E-Vox

UN SISTEMA DI VOTAZIONE ELETTRONICA

tesina scritta da
Fabrizio Bisi
Per l'esame di Crittografia

Introduzione

Questa tesina si propone di discutere un'applicazione della crittografia che va prendendo sempre più importanza, il voto elettronico. L'approccio scelto è quello di descrivere nei dettagli un protocollo specifico tra i tanti presenti in letteratura, allo scopo di cercare di approfondire tutte le problematiche e le tecniche crittografiche in esso utilizzate, evitando così il rischio di perdersi in un discorso sui massimi sistemi.

Il protocollo scelto è E-Vox, sviluppato dal DARPA nell'ambito della ricerca "Security for Distributed Computer Systems" e utilizzato nel 1998 per le elezioni studentesche al MIT.

E-Vox è basato a sua volta su un protocollo descritto nell'articolo "A Practical Secret Voting Scheme for Large Scale Elections" di Fujioka, Okamoto, Ohta a cui nella tesina si farà riferimento come protocollo FOO (dalle iniziali degli autori).

Nella prima parte della tesina farò una breve storia dei protocolli di voto elettronico con lo scopo di introdurre il protocollo FOO, che è l'argomento della seconda parte. Nella terza e ultima parte descriverò infine il sistema E-Vox vero e proprio.

Storia

Self-adjuticating protocols

I protocolli di voto elettronico più semplici sono quelli in cui gli elettori devono preoccuparsi della validità della votazione, senza intermediazione di nessuna autorità esterna.
Il protocollo più famoso di questo gruppo è quello di Merrit [2]. Nel caso di una votazione con solo 4 elettori (A,B,C,D) un voto assume la seguente forma:

$$E_A(R_A,E_B(R_B,E_C(R_C,E_D(R_D, (E_A(E_B(E_C(E_D(V,R))))))))$$

Chiaramente questi protocolli sono inadatti per votazioni che abbiano più di pochi elettori

Protocolli con un'autorità centrale (CVR)

Un passo nella giusta direzione è quello di creare un'autorità centrale di coordinamento, che si preoccupi di raccogliere i voti degli elettori e di eseguire lo spoglio finale. Chiamiamo tale autorità una Central Vote Repository (CVR).
Ci sono molti protocolli di questo tipo, alcuni relativamente semplici altri molto complessi. Tutti comunque si trovano a dover risolvere il problema di impedire alla CVR di violare la privacy degli elettori. Le varie soluzioni adottate indeboliscono il protocollo sotto altri punti di vista.

Per esempio, un possibile protocollo è il seguente (ogni elettore ha un sistema di chiavi pubbliche):

1. il CVR prepara una lista di elettori che intende votare e la pubblica
2. Ogni elettore riceve un numero identificativo ID usando un protocollo ANDOS (All-Or-Nothing-Disclosure-Of-Secrets)
3. Ogni elettore manda in maniera anonima al CVR ID e $E_k(ID,V)$ (V è il voto)
4. Il CVR pubblica tutti i voti criptati $E_k(ID,V)$
5. Dopo che il passo 4 è completo tutti gli elettori mandano in maniera anonima ID e la chiave privata
6. A questo punto tutti i voti sono decriptati e pubblicati in chiaro insieme all'ID, in maniera che ogni elettore possa controllare che il suo voto è stato conteggiato correttamente

I voti non possono essere associati agli elettori grazie all'uso del protocollo ANDOS. Comunque i protocolli ANDOS sono solitamente molto pesanti e tali da non essere adatti a più di qualche decina di utilizzatori. Inoltre questo protocollo non impedisce ad elettori non autorizzati di votare né ad elettori autorizzati di votare più volte.
Un altro problema importante, comune a tutti i protocolli di questa famiglia, è che il CVR costituisce un singolo punto di rottura. Se si riesce a compromettere quello si compromette tutto il sistema.

Protocolli con due organismi istituzionali

Il passo successivo è quello di utilizzare due autorità centrali, una Validation Authority (VA) che si preoccupa di certificare gli elettori che hanno diritto al voto e una Tabulation Facility (TF) che raccoglie materialmente i voti. La prima conosce gli aventi diritto al voto e chi tra loro ha votato ma non può risalire al loro voto, la seconda conosce i voti ma non può sapere quali elettori li hanno consegnati.
Uno protocollo di esempio potrebbe essere il seguente:

1. Ogni elettore (dopo aver provato la sua identità) chiede a VA un numero di autorizzazione. VA genera questi numeri di autorizzazione a caso e li distribuisce
2. Al termine di questa fase VA consegna la lista di tutti i numeri di autorizzazione assegnati a TF

3. Ogni elettore si sceglie un numero identificativo a caso e lo manda in modo anonimo a TF insieme al suo voto e al numero di autorizzazione assegnato da VA
4. TF controlla il numero di autorizzazione e, se è nella lista, pubblica il voto insieme al numero identificativo

Questo schema ha alcuni vantaggi su quello vista in precedenza. Nessuna delle due entità coinvolte può violare da sola la privacy degli elettori, mentre il numero di autorizzazione fornito da VA impedisce di votare a chi non ne ha diritto e di votare più volte a chi ne ha diritto.
Comunque questo sistema è lungi dall'essere perfetto. La Validation Authorithy può creare falsi elettori e votare per loro, mentre se le due autorità colludono possono ancora violare la privacy degli elettori.
Di seguito vedremo come FOO prende questo schema e lo migliora per garantire certe proprietà indispensabili.

Il protocollo FOO

Nell'introduzione dell'articolo [3] gli autori prendono rapidamente in esame alcuni protocolli di voto elettronico esistenti e mettono in evidenza che o non sono adatti per votazioni di larga scala o non soddisfano due requisiti fondamentali:

- la privacy, in quanto in caso di contestazione viene chiesto all'elettore di rivelare il proprio voto per poter dimostrare di essere nel giusto
- la fairness (imparzialità), ossia le autorità centrali sono in grado di conoscere i risultati parziali delle votazioni, avendo così la possibilità di influenzarne l'andamento

Il protocollo descritto si propone allora come il primo protocollo che rispetta tali requisiti e che allo stesso tempo sia adatto a votazioni di grande scala.

Elementi crittografici

Gli strumenti crittografici usati dal protocollo sono essenzialmente tre:

- firme digitali (digital signatures)
- firme cieche (blind signatures)
- (blind) commitment (consegna cieca)

Per implementare la tecnica di commitment si usano funzioni hash one-way. Nel seguito faremo un veloce ripasso di queste tecniche crittografiche, soffermandoci a specificare la tecnica esatta utilizzata da E-Vox.

Firme digitali (digital signatures)

Le firme digitali devono avere 3 requisiti:

- essere uniche
- non essere falsificabili (A non può creare la firma di B)
- essere verificabili (chiunque deve poter verificare che una firma sia autentica)

I sistemi a chiave asimmetrica possono essere usati come sistemi di firma digitale. La chiave privata può essere utilizzata come firma, in modo che chiunque possa verificare la legittimità della firma applicando la chiave pubblica al documento firmato.

E-Vox utilizza RSA per firmare.
Quindi il messaggio M può essere autenticato dal suo autore criptando con la chiave privata (d,n)

$$S = M^d \bmod n$$

Mentre chiunque può verificare che la firma è autentica usando la chiave pubblica (e,n) su S per riottenere il messaggio originale M

$$S^e \bmod n = (M^d)^e \bmod n = M^{de} \bmod n = M \bmod n = M$$

Firme cieche (blind signatures)

Le firme cieche sono necessarie in quelle situazioni in cui serve che un'autorità di cui si ha fiducia certifichi un messaggio (dichiarando di esserne venuta in possesso) senza poterne vedere il contenuto.
In molti protocolli di voto elettronico serve che un'autorità centrale di certificazione riceva il voto di un elettore e apponga la sua firma per convalidare il fatto che quel voto proviene da un avente diritto che non ha ancora esercitato il suo diritto di voto. L'autorità certificatrice (VA = Validation Authority) deve poter essere in grado di certificare il voto ma non deve poterlo leggerlo, per tutelare la privacy dell'elettore.

La tecnica adottata comunemente e usata anche da E-Vox è quella di usare un "blind factor", ossia moltiplicare il messaggio M per un fattore k scelto a caso che ne rende impossibile la lettura.

L'elettore prima "blinda" il messaggio moltiplicandolo per un valore k^e (dove "e" è la chiave pubblica della VA e k è un valore a caso tra 1 e n) e lo spedisce alla VA

$V \rightarrow VA: \quad B = Mk^e \bmod n$

Poi la VA lo firma e lo rispedisce al mittente (chiamiamo S' il messaggio blindato e firmato dalla VA)

$VA \rightarrow V: \quad S' = B^d \bmod n = (Mk^e)^d \bmod n = M^d k \bmod n$

A questo punto l'elettore può sblindare S' e quello che ottiene è il messaggio originale M firmato da VA (lo chiamiamo S)

$$S = (S' / k) \bmod n = M^d \bmod n$$

(Blind) commitment (consegna cieca)

Un commitment è un modo in cui A può mandare qualcosa a B senza che B possa vedere cos'è ma anche senza che A abbia più la possibilità di modificarlo.
Nei protocolli di voto elettronico con TF e VA separati, dopo che l'elettore ha inviato blindato il suo voto a VA e l'ha ricevuto firmato, il prossimo passo è quello di inviare su un canale anonimo il voto firmato da VA a TF perché venga conteggiato. Il problema è che TF non deve vedere il voto, che potrà vedere solo alla fine dello spoglio (in modo da evitare che possa vedere il risultato intermedio della votazione). Una volta inviato neppure l'elettore dovrà essere nella condizione di cambiare il voto.
V genera 2 stringhe di bit a caso R_1 e R_2
Quindi usa una funzione di hash
$(V) \rightarrow TF: \quad R_1, C = H(R_1, R_2, M)$
TF non può calcolare M da C per le proprietà delle funzioni one-way hash
V non può trovare (M', R') t.c.

$$C = H(R_1, R', M')$$

per le proprietà delle funzioni di hash. Quindi non potrà sostenere di aver votato M' invece di M.

Successivamente, quando la votazione è conclusa, i voti devono essere rivelati.Per far questo basta che l'elettore spedisca a TF, sempre anonimamente, l'intera stringa originale
$(V) \rightarrow TF: \quad R_1, R_2, M$
TF verifica che R_1 sia quello di prima, applica la funzione H a tutta la stringa e verifica che il risultato sia uguale a C.

Funzioni di hash one-way

Una funzione H

$$h = H(M)$$

è one-way hash quando gode delle seguenti proprietà:

- per qualsiasi input (M) in un certo range di dimensioni l'output (h) ha dimensione costante
- la funzione inversa H^{-1} è difficile da calcolare (dato H e h è difficile trovare M)
- dato un messaggio M, è difficile trovare un altro messaggio M' t.c. H(M) = H(M')

E-Vox utilizza come algoritmo di hashing HMAC-SHA.
SHA (Secure Hash Algorithm) è un algoritmo di hashing basato su MD4 che prende in input fino a 2^{64} bit e restituisce in output 160 bit.
HMAC è un algoritmo che prende una funzione di hash (nel nostro caso SHA) e ne rafforza la sicurezza.

$$h = SHA(k \oplus opad, SHA(M, k \oplus ipad)$$

Nella riga sopra k è una chiave generata casualmente che per motivi di sicurezza deve essere di almeno 160 bit e non più lunga di 64 byte, mentre ipad e opad sono due costanti definite da HMAC lunghe 64 byte (per fare gli XOR si aggiungono degli zeri a k per renderla lunga 64 byte).

Se invece di una chiave k usiamo due chiavi distinte R_1, R_2 abbiamo una one-way hash function che possiamo usare per il commitment.
Ecco la riga sopra senza dettagli implementativi e con le due chiavi distinte

$h = SHA(R_1, SHA(M, R_2))$

Descrizione del protocollo

In FOO il Tabulating Facility si chiama Counter, la Validation Authority si chiama Administrator. Le parti in causa quindi sono 3: l'elettore generico (V), l'administrator (A) e il counter (C).
Il protocollo si snoda in 6 passi:

1. l'elettore V prepara il voto e lo invia all'amministratore per farselo firmare
2. l'amministratore A verifica se è tutto a posto e in tal caso restituisce a V il voto firmato con la propria chiave privata (pubblicando a fine votazione una lista di tutti quelli che hanno richiesto la firma)
3. l'elettore V lo sblinda e ottiene il certificato elettorale che spedisce al counter
4. il counter C raccoglie tutti i voti e alla fine delle votazioni pubblica la lista di tutti i voti raccolti)
5. A questo punto gli elettori devono inviare al counter le chiavi con cui aprire il commitment
6. Infine il counter conta i voti, aggiunge i voti in chiaro alla lista dei voti e annuncia il risultato

Ecco i sei passi descritti in dettaglio

1. Preparazione

L'elettore prepara il voto e lo invia all'amministratore per farselo certificare.
I passi sono i seguenti:

- V prepara il plaintext
  M
- gli applica un commitment e un blind factor
  $\chi(M) = R_1, H(R_1, R_2, M)$
  $b(\chi(M))$
- e lo firma con la propria chiave privata
  $S_V(b(\chi(M)))$
- infine invia il proprio identificativo, il voto senza firma e il voto firmato all'amministratore
  $V \rightarrow A$: $V, S_V(b(\chi(M))), b(\chi(M))$

2. Amministrazione

L'amministratore riceve il voto dell'elettore, controlla se l'elettore ha diritto al voto e se non ha già votato, controlla l'autenticità del voto. Se tutti questi controlli vanno a buon fine restituisce il voto firmato con la propria chiave privata.
I passi sono i seguenti:

- A controlla se V è un elettore e se non lo è rifiuta l'amministrazione
- controlla se non ha già firmato il certificato elettorale di V; se lo ha fatto rifiuta l'amministrazione
- controlla se la firma è valida. Se lo è prosegue col protocollo
- firma il tutto con la propria chiave privata e lo restituisce al votante
  $A \rightarrow V$: $S_A(b(\chi(M)))$

Alla fine della fase di amministrazione rende pubblico il numero di voti firmati. Durante tutta la fase di amministrazione prepara una lista dei votanti e di ciò che gli è stato mandato (voto firmato dall'elettore e non). Tale lista verrà utile in caso di problemi nella fase di apertura dei voti.

3. Votazione

Ora che l'elettore ha in mano il certificato elettorale autenticato dall'amministratore, può apprestarsi al voto vero e proprio, che consiste nell'invio del certificato al counter attraverso un canale anonimo.
I passi sono i seguenti:

- V toglie il blind factor, ottenendo il certificato elettorale vero e proprio
  $S_A(\chi(M))$
- quindi verifica che la chiave dell'amministratore sia corretta. In caso contrario inoltra un reclamo mostrando che $S_A(\chi(M))$ non è la versione firmata di $\chi(M)$ (si noti che non è necessario rivelare il voto)
- altrimenti invia al counter tramite canale anonimo sia $S_A(\chi(M))$ sia $\chi(M)$
  (V) → C: $S_A(\chi(M,)), \chi(M)$

4. Raccolta dei voti

Il counter raccoglie i certificati elettorali che gli arrivano e prepara una lista pubblica di tutti i voti. Ogni voto viene etichettato con l'indice all'interno della lista.
Dopo che tutti hanno votato il counter pubblica la lista

Entry	**Voto e informazioni aggiuntive**	
1	$S_A(\chi(M_j))$	$\chi(M_j)$
…	…	…
L	$S_A(\chi(M_i))$	$\chi(M_i)$
…	…	…

5. Apertura

La fase di apertura è la seconda circostanza in cui si chiede all'elettore di intervenire. A fine votazione ogni elettore deve spedire al counter attraverso il canale anonimo le chiavi con cui aprire il voto, indicando a quale riga della lista corrisponde il voto.
I passi sono i seguenti:

- V controlla se il numero di voti conteggiati dal counter è identico al numero di voti certificati dall'administrator. Se non è così può inoltrare un reclamo inviando il proprio blind factor. Chi riceve il reclamo (l'autorità che si suppone debba supervisionare tutto) può verificare se il voto di V compare nella lista dei voti certificati preparata da A
- Poi V controlla se il suo voto appare nella lista preparata da C. Se il suo voto non è listato può inoltrare un reclamo all'autorità di supervisione mostrando il suo voto ancora col commitment $S_A(\chi(M,)), \chi(M)$. In tal modo la privacy del voto non viene violata, mentre l'autorità di supervisione, grazie alla firma di A, può verificare che quello è un voto certificato
- Se va tutto bene V spedisce a C attraverso il canale anonimo l'indice della lista a cui corrisponde il suo voto (L) e le chiavi necessarie per togliere il commitment (k)
  (V) → C: L, k

6. Conteggio

Con le informazioni che gli elettori spediscono il counter è capace, per ogni riga della tabella, di togliere il commitment ed estrarre il voto. Man mano aggiunge alla lista dei voti conteggiati la chiave k usata per eliminare il commitment (ossia i due numeri casuali R_1 e R_2 nel caso la tecnica usata sia quella con le funzioni hash one-way descritta sopra) e il voto in chiaro.
Alla fine conta i voti e annuncia il risultato.

Entry	Voto e informazioni aggiuntive			
1	$S_A(\chi(M_j))$	$\chi(M_j)$	k_j	M_j
…	…	…	…	…
L	$S_A(\chi(M_i))$	$\chi(M_i)$	k_i	M_i
…	…	…	…	…

Sicurezza del protocollo

Lo schema proposto sopra soddisfa i seguenti requisiti

Completezza (completness)

Premesso che tutte le parti in causa siano oneste, tutti i voti validi sono conteggiati correttamente e il risultato della votazione è affidabile

Solidità (soundness)

Anche se un elettore disonesto intende compromettere la votazione non può farlo.
L'unico modo che l'elettore ha per invalidare le elezioni è mandare voti non validi. Comunque, quando i voti vengono conteggiati, tutti i nodi vengono al pettine. Non solo l'elettore non può mandare voti non validi, ma anche non può cambiare il suo voto grazie allo schema di bit-commitment.
N.B.: un counter disonesto potrebbe scartare in fase di conteggio i voti della parte avversaria fingendo che le chiavi mandate dagli utenti non siano valide per aprire il voto. Una soluzione potrebbe essere quella che l'elettore mandi le chiavi a diversi counter che non hanno interesse a collaborare (magari i candidati dell'elezione).

Privacy

Tutti i voti devono restare segreti. Anche se l'amministratore e il counter cospirano non possono associare il voto all'elettore.
L'associazione tra l'elettore e il suo voto è protetta dalla tecnica di firma cieca. Inoltre sia il voto sia le chiavi per togliere il commitment al voto sono mandate attraverso un canale anonimo. Se il canale anonimo e lo schema di firma cieca tengono, come si presuppone che facciano, è impossibile violare la privacy.
Inoltre, mentre in altri protocolli si richiede all'elettore di rivelare il proprio voto in caso di contestazioni, qui non è necessario.
Durante la votazione, se l'amministratore rimanda all'elettore un voto firmato in maniera irregolare, basta che l'elettore mostri $S_A(\chi(M))$, $\chi(M)$ per rivelare l'imbroglio.
Durante l'apertura dei voti, se il contatore non ha incluso nella lista il voto di V, questi può inoltrare un messaggio mostrando nuovamente la coppia $S_A(\chi(M))$, $\chi(M)$.
In ogni caso non serve che mostri il voto.

Non riusabilità (unreusability)

Posto che non si riesca a rompere lo schema di firma cieca, nessun elettore può votare 2 volte
Se potesse farlo, avrebbe ottenuto due voti firmati regolarmente dall'amministratore, contro l'ipotesi

Eleggibilità (eligibility)

Posto che non si riesca a rompere lo schema di firma digitale ordinaria, nessun tra i non aventi diritto al voto può votare
Se potesse farlo, vorrebbe dire che è riuscito a passare all'amministratore due voti regolarmente firmati di cui almeno uno non sarebbe firmato con la propria firma. Cioè avrebbe contraffatto una firma di qualcun altro, contro l'ipotesi.

Imparzialità (fairness)

Niente deve influenzare il voto (per esempio mostrando prima del tempo l'andamento delle votazioni.)
Questo è banale, visto che il conteggio inizia solo dopo la fine della votazione e fino ad allora i voti sono protetti dallo schema di bit-commitment

Verificabilità (verifiability)

Nessuno può falsificare il risultato delle elezioni. Meglio: posto che nessuno si astenga dal votare e che nessuno possa contraffare lo schema di firma digitale ordinaria allora, anche se l'amministratore e il counter cospirano, non possono cambiare il risultato delle votazioni.
L'unica possibilità che il counter ha di imbrogliare è di non inserire un voto valido nella lista. Comunque, il protocollo è studiato perché questo genere di truffe risulti immediatamente durante l'apertura dei voti.
Così non resta che analizzare la possibilità di truffare dell'amministratore. Se nessun avente diritto al voto si astiene, non è possibile per l'amministratore inserire voti fasulli.

Il sistema E-Vox

Mentre il protocollo descritto in FOO è teorico e tralascia alcuni aspetti implementativi, E-Vox [4] è un sistema completo per le votazioni elettroniche. Ossia non solo definisce un protocollo completamente definito ma ne esiste anche un'implementazione in Java pienamente funzionante e utilizzata in occasione delle votazioni studentesche al MIT.

Altri sistemi

Il documento che descrive E-Vox cita altri sistemi di voto elettronico. Su tutti sostiene di aver alcuni vantaggi.

Pericles (MIT)

E' un sistema basato su C che gira sotto Mosaic.
Progettato e utilizzato per le elezioni studentesche
Difetti:

- gira sotto Kerberos e questo ne limita la diffusione
- è un sistema a singolo server, il che ne riduce la sicurezza

Pregi:

- ha un'ottima interfaccia grafica
- lavora bene per il tipo di elezione per cui è stato progettato

Princeton

Implementazione di FOO (non ne parla, sostiene solo di avere alcuni vantaggi su di esso

Sensus

Sviluppato da Lorrie Cranor e Ron Cytron [6] è un'altra implementazione di FOO.
Difetti:

- come FOO è un protocollo a 3 passi. Questo ne limita la facilità d'uso.
- Come FOO presuppone l'uso di un sistema di chiavi pubbliche per tutti gli elettori, cosa che comporta il problema della distribuzione delle chiavi

Sensus è senz'altro un'implementazione di FOO molto più aderente al protocollo originario rispetto a E-Vox, con caratteristiche quindi molto più simili a quelle di FOO. Questo lo rende più adatto a votazioni con un numero maggiore di elettori e inoltre mantiene la caratteristica del protocollo FOO per cui se due entità colludono non possono compromettere la votazione (questi pregi comunque non sono citati nel documento)

Aspetti del protocollo FOO su cui intervenire

Facilità d'uso

FOO fallisce soprattutto nell'essere user-friendly. L'elettore è coinvolto in 3 circostanze diverse, due delle quali obbligatorie per la validazione del voto:

- durante la votazione spedisce al counter il proprio certificato elettorale autenticato dall'amministratore
- dopo la fine della votazione, nella fase di apertura, deve spedire le chiavi al counter per togliere il commitment verificando di essere stato correttamente conteggiato
- infine, dopo lo spoglio, l'elettore dovrebbe controllare se al suo voto è stato correttamente tolto il commitment

Questo essere coinvolto in 3 giornate diverse (una per ogni circostanza) rende questo protocollo non adottabile da molti potenziali utenti.

Inoltre FOO è un sistema teorico e ci sono diversi aspetti pratici tralasciati. In particolare sono tralasciate le problematiche di comunicazione, la distribuzione delle chiavi e la gestione dei reclami.

Comunicazioni

Sono tralasciate tutte le problematiche classiche relative alle comunicazioni (intercettazione dei messaggi, manomissione dei dati durante la trasmissione e l'implementazione stessa del canale anonimo).

Distribuzione delle chiavi

La distribuzione delle chiavi non è presa in considerazione. Questo è particolarmente problematico perché ad ogni elettore deve essere assegnato un paio di chiavi nello schema delle firme digitali

Gestione dei reclami

FOO è progettato in modo che neppure in caso di contestazioni l'elettore sia costretto a dichiarare il proprio voto. Comunque non si fa menzione al sistema formale con cui i reclami devono essere inoltrati.

Obiettivi di E-Vox

L'obiettivo di E-Vox è sviluppare un sistema sicuro, user-friendly e stand-alone per un'elezione di piccola scala. L'efficienza non è tra gli obiettivi primari. Vediamo in dettaglio questi aspetti

Sicuro

E-Vox soddisfa le sette proprietà specificate in FOO con una premessa importante: tutte le proprietà reggono purché non ci sia una collusione tra due entità coinvolte nel protocollo.
Questa premessa è stata aggiunta per ridefinire le proprietà di sicurezza poiché alcune scelte implementative hanno indebolito il protocollo originario.
In particolare c'è il problema che mentre il canale anonimo in FOO si suppone sicuro (anche se non si specifica come implementarlo) qui l'uso di un server anonimyzer fa sì che se il counter e l'anonimyzer collaborano possono rompere la privacy dei votanti

User-friendly

Tutto il sistema è stato concepito con la semplicità d'uso in mente e così anche il protocollo descritto da FOO è stato ampiamente rivisto in vista di una maggiore comodità dell'utente.
In particolare, mentre il protocollo descritto in FOO e alcune sue implementazioni richiedono che l'utente interagisca in 3 circostanze differenti, due delle quali obbligatorie per convalidare il voto, E-Vox prevede che l'utente interagisca solo in 2 circostanze, una per votare e l'altra di controllo e che solo la prima sia obbligatoria. In questo modo solo la fase di registrazione e quella di voto sono obbligatorie in E-Vox (fasi obbligatorie in ogni sistema di votazione, elettronico o meno).
La questione della registrazione è lasciata appositamente aperta: non viene specificato il modo in cui preparare il database degli aventi diritto al voto. Tipicamente chi vuole votare andrà nell'ufficio di registrazione apposito, proverà la sua identità e sceglierà lo "user id" univoco e la "password" all'interno di un programma che l'utilizzatore di E-Vox si preoccuperà di mettere a disposizione.
Per votare l'elettore avrà bisogno semplicemente di un browser Web, dopodiché potrà fare il download dell'applet apposito, inserire il proprio "user id" e la password e premere pochi tasti e il gioco è fatto. Sia la registrazione che la votazione non dovrebbero richiedere più di qualche secondo.

Stand-alone

Il sistema è studiato per essere stand-alone. Non ha bisogno di strutture apposite, ma tutto quello che serve all'elettore è un browser Web che supporti gli applet Java .Il tutto è implementabile (e

implementato) usando la libreria crittografica del JDK 1.1, che oggi tutti i browser supportano. L'ambiente protetto in cui vengono eseguiti gli applet Java è una garanzia per l'integrità del sistema.

Dimensioni dell'elezione

Secondo gli autori, il codice da loro realizzato, anche se testato con solo un centinaio di utenti, è capace di gestire elezioni con fino a poche migliaia di elettori. Incrementando la potenza dei server e la larghezza di banda delle connessioni si può arrivare a gestire elezioni con decine di migliaia di elettori. Inoltre, per poterlo utilizzare per elezioni con un maggior numero di elettori si può ricorrere all'artifizio di spezzare le elezioni in un numero di votazioni più piccole.

Soluzioni adottate

Autenticazione

Mentre FOO si basa su un sistema a chiavi pubbliche usato per le firme digitali, E-Vox fa uso di un sistema a password. I motivi sono due:

- per facilità d'uso, in quanto le password sono conosciute e ben accette anche da utenti inesperti
- per aggirare il problema della distribuzione delle chiavi. Il momento più sicuro in cui distribuire le chiavi sarebbe la registrazione, ma questo obbligherebbe gli utenti a ricordarsi chiavi lunghe almeno un centinaio di bit (mentre le password sono di pochi caratteri e soprattutto sono scelte dagli utenti stessi)

Comunicazioni

Come abbiamo visto FOO prevede due tipi di comunicazioni, quelle normali e quelle su canale anonimo. Mentre per le prime non prevede nessun tipo di protezione oltre a quella crittografica fornita dal protocollo, per le seconde non suggerisce quale tecnologia adottare.
Anche se grazie alle protezioni crittografiche del protocollo un osservatore che si limita a intercettare e leggere i messaggi mandati sul canale normale non può provocare danni, nella realtà ci sono diversi tipi di attacchi contro cui tutelarsi o anche semplicemente cose che possono andare storte (rumore sul canale, perdita parziale o totale di un messaggio, sostituzione parziale o totale di un messaggio).

Canali sicuri

Per questa ragione tutte le comunicazioni in E-Vox avvengono su canali sicuri, ossia canali protetti da un protocollo crittografico.
Se A vuole aprire un canale sicuro con B, prima di tutto apre una connessione, per esempio un socket TCP (A deve avere un generatore di numeri casuali, B una coppia di chiavi pubblica/privata).
Quindi A genera una chiave di sessione S, che è una stinga di byte a caso che userà come chiave per l'algoritmo BlowFish[1]. Ora A prende il messaggio M, due stringhe casuali di byte di allineamento k1 e k2 e un MAC (Message Authentication Code) e cripta il tutto con la chiave di sessione S.
Infine A invia a B il messaggio criptato con S e la chiave S stessa criptata con la chiave pubblica di B.:

$$A \rightarrow B: \quad S(M, k_1, k_2, MAC = H(M, k_1, k_2,)), PK_B(S)$$

[1] BlowFish è un algoritmo di block cypher (quindi a chiave simmetrica) a 64 bit inventato da Schneier [1]. L'algoritmo è estremamente compatto e Schneier assicura che è significativamente più veloce di DES su macchine moderne (con processori a 32 bit e grosse chache)

Il MAC, come si vede nella riga sopra, è semplicemente un hash dei valori precedenti e serve per assicurare l'integrità del messaggio. Il destinatario dovrà semplicemente prendere M, k_1, k_2, riapplicare su di essi la funzione di hash H e confrontare il risultato con il MAC spedito.
Se il protocollo richiede che B risponda ad A potrà farlo usando la stessa chiave di sessione

B → A: S(…), PK_B(S)

In questo modo non c'è nessun bisogno che A abbia una coppia di chiavi asimmetriche.

La sicurezza del canale è garantita in diversi modi. L'uso di Blowfish rende la comunicazione segreta. Il layer TCP protegge dal rumore sul canale e dalla perdita occasionale di pacchetti. Il MAC fa sì che nessuna perdita o sostituzione parziale dei dati passi inosservata. Il codice che implementa l'invio e la ricezione dei messaggi farà uso di timeout. Infine, il protocollo stesso impedisce che si possa sostituire per intero il messaggio.

Canali anonimi

La soluzione adottata dai progettisti di E-Vox è quella di adottare un singolo server anonimizzatore (anonymizer) che lavora utilizzando le connessioni sicure viste sopra.
L'anonymizer (che chiamerò N) si pone tra l'elettore (V) e il counter (C). L'elettore non invia il certificato elettorale direttamente al counter perché altrimenti questi potrebbe risalire alla sua identità dall'indirizzo IP. Prepara comunque il certificato elettorale nel modo descritto sopra per le connessioni sicure in modo che solo il counter possa leggerlo.

S(certificato elettorale,…), PK_C(S)

(nella riga sopra sostituisco il padding e il MAC con dei puntini per semplificare la trattazione).
A questo punto V prende questi due oggetti criptati e li invia all'anonymizer premurandosi di usare una connessione sicura.

V → N: S'(S(certificato elettorale,…), PK_C(S),…), PK_N(S')

L'anonymizer raccoglie per tutto il tempo delle elezioni i voti criptati in questa maniera, salvandoli in un database dove non registra informazioni su chi li ha mandati. Dopo la fine delle elezioni li manda in blocco al counter in ordine sparso.
Si noti che l'anonymizer o chiunque ascolti sul canale tra V e N può risalire all'IP del votante ma non può conoscere il suo voto, mentre il counter o chiunque ascolti le comunicazioni tra N e A ha in mano i certificati elettorali ma non può associarli al votante.

Distribuzione delle chiavi

Prima di partire con le elezioni tutti i server devono aver generato le proprie chiavi e devono essere venuti in possesso delle chiavi degli altri server.
Il modo in cui questo avviene è volutamente non specificato e potrebbe essere fatto facendo uso di corrieri umani fidati.

Gestione degli errori

Rispetto al protocollo FOO viene aggiunto un server Commissioner (supervisore) a cui l'applet con cui gli elettori votano e gli altri server mandano i messaggi di errore. Il Commissioner provvede a memorizzarli in un file di log, mentre anche gli altri server mantengono un file di log nel caso ci siano problemi di comunicazione o comunque per verifica.
Il Commissioner sarà a sua volta supervisionato da una commissione di persone, che sia durante che dopo le elezioni prenderà decisioni su quali azioni intraprendere in risposta ai reclami. Se per esempio una percentuale elevata di elettori dovesse inoltrare reclami su firme non valide ricevute dall'amministratore, la commissione umana potrà sospendere immediatamente la votazione per verificare che il server di amministrazione non sia compromesso.

Il protocollo rivisto

In pratica nel protocollo di E-Vox ci sono le seguenti parti:

- l'elettore generico (V)
- il server amministratore (A)
- il server counter (C)
- il server anonymizer (N)
- il server commissioner

Le fasi in cui si snoda sono 5 invece di 6:

1. l'elettore prepara il voto e lo invia all'amministratore
2. l'amministratore lo restituisce firmato
3. l'elettore consegna all'anonymizer il proprio voto
4. l'anonymizer raccoglie i voti e dopo lo scadere del tempo utile per le votazioni invia tutti i voti al counter in ordine sparso
5. il counter fa le sue verifiche, pubblica la lista dei voti conteggiati, esegue il conteggio e annuncia il risultato

Vediamole in dettaglio

1. Preparazione

L'elettore prepara il voto e lo invia all'amministratore per farselo certificare.
I passi sono i seguenti:

- V prepara il plaintext
  M
- gli applica un commitment (usando HMAC-SHA) e un blind factor
  $\chi(M) = R_1, H(R_1, R_2, M)$
  $b(\chi(M))$
- infine invia all'amministratore il proprio userID, la password e il voto firmato
  $V \rightarrow A$: userID,password, $b(\chi(M))$

2. Amministrazione

L'amministratore riceve il voto dell'elettore, controlla se l'elettore ha diritto al voto e se non ha già votato. Se tutti questi controlli vanno a buon fine restituisce il voto firmato con la propria chiave privata.
I passi sono i seguenti:

- A controlla se userID e password sono corretti, altrimenti rifiuta l'amministrazione
- controlla se non ha già firmato il certificato elettorale di V; se lo ha fatto rifiuta l'amministrazione
- firma il tutto con la propria chiave privata e lo restituisce al votante
  $A \rightarrow V$: $S_A(b(\chi(M)))$

Alla fine della fase di amministrazione A pubblica una lista dei votanti per ciascuno dei quali memorizza userID, $b(\chi(M))$, $S_A(b(\chi(M)))$.

3. Votazione

Ora che l'elettore ha in mano il certificato elettorale autenticato dall'amministratore, può apprestarsi al voto vero e proprio, che consiste nell'invio del certificato all'anonymizer (di queste operazioni se ne occupa automaticamente l'applet)
I passi sono i seguenti:

- V verifica che la chiave dell'amministratore sia corretta (se non lo è inoltra un reclamo)
- quindi toglie il blind factor, ottenendo il certificato elettorale vero e proprio
  $S_A(\chi(M))$
- a questo punto V invia all'anonymizer il messaggio che l'anonymizer dovrà spedire in un secondo momento al counter. Questo messaggio contiene $S_A(\chi(M,))$, $\chi(M)$, M, k_1, k_2 (dove

k_1, k_2 sono le chiavi necessarie insieme al messaggio stesso per aprire il commitment). Si noti che c'è ridondanza ($\chi(M)$ potrebbe essere ricostruito da M, k_1, k2 e confrontato con quello firmato)

V → N: [V → C : S_A ($\chi(M,)$), $\chi(M)$, M, k_1, k_2]

4. Raccolta dei voti

L'anonymizer (non il counter) raccoglie i voti che gli arrivano via via e li salva ciascuno in un file separato, senza alcuna informazione sulla loro origine.
Allo scadere del tempo utile alla votazione invia tutti insieme i voti al counter in ordine sparso e pubblica una lista dei messaggi mandati.

5. Conteggio

Il counter prima rimuove i messaggi duplicati (cioè esattamente identici, chiavi di sessione incluse). Quindi controlla tutte le firme dell'amministratore. Infine come nel tradizionale FOO pubblica una lista di voti conteggiati, conta i voti e annuncia il risultato.
Tutte le liste sono rese pubbliche.

Entry	**Voto e informazioni aggiuntive**		
…	…	…	…
L	$S_A(\chi(M_i))$	$\chi(M_i)$	M_i, $k1_i$, $k2_i$
…	…	…	…

Osservazione

Si noti che il commitment non è più strettamente necessario. Nel protocollo originario è utilizzato principalmente per impedire al counter di contare i voti finché non si è al termine delle elezioni (nella fase dell'apertura). L'uso dell'hashing nel protocollo E-Vox è più dovuto alla necessità di rendere più efficiente la firma del messaggio in fase di amministrazione che a quello di fare commitment.

Sicurezza del protocollo rivisto

Completezza (completness)

Premesso che tutte le parti in causa siano oneste, tutti i voti validi sono conteggiati correttamente e il risultato della votazione è affidabile

Solidità (soundness)

Anche se un elettore disonesto intende compromettere la votazione non può farlo.
Ci sono due casi: o l'elettore (o chiunque altro) manda al counter un voto malcostruito (ad esempio con chiavi di commitment sbagliate), nel qual caso danneggia solo se stesso.
Oppure manda più volte il proprio voto regolarmente certificato (e questo caso è previsto in fase di conteggio dei voti).
Un possibile attacco potrebbe essere il DoS, nel caso un server riceva un gran numero di richieste che lo bloccano. Nell'implementazione di E-Vox ogni server ha una lista nera di indirizzi IP, che sono quelli da cui ha ricevuto troppe richieste.

Privacy

Tutti i voti restano segreti. Solo se l'anonymizer e il counter cospirano la privacy può essere rotta (ma questo viola l'assunto di non-collusione).
Il voto blindato consegnato all'amministratore non può essere associato a nessuna delle informazioni ricevute dal counter (voto in plaintext, voto con commitment, voto con commitment

firmato dall'amministratore). Quindi se amministratore e counter cospirano non possono violare la privacy. Qualsiasi parte esterna al sistema non avrebbe nessuna conoscenza in più per poterlo fare, potendo accedere solo alle liste pubbliche preparate dai due server.
In caso di contestazione l'elettore non è tenuto a rivelare il proprio voto. Se riceve un voto firmato male dall'amministratore basta che mostri il voto blindato con e senza firma dell'amministratore. Se il voto va perso prima o durante la fase di conteggio, l'elettore può mostrare ancora gli stessi valori.
Se il server anonimo è corrotto da solo non può fare nulla, a meno che non passi informazioni al counter sulla provenienza dei messaggi.

Non riusabilità (unreusability)

Posto che non si riesca a rompere lo schema di firma cieca, nessun elettore può votare 2 volte
Se potesse farlo, avrebbe ottenuto due voti firmati regolarmente dall'amministratore, contro l'ipotesi

Eleggibilità (eligibility)

Posto che non si riesca a rompere lo schema di firma digitale ordinaria, nessun tra i non aventi diritto al voto può votare
Se potesse farlo, vorrebbe dire che è riuscito a passare all'amministratore due voti regolarmente firmati di cui almeno uno non sarebbe firmato con la propria firma. Cioè avrebbe contraffatto una firma di qualcun altro, contro l'ipotesi.

Imparzialità (fairness)

Il conteggio dei voti non deve influenzare la votazione
è banalmente vero, visto che il conteggio inizia solo dopo la fine della votazione

Recuperabilità (Recoverability)

Posto che l'elettore faccia tutto quello che è necessario per farsi convalidare un voto valido e che nessuna delle parti in causa colluda, un voto scartato da una qualunque delle parti può sempre essere recuperato
Se il counter scarta un voto, il server anonimo può dare la sua lista al commissioner, il quale incaricherà il counter per decriptarla in modo da trovare il voto mancante.
Se l'anonymizer scarta un voto, l'elettore può mostrare il voto con commitment, blindato e firmato dall'amministratore.
Se infine è l'amministratore a scartare un voto (cioè rifiuta di firmare) l'elettore inoltra un reclamo al commissioner. Se risulta che l'amministratore l'ha segnato può mostrare il voto segnato al commissioner.

Possibili migliorie del protocollo

Ricevute di ritorno

Per evitare quei problemi in cui una parte dice di aver spedito un messaggio mentre l'altra sostiene di non averlo ricevuto, si può aggiungere un meccanismo di ricevute di ritorno.
Prendiamo il caso in cui A invii un messaggio a B e voglia essere sicuro che B non potrà negare di averlo ricevuto.
A → B: $S(M, k1, k2, MAC), PK_B(S)$ (come prima)
B → A: $SK_B(H(M, k1, k2, MAC))$
Se B nega di aver ricevuto il messaggio, A può mostrare la ricevuta e mostrare che B ha avuto in mano H(M, k1, k2, MAC) che solo A può riprodurre.
Quindi mostrando $SK_B(H(M, k1, k2, MAC))$, M, k1, k2, MAC è in grado di dimostrare che B ha letto il messaggio M.

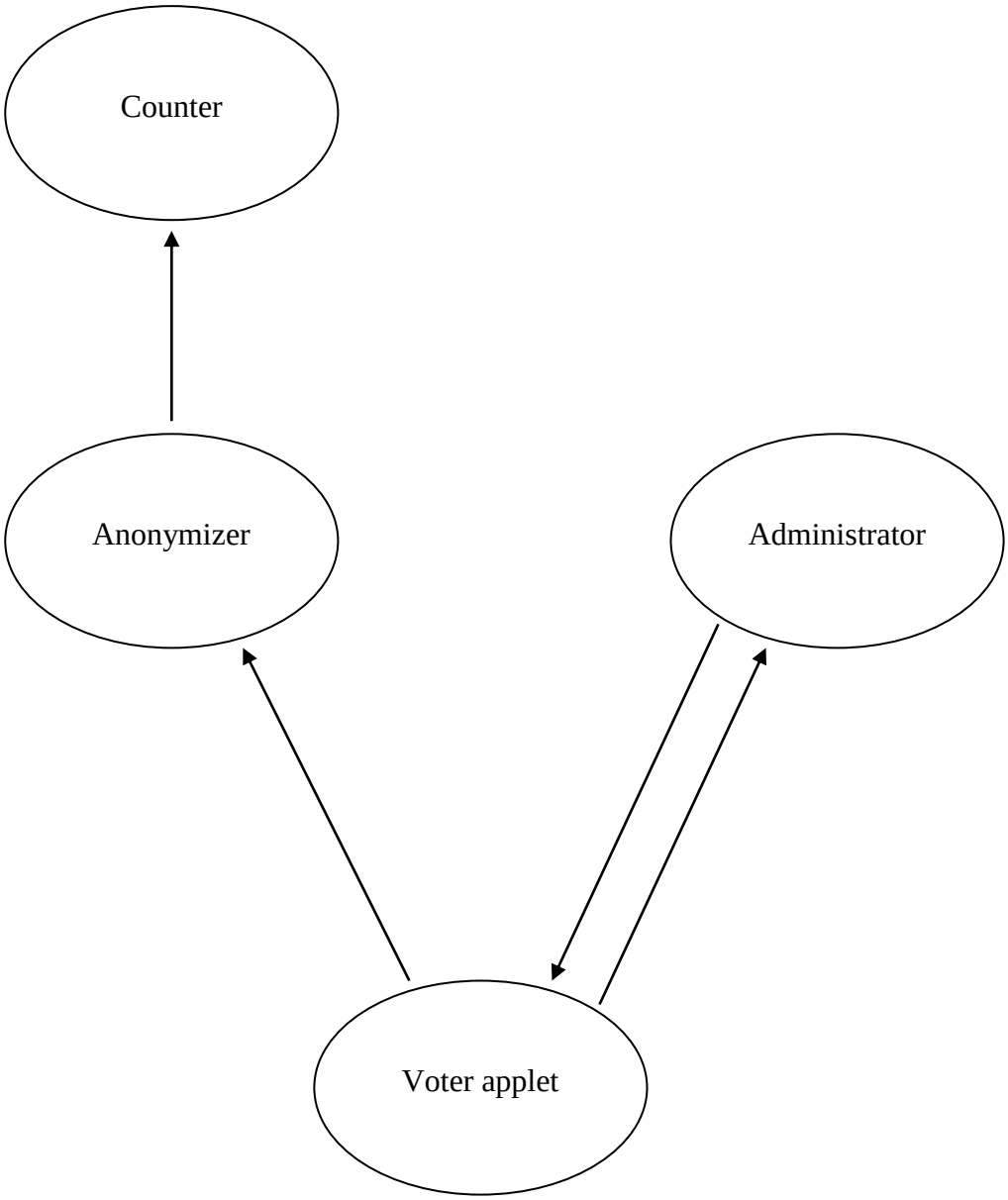
Amministratori ridondanti

L'amministratore è l'entità singola più potente. Solo la sua firma può convalidare un voto. Questo lascia a un amministratore disonesto un certo margine di manovra.
Per esempio, pochi minuti prima dello scadere del tempo utile per l'elezione, può prendere una percentuale tra gli elettori che non hanno votato e votare per loro. Chi non vota quasi certamente non controllerà se ci sono dei brogli fatti a suo nome, mentre per quanto riguarda eventuali ritardatari può sempre far credere che siano elettori disonesti che stanno cercando di votare due volte. In casi come questi, in cui si consente agli elettori di astenersi dal voto (senza mandare un messaggio di astensione), il protocollo non consente di distinguere tra elettore disonesto e amministratore disonesto.
Un possibile approccio a questo problema è creare tante unità amministrative che non hanno interesse a colludere (per esempio una per ogni candidato), ciascuna delle quali dovrà firmare il voto. Perché il counter accetti un voto dovrà avere raccogliere un numero minimo t di firme valide su n totali. Naturalmente occorre che sia $t > n/2$, altrimenti lo stesso elettore potrebbe preparare due voti validi facendoli firmare da due gruppi diversi di amministratori.

Anonymizer ridondanti

L'ultimo problema del protocollo che prendiamo in esame è quello in cui l'anonymizer scarti dei voti validi (anche in buona fede, magari per un crash temporaneo). La cosa salterà certamente fuori ma solo dopo che le elezioni si sono concluse, creando notevoli problemi.
Per proteggersi da questo problema basta far sì che l'elettore mandi il proprio voto a diversi anonymizer in parallelo: basterà che il voto arrivi da uno solo perché venga correttamente conteggiato.

Sistema E-Vox



Bibliografia

1. Bruce Schneier, *Applied Cryptography*, John Wiley & Sons, New York, 1994
2. M. Merritt. Cryptographic Protocols, Ph.D. dissertaion, Georgia Institute of Technology, GIT-ICS-83/6, Feb 1983.
3. Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptography*--AUSCRYPT '92, volume 718 of *Lecture Notes in Computer Science*, pages 244-251, Gold Coast, Queensland, Austrailia, 13-16 December 1992, Springer-Verlag.
   Presenta uno schema di votazione adatto per elezioni a larga scala che preserva privacy e fairness
4. Mark A. Herschberg. Secure Electronic Voting Using the World Wide Web, Master's Thesis, Massachusetts Institute of Technology, June 1997.
   E-Vox: p*resenta un'implementazione del protocollo descritto nell'articolo di Fujioka, Okamoto e Ohta.*
5. Brandon William DuRette. Multiple Administrators for Electronic Voting, Bachelor's Thesis, Massachusetts Institute of Technology, May 1999.
   Espande il programma EVOX per amministratori multipli
6. Lorrie Faith Cranor and Ron K. Cytron, Sensus: A Security-Conscious Electronic Polling System for the Internet. *Proceedings of the Hawai`i International Conference on System Sciences* , January 7-10, 1997, Wailea, Hawaii, USA
   Presenta un'implementazione dello schema di Fujioka, Okamoto, e Ohta