Protocolli per voto telematico

Versione 1 - 09/10/2018
di Paolo Russo

Essendo un programmatore di computer, mi pare un po' assurdo che per votare ci si debba recare in un seggio elettorale, dove tutto viene fatto a mano con pezzi di carta, matite e scatoloni e dove tutta la protezione dal pericolo di brogli si basa sulla fiducia nell'integrità di un gruppo di sconosciuti che si controllano a vicenda. Possibile che nel ventunesimo secolo non si possa fare di meglio? E i computer che ci stanno a fare?
Qualcuno dirà: ma certo, è facilissimo, basta computerizzare tutto: l'elettore si reca nel seggio e si siede davanti a un terminale, o vota direttamente dal PC di casa o con il suo smartphone, collegandosi a un server che... ALT!
Tutto bello e facile, eccetto per un problemino: cosa garantisce che non ci siano brogli? Che il software non sia stato manipolato, che qualcuno non si infiltri nel server, che nessuno sostituisca i numeri finali con altri... barare è possibilissimo; chi potrebbe controllare, e come?
Si dirà: c'è poco da fare, se si fanno le cose a mano devi fidarti delle persone, se si fanno tramite computer devi fidarti dei computer. E invece no. I computer possono fare cose che le persone, a mano, non possono fare. Non c'è un modo di farli lavorare in maniera tale che la correttezza del loro operato sia verificabile? Non dovrebbero gli elettori pretendere questo grado di verificabilità? In dettaglio, un sistema sicuro dovrebbe garantire le seguenti cose:
- che il voto del singolo elettore non venga alterato o eliminato;
- che non sia possibile aggiungere voti di elettori inesistenti o che non hanno votato;
- che il voto sia segreto.

E possibilmente dovrebbe garantire queste cose anche nel caso che i server vengano violati. In altre parole, la sicurezza del sistema non può essere un mero atto di fede nella serietà degli addetti ma deve essere intrinseca o verificabile dall'esterno. Naturalmente, questo implica che se qualcosa va storto, se ad esempio un elettore si accorge che il suo voto è stato alterato, deve essere possibile dimostrarlo.
Il problema è se un protocollo di votazione telematica ad affidabilità verificabile sia possibile e sufficientemente pratico da usare. Nel seguito esporrò un paio di metodi di votazione. Non suggerisco che si debba adottarli così come sono (un protocollo reale sarebbe certamente più complicato), ma si potrebbe almeno usarli come spunto per rifletterci su. Il mio obiettivo è di mostrare che un sistema di votazione informatizzato e affidabile è probabilmente possibile e che questa possibilità meriterebbe di essere presa abbastanza sul serio.
Premetto che non so se sistemi del genere siano già in uso o allo studio, non mi sono documentato. So che c'è un sistema di voto elettronico in Svizzera che consente di verificare il proprio voto, ma non l'ho studiato in dettaglio e non so quali altre garanzie fornisca.
Ovviamente qui non mi sto ponendo il problema di quanto sicuri siano il computer, il sistema operativo o l'antivirus usato dall'elettore; il sistema che l'elettore usa per collegarsi è sotto la sua responsabilità.

Premessa: crittografia

La comune crittografia informatica si basa su una chiave (un blocco di byte), che viene usata per cifrare e decifrare. Esistono algoritmi di crittografia standardizzati e ritenuti ragionevolmente sicuri. Ci sono fior di esperti di crittografia, in tutto il mondo, che fanno di tutto per trovarci delle falle.

Premessa: hash crittografico

Uno hash è un piccolo blocco di dati che dipende da un blocco di dati più grande. Ad esempio, se prendo la Divina Commedia, sostituisco ogni lettera dell'alfabeto che vi compare con un numero (A = 1, B = 2... Z = 26) e li sommo tutti quanti, il numerone che ottengo alla fine è uno hash della Divina Commedia. Se uno cambia anche un solo carattere di un'unica terzina, ad esempio una A con una B, la somma viene diversa. Gli hash hanno vari utilizzi in informatica. Il più classico è quello di consentire di verificare che un blocco di dati inviato lungo una linea di comunicazione inaffidabile non sia stato alterato.
Uno hash crittografico è uno hash con una caratteristica aggiuntiva: dato un blocco di dati e il suo hash, non deve esistere un modo ragionevolmente efficiente per generare un blocco di dati diverso (in una qualsiasi parte, anche piccola) che abbia lo stesso hash. Ad esempio, la somma dei caratteri dell'esempio precedente non è uno hash crittografico: prendo la Divina Commedia, sostituisco una A con una B, poi da qualche altra parte una B con una A, e il gioco è fatto: ho alterato il testo ma la somma dei caratteri rimane la stessa. Ho "craccato" lo hash (non che fosse difficile).
Naturalmente, un software che tenta a casaccio tutti i blocchi di dati possibili prima o poi ne produrrà uno con lo hash desiderato. E' solo questione di tempo e qualunque hash può essere craccato. E` impossibile evitarlo. Tuttavia, se si può stimare che il tempo medio che un computer potente possa impiegare a farlo sia dell'ordine dei miliardi di anni, si può concludere che non esiste un modo efficiente di "craccare" lo hash, che quindi può dirsi crittografico. Ovviamente, gli hash crittografici fanno cose molto più complicate che sommare i codici dei caratteri. Esistono algoritmi standardizzati, come il classico MD5 (ritenuto ormai non più tanto sicuro) e varie varianti di SHA.

Premessa: crittografia a chiave pubblica e firma digitale

A differenza della crittografia tradizionale che è a chiave simmetrica, cioè la stessa chiave che si usa per cifrare un testo serve anche a decifrarlo, la crittografia a chiave pubblica (o asimmetrica) si basa sul fatto che servono due chiavi diverse per cifrare e decifrare. Generalmente quella per cifrare viene divulgata liberamente, per cui viene detta "pubblica". Io genero una coppia di chiavi e ne divulgo una. Chiunque voglia mandarmi un messaggio segreto può cifrarlo con la mia chiave pubblica; solo io posso decifrarlo perché non ho mai dato a nessuno la chiave privata. In pratica, dato che la crittografia asimmetrica richiede un sacco di calcoli ed è quindi ragionevolmente usabile solo per cifrare dati piccoli, il mittente fa una cosa un po' diversa: genera una chiave simmetrica casuale, cifra il messaggio con quella, cifra la chiave simmetrica con la mia chiave pubblica e mi manda il tutto. Con la mia chiave privata decifro la chiave simmetrica e con quella decifro il messaggio.
Tramite crittografia a chiave pubblica si può anche firmare un documento. Per firmare un documento praticamente cifro il documento (o meglio, cifro uno hash crittografico del documento, per risparmiare tempo di elaborazione) con la mia chiave privata. Tutti possono decifrare la firma usando la mia chiave pubblica, verificando così che era stato cifrato usando (evidentemente) la mia chiave privata, che solo io possiedo. Ciò dimostra che sono stato io a farlo, presumibilmente di mia volontà, per cui quest'atto equivale a una firma. Generalmente è opportuno includere nei dati da firmare anche una marca temporale (data e ora); nel seguito sarà dato spesso per sottinteso.

Generalità: server, firme, controlli

In ogni metodo è previsto che esistano vari server con compiti ben definiti. Userò il termine "server" in senso molto lato: potrebbe essere un singolo computer, come anche un cluster; potrebbe essere una macchina fisica come un cluster di macchine virtuali in cloud... L'importante è che all'atto pratico, dal punto di vista del client che ci si collega, il tutto si comporti come se fosse un singolo computer (come succede ad esempio con Google; www.google.it è un singolo URL, ma ovviamente nessuno si aspetta che l'intero servizio sia ospitato in un singolo computer).

In generale ogni server deve avere una chiave privata con cui firma certificati. Quasi sempre, quando un server fornisce un servizio, deve fornire un certificato firmato. Spesso l'intera transazione con il server è firmata, nel senso che ogni messaggio scambiato nella transazione non solo è firmato ma contiene anche tutti i messaggi precedentemente scambiati con le loro firme, un po' come una tipica email aziendale in cui si lasciano quotati tutti i messaggi precedenti in modo che da ogni messaggio si possa ricostruire tutta la discussione precedente. Se un server rifiuta di fornire un servizio, deve sempre fornire una motivazione documentata e firmata. **Nota**: una vulnerabilità intrinseca di questo tipo di transazione è l'ostruzionismo. Un server potrebbe rifiutare di fornire il servizio senza fornire giustificazioni e facendo cadere la connessione. Naturalmente la cosa è un po' scoperta. Le transazioni interamente firmate possono sempre essere riprese dal messaggio a cui sono state interrotte, per cui il client può riaprire la connessione. Se un server continua ripetutamente a rifiutare di portare a termine una transazione è davvero una violazione un po' troppo scoperta, quindi mi pare difficile che qualche server malevolo possa tentare questa strada, ma comunque è una possibilità da tenere a mente. Alla peggio, si potrebbero istituire dei server di controllo (anche privati, più sono e indipendenti l'uno dall'altro, meglio è) attraverso cui sia possibile far transitare tutti i dati della connessione. Se un server rifiutasse di rispondere a una richiesta, il server di controllo potrebbe testimoniarlo, anche se non potrebbe dimostrarlo. Se una mezza dozzina di tentativi di richiesta fatti passare attraverso server di controllo diversi e indipendenti fallissero tutti, non sarebbe ancora una prova schiacciante ma ci andrebbe molto vicino. Ciò richiederebbe però di usare protocolli di sicurezza un po' diversi dal classico e standard SSL per le singole connessioni; bisognerebbe inventare un protocollo apposta che consenta al server di controllo di decifrare i dati che passano attraverso di lui, ma non di alterarli. In realtà il pericolo di alterazione sarebbe minimo, dato che la tipica transazione è costituita da messaggi firmati.

Primo metodo

Il server delle identità

Per votare, ogni elettore deve avere un'identità elettorale informatica custodita dallo Stato sui suoi server e usabile sia per le elezioni e i referendum previsti dalla Costituzione sia, eventualmente, per votazioni volute e gestite da privati per fini loro.
L'elettore genera sul suo computer una coppia di chiavi asimmetriche (con opportuno software preferibilmente open source), chiamiamole chiavi di identità, e si reca in Comune per depositare la chiave pubblica con il suo nominativo (esibendo la carta d'identità). Nei server statali viene memorizzato un record contenente nome, indirizzo, codice fiscale e chiave pubblica. I server verificano che non ci siano doppioni. L'elettore può depositare quando vuole una nuova chiave pubblica, che va a sostituire la precedente. L'elettore deve custodire bene la sua chiave privata, preferibilmente in un file criptato con password (se perde il file o la password o teme che qualcuno se ne sia impadronito può generare un'altra coppia di chiavi e ridepositare in Comune quella pubblica).
L'archivio statale delle identità è interrogabile via rete da chiunque: dato un codice fiscale, i server forniscono la chiave pubblica corrispondente (e firmano il tutto a richiesta con una loro chiave privata). L'elettore può quindi verificare in qualsiasi momento che la sua identità elettorale sia valida (e che nessuno abbia imbrogliato o pasticciato).

I server di voto

Per votare si usano dei server di voto, che non hanno niente a che fare con quelli statali che archiviano le identità. Potrebbe esserci un unico server, ma almeno per le votazioni di Stato (elezioni, referendum...) è più sicuro che siano tanti. Ad esempio potrebbe essercene uno per ogni regione. E' ancora più sicuro che il voto non venga inviato a un solo server, ma sia inviato in copia a

dei server secondari. Si potrebbe stabilire una regola per cui, ad esempio, chi risiede in Veneto deve inviare il voto al server del Veneto e in copia a Lazio e Sicilia; chi risiede in Lazio invia in Lazio e in copia a Lombardia e Puglia; eccetera. S'intende che ogni server deve sapere se sta ricevendo un voto realmente per lui o solo come copia. Tutto questo unicamente per far sì che manomissioni a un singolo server vengano facilmente scoperte confrontando i dati con quelli degli altri server.

Il voto

L'elettore vota via rete dal proprio computer, con opportuno software client che si presume affidabile in quanto sotto il completo controllo dell'elettore, che può scegliere di usare il software che gli pare (preferibilmente open source gestito dallo Stato, ma in generale basta che rispetti le specifiche del protocollo di comunicazione). L'elettore inserisce nel software una stringa identificativa della votazione (pubblicamente disponibile nei server di voto), il suo voto (chiamiamolo V), il suo codice fiscale e la sua chiave di identità privata. La procedura di voto si articola in tre fasi distinte.

Fase 1: creazione di un alias

Il software genera un'altra coppia di chiavi asimmetriche; chiamiamole chiavi di alias. Contatta il server delle identità con una connessione protetta (SSL), si fa identificare tramite le chiavi di identità (ad esempio con un classico challenge-response: il server genera dati casuali e il client glieli restituisce firmati, non entro nei dettagli) e gli invia una richiesta di definizione di un alias. La richiesta è un messaggio contenente il codice fiscale, la chiave pubblica di alias e la stringa identificativa della votazione ed è firmato con la chiave privata di identità. Il server verifica la firma del richiedente, poi verifica che il richiedente non possieda già un alias per quella votazione e che quell'alias non sia già stato scelto da qualcun altro. Se non ci sono problemi il server memorizza l'alias per quella votazione, associandolo all'identità del richiedente. Il server dovrà mantenere l'alias nell'archivio per un certo tempo (30 giorni?), in modo da consentire eventuali indagini in caso di contestazioni. Se esiste già un alias, il server avverte il client che avverte l'utente; quest'ultimo può scegliere tra richiedere al server l'alias precedente o sostituirlo con quello nuovo.
A questo punto il client, durante la stessa sessione con il server o anche in una successiva, sempre dopo essersi fatto identificare, può richiedere al server delle identità vari certificati, cioè messaggi contenenti vari dati, sempre comprensivi di marca temporale (data e ora), firmati dal server con una sua chiave privata che deve avere valore di prova legale.
Tanto per cominciare è buona cosa, per ogni evenienza, richiedere al server di certificare l'insieme identità + alias + identificativo di votazione. Quello che però serve per andare avanti con la procedura di voto è un certificato "anonimo" alias + identificativo di votazione, senza l'identità del votante.

Fase 2: invio del voto parziale

Il software client genera una stringa di bit casuali, lunga quanto il voto (che dev'essere di lunghezza fissa, specificata nei parametri della votazione). Chiamiamola A. Calcola anche B = V xor A (xor = OR esclusivo: in binario, 0 xor 0 = 0, 0 xor 1 = 1, 1 xor 1 = 0). Genera inoltre un altro dato casuale, chiamiamolo Z, abbastanza lungo da rendere poco probabile che due elettori generino per caso lo stesso Z.

Il software contatta uno ad uno tutti i server di voto configurati (uno o più), seguendo la stessa procedura con ognuno, tramite connessione protetta ma senza alcun controllo dell'identità dell'elettore. Prima di tutto invia il certificato "anonimo" rilasciato dal server statale delle identità. Il server di voto verifica la firma del certificato e crea un record nel suo archivio dei voti. Questo

record contiene il certificato e lo spazio per il voto. Il server risponde certificando la richiesta ricevuta più l'indice del record che ha creato. Se per caso l'alias che ha ricevuto esisteva già nell'archivio, il server lo accetta ugualmente e si limita a ricertificare la nuova richiesta con l'indice del record già assegnato. In ogni caso il server DEVE rispondere certificando qualcosa se l'alias che ha ricevuto è valido (firmato validamente dal server statale).
Il client risponde rimandando al server il messaggio appena ricevuto con in più B, Z, marca temporale e firma digitale (ottenuta con la chiave segreta di alias). Il server, verificata la firma, deve rispondere controfirmando il messaggio.
Se la connessione cade in un qualunque momento, il client può riprenderla dall'ultimo messaggio ricevuto in poi come se nulla fosse successo, dato che le risposte del server devono dipendere solo dai dati che ha in archivio.

Fase 3: completamento del voto

A questo punto il server delle identità sa solo che Tizio ha intenzione di votare, ma non conosce il voto. I server di voto sanno solo che qualcuno ha votato qualcosa, ma non sanno né chi né cosa.
Il client contatta nuovamente il server statale delle identità, si fa identificare come nella fase 1 e gli certifica alias + identificativo votazione + A. Il server memorizza anche A nel suo archivio accanto all'alias e risponde controfirmando. Il server deve comunque memorizzare anche le richieste firmate che ha ricevuto, per poterle esibire in caso di contestazione (quanto meno, l'ultima ricevuta per ogni tipo di richiesta di modifica).
A questo punto, per quanto concerne l'elettore, la procedura di voto si è conclusa.

I server di voto tirano le somme

A "urne" chiuse, ogni server di voto completa la procedura. Invia al server delle identità un grosso insieme di alias (minimo mille, altrimenti la richiesta viene rifiutata). Riceve in cambio una ricevuta (un identificativo del file trasferito). Il server delle identità deve mantenere memorizzato quel file per un certo tempo (ore o giorni). Ricevuto l'identificativo, il server di voto chiede al server delle identità quali identità corrispondono a quel file di alias con dato identificativo (il server di voto non deve mai mandare due file identici o comunque file contenenti anche un solo alias già inviato in un file precedente; in caso di crollo della connessione durante il trasferimento si fa tabula rasa e si ricomincia, in caso di crollo della connessione successivo all'invio si usa l'identificativo). Il server delle identità risponde con un insieme di identità reali in ordine casuale e un insieme di "voti" A associati a quegli alias in ordine corretto. Il server di voto può adesso calcolare i vari voti corretti V = A xor B, ma dato che le identità sono in ordine casuale non sa chi ha votato cosa. Alla fine il server di voto ospita tre file, uno con le identità dei votanti (come minimo: codice fiscale e città, consigliabile l'identità completa con tanto di indirizzo) in ordine sparso, uno con i soli voti nell'ordine originale, uno con i voti nell'ordine originale ognuno accompagnato dallo Z inviato assieme al voto. Questi file devono essere pubblicamente consultabili da chiunque. In particolare, l'elettore sa in che posizione nel file si trova il suo voto (perché ha ricevuto l'indice del record durante la transazione) e può verificare che, almeno per quanto concerne il suo voto, non ci siano stati brogli. Può bastargli scaricare il file dei voti, che dovrebbe essere piuttosto piccolo, ma se vuole essere sicuro che quel voto sia davvero il suo scarica il file contenente i voti e gli Z e controlla che lo Z sia quello che ha inviato lui. In entrambi i casi potrebbe richiedere non l'intero file ma solo la zona dove c'è il suo voto (calcolabile dal'indice di record). La verifica dello Z è importante nel caso in cui sia i server di identità che di voto siano stati compromessi e il server di voto abbia avuto modo di decodificare il voto subito dopo averlo ricevuto. In tal cso il server potrebbe barare e assegnare un indice di record già usato per altri votanti che hanno espresso lo stesso voto, in modo da far spazio nel file per voti opposti. Il controllo di Z serve appunto a garantire che a ogni elettore sia stato assegnato un record distinto.

Se ci sono stati brogli, l'elettore può dimostrarlo grazie alle ricevute firmate dai server. Ogni bravo cittadino coscenzioso dovrebbe verificare il suo voto (e più di qualcuno dovrebbe ricontarli tutti). Basta che una piccola percentuale di cittadini lo faccia per rendere praticamente impossibile che brogli di entità significativa passino inosservati.

Rischi di brogli

La principale controindicazione di questo schema di voto è che è impossibile evitare che qualcuno si faccia consegnare (ad esempio a pagamento o dietro minaccia) la chiave di identità privata di qualcun altro. Non c'è niente da fare, è un problema intrinseco del voto a distanza. D'altra parte, anche se nel normale voto in cabina elettorale è formalmente vietato portarsi dietro telefonini con fotocamera, proprio per evitare che qualcuno venga minacciato a votare in un certo modo, non ho l'impressione che tale divieto venga fatto rispettare molto. Non ho mai visto perquisire nessuno.

Il secondo maggior pericolo in quest'approccio è che qualcuno manometta il server delle identità e riesca ad aggiungere identità fittizie. E' per ridurre questo pericolo che le identità reali devono essere in chiaro nel file finale del server di voto. Se una data città risulta avere un numero anomalo di votanti (troppi rispetto alla popolazione nota), la cosa è sospetta. Se sono pubblici gli indirizzi, diventa facile (per chi vive davvero in quella zona) accorgersi di un indirizzo inesistente o di persone che non risultano esistere nel proprio caseggiato. Sarebbe comunque consigliabile che le autorità effettuassero controlli a campione. Basterebbe verificare mille votanti a caso per rendere molto probabile scoprire un qualsiasi broglio di incidenza superiore all'1% dei votanti.
Aggiungere voti con identità fittizie direttamente nel server di voto è più difficile, sia perché il server statale delle identità potrebbe tenere il conto di quante identità ha dovuto decodificare per ogni server di voto, sia perché una volta che esiste un archivio di identità pubblicamente consultabile non è difficile verificare in automatico l'intero file delle identità pubblicato dal server di voto. Anche solo una verifica a campione darebbe buoni risultati.
E' invece poco plausibile che qualcuno tenti di creare voti fittizi associati a identità reali di persone che non hanno votato: basterebbe che una minima percentuale di non votanti andasse a verificare di non comparire nel file delle identità per smascherare l'imbroglio.
La segretezza del voto è fragile se uno riesce ad accedere al server delle identità. Tuttavia è un server statale e lo stato non ha molto interesse a sapere chi ha votato cosa (semmai, un governo tirannico potrebbe cercare di alterare il voto, ma questo, come visto, è molto più difficile).
Se qualcuno se lo stesse chiedendo: il motivo per suddividere il voto V in A e B è quello di evitare che il server di voto possa tentare di barare facendo cadere la connessione ogni volta che si tenta di votare in un certo modo. Serve anche a rendere meno facile assegnare lo stesso record a più votanti. Naturalmente, se a imbrogliare è lo Stato stesso controllando tutti i server tutti insieme, questa precauzione di suddividere il voto non serve a nulla.

Il fatto che l'elettore possa in qualsiasi momento reimpostare alias, A e B (il che non è strettamente necessario, ma sarebbe utile per poter recuperare da pasticci vari) può complicare un po' il protocollo. Ad esempio può accadere che durante la fase in cui si tirano le somme il server di voto scopra che l'alias che gli è stato inviato non è più valido perché sostituito da un altro (in tal caso il server statale deve certificargli questo fatto). In questo caso il server di voto deve lasciare vuoto lo spazio per il voto nel record corrispondente; è un voto annullato a cui non corrisponde nessuna identità. Il file delle identità deve avere tanti elementi quanti i voti validi nei file dei voti.

Secondo metodo

Un inconveniente del primo metodo è che non garantisce tanto bene l'anonimato qualora i server

delle identità e di voto siano gestiti dallo stesso ente (o siano stati entrambi craccati). Si può fare di meglio? Un'ideuzza l'avrei, ma temo che non sia di uso molto pratico. Comunque, vediamola; magari può costituire uno spunto per ulteriori riflessioni.

L'equivalente non informatico di questo secondo schema è il seguente: c'è una cesta di schede elettorali numerate. Queste schede sono come banconote, difficili da falsificare e ognuna ha un numero di serie diverso. Chiunque può prendere anonimamente una scheda dalla cesta (o anche più d'una, se ne ha voglia). Però va timbrata e non si può farlo in maniera anonima. L'addetto alla timbratura controlla i documenti dell'elettore, appone un timbro sulla scheda (mentre l'elettore ne copre il numero di serie con una mano perché l'addetto non possa leggerlo) e segna sui suoi registri che l'elettore tal dei tali ha avuto il suo timbro e non può averne altri. L'elettore scrive il suo voto sulla scheda e la mette anonimamente nell'urna. Solo lui sa qual e` il numero della sua scheda e potrà in seguito verificare che nessuno abbia alterato il suo voto.
Detto così sembra inutilmente complicato, ma informaticamente ha senso.

Premessa: scambio di chiavi Diffie-Hellman-Merkle

Riassumo qui brevemente un protocollo di scambio di chiavi che si basa sul fatto che per un computer è abbastanza facile calcolare $[A^B]_P$ (A elevato alla B, il tutto in aritmetica intera modulo P), dove P è un numero primo e A, B e P hanno centinaia di cifre, mentre ricavare B da $[A^B]_P$ e da A è praticamente impossibile in quanto non è noto un algoritmo efficiente per farlo (nonostante sia stato ovviamente cercato).
Alice e Bob devono comunicare, ma qualcuno può intercettare le loro comunicazioni. Potrebbero usare la crittografia a chiave pubblica, ma facciamo finta che non esista. La crittografia simmetrica richiede che sia Alice che Bob usino la stessa chiave segreta. Come fanno a stabilirne una se tutte le loro comunicazioni sono spiate?
Alice e Bob si mettono d'accordo su un numero primo P molto grande e su un numero G che sia generatore del gruppo ciclico dell'aritmetica modulo P (generalmente si usa 2 o 5). Per i dettagli rimando alla pagina di Wikipedia (https://it.wikipedia.org/wiki/Scambio_di_chiavi_Diffie-Hellman). Naturalmente queste comunicazioni sono spiabili, quindi anche la spia ora conosce P e G.
Alice genera un numero casuale molto grande A. Calcola $[G^A]_P$ e lo comunica a Bob.
Bob genera un numero casuale molto grande B. Calcola $[G^B]_P$ e lo comunica ad Alice.
Alice prende il $[G^B]_P$ comunicatole da Bob, lo eleva alla A ed ottiene $[([G^B]_P)^A]_P = [G^{AB}]_P$.
Bob fa la cosa speculare: $[([G^A]_P)^B]_P = [G^{AB}]_P$.
Entrambi ora conoscono $[G^{AB}]_P$, che useranno come chiave crittografica per le loro comunicazioni future. La spia conosce P, G, $[G^A]_P$ e $[G^B]_P$, ma da questi dati non può ricavare $[G^{AB}]_P$.
Questo metodo mi ha sempre colpito per la sua astuzia. E poi c'è chi pensa che la matematica sia fine a sé stessa.
Nel seguito adotterò uno schema Diffie-Hellman-Merkle un po' modificato per adattarlo a uno scopo diverso. I numeri segreti saranno ben quattro: A, B, C e D.

Secondo metodo: la votazione inizia

In questo schema esistono quattro tipi di server: il server delle identità che abbiamo già visto, il server di imparzialità, il server dei certificati di voto e il server di voto (può esserci più di un server di voto). Questi ultimi due tipi di server possono essere gestiti dallo stesso ente come anche no, ma comunque concettualmente fanno due cose distinte. Ogni server ha una chiave privata e una pubblica. Sono inoltre pubblici un numero primo P e un generatore G, usati da tutti.
Quando si decide di fare una votazione, viene scelto un codice identificativo per la votazione. Il server statale delle identità genera un numero A casuale dedicato a quella votazione e che rimarrà segreto. Se i server di voto sono più d'uno, il server delle identità deve generare un A diverso per

ogni server di voto. Il server di imparzialità fa pochissimo: deve solo generare una singola coppia di chiavi asimmetriche (chiamiamole chiavi di imparzialità), pubblicare quella pubblica (comunicandola a chiunque gliela chieda) e pubblicare quella privata subito dopo la fine del periodo in cui si può votare (alla "chiusura delle urne").

Fase 1: richiesta del certificato di voto (pescare una scheda dalla cesta)

Il client genera una coppia di chiavi di alias (ma a differenza del metodo precedente NON la registra presso il server delle identità: rimane un alias totalmente anonimo), poi contatta il server dei certificati di voto, gli invia la chiave pubblica di alias e richiede un certificato di voto (specificando votazione e server di voto). Il server genera un numero casuale B e risponde con un certificato firmato contenente copia della richiesta, eventualmente un numero progressivo per rendere unico ogni certificato (non sono sicuro che serva a qualcosa), il numero B cifrato con la chiave pubblica del server di voto desiderato e il numero $[G^B]_P$ in chiaro. Questo server è dedicato a fare solo questo genere di transazioni, non fa altro e non ha necessità di memorizzare niente delle transazioni effettuate (cosa forse utile per mitigare eventuali attacchi DoS), a parte incrementare l'eventuale numero progressivo.

Fase 2: richiesta del diritto di voto (far timbrare la scheda)

Il client genera un numero casuale C e calcola $[([G^B]_P)^C]_P = [G^{BC}]_P$, poi contatta il server statale delle identità, autenticandosi come al solito con le chiavi di identità. Fa richiesta firmata di elevazione a potenza del numero $[G^{BC}]_P$, precisando per quale votazione e quale server di voto ne ha bisogno, al che il server statale trova l'A corrispondente (dopo aver controllato che l'elettore abbia diritto di votare su quel server di voto, ad esempio guardando in che regione risiede l'elettore) e risponde certificando $[G^{ABC}]_P$ (oltre alla copia della richiesta, per buona misura) e segna nel suo archivio che quella persona ha appena usato e consumato il suo diritto di votare in quella votazione. Naturalmente memorizza il certificato che ha inviato. Nessun'altra richiesta di elevazione a potenza per quella stessa votazione avanzata da quella persona verrà soddisfatta; naturalmente il rifiuto deve venire giustificato fornendo copia del risultato firmato precedente.

Fase 3: voto (scrivere il voto e mettere la scheda nell'urna)

Il client genera un altro numero casuale D e calcola due numeri X e Y:
$X = [([G^{ABC}]_P)^D]_P = [G^{ABCD}]_P$,
$Y = [([G^C]_P)^D]_P = [G^{CD}]_P$.
Comunica questi due numeri al server di voto, in un messaggio firmato contenente anche il certificato di voto, nonché il voto V cifrato con la chiave pubblica di imparzialità e il solito dato casuale Z. Il server di voto verifica la firma del certificato rilasciato dal server dei certificati di voto, verifica che la firma del client sia coerente con la chiave pubblica di alias contenuta nel certificato, verifica che quel certificato non sia stato già usato da qualcuno per votare (nel qual caso deve giustificare il suo rifiuto fornendo copia della precedente richiesta di voto firmata: è responsabilità dell'elettore assicurarsi che la sua chiave di alias privata non sia usata da altri), decifra B contenuto nel certificato usando la sua chiave privata, verifica che quadri con il $[G^B]_P$ nel certificato, poi calcola $[Y^B]_P$ (che dovrebbe corrispondere a $[G^{BCD}]_P$). Infine, il server di voto contatta il server delle identità, comunicandogli X e $[Y^B]_P$ e chiedendogli di verificare che sia $[([Y^B]_P)^A]_P = X$. Se il server risponde di sì il voto deve essere accettato e memorizzato assieme a Z e il server deve rilasciare una ricevuta firmata di tutta la transazione comprensiva dell'indice del record (come nel primo metodo). Naturalmente, se una qualunque delle verifiche fallisce, il server di voto è tenuto a dare una

spiegazione documentata e firmata del motivo.

I server di voto tirano le somme

A urne chiuse il server di voto deve contattare il server di imparzialità per ottenere la chiave privata con cui decifrare tutti i voti ricevuti.
In questo secondo metodo il server di voto non ha alcuna conoscenza delle identità dei votanti. Dev'essere il server delle identità a pubblicare un file con le identità di tutti gli elettori che hanno richiesto il diritto di votare. Naturalmente, è possibile che qualcuno richieda il diritto di voto ma poi non concluda la procedura e non voti (chiamiamolo "votante inconcludente"). Il file dei voti nel server di voto potrebbe quindi contenere meno voti del previsto. Questo potrebbe creare una falla: se la percentuale di voti in meno fosse stimabile (dopo un certo numero di votazioni simili), o se qualcuno ne conoscesse il numero esatto prima della conclusione della votazione avendo manomesso il server delle identità, qualcuno potrebbe manomettere il server di voto e generare un po' di voti falsi, non collegati a nessuna identità reale, in numero non superiore a quello degli inconcludenti. Sarebbe piuttosto difficile accorgersi di una tale manipolazione. Questo potrebbe essere un problema se la percentuale di elettori inconcludenti fosse rilevante. Faccio notare che l'astensione deve essere comunque sempre prevista tra i voti possibili. E` consigliabile che ogni elettore consumi il suo diritto di voto, anche solo per astenersi, giusto per poter verificare che il suo diritto di voto non sia stato rubato da qualcuno.

La logica dietro le elevazioni a potenza

Se per caso qualcuno fosse stato frastornato da tutte quelle elevazioni a potenza, provo a rispiegare la cosa in termini più umani. Se A è un numero segreto, $[G^A]_P$ può essere considerato la sua versione pubblica. Per calcolare un numero come $[G^{ABCDE...}]_P$ bisogna per forza conoscere tutti i numeri segreti A, B, C, D, E... tranne al più uno, uno qualunque, di cui è sufficiente conoscere la versione pubblica. Se ad esempio conosciamo tutti i numeri tranne D, ma conosciamo $[G^D]_P$, lo eleviamo a tutti gli altri: $[(((([G^D]_P)^A)^B)^C)^E]_P = [G^{ABCDE}]_P$. In sostanza, fornire un numero come quello equivale a dimostrare di conoscere tutti i segreti che contiene, tranne al più uno.
Per poter votare, l'elettore deve dimostrare di conoscere A, o più esattamente di aver potuto usufruire almeno una volta della conoscenza di A posseduta dal server delle identità, e il server la fornisce a ogni elettore solo una volta per ogni votazione.
La verifica che X coincida con $[([Y^B]_P)^A]_P$ dice al server di voto due cose: primo, che il client ha inserito in X una qualche conoscenza di B, ma dato che B è segreto, doveva essere la conoscenza della variante pubblica di B ricevuta nel certificato; secondo, che il client ha inserito in X una qualche conoscenza di A, che però doveva essere dell'A segreto visto che il B utilizzato era quello pubblico e non può essere stata usata la forma pubblica di più d'uno di quei numeri. Dato che anche A è segreto, il client deve aver ottenuto quella conoscenza dal server delle identità.
Il motivo per cui con questo metodo il voto rimane segreto (almeno in teoria) è che anche se i server barassero e si parlassero non avrebbero comunque modo di collegare il $[G^B]_P$ e il $[G^{CD}]_P$ presso il server di voto con il $[G^{BC}]_P$ che è arrivato al server delle identità. L'elevazione alla C o alla D, numeri casuali, distrugge l'informazione più o meno come farebbe l'or esclusivo con un numero casuale.

Il motivo per cui temo che questo metodo, per quanto sensato sulla carta, sia poco pratico nella realtà è che per evitare di essere identificato l'elettore dovrebbe sincerarsi di collegarsi ai vari server in tre sessioni distinte, lasciando passare un po' di tempo tra l'una e l'altra e cercando di cambiare indirizzo IP ogni volta, il che spesso si ottiene resettando il router, ma non è proprio garantito. Diciamo che l'accesso ai primi due server potrebbe essere fatto durante la campagna elettorale, ma farlo troppo tempo prima aumenterebbe la pericolosa percentuale degli inconcludenti. Inoltre, c'è il

problema che il server dei certificati deve effettuare operazioni computazionalmente pesantine per produrre i certificati di voto e non c'è garanzia che qualche client malevolo non gliene richieda a bizzeffe. Un modo per evitarlo potrebbe consistere nel rilasciare il certificato solo se il client risponde correttamente a un calcolo ancora più pesante di cui il server conosce facilmente il risultato (tecnicamente è una cosa fattibile) e che richieda al client alcuni secondi di elaborazione.

Conclusione

E' probabile che i metodi sopra descritti abbiano lacune e vulnerabilità a cui non ho pensato. Ammetto di non averci pensato tanto a lungo prima di scriverne. Spero però che possano essere comunque utili come base di partenza per riflessioni sul tema. Se qualcuno vuole segnalarmi falle o lacune è il benvenuto. Il mio indirizzo email si trova nel mio sito (da cui dovreste aver scaricato questo testo).