Università
Ca'Foscari
Venezia

Corso di Laurea Specialistica (*ordinamento ex D.M. 509/1999*)

In Informatica

Ca' Foscari
Dorsoduro 3246
30123 Venezia

Tesi di Laurea

Analisi statica della sicurezza dei protocolli di voto

Relatore
Ch. Prof. Agostino Cortesi

Laureando
Barletta Massimiliano
Matricola 802646

Anno Accademico
2011 / 2012

email:: mbarlett@dsi.unive.it

email::

Dipartimento di Informatica
Università Ca' Foscari di Venezia
Via Torino, 155
30172 Venezia Mestre – Italia
tel. +39 041 2348411
fax. +39 041 2348419
web: http://informatica.dsi.unive.it

Alla mia famiglia ed Anna
che mi hanno sostenuto
in questo percorso di studi!

Indice

Elenco delle figure

Elenco delle tabelle

1
Introduzione

Con gli anni la tecnologia ha cambiato sempre più spesso le nostre abitudini e sempre più persone hanno ora un accesso ad internet. Questa evoluzione ha reso possibili comunicazioni rapide (spesso istantanee) ed ha offerto la possibilità di ridurre i costi (possiamo anche non spostare l'auto per acquistare un prodotto) e il tempo (spesso questioni di pochi minuti) per effettuare alcune tipiche operazioni che un tempo richiedevano interminabili code agli sportelli. Perchè quindi non sfruttare la tecnologia per migliorare, se possibile, un'operazione tipica dei paesi democratici, quale il voto alle elezioni?.
Molte sono le soluzioni che i ricercatori hanno proposto negli ultimi anni sia a livello pratico sia a livello teorico. Tuttora la sperimentazione procede in alcuni paesi alla ricerca di una soluzione che eviti la possibilità di frode nel rispetto della riservatezza di chi vota.
La rete d'altra parte presenta dei problemi di sicurezza che vanno valutati attentamente affinchè i protocolli di voto non presentino falle che un possibile attaccante possa sfruttare per manipolare, distruggere, falsificare il voto dell'elettore. L'obiettivo pertanto è quello di presentare un metodo per poter validare automaticamente le proprietà che un protocollo di voto elettronico deve rispettare per garantire la volontà degli elettori.

1.1 Sistemi tradizionali di voto

La caratteristica distintiva di ogni democrazia sono le elezioni. Le persone hanno il diritto e il dovere di votare, cioè effettuare una scelta tra alcuni candidati, oppure rispondere in maniera affermativa o negativa (referendum) ad alcune proposte e/o quesiti.
Il sistema tradizionale di voto prevede che le elezioni si svolgano in prefissati seggi dislocati nel paese e ciascuna **persona avente diritto al voto** esprima la propria preferenza **recandosi fisicamente** al seggio di appartenenza e compilando a mano la scheda o le schede presenti. Al termine delle votazioni, gli scrutatori effettuano la validazione delle schede e procedono con lo scrutinio.
Solo dopo aver effettuato i conteggi vengono resi pubblici i risultati.

1.2 Problemi tipici

Vediamo alcuni problemi ([1]) che si presentano utilizzando il sistema di voto tradizionale per comprendere successivamente i possibili vantaggi che si possono ottenere utilizzando tecniche di voto elettronico:

1. **Costo**: indire le elezioni oppure un referendum, implica una notevole spesa per la Pubblica Amministrazione. Scrutatori, presidenti di seggio, attrezzature per le operazioni di voto hanno un costo elevato.

2. **Tempo**: Possono in genere votare poche persone alla volta. Lo scrutinio delle schede è un lavoro non difficile ma molto laborioso. Dalla conclusione delle operazioni di voto alla pubblicazione dei risultati finali possono passare molte ore se non addirittura giorni.

3. **Distanza dal seggio**: L'obbligo dell'elettore di dover raggiungere il proprio seggio è statisticamente una delle cause maggiori di astensione dal voto.

4. **Errori nello scrutinio e nel voto**: Le procedure di voto, e soprattutto di conteggio dei voti, non sono esenti da errori. L'interpretazione da parte di chi esamina le schede può essere soggettiva: crocette non evidenti, segni ambigui, schede nulle e altre varianti.

1.3 Protocolli di voto elettronico

Il voto può essere immaginato come una transazione tra un elettore ed il sistema elettorale (fig.1.1). Possiamo pertanto dare una definizione di voto elettronico come l'implementazione di tale transazione con l'ausilio di strumenti informatici. Si richiede pertanto di sviluppare un certo **protocollo** che garantisca le proprietà fondamentali legate al voto.

1.4 I requisiti dei protocolli di voto elettronico

Un protocollo di voto elettronico deve soddisfare dei requisiti minimi di sicurezza per poter essere utilizzato con efficacia in uno scenario reale:

- **Eleggibilità**: solamente gli elettori che ne hanno diritto possono partecipare alle votazioni. Possono votare una e una sola volta.

- **Accuratezza**: non deve essere possibile alterare il voto, eliminare dal computo finale un voto valido e/o mantenere nel conteggio finale un voto non valido;

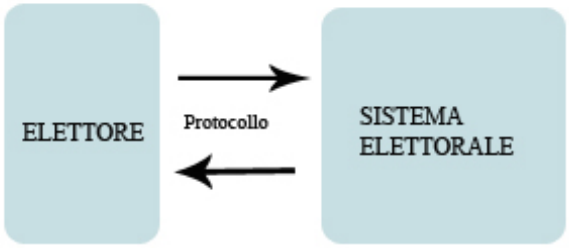

Figura 1.1: Sistema elettorale

- **Privacy**: nessuna entità può collegare una scheda elettorale (proprio voto o preferenza) all'elettore.
- **Verificabilità**: in caso di dubbi e/o contestazioni deve essere garantita una facile verifica. Inoltre gli elettori devono essere certi che il proprio voto sia stato conteggiato correttamente;
- **Efficienza**: Le fasi di voto devono essere svolte in tempi ragionevoli. Un elettore non deve attendere che termini un altro elettore per effettuare la propria votazione e deve poterlo fare senza l'ausilio di particolari strumenti e senza avere particolari conoscenze;
- **Scalabilità**: La dimensione delle elezioni non deve assolutamente incidere negativamente sull'efficienza.

Requisito opzionale:
Responsabilità: poichè in alcuni paesi è obbligatorio votare deve essere possibile individuare gli elettori che non hanno votato e non hanno presentato una valida giustificazione.

1.5 Elementi base dei protocolli di voto

I principali protocolli di voto elettronico sono caratterizzati dall'utilizzo di alcune tecniche crittografiche comuni.
Si presentano pertanto alcuni concetti utili alla comprensione dei temi successivi.

1.5.1 Firma cieca (Blind signature)

La tecnica di firma cieca permette di far firmare un messaggio (un documento) ad un'autorità senza che ne sia rivelato il contenuto. Per spiegare meglio il concetto è

utile il seguente esempio: si inserisca all'interno di una busta il documento e sopra un foglio di carta carbone e si chiuda la busta. Nel momento in cui si firma la busta automaticamente si firma anche il documento ma non si conosce il contenuto dello stesso. Pertanto anche quando il documento viene estratto dalla busta la firma rimane impressa sul documento.
Per i protocolli di voto questa tecnica è utilizzata per poter validare una scheda senza però conoscerne il voto **rispettando in questo modo la privacy dell'elettore**.

1.5.2 Bit commitment

Letteralmente impegno sul bit. Permette di comunicare di aver effettuato una scelta e che questa non è modificabile. Si desidera, tramite una procedura commit(b), assicurare ad una autorità A che si ha in mente il bit b e che non è possibile modificarlo. Però A ha a disposizione solo questa prova, quindi non conosce il valore di b. Successivamente si può rivelare il valore di b e provare ad A che la prova inviata in precedenza corrisponde esattatamente al bit b.

Esempio:
Durante un'asta decido di fare un'offerta, ma la scrivo e la metto in una busta chiusa perchè non voglio che si sappia la mia offerta finchè l'asta non è conclusa; allo stesso tempo però gli altri vogliono essere sicuri che io non possa modificare la mia offerta.

A livello di voto elettronico si applica questa tecnica per garantire che altre entità non possano venire a conoscenza del voto sino a quando l'elettore non spedisce la propria chiave per decriptare il messaggio in cui è definito il voto espresso.

1.5.3 Firma digitale

Come detto in precedenza solamente gli aventi diritto al voto (**eleggibilità**) possono votare. Per fare ciò è innanzitutto fondamentale identificare l'elettore. La firma digitale può essere implementata tramite tecniche di crittografia a chiave asimmetrica. Se si cifra il messaggio con la chiave pubblica di qualcuno solamente costui potrà decifrare il messaggio poichè possessore della chiave segreta. La chiave pubblica è generata in funzione della chiave privata, ma il processo inverso è computazionalmente oneroso. Viceversa se qualcuno cifra un messaggio con la chiave privata, tutti gli altri saranno certi della sua identità poichè sarà possibile decifrarlo con la sua chiave pubblica.

1.5.4 Crittografia a chiave pubblica

Lo scambio di chiavi segrete tra molteplici entità non è la soluzione ideale per implementare un sistema di autenticazione. Per soddisfare la proprietà di eleggibilità è conveniente utilizzare la tecnica di crittografia a chiave pubblica. Le entità che comunicano in questo caso hanno a disposizione due chiavi: una chiave pubblica accessibile da tutte le parti in gioco e una chiave privata posseduta segretamente. Si

ottiene **autenticità** generando un messaggio con la chiave privata. Poichè privata coloro che possiedono la chiave pubblica possono decodificare il messaggio ed essere certi che il messaggio sia stato creato sicuramente dall'entità voluta.

1.5.5 Canale anonimo

Si definisce un canale di comunicazione anonimo un canale che non permette di rintracciare l'identità del mittente di un certo messaggio. Questo dovrebbe garantire l'anonimato (**privacy**) durante le operazioni di voto (una caratteristica essenziale che contraddistingue anche le operazioni di voto tradizionale). Si può realizzare un canale di comunicazione anonimo mediante i cosiddetti **Mix Server**, il cui scopo è quello di scambiare (permutare) i messaggi che vi transitano col fine di eliminare qualsiasi corrispondenza tra messaggio e mittente. Consideriamo per esempio una rete con diversi Mix server M_1, M_2, ..., M_n. Ciascuno di essi è contraddistinto dalla propria chiave pubblica e privata.
Quando il mittente vuole spedire un messaggio, lo cripta con le chiavi pubbliche di tutti i server che compongono la rete e lo spedisce al primo server (M_1). Quando M_1 ha ricevuto un certo numero di messaggi da diversi mittenti rimuove il primo livello di cifratura con la propria chiave privata, ne permuta casualmente l'ordine e li spedisce a M_2. Quindi, M_2 rimuove il proprio livello di cifratura, permuta, spedisce i messaggi a M_3. Così faranno tutti i server che compongono la rete.
In questo modo quando il messaggio arriva al destinatario non è possibile risalire al mittente o quanto meno così dovrebbe essere. Infatti **se solo un server segue il protocollo 'onestamente', la privacy del mittente è garantita**. Pertanto la soluzione migliore è che siano presenti più server per diminuire la probabilità di frode.

1.5.6 Reti ad Hoc

Una rete mobile ad hoc (MANET) può essere facilmente creata con alcuni dispositivi mobili che abbiano interfaccia wireless. Questo permette di creare una rete temporanea. I nodi della rete pertanto comunicano tramite onde radio se e solo se rientrano nello stesso range. Ciascun nodo quindi ha la duplice funzione di trasmettitore e ricevitore.

Le reti ad hoc presentano alcuni difetti:

1. **Limitata memoria**
2. **Limitata capacità di calcolo**
3. **Limitata potenza**

Potenzialmente in una rete così composta un potenziale intrusore può facilmente restare in ascolto e carpire informazioni sensibili oppure ancor più pericoloso, inserire nella rete falsi messaggi o alterare messaggi esistenti. Compiere pertanto i tipici attacchi passivi e/o attivi di un possibile attaccante di una rete. Come spesso se ne parla, per sopperire a questi possibili attacchi si inseriscono nei protocolli alcune tecniche crittografiche quali schemi a chiave simmetrica, a chiave asimmetrica e funzioni one-way. E' possibile approfondire l'argomento con un esempio di voto elettronico presente in ([3]).

1.6 Process calculi

Process calculi (o process algebras) sono una famiglia di linguaggi per la descrizione formale di sistemi concorrenti e distribuiti. Classici esempi sono CSP, CCS, $\pi -$ *calcolo*, spi-calcolo. Nonostante ci siano differenze che contraddistinguono i vari tipi di calcolo questi condividono alcune caratteristiche comuni: le componenti attive sono i **processi**. Ciascun processo può evolvere in maniera indipendente o può comunicare con altri processi mediante uno scambio di messaggi.
Tramite alcuni operatori è possibile combinare tra loro i processi stessi. Si parla di **composizione parallela** o **composizione sequenziale**.
Dati due processi P e Q si rappresenta con P$|$ Q la composizione parallela. In questo caso i processi possono procedere in parallelo, in maniera indipendente oppure mediante uno scambio di messaggi interagire tra loro.
La composizione sequenziale può essere invece utilizzata per dare un ordine a come devono procedere le interazioni tra processi. In genere questo tipo di composizione è utile qualora un processo debba attendere dei dati prima di compiere una determinata azione (se un processo deve inviare dei dati di cui non è in possesso, prima chiaramente si pone in stato di attesa, una volta ricevuti si attiva per spedirli).
Per i nostri obiettivi faremo riferimento al calcolo di processi Lysa in quanto, sviluppato principalmente per modellare protocolli di sicurezza, si presta bene a modellare il nostro caso studio incorporando nativamente operazioni crittografiche quali crittografia a chiave simmetrica e crittografia a chiave asimmetrica. Questo calcolo inoltre rappresenta l'ambiente come **rete globale** e si avvicina molto alle condizioni di uno scenario reale di comunicazione in rete.

1.7 Obiettivi della tesi

Gli obiettivi principali della tesi son quelli di definire in Lysa protocolli di voto elettronico al fine di applicare un'analisi control flow in relazione alle principali proprietà che un protocollo, per elezioni in rete, deve rispettare per garantire un corretto svolgimento delle votazioni.
Capire inoltre se esistono particolari impedimenti, utilizzando la sintassi Lysa, al-

la formalizzazione dei protocolli e successiva analisi (es. la sintassi non prevede nativamente alcune operazioni crittografiche).

1.8 Risultati ottenuti

In una prima fase è stato necessario capire quali strumenti fossero a disposizione per procedere con l'analisi statica dei protocolli. Ci è accorti che Lysa è un'ottima tecnica per descrivere protocolli di sicurezza ed analizzarne le principali proprietà, poichè incorpora nativamente le principali operazioni crittografiche tipiche dei protocolli di comunicazione. Permette quindi di rappresentare scenari reali di comunicazione in rete. Il passo successivo è stato quello di definire in Lysa il protocollo di voto elettronico FOO92.
Infine son state definite due recenti implementazioni, per le quali si è messo in evidenza come siano verificate informalmente le proprietà di maggiore interesse. Questi protocolli non sono stati definiti anche in Lysa, sarà possibile pertanto ottenere dei risultati più significativi esclusivamente dopo aver esteso con la sintassi Lysa il protocollo e definiti i processi dei singoli "giocatori" del protocollo stesso.

1.9 Tecnica di analisi

E' possibile immaginare un programma come una notazione formale per comunicare delle idee tra persone o tra una persona e una macchina. L'analisi statica ([4]) offre uno strumento per catturare informazioni sul comportamento di un programma, potenzialmente derivabili a tempo d'esecuzione ma senza effettivamente eseguire il programma stesso. L'obiettivo è pertanto quello di definire delle descrizioni di alcune proprietà rilevanti applicabile a tutte le potenziali esecuzioni per arbitrari valori in input. Tuttavia, facendo riferimento al teorema di Rice, alcune proprietà dei programmi sono indecidibili. Come conseguenza diretta del teorema, l'analisi dei programmi può garantire solo delle approssimazioni.

1.10 Struttura della tesi

La tesi è organizzata come segue. Nel capitolo 2 vengono descritti sinteticamente le proprietà di sicurezza dei protocolli di comunicazione ed i tipi di attacco generalmente più frequenti. Le motivazioni che hanno indotto negli anni ad avvicinarsi alla tecnica di analisi statica sono descritte nel capitolo 3. E' necessario colmare il gap tra notazione formale e informale per garantire una implementazione più sicura dei protocolli di comunicazione.
Nel capitolo 4 viene introdotto Lysa, sintassi e semantica per analisi control flow. Nel capitolo 5 è descritto uno dei primissimi protocolli di voto elettronico (FOO92) e successivamente definito secondo la sintassi Lysa. Infine nel capitolo 6 si mettono

in evidenza due recenti implementazioni di votazione elettronica per le quali si può osservare come siano verificate informalmente le proprietà di voto (ausilio di smart card, proxy server, funzioni one way).

2

Proprietà di sicurezza informatica

In questo capitolo illustriamo in maniera sintetica le principali proprietà di sicurezza dei protocolli di comunicazione che hanno impatto sulle problematiche legate al voto elettronico. Per una descrizione formale e completa si faccia riferimento alla documentazione presente in ([2],[13],[14]).
I protocolli crittografici permettono mediante appositi algoritmi la protezione di dati sensibili. Qualora due entità debbano comunicare e queste non desiderano che altre entità possano intromettersi nella comunicazione, le due parti cominciano uno scambio di messaggi con l'obiettivo di ottenere questa sicurezza durante tutto l'arco della comunicazione.
Ma cos'è un protocollo crittografico?.
Supponiamo lo scenario in cui le due entità che per convenzione definiamo Alice (A) e Bob (B) vogliano comunicarsi un'informazione privata. La rete sulla quale viaggia l'informazione è insicura, cioè è possibile che una terza entità possa intromettersi nella comunicazione (come può accadere in uno scenario reale quale ad esempio internet).

Nel caso più semplice (crittografia a chiave simmetrica) Alice e Bob condividono una chiave segreta che hanno scelto in maniera sicura in precedenza. Ci si aspetta quindi che qualora Alice invii a Bob un messaggio criptato con la chiave scelta, solo Bob sia in grado di decifrare e leggere il messaggio di Alice. L'obiettivo del protocollo sarebbe pertanto quello di 'nascondere' il messaggio a tutti eccetto Alice (mittente del messaggio) e Bob (destinatario del messaggio).

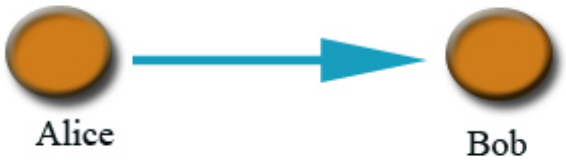

Figura 2.1: Flusso di comunicazione normale di informazioni

E' stato dimostrato (in molti casi la sicurezza di alcuni protocolli è stata violata) che però la crittografia non è la soluzione a tutti i mali poichè molteplici sono i possibili attacchi che entità non del tutto oneste possono compiere all'interno di una rete cosiddetta aperta.
Ovviamente non è possibile definire a priori cosa si intenda per correttezza di un protocollo. Questo perchè molto dipende dal contesto in cui un protocollo viene utilizzato e dagli obiettivi ed intenzioni del protocollo stesso.
Riprendendo brevemente la descrizione del protocollo precedentemente definito possiamo pensare di ritenere corretto il protocollo solamente se al termine della comunicazione solo A e B conoscono il flusso informativo che ha viaggiato sulla rete. Qualora così non fosse sarebbe stata violata l'intenzione del protocollo.
Esistono però diversi aspetti di sicurezza e conseguentemente esistono anche diversi obiettivi.
A seguire vediamo alcuni esempi di possibili attacchi per farci un'idea di quanto sia difficile affrontare il problema della sicurezza e quanto ardua sia la verifica della correttezza dei protocolli che implementiamo.

2.1 Tipi di attacco

Per semplicità ma soprattutto per convenzione rappresentiamo la comunicazione come un flusso informativo che parte da una sorgente e arriva ad una destinazione(fig.2.1). Non necessariamente mittente e destinatario si trovano all'interno di una rete (casi tipici) ma possono ad esempio essere all'interno di un sistema. In base a questa rappresentazione descriviamo i possibili tipi di attacco alla rete e alla sicurezza.

2.1.1 Intercettazione

E' il caso in cui una terza entità ha accesso alla rete di comunicazione (fig.(2.2), può intercettare il messaggio che mittente e destinatario si scambiano, catturare i dati che vi transitano e illegalmente può avere accesso ad informazioni non autorizzate. Può così copiare informazioni, files, e tutto ciò che transita durante la comunicazione. Non viene alterato il flusso informativo per cui per il destinatario è difficile se non quasi impossibile individuare l'azione del potenziale attaccante. Per quest'ultimo motivo è classificato come **attacco di tipo passivo**. E' un tipo di attacco alla segretezza delle informazioni.

2.1.2 Modifica

In questo tipo di attacco (fig.2.3) non solo l'attaccante intercetta il messaggio che transita in rete ma contestualmente lo modifica (inserisce dati contraffatti) e lo invia al destinatario. Il destinatario accetta il messaggio credendo in realtà che i dati siano

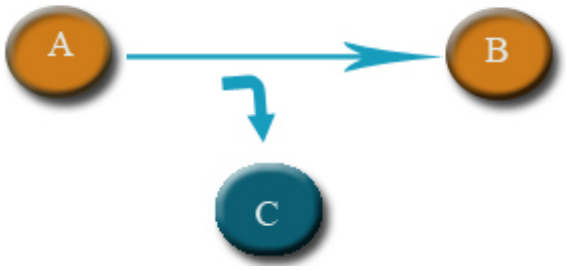

Figura 2.2: Attacco: intercettazione

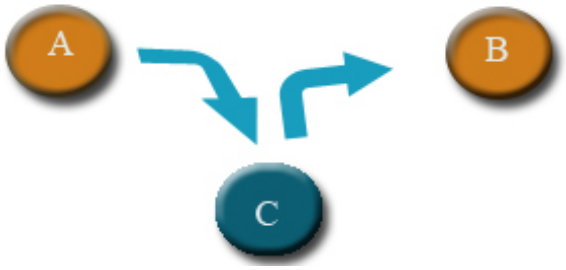

Figura 2.3: Attacco: modifica

stati spediti da un'altra entità. Questo tipo di attacco è classificato come **tipo di attacco attivo** ed è un attacco all'integrità del messaggio stesso.

2.1.3 Falsificazione

L'attaccante in questo caso (fig.2.4) manda al destinatario un messaggio completamente nuovo. A differenza della modifica, la terza entità in gioco non ha nulla da cui partire per fingersi il mittente originale. Questo tipo di attacco è una violazione dell'autenticità ed è un **attacco di tipo attivo**.

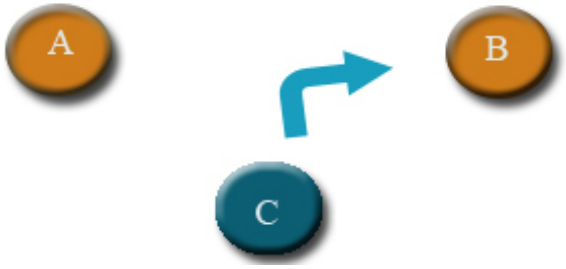

Figura 2.4: Attacco: falsificazione

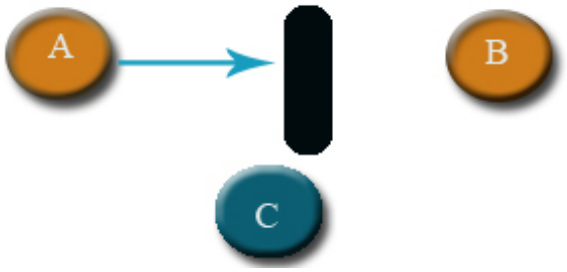

Figura 2.5: Attacco: interruzione

2.1.4 Interruzione

Fanno parte di questo attacco i Denial of Service (DoS). In questo caso viene interrotta (fig.2.5) la rete di comunicazione tra mittente e destinatario (attacco alla disponibilità). Un **attacco di tipo attivo** in cui una o generalmente più entità si mettono in azione per interrompere il flusso informativo.

2.2 Proprietà dei protocolli di sicurezza

Definiamo ora alcune tipiche proprietà che in genere viene richiesto di verificare, sia per specificare meglio alcuni concetti già espressi sia per lasciare un piccolo riferimento, qualora fosse necessario, per i temi che verranno affrontati nei capitoli successivi.

2.2.1 Autenticazione

Con autenticazione si intende che l'origine di un messaggio deve essere correttamente identificata. La falsificazione come abbiamo visto è un attacco all'autenticità. Come vedremo nei nostri protocolli di voto si desidera che un elettore possa votare se ne ha diritto e soprattutto non può farlo più volte. E' indispensabile quindi 'autenticarlo'.

2.2.2 Integrità

Se una terza entità s'intromette nella comunicazione e riesce a modificare, aggiungere, eliminare informazioni da un flusso informativo, viola l'integrità e la correttezza del messaggio stesso. In ambito di voto elettronico desideriamo che nessuno possa ad esempio alterare un voto.

2.2.3 Disponibilità

Una informazione, un messaggio, qualsiasi tipo di flusso deve essere accessibile e disponibile a tutti coloro i quali ne abbiamo l'autorizzazione per visualizzare tali informazioni.

2.2.4 Segretezza

Come già visto in un esempio precedente, tra Alice e Bob ci si aspetta che non si possa intromettere nessun'altra entità che possa così violare la confidenzialità tra Alice e Bob. Pertanto le informazioni devono essere accessibili solamente a coloro i quali ne abbiamo la corretta autorizzazione.

3

Analisi statica dei protocolli di sicurezza

3.1 Protocolli di sicurezza

Questo capitolo sottolinea come sia indispensabile descrivere con molta accuratezza i protocolli di comunicazione: definire con precisione le intenzioni, le finalità, gli obiettivi, le regole per ogni entità coinvolta col fine implementare protocolli di comunicazione sicuri. Lo stesso tipo di approccio può essere così utilizzato per descrivere i protocolli di voto elettronico. Per un approfondimento è possibile far riferimento a ([4],[8],[10]).
A livello generale possiamo definire con sicurezza la possibilità di proteggere alcune informazioni, sia nel caso in cui esse siano memorizzate all'interno di un sistema informativo sia esse vengano trasmesse su reti. Con la diffusione delle reti che condividono informazioni, la sicurezza è diventata sempre più un'esigenza indispensabile. Si sono sviluppati pertanto diversi protocolli per garantire diverse proprietà fondamentali durante le comunicazioni: segretezza, autenticità, integrità dei messaggi, disponibilità, ... ecc.
Per la descrizione di questi protocolli si utilizzano in genere dei modelli (**notazioni informali**) che rappresentano le entità coinvolte durante la comunicazione e i messaggi che esse si scambiano per garantire queste proprietà.
L'analisi dei protocolli è però un problema spesso arduo in quanto non è semplice specificare esplicitamente le proprietà che ci si aspetta siano rispettate.
Si è scoperto infatti che alcuni protocolli ritenuti sicuri per molti anni erano in realtà non sicuri proprio perchè le descrizioni erano imprecise (**spesso assunzioni implicite**), gli **obiettivi mal definiti** e non erano focalizzati con dettaglio i **controlli obbligatori** per il protocollo affinchè fosse garantita sicurezza. Oltre a ciò spesso non erano specificate neppure le **azioni interne** che coinvolgevano le singole entità della comunicazione.

Protocolli non ben implementati (molti sono gli errori che può commettere un programmatore) possono essere sfruttati per violare la sicurezza.
Sono stati proposti alcuni approcci per formalizzare con più dettaglio le intenzioni di un protocollo di sicurezza. A seguire vedremo un esempio e valuteremo

come estendere la descrizione per ottenere maggiori informazioni per una corretta implementazione del protocollo.

Consideriamo ad esempio la versione seguente del protocollo Wide Mounthed Frog (WMF) il cui scopo è quello di stabilire una chiave simmetrica di sessione K tra due entità A e B che condividono rispettivamente le chiavi K_a e K_b con un Server fidato S.

1. A $\rightarrow$ S: A, $\{\mathrm{B,K}\}_{K_a}$

2. S $\rightarrow$ B: $\{\mathrm{A,K}\}_{K_b}$

3. A $\rightarrow$ B: $\{\mathrm{m_1,m_2,..,m_k}\}_K$

Il protocollo viene descritto mediante una lista dei messaggi che le diverse parti in gioco si scambiano. Per stabilire la chiave di sessione K l'entità A invia ad S il proprio identificativo, una nuova chiave K di sessione e l'identificatore del destinatario criptato con la chiave che condivide con il server (1). Nel secondo messaggio il server invia a B un messaggio criptato con la chiave che condivide con B specificando mittente e chiave K. Nel terzo ed ultimo messaggio A invia a B il messaggio criptandolo con la chiave K di sessione che ora conosce anche B. L'entità B conosce la chiave K e quindi riesce e decriptare il messaggio e leggerlo.

L'obiettivo principale è quello di **colmare il gap tra notazione informale e specifica formale**. Cioè dare tutte le informazioni necessarie affichè il programmatore possa implementare correttamente il protocollo.

La descrizione data precedentemente del protocollo è piuttosto imprecisa e lascia molti dettagli ed assunzioni implicite. Soprattutto quali azioni devono compiere le entità quando ricevono un messaggio e/o prima di inviare un proprio messaggio.
Un primo passo per avvicinarci ai nostri obiettivi per il voto elettronico è quello di comprendere come rendere più evidenti le descrizioni dei protocolli distinguendo innanzitutto tra input e output, tra criptazione e decriptazione. Quali controlli sono fondamentali quando si riceve un messaggio?.
I messaggi sono così arricchiti con indirizzo sorgente e indirizzo del destinatario. La struttura in generale di ogni singolo messaggio per le nostre descrizioni sarà del tipo: sorgente, destinazione, $\mathrm{messaggio_1}$, $\mathrm{messaggio_2}$,..., $\mathrm{messaggio_k}$, seguito da assunzioni e/o controlli tra parentesi quadre.

Il protocollo WMF, secondo il nuovo schema e con l'obiettivo di esplicitare tutte le singole operazioni, viene sviluppato come segue:

1.A	→		:A,S,A,$\{B,K\}_{K_a}$	[Assumiamo sia K una nuova chiave]
2.	→	S	:x_a,x_s,x'_A,x	[verifica che x_s=S, x_a=x'_A]
3.		S	:decrypt x as $\{x_b, x_k\}_{K_A}$	
4.S	→		:S,x_b,$\{x_a,x_K\}kx_B$	
5.	→	B	:y_S,y_B,y	[verifica che y_B=B]
6.		B	:decrypt y as $\{y_A,y_K\}_{K_B}$	
7.A	→		:A,B,$\{m_1,m_2,..,m_k\}_K$	
8.	→	B	:z_A,z_B,z	[verifica che z_B= B, z_A=y_A]
9.		B	:decrypt z as $\{z_1,z_2,..,z_k\}_{y_k}$	

La descrizione del protocollo viene quindi estesa e per ciascun messaggio si considerano ora l'informazione che il mittente invia al destinatario (1-4-7), il controllo dell'integrità del messaggio (2-5-8) e l'azione interna che deve effettuare il destinatario del messaggio(3-6-9). Ad esempio la prima riga rappresenta l'invio del messaggio da parte di A verso il server fidato S. Dopo l'input, S controlla la correttezza e consistenza del messaggio (punti 2-5-8). Il server S come azione decripta il messaggio inviato da A utilizzando la chiave che condivide con A.

I controlli sono interni al destinatario, non vi è una conoscenza a priori circa il mittente del messaggio.

Un ulteriore passo verso una descrizione di un protocollo per poter essere analizzato con maggiore facilità, include un'esplicito inserimento degli **obiettivi di sicurezza** all'interno del protocollo stesso. Si inseriscono cosiddette **asserzioni** per specificare le proprietà fondamentali per il protocollo che andiamo ad analizzare.
Ad esempio qualora volessimo analizzare il problema dell'autenticità ci aspettiamo che l'informazione sia spedita e ricevuta esclusivamente dalle entità specificate nel protocollo stesso.
Completiamo pertanto la nostra descrizione del protocollo aggiungendo origine (mittente) e destinazione (destinatario) del messaggio criptato. Ritornando per semplicità al nostro esempio, rappresentiamo con $\{B,K\}k_A$ [dest S] il fatto che il messaggio criptato è destinato esclusivamente ad S.

Fatte queste considerazioni possiamo descrivere il nostro protocollo WMF con la seguente estensione:

A	→		:A,S,A,$\{B,K\}_{K_a}$ [dest S]	[Assumiamo sia K una nuova chiave]
	→	S	:x_a,x_s,x'_A,x	[verifica che x_s=S, x_a=x'_A]
		S	:decrypt x as { x_b, x_k }$_{K_A}$ [orig x_a]	
S	→		:S,x_b,$\{x_a,x_K\}kx_B$ [dest x_B]	
	→	B	:y_S,y_B,y	[verifica che y_B=B]
		B	:decrypt y as $\{y_A,y_K\}_{K_B}$ [orig S]	
A	→		:A,B,$\{m_1,m_2,..,m_k\}_K$[dest B]	
	→	B	: z_A,z_B,z	[verifica che z_B= B, z_A=y_A]
		B	: decrypt z as $\{z_1,z_2,..,z_k\}_{y_k}$ [orig z_a]	

Questo stesso approccio sarà utilizzato per formalizzare i protocolli di voto elettronico, ma in generale è un approccio che permette un'analisi formale delle proprietà dei protocolli di sicurezza. Chiarisce meglio obiettivi e assunzioni e si pone meglio come strumento base per una futura implementazione dei protocolli corretti.

4

Lysa

In letteratura, il process calculus ha un ruolo importante come framework per modellare e ragionare sui sistemi concorrenti e distribuiti. I sistemi sono specificati come espressioni del calcolo, **i processi**. A ciascun process calculus viene associata una semantica operazionale che descrive il comportamento del processo, quindi descrive il tipo di operazioni che può eseguire. Il calcolo di processi Lysa è stato sviluppato per modellare i protocolli di sicurezza. A differenza di altri approcci in Lysa tutte le comunicazioni prendono luogo in una **rete globale**. Questo linguaggio si presta bene alla rappresentazione dei protocolli di sicurezza e incorpora le principali operazioni crittografiche tipiche dei protocolli di sicurezza.

4.1 Sintassi

Lysa consiste di termini e processi. La versione originale di Lysa è stata sviluppata per modellare e analizzare protocolli per lo scambio di chiavi e così incorpora le principali operazioni crittografiche quali crittografia simmetrica e asimmetrica. Nella tabella seguente (tab.4.1) è possibile visualizzare la sintassi del calcolo che ci permette di modellare anche il nostro caso studio. Una espressione E $\in$ Expr può rappresentare una variabile, un nome oppure un'operazione di crittografia (criptazione o decriptazione).

L'insieme Expr ingloba due insiemi disgiunti Name (nonce, chiavi simmetriche, coppie di chiavi per crittografia asimmetrica) e Var (solamente variabili). Le espressioni rimanenti rappresentano la crittografia e decriptazione di una k-tupla di elementi in cui E_0 rappresenta la chiave simmetrica o la coppia di chiavi per crittografia asimmetrica.

Tuttavia, non potendo gestire nativamente l'operazione crittografica di firma cieca, vedremo come estendere la sintassi per poter modellare anche il nostro caso studio. Per approfondire la dimostrazione dell'estensione di Lysa per incorporare la firma cieca si faccia riferimento a ([7]).

- Il processo $\langle E_1, ..., E_k \rangle$.P invia una k-tupla di valori sulla rete globale e nel momento in cui l'operazione ha successo, procede come P.

Tabella 4.1: Sintassi del calcolo Lysa

E::=	termini
n	nomi
x	variabile
m^+	chiave privata
m^-	chiave pubblica
$\{E_1, ..., E_k\}_{E_0}$	crittografia simmetrica
$\{\mid E_1, ..., E_k \mid\}_{E_0}$	crittografia asimmetrica
P::=	processi
$\langle E_1, ..., E_k \rangle$.P	output
$(E_1, ..., E_j; x_{j+1}, ..., x_k)$.P	input
decrypt E as $\{E_1, ..., E_j; x_{j+1}, ..., x_k\}_{E_0}$ in P	decriptazione simmetrica
decrypt E as $\{\mid E_1, ..., E_j; x_{j+1}, ..., x_k \mid\}_{E_0}$ in P	decriptazione asimmetrica
(v n)P	restrizione
($v \pm m$)P	restrizione rispetto ad una coppia
$P_1 \mid P_2$	composizione parallela
!P	replicazione
0	nil

- Il processo $(E_1, ..., E_k; x_{j+1}, ..., x_k)$.P legge una k-tupla di valori dalla rete. Verifica se i primi j valori sono identici ad $E_1, ..., E_j$, se è così i rimanenti $k - j$ valori sono legati alle variabili $x_{j+1}, ..., x_k$.

- Il processo decrypt E as $\{E_1, ..., E_j; x_{j+1}, ..., x_k\}_{E_0}$ in P denota la decrittazione simmetrica e si comporta in modo analogo al processo di input. Se la chiave crittografica è uguale ad $_{E_0}$, il processo decripta la k-tupla , allora verifica se i valori sono identici a $E_1, ..., E_j$. Se è così i rimanenti $k - j$ valori sono legati alle variabili $x_{j+1}, ..., x_k$ e continua come P (scope delle variabili).

- Il processo decrypt E as $\{\mid E_1, ..., E_j; x_{j+1}, ..., x_k \mid\}_{E_0}$ in P denota la decrittazione asimmetrica. Si comporta in modo analogo alla decriptazione simmetrica eccezion fatta per la chiave (coppia di chiavi).

- Il processo (vn)P genera un nuovo nome n e procede come P (n nello scope di P).

- Il processo (v ± m) genera una coppia di chiavi (pubblica e privata) e procede come P.

- Il processo P1| $P2$ rappresenta due processi che procedono in parallelo e che possono sincronizzarsi attraverso uno scambio di messaggi, oppure procedere in maniera indipendente.

Tabella 4.2: Funzione fn(P) per i nomi liberi

fn(n)	$\stackrel{\text{def}}{=} \{n\}$
fn(m^+)	$\stackrel{\text{def}}{=} \{m^+\}$
fn(m^-)	$\stackrel{\text{def}}{=} \{m^-\}$
fn(x)	$\stackrel{\text{def}}{=} 0$
fn($\{E_1,...,E_k\}_{E_0}$)	$\stackrel{\text{def}}{=}$fn(E_0)$\bigcup \ldots \bigcup$fn(E_k)
fn($\{\|E_1,...,E_k\|\}_{E_0}$)	$\stackrel{\text{def}}{=}$fn(E_0)$\bigcup \ldots \bigcup$fn(E_k)
fn($\langle E_1,...,E_k\rangle$.P)	$\stackrel{\text{def}}{=}$fn(E_1)$\bigcup \ldots \bigcup$fn(E_k)$\bigcup$fn(P)
fn((E_1,...,E_j;x_{j+1},...,x_k).P)	$\stackrel{\text{def}}{=}$fn(E_1)$\bigcup \ldots \bigcup$fn(E_k)$\bigcup$fn(P)
fn((decrypt E as $\{E_1,...,E_j;x_{j+1},...,x_k\}$)$_{E_0}$)	$\stackrel{\text{def}}{=}$fn(E)$\bigcup$fn(E_0)$\bigcup \ldots \bigcup$fn(E_j)$\bigcup$fn(P)
fn((decrypt E as $\{\|E_1,...,E_j;x_{j+1},...,x_k\|\}$)$_{E_0}$)	$\stackrel{\text{def}}{=}$fn(E)$\bigcup$fn(E_0)$\bigcup \ldots \bigcup$fn(E_j)$\bigcup$fn(P)
fn((vn)P)	$\stackrel{\text{def}}{=}$fn(P)$\setminus\{n\}$
fn((v±m)P)	$\stackrel{\text{def}}{=}$fn(P)$\setminus \{ m^+, m^-\}$
fn(P1|P2)	$\stackrel{\text{def}}{=}$fn(P1)$\bigcup$fn(P2)
fn(!P)	$\stackrel{\text{def}}{=}$fn(P)
fn(0)	$\stackrel{\text{def}}{=}0$

- Il processo !P opera come un numero arbitrario di processi in parallelo.
- 0 rappresenta un processo inattivo oppure un processo che non fa nulla.

Definiamo "**binder**" un costrutto che crea nuovi nomi o variabili che hanno visibilità nel resto del processo. Ad esempio il prefisso (vn) nel processo (vn)P è un binder poichè crea un nuovo nome che ha visibilità nel processo P. Definiamo quindi un nome libero o variabile libera, un nome oppure una variabile che non hanno visibilità in nessun processo. La funzione fn(P) colleziona tutti i nomi che sono liberi nel processo P. E' definita in maniera induttiva nella tabella (4.2). In modo analogo la funzione fv(P) colleziona tutte le variabili libere nel processo P. Questa funzione è definita induttivamente nella tabella (4.3).

4.2 Semantica

Lysa presenta una semantica di riduzione che descrive l'evoluzione del processo passo dopo passo, utilizzando una relazione di riduzione tra due processi: P$\rightarrow P'$. Se la relazione di riduzione vale allora il processo P può evolvere nel processo P' usando le regole visibili nella tabella (4.8), che mostra una definizione induttiva delle relazioni utilizzando assiomi e regole di inferenza.
La congruenza strutturale tra due processi, P $\equiv P'$ rappresenta il fatto che P è uguale a P' ad eccezione di aspetti sintattici che non interferiscono nel modo in cui i processi evolvono.

Tabella 4.3: Funzione fv(P) per le variabili libere

fv(n)	$\stackrel{\text{def}}{=}0$
fv(m^+)	$\stackrel{\text{def}}{=}0$
fv(m^-)	$\stackrel{\text{def}}{=}0$
fv(x)	$\stackrel{\text{def}}{=}$ {x}
fv($\{E_1,...,E_k\}_{E_0}$)	$\stackrel{\text{def}}{=}$fv(E_0)$\bigcup\ldots\bigcup$fv(E_k)
fv($\{\|E_1,...,E_k\|\}_{E_0}$)	$\stackrel{\text{def}}{=}$fv(E_0)$\bigcup\ldots\bigcup$fv(E_k)
fv($\langle E_1,...,E_k\rangle$.P)	$\stackrel{\text{def}}{=}$fv(E_1)$\bigcup\ldots\bigcup$fv(E_k)$\bigcup$fv(P)
fv((E_1,...,E_j;x_{j+1},...,x_k).P)	$\stackrel{\text{def}}{=}$fv(E_1)$\bigcup\ldots\bigcup$fn(E_j)$\bigcup$fv(P)\{x_{j+1},...,x_k}
fv((decrypt E as {E_1,...,E_j;x_{j+1},...,x_k})$_{E_0}$)	$\stackrel{\text{def}}{=}$fv(E)$\bigcup$fn(E_0)$\bigcup\ldots\bigcup$fv(E_j)$\bigcup$fv(P)\{x_{j+1},...,x_k}
fv((decrypt E as {|E_1,...,E_j;x_{j+1},...,x_k|})$_{E_0}$)	$\stackrel{\text{def}}{=}$fv(E)$\bigcup$fn(E_0)$\bigcup\ldots\bigcup$fv(E_j)$\bigcup$fv(P)\{x_{j+1},...,x_k}
fv((vn)P)	$\stackrel{\text{def}}{=}$fv(P)\{n}
fv((v±m)P)	$\stackrel{\text{def}}{=}$fv(P)\ { m^+,m^-}
fv(P1|P2)	$\stackrel{\text{def}}{=}$fv(P1)$\bigcup$fv(P2)
fv(!P)	$\stackrel{\text{def}}{=}$fv(P)
fv(0)	$\stackrel{\text{def}}{=}0$

La congruenza strutturale è definita come la più piccola relazione che soddisfa le regole della tabella 4.4. La regola (Com) rappresenta la composizione parallela di un processo output e un processo d'input. Questo significa che la comunicazione tra i processi può esserci solo se i due processi procedono in parallelo. Se i primi j valori spediti $V_1,\ldots,V_j$ sono identici ai primi j valori $V_1^{'},\ldots,V_j^{'}$ che il processo input si aspetta, allora le variabili sono sosituite con i valori $V_{j+1},\ldots,V_k$. Le regole (Dec), (Adec), (Asig) sono utilizzate per decriptare i messaggi con chiave simmetrica, chiave privata e chiave pubblica. Come per la regola (Com) anche in questi casi, i primi j valori criptati devono essere identici ai primi j valori che il destinatario del messaggio si aspetta.
Le regole (New) e (ANew) restringono la visibilità dei nomi generati, quindi questi sono visibili esclusivamente nel rispettivo processo.
La regola (Par) è la composizione parallela che può evolvere in un nuova composizione parallela nel quale uno dei due processi coinvolti evolve e l'altro rimane inalterato.
Infine la regola (Congr) permette di applicare la relazione di riduzione a un qualsiasi processo che è strutturalmente congruente a processi trovati in altre regole.

Si definiscono due processi P_1 e P_2 $\propto$-equivalenti quando sono identici eccetto nella scelta dei nomi.
Nella tabella 4.7 è possibile visualizzare la definizione della relazione di equivalenza. Si osservi come la sostituzione P $[n_1\mapsto n_2]$ sostituisce con n_2 tutte le occorrenze libere di n_1.

Tabella 4.4: Congruenza Strutturale

P≡P
$P_1 \equiv P_2 \Rightarrow P_2 \equiv P_1$
$P_1 \equiv P_2 \wedge P_3 \equiv P_3 \Rightarrow P_1 \equiv P_3$
$P_1 \mid P_2 \equiv P_2 \mid P_1$
$P_1 \equiv P_2 \Rightarrow \langle\ E_1,...,E_k\ \rangle.P_1 \equiv \langle\ E_1,...,E_k\ \rangle.P_2$
$(\mathrm{vn})P_1 \equiv (\mathrm{vn})P_2$
$(P_1 \mid P_2) \mid P_3 \equiv P_1 \mid (P_2 \mid P_3)$
P $\mid$ 0 ≡ P
!P ≡ P $\mid$!P
$(\mathrm{vn})0 \equiv 0$
$(\mathrm{v}n_1)(\mathrm{v}n_2) \equiv (\mathrm{v}n_2)(\mathrm{v}n_1)$

Infine definiamo i valori V ∈ Val utilizzati nella riduzione come espressioni senza variabili x ∈ Var.

$$
\begin{array}{rcl}
\mathrm{V} :: = & & \mathrm{n} \\
& \mid & m^{+} \\
& \mid & m^{-} \\
& \mid & V_1,\ldots,V_k \\
& \mid & V_1,\ldots,V_k
\end{array}
$$

La relazione di riduzione permette di descrivere come il processo evolve in un altro processo. E' definita come la più piccola relazione che soddisfa le regole nella tabella. Si utilizza un reference monitor per verificare ogni passo prima che ne sia permessa l'esecuzione. Il reference monitor può essere attivo oppure spento. Nel primo caso alcune proprietà sono verificate a tempo d'esecuzione, qualora l'esito sia negativo viene interrotta l'esecuzione. Nel secondo caso non esistono dei requisiti minimi per l'esecuzione.

Per le regole di riduzione viene applicata una regola di sostituzione definita come P [V/x]. Questa funzione sostituisce una variabile x con il valore V.

4.3 Semantica di meta livello

Ciascun protocollo può essere utilizzato in un'ampia varietà di scenari, non necessariamente conosciuto in fase di progettazione e sviluppo.
Spesso è necessario fare un certo numero di assunzioni, come ad esempio il numero di possibili partecipanti contemporaneamente oppure se ciascun partecipante la comunicazione potrà assumere simultaneamente diversi ruoli allo stesso tempo. Per come è stata definita sintassi e semantica fin ora ci accorgiamo che è difficile rappresentare queste considerazioni.

Tuttavia è stato introdotta una estensione di Lysa che può essere usata per descrivere differenti scenari in cui molteplici entità possono eseguire contemporaneamente il protocollo (molti mittenti e/o destinatari). Si caricano diverse copie dei processi e si indicizzano variabili e nomi per rendere i processi univoci. Ciò permette di realizzare un ambiente realistico per le simulazioni. I termini sono identici a quelli definiti in precedenza con l'eccezion fatta che ora tutto è indicizzato. Per convenzione si scriverà $\overline{i}$ per rappresentare la sequenza $i_1, \ldots, i_k$. Rispetto ai termini presentati in precedenza, i processi di meta livello aggiungono tre nuovi costrutti. Tutti loro incorporano un insieme S di indicizzazione, che include l'insieme di variabili X.

1. $|_{i \in S} MP$: la composizione parallela di istanze del processo MP, in cui l'indice i è sostituito da un elemento dell'insieme S.

2. let $X \subseteq S$ in MP: definisce un insieme di identificatori X che sarà usato nel processo M, nel quale $X \in SetId$ fa riferimento ad un sottoinsieme di valori dell'insieme di indici S nel processo MP. Per ragioni di consistenza ogni qualvolta X è istanziato ad un sottoinsieme di S, la stessa istanziazione sarà applicata a tutte le occorrenze X nel processo M.

3. $(v_{\overline{i} \in \overline{S}} n_{\overline{ai}})MP$: il processo descrive la restrizione di tutti i nomi $n_{\overline{ai}}$;$\overline{a}$ è un prefisso dell'indice che può essere anche vuoto;

4. $(\pm v_{\overline{i} \in \overline{S}} m_{\overline{ai}})MP$:il processo descrive la restrizione di tutte le coppie di chiavi $m^{+}_{\overline{ai}}$ e $m^{-}_{\overline{ai}}$; $\overline{a}$ è un prefisso dell'indice che può essere anche vuoto;

5. $\langle ME_1, \ldots, ME_k \rangle.MP$: il processo invia una k-tupla di valori sulla rete globale. Se il messaggio ragigunge la destinazione procede come MP.

6. $(ME_1, \ldots, ME_j; mx_{j+1}, \ldots, mx_k).MP$: il processo legge una k-tupla di valori, verifica se i valori sono identici a $ME_1, \ldots, ME_j$ e se il controllo ha successo allora le variabili $mx_{j+1}, \ldots, mx_k$ assumono i k-j valori. Successivamente procede come MP.

7. decrypt ME as $\{ME_1, \ldots, ME_j; mx_{j+1}, \ldots, mx_k\}_{ME_0} in MP$: il processo rappresenta la decriptazione asimmetrica. Verifica se la chiave per criptare è identica a E_0. Una volta decriptata la k-tupla si comporta come il processo precedente.

8. decrypt ME as $\{|ME_1, \ldots, ME_j; mx_{j+1}, \ldots, mx_k|\}_{ME_0} in MP$: il processo rappresenta la decriptazione asimmetrica e si comporta come la decriptazione simmetrica. La differenza consiste nella chiave. In questo caso è un coppia di chiavi, pubblica e privata.

9. $(v_{\overline{i}})MP$: il processo crea k nuovi nomi, n_i con $i \in [i, \ldots, k]$ e continua come MP.

Tabella 4.5: Sintassi del calcolo Lysa esteso

E::=	termini
n	nomi
x	variabile
m^+	chiave privata
m^-	chiave pubblica
$\{E_1, ..., E_k\}E_0$	crittografia simmetrica
$\{\mid E_1, ..., E_k \mid\}E_0$	crittografia asimmetrica
$\|E_1, ..., E_k\|E_0$	**firma cieca**

10. $(v \pm m_{\bar{i}})$: il processo crea k coppie di nuovi chiavi m_i^+ e m_i^- e continua come MP.

11. $MP_1|MP_2$: rappresenta due sottoprocessi di meta livello che procedono in parallelo, che come già specificato più volte possono sincronizzarsi tramite la comunicazione (es. messaggi), oppure operare in maniera indipendente.

12. !MP: il processo agisce come un numero arbitrario di processi MP composti in parallelo.

13. 0: rappresenta un processo che non fa nulla o un processo inattivo.

4.4 Estensione

La firma cieca, come già detto in precedenza, è un costrutto non incorporato nativamente nel calcolo Lysa. Per i nostri obiettivi è indispensabile estendere quindi sia la sintassi sia la semantica. Assumendo la crittografia perfetta l'operazione crittografica di firma cieca segue le seguenti regole:

1. (Unblind1) unblind(b, blind(b,*msg*)) = msg

2. (Unblind2) unblind(b, sign(s,blind(b,*msg*))) = sign(s,*msg*)

Una relazione di istanziazione, $MP \rightarrow_\gamma P$, è definita per descrivere come un processo P è un'istanza di un processo di meta livello MP (4.9).

4.5 Control Flow Analysis

L'obiettivo è quello di collezionare le informazioni sul comportamento del processo e memorizzarle in alcune strutture dati definite componenti dell'analisi. L'analisi deve

Tabella 4.6: Sintassi del calcolo Lysa esteso

P::=	processi
$\langle E_1, ..., E_k \rangle$.P	output
$\langle E_1, ..., E_k; x_{j+1}, ..., x_k \rangle$.P	input
decrypt E as $\{E_1, ..., E_j; x_{j+1}, ..., x_k\}E_0$ in P	decriptazione simmetrica
decrypt E as $\{\mid E_1, ..., E_j; x_{j+1}, ..., x_k \mid\}E_0$ in P	decriptazione asimmetrica
unblind E as $\|E_1, ..., E_j; x_{j+1}, ..., x_k\|E_0$	**unblinding**
(v n)P	restrizione
($v \pm m$)P	restrizione rispetto ad una coppia
$P_1 \mid P_2$	composizione parallela
!P	replicazione
0	nil

Tabella 4.7: α - equivalenza

P$\stackrel{\alpha}{\equiv}$P
$P_1 \stackrel{\alpha}{\equiv} P_2 \Rightarrow P_2 \stackrel{\alpha}{\equiv} P_1$
$P_1 \stackrel{\alpha}{\equiv} P_2 \wedge P_2 \stackrel{\alpha}{\equiv} P_3 \Rightarrow P_1 \stackrel{\alpha}{\equiv} P_3$
(vn_1)P $\stackrel{\alpha}{\equiv}$ (vn_2)(P$[n_1 \mapsto n_2]$) se $n_2 \notin$ fn(P)
(v$\pm m_1$)P $\stackrel{\alpha}{\equiv}$ (v$\pm m_2$)(P$[m_1^+ \mapsto m_2^+, m_1^- \mapsto m_2^-]$) se $m_2^+, m_2^- \notin$ fn(P)

Tabella 4.8: Semantica

(Com)	$\frac{\bigwedge_{i=1}^{j} V_i = V_i'}{\langle V_1,...,V_k \rangle.P \mid (V_1^k,...,V_j';x_{j+1},...,x_k).P' \rightarrow_R P \mid P'[V_{j+1}/x_{j+1},...,V_k/x_k]}$
(Dec)	$\frac{\bigwedge_{i=0}^{j} V_i = V_i'}{decrypt\{V_1,...,V_k\}_{v_0} as \{V_1',...,V_j';x_{j+1},...,x_k\}_{V_0'} in P \rightarrow_R P[V_{j+1}/x_{j+1},...,V_k/x_k]}$
(ADec)	$\frac{\bigwedge_{i=1}^{j} V_i = V_i'}{decrypt\{\mid V_1,...,V_k \mid\}_{m^-} as \{\mid V_1',...,V_j';x_{j+1},...,x_k \mid\}_{m^-} in P \rightarrow_R P[V_{j+1}/x_{j+1},...,V_k/x_k]}$
(ASig)	$\frac{\bigwedge_{i=1}^{j} V_i = V_i'}{decrypt\{\mid V_1,...,V_k \mid\}_{m^-} as \{\mid V_1',...,V_j';x_{j+1},...,x_k \mid\}_{m^-} in P \rightarrow_R P[V_{j+1}/x_{j+1},...,V_k/x_k]}$
(New)	$\frac{P \rightarrow_R P'}{(vn)P \rightarrow_R (vn)P')}$
(ANew)	$\frac{P \rightarrow_R P'}{(v \pm m)P \rightarrow_R (v \pm m)P')}$
(Par)	$\frac{P_1 \rightarrow_R P_1'}{P_1 \mid P_2 \rightarrow_R P_1' \mid P_2}$
(Congr)	$\frac{P \equiv P' \wedge P' \rightarrow_R P'' \wedge P'' \equiv P'''}{P \rightarrow_R P'''}$

Tabella 4.9: $MP \rightarrow_\gamma P$

(ILet)	$\frac{MP[X \mapsto S']\Rrightarrow P}{letX \subseteq S \in MP \Rrightarrow P}$
(IIPar)	$\frac{MP[\tilde{i} \mapsto a_1] \Rrightarrow P_1 ... MP[i \mapsto a_k] \Rrightarrow P_2}{\mid_{i \in \{a_1,...,a_k\}} MP \Rrightarrow P_1 \mid ... \mid P_k}$
(IInew)	$\frac{MP \Rrightarrow P}{(v_{\tilde{i} \in \{\overline{a_1},...,\overline{a_k}\}} n_{\overline{ai}}) MP \Rrightarrow (vn_{\overline{aa_1}}) ... (vn_{\overline{aa_k}}) P}$
(IIANew)	$\frac{MP \Rrightarrow P}{(v_{\pm\tilde{i} \in \{\overline{a_1},...,\overline{a_k}\}} m_{\overline{ai}}) MP \Rrightarrow (v \pm m_{\overline{aa_1}}) ... (v \pm m_{\overline{aa_k}}) P}$
(IOut)	$\frac{MP \Rrightarrow P}{\langle ME_1,...,ME_k \rangle . MP \Rrightarrow \langle ME_1,...,ME_k \rangle . P}$
(IInp)	$\frac{MP \Rrightarrow P}{(ME_1,...,ME_j; mx_{j+1},...,mx_k) . MP \Rrightarrow (ME_1,...,ME_j; mx_{j+1},...,mx_k) . P}$
(IDec)	$\frac{MP \Rrightarrow P}{decryptMEas\{ME_1,...,ME_j; mx_{j+1},...,mx_k\}_{ME_0} inMP \Rrightarrow decryptMEas\{ME_1,...,ME_j; mx_{j+1},...,mx_k\}_{ME_0} inP}$
(IADec)	$\frac{MP \Rrightarrow P}{decryptMEas\{\mid ME_1,...,ME_j; mx_{j+1},...,mx_k \mid\}_{ME_0} inMP \Rrightarrow decryptMEas\{\mid ME_1,...,ME_j; mx_{j+1},...,mx_k \mid\}_{ME_0} inP}$
(Inew)	$\frac{MP \Rrightarrow P}{(v \pm n_{\overline{a}}) MP \Rrightarrow (v \pm n_{\overline{a}}) P}$
(IANew)	$\frac{MP \Rrightarrow P}{(v \pm m_{\overline{a}}) MP \Rrightarrow (v \pm m_{\overline{a}}) P}$
(IRep)	$\frac{MP \Rrightarrow P}{!MP \Rrightarrow !P}$
(IPar)	$\frac{MP_1 \Rrightarrow P_1 \; MP_2 \Rrightarrow P_2}{MP_1 \mid MP_2 \Rrightarrow P_1 \mid P_2}$
(INil)	$0 \Rrightarrow 0$

essere finita, quindi l'analisi statica è obbligatoriamente indirizzata per calcolare delle approssimazioni piuttosto che calcolare risposte esatte. L'analisi può restituire falsi positivi. Per la nostra analisi useremo il formalismo Flow Logic per la specifica e le dimostrazioni.

L'analisi deve essere finita (terminare). L'analisi di tipo Control Flow astrae le esecuzioni e rappresenta solo alcuni aspetti del comportamento di un processo. Si prova la correttezza dell'analisi dimostrando che le componenti d'analisi sono tali che le proprietà che loro rappresentano sono valide anche quando evolve il processo.

4.5.1 Componenti dell'analisi

Le componenti dell'analisi di Lysa sono κ e ρ. La componente κ memorizza le tuple di valori canonici che corrispondono alle tuple di valori che possono essere comunicate sulla rete durante l'esecuzione di un processo. Formalmente abbiamo: $\kappa \in P(\lfloor Val \rfloor^*)$ La componente ρ memorizza un insieme di valori canonici, per ogni variabile, che corrisponde all'insieme di valori che ogni variabile può assumere durante l'esecuzione del processo. Formalmente abbiamo: $\rho : \lfloor Var \rfloor \rightarrow P(\lfloor Val \rfloor)$ Il risultato dell'analisi di un processo P è quindi una coppia (κ,ρ). La prima componente rappresenta un ambiente astratto che restituisce le informazioni che da informazioni circa l'insieme di valori canonici. La seconda componente restituisce informazioni circa le tuple di valori che possono viaggiare sulla rete. Più precisamente abbiamo: $\rho, \kappa \models P$ che esprime che ρ e κ sono una valida stima del processo P.

Tabella 4.10: Analisi di termini e processi

(AN)	$\rho \models n : \vartheta$	sse $\lfloor n \rfloor \in \vartheta$
(ANp)	$\rho \models m^+ : \vartheta$	sse $\lfloor m^+ \rfloor \in \vartheta$
(ANm)	$\rho \models m^- : \vartheta$	sse $\lfloor m^- \rfloor \in \vartheta$
(AVar)	$\rho \models x : \vartheta$	sse $\rho(\lfloor x \rfloor) \subseteq \vartheta$
(AEnc)	$\rho \models \{E_1, \ldots, E_k\}_{E_0} : \vartheta$ $\Rightarrow \{U_1, \ldots, U_k\}_{U_0} \in \vartheta$	sse $\bigwedge_{i=0}^{k} \rho \models E_i : \vartheta_i \wedge \forall U_0, \ldots,$
(AAEnc)	$\rho \models \{\lvert E_1, \ldots, E_k \rvert\}_{E_0} : \vartheta$ $\Rightarrow \{\lvert U_1, \ldots, U_k \rvert\}_{U_0} \in \vartheta$	sse $\bigwedge_{i=0}^{k} \rho \models E_i : \vartheta_i \wedge \forall U_0, \ldots,$
(AOut)	$\rho, \kappa \models \langle E_1, \ldots, E_k \rangle.P$ $\Rightarrow \langle U_1, \ldots, U_k \rangle \in \kappa \wedge \rho, \kappa \models P$	sse $\bigwedge_{i=1}^{k} \rho \models E_i : \vartheta_i \wedge \forall U_1, \ldots,$
(AInp)	$\rho, \kappa \models (E_1, \ldots, E_k; x_{j+1}, \ldots, x_k).P$ sse $\bigwedge_{i=1}^{j} \rho \models E_i : \vartheta_i \wedge \forall \langle U_1, \ldots, U_k \rangle \in \kappa : \bigwedge_{i=1}^{j} U_i \in \vartheta_i$ $\Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho, \kappa \models P)$	
(ASDec)	$\rho, \kappa \models decryptEas\{E_1, \ldots, E_k; x_{j+1}, \ldots, x_k\}_{E_0}$ in P sse $\rho \models E : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models E_i : \vartheta_i \wedge \forall \{U_1, \ldots, U_k\}_{U_0} \in \vartheta$ $\wedge \bigwedge_{i=0}^{j} U_i \in \vartheta_i \Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho, \kappa \models P)$	
(AADec)	$\rho, \kappa \models decryptEas\{\lvert E_1, \ldots, E_k; x_{j+1}, \ldots, x_k \rvert\}_{E_0}$ in P sse $\rho \models E : \vartheta \wedge \bigwedge_{i=0}^{j} \rho E_i : \vartheta_i \wedge \forall \{\lvert U_1, \ldots, U_k \rvert\}_{U_0} \in \vartheta$ $\forall U_0^{'} \in \vartheta_0 : \forall (m^+, m^-) : (U_0, U_0^{'}) = (\lfloor m^- \rfloor, \lfloor m^+ \rfloor)$ $\wedge \bigwedge_{i=1}^{j} U_i \in \vartheta_i \Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho, \kappa \models P)$	
(AASig)	$\rho, \kappa \models decryptEas\{\lvert E_1, \ldots, E_k; x_{j+1}, \ldots, x_k \rvert\}_{E_0}$ in P sse $\rho \models E : \vartheta \wedge \bigwedge_{i=0}^{j} \rho E_i : \vartheta_i \wedge \forall \{\lvert U_1, \ldots, U_k \rvert\}_{U_0} \in \vartheta$ $\forall U_0^{'} \in \vartheta_0 : \forall (m^+, m^-) : (U_0, U_0^{'}) = (\lfloor m^+ \rfloor, \lfloor m^- \rfloor)$ $\wedge \bigwedge_{i=1}^{j} U_i \in \vartheta_i \Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho, \kappa \models P)$	
(ANew)	$\rho, \kappa \models (\nu n)P$	sse $\rho, \kappa \models P$
(AANew)	$\rho, \kappa \models (v \pm m)P$	sse $\rho, \kappa \models P$
(APar)	$\rho, \kappa \models P_1 \lvert P_2$	sse $\rho, \kappa \models P_1 \wedge \rho, \kappa \models P_2$
(ARep)	$\rho, \kappa \models !P$	sse $\rho, \kappa \models P$
(ANil)	$\rho, \kappa \models 0$	sse true

5

Caso studio

5.1 Protocollo FOO92

In questo capitolo analizziamo uno dei primi protocolli di voto elettronico e contestualmente descriviamo la procedura di voto applicando la sintassi Lysa. Nel 1992, Fujioka, Okamoto e Ohta ([6]) proposero una realizzazione concreta dello schema di voto che utilizza la firma cieca, la firma digitale e il canale di comunicazione anonimo presentate nel precedente capitolo. Le entità coinvolte nel protocollo sono gli elettori, l'amministratore e il tallier.
L'amministratore ha il compito di verificare che solamente gli elettori che ne hanno diritto possano votare e lo possano fare una volta sola (**eleggibilità**). Il tallier colleziona e rende pubblici i voti.
Lo descriviamo con i seguenti passi:

1. **Preparazione** : L'elettore compila la scheda elettorale elettronica, crea il messaggio usando la tecnica di firma cieca per ottenere la firma dell'amministratore, ed infine lo invia a quest'ultimo per l'autenticazione.

2. **Validazione** : l'amministratore firma il messaggio nel quale è nascosta la scheda elettorale dell'elettore, la firma e la rispedisce all'elettore.

3. **Votazione** : L'elettore riceve la scheda firmata dall'amministratore e la invia all'entità predisposta al conteggio attraverso un canale anonimo.

4. **Raccolta** : Il tallier pubblica una lista delle schede ricevute.

5. **Apertura** : L'elettore spedisce in maniera anonima la chiave che servirà a decriptare la scheda in cui è espresso il voto.

6. **Scrutinio** : Il tallier conta i voti e annuncia il risultato.

5.1.1 Assunzioni

Il protocollo descritto richiede alcuni assunzioni necessarie affinchè possa svolgere correttamente il proprio lavoro:

Tabella 5.1: Protocollo FOO92

1. PREPARAZIONE	:	V $\rightarrow$ A : ID, $\sigma_v(\chi_r(\xi_k(\mathrm{v})))$
2. AMMINISTRAZIONE	:	A $\rightarrow$ V : $\sigma_A(\chi_r(\xi_k(\mathrm{v})))$
3. VOTAZIONE	:	V $\rightarrow$ T : $\sigma_A(\xi_k(\mathrm{v}))$
4. PUBBLICAZIONE	:	T $\rightarrow$: l,$\sigma_A(\xi_k(\mathrm{v}))$
5. APERTURA	:	V $\rightarrow$ T : l,r

- I voti "bit committed" sono univoci;
- L'amministratore può firmare una ed una sola volta un voto per ciascun elettore;
- Il tallier è un'entità fidata. Se riceve un voto allora lo conta correttamente nel conteggio finale;
- Il tallier prima riceve tutti i voti e solo allora comincia la pubblicazione;
- Il voto è "accettato" se il numero di voti conteggiati dal tallier è uguale al numero di voti firmati dall'amministratore;
- Il voto è accettato se il tallier durante la fase di scrutinio riceve tutte le chiavi per i voti pubblicati.

5.2 Protocollo in Lysa

Il protocollo FOO92 può essere descritto utilizzando la seguente notazione Lysa. Utilizziamo la notazione '(V)' per rappresentare un flusso informativo attraverso un canale anonimo (1.5.5).

V $\longrightarrow$ A	:	V,$\{\|[\{\mathrm{v}\}_r]_b\|\}_{k_v^-}$
A $\longrightarrow$ V	:	$\{\|[\{\mathrm{v}\}_r]_b\|\}_{k_A^-}$
(V) $\longrightarrow$ T	:	$\{\|\{\mathrm{v}\}_r\|\}_{k_A^-}$
T $\longrightarrow$	:	l, $\{\|\{\mathrm{v}\}_r\|\}_{k_A^-}$
(V) $\longrightarrow$	:	l,r

5.3 Estendiamo la descrizione del protocollo

Come spiegato precedentemente maggiori informazioni forniamo, minori sono i rischi di implementare un protocollo non sicuro.
Pertanto seguendo la logica già spiegata possiamo descrivere il protocollo in modo

esteso distinguendo tra input, output e controlli obbligatori che ciascuna entità in gioco deve effettuare:

1	V $\longrightarrow$	:V,A,V,$\{\|[\{\mathrm{v}\}_r]_b\|\}_{k_v^-}$	
$1'$	$\longrightarrow$ A	: y_v,y_a,y_v',y_1	(controllo $\mathrm{y}_v = \mathrm{y}_v'$,$\mathrm{y}_1$, y_a=A)
$1''$	A	:decrypt y_1 as $\{\| y_2 \|\}_{k_{yv}^+}$	(controlla la firma di V)
2	A $\longrightarrow$	:A,y_v,$\{\| y_2 \|\}_{k_A^-}$	
$2'$	$\longrightarrow$ V	:x_A,x_V,x_1	(controlla x_a=A,x_V=V)
$2''$	V	:decrypt x_1 as $\{\| x_2 \|\}_{k_A^+}$	(controlla x_2=$[\{\mathrm{v}\}_r]_b$)
$2'''$	V	:unblind x_1 as $[\mathrm{x}_3]_b$	
3	(V)$\longrightarrow$	:D,T,x_3	
$3'$	$\longrightarrow$ T	:z_D,z_T,z_1	
$3''$	T	:decrypt z_1 as $\{\| z_2 \|\}_{k_A^+}$	
4	T$\longrightarrow$	:T,D,z_1,l	
$4'$	$\longrightarrow$V	:x_T,x_D,x_4,x_5	(controlla x_4=x_3,x_T=T,x_d=x_D)
5	(V)$\longrightarrow$	:D,T,x_5,r	
$5'$	$\longrightarrow$ T	:z_D',z_T',z_3,z_4	(controllo z_3 = l,z_T'=T)
$5''$	T	:decrypt z_2 as $\{\mathrm{z}\}_{z4}$	

Come è possibile osservare si evidenzia facilmente come la narrazione composta da cinque messaggi sia poi estesa esplicitando per ciascuna entità quali azioni deve compiere per una esecuzione corretta del protocollo.
Per ciascuna entità generalmente il primo messaggio identifica il messaggio inviato sulla rete, il secondo come comportarsi nel momento in cui si riceve un messaggio e il terzo l'azione che deve compiere per seguire correttamente il protocollo.

5.4 Il protocollo diviso in processi

La fase successiva è quella di distinguere le attività di ciascuna entità. Seguendo come già descritto la tecnica illustrata precedentemente, è possibile rappresentare il protocollo come segue:

$(v \pm K_v)(v \pm K_A)$
((vv)(vr)(vb)

Elettore:

1 $\langle V,A,V\{|[\{v\}_r^{v1}[\text{dest } L_{v1}]]_b^{v2}[\text{dest } L_{v2}]|\}_{K_V^-}^{v3}[\text{dest } L_{v1}]\rangle$.
$2'$ $(A,V;x_1)$.
$2''$ decrypt x_1 as $\{;x_2\}_{K_A^+}^{v4}[\text{orig } L_{v4}\ ]$ in
$2'''$ unblind x_2 as $[;x_3]_b^{v5}[\text{orig } L_{v5}\ ]$ in
3 $\langle D,T,\{[x_3]_{K_A^-}^{v6}[\text{dest } L_{v6}]\rangle$.
$4'$ $(T,D,\{[x_3]_{K_A^-}^{v7}[\text{dest } L_{v7}];x_4)$.
5 $\langle D,T,x_4,r\rangle.0$

Amministratore:

$1'$ $(V,A,V;\ y_1)$
$1''$ decrypt y_1 as $\{;y_2\}_{K_V^+}^{a1}[\text{orig } L_{a1}\ ]$ in
2 $\langle A,V,\{[y_2]\}_{K_A^-}^{a2}[\text{dest } L_{a2}],l\rangle.0$

Tallier:

| (vl)
$3'$ $(D,T;z_1)$.
$3''$ decrypt z_1 as $\{;z_2\}_{K_A^+}^{c1}[\text{orig } L_{c1}\ ]$ in
4 $\langle T,D,\{[z_2]\}_{K_A^-}^{c2}[\text{dest } L_{c2}],l\rangle$.
$5'$ $(D,T,l;z_3)$.
$5''$ decrypt z_2 as $\{;z_4\}_{z3}^{c3}[\text{orig } L_{c3}\ ]$ in 0
| $\langle K_V^+, K_A^+\rangle.0$ (chiavi pubbliche in chiaro)

5.4.1 Analisi

Tra le principali proprietà di sicurezza che un protocollo di voto deve garantire prendiamo innanzitutto in considerazione la **privacy**. Come già più volte espresso, garantire la privacy significa che nessuno può ottenere un collegamento diretto tra voto ed elettore. In letteratura si evidenzia come non sia possibile dimostrare tale proprietà utilizzando il tool Lysa. In particolare nel protocollo proposto un potenziale attaccante è in grado di conoscere sia il voto sia l'identità dell'elettore. Ma solo qualora l'attaccante possa collegare le due informazioni la proprietà è violata. Tuttavia è stata dimostrata la privacy del protocollo utilizzando le teoria dell'equivalenze: se il votante V_1 vota v_1 e il votante V_2 vota v_2, allora questo è equivalente alla situazione in cui V_1 vota v_2 e il votante V_2 vota v_1. Tutto ciò dimostrato aggiungendo una ulteriore assunzione: **tutti i votanti devono aver terminato la fase di amministrazione prima di procedere con la fase di voto** ([15]).

E' inoltre interessante porre l'attenzione sulla proprietà di verificabilità. Ricordiamo che un sistema è verificabile qualora tutti i votanti possano controllare in maniera del tutto indipendente che il proprio voto sia stato conteggiato correttamente. Per quanto concerne il protocollo FOO92, significa verificare l'autenticazione della lista pubblicata dal tallier. Cioè si richiede che chi pubblica la lista sia necessariamente il tallier. Così come implementato il protocollo l'analisi del protocollo evidenzia un potenziale attacco: la lista può essere pubblicata dall'attaccante. Questa falla può essere corretta facendo in modo che, tramite crittografia, anche il tallier firmi la lista prima di pubblicarla.

6

Analisi di alcune implementazioni recenti

Come più volte espresso, l'interesse per i protocolli di voto elettronico è cresciuto negli ultimi anni e si continua a cercare la soluzione migliore per sostituire le elezioni tradizionali. A scorrere vediamo due recenti lavori significativi per carpire quali soluzioni son state proposte recentemente, quali assunzioni è necessario presentare, quali proprietà è necessario rispettare e quali problemi è necessario affrontare per sviluppare protocolli di voto elettronico sicuri ed efficienti. Si può così osservare come **informalmente** siano state validate le proprietà di sicurezza di queste implementazioni.

6.1 Un protocollo di voto elettronico per generiche elezioni

Come già espresso precedentemente, l'idea di sviluppare un protocollo di voto elettronico è indotto dal recente utilizzo massivo della rete internet. Si cerca così di sfruttare questo canale ed introdurre una ulteriore tecnica per la votazione elettronica ([12]).
Immancabile la motivazione di ridurre i costi, sia in termini di denaro (risorse umane e materiali) sia in termini di tempo per portare a termine preparazione e scrutinii.
Come per ogni protocollo è necessario tener presente le difficoltà tecniche che si presentano quando si implementano protocolli su rete aperta.
A seguire si definiscono le proprietà che il protocollo deve rispettare per garantire sicurezza e si descrivono le fasi dell'implementazione. Successivamente si pone l'attenzione sulla validazione informale delle proprietà di voto elettronico.

Implementare il protocollo significa innanzitutto risolvere i seguenti problemi:

1. **Costi : economici / risorse umane;**
2. **Locazione / Mobilità.**

Riportiamo a seguire le definizioni, definite dall'autore del protocollo, delle proprietà di sicurezza per capire gli obiettivi del protocollo stesso. Successivamente verifichiamo se l'implementazione descritta soddisfa tali requisiti di sicurezza.

1. **Completezza:** E' una proprietà importante in un protocollo di voto elettronico. In un sistema di voto tradizionale si verifica l'identità del votante nel momento in cui l'elettore si presenta fisicamente al seggio. In un sistema di voto elettronico è possibile votare da diverse locazioni e ciò può generare molteplici errori in fase di identificazione. In particolare elettori che hanno diritto di voto potrebbero non poter votare senza valide ragioni.
   Per definizione quindi: **ad un elettore legittimo non viene mai negata la possibilità di votare da parte dell'amministratore.**

2. **Non coercibilità**: E' una proprietà interessante che si desidera implementare in protocolli reali di voto elettronico. Per definizione :

   (a) Un votante non può dimostrare ad un terzo come ha votato;

   (b) L'elettore vota secondo la propria coscienza.

3. **Non frode**: è una importante proprietà per garantire la robustezza del protocollo di voto elettronico. In particolare l'elettore deve poter accusare l'autorità di frode senza però rivelare il voto espresso dagli altri elettori.

4. **Robustezza**: anche questa è una importante proprietà per uno schema di voto elettronico. In particolare nessuno deve poter alterare, distruggere, modificare il corretto sviluppo dell'elezione. I processi devono poter essere indipendenti in modo tale che un potenziale intruso non possa interrompere l'intera elezione.

5. **Unicità**: nessun votante può esprimere il proprio voto più di una volta.
   Per definizione: ogni elettore può votare esattamente una volta.

6. **Verificabilità**: è un'essenziale proprietà di un protocollo di voto elettronico. In un sistema di voto tradizionale lo scrutinio permette di verificare il corretto conteggio dei voti. In una rete aperta voti non corretti non possono essere rivelati senza lo scrutinio diretto degli elettori. Pertanto per definizione: Gli elettori possono (ne hanno pieno diritto) verificare che i propri voti siano stati correttamente conteggiati.

7. **Equità**: nessuno può ottenere delle informazioni sul risultato della votazione prima della fase di pubblicazione dei voti.

8. **Anonimato**: nessuno deve poter trovare il legame tra votante e voto espresso.

9. **Semplicità / Convenienza** : ciascun elettore deve poter votare con la massima semplicità, rapidità e con le minime conoscenze utili.

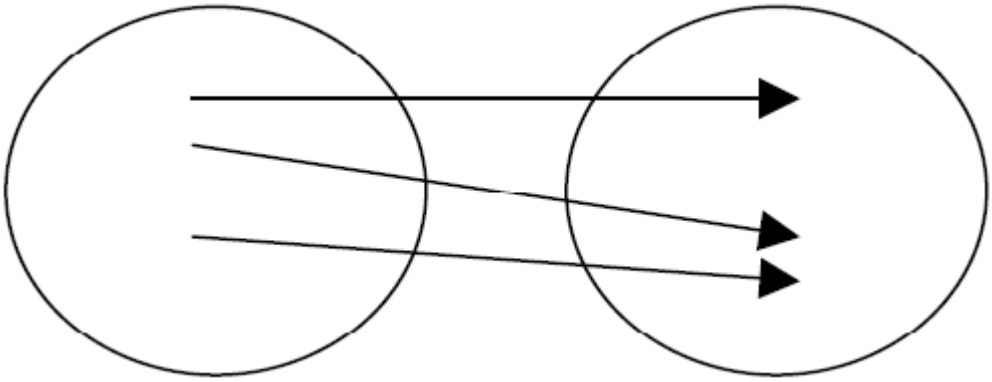

Figura 6.1: Funzione one-way

10. **Efficienza**: tutte le computazioni non devono essere onerose. Ciascun elettore non deve attendere molto tempo per esprimere la propria preferenza.

11. **Mobilità**: Non ci devono essere restrizioni per quanto concerne il luogo in cui l'elettore può esprimere il proprio voto.

6.1.1 Implementazione del protocollo

Juang e Lei hanno presentato uno schema unico di firma cieca che utilizza una funzione one-way (fig.6.1) per ottenere completezza. Una funzione di permutazione one-way è iniettiva e suriettiva. I dati possono mantenere la loro unicità dopo la computazione con una funzione di permutazione one-way. Ad esempio: se consideriamo un identificativo univoco X e applichiamo una funzione di permutazione one-way f allora anche f(X) sarà univoco.

Lo schema proposto

Lo schema presenta 3 entità:

1. Un elettore;

2. Una entità che ha i permessi per firmare i messaggi (**signer**);

3. Una entità che pubblica i risultati.

Si assumono inoltre due dispositivi per soddisfare le proprietà precedentemente espresse:

1. Una smart card

2. Decriptatore

E si assume che:

1. Gli elettori devono poter comunicare con tutte le entità e seguire correttamente il protocollo;

2. Esiste un sistema sicuro di email elettronica;

3. Il sistema crittografico RSA è sicuro se la fattorizzazione è intrattabile;

4. Ogni votante possiede una smartcard e lettore che non è possibile manomettere.

Notazione

d: la chiave privata dell'entità X
e,n: chiave pubblica dell'entità X
d_i: la chiave privata dell'elettore i-esimo
e_i, n_i: la chiave pubblica dell'elettore i-esimo
d_s: la chiave privata dell'entità che firma i messaggi
e_s, n_s: la chiave pubblica dell'entità che firma i messaggi
d_p: la chiave privata dell'entità che pubblica i messaggi
e_p, n_p: la chiave pubblica dell'entità che pubblica i voti
ID: identificativo univoco memorizzato nella smart card
f : una funzione one-way
RD: un numero random generato dal centro di voto
V_i: la scelta dell'elettore i-esimo
R_i: stringa random generata dall'elettore i-esimo
r_i: un numero random generato dall'elettore i-esimo che soddisfa gcd(r_i,n)=1
t_i: un numero random generato dall'elettore i-esimo che soddisfa gcd(t_i,n)=1

Prima fase, la preparazione

Nella prima fase ciascuna entità in gioco genera autonomamente la propria chiave pubblica e chiave privata. Quando una entità desidera conoscere la chiave privata di un'altra può farlo tramite un canale autorizzato da una **autorità di certificazione** (CA). Ciascun votante è in possesso di una smart card con un identificativo autorizzato dall'autorità di certificazione. Il centro di voto genera una funzione f di permutazione one-way e un numero random RD per verificare la validità dei voti ed è equipaggiato con un decriptatore non tracciabile. Il centro di voto annuncia la funzione ed il numero random. Ciascuna entità tramite l'autorità di certificazione può ottenere queste informazioni.

La votazione

Ciascun elettore genera la propria preferenza (voto), una stringa R_i e due numeri random r_i e t_i. Salva quindi tali informazioni (R_i, r_i e t_i) nella propria smartcard. A questo punto la smartcard calcola VT_i eseguendo la sequente equazione:

$VT_i = (r_i^{es}(t_i^{ep}(H_i \| V_i) \bmod n_p \| RD)) \bmod n_s \| REG_i$
$REG_i = RD^{di} \bmod n_i$
$H_i = f(ID_i, R_i)$
$M_i = (t_i^{ep}(H_i \| V_i) \bmod n_p \| RD)$
$Y_i = (r_i^{es}(t_i^{ep}(H_i \| V_i) \bmod n_p \| RD)) \bmod n_s$

Successivamente il votante i-esimo invia VT_i al **signer**.
Una volta ricevuto VT_i il signer verifica che REG_i sia corretto:
$REG_i^{ei} \bmod n_i = (RD^{di} \bmod n_i)^{ei} \bmod n_i = RD$

Se REG_i^{ei} è diverso da RD allora VT_i viene rifiutato (**autenticazione**).
Se l'elettore è un elettore legittimo, il signer calcola la seguente equazione:
$Z_i = Y_i^{ds} \bmod n_s$
e spedisce l'esito della computazione al votante i-esimo.
A questo punto la smartcard utilizza tale informazione per calcolare X_i come segue:

$X_i = Z_i r_i^{-1} \bmod n_s$
$= ((r_i^{es} M_i \bmod n_s)^{ds} \bmod n_s) r_i^{-1} \bmod n_s$
$= M_i^{ds} \bmod n_s$

In questa fase la smart card invia il voto (X_i,M_i,t_i) al signer e tiene copia della ricevuta. Il signer verifica X_i nel momento in cui riceve il voto e successivamente invia (X_i,M_i,t_i) ad un decriptatore:

$X_i^{es} \bmod n_s = (M_i^{ds} \bmod n_s)^{es} \bmod n_s = M_i$.

Alla fine della fase di voto il decriptatore inizia la fase di pubblicazione dei voti. L'entità adibita alla pubblicazione calcola la seguente equazione:

$M_i = t_i^{ep}(H_i \| V_i) \bmod n_p \| RD$. quindi:

$((M_i'^{dp} \bmod n_p) t_i^{-1} \bmod n_p$
$=((t_{ep}^i(H_i \| V_i) \bmod n_p)^{dp} \bmod n_p) t_i^{-1} \bmod n_p$
$=(H_i \| V_i)^{dp} \bmod n_p = M_i''$
Infine viene pubblicato il risultato:

$(M_i'')^{ep} \bmod n_p = ((H_i \| V_i)^{dp} \bmod n_p)^{ep} \bmod n_p = (H_i \| V_i)$.

Il votante i-esimo può utilizzare la ricevuta (X_i,M_i,t_i) e la chiave privata d_p resa visibile nel periodo di pubblicazione dei voti, per verificare se il proprio voto sia stato calcolato/conteggiato correttamente. Qualora così non fosse, l'elettore ha la possibilità di inviare una richiesta affinchè il proprio voto sia riconteggiato correttamente.

6.1.2 Analisi di sicurezza del protocollo

A seguire analizziamo quali considerazioni son state effettuate per verificare le proprietà di sicurezza del protocollo.

1. **Completezza**

   Se ciascun votante genera un numero identificativo in luoghi diversi, allora può capitare che diversi elettori possano generare il medesimo identificativo. Come conseguenza logica, ad elettori legittimi sarà negata la possibilità di votare. Nello schema presentato ciò non può accadere poiché **ID è un numero identificativo univoco**, **f è una funzione di permutazione one way** ed **R è un numero rando**m selezionato dall'elettore. Quindi per definizione l'output **f(ID,R) è univoco**.
   In sistemi in cui si utilizza una entità centrale globale per permettere ai votanti di ottenere un numero identificativo univoco non dovrebbe mai accadere che legittimi votanti siano rifiutati dall'amministratore. In questo modo si ottiene sicuramente completezza ma non viene soddisfatta la proprietà di robustezza, inoltre sono alti i costi di computazione dei dati. Un tale approccio non è compatibile con elezioni in larga scala. Nello schema proposto ID è un numero identificativo univoco memorizzato in una smart card. Non è necessaria computazione in rete. Non sono necessari costi computazionali onerosi.

2. **Non coercibilità**

   Lo schema proposto utilizza una smart card per nascondere la ricevuta di voto temporaneamente fino a quando viene pubblicato il risultato finale dell'elezione. A questo punto gli elettori possono recuperare la chiave privata dell'entità che pubblica e decriptare la ricevuta di voto. Lo schema utilizza la ricevuta del voto per ottenere verificabilità.

3. **Unicità**

   Ogni votante può votare una ed una sola volta. Una proprietà fondamentale sia per sistemi tradizionali sia per sistemi di voto elettronico. Lo schema utilizza REG_i per soddisfare tale proprietà. Se REG_i può essere decriptato per ottenere RD e contestualmente REG_i non era mai stato autorizzato, implica che nessuno può votare più di una volta.

4. **Verificabilità**

   Un sistema di voto elettronico ha come contesto internet e i dati che non son stati criptati possono essere modificati e duplicati. Senza lo scrutinio da parte degli elettori, potrebbe non esser possibile verificare l'attendibilità dei risultati. **Lo schema proposto utilizza le ricevute dei voti per ottenere tale proprietà.** I votanti utilizzano la ricevuta di voto, la decriptano utilizzando la chiave privata del pubblicatore per ottenere (H_i,V_i).

5. **Fairness**

   Lo protocollo proposto utilizza RSA per criptare i messaggi. Assumendo che RSA sia sicuro, possiamo ottenere fairness.

6. **Anonimato**

   Lo schema proposto utilizza email non tracciabili e decriptatori per interrompere il legame tra voti e votanti. Un potenziale attaccante non può tracciare questo legame, pertanto l'identità dei votanti restano segrete.

7. **Convenienza**

   Un sistema di voto elettronico conveniente dovrebbe permettere di votare eliminando tutte le restrizioni fisiche e senza costringere l'elettore a dover imparare tecniche troppo complesse. Lo schema proposto ha due assunzioni fisiche: una smart card ed un lettore (ormai di pubblico dominio). I votanti non devono imparare particolari tecniche per procedere con le votazioni.

8. **Efficienza**

   Tutte le computazioni son eseguite in una quantità ragionevole di tempo. Ai votanti non è richiesto di aspettare che altri elettori terminino le operazioni di voto prima di poter esprimere la propria preferenza.

9. **Mobilità**

   Utilizzando una smart card i votanti possono votare attraverso internet e non hanno restrizioni di locazione.

Conclusioni

Questo lavoro propone un completo e sicuro sistema di voto applicabile in una situazione reale. E' un sistema efficiente e soddisfa le proprietà essenziali per un protocollo di voto elettronico.

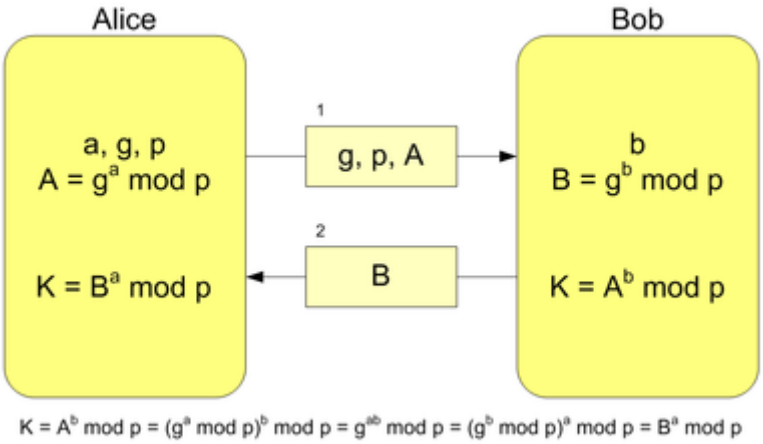

Figura 6.2: Protocollo Diffie Hellman

6.2 Un protocollo di voto basato su un protocollo di scambio di chiave

A seguire viene descritta una tecnica di voto ([11]) che utilizza **schemi di firma cieca ed il protocollo Diffie Hellman**([16]) per lo scambio di chiavi.

Nel seguente lavoro si adotta il protocollo per scambio di chiavi Diffie Hellman per garantire efficienza, un **proxy server** e firma cieca per nascondere il legame tra voto ed elettore. Il proxy server pubblico può rimpiazzare l'indirizzo dal quale proviene il voto in modo tale da garantire anonimato.

6.2.1 Implementazione del protocollo

Vediamo ora nel dettaglio le caratteristiche implementative del protocollo proposto.

Il protocollo coinvolge sei entità:

- Un centro di registrazione (RC);
- Un centro di certificazione (CC);
- Un centro di monitoraggio (MC);
- Un conteggiatore di voti (VC);
- Un proxy server;
- L'elettore.

Definiamo ora la notazione utile a formalizzare il protocollo:
(pk,sk): la chiave pubblica e privata per CC, generata per mezzo del sistema RSA;

N = il prodotto P_1 x P_2 in cui P_1 e P_2 son due grandi numeri primi in cui P_1 mod 4 = 3 e P_2 mod 4 = 3
p = un grande numero primo
g = un elemento primitivo in GF(p)
V_i = elettore i-esimo
x_r = la chiave privata di RC
y_r = la chiave pubblica di RC in cui $y_r = g^{xr}$ mod p
x_m = la chiave privata di MC
y_m = la chiave pubblica di MC in cui $y_m = g^{xm}$ mod p
x_v = la chiave privata di VC
y_v = la chiave pubblica di VC in cui $y_v = g^{xv}$ mod p
x_i = la chiave privata dell'elettore i-esimo
y_i = la chiave pubblica dell'elettore i-esimo in cui $y_i = g^{xi}$ mod p
h() = una funzione pubblica one-way
m_i = il voto firmato di Vi
t_i=un timestamp generato da Vi
$E_k()$ = la funzione per criptare con chiave k
D_k () = la funzione per decriptare con chiave k
TR/TR' = il tabellone dei risultati

A seguire analizziamo le fasi del protocollo:

Fase iniziale

Nella prima fase MC genera un nonce N_1 e calcola :

$k = y_v^{xm}$ mod p = g^{xvxm} mod p e quindi ottiene $E_k(N_1)$.
Quindi MC invia $E_k(N_1)$ a VC.
VC, una volta ricevuto il messaggio da MC, otterrà k e calcolerà $E_k(N_1+1)$ da inviare a MC.
In questo modo MC e VC condividono la chiave di sessione k. Questa chiave viene utilizzata per calcolare $k^{'}$ che viene utilizzata per ottenere la chiave di sessione k^*

Fase di voto

Nella fase di voto V_i sceglie due numeri random RD_1 e RD_2 e calcola:

$M_i = RD_1 \oplus RD_2 \oplus m_i$.

Quindi V_i calcola:

$k = y_i^{xr}$ mod p e
$E_k(M_i$, informazione elettore,$N_3)$ e lo invia ad RC (N_3 è un nonce).

Nel momento in cui RC riceve il messaggio inviato dall'elettore i-esimo ottiene la chiave e calcola: D_k (E_k(M_i, informazione elettore,N_3)).

Contestualmente verifica l'identità di V_i. Se V_i è un elettore legittimo allora calcola:

$B_i = E_{k*}(M_i \parallel t_i)$ e genera un numero identificativo univoco SN_V_i per V_i, in cui t_i è un timestamp. RC genera un numero seriale per ciascun elettore una sola volta. Quindi calcola:

E_k(B_i,SN_V_i,N_3) e lo spedisce a V_i.

Una volta ricevuto il messaggio da RC, V_i calcola:

$D_k(E_k(B_i,SN\_V_i,N_3))$

Quindi V_i calcola:

$C_i = (h(B_i)RM)^{pk} \bmod N$

in cui RM è un numero random e RM $\in Z_N$. V_i invia C_i a CC.
CC nel momento in cui riceve il messaggio inviato da V_i calcola la seguente equazione:

$BG_i = (C_i)^{sk} \bmod N = h(B_i)^{sk}RM \bmod N$ e spedisce il risultato a V_i

V_i a sua volta calcola:

$SG_i = BG_iRM^{-1} = h(B_i)^{sk} \bmod N$ per recuperare la firma di $h(B_i)$.

Successivamente V_i calcola $h(SN\_V_i)$ e spedisce il messaggio $\{h(SN\_V_i), SG_i,B_i,RD_1\}$ e $\{h(SN\_V_i), SG_i,B_i,RD_2\}$ rispettivamente a MC e VC attraverso un proxy server che nasconde l'indirizzo originale del votante i-esimo.
Quando MC e VC ricevono il messaggio dell'elettore eseguono la seguente procedura per confermare la validità di V_i.

$h(B_i) = ?(SG_i)^{pk} \bmod N$.

Se la verifica restituisce un risultato positivo MC e VC salvano $\{h(SN\_V_i), SG_i,B_i,RD_1\}$ e $\{h(SN\_V_i), SG_i,B_i,RD_2\}$ nei propri archivi controllando che SN_V_i sia presente una sola volta. Qualora così non fosse la duplicazione viene cancellata.

Al termine della fase di voto VC calcola $E_k(RD_2)$ e lo spedisce a MC. Contestualmente MC calcola $E_k(RD_1)$ e lo spedisce a VC. In questo modo MC ottiene RD_2

mentre VC ricava RD_1. MC e VC quindi conoscono RD_1 e RD_2 e possono calcolare:

$m_i = RD_1 \oplus RD_2 \oplus M_i$. MC e VC devono necessariamente ottenere lo stesso risultato affinchè il voto possa essere confermato. Qualora così non fosse il risultato (voto) deve essere annullato.

6.2.2 Analisi di sicurezza del protocollo

1. **Anonimato**

   Il protocollo proposto utilizza lo **schema di firma cieca** per interrompere il legame esistente tra elettore e corrispondente voto espresso. Inoltre viene utilizzato un **proxy server** per rimpiazzare l'indirizzo di rete del voto in modo tale che sia MC sia VC non possano trovare il legame tra voto ed indirizzo di rete del votante. In questo modo il requisito è rispettato.

2. **Fairness**

   Nessuno può venire al corrente di informazioni sul risultato finale prima che esso sia pubblicato. MC e VC non possono decriptare il risultato senza conoscere preventivamente RD_1 e RD_2. MC e VC si scambiano la conoscenza di RD_1 e RD_2 in modo tale da calcolare $m_i = RD_1 \oplus RD_2 \oplus M_i$ ed ottenere m_i. Il requisito è pertanto rispettato.

3. **Convenienza**

   L'unico requisito richiesto all'elettore è l'accesso alla rete. Oramai una pratica comune. Non è richiesto uno studio di particolari tecniche per le procedure di voto. **Lo schema proposto richiede un proxy server**. Il requisito è pertanto rispettato.

4. **Robustezza**

   Lo schema proposto si basa sul protocollo Diffie-Hellman. I messaggi trasmessi in rete dalle entità coinvolte sono criptati per mezzo di chiavi di sessione. In questo modo nessuno può alterare o conoscere m_i senza essere in possesso delle chiavi di sessione coinvolte. Il requisito è pertanto rispettato.

5. **Mobilità**

   Lo schema è proposto per essere implementato in rete. E' semplice per l'elettore esprimere il proprio voto attraverso internet. Se il votante ha accesso ad internet non ha particolari restrizioni al luogo in cui votare. Il requisito è pertanto rispettato.

6. **Unicità**

RC rilascia esclusivamente un numero seriale SN_V_i per volta, il votante V_i non può votare due volte. Se così fosse sia MC sia VC rilevano duplicati nei loro database e procedono alla cancellazione. Il requisito è pertanto rispettato.

7. **Completezza**

   Assumiamo che nessuno possa effettuare un attacco con successo sullo schema a firma cieca per falsificare $h(B_i)$. Quindi è permesso di votare solo ad elettori legittimi. Ciascun votante ha un unico numero seriale.

8. **Non coercibilità**

   V_i non può rivelare m_i ad altri finchè B_i è sotto la protezione della chiave condivisa k*.

9. **Correttezza**

   Per garantire la correttezza tutti i voti non possono esser rimossi, alterati o duplicati. **Sia MC che VC possono solo cooperare per rivelare m_i** calcolando $m_i = RD_1 \oplus RD_2 \oplus M_i$.

10. **Efficienza**

    Il votante perde poco tempo per concludere le procedure di voto. Questo anche perchè molte computazioni sono eseguite da MC e VC. In particolare ciascun elettore che abbia diritto a votare ha bisogno di eseguire solo due operazioni esponenziali nella fase di voto. Quindi è applicabile in un contesto reale.

Conclusioni

Nel corso degli ultimi anni l'interesse per i protocolli di votazione elettronica e' cresciuto anche in rapporto allo sviluppo delle tecnologie. Come tutti i protocolli di comunicazione in rete (sia essa internet o intranet) e' necessario definire con estrema precisione quali siano gli obiettivi e garantire che nessuna entita' esterna possa in qualche modo alterare la validita' del protocollo stesso. FOO92 e' stato uno dei primi protocolli di voto elettronico e solamente alcuni anni dopo son state evidenziate alcune problematiche che non garantiscono all'elettore una completa democrazia. Utilizzando tecniche di analisi statica e' possibile analizzare e scovare eventuali falle dei protocolli di comunicazione. Non sempre pero' e' semplice definire formalmente le proprieta' di sicurezza.
Come gia' evidenziato si riscontrano notevoli difficolta' nell'analizzare la proprieta' di anonimato utilizzando solo Lysa.
L'analisi statica e' una importante tecnica di verifica dei programmi poiche' aiuta ad implementare correttamente un protocollo non tralasciando gli obiettivi e le azioni principali di ciascuna entita' coinvolta. Abbiamo osservato, presentando alcuni protocolli recenti, come alcune proprieta' di particolare interesse vengono validate utilizzando dispositivi fisici quali smartcard (garantiscono autenticazione), proxy server (es. rompono il legame tra voto ed elettore), decriptatori.
Come gia' precedentemente espresso i risultati significativi potranno essere raggiunti solo dopo la formalizzazione in Lysa dei nuovi protocolli descritti per poter successivamente procedere all'analisi control flow relativamente alle proprieta' di sicurezza presentate nel primo capitolo.

Bibliografia

[1] A. Cerioni, F.Cenacchi
Aspetti di sicurezza nelle elezioni elettroniche, Corso di sicurezza, Universita' di Bologna, 2003

[2] Doungles R. Stinson
Cryptography, Theory and Practice, CRC Press, 1995

[3] Chun-Ta Li, Min-Shiang Hwang, Chi-Yu Liu
An electronic voting protocol with deniable authentication for mobile ad hoc networks, Computer Communications, pag. 2534-2540, 2008

[4] F. Nielson, H.R. Nielson, and C. Hankin
Principles of Program Analysis, Springer 2003

[5] Han Gao - Technical University of Denmark
Analysis of Security Protocols by Annotations, Kongens Lyngby, Ph.D Thesis, 2008

[6] Atsushi Fujioka, Tatsuaki Okamoto, Kazuo Ohta
A Pratical Secret Voting Scheme for Large Scale Elections. Lecture Notes in Computer Science: Advances in Cryptology AUSCRYPT, pag. 244-250, 1992

[7] C.R.Nielsen, H.R.Nielson
Static Analysis for Blinding,Nordic Journal of Computing, 17 Agosto, 2006

[8] C.Bodei, M. Buchholtz, P.Degano F.Nielson,H.R.Nielson
Static Validation of Security Protocols,Tecnical University of Denmark, 23 Marzo, 2004

[9] Mayla Brusò, Agostino Cortesi
Non-repudiation Analysis Using Lysa with annotations,Computer Languages, System & Structure, pag .352-277, 2010

[10] Nielsen,Andersen
Static Validation of a Voting Protocol, Electronic Notes in Theoritical Computer Science, Vol. 135, pag 115-134, 2005

[11] Chin-Chen Chang, Jung-San Lee
An anonymous voting mechanism based on the key exchange protocol, Computer & Security Vol. 25 pag 307-314, 2006

[12] Horng-Twu Liaw
A secure electronic voting protocol for general elections, Computer & Security Vol. 23 pag. 107-199, 2004

[13] P. van Oorschot, S. Vanstone
Handbook of Applied Cryptography, CRC Press,1996

[14] William Stallings
Cryptography and Network Security, Principles and Practice, Prentice Hall, 1999

[15] S. Delaune, S. Kremer, M. D. Ryan
Verifying Privacy-type Properties of Electronic Voting Protocols. Journal of Computer Security, 2009

[16] A. Nicolosi
Sicurezza e applicazioni crittografiche della funzione Diffie-Hellman, Tesi di laurea, A.A.2000-2001