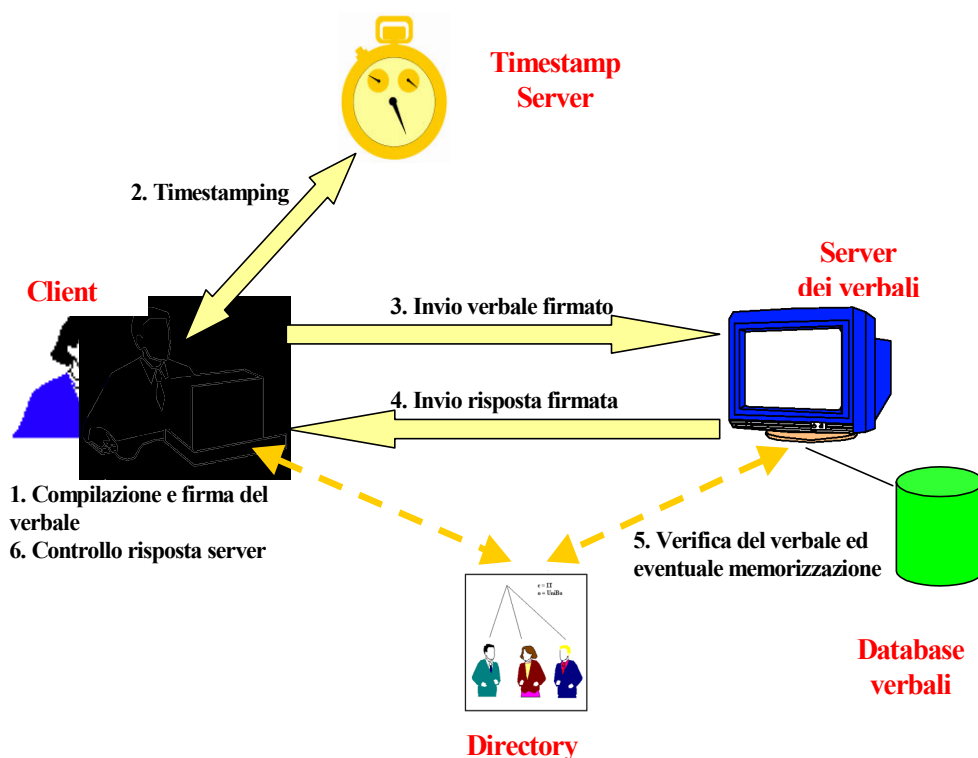


Figura 5.3: Verbalizzazione di un esame

Il protocollo di comunicazione tra client e server deve ovviamente tenere in conto la possibilità di eventuali malfunzionamenti nella comunicazione tra i due nodi remoti, che possono comportare la perdita di messaggi. Inoltre ci dovrà essere un appropriato grado di replicazione dei verbali sia dal lato client sia dal lato server affinché l'intero sistema presenti un certo livello di *fault tolerance*, che lo renda robusto nei confronti di guasti dei dispositivi fisici di memorizzazione. Perciò, nella fase di realizzazione, dovremmo tener conto di queste due esigenze: garanzia di una comunicazione affidabile e robustezza a fronte di guasti.

Riassumendo i vari componenti software del sistema abbiamo:

- Applicazione client: consente, attraverso un'interfaccia grafica *user-friendly*, un veloce e intuitivo utilizzo da parte dei docenti, che tramite quest'applicazione possono inicializzarsi e successivamente compilare, firmare e poi spedire al server i vari verbali.
- Applicazione server: riceve i verbali firmati dai docenti, verifica la loro validità e provvede alla loro archiviazione.
- Sistema PKI: costituito dall'Autorità di registrazione, da quella di certificazione e dal sistema di directory, consente l'inicializzazione dei docenti e dell'applicazione server e gestisce tutti i compiti tipici di un'infrastruttura a chiave pubblica, quali la gestione delle chiavi e dei certificati elettronici.

5.5 Il protocollo di comunicazione

Come è stato descritto precedentemente, la comunicazione tra le due end-entity avviene secondo un semplice protocollo tipico dei modelli cliente - servitore, ove il cliente manda un messaggio di

"richiesta" (che consiste nel verbale di un esame) e il server manda una risposta (*acknowledge*) che rappresenta la riceva della richiesta del cliente. I messaggi possono viaggiare in una qualsiasi rete di interconnessione (ad esempio Internet) senza aver problemi di sicurezza, poiché la sicurezza non è demandata al mezzo trasmissivo ma è già insita negli stessi messaggi in virtù dell'utilizzo della tecnologia a chiave asimmetrica.

Ciò che ancora non abbiamo considerato è innanzitutto come gestire il fatto che la risposta del server possa non essere positiva, in quanto la procedura di verifica della firma del docente può fallire. Inoltre bisogna tener conto del fatto che stiamo utilizzando un sistema di reti di calcolatori, in cui i messaggi possono essere persi o corrotti, in cui gli elaboratori possono essere temporaneamente non disponibili o irraggiungibili; pertanto nella progettazione del protocollo dovremo considerare i possibili malfunzionamenti dovuti alla rete di interconnessione.

A questi problemi tipici nelle comunicazioni di rete, si aggiunge un altro problema di natura locale che riguarda il fatto che sia da parte del client, sia da parte del server, i verbali devono essere memorizzati in formato elettronico (*file*) su un supporto di memorizzazione fisico (memoria secondaria) che, come tutti i dispositivi fisici, è soggetta a possibili guasti che possono compromettere la reperibilità delle informazioni ivi memorizzate. Pertanto nella progettazione dovremo tenere conto anche di questo aspetto.

5.5.1 Fallimento della verifica della firma del docente

In base a quanto detto precedentemente, il primo caso da gestire è quando l'applicazione server non riconosce la firma del

docente nella fase di verifica. Questo può succedere sostanzialmente per due motivi:

1. Il pacchetto (o i pacchetti) su cui viaggia il messaggio spedito dal cliente viene alterato o a causa di un qualche malfunzionamento verificatosi durante la trasmissione oppure a causa di un attacco attivo da parte di un intrusore che volontariamente tenta di cambiare il contenuto del verbale, violando l'integrità del messaggio.
2. La firma del docente non è più valida in quanto il certificato elettronico contenente la corrispondente chiave pubblica di firma è stato revocato e l'applicazione server se n'è accorta accedendo alla Certificate Revocation List (CRL).

Poiché nel nostro sistema il caso più frequente sarà il primo (molto più rare saranno firme con certificati revocati), si è scelto che l'applicazione server mandi al client un messaggio di notifica di tale insuccesso (messaggio firmato con la sua chiave privata) e che l'applicazione client, a seguito di tale notifica, effettuerà una ritrasmissione del verbale. A questo scopo il verbale firmato, prima di essere spedito, viene memorizzato temporaneamente nella memoria di massa del client (il suo hard disk) e successivamente cancellato quando il server dà la conferma di memorizzazione del verbale nell'apposito database. In questo modo l'applicazione può ritrasmettere il verbale in modo completamente trasparente all'utente, ossia senza che il docente se ne accorga.

Se la verifica di firma questa volta ha successo, l'applicazione server manderà l'acknowledge previsto e il docente riceverà tale notifica senza accorgersi dell'avvenuta ritrasmissione. Nel caso in cui la verifica di firma fallisca ancora, dopo un adeguato numero di ritrasmissioni, l'applicazione client segnalerà al docente la mancata verifica della sua firma. A questo punto il verbale non sarà cancellato dall'hard disk del client: il docente sarà

obbligato dall'applicazione a memorizzare sul suo floppy il verbale non trasmesso (successivamente vedremo come gestire questi verbali non trasmessi.)

Va osservato che l'applicazione client è strutturata in modo da sospendere il docente dalla compilazione di un altro verbale fin quando non riceva una risposta, positiva o negativa, da parte del server. Inoltre il server farà un'attività di *auditing*, ossia memorizzerà in un file tutti i tentativi di fallimento di verifica della firma.

5.5.2 Fallimento delle verifica delle capability del docente

Un altro motivo per cui il server non può dare una risposta positiva al client è dovuta al mancato riscontro delle capability del docente. Il server, se constata che il verbale è relativo ad un esame di cui il docente non ha diritto di firma, notifica tale fatto all'applicazione client: a questo punto sarà premura del docente verificare la correttezza del file contenuto nel suo floppy disk che fornisce all'applicazione le informazioni con cui è possibile inizializzare i verbali (corsi, codice docente, ecc.)

5.5.3 Fallimento della verifica della firma del server

Nella comunicazione tra cliente e servitore, non solo il verbale è firmato, ma lo è anche il messaggio di risposta da parte dell'applicazione server, cosicché il client possa verificare l'autenticità della risposta. Pertanto, si potrà verificare un fallimento anche nella procedura di verifica della firma del server. In tal caso, il client, non essendo notificato dell'avvenuta ricezione

del verbale da parte del server, ricorre alla ritrasmissione, come visto nel paragrafo precedente.

Tale caso, però, comporta un ulteriore problema: supponiamo che il client invii un verbale firmato e che il server, dopo averne verificato la validità, lo memorizzi nel database e spedisca il messaggio di risposta firmato. Supponiamo che questo messaggio venga in qualche modo alterato; di conseguenza il client non riconosce la firma del server e ritrasmette il verbale. A questo punto al server giunge una copia del verbale già memorizzato; ovviamente tale copia non deve essere nuovamente archiviata, ma serve semplicemente per far sì che il server rispedisca un altro acknowledge firmato. Ciò implica che il servitore debba avere uno stato dei verbali ricevuti in modo da poter scartare la ricezione di verbali già precedentemente memorizzati. Realizzeremo quindi uno *stateful server*, cioè un server con stato che realizza una semantica *exactly-once*, ossia una semantica in cui ogni messaggio (verbale) è ricevuto una ed una sola volta. Per identificare in modo univoco un verbale, nel floppy disk di ogni docente, per ogni corso è memorizzato un numero progressivo dei verbali firmati, aggiornato ogni volta che un nuovo verbale viene compilato e firmato; in questo modo la tripla (docente, numero di verbale, corso) identifica univocamente ogni verbale firmato.

Naturalmente, in caso di superamento del numero massimo di ritrasmissioni previste, si procede con la memorizzazione dei verbali non trasmessi su floppy del docente.

5.5.4 Problemi dovuti alla rete di interconnessione

Un altro problema da gestire si verifica qualora, per un qualche malfunzionamento della rete di interconnessione, i messaggi di richiesta da parte del cliente o i messaggi di risposta da

parte del server vengono persi, non giungendo mai a destinazione. Per gestire tale situazione si ricorre ad un meccanismo basato sull'utilizzo di *timeout*. Ogni volta che il client spedisce un verbale fa partire un contatore con un opportuno valore iniziale che viene decrementato al passare del tempo. Se il timeout scade prima che giunga il messaggio di risposta del server, allora l'applicazione client provvede a ritrasmetterlo (presumibilmente il verbale è stato perso durante la trasmissione.) Come al solito avremo un numero massimo di ritrasmissioni sopra il quale il verbale non verrà più rispedito ma memorizzato nel floppy del docente. Utilizzando questo meccanismo notiamo che ancora una volta il server deve avere uno stato dei verbali ricevuti; infatti, supponiamo che scada il timeout e che in realtà il messaggio non sia stato perso, ma che a causa di problemi di congestione della rete abbia subito un pesante ritardo nella trasmissione; a questo punto al server giungeranno due verbali identici e pertanto dovrà essere in grado di considerare solo il primo.

Con questa soluzione possiamo realizzare un protocollo di comunicazione affidabile che garantisca che tutti i messaggi giungano a destinazione. L'affidabilità del protocollo può essere già fornita in parte dal livello di trasporto, nel caso in cui si utilizzi un protocollo di per sé affidabile, quale TCP (Transport Control Protocol); se invece si utilizza un protocollo di trasporto inaffidabile, quale UDP (User Datagram Protocol), allora a livello applicativo dovrà essere realizzata l'intera affidabilità della comunicazione, lasciando quindi tale onere al programmatore [COM95].

5.5.5 Fault Tolerance dei sistemi di memorizzazione

Finora abbiamo considerato problemi che possono insorgere a causa di malfunzionamenti ascrivibili alla rete di interconnessione. Resta da considerare come gestire eventuali guasti che possono verificarsi ai dispositivi di memorizzazione: hard disk e floppy disk. Poiché non possiamo assumere di disporre di una memoria immune da guasti garantisca l'inalterabilità delle informazioni memorizzate. Ci si può avvicinare a tale astrazione utilizzando meccanismi di replicazione. Studiare soluzioni per garantire la robustezza del sistema a fronte dei guasti esula dell'obiettivo della nostro progetto; pertanto indicheremo di seguito unicamente le ipotesi su cui si basa il sistema.

Dal lato client abbiamo due memorie di massa, gli hard disk delle varie postazioni client e i floppy disk dei docenti, mentre da lato server abbiamo il dispositivo fisico su cui risiede il database dei verbali. Noi supponiamo tutti questi dispositivi di memorizzazione "stabili", nel senso che supporremo che le informazioni memorizzate siano sempre reperibili e mai danneggiate. Sarà, per esempio, onere del docente farsi una copia di backup del proprio floppy disk oppure dovremo ricorrere per memorizzare il database dei verbali a hard disk di tipo RAID (Redundant Array of Inexpensive Disks) che consentono, mediante la ridondanza dei dati, una maggior robustezza agli eventuali guasti.

5.5.6 Recovery dei verbali non trasmessi

Fatte queste premesse, resta da vedere come realizzare il meccanismo che consenta il recupero (*recovery*) dei verbali che non sono stati ancora archiviati dall'applicazione server nel

database. Questi verbali sono memorizzati nel floppy disk del docente. L'applicazione client obbliga il docente a inserire tale floppy disk, ogni volta che accede per la prima volta al client, e verifica se in tale dischetto ci siano dei verbali che devono essere rispediti. In caso positivo, l'applicazione provvede a spedire tali verbali utilizzando i soliti meccanismi. Va osservato che non è necessaria la verifica della firma del docente per i verbali del floppy disk, in quanto si è supposto che tali verbali non siano stati in alcun modo danneggiati.

I vari client dispongono nel loro hard disk locale di una copia di tutti i verbali non trasmessi. Sarà, pertanto, possibile progettare una procedura che periodicamente rilevi nelle diversi postazioni client quali siano i verbali che non erano stati archiviati e in base a questi interrogare il database per verificare se i docenti siano riusciti in seguito a trasmettere questi verbali. Se la procedura verificherà che il verbale è ora presente nel database potrà cancellare dalla specifica postazione client la copia del verbale.

Capitolo 6. IMPLEMENTAZIONE

Nel capitolo precedente abbiamo descritto le specifiche del progetto. In questo capitolo passiamo alla descrizione dell'implementazione.

6.1 Introduzione

Il processo di costruzione dell'applicazione si è svolto in diverse fasi distinte. In una prima fase ci siamo occupati della progettazione del sistema che realizzava l'infrastruttura a chiave pubblica, cioè abbiamo individuato i servizi PKI di cui la nostra applicazione ha bisogno. Poi, in una seconda fase, sono stati installati e configurati i toolkit aggiuntivi per la gestione delle smart card e per la gestione del servizio di marcatura temporale. Infine, si è passati alla programmazione delle due applicazioni, quella client e quella server. Una volta realizzato il prototipo delle applicazioni si è svolta un'adeguata fase di testing e di analisi delle prestazioni, che sarà oggetto del capitolo successivo.

6.2 La progettazione della PKI

Per la realizzazione dell'infrastruttura a chiave pubblica si è scelto di utilizzare uno specifico prodotto software, Entrust/PKI, fornito da un'azienda canadese, la Entrust Technologies Limited, su piattaforme con sistema operativo Windows¹⁷. Al momento, tale prodotto è indubbiamente uno dei migliori prodotti disponibili sul

¹⁷ Esiste anche una versione di Entrust per sistemi operativi Unix.

mercato, soprattutto per quanto riguarda la completezza dei servizi forniti (si tenga presente, per esempio, che attualmente è uno dei pochi prodotti, se non a volte l'unico, a mettere a disposizione servizi quali il timestamping, la gestione della doppia coppia di chiavi, una per la cifratura, una per la firma, l'aggiornamento automatico delle chiavi, il key recovery). A questo si aggiunge il fatto che l'azienda canadese è stata sicuramente pioniera nella progettazione di soluzioni per sistemi PKI ed è tuttora leader in tale settore. Per un'analisi approfondita di mercato si possono consultare alcuni report quali quello dell'Aberdeen Group [AG99] oppure quello del Burton Group [L97]. In questo paragrafo il nostro intento è solo quello di mostrare sinteticamente come l'utilizzo di tale toolkit ci abbia permesso di realizzare una PKI. Per una descrizione più dettagliata delle funzionalità del programma si rimanda all'appendice, oppure direttamente ai manuali del programma [ENT98], [EI98].

Entrust/PKI versione 4.0 è costituito da quattro applicativi software, distinti da un punto di vista funzionale:

- Entrust/Authority
- Entrust/Admin
- Entrust/Directory
- Entrust/Entelligence

Nel seguito considereremo solo Entrust/Admin che è il componente che fornisce un'interfaccia grafica per l'utilizzo di Entrust/PKI e ci ha permesso di implementare una PKI pilota da utilizzare all'interno dell'organizzazione universitaria per sperimentare la nostra applicazione.

Con questo componente software, una volta che è stato inizializzato e configurato il sistema, è possibile creare dei nuovi utenti di Entrust (le *end-entity* secondo gli standard PKI.) Nel nostro caso si tratterà di creare dei nuovi docenti che, ovviamente,

saranno stati precedentemente identificati da un'opportuna Autorità di Registrazione. A questo punto la creazione del un nuovo utente passa attraverso le seguenti tre fasi:

1. Aggiunta dell'utente alla Directory
2. Abilitazione dell'utente
3. Distribuzione sicura delle informazioni per l'inizializzazione dell'utente

6.2.1 Aggiunta dell'utente alla Directory

Con la prima fase si crea la *Directory entry* dell'utente; per far ciò si compila un'apposita form (si veda figura 6.1) ove si inseriscono i dati anagrafici dell'utente (nome e cognome), un codice che identifica univocamente l'utente all'interno dell'organizzazione (*serial number*) e l'indirizzo di posta elettronica qualora si voglia che venga anch'esso pubblicato nel certificato.

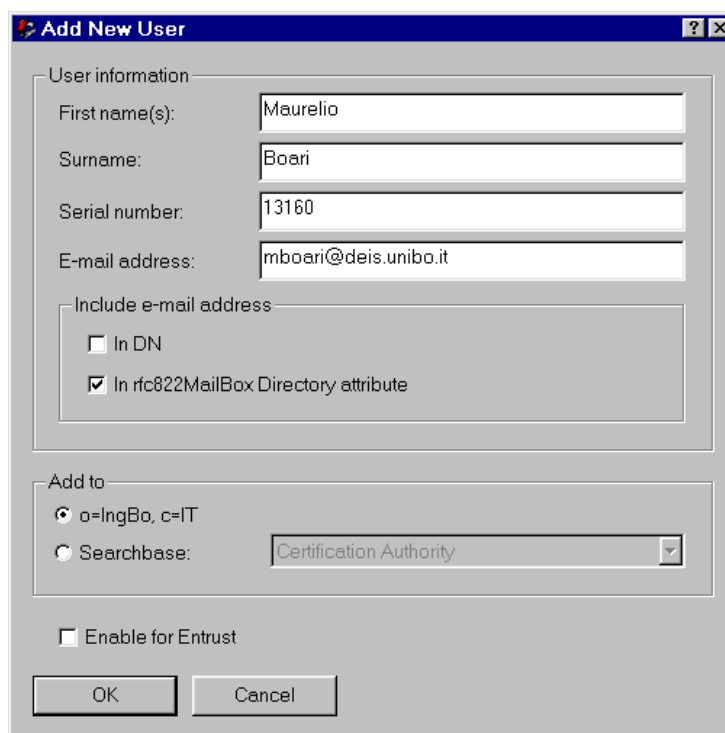


Figura 6.1: Form per aggiungere un nuovo utente in Entrust

A questo punto l'utente è stato registrato presso l'Autorità di Certificazione¹⁸ e, non appena saranno pubblicati i suoi certificati, sarà presente nella Directory con un suo Distinguished Name (DN)¹⁹, costituito dal serial number (SN), dal Common Name (CN) e il nome della CA. Nell'esempio della figura il DN dell'utente risulta il seguente:

"serialNumber = 13160 + cn = Maurolio Boari, o = IngBo, c = IT"

6.2.2 Abilitazione dell'utente

Una volta creato l'utente nella directory, si può passare alla seconda fase che consiste nell'abilitare l'utente, cioè nel definire la politica di gestione delle chiavi (per esempio determinare il loro tempo di vita, *lifetime*), la politica di gestione dei certificati e più in generale definire altre informazioni sulla gestione dell'utente appena creato. È in questa fase che vengono anche decise le estensioni da associare ai certificati secondo lo standard X.509v3, estensioni che nel nostro caso ci permettono di distinguere quali utenti sono docenti e quindi sono autorizzati a firmare i verbali degli esami.

Per far questo Entrust ricorre agli Object Identifiers (OIDs) che forniscono, in generale, un meccanismo standard per identificare i vari "oggetti". Nel nostro caso specifico gli "oggetti" sono le diverse qualifiche che gli utenti hanno nell'organizzazione e, pubblicando tali OIDs come estensioni nei certificati, è possibile gestire le diverse tipologie di utenti (si parla di *certificate policies*.)

¹⁸ In questa sperimentazione la nostra Certificate Authority è la Facoltà di Ingegneria di Bologna che ha come Distinguished Name (DN) o=IngBo, c=IT.

¹⁹ Col termine di Distinguished Name si indica il nome completo con il quale un utente compare nella Directory e tramite il quale viene identificato.

I vari OIDs sono inquadrati all'interno di una struttura gerarchica a forma d'albero (OID tree). La connessione tra due qualsiasi nodi è rappresentata da un numero intero e l'OID di un oggetto è dato dalla concatenazione di questi numeri dalla radice dell'albero fino all'oggetto che interessa. Una tale struttura assicura che ogni "oggetto" abbia un identificatore unico e non ambiguo.

Nella figura 6.2 si può vedere un esempio di albero OID che è stato utilizzato nella nostra applicazione. Con OID 0.0.10 si identifica l'utente che abilita la postazione server per la ricezione e convalida dei verbali; con OID 0.2 si identificano tutti i docenti, mentre con OID 0.2.2 si identificano i docenti di seconda fascia.

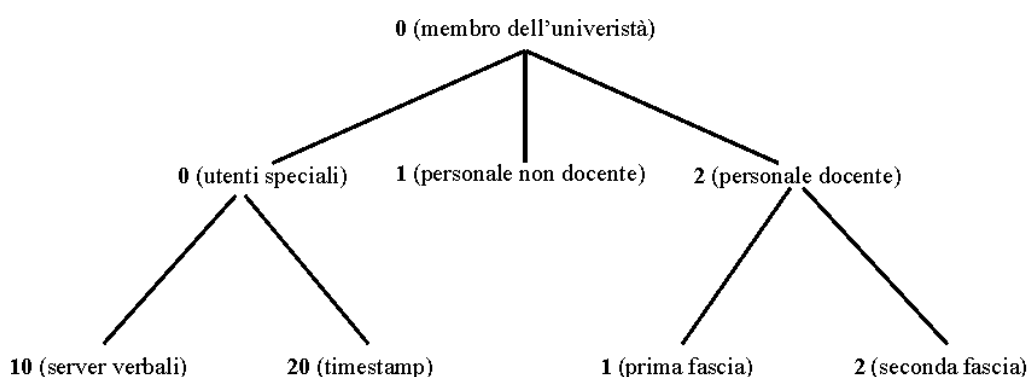
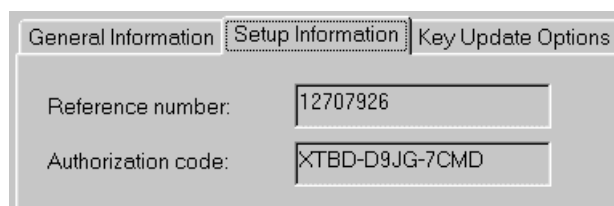


Figura 6.2: Esempio di OID tree per l'Ateneo di Bologna

6.2.3 Inizializzazione dell'utente

Dopo aver creato e abilitato un utente, la CA genera due segreti iniziali che vengono consegnati al docente e che gli consentono di terminare la fase di inizializzazione e di diventare a tutti gli effetti operativi. Entrust/Admin genera automaticamente questi due codici (Reference Number e Authorization Code) che

servono per creare una connessione sicura tra il client e la PKI attraverso un protocollo proprietario di Entrust chiamato SEP (Secure Exchange Protocol). Tramite questa connessione, utilizzando l'applicazione client, l'utente può generare un suo file di profilo (Entrust Profile) che contiene la chiave privata di firma, la chiave privata di cifratura, i certificati di verifica e di cifratura, il certificato dell'Autorità di Certificazione e altre informazioni quali ad esempio gli algoritmi utilizzati. Il file di profilo è protetto da una password scelta dall'utente ed è cifrato con un algoritmo a chiave simmetrica (di solito con un CAST-40 o con un CAST-128 a seconda che si usi crittografia debole o forte²⁰.)



The image shows a screenshot of a software interface with three tabs: 'General Information', 'Setup Information', and 'Key Update Options'. The 'Setup Information' tab is active. It contains two input fields: 'Reference number:' with the value '12707926' and 'Authorization code:' with the value 'XTBD-D9JG-7CMD'.

Figura 6.3: Esempio di Reference Number e di Authorization Code

L'intera operazione di creazione del profilo è completamente trasparente all'utente e dà come risultato finale la memorizzazione del profilo nella smart card. Per maggior dettagli si veda in appendice; qui va solo ricordato che Entrust utilizza meccanismi diversi per generare le due coppie di chiavi e solo la coppia di chiavi di firma viene effettivamente creata all'interno della smart card, mentre la coppia di chiavi di cifratura viene creata dalla PKI e poi trasferita sulla carta. Il motivo di tale scelta è riconducibile alla

²⁰ In virtù delle norme americane che governano le regole di esportazione riguardo la crittografia, prodotti che utilizzano crittografia forte (es: CAST-128) non possono essere esportati fuori dagli Stati Uniti e dal Canada. Pertanto esistono due varianti di un medesimo prodotto: la versione "domestica" per utilizzi all'interno degli Stati Uniti e del Canada e la versione "internazionale" che supportando solo crittografia debole può essere esportata anche in Europa.

diversa gestione che necessitano le due coppie di chiavi, come già spiegato nel capitolo secondo.

6.3 Il servizio di marcatura temporale (timestamping)

6.3.1 Introduzione al timestamping

Tramite l'utilizzo della firma digitale si è in grado di garantire l'integrità, l'autenticità e il non-ripudio di un documento informatico. Spesso può anche essere necessario disporre di un meccanismo che dia una validazione temporale del documento, ossia disporre di un qualcosa che dia un riferimento temporale certo e sicuro. Lo stesso governo italiano nelle regole tecniche dell'AIPA ha introdotto il concetto di marca temporale e di sistema di validazione temporale.

La marca temporale, detta anche timestamp, serve per certificare la data di creazione o di ultima modifica di un documento. In tal senso ha una duplice funzione: una funzione che ha lo scopo di collocare temporalmente un documento, qualora la validità del documento dipenda anche dalla data in cui il documento viene creato e firmato; una seconda funzione che consente di verificare l'effettiva validità della chiave privata utilizzata per firmare un documento. Questa esigenza nasce dal fatto che, per svariati motivi, la chiave privata di firma di un utente potrebbe essere compromessa. Per esempio, nella nostra applicazione, se un docente sventuratamente perde la propria smart card, comunicandolo all'amministratore della Directory, potrà farsi tempestivamente revocare il certificato. Utilizzando un sistema di validazione temporale è possibile sapere quando i verbali sono stati compilati e quindi nell'esempio precedente tutti i verbali che

arrechino una marca temporale successiva allo smarrimento della smart card non saranno ritenuti validi.

Va precisato che nella nostra applicazione la marca temporale corrisponde alla data e l'ora in cui i verbali vengono spediti al server che ha il compito di verificarne l'autenticità e non con la data in cui viene effettivamente svolta la verbalizzazione dell'esame. Questa scelta è stata fatta per permettere al docente di compilare il verbale anche quando per qualche motivo non abbia disponibilità di accedere alla rete (si parla di modalità di lavoro *offline*.) Infatti, come si vedrà in seguito, per accedere a un servizio di timestamping è necessario poter lavorare col supporto della rete. Così facendo, si svincola la fase di compilazione del verbale dalla fase di trasmissione.

Nel paragrafo successivo vedremo come è stato realizzato un servizio di validazione temporale nell'applicazione di verbalizzazione degli esami; per un'analisi più approfondita degli standard sul timestamping e i suoi protocolli si rimanda alle pubblicazioni del gruppo Public Key Infrastructure X.509 (PKIX) dell'Internet Engineering Task Force (IETF) che sta sviluppando il protocollo denominato Time Stamp Protocol (TSP) [ACPZ99] e alle pubblicazioni del gruppo europeo European Trusted Service (ETS) che con il progetto Public Key Infrastructure with Time-Stamping Authority (PKITS) ha tracciato delle linee fondamentali per lo sviluppo di un'autorità per la marcatura temporale di documenti digitali all'interno di infrastrutture a chiave pubblica [FNMT98]. Si tratta, in entrambi i casi, di progetti in via di sviluppo e non di protocolli definitivi, in quanto la materia risulta di stretta attualità e, pertanto, soggetta a possibili evoluzioni future.

6.3.2 L'implementazione della marcatura temporale con Entrust/Timestamp

Per realizzare il servizio di marcatura temporale, nella nostra applicazione ci siamo serviti di un apposito prodotto software sempre fornito dalla Entrust Technologies Limited: Entrust/Timestamp [ETS98]. Questo programma deve essere installato in un elaboratore sicuro (Timestamp Server), separato dal resto del sistema PKI, con l'obiettivo di poter sempre garantire la veridicità della data e dell'ora di tale macchina.

L'installazione e l'esecuzione del servizio di timestamping è consentito solo a uno speciale utente Entrust, il Timestamp Server Agent (TSA). Questo utente, al pari degli altri utenti della PKI, dispone di un suo profilo contenente le sue coppie di chiavi e i suoi certificati. Con tale profilo può mettere in esecuzione il servizio di marcatura temporale (come *services* in sistemi Win32 oppure come demoni in sistemi Unix). Una volta attivato, il TSA deve fornire al sistema PKI un file di configurazione ove ci sia scritto l'indirizzo IP (o, in alternativa, il DNS name) della macchina su cui è installato Entrust/Timestamp e la porta TCP dedicata all'ascolto di tale servizio. Tramite questo file di configurazione i vari client, collegandosi alla Directory, possono ricevere le informazioni sull'indirizzo del servizio di timestamping.

Va osservato che il server che fornisce il timestamping può facilmente diventare un collo di bottiglia in cui tutti i client devono passare e pertanto è possibile distribuire tale servizio su più elaboratori. Così facendo si ottiene una riduzione del carico di lavoro del Timestamp Server, rendendo inoltre il servizio ridondante e quindi più robusto ai guasti. (Sarà a carico del programmatore realizzare un servizio di sincronizzazione tra i diversi orologi.)

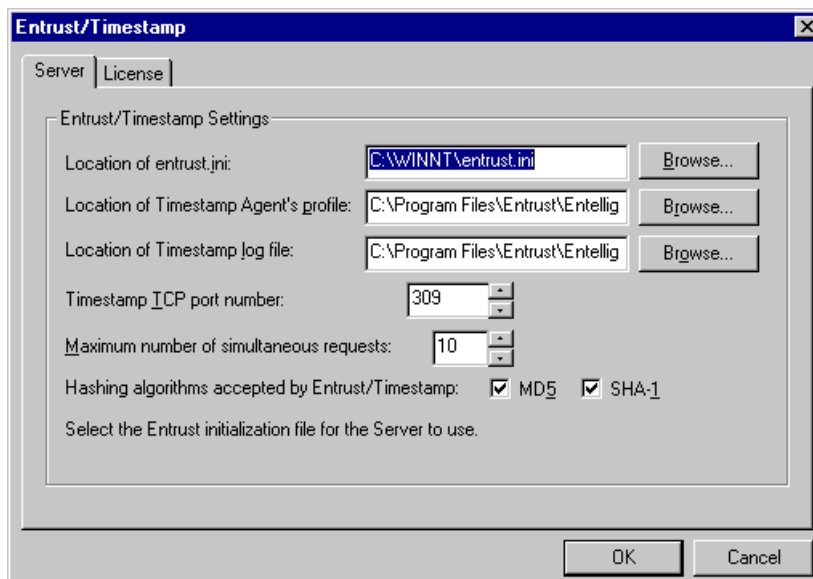


Figura 6.4: Schermata di configurazione di Entrust/Timestamp

Un'applicazione, quando vuole apporre una marca temporale su un documento informatico, ne calcola prima l'hash, poi trasmette tale hash al Timestamp Server. La data e l'ora di sistema dell'elaboratore che ospita tale servizio vengono aggiunti all'hash e il tutto viene firmato digitalmente con la chiave privata di firma del TSA. Tale operazione crea quella che tecnicamente si chiama *time-stamp token*, che viene restituito all'applicazione richiedente. A questo punto al documento è stata aggiunta una marca temporale che chiunque potrà verificare utilizzando il certificato di verifica del TSA. La combinazione dell'hash con il *time-stamp token* garantisce sia l'integrità del documento sia l'autenticità della data.

Se il documento oggetto di timestamping è stato precedentemente firmato (come nel caso del verbale d'esame), allora si può utilizzare la marca temporale anche per verificare la validità temporale della firma elettronica: se il certificato di verifica dell'utente (docente) è stato revocato prima della data di firma, la firma non è ritenuta valida; se invece il certificato di verifica non è

stato revocato o è stato revocato dopo la data di firma, allora la firma è ritenuta valida.

A tal riguardo va precisato che Entrust gestisce la revoca di un certificato nel caso in cui la chiave privata sia stata compromessa (*key compromise*²¹). Entrust consente il cosiddetto *back-dating*, ossia l'amministratore della Directory specifica come data di revoca del certificato l'ultima data in cui si ha la certezza che la chiave non fosse compromessa. In tal modo si pone un rimedio a quei problemi che si potrebbero verificare a causa del ritardo che intercorre tra quando si verifica la compromissione della chiave e quando l'amministratore riceve tale notifica: un documento che presenta un timestamp valido ma con una data successiva alla data di revoca viene comunque invalidato.

Riassumendo, nel processo di verifica della firma di un documento si possono avere i seguenti casi:

1. Se il certificato di verifica associato alla firma del documento è valido, allora la verifica della firma avverrà indipendentemente dal fatto che il documento abbia o no una marca temporale.
2. Se il certificato di verifica è invece scaduto, allora la firma del documento non sarà ritenuta valida indipendentemente dal timestamping.
3. Se il certificato di verifica è stato revocato:
 - a) se il documento non ha un timestamp, allora la firma viene ritenuta non valida non potendo disporre di un riferimento temporale sicuro.
 - b) se la data di revoca è anteriore alla data del timestamp, allora la firma è ritenuta non valida, in quanto il documento è stato firmato con una chiave che era già stata compromessa.

²¹ Entrust prevede vari motivi per i quali un certificato possa essere revocato. Oltre al *key compromise*, un certificato può essere revocato perché l'utente non fa più parte dell'organizzazione (*affiliation change*), perché la coppia di chiavi è stata rimpiazzata con una

- c) se la data di revoca è posteriore alla data di timestamp, allora la firma è ritenuta valida, in quanto il documento è stato firmato con una chiave che ancora non era stata compromessa.

6.4 Smart Card

Attualmente Entrust supporta solo lo standard PKCS #11 del RSA Laboratories. Inoltre, Entrust ha un suo modo proprietario di memorizzare i certificati e le chiavi (i file di profilo) e quindi è necessario disporre di smart card che supportino tali strutture. Per questo la nostra scelta si è orientata verso le carte SignaSure Cip dell'azienda americana Datakey²² [DK98] [T97].

Le carte da noi utilizzate sono le Model 310 Smart Card Token con lettore seriale Model 10XC. Hanno 256 bit di RAM, 8 KB di EEPROM e 14 KB di ROM. Dispongono di un processore a 8 bit (standard 8051) e di un coprocessore aritmetico a 540 bit, necessario per poter eseguire con una certa velocità operazioni di crittografia (firma, cifratura, generazione chiavi.) Dispongono della *key generation* (RSA e DSA) e possono eseguire la firma all'interno della carta (RSA a 512-1024 bit o DSA a 512 bit). Possono inoltre cifrare e decifrare documenti con l'algoritmo simmetrico DES; sono protette all'esterno da una password e all'interno dal cosiddetto "*silicon firewall*", ossia la stessa struttura hardware della carta fa sì che tutte le operazioni che avvengono al suo interno siano protette da attacchi esterni.

nuova (*superseded*), perché il certificato contiene informazioni non più attuali (*cessation of operation*.)

²² Tra gli altri produttori di smart card supportati da Entrust segnaliamo ActivCard, Schlumberger e Hewlett Packard.

6.5 EntrustFile Toolkit

6.5.1 La scelta del toolkit di sviluppo

Le principali funzioni di sicurezza di Entrust quali ad esempio la firma digitale e la cifratura sono fornite da un dedicato e sicuro supporto chiamato Entrust Engine. L'EntrustFile Toolkit fornisce delle application programming interface (API) che permettono di interfacciarsi con il software che costituisce Entrust Engine; in tal modo è possibile integrare nelle proprie applicazioni le funzioni di crittografia dell'Entrust Engine.

Esistono tre tipi di toolkit a seconda che le API siano state scritte in C, in C++ o in Java. Poiché nella progettazione dell'applicazione si è scelto di adottare una metodologia orientata agli oggetti la scelta era tra il linguaggio C++ e il linguaggio Java, entrambi linguaggi *object oriented*. Il toolkit Java non supporta però le smart card, a differenza del toolkit in C++ che supporta qualsiasi hardware token che soddisfi lo standard PKCS #11. Avendo deciso di adottare le smart card come dispositivo di memorizzazione delle chiavi private per soddisfare i requisiti dell'AIPA, la nostra scelta è stata quindi a favore del toolkit in C++.

6.5.2 Le funzionalità di EntrustFile

Vediamo in sintesi quali sono le principali funzionalità che il toolkit mette a disposizione, mentre per i dettagli delle librerie si rimanda all'appendice.

Logon autenticato

Ogni utente che vuole utilizzare una qualsiasi operazione di crittografia deve prima essere autenticato; l'autenticazione può essere fatta in due modalità: in modalità *offline* l'autenticazione consiste solo nel disporre del file di profilo (contenuto nella smart card) e nella verifica della password di tale profilo, in modalità *online*, oltre alla verifica della password, l'utente viene autenticato dalla Directory, che verifica l'appartenenza e che i suoi certificati non siano stati revocati.

Cifratura e decifratura

Si può cifrare un qualsiasi tipo di file per una serie di utenti destinatari dei quali si dispone della loro chiave pubblica di cifratura. Tale chiave si trova nel corrispondente certificato elettronico che può essere fornito direttamente dallo stesso utente destinatario oppure può essere reperito dalla Directory se l'utente per cui si vuole cifrare appartiene alla stessa CA di colui che cifra o ad una cross-certified CA.

Firma digitale

Si può firmare digitalmente un qualsiasi tipo di file; nel processo di verifica della firma sarà necessario disporre del certificato di verifica di colui che ha firmato il file. Se il destinatario non dispone già di tale certificato, si potrà includerlo nel file firmato²³. Si osservi che nel caso in cui si verifichi un file firmato da un membro appartenente ad un'altra CA, sarà necessario disporre anche del certificato di verifica di tale CA per poter verificare l'integrità del certificato di verifica del firmatario.

²³ Va osservato che Entrust ha fatto la scelta di pubblicare nella Directory solo i certificati di cifratura e non anche quelli di firma.

Meccanismi di sicurezza

Il toolkit supporta tre meccanismi di sicurezza:

- PKCS #7 della RSA Laboratories [PKCS7], che il meccanismo che sta alla base dello standard S/MIME [RFC2311].
- PEM (Privacy Enhanced Mail), uno standard internet [RFC1421].
- Entrust archive, meccanismo proprietario per la protezione dei dati.

Supporto per smart card

Tutte le funzioni viste precedentemente possono essere svolte sia dalla CPU del computer agendo su un profilo memorizzato in una memoria secondaria locale (floppy disk o disco rigido), sia dal processore della smart card che agisce sul profilo contenuto all'interno.

6.6 Realizzazione dei meccanismi di sicurezza: la classe UtenteEntrust

Grazie alle funzioni messe a disposizione dal toolkit abbiamo a disposizione tutto ciò che ci occorre per potersi interfacciare con la PKI di Entrust. Con le API del toolkit abbiamo progettato una classe, chiamata UtenteEntrust, che ci fornisce tutti i metodi necessari per poi poter implementare le due applicazioni del nostro progetto (la versione client e la versione server.) Di seguito descriveremo quali sono i principali metodi di questa classe.

Per istanziare un oggetto di classe UtenteEntrust bisogna fornire come parametro di ingresso il file di configurazione di Entrust (*entrust.ini*) che contiene le informazioni di configurazione necessarie ad ogni utente per poter dialogare con la PKI. In

particolare nel file ci sono gli indirizzi IP e i relativi numeri di porta per poter accedere alla CA (Entrust/Authority) e alla directory tramite il protocollo d'accesso LDAP (Lightweight Directory Access Protocol) [RFC1777] e gli algoritmi utilizzati per la firma e per la cifratura.

```
[Entrust Settings]
Manager=137.204.61.133+709
Server=137.204.61.133+389
EncryptWith=Cast
SigningKey=RSA
```

Figura 6.5: Estratto del file entrust.ini

A questo punto se dobbiamo generare un nuovo utente dobbiamo invocare il metodo *CreaProfilo()*, dandogli in ingresso la coppia Reference Number - Authorization Code, la password scelta dall'utente, il nome del file di profilo e indicare se l'utente possiede la smart card.

Se l'utente esiste già allora può invocare il metodo *LogonProfilo()* che consente all'utente di autenticarsi fornendo la password del suo profilo. Come parametro di ingresso di questo metodo bisogna anche specificare la modalità di lavoro, *online* o *offline*. Se si decide di lavorare *online*, il processo di autenticazione, oltre a verificare la correttezza della password, verificherà, accedendo alla Certificate Revocation List, se il certificato non è stato revocato. Se, invece, si decide di lavorare in modalità *offline* (ad esempio, non si dispone in quel momento di collegamento alla rete), allora il processo di autenticazione verificherà solo la correttezza della password del profilo; inoltre non si potrà accedere né al servizio di directory, né a quello di timestamping.

Prima di fare il logon è possibile verificare se è presente una smart card, chiamando il metodo *TokenHardwareInfo()*. Tale metodo rileva se è inserita una smart card nell'apposito lettore e in caso positivo estrae dalla carta il nome del profilo. (Potrebbe essere presenti più profili nella carta, in tal caso viene data la possibilità di scelta.) Se, invece, non viene rilevata nessuna carta, allora invocando il metodo *SetProfilo()* si ha la possibilità di specificare la locazione del profilo su il dispositivo di massa scelto (floppy disk o hard disk.)

Col metodo *FirmaFile()* si ha la possibilità di firmare digitalmente un file utilizzando RSA come algoritmo di firma e SHA-1 come funzione hash (sono due algoritmi specificati nell'allegato tecnico dell'AIPA e al contempo supportati da Entrust.) Il risultato di questa operazione è un file firmato secondo lo standard S/MIME [RFC2311] (standard per la posta elettronica sicura.) A questo file firmato si può includere il proprio certificato di verifica, necessario nel caso in cui il ricevente del file firmato non l'abbia, e anche il certificato della Autorità di Certificazione, necessario per verificare la validità del certificato di verifica, nel caso in cui il ricevente appartenga a una CA differente. A questo proposito va osservato che l'AIPA impone che alla firma digitale sia sempre allegato il certificato di verifica corrispondente (art. 9 comma 2); pertanto, per applicazioni in cui è richiesto che i documenti informatici abbiano una validità legale, dovremmo sempre allegare il certificato al messaggio.

Col metodo *VerificaFirma()* è invece possibile verificare la firma di un file e ottenere il file in chiaro, sempre secondo lo standard S/MIME. La verifica della firma può essere fatta utilizzando il certificato di verifica del mittente, se incluso nel messaggio firmato, oppure accedendo a un personale indirizzario in cui sia contenuto il certificato di colui che ha spedito il file. Si tenga presente che non è possibile reperire dalla Directory il

certificato di verifica in quanto Entrust ha fatto la scelta di pubblicare solo i certificati di cifratura²⁴. Successivamente col metodo *ottieniFirmatario()* è possibile ottenere il nome di chi ha firmato il file.

Abbiamo poi dei metodi che ci permettono di marcare temporaneamente un file. Con *MettiTimeStamping()* possiamo appendere ad un file il timestamp fornito dal server dedicato a tale compito, mentre col metodo *VerificaTimeStamping()* possiamo estrarre dal file il timestamp e verificarne la validità. Infine, invocando *dataTimeStamping()* possiamo ottenere la data autenticata dal servizio di marcatura temporale.

La classe *UtenteEntrust* mette a disposizione dei metodi per accedere ai certificati contenuti nel profilo dell'utente. In particolare, con *Certificati()* possiamo estrarre dal profilo i certificati di verifica e di cifratura. Con *TipoDiChiave()* possiamo determinare il tipo di chiave pubblica contenuto nel certificato elettronico (chiave di firma, chiave di cifratura, chiave simmetrica, chiave di firma della CRL, ecc.) Con *MostraCertificati()* possiamo visualizzare gli attributi di un certificato secondo lo standard X.509. Con *EstensioniX509()* possiamo sapere quante e quali sono le estensioni (OIDs) contenute in un certificato.

Infine sottolineiamo l'importanza di un metodo per la cancellazione sicura di un file: *CancellaFile()*. Tipicamente le primitive di sistema per la cancellazione di un file marcano solo tale area di memoria come area riutilizzabile, senza però effettuare un'effettiva cancellazione fisica. Questo comporta la possibilità di recupero di tale file. Mediante l'operazione *CancellaFile()* si distrugge in maniera irreversibile l'area di memoria dove era contenuto il file.

²⁴ Va osservato che è possibile utilizzare sistemi di Directory diversi da quello fornito da Entrust. La stessa Entrust si qualifica più come produttori di CA che Directory tanto che Entrust/Directory è solo per organizzazione piuttosto piccole (massimo 5000 utenti.)

Di seguito riportiamo sinteticamente gli altri metodi:

- *CifraFile()*: esegue l'operazione di cifratura di un file per un insieme di destinatari.
- *DecifraFile()*: esegue l'operazione di decifratura di un file.
- *FirmaECifraFile()*: esegue contemporaneamente l'operazione di cifratura e di firma di un file.
- *VerificaEDecifraFile()*: verifica la firma di un file e poi ne decifra il contenuto.
- *NomeUtente()*: restituisce il nome dell'utente
- *NomeCA()*: restituisce il nome della Certification Authority.
- *MostraProfilo()*: mostra le informazioni di un file di profilo.
- *ModificaPercorsoSM()*: modifica il percorso (path) dove vengono archiviati i vari file di informazione dell'utente (rubrica personale, cache dei certificati e della CRL, ecc.)
- *PasswordValida()*: verifica che la password scelta dall'utente soddisfi le regole imposte da Entrust.
- *MostraAlgoritmiSimmetrici()*: mostra gli algoritmi simmetrici supportati per cifratura.
- *OttieniDataLocaleFirma()*: dà l'ora locale di firma di un documento, ossia quello che risulta all'elaboratore in cui l'utente ha firmato il file (non è quella del timestamping.)
- *OttieniUltimoErrore()*: fornisce l'ultimo errore verificatosi dovuto all'invocazione di metodi della classe UtenteEntrust.
- *DurataOperazione()*: fornisce la durata di un'operazione in secondi e millisecondi (utile per misurare le performance dell'applicazione.)

6.7 Le classi per la comunicazione

Per la comunicazione tra applicazione client, applicazione server, Directory e CA abbiamo scelto di utilizzare il protocollo TCP/IP. In particolare, utilizzando come protocollo di trasporto il TCP ci siamo liberati dal dover gestire in fase di programmazione la consegna affidabile dei messaggi, in quanto TCP ha un suo meccanismo di acknowledge e ritrasmissioni che garantisce l'affidabilità della trasmissione [COM95].

Abbiamo così realizzato due classi: la classe Socket e la classe ServerSocket. Queste due classi sono semplici estensioni delle socket stream BSD (versione per sistemi operativi Win32); in particolare, la prima classe consente di creare una socket di comunicazione per il client che si vuole connettere ad un server, mentre la seconda consente di creare, da parte del server, una socket d'ascolto per accettare nuove connessioni. Inoltre queste classi sono state arricchite con alcuni metodi per eseguire operazioni di trasmissioni e ricezione dei verbali e dei relativi messaggio di acknowledge.

6.8 L'applicazione Client

Utilizzando la classe UtenteEntrust per le operazioni di sicurezza e le classi Socket e ServerSocket per la comunicazione, siamo passati alla programmazione delle due applicazioni.

L'applicazione client consente al docente che la esegue due funzioni: l'inizializzazione e la compilazione di verbali d'esame. L'inizializzazione del docente avviene semplicemente inserendo la smart card nell'apposito lettore e fornendo all'applicazione la coppia Reference Number - Authorization Code e scegliendo una password. A questo punto l'applicazione comunica con la PKI e

genera all'interno della carta il file di profilo di Entrust contenente le due coppie di chiavi e i relativi certificati elettronici.

La fase di compilazione del verbale, invece, è un po' più articolata. Prima di tutto, bisogna effettuare il logon: viene prima verificata la presenza di una smart card (smart card detection), poi si accede al profilo contenuto nella carta, si inserisce la password associata al profilo e infine si verificano le estensioni X.509v3 che indicano se tale profilo appartiene ad un docente. In caso affermativo, il logon è ritenuto valido e si può procedere alla compilazione del verbale.

L'applicazione legge dal floppy disk del docente il suo file personale dove sono contenute informazioni che facilitano la compilazione del verbale (codice docente, codice facoltà e corso di laurea, codice esame, nome esame, numero progressivo di verbale.) Poi il docente inserisce i restanti dati necessari (numero di matricola del docente, voto, domande sottoposte al candidato).

Una volta che il verbale in chiaro è stato preparato, il docente deve dare il consenso all'applicazione per poterlo firmare cosicché la smart card possa iniziare il processo di firma.

Un verbale d'esame ha bisogno della firma di due docenti (il presidente di commissione e un membro della commissione), quindi l'applicazione fa il logoff del docente che ha compilato il verbale e fa il logon del secondo docente, che, dopo aver inserito la sua smart card ed essere stato autenticato, procederà a firmare il verbale. Va osservato che in questa fase di sperimentazione abbiamo supposto che le due firme avvengano sequenzialmente, una dopo l'altra.

A questo punto il verbale recante le firme dei due docenti è pronto per essere spedito. Il docente responsabile della compilazione effettua nuovamente il logon (l'applicazione verificherà che sia effettivamente il docente che aveva precedentemente compilato il verbale) e dà l'autorizzazione a

spedire il verbale. Il verbale viene prima inviato al server che gestisce il servizio di marcatura temporale, il quale appone sul verbale la data autenticata di trasmissione e restituisce il verbale all'applicazione. Poi l'applicazione client si connette con l'applicazione server responsabile della ricezione dei verbali, gli trasmette il verbale e si pone in ricezione di una risposta (l'acknowledge firmato dal server).

Ricevuta la risposta, ne verifica l'autenticità della firma; se la firma non risulta valida, si procede con la ritrasmissione del verbale (non si ha la certezza che il verbale sia stato ricevuto dal server), altrimenti si legge il contenuto del messaggio di risposta. Se il messaggio conferma che il server ha ritenuto valido il verbale e ha provveduto a memorizzarlo, allora l'applicazione cancella il verbale che aveva creato temporaneamente e incrementa il numero di verbale, aggiornando tale informazione sul file personale del docente ("positive acknowledge").

Se, invece, il server risponde con un "negative acknowlege", allora, leggendone il contenuto si può capirne la causa:

- È fallito il processo di verifica delle firme del verbale, ossia una delle due firme del docente o la firma del timestamp server è stata ritenuta non valida.
- È fallita la verifica dell'identità del docente: la firma del verbale, pur essendo valida, corrisponde ad un docente diverso da quello che risulta nel verbale.
- È fallita la verifica delle capability del docente, ossia il docente sta compilando un verbale per un esame di cui non è autorizzato a verbalizzare.

Nel primo caso si procede alla ritrasmissione del verbale in quanto il verbale ha subito una qualche variazione durante la trasmissione. Negli altri due casi, invece, è successo che, pur essendo valide le firme del verbale, non risulta valido il contenuto

del verbale: l'applicazione segnala il fatto al docente che dovrà provvedere a controllare il suo file personale. Il verbale verrà memorizzato temporaneamente sul floppy disk del docente.

Nel caso in cui si superi il numero massimo di ritrasmissioni consentite, qualunque sia stata la causa, si procede all'archiviazione del verbale su floppy disk del docente, in quanto il verbale non è stato memorizzato dal server.

Riportiamo di seguito i diagrammi di flusso del comportamento dell'applicazione client.

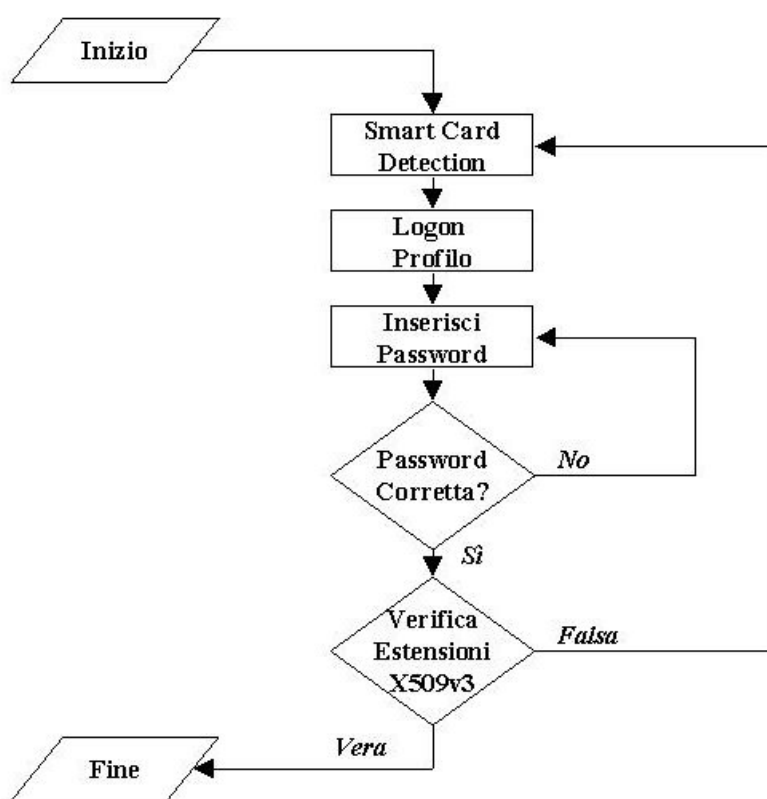


Figura 6.6: Diagramma di flusso del logon di un docente

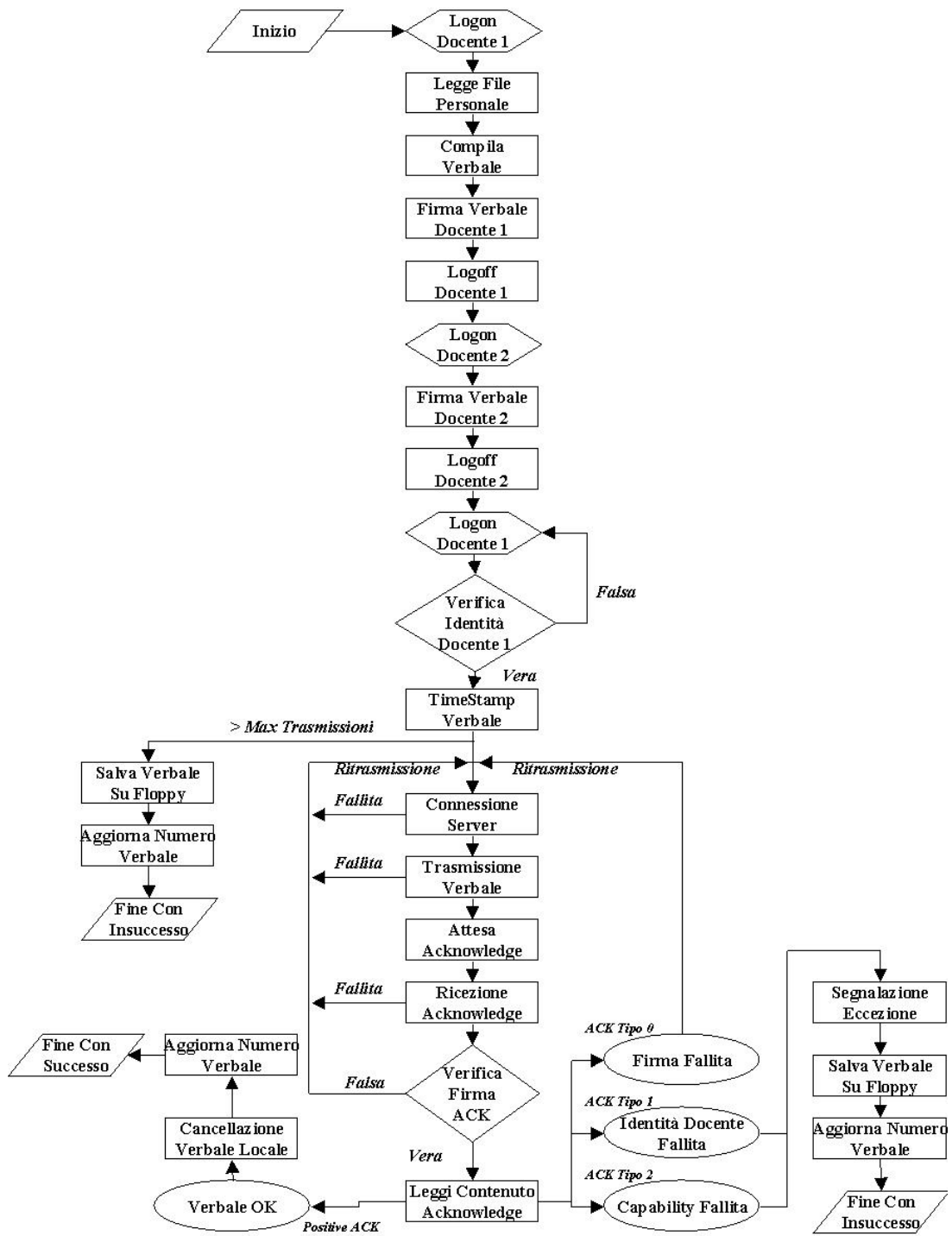


Figura 6.7: Diagramma di flusso dell'applicazione client

6.9 L'applicazione Server

L'applicazione server ha il compito di ricevere i verbali, verificarne le firme, controllarne il contenuto, memorizzarli e spedire i messaggi di risposta alle applicazioni client. Tutte queste operazioni vengono svolte da utente Entrust "speciale" (OID=0.0.10). Questo utente, in maniera analoga a quanto fanno i docenti, deve effettuare un logon. Una volta autenticato, crea una coda d'ascolto e si mette in attesa di connessioni.

Quando riceve una nuova connessione, si prepara alla ricezione di un verbale. Se il verbale non era mai stato ricevuto, allora procede alla verifica delle firme, quella del timestamp server e quelle dei due docenti. Se tutte e tre le firme sono valide, legge il contenuto del verbale e verifica l'identità del docente e le sue capability. Se tutte le verifiche vanno a buon esito, memorizza il verbale e prepara un messaggio di conferma per il client ("positive acknowledge"). Altrimenti prepara un "negative acknowledge" con il motivo del fallimento (tipo 0, 1 o 2). Spedisce l'acknowledge e si mette in attesa di una nuova connessione²⁵.

Se il verbale era già stato ricevuto, allora il server rispedisce subito un messaggio di "positive acknowledge": questo è il caso in cui al client non è arrivata integra la risposta di conferma e quindi ha rispedito il verbale (*stateful server*).

²⁵ In realtà, il server è concorrente, quindi alla ricezione di una connessione fa nascere un nuovo processo che la gestisce, mentre il processo padre si rimette subito in attesa di nuove connessioni.

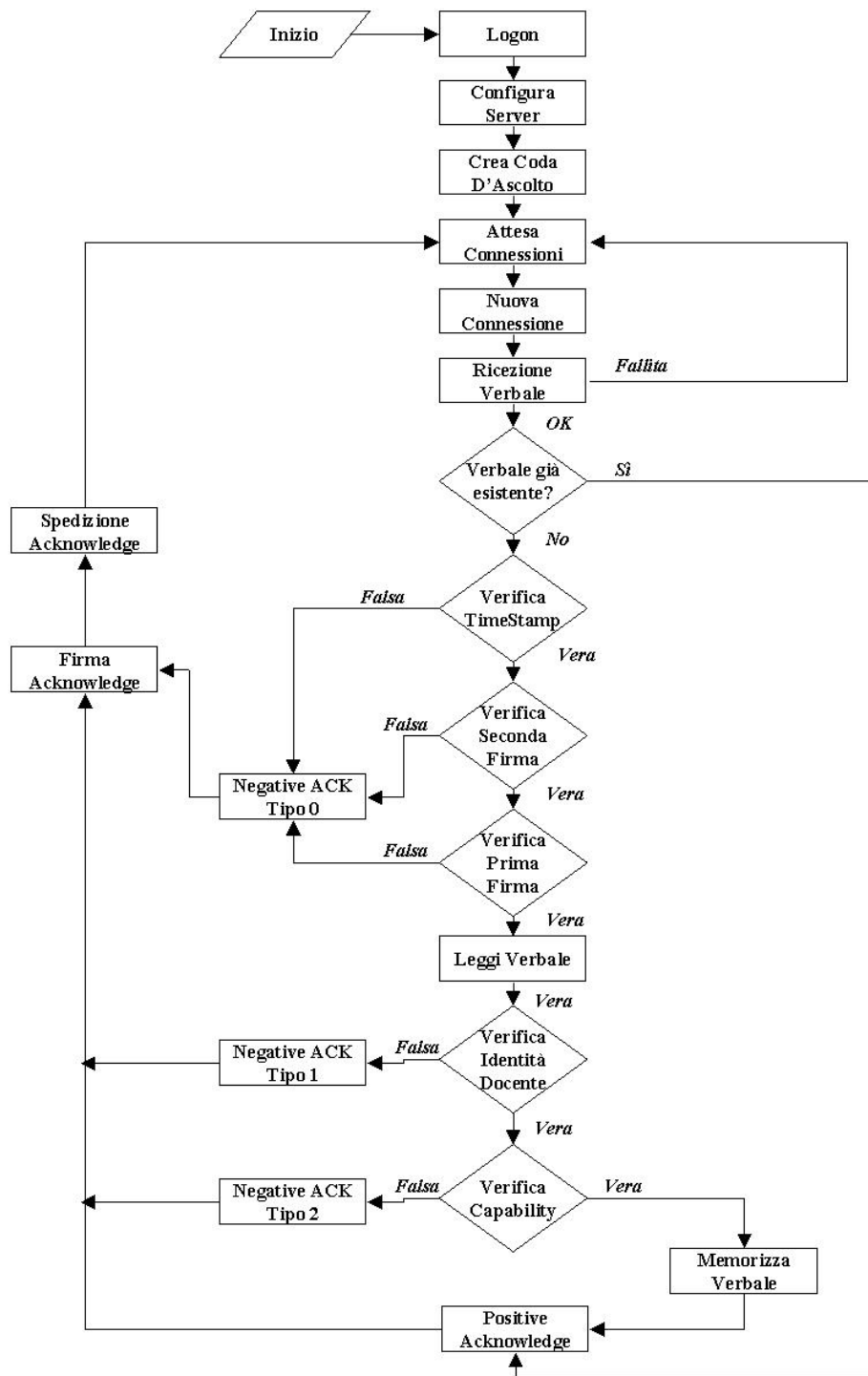


Figura 6.8: Diagramma di flusso dell'applicazione server

6.10 Le strutture dati

6.10.1 Record d'esame

Di seguito riportiamo la struttura dati che costituisce un record d'esame e che costituisce il verbale in chiaro.

Campo	Dimensione
Numero di serie dell'esame	3 bytes
Codice facoltà	2 bytes
Codice corso di laurea	2 bytes
Codice esame	5 bytes
Codice docente	5 bytes
Matricola studente	9 bytes
Data esame	8 bytes
Cognome (prime 4 lettere)	4 bytes
Voto	2 bytes
Quesiti	2 x 80 bytes = 160 bytes
Totale	200 bytes

Tabella 6.1: Record d'esame

6.10.2 Contenuto floppy disk

Nel floppy disk, oltre a essere contenute il nome, il cognome e il codice del docente, informazioni tra l'altro già contenute nel certificato elettronico, ci sono le informazioni relative a i corsi di cui il docente è titolare, che vengono utilizzate per facilitare la compilazione del verbale e prevenire possibili errori di digitazione.

Tutte queste informazioni sono memorizzate in quello che abbiamo chiamato file personale del docente

<codice corso, denominazione corso, codice facoltà, codice corso di laurea, numero ultimo verbale>

Ad esempio:

Nome: Maurelio
Cognome: Boari
Codice Docente: 13160

Codice Corso	Denominazione	Facoltà	CdL	Numero verbale
10384	Calcolatori elettronici II (elettr.)	21	46	30
10384	Calcolatori elettronici II (inf.)	21	48	100

Tabella 6.2: File personale del docente

6.10.3 Il file delle capability

Questo è il file che ha disposizione il server per verificare che il docente abbia compilato un verbale per il quale aveva diritto. Il file contiene una lista di stringhe, ciascuna che identifica un determinato esame.

<codice docente, codice facoltà, codice corso di laurea, numero verbale>

Di seguito riportiamo un esempio di tale file.

<i>13160214810384</i>	<i>Boari Calcolatori Elettronici II (INF)</i>
<i>13160214610384</i>	<i>Boari Calcolatori Elettronici II (EL)</i>
<i>15896214607946</i>	<i>Corradi Fondamenti di Informatica II (EL)</i>
<i>15896214807946</i>	<i>Corradi Fondamenti di Informatica II (INF)</i>
<i>15896214307946</i>	<i>Corradi Fondamenti di Informatica II (TELE)</i>
<i>15896214811145</i>	<i>Corradi Reti di Calcolatori (INF)</i>
<i>30202214909730</i>	<i>Ciampolini Fondamenti di Informatica (MEC)</i>

Figura 6.9: Estratto dal file delle capability

Capitolo 7. RISULTATI

A conclusione del nostro lavoro abbiamo effettuato alcuni test per verificare le performance del prototipo realizzato.

7.1 Prestazioni delle operazioni crittografiche

Innanzitutto abbiamo effettuato dei test di performance della classe `UtenteEntrust` che mette a disposizione le principali operazioni crittografiche, utilizzate nella progettazione dell'applicazione.

Le prove sono state effettuate per profili utenti su smart card, floppy disk e hard disk, utilizzando come elaboratore un Pentium II 233 MHz con 64 MB di RAM e seguendo le seguenti specifiche²⁶:

- file in ingresso di 100 KB
- modalità di lavoro *offline* (per rendere indipendenti le misure dalla velocità della rete)
- in ogni file firmato viene incluso anche il certificato di verifica del firmatario
- il processo di verifica della firma utilizza il certificato incluso nel file firmato

I risultati di queste prove sperimentali sono riassunti in figura 7.1; in particolare le prove si riferiscono ai seguenti test:

- rilevazione smart card: è il tempo necessario per rilevare la presenza di una smart card nell'apposito lettore e di verificare

²⁶ I tempi sono stati presi utilizzando la funzione `clock()` messa a disposizione dal linguaggio C ANSI che consente di calcolare i *clock ticks* che ci impiega il processore per elaborare una certa operazione.

all'interno della carta la presenza di profili utenti (corrisponde all'invocazione del metodo *TokenHardwareInfo()*.)

- Logon: è il tempo impiegato dall'applicazione per autenticare un utente; viene verificata la password associata al profilo e viene testata l'integrità delle chiavi e dei i certificati. Inoltre, se si lavora in modalità *online*, si verifica, accedendo alla CRL, se i certificati sono validi (corrisponde all'invocazione del metodo *LogonProfilo()*.)
- Firma: è il tempo necessario per firmare digitalmente il file (corrisponde all'invocazione del metodo *FirmaFile()*.)
- Verifica firma: è il tempo necessario per verificare la firma digitale di un file e ottenere le informazioni sul firmatario (corrisponde all'invocazione del metodo *VerificaFirma()*.)

Operazione	Smart Card	Floppy Disk	Hard Disk
Rilevazione smart card	0.877	0.661	0.600
Logon	2.766	1.390	0.949
Firma	2.480	0.587	0.184
Verifica Firma	0.056	0.050	0.050

Tabella 7.1: Tempi medi (in secondi) delle principali operazioni crittografiche

Analizzando la tabella 7.1, si può vedere come i tempi ottenuti utilizzando le smart card siano superiori. Questo perché parte delle operazioni vengono fatte direttamente dal processore della carta, che è comunque un processore con prestazioni modeste (nel nostro caso è un processore a 8 bit con 256 byte di RAM), mentre memorizzando il profilo su floppy disk o hard disk le

elaborazioni vengono effettuate completamente dal processore della macchina.

In particolar modo, questo è vero per quanto riguarda il processo di firma. La firma del documento passa attraverso due fasi: il calcolo dell'hash e poi la firma dell'hash. Nel caso si utilizzino smart card, la firma dell'hash viene effettuata dal microprocessore all'interno della carta (coadiuvato da un coprocessore aritmetico).

Un analogo discorso si può fare per il logon, in quanto in tale fase si accede al file di profilo dell'utente per controllarne l'integrità e verificarne la password: se si utilizza una smart card, l'intera operazione avviene all'interno della carta.

Invece non si notano differenze per quanto riguarda la verifica della firma, poiché per il processo di verifica la carta non viene chiamata in causa ma si utilizza il certificato del firmatario contenuto nel messaggio firmato.

La piccola differenza riscontrabile in fase di rilevamento della smart card è solo dovuto al fatto che in caso di successo di tale verifica si guardano quanti sono i profili contenuti nella carta e questo comporta un lieve overhead di tempo.

Le differenze che sussistono tra profili su hard disk e profili su floppy disk dipendono unicamente dalla diversa velocità di accesso alle periferiche. È più significativo il caso dei profili su floppy disk, in quanto rappresentano un dispositivo di memorizzazione delle chiavi alternativo alle smart card, qualora non sia necessario soddisfare le regole dell'AIPA, mentre la memorizzazione delle chiavi direttamente su hard disk è praticabile solo quando l'intera macchina è custodita in un posto sicuro e le misure riportate hanno solo lo scopo di evidenziare ulteriormente la differenza, in termini di potenza di elaborazione, rispetto alle smart card.

Operazione	Smart card vs. Floppy Disk	Smart card vs. Hard disk
Rilevazione smart card	+0.216	+0.277
Logon	+1.376	+1.817
Firma	+1.893	+2.296
Verifica Firma	+0.006	+0.006

Tabella 7.2: Overhead introdotto dalle smart card (tempi in secondi)

7.2 Le strutture dati

7.2.1 I profili utenti

Ogni utente Entrust dispone di un file di profilo che contiene la chiave privata di firma, la chiave privata di cifratura, i certificati di verifica e di cifratura, il certificato dell'Autorità di Certificazione e altre informazioni quali ad esempio gli algoritmi utilizzati. Il file di profilo è protetto da una password scelta dall'utente ed è cifrato con un algoritmo a chiave simmetrica. Questo file di profilo deve essere memorizzato nella EEPROM della smart card che, per motivi tecnologici, ha una capacità di memorizzazione piuttosto limitata; pertanto la dimensione di un profilo rappresenta un parametro importante per la valutazione dell'efficienza delle applicazioni.

Mediamente un profilo occupa dai 5000 ai 5400 byte; pertanto, utilizzando carte con EEPROM a 8 KB (come quelle

usate nel nostro progetto) è possibile memorizzare un unico profilo nella carta²⁷.

La generazione di un profilo avviene in fase di inizializzazione quando un nuovo utente, fornendo la coppia di codici segreti Reference Number - Authorization Code, crea un canale sicuro di comunicazione con la PKI. Ovviamente l'intero processo deve avvenire in modalità *online*. (*CreaProfilo()* è il metodo messo a disposizione dalla classe *UtenteEntrust*.)

La generazione di un profilo comporta la creazione di due coppie di chiavi. Nel caso si utilizzino le smart card, la generazione della coppia di chiave di firma avviene all'interno della carta grazie alla *key generation*. L'esecuzione dell'algoritmo di generazione è computazionalmente piuttosto pesante per la carta (tanto che deve essere presente un coprocessore aritmetico per avere tempi accettabili.)

In figura 7.3 riportiamo una tabella ove si può notare tale overhead. Per queste misure è stato usato un Pentium II 233 MHz con 64 MB di RAM. L'operazione di generazione del profilo è stata fatta sullo stesso computer che ospita la PKI in modo da non dover considerare il contributo dato dalla comunicazione, mentre come algoritmo è stato utilizzato RSA.

Dispositivo	Tempi
Smart Card	10 minuti
Floppy Disk	33 secondi
Hard Disk	28 secondi

Tabella 7.3: Tempi medi di generazione di un profilo utente

²⁷ Attualmente sono disponibili smart card con 8 o 16 KB di EEPROM, anche se stanno già comparando sul mercato carte con EEPROM a 32 KB.

7.2.2 Dimensioni dei verbali

Ricordando la struttura dei verbali vista nel precedente capitolo, abbiamo visto che la massima dimensione di un verbale in chiaro (cioè non ancora firmato) era di 200 byte. In particolare, il verbale è costituito da una parte fissa di 40 byte e da una parte variabile che conteneva i due quesiti sottoposti al candidato durante l'esame e che abbiamo supposto non superiore ai 160 byte.

Nel processo di firma al verbale in chiaro viene incluso l'hash²⁸ firmato e il certificato di verifica del firmatario come richiesto dall'AIPA. Il certificato occupa una dimensione di circa 1 KB (nei nostri test era di 808 byte).

Nel processo di validazione temporale viene allegato, oltre al certificato di verifica del Timestamp Agent, anche il suo certificato di cifratura e il certificato dell'Autorità di Certificazione d'appartenenza. Gli ultimi due certificati non sarebbero necessari in quanto il timestamp agent non copia alcuna operazione di cifratura e fa parte della stessa CA dei docenti; però, a livello di programmazione, non è stato possibile fare diversamente in quanto le primitive API messe a disposizione dal EntrustFile Toolkit tuttora non lo consentono.

In virtù di queste considerazioni vediamo come cambiano le dimensioni dei verbali attraverso i vari processi di firma:

²⁸ L'operazione di firma del hash viene effettuata, anziché sulla sola impronta a 160 bit, su una struttura di dati che la contiene insieme con altre informazioni utili, quali ad esempio l'indicazione della funzione hash usata per la sua generazione.

Verbale	Dimensione
Verbale in chiaro	200 byte
Verbale firmato da un docente	1480 byte
Verbale firmato da due docenti	2780 byte
Verbale firmato da due docenti e dal timestamp agent	4544 byte

Tabella 7.4: Dimensioni dei verbali

Come si deduce dalla tabella 7.4, il verbale pronto ad essere spedito al server avrà una dimensione di circa 4,5 KB. Ricordiamo che in questa sperimentazione ogni verbale contiene un unico esame.

7.2.3 Il file delle capability

Come visto nel capitolo precedente, il file delle capability è quel file che utilizza l'applicazione server per verificare che il docente abbia il diritto di firmare un certo esame. Nei nostri test abbiamo supposto che contenesse 1000 esami. Questi esami sono ordinati tramite la chiave primaria (codice docente, codice facoltà, codice corso di laurea, codice esame). La realizzazione di un database che contenga le capability esula dall'obiettivo della nostra tesi. Pertanto, tale file è realizzato semplicemente come un file di testo.

Il file delle capability contiene una struttura dati piuttosto semplice, ha una dimensione piccola (circa 60 KB) ed è memorizzato direttamente nell'hard disk del server; pertanto, l'accesso e la ricerca all'interno del file non comportano alcun overhead (al massimo qualche decimo di secondo).

7.3 I test dell'applicazione

I fattori che influiscono sulle prestazioni della nostra applicazione sono sostanzialmente due: la velocità nell'elaborare operazioni crittografiche e la velocità di trasmissione della rete. Questi due fattori sono strettamente correlati. All'inizio del capitolo abbiamo mostrato alcuni test significativi sulla classe UtenteEntrust. Questi test erano stati eseguiti in modalità *offline*; adesso nel testare l'intera applicazione passeremo necessariamente ad una modalità *online*, in cui inciderà inevitabilmente la velocità del mezzo di trasmissione. Infatti, operazioni come il logon, la verifica della firma, la validazione temporale richiedono il supporto di rete. In più, l'intera applicazione si basa su un modello client-server per la trasmissione dei verbali.

Le prestazioni di un protocollo di comunicazione quale il TCP/IP variano sensibilmente al variare di molteplici fattori (carico della rete, velocità delle linee fisiche di trasmissione, routing, ecc.) e tale variabilità può ulteriormente aumentare quando si trasmettono piccole informazioni come i nostri verbali. Il nostro intento non è assolutamente quello di fare un'analisi di performance del protocollo TCP/IP, ma unicamente di presentare i risultati di alcuni test significativi per la nostra applicazione.

Per decidere i test da effettuare, dobbiamo innanzitutto considerare l'architettura del sistema e vedere poi quali sono i fattori che possono influenzare le prestazioni dell'applicazione.

Riprendendo lo schema considerato nel quinto capitolo, abbiamo che il sistema è composto da quattro entità:

1. La PKI che comprende la Certification Authority e la Directory²⁹.

²⁹ In realtà, la CA e la Directory costituiscono a tutte gli effetti due entità diverse che possono trovarsi anche fisicamente su due macchine diverse. Inoltre, in caso di PKI con molti utenti, la Directory può essere ulteriormente partizionata.

2. Il server che gestisce la convalida e la memorizzazione dei verbali.
3. Il sistema di validazione temporale (timestamp server).
4. Le diverse postazioni client.

Supponiamo che la CA, la Directory, il server dei verbali e il timestamp server stiano tutte in un'unica rete locale; questa ipotesi è giustificata dal fatto che, come richiede lo stessa normativa dell'AIPA, le macchine che forniscono tali servizi si devono trovare in un posto sicuro, in cui vengano controllati gli accessi. A questo punto l'unica variabile del sistema è la collocazione del client; pertanto potremmo identificare alcuni test significativi a seconda di dove si trovi il client rispetto ai server. In particolare, possiamo identificare tre casi, ognuno corrispondente ad una diversa prova sperimentale:

1. Il client è situato nella medesima rete degli altri server.
2. Il client si trova in una rete diversa dalla rete del server, però sempre interna alla facoltà.
3. Il client è posizionato all'esterno della rete di facoltà.

Nel primo caso ci mettiamo in una condizione di massima efficienza da un punto di vista della comunicazione, in quanto nella nostra sperimentazione tutti i quattro elaboratori sono collegati ad un unico "hub" a 10 Mbps (vedi figura 7.1). Effettuando un test in queste condizioni, otteniamo pertanto un limite superiore per le prestazioni del sistema (*best-case*).

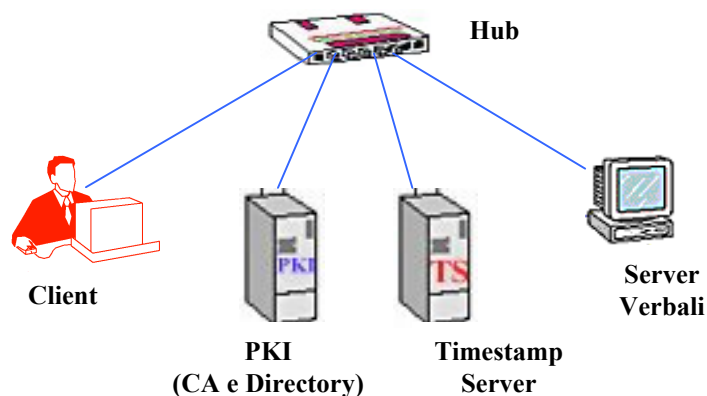


Figura 7.1: Test n. 1

Nel secondo caso il client si trova sempre all'interno della rete di facoltà, ma non nella sottorete a cui appartengono i diversi server. Stiamo considerando un caso più realistico (quando ad esempio il docente si trova in un qualche dipartimento della facoltà, da cui, utilizzando l'applicazione client, decide di verbalizzare digitalmente gli esami.)

Per valutare le prestazioni del sistema in questo caso, abbiamo collocato il client nella rete del Dipartimento di Elettronica, Informatica e Sistemistica (DEIS), e la rete dei server all'interno del Centro di Calcolo (CCIB). In figura 7.2 si può vedere qual è l'esatto schema di collegamento.

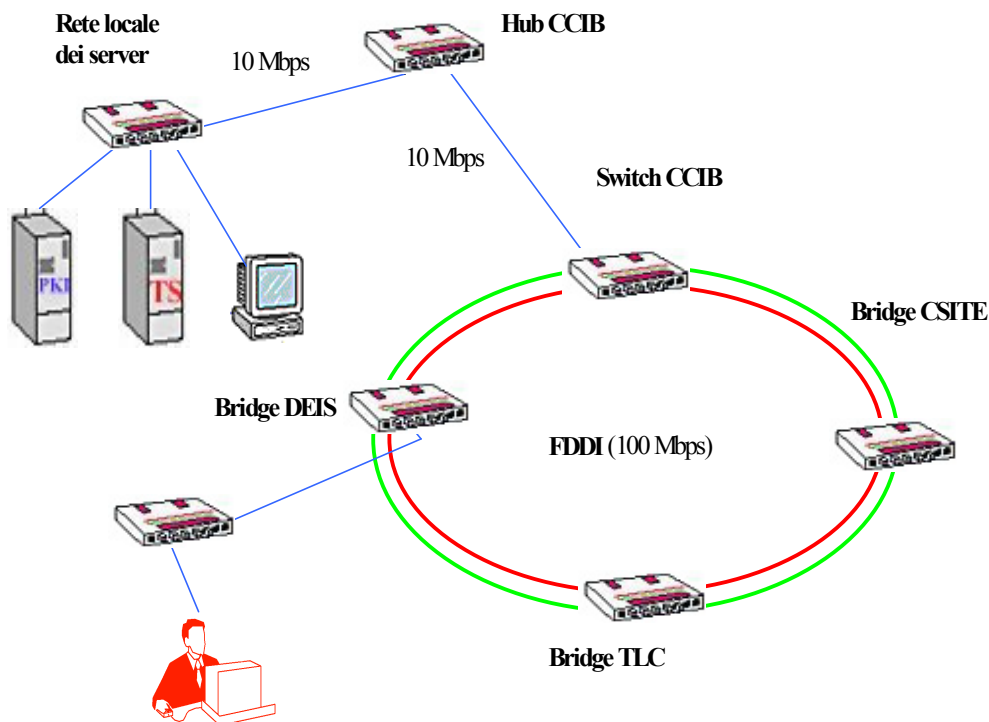


Figura 7.2: Test n. 2

Il terzo test simula una situazione in cui il client si trovi esternamente alla rete della facoltà. Una tale situazione potrebbe verificarsi se il docente si trova in un dipartimento fisicamente esterno alla facoltà (ad esempio, il CIRAM o il laboratorio di Montecuccolino) oppure se il docente si trova a casa sua, accedendo a Internet mediante la linea telefonica. È evidente che quest'ultimo caso è il più variabile di tutti, in quanto dipende da molteplici fattori, primi fra tutti la velocità di trasmissione del modem, il percorso necessario per raggiungere la rete interna (*routing*) e le condizioni di traffico della rete (vedi figura 7.3).

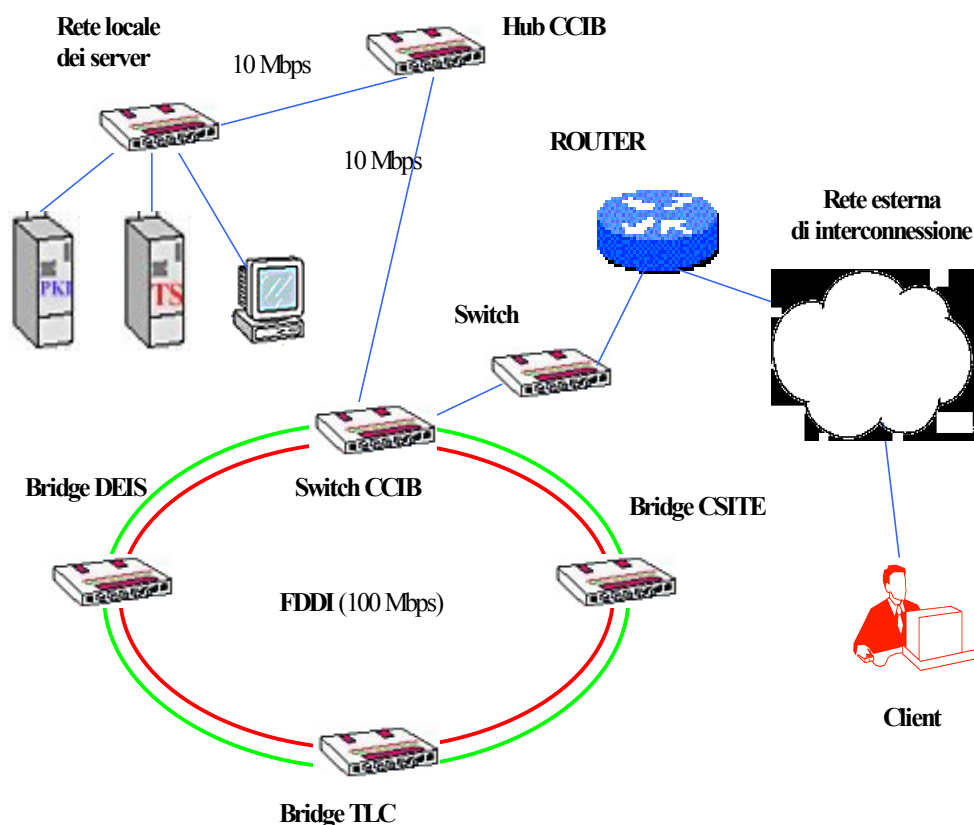


Figura 7.3: Test 3

I risultati dei tre test sono riportati nella tabella 7.5 in termini di valori massimi e minimi misurati (non abbiamo riportato valori medi data l'eterogeneità e la variabilità dei test effettuati rispetto al numero di campioni considerati.)

Locazione client	Tempo minimo	Tempo massimo
CCIB	19 secondi	23 secondi
DEIS	20 secondi	25 secondi
Esterno	42 secondi	91 secondi

Tabella 7.5: Risultati dei test sperimentali

I tempi sono dati dalla somma delle varie operazioni che il client svolge, come visto nel capitolo sei (si veda le figure 6.6 e 6.7). Nel dettaglio le operazioni considerate sono state:

1. rilevazione della smart card
2. logon (accesso al profilo e autenticazione dell'utente)
3. accesso alle estensioni X.509v3 del certificato elettronico per verificare se l'utente è effettivamente un docente
4. firma del verbale
5. ripetizione delle quattro operazioni precedenti per il secondo docente (seconda firma)
6. validazione temporale del verbale (timestamping)
7. trasmissione del verbale al server e attesa dell'acknowledge
8. ricezione dell'acknowledge da parte del server
9. verifica della firma dell'acknowledge
10. valutazione del contenuto acknowledge³⁰
11. cancellazione del verbale memorizzato localmente nel client
12. aggiornamento del numero progressivo di verbale

Ovviamente non sono stati conteggiati i tempi di attesa dovuti all'interazione tra applicazione e docente (inserimento della smart card, compilazione del verbale, ecc.) Naturalmente i valori del terzo caso sono puramente indicativi, dato che dipendono fortemente dalle condizioni in cui sono state effettuate le misure (provider utilizzato per la connessione telefonica, traffico della rete, ecc.)

Da queste misure possiamo giungere alla conclusione che i tempi ottenuti sono più che accettabili per la nostra applicazione.

³⁰ Nei test abbiamo sempre supposto che la verifica del verbale da parte del server andasse sempre a buon fine (non sono state considerate eventuali ritrasmissioni); pertanto, il client riceve sempre un positive acknowledge.

Capitolo 8. CONCLUSIONI

Lo scopo della tesi era quello di studiare la realizzazione di sistemi sicuri per la gestione di documenti informatici. Nel corso di questo studio abbiamo raggiunto i seguenti obiettivi:

- Abbiamo analizzato il contesto legislativo italiano per comprendere quali requisiti tecnici e organizzativi bisognasse soddisfare per poter dare validità legale al documento informatico.
- Abbiamo studiato in dettaglio la tecnologia delle smart card, spiegando come si renda necessaria l'utilizzo di questa tecnologia nella realizzazione di sistemi per la gestione di documenti informatici.
- Abbiamo considerato il problema di come realizzare un'infrastruttura a chiave pubblica; tale studio ci ha portato alla scelta di un prodotto software, Entrust, in quanto forniva tutti i servizi richiesti dalle specifiche del nostro progetto e si proponeva sul mercato come leader dei tool per le PKI.
- Abbiamo scelto la verbalizzazione degli esami come applicazione per sperimentare l'adozione del documento informatico all'interno dell'università.
- Come risultato finale della tesi, abbiamo progettato, realizzato e verificato sperimentalmente il prototipo dell'applicazione, che si integra con la PKI realizzata con Entrust e soddisfa, ove gli strumenti a nostra disposizione lo consentivano, i requisiti tecnici richiesti dall'AIPA. In particolare, il prototipo realizzato ha consentito di sperimentare l'utilizzo delle smart card.

L'applicazione realizzata non dispone ancora di un'interfaccia grafica; pertanto una prima estensione del lavoro

svolto potrebbe essere la realizzazione di un'interfaccia utente che faciliti la compilazione dei verbali da parte del docente. Eventualmente si potrebbe pensare di integrare l'interfaccia grafica con gli strumenti Web. Originariamente il nostro progetto comprendeva questo obiettivo; purtroppo non è stato possibile, in quanto il toolkit di sviluppo in Java non consente ancora il supporto per smart card.

Una volta realizzata un'infrastruttura a chiave pubblica a livello di Ateneo, è possibile sperimentare in altre applicazioni la firma digitale. Ad esempio, una possibile applicazione sarebbe la realizzazione di un sistema per le votazioni interne all'università.

Un'altra estensione potrebbe essere la possibilità di interazione con Autorità di Certificazione di altri Atenei. Si potrebbe studiare le problematiche e le possibili soluzioni per la *cross-certification* tra CA.

Si potrebbe pensare di fornire le smart card anche agli studenti. In questo modo il processo di compilazione dei verbali risulterebbe ancora più veloce, perché l'applicazione potrebbe reperire direttamente dalla smart card le informazioni sullo studente (matricola, corso di laurea, nome e cognome). Inoltre, si darebbe la possibilità allo studente di firmare digitalmente il verbale di esame o altri documenti, qualora fosse richiesto.

Appendice A. ENTRUST

Entrust/PKI versione 4.0 è costituito da quattro applicativi software distinti da un punto di vista funzionale:

- Entrust/Authority
- Entrust/Admin
- Entrust/Directory
- Entrust/Entelligence

In più esistono alcuni software aggiuntivi che consentono di disporre di ulteriori funzionalità. I principali prodotti che si integrano con Entrust/PKI sono:

- Entrust/Web Connector
- Entrust/Commerce Connector
- Entrust Timestamp
- Entrust/VPN Connector
- Entrust-Ready application

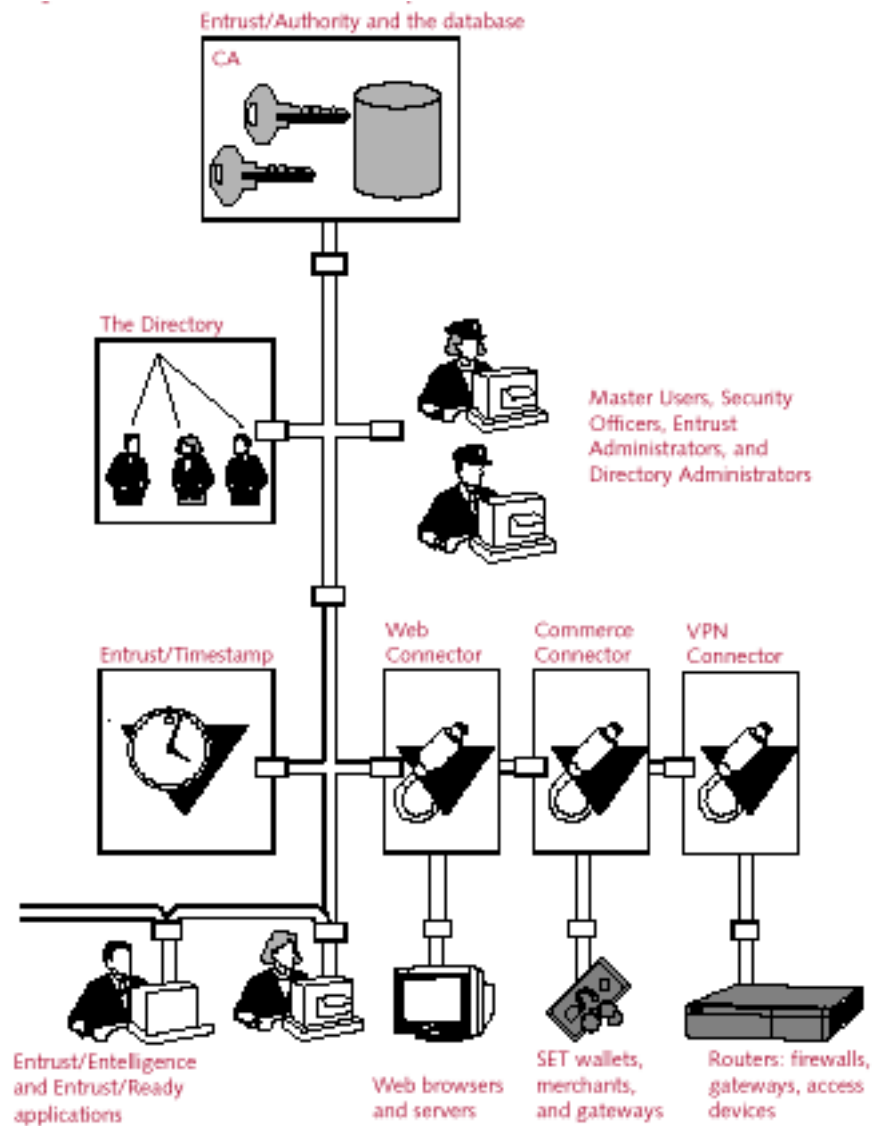


Figura A.1: I componenti del sistema Entrust/PKI

Entrust/Authority

Costituisce il cuore del sistema Entrust/PKI e le sue principali funzioni sono:

- Crea i certificati di verifica e di cifratura.
- Crea le coppie di chiavi di cifratura per gli utenti.
- Gestisce il database sicuro della CA in cui sono memorizzate le informazioni cruciali sul sistema e sugli utenti.

- Definisce la politica di sicurezza dell'organizzazione
- Consente di personalizzare i certificati attraverso le estensioni del formato X.509v3.
- Gestisce le regole che gli utenti devono seguire per scegliere le loro password personali.
- Consente di scegliere il tipo di algoritmo (RSA o DSA) per la firma della CA

In particolare, il database della CA gestito da Entrust/Authority (spesso chiamato Informix Database) contiene:

- la coppia di chiavi della CA.
- le informazioni di stato di tutti gli utenti.
- la storia delle coppie di chiavi di cifratura di tutti gli utenti, ossia tutte le chiavi private di decifratura, anche quelle scadute, e tutti i corrispondenti certificati elettronici. In questo modo è possibile il *key recovery* nel caso in cui gli utenti, per esempio, abbiano dimenticato la password.
- la storia di tutti i certificati di verifica (le chiavi private di firma invece non vengono archiviate in quanto non necessitano di backup e inoltre non sono generate da Entrust/Authority, ma dall'applicazione utente.)
- i periodi di validità della coppie di chiavi di firma e di cifratura e dei *cross-certificate*, cioè di quei certificati che consentono di instaurare relazioni di fiducia con le altre CA.
- le informazioni sugli amministratori del sistema.

Entrust/Admin

Fornisce la parte grafica di interfaccia a Entrust/PKI; può essere utilizzato nella stessa macchina dove risiede Entrust/admin sebbene per motivi di sicurezza sia sconsigliato. È utilizzato dai

due speciali utenti (Security Officers e Administrators) e mette a disposizione tutte le funzioni di amministrazione per una PKI. (Vedremo nel dettaglio tali funzioni quando parleremo dei vari tipi di utenti previsti nel modello di Entrust.)

Entrust/Directory

È il componente che realizza il sistema di Directory di una PKI. In realtà, questo componente può essere rimpiazzato da un qualsiasi altro prodotto che realizzi una Directory che comunichi tramite il protocollo LDAP. (La stessa Entrust dichiara che la sua Directory può essere utilizzata solo per piccole organizzazioni che non hanno bisogno di tutte le funzionalità di una Directory secondo lo standard X.500)

Entrust/Directory contiene tutti i certificati di cifratura degli utenti, mentre non contiene i certificati di verifica. Inoltre gestisce le liste di revoca (CRL) e, nel caso in cui la CA abbia stretto rapporti di fiducia con altre CA, contiene i certificati che attestano tale relazione (*cross-certificate*) e la relativa lista di revoca (ARL – Authority Revocation List).

Entrust/Entelligence

È il componente client che Entrust mette a disposizione per gli utenti finali nel caso in cui non si disponga di un'applicazione ad hoc per interfacciarsi con la PKI. Le sue principali funzioni sono:

- Crea la coppia di chiavi di firma.
- Aggiorna le coppie di chiavi di cifratura e di firma.
- Verifica automaticamente la validità dei certificati di cifratura e di verifica prima che ogni operazione crittografica venga eseguita.

- Consente di scambiare le chiavi pubbliche con utenti Entrust di altre CA che non hanno rapporti di *cross-certification* con la propria CA.
- Utilizzando un meccanismo di sicurezza proprietario (Entrust archive) consente agli utenti di firmare, cifrare, decifrare ogni tipo di file.

Entrust/Entelligence e Entrust/Authority raramente hanno bisogno di comunicare tra di loro; quando nasce questa esigenza (creazione di nuovi utenti, aggiornamento delle chiavi di cifratura e di firma, key recovering), la comunicazione avviene tramite un protocollo proprietario SEP (Secure Exchange Protocol) che crea un canale sicuro e autenticato.

Vediamo brevemente gli altri componenti aggiuntivi di Entrust:

- **Entrust/Web Connector:** consente di pubblicare i certificati Web gestibili dai browser Netscape e Explore e utilizzabili da applicazioni che usano gli standard S/MIME o SSL.
- **Entrust/Commerce Connector:** consente di pubblicare certificati compatibili con lo standard universalmente riconosciuto nel commercio elettronico, SET (Secure Electronic Transaction), che consente transazioni sicure con carta di credito su internet.
- **Entrust/Timestamp:** fornisce un servizio di validazione temporale.
- **Entrust/VPN Connector:** distribuisce certificati per dispositivi VPN (Virtual Private Network) come router, VPN gateway, firewall.
- **Entrust-Ready application:** sono i vari prodotti destinati agli utenti finali. Con Entrust/Express si può mandare e ricevere e-

mail sicure; con Entrust/Direct si può accedere a siti web sicuri; con Entrust/ICE si può archiviare i propri file in maniera sicura.

Gli amministratori di Entrust/PKI

Entrust prevede una gerarchia di amministratori, ognuno dei quali ha dei specifici compiti nell'amministrazione del sistema PKI.

1. Master User:

Sono coloro che installano e configurano il sistema PKI. Successivamente sono responsabili di gestire il database della CA, di avviare o fermare i vari servizi di amministrazione, di gestione delle chiavi e di Directory.

2. Security Officer:

Sono coloro che, attraverso Entrust/Admin, decidono la politica di sicurezza dell'organizzazione, aggiungono e rimuovono Administrators, Directory Administrators e Security Officers; inoltre si occupano della *cross-certification* con le altre CA.

3. Administrators:

Sono coloro che, attraverso Entrust/Admin, aggiungo nuovi utenti, abilitano e disabilitano gli utenti, revocano i certificati, gestiscono la validità temporale delle chiavi, svolgono un'attività di auditing.

4. Directory Administrators:

Sono administrators che possono utilizzare Entrust/Admin unicamente per gestire la Directory.

La creazione delle chiavi

Entrust gestisce in maniera differente le coppie di chiavi destinate alle operazioni di cifratura e decifratura da quelle destinate alla generazione e verifica della firma.

La coppia di chiavi di cifratura e il corrispondente certificato elettronico sono create da Entrust/Authority; la chiave privata di decifratura viene poi spedita attraverso il protocollo SET all'applicazione client (ad esempio, Entrust/Entelligence) la quale provvede a memorizzarla nel profilo dell'utente; inoltre una copia di tale chiave viene memorizzata nel database della CA. La chiave pubblica di cifratura viene memorizzata all'interno del corrisponde certificato; il certificato viene poi memorizzato nel profilo e pubblicato nella Directory; in più, il database della CA ne fa una copia di backup.

Oggetto	Creata da	Messa in	Backup
Chiave privata di decifratura	Entrust/Authority	Profilo utente	Nel database della CA
Chiave pubblica di cifratura	Entrust/Authority	Certificato di cifratura	Nel database della CA solo finché non viene fatto il backup del certificato di cifratura
Certificato di cifratura	Entrust/Authority	Directory e profilo utente	Nel database della CA

Tabella A.1: Creazione e locazione delle chiavi di cifratura

La coppia di chiavi di firma è creata localmente dall'applicazione client. La chiave privata di firma viene memorizzata nel profilo dell'utente, mentre la chiave pubblica di verifica viene spedita a Entrust/Authority che provvede a generare il corrispondente certificato. Il certificato di verifica viene poi

rispedito all'applicazione client, mentre una copia viene archiviata nel database della CA (non viene pubblicato nella Directory.)

Oggetto	Creata da	Messa in	Backup
Chiave privata di firma	Applicazione client	Profilo utente	Nessuno
Chiave pubblica di verifica	Applicazione client	Certificato di verifica	Nessuno
Certificato di verifica	Entrust/Authority	Profilo utente	Nel database della CA

Tabella A.2: Creazione e locazione delle chiavi di firma

La firma e la cifratura di un file

Per cifrare e firmare un file, Entrust utilizza una tecnologia “ibrida”, ossia una tecnologia basata sia sulla crittografia a chiave pubblica sia su quella a chiave simmetrica. Vediamo quali sono le fasi attraverso le quali è possibile firma e cifrare un file e poi successivamente decifrarlo e poi verificarne la firma.

1. Si prende la chiave privata dell'utente con cui si firma digitalmente il file. La chiave pubblica di verifica corrispondente è inclusa nel file per consentire agli altri utenti di verificare la firma.
2. Si genera una chiave simmetrica casuale “monouso” e si cifra il file con tale chiave.
3. Si ricerca nella Directory le chiavi pubbliche di coloro a cui si vuole spedire il file, se ne verifica la loro validità e si usano queste chiavi per cifrare la chiave simmetrica che è stata usata per criptare il file. Si noti che la chiave simmetrica è cifrata tante volte quanti sono i destinatari del file. Fra i destinatari, normalmente, c'è sempre anche lo stesso mittente cosicché

l'autore del file è sempre in grado di decifrare file da lui stesso cifrati.

4. All'inizio del file criptato vengono incluse le copie delle chiavi simmetriche criptate con il nome del corrispondente destinatario.
5. Viene spedito ai destinatari il file firmato e criptato.

A questo punto i riceventi del file criptato e firmato possono procedere alla decifrazione e verifica di firma.

1. Nel file criptato si cerca nella lista dei destinatari il proprio nome e la corrispondente chiave simmetrica criptata.
2. Con la propria chiave privata di decifrazione si decifra la chiave simmetrica.
3. Con la chiave simmetrica ottenuta si decifra il file criptato.
4. Si prende la chiave pubblica di verifica dell'utente che ha firmato il file, se ne verifica la validità e la si usa per verificare la firma. Se il file non ha subito cambiamenti, la firma viene riconosciuta.

Aggiornamento delle chiavi

Entrust fornisce un meccanismo automatico e trasparente di aggiornare periodicamente le chiavi. Quando una coppia di chiavi (di firma o di cifratura) sta per scadere, vengono generati una nuova coppia di chiavi e il corrispondente certificato elettronico e rimpiazzano nel profilo utente i vecchi valori.

A seconda del periodo di validità delle chiavi è possibile stabilire quando deve partire il processo di aggiornamento, cosicché l'intera procedura può risultare completamente trasparente all'utente.

Appendice B. ENTRUSTFILE TOOLKIT

Nel capitolo sesto abbiamo visto quali sono le principali funzionalità che il EntrustFile Toolkit mette a disposizione a livello di programmazione. Adesso vogliamo fare una panoramica sulle classi del toolkit, tramite le quali è stata costruita la classe UtenteEntrust.

Classe EntFile

È la classe principale del toolkit, quella che fornisce le operazioni di firma, verifica, cifratura e decifratura. Il costruttore della classe EntFile crea un oggetto associato ad un utente Entrust; i vari metodi consentono di gestire i tre meccanismi di sicurezza supportati (PKCS #7, PEM, Entrust archive), di utilizzare i servizi di timestamping, di determinare la presenza di virus, di gestire un proprio indirizzario di utenti con i relativi certificati (Personal Address Book), di interfacciarsi con la Directory.

Classe EntFileList

Questa classe consente di creare una lista di file da firmare e cifrare col meccanismo di sicurezza Entrust Archive. I metodi consentono di gestire tale lista.

Classe EntCertificate

Questa classe consente di gestire i certificati elettronici secondo il formato X.509. Il costruttore viene istanziato con un oggetto di tipo EntFile che contenga un certificato; i vari metodi

permettono di accedere alle informazioni contenute nel certificato (nome, numero di serie, versione, validità temporale, uso della chiave, CA di appartenenza, estensioni, ecc.)

Classe EntCertificateList

La classe EntCertificateList consente di estrarre da messaggi PKCS #7 i certificati dei mittenti. In questo modo, dopo aver verificato la validità dei certificati, è possibile includerli nel personale indirizzario. Questo consente di disporre già dei certificati degli utenti con cui si comunica senza dover accedere alla Directory; inoltre consente di poter interagire con utenti appartenenti ad altre CA, anche con utenti che non usino il sistema Entrust, ma che utilizzino un meccanismo di posta sicura che rispetti lo standard S/MIME.

Classe EntrustName

Questa classe fornisce un modo pratico per gestire le informazioni degli utenti Entrust senza dover ricorrere direttamente ai certificati X.509; ovviamente può essere utilizzata solo all'interno di sistemi PKI costruiti con Entrust.

Classe EntrustNameList

Consente di gestire una lista di oggetti EntrustName, che può essere ad esempio utile per poter cifrare un messaggio per più utenti (sempre all'interno di sistemi Entrust.)

Classe EntHeader

Gli header sono le parti iniziali dei messaggi in cui vengono memorizzate le informazioni crittografiche quali la firma digitale o le chiavi simmetriche cifrate necessarie per decifrare il file. Questa classe consente di accedere direttamente a queste informazioni (di solito, i metodi sono trasparenti e non consentono di accedere a tale informazioni.)

Classe TimestampInfo

Consente di accedere alla marca temporale e di estrarre le informazioni ivi contenute (ora e data di firma, firmatario, algoritmo di firma, Autorità di Certificazione).

Classe GString

La classe Gstring (Generic String) fornisce dei metodi per manipolare le stringhe che vengono utilizzate dai metodi delle varie classi.

Classe BinData

Un BinData è un array di byte e questa classe fornisce dei metodi per operare sui BinData con lo scopo di realizzare, per motivi di efficienza, un buffer tra i dati dei file (che possono essere anche molto grandi) e gli algoritmi di sicurezza che devono processare tali dati.

Funzioni di utilità

Il toolkit mette a disposizione una serie di funzioni tra le quali ricordiamo quelle per utilizzare le smart card e quelle per gestire gli errori.

BIBLIOGRAFIA

- [ABC99] American Biometric Company, "Biometric and Smart Card User Authentication",
<http://www.abio.com/technology.html>, Gennaio 99.
- [ACPZ99] C. Adams, P. Cain, D. Pinkas, R. Zuccherato, "Internet X.509 Public Key Infrastructure - Time Stamp Protocols", PKIX Working Group *Draft*, Maggio 1999.
- [AG99] "Profile of Entrust", Aberdeen Group, Marzo 1999.
- [ATS95] J. Priisalu, "A frequently Asked Questions list for alt.technology.smartcards",
<http://www.ioc.ee/atse/faq.html>, Luglio 1995.
- [COM95] D. E. Comer, "Internetworking with TCP/IP volume I: Principles protocols and architecture", Prentice Hall, 1995.
- [CUR95] I. Curry, "The concept of trust in network security", Entrust Technologies White Paper, Dicembre 1995.
- [CUR96] I. Curry, "Version 3 X.509 Certificates", Entrust Technologies White Paper, Luglio 1996.
- [CD98] Z. Chen, R. Di Giorgio, "Understanding Java Card 2.0 – Java World", Marzo 1998.

- [CS97] D. E. Comer, D. L. Stevens, "Internetworking with TCP/IP volume III: Client-server programming and applications", Prentice Hall, 1997.
- [CSCS98] "Contactless Smart Card in Seoul",
<http://www.cardshow.com/guide/card/korea.html>, 1998.
- [DPB96] H. Dobbertin, A. Bosselaers, B. Preneel, "RIPEMD-160: A strengthened version of RIPEMD", in "Proceedings of 3rd International Workshop on Fast Software Encryption", 1996.
- [DH76] W. Diffie, M. Hellman, "New directions in cryptography", in "IEEE Transactions on Information Theory", vol. 22 (1976), pp. 644-654, 1976.
- [DK98] "SignaSURE Cryptographic Interface Provider - SignaSsure CIP 2.0 User's Guide", Datakey, 1998.
- [EI98] "Entrust Infrastructure - User Guide (release 4.0)", Entrust Technologies Limited, 1998.
- [ENT98] "Entrust/PKI 4.0 NT Installation", Entrust Technologies Limited, 1998.
- [EPF98] "Entrust File Toolkit: C++ Programmers' Reference", Entrust Technologies Limited, Giugno 1998.
- [EPG98] "Entrust File Toolkit: C++ Programmers' Guide", Entrust Technologies Limited, Giugno 1998.

- [ETS98] "Entrust Timestamp - User Guide", Entrust Technologies Limited, 1998.
- [F94] W. Ford, "The need for separate key pairs for symmetric key transfer and digital signature", Febbraio 1994.
- [FNMT98] FNMT, "Deliverable D3 - Architecture of Time-Stamping Service and Scenarios of Use: Service and Features", PKITS, <http://selva.dit.upm.es/~pepe/pkits> Giugno 1998.
- [ITSEC] Information Technology Security Evaluation Criteria, <http://www.itsec.gov.uk>
- [JC99] "Java Card 2.1 Application Programming Interface", Sun Microsystem, <http://java.sun.com/products/javacard/index.html>, Febbraio 1999.
- [JFAQ98] "Java Card Technologies - Frequently Asked Question", <http://java.sun.com/products/javacard/faq.html>, Sun Microsystem, Luglio 1998.
- [K98] M. Kaiserswerth, "The OpenCard Framework and PC/SC", IBM, Marzo 1998.
- [L97] Jamie Lewis, "Public Key Infrastructure", The Burton Group, Luglio 1997.
- [MAM99] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, "X.509 Internet Public Key Infrastructure

Online Certificate Status Protocol – OCSP”, PKIX Working Group *Draft*, Marzo 1999.

- [MFAQ98] "Smart Card - Frequently Asked Question", Microsoft Corporation, <http://www.smartcardsys.com>, 1998.

- [OCF98] "OpenCard Framework - General Information Web Document", IBM, <http://www.opencard.org>, Ottobre 1998.

- [PCS99] "The smart card market by the year 2000", <http://www.cardshow.com/statistics/uk/philips.html>, Philips Communication Systems, 1999.

- [PCSC97] "Interoperability Specification for ICCs and Personal Computer System”, Part 1. Introduction and Architecture Overview, Revision 1.0, Dicembre 1997.

- [PKCS6] "PKCS #6: Extended-Certificate Syntax Standard”, RSA Laboratories, Novembre 1993.

- [PKCS7] B. Kaliski, "PKCS #7: Cryptographic Message Syntax Version 1.5", Request For Comment 2315, Marzo 1998.

- [PKCS9] "PKCS #9: Selected Attribute Types”, RSA Laboratories, Novembre 1993.

- [PKCS11] "PKCS #11: Cryptographic Token Interface Standard (versione 2.01)”, RSA Laboratories, Dicembre 1997.

- [PKCS15] "PKCS #15 v1.0: Cryptographic Token Information Format Standard", RSA Laboratories, Aprile 1999.
- [RFAQ99] "PKCS #11 Frequently Asked Question", RSA Laboratories, <http://www.rsa.com>, 1999.
- [RFC1319] B. Kaliski, "The MD2 Message-Digest Algorithm", Request For Comment 1319, Aprile 1992.
- [RFC1320] R. Rivest, "The MD4 Message-Digest Algorithm", Request For Comment 1320, Aprile 1992.
- [RFC1321] R. Rivest, "The MD5 Message-Digest Algorithm", Request For Comment 1321, Aprile 1992.
- [RFC1421] J. Linn, "Privacy Enhancement for Internet Electronic Mail - Part I: Message Encryption and Authentication Procedures", Request For Comment 1421, Febbraio 1993.
- [RFC1777] W. Yeong, T. Howes, S. Kille, "Lightweight Directory Access Protocol", Request For Comment 1777, Marzo 1995.
- [RFC2311] S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, L. Repka, "S/MIME Version 2 Message Specification", Request For Comment 2311, Marzo 1998.
- [RFC2459] R. Housley, W. Ford, W. Polk, D. Solo, "Internet X.509 Public Key Infrastructure - Certificate and CRL Profile", Request For Comment 2459, Gennaio 1999.

- [RFC2510] C. Adams, S. Farrell, "Internet X.509 Public Key Infrastructure – Certificate Management Protocols", Request For Comment 2510, Marzo 1999.
- [RSA78] R. L. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", in "Communications of the ACM", vol. 21, n. 2, pp. 120-126, 1978.
- [RSA98] "Frequently asked questions about today's cryptography", RSA Laboratories, 1998.
- [S90] C. P. Schnorr, "Efficient identification and signatures for smart cards", Advances in Cryptology, 1990.
- [S96] B. Schneier, "Applied cryptography", John Wiley & Sons, 1996.
- [SCM98] "The Smart Card Museum – The Birth of Smart Cards", <http://www.cardshow.com/museum/exhibition.html>, Marzo 1998.
- [SCO99] "Smart Card Overview", Sun Microsystems, <http://java.sun.com/products/javacard/smartcards.html>, Marzo 1999.
- [T97] S. Taschler, "Technical Introduction to SignaSure CIP", Datakey, <http://www.datakey.com>, Settembre 1997.

Questo lavoro non sarebbe completo senza esprimere il mio personale ringraziamento a tutto il personale del Centro di Calcolo (C.C.I.B) della Facoltà di Ingegneria di Bologna che ne ha permesso lo svolgimento presso i suoi laboratori. Inoltre ringrazio vivamente il Prof. Aurelio Boari, la Prof.ssa Anna Ciampolini, il Prof. Roberto Laschi, l'Ing. Rebecca Montanari, la CryptoNet s.r.l., Franco Gola e quanti mi hanno aiutato in questi mesi.