UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IAȘI
FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Rețele de mixare cu aplicații în protocoale de securitate

propusă de

Marius-Alexandru Avram

Sesiunea: februarie, 2020

Coordonator științific

Lect. Dr. Sorin Iftene

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IAȘI
FACULTATEA DE INFORMATICĂ

Rețele de mixare cu aplicații în protocoale de securitate

Marius-Alexandru Avram

Sesiunea: februarie, 2020

Coordonator științific

Lect. Dr. Sorin Iftene

Avizat,
Îndrumător lucrare de licență,
Lect. Dr. Sorin Iftene.
Data: ..... Semnătura: .....

Declarație privind originalitatea conținutului lucrării de licență

Subsemnatul **Avram Marius-Alexandru** domiciliat în **România, jud. Galați, mun. Galați, Str. Aviatorilor, nr. 1, bl. U10, sc.3, et. 1, ap. 44**, născut la data de **21 iulie 1997**, identificat prin CNP **1970721170011**, absolvent al Universității "Alexandru Ioan Cuza", **Facultatea de informatică** specializarea **informatică**, promoția 2020, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **Rețele de mixare cu aplicații în protocoale de securitate** elaborată sub îndrumarea domnului **Lect. Dr. Sorin Iftene**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data: ..... Semnătura:
.....

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Rețele de mixare cu aplicații în protocoale de securitate**, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test, etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea "Alexandru-Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent **Marius-Alexandru Avram**

Data: ..... Semnătura:

.....

Cuprins

Motivatie **2**

Introducere **4**

1 Algoritmi Criptografici **5**
1.1 RSA 5
1.2 Elgamal 6
1.3 Dovezi ZK 7

2 Rețele de mixare **8**
2.1 Tipuri de rețele 8
2.1.1 Mixnet de Decriptare 8
2.1.2 Mixnet Hibrid 9
2.1.3 Mixnet de Recriptare 10
2.1.4 Mixnet de Recriptare Universal 12
2.2 Vulnerabilități 13
2.3 Verificabilitate 16
2.3.1 Verificabilitatea Realizata de Expeditor 16
2.3.2 Verificabilitatea Realizata de Rețea 17
2.3.3 Verificabilitate universala 21

3 Aplicații în protocoale de securitate **23**
3.1 Tipuri de aplicații 23
3.2 Vot electronic 23
3.2.1 Specificații 24
3.2.2 Structura 26
3.2.3 Modele de scheme de votare 28
3.3 Licitație electronică 30
3.3.1 Specificații 31
3.3.2 Structura 33
3.3.3 Modele de scheme de licitație 34

4 Aplicație de vot electronic **36**
4.1 Arhitectura aplicației 36
4.1.1 Structura Aplicației 36

Concluzii **43**

Motivație

În trecut împreună cu apariția internetului și a răspândirii sale s-a observat o lipsă a intimității personale datorită așa numitei "traffic analysis problem". Ca răspuns la această problemă au apărut diferite scheme pentru păstrarea anonimității precum DC-nets bazată pe cunoscuta problema Dining Cryptographers [4] sau schema lui Benaloh [2] bazată pe sisteme Zero Knowledge Interactive Proof și scheme de partajare a secretelor. Din păcate aceste scheme sunt ineficiente pe scala mare sau necesită pregătiri speciale și nu pot fi folosite întotdeauna în practică din acest motiv. În lucrarea sa [5] din 1981, Chaum a construit un nou mecanism de păstrare a anonimității numit rețea de mixare sau mixnet. Ideea lui Chaum nu se bazează pe o autoritate de încredere și susține că este vulnerabilă doar în cazul în care toate serverele ce participă la mixare colaborează. Cu toate că în lucrarea sa inițială există probleme de eficiență și securitate, ea stă la baza multor scheme de anonimitate din ziua de astăzi care prin abordări diferite reușesc să evite aceste lipsuri. Schema lui Chaum se bazează pe criptografie asimetrică, mai exact algoritmul RSA [25]. O problema de eficiență apare astfel în urma duratei de criptare necesară pentru mascarea unui mesaj, schema inițială necesitând ca procedura de criptare să fie repetată pentru fiecare server prin care mesajul urma să treacă. Ineficiența creată de lungimea criptotextelor a fost rezolvată într-o nouă abordare [22] a lui Kurosawa et al. în care se folosește algoritmul lui Elgamal [9] pentru procesul de criptare. Avantajul acestui algoritm constă in faptul că procesul de criptare este realizat o singură dată de către utilizator folosind o singură cheie publică.

Din punct de vedere al securității au apărut modalități de prevenire atât active cât și reactive a atacurilor. Rețele de mixare cu reputație au fost sugerate in [8]. Ele constă în ținerea evidenței serverelor ce nu respectă protocolul. Se prefera însă schemele ce dau dovada de reziliență, mai exact combină păstrarea confidențialității cu posibilitatea de verificare a corectei funcționări. Aceste tipuri de rețele există, însă chiar dacă sunt extrem de puternice din punct de vedere al securității trebuie luată în calcul și eficiența lor pentru a putea fi folosite în practică. Această lucrare are astfel scopul de a prezenta aceste instrumente puternice din punct de vedere teoretic în aplicații critice precum votul electronic sau licitația electronică urmând ca la final să fie prezentată o implementare a unei aplicații de vot electronic.

Introducere

Libertatea de exprimare reprezintă un drept fundamental al omului ce permite unui individ sau unei comunități să își exprime opinia fără a exista o frică a cenzurii, discriminării sau a sancțiunilor legale. Trebuie luat în considerare faptul că acest drept nu permite încălcarea drepturilor sau defăimarea altor indivizi. În același timp trebuie respectate securitatea națională, ordinea publică și principiile morale ale populației.

Cu timpul se poate observa o trecere înspre publicarea informațiilor personale și a opiniei personale în mediul online. Acest lucru a pornit o nouă industrie bazată pe colectarea și gestionarea acestor informații în scopul generării de oportunități de marketing sau politice. În același timp libertatea de exprimare a avut de suferit datorită existenței unui public mult mai larg și divers și datorită fricii de repercursiuni generate de acest public.

Pe lângă libertatea de exprimare există și noi mecanisme de urmărire a comportamentului și a nevoilor personale. Aceste mecanisme sunt construite pentru a ușura viața de zi cu zi oferind informații special adaptate pentru fiecare utilizator în parte. Cu toate acestea se poate ajunge la cazuri de încălcare a vieții personale. Din aceste motive a apărut o mișcare ce dorește realizarea anonimității.

Prin anonimitate un individ își poate exercita dreptul de liberă exprimare fară a putea fi identificat. În același timp anonimitatea permite desfășurarea activităților fără a putea fi urmărit sau observat de alți indivizi. Dispare astfel frica de a fi judecat de cei din jur. În același timp anonimitatea contribuie la siguranța personală impiedicând spre exemplu aflarea locului în care un individ anonim se află. Datele personale, informațiile despre tranzacții bancare, parole secrete și altele nu pot fi legate de utilizator. Un alt aspect constă în separarea vieții personale de hobbyurile sau interesele unui individ.

Anonimitatea poate duce însă la probleme grave datorită lipsei de responsabilitate a celor ce se folosesc de ea. Anonimitatea permite ascunderea identității utilizatorilor ferindu-i de repercursiuni în cazuri ilegale. Astfel, cazuri de fraudă, atacuri cibernetice sau alte comportamente ce nu se află sub libera exprimare prezentate mai sus pot apărea datorită anonimității. În consecință folosirea acesteia trebuie să apară în scenarii responsabile.

În această lucrare vor fi prezentate protocoale ce facilitează libertatea de exprimare cu ajutorul unui mod responsabil de folosire a anonimității. Lucrarea se va sfârși cu prezentarea unei implementări a unei aplicații de vot electronic bazată pe rețele de mixare. Această abordare este benefică datorită avantajelor ce vor fi

prezentate în cele ce urmează.

În Capitolul 1 vor fi prezentate diferite instrumente criptografice folosite în rețele de mixare și în protocoale de securitate. Capitolul 2 constă într-o descriere a rețelelor de mixare prezentând tipuri generale de rețele și moduri de verificare a acestora. Capitolul 3 descrie aplicații ce utilizează rețele de mixare și o serie de protocoale folosite în aceste aplicații. Capitolul 4 reprezintă rezultatul informațiilor din capitolele anterioare. Acesta se ocupă de documentarea unei aplicații de vot.

Capitolul 1

Algoritmi Criptografici

Înainte de a descrie rețelele de mixare trebuie detaliați algoritmii criptografici pe care se bazează. Utilizarea criptografiei în aceste rețele are rolul de a ascunde informațiile transmise astfel încât observatorii traficului rețelei să nu le poată descifra.

1.1 RSA

Algoritmul lui R. Rivest, A. Shamir si L. Adleman cunoscut sub numele de RSA [25] stă la bazele criptografiei cu chei publice. Acesta se bazează pe dificultatea găsirii divizorilor întregi a unui număr de dimensiuni mari. Mai exact, dat fiind un număr compus de dimensiuni mari n, este dificil de găsit descompunerea sa $n = p_1^{e_1} * p_2^{e_2} * ... * p_k^{e_k}$ unde p_i reprezintă numere prime distincte și $e_i \geq 1$.

Vom începe prin a descrie generarea cheii publice și a celei private. Acest proces se desfășoară astfel:

1. Se generează două numere prime aleatoare și distincte p și q de dimensiuni apropiate și mari.
2. Se calculează $n = p * q$ și $\phi = (p-1)(q-1)$.
3. Se alege un număr întreg aleator e unde $1 < e < \phi$ și cel mai mare divizor comun dintre e și ϕ este 1.
4. Se calculează numărul întreg $d, 1 < d < \phi$ unde $ed \equiv 1 \ mod \ \phi$.

Cheia publică este reprezentată de perechea (n, e) iar cheia privată este d. Acestea se folosesc pentru criptarea respectiv decriptarea mesajelor destinate deținătorului cheilor.

Pentru criptarea unui mesaj se procedează astfel:

1. Se obține cheia publică (n, e) a destinatarului
2. Mesajul este reprezentat sub forma unui număr m în intervalul $[0, n-1]$. În cazul în care acesta este prea mare se poate trimite pe rând câte o parte din el.

3. Se calculează mesajul criptat $c = m^e \ mod \ n$.

Pentru decriptarea mesajului se folosește d pentru a obține $m = c^d \ mod \ n$ dat fiind că $ed \equiv 1 \ mod \ \phi$

1.2 Elgamal

Algoritmul lui Elgamal [9] este inspirat din protocolul de schimb de chei Diffie-Hellman. Acesta se bazează pe problema logaritmului discret și problema Diffie-Hellman.
Problema logaritmului discret:

Def. 1 *Dat fiind un număr prim p, un generator α a lui $\mathbb{Z}_p^*$ și $\beta \in \mathbb{Z}_p^*$ să se găsească numărul întreg $x, 0 \leq x \leq p-2$, astfel încât $\alpha^x \equiv \beta \ mod \ p$.*

Problema Diffie-Hellman:

Def. 2 *Dat fiind un număr prim p, un generator α a lui $\mathbb{Z}_p^*$ și $\alpha^a \ mod \ p$ și $\alpha^b \ mod \ p$, să se găsească $\alpha^{ab} \ mod \ p$.*

Avem în consecință următorul protocol asimetric:

- Generarea cheii publice si a celei private:
  1. Se generează aleator un număr prim de dimensiune mare p și un generator α al grupului $\mathbb{Z}_p^*$
  2. Se alege aleator un număr întreg a unde $1 \leq a \leq p-2$ după care se calculează $\alpha^a \ mod \ p$
  3. Cheia publică este (p, α, α^a). Cheia privată este a.
- Criptarea unui mesaj:
  1. Se obține cheia publică (p, α, α^a) a destinatarului.
  2. Mesajul este reprezentat sub forma unui număr m în intervalul $[0, n-1]$. În cazul în care acesta este prea mare se poate trimite pe rând câte o parte din el.
  3. Se alege un număr întreg aleator k din intervalul $1 \leq k \leq p-2$
  4. Se calculează $\gamma = \alpha^k$ și $\delta = m * (\alpha^a)^k \ mod \ p$
  5. Mesajul criptat $c = (\gamma, \delta)$ este trimis destinatarului
- Decriptarea unui mesaj:
  1. Se obține mesajul criptat $c = (\gamma, \delta)$
  2. Folosind cheia privată a se calculează $\gamma^{p-1-a} \ mod \ p$ unde $\gamma^{p-1-a} = \gamma^{-a} = \alpha^{-ak}$
  3. Se obține mesajul m calculând $(\gamma^{-a}) * \delta \ mod \ p$. (Se observă că $(\alpha^k)^{-a} * (\alpha^a)^k * m = \alpha^{ak-ak} * m = m \ mod \ p$)

1.3 Dovezi ZK

Pentru a dovedi faptul că o entitate cunoaște anumite informații secrete fără a divulga aceste informații se folosesc dovezi ZK (Zero Knowledge). Aceste dovezi sunt foarte folositoare în verificarea corectitudinii și integrității unui participant în protocoale criptografice. Ele sunt însă adesea costisitoare din punct de vedere a puterii de calcul lucru de care trebuie ținut cont în utilizarea lor repetată.

Protocolul Chaum-Pedersen

În lucrarea [6] este prezentat un protocol bazat pe problema logaritmului discret. Avem valorile (p, q, g, h) considerate drept cheia publică a schemei unde $h \in G_q \setminus \{1\}$. Cheia secretă corespunzătoare este $x = log_g h$. Fie $m \in G_q$ un mesaj. Semnătura lui m este $z = m^x$ unde $log_g h = log_m z$.Date fiind m și z avem astfel protocolul:

1. Se alege aleator un $s \in \mathbb{Z}_q^*$ și se calculează $(a, b) = (g^s, m^s)$. Această dovadă este trimisă celui ce se ocupă de verificare.

2. Odată primită dovada, verificatorul alege o valoare aleatorie $c \in \mathbb{Z}_q^*$ și o trimite înapoi.

3. Cel ce se ocupă de dovadă răspunde cu r unde $r = s + cx$.

4. În cazul in care $g^r = ah^c$ și $m^r = bz^c$ dovada este acceptată ca fiind corectă.

Capitolul 2

Rețele de mixare

2.1 Tipuri de rețele

În continuare vor fi prezentate diferite tipuri de rețele de mixare. Ele se bazează în general pe două tipuri de rețele. Avem rețele care se bazează pe decriptarea unui mesaj și rețele care recriptează un mesaj. Ambele tipuri de rețele schimbă aspectul mesajului pentru a face dificilă realizarea corespondenței dintre expeditor și destinatar.

2.1.1 Mixnet de Decriptare

Rețeaua originală propusă de Chaum [5] include adresa următorului corespondent în corpul mesajului. Pentru a opri un observator al rețelei din a citi conținutul mesajului se folosește algoritmul de criptare cu chei publice RSA [25]. Nu este exclusă și posibilitatea de a folosi un algoritm simetric, acesta necesitând un schimb de chei între servere și expeditor. Folosind cheia publică a serverului de mixare K_1 și cheia publică a destinatarului K_a , expeditorul trimite un mesaj de forma:

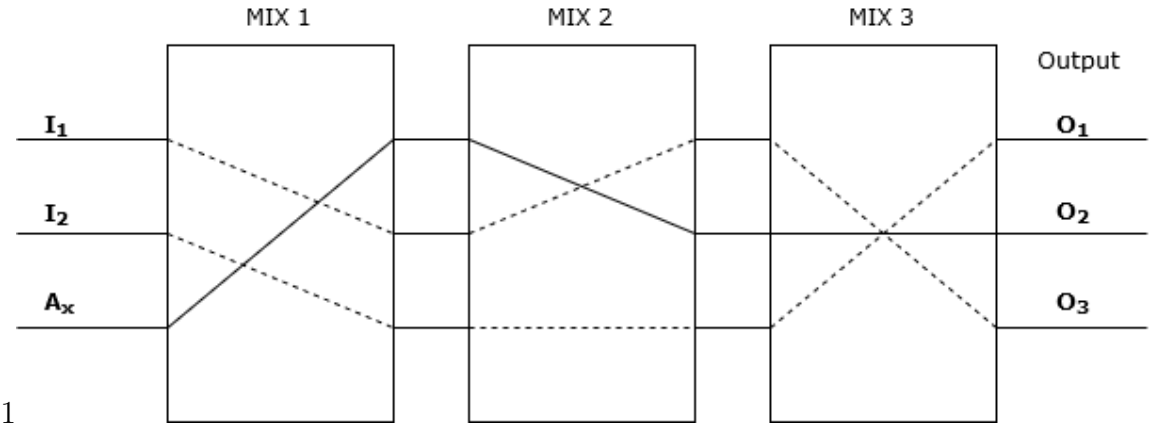
$$K_1(R_1, K_a(R_0, M), A) \longrightarrow^{K_1^{-1}} K_a(R_0, M), A$$

Obs. **În ecuația de mai sus precum și în cele următoare "," reprezintă concatenarea elementelor**

$K_1(M)$ reprezintă criptarea textului M folosind cheia publică K_1 . K_1^{-1} reprezintă cheia privată a serverului de mixare și este folosită pentru decriptarea mesajului unde $K_1^{-1}(K_1(M)) \longrightarrow M$.

RSA este un algoritm de criptare determinist. Folosind K_1 un atacator ce observă traficul rețelei poate cripta diferite posibile mesaje pentru a ajunge la inputul expeditorului. R_i este un număr aleator utilizat pentru a preveni ghicirea mesajului de către un atacator și este înlăturat după decriptare. M reprezintă mesajul propriu-zis și A este adresa destinatarului.

Securitatea rețelelor de mixare apare atunci când folosim o combinație de servere pentru a amesteca mesajele. Pentru fiecare server folosit se ia cheia publică și se criptează mesajul astfel:



11

Figura 2.1: Drumul unui mesaj de la A_x la ieșire.

$$c = K_n(R_n, K_{n-1}(R_{n-1}, ...K_2(R_2, K_1(R_1, K_a(R_0, M), A), A_1)..., A_{n-2})A_{n-1})$$

Mesajul trece prin cele n servere unde este decriptat și i se aplică o permutare secretă după cum se poate vedea și în figura 2.1. Dimensiunea și forma conținutului se schimbă vizibil în timpul decriptării și prin consecință poate fi urmărit traseul său. De asemenea apar probleme de eficiență la mesajele lungi datorită RSA [25].

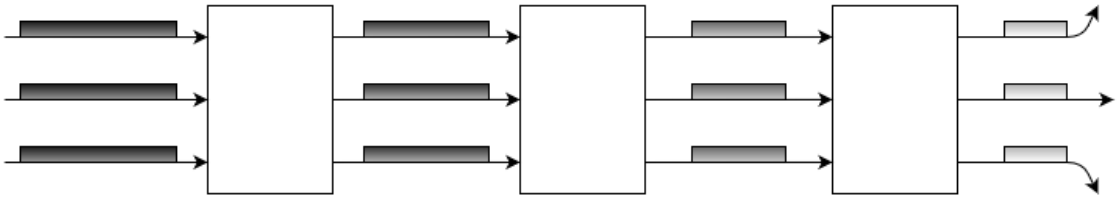
2.1.2 Mixnet Hibrid

Rețelele hibride sunt o variantă mai eficientă pentru rețelele de decriptare. Creșterea în dimensiune a criptotextului este mare la rețelele de decriptare. Criptografia cu chei simetrice nu are probleme când vine vorba de texte de dimensiuni mari și nu crește dimensiunea criptotextului așa cum o face un algoritm asimetric. Aceste rețele funcționează după același principiu precum cele de decriptare diferența fiind în faptul că se folosește un algoritm de criptare simetric pentru mesaj. Cheia pentru algoritmul simetric este inclusă împreună cu mesajul dar aceasta este criptată folosind cheia publică a serverului. Cheia simetrică nu crește în dimensiune pe parcursul operațiilor de criptare precum mesajele din rețelele de decriptare. O intrare pentru o rețea hibridă va arăta astfel:

$$c = A_1, K_1(k_1), k_1(A_2, K_2(k_2), k_2(...(A_{n-1}, K_{n-1}(k_{n-1}), k_{n-1}(A_n, K_n(A_a, K_a(m), R)))$$

Serverul decriptează mai întâi cheia simetrică k_i și o folosește pentru a decripta restul mesajului pe care îl trimite la adresa următoare.

Acesta este un exemplu general pentru rețelele hibride. În [16] în locul unei chei private se folosește o altă cheie publică pentru verificarea integrității și validității mesajului folosind ZKP (Zero Knowledge Proof). Există alte variante de rețele hibride și rețele de decriptare care adaugă informații de control. Aceste informații suplimentare îmbunătățesc securitatea și disponibilitatea rețelelor.



11

Figura 2.2: Decriptarea unui mesaj în rețelele mixte și de decriptare. Dimensiunea mesajului scade după trecerea printr-un server. Mesajele trebuie să aibă dimensiuni apropiate pentru a împiedica urmărirea și identificarea lor.

2.1.3 Mixnet de Recriptare

Rețelele de decriptare după cum sunt prezentate în secțiunile anterioare au unele dezavantaje evidente precum:

- Dimensiunea mesajului scade în timp ce traversează rețeaua.
- Expeditorul cunoaște conținutul mesajului și îl poate urmări.
- Expeditorul trebuie să adauge un strat de criptare pentru fiecare server din rețea.
- Ordinea decriptării trebuie să fie fixă. Astfel dacă un server este indisponibil procesul poate eșua.

În [22] apare o soluție pentru aceste probleme. Se folosește un nou sistem de criptare cu chei publice numit Elgamal [9] inspirat din protocolul Diffie-Hellman (DH). Este propusă o rețea de decriptare bazată pe algoritmul lui Elgamal.

Expeditorul relizează o singură operație de criptare pentru a pregăti mesajul folosind cheia publică a rețelei. Aceasta este calculată și publicată dinainte. Este compusă din cheile publice a fiecărui server după cum urmează:

$$K = \prod_{j=1}^{n} K_j = \prod_{j=1}^{n} g^{d_j} = g^{\sum_{j=1}^{n} d_j}$$

$K_j = g^{d_j}$ este cheia publică a serverului j, d_j este cheia privată și g este o rădăcină modulo p unde p este un număr prim mare. Atât g cât și p sunt publice. Mesajul formulat de expeditor va arăta astfel:

$$c = (g^r, (A, M)K^r)$$

A este adresa destinatarului iar r este un număr aleator. Decriptarea mesajului va arăta astfel:

$$
\begin{aligned}
D_j(c) &= g^r g^{r_j}, (A, M)(K^r)(g^r)^{-d_j}(\prod_{a=1, a\neq j}^{n} g^{d_a})^{r_j} \\
&= g^{r+r_j}, (A, M)(g^{\sum_{a=1,a\neq j}^{n} d_a r} g^{d_j r})(g^{-d_j r})(\prod_{a=1, a\neq j}^{n} g^{d_a r_j}) \\
&= g^{r+r_j}, (A, M)(\prod_{a=1, a\neq j}^{n} g^{d_a r})(\prod_{a=1, a\neq j}^{n} g^{d_a r_j}) \\
&= g^{r+r_j}, (A, M) \prod_{a=1, a\neq j}^{n} g^{d_a (r+r_j)}
\end{aligned}
$$

Se observă faptul că $\prod_{a=1,a\neq j}^{n} g^{d_a}$ este produsul cheilor care nu au decriptat încă mesajul iar g^r este obținut din prima parte a mesajului c. Mesajul obținut în urma decriptării este trimis împreună cu altele către restul $n-1$ servere până se obține mesajul final $g^{r+\sum_{j=1}^{n} r_j}, (A, M)$.

Dimensiunea mesajului rămâne aceeași pe parcursul procesului de decriptare și expeditorul nu mai poate recunoaște mesajul după ce acesta trece de vreun server fără a ști cheia privată a serverului. În același timp nu trebuie realizată decât o criptare cu cheia publică comună a serverelor și ordinea decriptării nu contează deoarece fiecare server își poate folosi cheia privată în orice moment al procesului de decriptare. Observăm astfel că problemele rețelelor de decriptare enumerate mai sus dispar. Apar însă noi probleme.

- Nu pot fi adăugate informații de control sau adresa următorului server în conținutul mesajului deoarece acesta nu poate fi citit decât la finalul procesului de decriptare.
- Nu poate fi folosită o variantă hibridă pentru a beneficia de viteza criptografiei simetrice.
- La fel ca și la retelele de mixare bazate pe RSA, fiecare server trebuie să participe la decriptare pentru a obține mesajul final.

Apare o soluție pentru ultima problemă însă sub forma rețelelor de recriptare din [22] ce se bazează pe proprietatea de homomorfism a sistemului Elgamal [9]. Expeditorul criptează mesajul folosind cheia publică comună a rețelei la fel ca la varianta de decriptare. Serverele vor cripta acest mesaj din nou folosind un număr aleator propriu r_j. Vom obține următorul mesaj:

$$
\begin{aligned}
c' &= (g^r g^{r_j}, (A, M), K^r K^{r_j}) \\
&= (g^{r+r_j}, (A, M), K^{r+r_j})
\end{aligned}
$$

Odată ce toate serverele termină procesul de criptare vom obține un c'' de forma:

$$c'' = (g^{r+\sum_{j=1}^{n} r_j}, (A, M), K^{r+\sum_{j=1}^{n} r_j})$$

La final cele n servere decriptează împreună mesajele de la ieșire astfel:

$$\begin{aligned} D(c'') &= \frac{(A, M)K^{r+\sum_{j=1}^{n} r_j}}{(g^{r+\sum_{j=1}^{n} r_j})^d} \\ &= \frac{(A, M)K^{r+\sum_{j=1}^{n} r_j}}{K^{r+\sum_{j=1}^{n} r_j}} \\ &= A, M \end{aligned}$$

În ecuație, d este cheia privată partajată a serverelor unde $K = g^d$. Această cheie poate fi partajată folosind o schemă de partajare a secretelor precum cea a lui Shamir [28]. Rețeaua păstrează toate avantajele rețelei de decriptare bazată pe Elgamal [9] prezentată mai sus și în plus permite recuperarea mesajelor când cel puțin un număr t de servere din cele n sunt funcționale. În consecință rețeaua de recriptare este o versiune mult mai eficientă a rețelei de decriptare prezentată anterior.

Un alt aspect foarte important al rețelei de recriptare constă în posibilitatea de a folosi cheile unui destinatar atunci când toate mesajele sunt menite să ajungă la acesta. Această proprietate poate fi folosită în sistemele de votare unde există o singură autoritate care numără voturile. Votanții folosesc cheia publică a autorității pentru a cripta mesajul iar serverele recriptează acest mesaj și îl trimit la destinatarul unic. Acesta cunoaște d și obține $g^{\sum_{j=1}^{n} r_j}$ de la începutul mesajului fiind astfel singurul care poate să decripteze.

2.1.4 Mixnet de Recriptare Universal

Rețeaua de recriptare cu mai mult de un destinatar necesită două etape. O etapă de criptare și amestecare a mesajului de către cele n servere și una de decriptare comună. Acest ultim pas necesită o cheie privată comună și timp și putere computațională suplimentare pentru a finaliza procesul. Rețelele universale elimină acest ultim pas.

Pentru început un expeditor trimite doua mesaje criptate folosind Elgamal de forma:

$$(g^r, MK^r), (g^{r'}, K^{r'})$$

După care serverul j realizează următorul pas de recriptare:

$$(g^r g^{r'r_j}, MK^r K^{r'r_j}), (g^{r'r'_j}, K^{r'r'_j}) = (g^{r+r'r_j}, MK^{r+r'r_j}), (g^{r'r'_j}, K^{r'r'_j})$$

r și r' sunt numere aleatoare generate de expeditor iar r_j și r'_j sunt numere aleatoare generate de serverul j. În continuare următoarele $n-1$ servere repetă procesul. Datorită incluziunii celui de al doilea mesaj, serverele pot realiza procesul de recriptare fără a cunoaște cheia K . După repetate criptări exponentul primei părți a mesajului va fi de forma:

$$R = r + r'r_1 + r'r'_1r_2 + ... + r'r'_1r'_2...r_n$$

La final, ultimul server ce realizează recriptarea trimite către ieșire mesajul. Rețelele de recriptare universală au același dezavantaj ca și cele prezentate înainte și anume nu pot afla adresa destinatarului fără a decripta mesajul. Astfel, pentru a obține mesajul, un destinatar trebuie să caute exhaustiv în toate ieșirile rețelei de mixare un mesaj asociat cheii sale publice. Odată găsit acesta este decriptat de către destinatar folosind cheia privată d asociată propriului K astfel:

$$\frac{MK^R}{(g^R)^d} = \frac{MK^R}{K^R} = M$$

2.2 Vulnerabilități

În general pentru a avea o securitate cât mai bună nu este suficient să folosim doar un mecanism de securitate. Rețelele de mixare în forma prezentată mai sus sunt vulnerabile la diferite forme de atacuri atunci când sunt folosite în forma lor pură. În continuare vor fi prezentate unele vulnerabilități și posibile soluții care trebuie să fie luate în vedere în implementarea acestor rețele.

Un aspect foarte important al rețelelor de mixare constă în faptul că sunt necesari mai mulți utilizatori pentru a avea o anonimitate bună. Atunci când numărul lor este mic, șansele de a ghici corespondența dintre expeditor și destinatar cresc considerabil. Un sistem extrem de populat poate avea o securitate mai slabă date fiind șansele extrem de mici de a identifica relația menționată anterior. Numărul mare de utilizatori generează mult zgomot și astfel mesajele se pot pierde în mulțime.

Serverele nu trebuie să filtreze traficul la intrarea rețelei. Odată ce numai un grup specific (e.g. o companie are propriul server de mixare) poate folosi rețeaua utilitatea ei devine slabă. În același timp utilizatorii vor dori să folosească servere cu origini diferite pentru a împiedica colaborare acestora în a afla informații critice despre mesaj.

Există atacatori de categorii diferite ce trebuie luați în considerare. Avem adversari de tip:

- Extern sau Intern
- Pasiv sau Activ
- Local sau Global
- Static sau Adaptiv

Aceste categorii pot fi combinate în diferite moduri. Avem spre exemplu atacatori globali pasivi care pot observa tot traficul rețelelor, aceștia fiind adesea principalul adversar în multe lucrări de specialitate. Un alt exemplu este un atacator extern activ ce trimite o multitudine de mesaje cu scopul de a controla traficul rețelei.

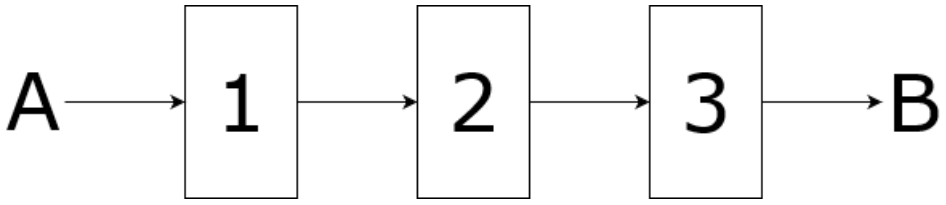


Figura 2.3: Drumul unui mesaj anonim de la Alice către Bob

În general se dorește folosirea unei combinații de servere. În cazul naiv în care există un singur server care amestecă traficul și îl decriptează, atunci când există o adresă asociată mesajului confidențialitatea legăturii dintre expeditor și destinatar dispare. Desigur, dacă toate serverele utilizate colaborează acest lucru nu poate fi evitat. De aceea este necesar cel puțin un server onest pentru păstrarea anonimității. În figura 2.3 odată ce toate serverele cunosc permutările realizate anonimitatea dispare în schimb dacă cel puțin un server (ex. serverul 2) păstrează permutarea secretă și nu cunoaște permutările celorlalte servere anonimitatea se păstrează.

Odată ce legătura dintre expeditor și destinatar este aflată dar mesajul este criptat un atacator poate afla conținutul mesajului prin mijloace legale sau prin intimidare. În același timp atacatorul poate pur și simplu să înregistreze mesajele trimise de către expeditor pentru a îi cere mai târziu acestuia să le descifreze. O soluție pentru această problema constă în folosirea de chei efemere (engl. ephemeral keys). Astfel conținutul mesajului nu poate fi recreat.

Ordinea în care mesajele ajung și părăsesc rețeaua este importantă. Odată ce un mesaj intră în rețea acesta poate fi urmărit de un atacator. Ordinea în care ajung mesajele poate divulga informații pe termen lung. Din acest motiv majoritatea rețelelor folosesc un mecanism de grupare (batching) și reordonare a mesajelor precum este ilustrat în 2.4.

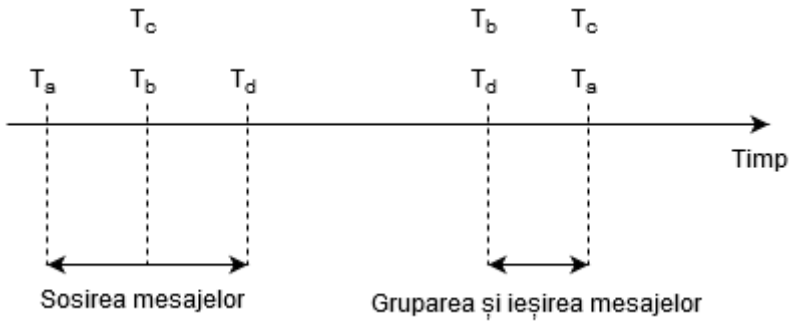


Figura 2.4: Ordinea temporală în care mesajele ajung și sunt grupate și trimise

Folosind batchuri de dimensiune l pentru mesaje un adversar poate încerca să ghicească care este mesajul urmărit cu probabilitate de $1/l$. Trebuie luată în considerare însă dimensiunea mesajelor după cum a fost precizat și în secțiunea anterioară. Dacă mesajele au dimensiuni diferite nu se mai poate vorbi de această probabilitate de $1/l$. Prin urmare trebuie adăugat un padding pentru a obține mesaje de aceeași dimensiune.

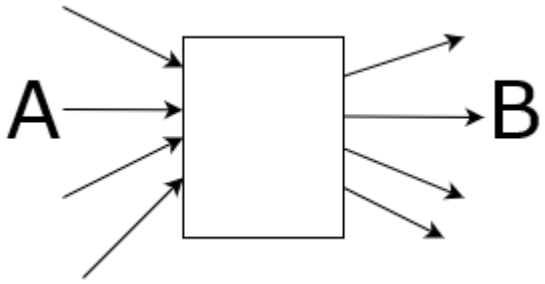


Figura 2.5: Atac de tip flood in care toate mesajele mai puțin cel a lui Alice sunt controlate de un adversar

Un alt atac cunoscut sub numele de replay attack constă în reutilizarea mesajelor anterioare pentru a obține informații despre conținut. Știm că algoritmii de criptare sunt determiniști. Prin urmare dacă un mesaj vechi este trimis din nou prin rețea în urmă decriptării se va ajunge la același rezultat. În consecință un atacator poate folosi această proprietate pentru a trimite un mesaj în repetate rânduri și a observa rezultatul acestei operații. În același timp aceste mesaje pot fi folosite de atacator pentru a își falsifica identitatea. Soluția pentru astfel de atacuri este păstrarea unui istoric de hashuri ale mesajelor ce trec prin rețea. Odată identificat un mesaj ce se repetă acesta este anulat. Desigur această soluție poate duce la probleme de stocare a mesajelor sau chiar mai grav Denial of Service. Un atacator poate trimite un număr mare de mesaje umplând baza de date cu posibile mesaje ce vor fi anulate în viitor. Din acest motiv trebuie ca baza de date să fie resetată împreună cu schimbarea cheii publice, lucru ce poate fi dificil. O alternativă este ca mesajele să aibă o perioadă de expirare aleatoare (pentru a evita corelarea dintre momentul când mesajul a ajuns în server și conținutul său) într-un interval de timp rezonabil. Un server răuvoitor poate însă să păstreze un mesaj pentru o scurtă perioada până când circumstanțele sunt favorabile.

În rețelele mai puțin populate un adversar poate trimite un număr mare de mesaje pentru a identifica mesajele ce nu îi aparțin. În 2.5 se poate vedea un astfel de atac. Toate mesajele din intrare și ieșire mai puțin cel a lui Alice sunt cunoscute de către adversar și astfel anonimitatea dispare. O soluție pentru astfel de atacuri constă în crearea unor pooluri de mesaje. Un pool este un loc unde mesajele destinate rețelei de mixare sunt stocate înainte de a fi trimise. Acest lucru îngreunează misiunea unui răufăcător de a identifica și izola un mesaj. Un număr minim de mesaje trebuie astfel menținut în acest pool mesajele fiind alese aleator și părăsind astfel poolul în intervale de timp regulate. Desigur acest lucru nu rezolvă problema in situațiile in care un singur utilizator folosește rețeaua. Din acest motiv utilizatorul sau alte entități pot trimite mesaje false ce au rol de a crea confuzie. Aceste mesaje pot sau nu pot avea o destinație clară. Scopul lor este doar de a crea trafic in rețea.

Drumul folosit trebuie să fie același pentru toți utilizatorii. În cazul în care doi utilizatori folosesc drumuri diferite mesajele pot fi diferențiate cu ușurință de către un observator pasiv. Trebuie să existe astfel servere ce se ocupă de crearea acestor

drumuri pe care utilizatorii să le poată utiliza. Aceste servere trebuie să dețină mecanisme de sincronizare prin care să fie de comun acord în privința structurii rețelei. În același timp un astfel de server poate avea intenții rele și să trimită o structură ce poate identifica un utilizator. Drept soluție utilizatorii trebuie să folosească o grupare de servere ce trebuie să cadă de comun acord asupra structurii. Un server răuvoitor poate să afecteze acest proces oferind drumuri greșite, refuzând să cadă de acord cu ceilalți participanți. În consecință se poate folosi un prag minim de servere ce trebuie să cadă de comun acord pentru a continua procesul.

În [8] sunt menționate servere ce nu participă la procesul de permutare ci păstrează statistici despre cele care mixează. Astfel de servere verifică disponibilitatea rețelei. În lucrare mai sunt prezentate și mecanisme de verificare și pedepsire a serverelor răuvoitoare. Aceste tipuri de mecanisme de reputație au însă dezavantaje. Ele pot fi influențate de către adversari ce doresc să își îmbunătățească reputația ca apoi să profite de aceasta. Slăbiciunea majoră constă astfel în imposibilitatea de a prezice faptul că un server care pare onest va face un lucru rău.

Un alt atac important în rețelele de decriptare este atacul de tip tagging . Acesta poate fi aplicat mesajelor ce conțin date de control a căror poziție este cunoscută. În acest atac două servere (în general serverul de intrare și cel de ieșire) colaborează pentru a afla destinatarul și/sau conținutul mesajului. Primul server schimbă un header cu informații de control stricându-l pentru ca la final ultimul server de mixare să poată identifica mesajul marcat in acest mod. O soluție pentru un astfel de atac este crearea unui hash ce acoperă mesajul la fiecare strat. În acest mod dacă un server observă o modificare mesajul este anulat.

O problemă majoră nerezolvată ce afectează toate sistemele de anonimitate este atacul de intersecție pe termen lung. În general acest tip de rețele nu este folosit des de anumiți utilizatori așa ca un observator pasiv poate păstra traficul rețelei și să afle prin eliminare și corelare cine unde și cum trimite mesaje. Tot ce trebuie să facă un atacator este să intersecteze mulțimile persoanelor care comunicau în diferite perioade de timp. Astfel dacă două persoane comunică în repetate rânduri acest lucru poate fi observat pe termen lung.

2.3 Verificabilitate

Atunci când ne uităm la cine poate verifica operațiile realizate în cadrul rețelelor avem trei categorii de verificabilitate, rețele verificabile de către expeditor, server sau universal verificabilă.

2.3.1 Verificabilitatea Realizata de Expeditor

Expeditorul introduce mecanisme pentru verificarea mesajului rezultat la ieșirea rețelei. Acesta este cel mai simplu tip de verificabilitate fiind apărut încă din lucrarea lui Chaum [5]. Acest lucru ajută însă nu garantează faptul că restul mesajelor au fost transmise corect. Mai mult de atât, dacă un expeditor nu verifică dacă mesajul a fost transmis atunci acesta poate fi șters fără ca acest lucru să fie detactat.

Un exemplu de astfel de verificare constă in folosirea de semnături digitale. Expeditorul trimite un mesaj care arată astfel:

$$K(K_e, D_{K_e}(m, 0^w), r)$$

În mesajul de mai sus $D_{K_e}(m, 0^w)$ poate fi creat doar de către expeditor folosind cheia sa privată. Acest lucru asigură faptul că nimeni nu poate falsifica mesajul. În același timp 0^w demonstrează faptul că nimeni nu a modificat mesajul. În cazul în care 0^w este modificat înseamnă că mesajul a fost corupt. Detecția coruperii unui mesaj la ieșire este însă un dezavantaj deoarece nu poate fi identificat serverul care a dus la acest rezultat.

2.3.2 Verificabilitatea Realizata de Rețea

Pentru o rețea ce nu necesită ca verificarea să fie realizată de persoane din exterior ci se bazează pe încredere se folosesc servere care preiau această responsabilitate. Verificabilitatea realizată de servere se folosește de diferite tehnici criptografice pentru a identifica eventualele probleme.

Verificabilitate in retele de recriptare

Avem astfel tehnici precum cele folosite de Jakobsson in lucrările sale [15] și [14]. Cele două se folosesc de algoritmul lui Elgamal pentru criptarea traficului. In lucrările sale Jakobsson propune o robustețe prin repetiție realizată folosind următoarele protocoale:

1. **Se introduc copii ale intrărilor anterioare în rețea**. După realizarea procesului de mixare a unui set de mesaje se introduce din nou o copie a acestui set pe care se realizează o nouă mixare. Acest lucru se poate repeta de x ori pentru a maximiza șansele de detectare a unei probleme. Dacă o copie devine coruptă, șansele ca acest lucru să poată fi repetat scad considerabil datorită permutarilor și operațiilor aleatorii folosite de servere. Această tehnică este ideală atunci când primul server al rețelei este de încredere dat fiind faptul că acesta este singurul care poate identifica copiile.

2. **Se repetă procesul de mixare**. Protocolul anterior este slab atunci când primul server este corupt. Pentru a identifica astfel probleme ce provin din primul server, se repetă procesul de mixare folosind mesajele obținute la finalul rețelei ca nou set de intrare. Mai exact odată ce mesajele ajung la ultimul server din rețea acesta le trimite din nou la primul server pentru a repeta acest proces. Dacă cel puțin un server este onest, primul server nu va ști ordinea mesajelor și prin urmare nu va putea modifica mesajul dorit. Se poate astfel detecta dacă primul server este compromis.

3. **Sunt dezvăluite secretele folosite de servere și se compară rezultatele**. Acest lucru este necesar pentru a descoperi serverul corupt. Un mesaj corupt de primul server din rețea va arăta astfel:

$$(g^{r+r_1}, K^{r+r_1}mm') = (g^{r+r_1}, g^{(r+r_1)d}mm')$$

În algoritmul lui Elgamal [9] g este generator pentru grupul $\mathbb{Z}_p^*$, grup din care face parte și m'. Prin consecință există x astfel încât $m' = g^x$ și mesajul devine:

$$(g^{r+r_1}, g^{(r+r_1)d}g^x m) = (g^{r+r_1}, g^{(r+r_1)d+x}m)$$

Odată ce primul server dezvăluie numărul său aleator secret r_1 se poate detecta adăugarea lui g^x comparând mesajul obținut de server cu mesajul corect. Apare însă o problema în urma folosirii acestei tehnici și anume ruperea anonimității. În consecință procedura trebuie folosită doar în combinație cu protocolul de repetare a mixării. După terminarea celei de a două mixări serverele fac publice secretele folosite la prima mixare pentru a verifica dacă aceasta a fost realizată corect. Serverele calculează din nou rezultatele primei mixări după care sortează aceste rezultate și rezultatele celei de a doua mixări. Tot acest proces este executat în același timp și pentru copiile mesajelor. La final se compară listele de rezultate obținute din mesajele originale și copiile acestora și dacă nu apar anomalii atunci procesul a fost executat corect.

4. **Se depistează serverele compromise**. În cazul în care se găsește o problemă în timpul verificării se pornește depistarea serverelor compromise urmând drumul spre intrarea rețelei pornind de la ieșire. Pe rând fiecare participant își face publice secretele pentru ca ceilalți participanți să verifice rezultatele acestuia. Odată găsite, serverele corupte sunt înlocuite și procesul o ia de la început. Este important că mesajele finale să nu fie decriptate până la finalul verificării pentru a se păstra anonimitatea.

5. **Generarea de dovezi și verificarea lor**. Un alt protocol constă în a se verifica dacă au fost adăugate sau șterse mesaje pe parcursul procesului. Pentru acest lucru se folosesc dovezi ale corectidunii procesului numite ZKP (Zero Knowledge Proof). Serverele calculează produsul mesajelor de la intrare și încearcă să dovedească faptul că acesta este egal cu produsul obținut la ieșire. În cazul în care se observă greșeli în dovada vreunui server acesta este eliminat și se repornește procesul.

6. **Se verifică validitatea serverelor**. Dovezile de tipul ZKP pot fi folosite pentru a verifica veridicitatea cheilor private deținute de fiecare server în parte. Dacă se dovedește că un server nu cunoaște o componentă a cheii private a rețelei de recriptare acesta este exclus din rețea .

Protocoalele prezentate mai sus sunt folosite în [14] și [15] însă cu toate acestea rețelele prezentate în aceste lucrări nu sunt sigure. Integritatea rețelei prezentate în [14] poate fi compromisă folosind proprietatea de homomorfism al sistemului Elgamal precum a fost arătat în [7]. Homomorfismul constă în obținerea unui mesaj de forma

$(g^{r_1+r_2}, m_1 m_2 K^{r_1+r_2})$ folosind $c_1 = (g^{r_1}, m_1 K^{r_1})$ și $c_2 = (g^{r_2}, m_2 K^{r_2})$. Se observă că acest lucru se poate realiza cu ușurință înmulțind cele două cantități.

$$\begin{aligned} c_1 c_2 &= (g^{r_1} g^{r_2}, m_1 m_2 K^{r_1} K^{r_2})) \\ &= (g^{r_1+r_2}, m_1 m_2 K^{r_1+r_2}) \end{aligned}$$

Astfel un server malițios poate modifica toate mesajele introduse în rețea fără a fi detectat. În [15] este eliminată această problemă prin folosirea a două mesaje false generate în comun de către serverele rețelei. Din păcate în [19] faptul că odată ce primul server știe care sunt mesajele false integritatea este compromisă .

Un alt mod de a verifica corectitudinea operațiilor se bazează pe împărțirea serverelor în blocuri de încredere. În [7] se construiește o astfel de rețea care se bazează pe un număr mare de servere în schimbul dovezilor de tipul ZK. În această rețea se consideră că există maxim $t-1$ servere ce nu sunt oneste. Serverele sunt aranjate în blocuri de dimensiune t unde fiecare server este verificat de celelalte $t-1$ servere din același bloc. Sunt necesare astfel t^2 servere spre deosebire de cele t din rețelele prezentate anterior. Pentru verificare fiecare server trimite celor $t-1$ verificatori din bloc componentele secrete folosite în operația de mixare. Cei $t-1$ vecini vor verifica dacă procedura a fost respectată iar în caz contrar preiau intrarea serverului compromis. Acest lucru nu necesită repornirea mixării precum în protocoalele prezentate anterior. În același timp dacă un participant din fiecare bloc este corupt și aceștia colaborează anonimitatea este anulată dat fiind că aceștia cunosc secretele vecinilor. În consecință t trebuie ales cu atenție. Acest tip de verificare folosind blocuri a fost folosită și în [21] după cum va fi prezentat în continuare.

Verificabilitatea in rețele hibride

Toate tehnicile prezentate anterior sunt valabile în rețele bazate pe Elgamal [9]. În continuare vor fi prezentate protocoale valabile pentru rețele de mixare hibride. Două astfel de rețele sunt [17] care se folosește de tehnici de detectare a erorilor și [21] ce folosește o structura bazată pe blocuri de servere. Rețeaua prezentată în [21] folosește în rețelele de decriptare hibride, tehnicile prezentate în [7]. În același timp [17] îmbunătățește abordarea din [21] folosind dovezi ZK eficiente. În [17] este eliminată structura de tip bloc evitând astfel și dezvăluirea secretelor folosite de acestea. În continuare vor fi prezentate protocoalele folosite:

1. **MAC**. MAC sau coduri de autentificare a mesajelor sunt folosite pentru verificarea integrității mesajelor. Ele se notează ca $MAC_j(m) = \mathcal{H}_{k'_j}(m)$ unde j este numărul serverului și k'_j este cheia folosită pentru generarea codului MAC. Aceste coduri previn coruperea mesajelor de către celelalte servere dar necesită o altă cheie publică pentru distribuirea cheii folosite de MAC. Pentru a corupe un mesaj, un atacator trebuie să schimbe și MAC-ul acestuia lucru ce este dificil. Este importantă protecția cheilor folosite pentru generarea MAC acestea fiind obținute doar prin decriptare folosind o cheie privată adecvată.

2. **Folosirea cheii publice pentru realizarea criptării simetrice**. Un server se folosește de cheia sa publică pentru a obține cheia simetrică necesară pentru decriptarea mesajului. Astfel dacă avem o cheie publică $K_j = g^{d_j}$ unde j reprezintă serverul, acesta va primi de la vecinul său anterior $j-1$ un generator de forma:

$$gen_{j-1} = g^{r \sum_{i=1}^{j-1} d_i}$$

Si un mesaj de forma:

$$K_j(A_{j+1}, K_{j+1}(...)), \mathcal{H}_{k'_j}(k_j(A_{j+1}, (E_{K_{j+1}...})), 0^w)$$

Unde $gen_0 = g^r$ este inclus de expeditor iar r este un număr aleator. Se observă că acest protocol se folosește de MAC pentru păstrarea integrității. Cheile simetrice k_j și k'_j se obțin din gen_{j-1} și încă două chei private având astfel un total de trei chei publice folosite de expeditor.

3. **Simularea unui server compromis**. Pe lângă folosirea MAC se folosesc dovezi ZK pentru verificarea integrității generatorului. Fiecare server $j-1$ trebuie să dovedească astfel faptul că deține cheia privată folosită în gen_{j-1} . Treaba serverului j este de a verifica dacă această dovadă este corectă și dacă mesajul transmis de către predecesorul său este corect. În cazul în care serverul $j-1$ eșuează în a își dovedi onestitatea există trei posibilități:

   - **Serverul $j-1$ este compromis**. La fel ca în [7] se folosește un prag de încredere t ce reprezintă numărul minim de servere ce trebuie compromise pentru a anula anonimitatea rețelei. Astfel dacă $j-1$ este compromis se vor folosi t servere pentru a genera cheile private ale lui $j-1$. Cele t servere alese vor verifica corectitudinea operațiilor realizate de $j-1$ și, în cazul în care se dovedește că acestea sunt greșite, vor simula mixarea realizată de acesta.
   - **Serverul j este compromis**. În cazul în care $j-1$ nu este compromis se verifică în același mod și j. Dacă se dovedește faptul că acesta este corupt se vor urma aceeași pași precum în cazul precedent.
   - **Datele de intrare pentru $j-1$ sunt greșite** . În cazul în care nici serverul j și nici $j-1$ nu sunt compromise se merge înapoi și se verifică pe rând serverele precedente până se găsește responsabilul care este eliminat. Operația de mixare se reia din punctul în care problema a fost rezolvată.

4. **Simularea ultimului server**. O problemă care apare în protocolul prezentat anterior constă în lipsa verificării ultimului server. În cazul în care acesta nu este verificat el poate schimba cu ușurință rezultatul final. Drept soluție toate cele n servere vor colabora pentru a simula un server $n+1$ care să poată verifica și decripta ultimul mesaj.

O vulnerabilitate observată în combinația de protocoale prezentate mai sus apare atunci când un atacator trimite n mesaje greșite. Acesta poate greși intenționat mesajul pentru a forța fiecare server să își dezvăluie cheile private. Prin colaborarea unui astfel de atacator cu un server compromis ce cunoaște cheile publice, anonimitatea poate fi încălcată.

O altă problemă importantă observată în toate protocoalele prezentate până în acest moment este imposibilitatea detectării operațiunii greșite a rețelei în anumite condiții. În [17] dacă spre exemplu toate serverele sunt corupte acest lucru nu poate fi detectat și prin urmare integritatea rețelei este compromisă fără ca expeditorii sau destinatarii să știe. La fel și în [7] unde dacă există t servere compromise.

2.3.3 Verificabilitate universala

Verificabilitatea universală așa cum apare în [27] constă în posibilitatea verificării procesului de mixare de către orice utilizator al rețelei sau observatori din afara rețelei. Prin urmare rețelele de mixare nu trebuie să aibă posibilitatea de a crea rezultate greșite sau corupte fără ca acest lucru să nu fie detectat. Pentru acest lucru se folosesc în general dovezi ZK ce garantează corectitudinea în detrimentul vitezei de operare a rețelei.

Un mod de a obține astfel de verificabilitate este prin utilizarea protocolului 3 prezentat în cadrul verificabilitatii în rețele hibride. Diferența constă în introducerea unui participant extern ce poate verifica corectitudinea rezultatelor. Astfel, oricine poate lua acest rol pentru a vedea dacă mesajele sunt corecte și anonimitatea nu s-a pierdut.

O primă abordare constă în protocolul cut-and-choose prezentat în [20] și mai apoi în [1].

Pentru o probabilitate cat mai mare de a detecta un server compromis protocolul trebuie repetat de mai multe ori. Un exemplu bun de numar de repetari este $\sigma = -log_2\epsilon$ unde $\epsilon \leq 2^{-80}$. Un dezavantaj constă in ineficiența acestui protocol când vine vorba de puterea de calcul și de comunicare.

O altă clasă de rețele verificabile universal apare în urma folosirii protocolului de verificare parțială aleatorie propus prima oară în [18]. Această tehnică nu necesită dovezi de tipul ZK și poate fi folosită în orice tip de rețea. În continuare va fi prezentat modul de funcționare a protocolului:

1. Se formează perechi de servere fiind necesar un număr par de participanți. Aceste perechi trebuie să conțină câte două servere consecutive unde ieșirea serverului j este intrarea serverului $j + 1$.

2. Se alege o submulțime din mulțimea mesajelor aparținând ieșirii serverului j. Serverul j trebuie să facă publice intrările care corespund submulțimii alese și componentele secrete (exponenții aleatori, cheile private etc.) folosite in timpul permutării și transformării mesajelor alese. Astfel o parte din permutarea realizată de serverul j nu mai este secretă.

3. La fel ca la pasul anterior se alege aleator o submulțime de intrări ale serverului $j+1$. Mesajele acestei submulțimi nu trebuie să aibă nicio legătură cu submulțimea aleasă de serverul j. Și din nou serverul $j+1$ face publice mesajele din ieșirea sa ce corespund submulțimii de intrări alese.

4. Pașii 2 și 3 se repetă pentru fiecare pereche de servere. Cu toate că o parte din permutările realizate de servere vor fi compromise acest mod de verificare probabilist este rapid și, datorită alegerii aleatoare de submulțimi, greu de compromis.

O alternativă ce nu se bazează pe dezvăluirea secretelor folosite este dată în [3]. Tehnica de "verificare optimistă" folosește dovezi ZK pentru a demonstra corectitudinea în operare. Fiecare server produce două astfel de dovezi. Prima arată că mesajele primite la intrare se găsesc în ieșirea sa, demonstrând că nu au fost șterse sau adăugate mesaje de către server. Pentru demonstrație se folosește produsul mesajului de intrare și produsul mesajelor obținute la ieșirea serverului. A două dovadă se bazează pe o submulțime a mesajelor obținute la ieșirea serverului și încearcă să demonstreze că acestea provin din valorile introduse la intrarea sa. Diferența în cea de a două dovadă și protocolul de verificare parțială aleatorie stă în modul în care relația este verificată. În cazul protocolului anterior serverul își divulgă secretele folosite în timp ce în protocolul actual nu se întâmplă acest lucru ci se folosește o dovadă ZK eficientă. În [11], pe lângă acest protocol de verificare optimistă sunt folosite hashuri pentru verificarea integrității și o tehnică numită dublă criptare folosită pentru păstrarea anonimității chiar și atunci când rețeaua este compromisă. Pentru acest lucru fiecare mesaj folosește trei criptări Elgamal [9] astfel:

$$K(g^R, r), K(M, r'), K(\mathcal{H}(g^R, M), r'')$$

Unde $M = mg^{dR}$ și $\mathcal{H} : 0,1^* \rightarrow G$ este o funcție hash one-way. După recriptare fiecare server generează o dovadă precum în [3] bazată pe protocolul Chaum-Pederson.

Capitolul 3

Aplicații în protocoale de securitate

3.1 Tipuri de aplicații

Rețelele de mixare sunt un mecanism de păstrare a anonimității la care participă mai mulți utilizatori. Acestea sunt folosite astfel in aplicații cum ar fi votul electronic, e-mailuri anonime și licitație electronică unde anonimitatea este un punct cheie. Este important de observat faptul că odată cu creșterea securității apare și o creștere a complexității. Din acest motiv există multe lucrări teoretice în acest domeniu, care nu pot fi folosite in practică datorită costului mare a operațiilor criptografice și de comunicare. Prin urmare nu vom putea avea o securitate perfectă în practică. În continuare vor fi prezentate două tipuri de aplicații, licitație și vot electronic, urmând ca un tip de aplicație să fie mai apoi implementat.

3.2 Vot electronic

Procesul de vot este în ziua de astăzi o parte foarte importantă a societății, stând la baza democrației și a drepturilor omului. Această unealtă este prin urmare o țintă de mare interes din punct de vedere al securității. Se poate ajunge în cazuri în care decizii critice pentru o țară pot fi afectate de către adversari puternici. Datorită acestui motiv schimbarea acestui proces este un lucru foarte sensibil pentru multe persoane și poate genera reacții puternice.

Votul electronic poate face procesul de votare mai ușor, rapid și eficient. În același timp previne irosirea hârtiei și apariția greșelilor în timpul votului datorită restricțiilor ce pot fi implementate în program. Utilizarea internetului pentru facilitarea procesului de vot poate fi un lucru avantajos pentru votanți și pentru organizatorii votului din cauza disponibilității și accesibilității sistemului pe parcursul procesului de votare. Vor dispărea cazuri în care, date fiind cozile lungi și timpul insuficient, unele persoane nu își vor putea exercita dreptul de vot. Mai mult, procentul celor care vor participa va crește considerabil dată fiind ușurința procesului.

O foarte importantă cerință a procesului de votare este confidențialitatea alegerii. Aceasta împiedică extorcarea, cumpărarea de voturi sau persecuția alegătorilor, lucruri ce pot afecta deplin modul de viață și siguranța lor. În consecință, o principală motivație pentru cei care se împotrivesc votului electronic este prezervarea confidențialității. O altă posibilă problemă este lipsa de încredere în privința credibilității voturilor. Această teamă vine din dificultatea prevenirii unei persoane de a vota de mai multe ori. Se poate observa că cele două probleme sunt strâns legate între ele dat fiind faptul că inexistența unei legături dintre vot și votant poate duce la lipsa unui mod de restricționare a numărului de voturi a unei persoane.

Pe lângă securitatea informației trebuie luată în considerare și posibilitatea de cumpărare a voturilor sau de influențare a votului de către surse rău intenționate. Dovezile de vot trebuie să nu divulge prin urmare alegerea votantului pentru a împiedica aceste scenarii nedorite. O astfel de soluție este oferită în [27] în care Sako et al. propun o schemă de vot fără adeverință a votului dar cu posibilitatea de verificare a corectitudinii rezultatului procesului de vot. În cazul în care alegătorii pot fi supravegheați de personaje rău intenționate în timpul votului trebuie oferită și posibilitatea de schimbare a opțiunii alese în viitor.

Pentru a obține protocoale eficiente din punct de vedere al calculelor efectuate și a dimensiunii informațiilor transmise este necesară o relaxare a cerințelor sistemului. Se folosește deseori un grad minim de încredere într-o parte din participanții protocolului pentru a evita dovezi costisitoare și pași în plus. Acest lucru nu este rău dat fiind faptul că există astfel de cerințe de încredere și în alegerile fizice, mai exact alegătorii au încredere ca voturile sunt procesate, transportate și numărate așa cum trebuie de către organizatorii procesului de votare. Ar putea fi argumentat faptul ca în unele cazuri protocoalele de vot electronic sunt mai sigure decât cele actuale.

3.2.1 Specificații

În continuare vor fi prezentate o serie de proprietăți întâlnite în protocoale de vot electronic. Este important de menționat faptul ca nu toate aceste caracteristici sunt necesare pentru obținerea unui sistem sigur, o parte din acestea fiind opționale.

- **Robustețe** - Protocolul de vot trebuie sa fie rezistent la erori sau încercări de perturbare apărute în orice moment.

  Această proprietate este întâlnită și în rețelele de mixare și este necesară în practică datorită evenimentelor neașteptate ce pot apărea. Un protocol trebuie astfel să reziste atunci când spre exemplu un server cedează sau are loc un atac de tip DoS. Robustețea este garantată de verificabilitate și asigură corectitudinea procesului de votare.

- **Corectitudine** - Nimic nu trebuie să afecteze procesul de votare.

  Pentru a avea un rezultat ce reprezintă alegerea tuturor alegătorilor trebuie ca toți participanții legitimi să poată vota și votul lor sa fie luat în calcul. Procesul

este corect pentru alegători, oferindu-le dreptul de a își exprima alegerea. Este corect și pentru organizatorii alegerilor sau cei ce sunt aleși prin numărarea corectă a tuturor voturilor.

- **Verificabilitate** - Procesul de vot poate fi verificat pentru a dovedi corectitudinea sa.

  În protocoalele de vot există două tipuri de verificabilitate, individuală și publică. Verificabilitatea individuală permite unui alegător să dovedească faptul că votul său a fost numărat cum trebuie. Aceasta este însă mai slabă decât al doilea tip. Verificabilitatea publică permite oricărei entități să verifice funcționarea corectă a protocolului de vot. Responsabilitatea este luată de la alegători și se creează o încredere mult mai mare în sistem. Pot apărea și un amestec între cele două, spre exemplu alegătorul dovedește faptul că votul său a fost înregistrat de sistem și rezultatul final poate fi verificat de oricine.

- **Anonimitate** - Voturile sunt secrete.

  Fie doi alegători A_1 și A_2 și două voturi v_1 și v_2. Nimeni nu trebuie să fie capabil să spună cu o probabilitate mai mare de 50% dacă A_1 a ales v_1 și A_2 a ales v_2 sau A_1 a ales v_2 și A_2 a ales v_1. Altfel spus, relația dintre vot și alegător trebuie să nu fie cunoscută. Această proprietate este folosită pentru siguranța alegătorilor. Pentru asigurarea anonimității se folosesc rețelele de mixare.

- **Integritate** - Rezultatul nu poate fi manipulat sau modificat.

  Pentru asigurarea integrității trebuie respectate trei cerințe:

  Votul trebuie sa fie înregistrat în modul dorit de alegător. Nu trebuie să existe cazuri în care, spre exemplu, sistemul alege varianta opusă față de cea selectată.

  Votul trebuie să fie folosit în aceeași formă în care a fost înregistrat. Sistemul nu poate modifica astfel alegerea făcută. Votul poate lua altă formă în urma operațiilor de criptare însă conținutul acestuia nu trebuie să fie modificat.

  Voturile trebuie numărate în concordanță cu rezultatul procesării lor. Rezultatul va corespunde astfel voturilor.

- **Receipt-Free** - Un alegător nu trebuie să poată dovedi ce a votat.

  Se poate dovedi în schimb faptul că s-a votat sau că votul a fost înregistrat cum trebuie. Această proprietate este folosită împotriva persoanelor care cumpără voturi sau care forțează sau constrâng alți alegători să voteze într-un anumit mod. Este esențială pentru păstrarea unei anonimități totale.

- **Vot Unic** - Orice alegător poate vota o singură dată.

  Una dintre proprietățile obligatorii pentru orice sistem de vot electronic. Folosind mecanisme de verificare sau un administrator, un alegător nu trebuie sa aibă posibilitatea de a vota de două sau mai multe ori. Această proprietate asigură alegeri corecte.

- **Drept de Vot** - Numai persoanele autorizate pot vota.

  Pentru a asigura acest lucru trebuie să existe o etapă de înscriere înainte de cea de votare. În urma înscrierii alegătorii vor avea un secret sau un identificator pe care îl vor folosi în timpul votului.

- **Vote-and-go** - Protocolul trebuie să fie intuitiv și ușor de folosit pentru alegători.

  Nu trebuie ca alegătorii să fie profesioniști sau să aibă cunoștințe avansate pentru a vota. În același timp alegătorul nu trebuie să participe în mod activ la tot procesul de vot. Ei trebuie să își exprime votul într-un mod care să nu impună dificultăți sau confuzie după care participarea lor se termină. Poate exista posibilitatea de a verifica procesul de vot dar acest lucru nu trebuie sa fie obligatoriu pentru alegători pentru ca procesul să ajungă la un rezultat. Acestă proprietate este esențială pentru ca protocolul să fie acceptat de o masă largă de utilizatori.

- **Flexibilitate** - Protocolul poate fi folosit pentru diferite reguli de vot.

  Sunt cazuri în care votul poate fi da/nu cum este în cazul referendumului sau cazuri în care există mai multe opțiuni pe care alegătorii le ordonează în funcție de preferință. Un protocol flexibil trebuie să poată fi folosit indiferent de structura votului.

- **Cabină de Vot** - Cabinele de vot sunt spații special amenajate în care alegătorul nu este observat în niciun fel.

  Acestea sunt necesare în anumite protocoale de vot și oferă alegătorilor siguranță împotriva persoanelor ce doresc să le influențeze votul. Împreună cu proprietatea de receipt-free ele descurajează, spre exemplu, cumpărătorii de voturi care nu au cum să știe dacă cei plătiți își onorează promisiunile. Aceste cabine necesită prezența fizică a alegătorului și prin urmare nu se folosesc la votul prin internet.

- **Bulletin-board** - Loc unde se afișează informații publice.

  Acest panou informativ este folosit pentru verificarea procesului de votare de către participanți și entități din exterior sau pentru comunicarea în cadrul protocolului. Informațiile de pe acest panou nu pot fi șterse și numai entitățile autorizate pot adăuga informații noi. Oferă transparență și încredere și poate fi folosit pentru contestarea eventualelor probleme.

3.2.2 Structura

Votul electronic este în general împărțit în etape bine definite. Acestea au un interval de timp bine definit și nu se suprapun. Ele sunt interdependente și se execută secvențial. Nu toate sistemele de vot electronic se aseamănă, prin urmare unele pot avea etape în plus iar altele pot funcționa fără anumite etape.

1. Pregătire

   În această etapă se inițializează parametrii necesari procesului de votare. Se stabilesc candidații și autoritățile responsabile pentru organizarea și procesarea voturilor. Regulile sunt de asemenea stabilite. Sunt făcute publice condițiile pe care trebuie să le respecte un alegător pentru a avea drept de vot cât și cum arată un buletin de vot valid. Aici se stabilesc și mare parte din cheile criptografice ce vor fi folosite în timpul votului.

2. Înregistrare

   Această etapă este de obicei în afara atribuțiilor sistemului de vot electronic. Aici cei care doresc să participe la votul electronic trebuie să se înscrie pentru a primi un identificator unic ce poate fi folosit pentru a dovedi faptul că votul este legitim și unic.

3. Votare și amestecare

   În această etapă are loc votul propriu-zis. Alegătorii cu drept de vot (înregistrați dacă este cazul) își exprimă opțiunile. Celor ce nu sunt înregistrați sau nu au drept de vot le este interzis accesul. Votul poate fi exprimat în cadrul unei cabine de vot unde este cazul. După trimiterea votului el devine anonim. Alegătorul poate primi în anumite cazuri o adeverință a votului cu care poate semnala probleme dar nu poate demonstra ce a votat.

   Voturile sunt amestecate folosind rețele de mixare și ajung la autoritățile responsabile de numărarea lor. Acesta este un pas esențial pentru anonimitate. Pentru a obține un sistem cu un grad de încredere ridicat dar și pentru a evita eventualele greșeli sunt necesare mecanisme de verificare a rezultatului rețelei.

4. Deschidere

   Odată ce voturile au fost adunate și perioada valabilă pentru exercitarea dreptului de vot s-a sfârșit, voturile sunt decriptate. Acestea sunt făcute publice și se verifica corectitudinea lor. Voturile valide sunt păstrate pentru a fi numărate.

   Este important ca această etapă să aibă loc după ce perioada dedicată alegerii s-a terminat pentru a evita cazuri în care rezultate parțiale devin publice. În aceste situații opinia publică se poate schimba radical și variantele cu cele mai puține voturi nu vor mai fi alese.

5. Numărare

   Entitățile responsabile se ocupă de numărarea și publicarea rezultatelor. În unele cazuri numărătoarea se întâmplă în timpul amestecării, ele publicând în schimb o dovadă de echivalență a rezultatului. După această etapă se poate realiza încă o verificare mai amănunțită a rezultatelor și o renumărare. Renumărarea poate dura mai mult timp și din acest motiv se întâmplă după ce primele rezultate sunt afișate. Aceasta este de asemenea importantă atunci când rezultatele sunt apropiate.

3.2.3 Modele de scheme de votare

Schemele de vot flexibile pot fi ineficiente din multe puncte de vedere. În general alegerile au un set prestabilit de reguli și se doresc costuri reduse și eficiență cât mai bună. Din aceste motive există diferite protocoale de vot folosite special pentru regulile dorite ce nu sunt complexe în implementare. În același timp există alegeri de dimensiuni diferite. Un sistem ineficient pe scală largă poate fi foarte bun pentru alegeri realizate în grupuri mici cum ar fi de exemplu deciderea unui lider de echipă. În același timp sisteme pentru alegeri de proporții mari ce se bazează pe un număr ridicat de utilizatori pentru securitate nu vor putea fi folosite la alegeri mici. Acestea riscă trecerea unui procent mare de voturi greșite sau corupte care pot fi ignorate la alegeri naționale datorită procentului relativ nesemnificativ. Din aceste motive este mai bine să folosim diferite sisteme de votare.

În lucrarea [10] este prezentat un sistem sigur și privat pentru vot electronic care însă are un dezavantaj. Acest protocol susține că poate fi folosit pentru alegeri pe scală largă ce nu se bazează pe încrederea în autorități pentru procesarea corectă a voturilor. Responsabilitatea este însă mutată la alegători pentru verificare și pentru realizarea etapei de votare, pregătire/înregistrare și deschidere.

Protocolul este alcătuit din alegători, un administrator și o autoritate care se ocupă de numărarea voturilor. Pentru evitarea unui singur punct de eșec se pot folosi mai mulți administratori. În același timp autoritatea ce se ocupă de numărat poate fi înlocuită de un bulletin board. Sunt necesare canale anonime de comunicare acestea fiind reprezentate de rețelele de mixare. Se folosesc și diferite mecanisme criptografice pentru verificarea și garantarea corectitudinii.

Alegătorii trebuie să dețină chei criptografice și cunoștințe despre folosirea lor într-un mod sigur. În acest caz trebuie să existe entități care să se ocupe de îndrumarea alegătorilor astfel încât instrumentele criptografice să nu fie folosite greșit.

După ce alegătorii și-au pregătit votul acesta este trimis către un administrator care se ocupă de autorizarea acestora. Administratorii trebuie să dețină liste de identități pentru a verifica dacă identitatea asumată de un vot a fost deja folosită sau este validă. Pe scală largă acești administratori vor fi solicitați. Administratorii nu vor putea verifica dacă voturile sunt valide din punct de vedere a conținutului. Aceștia pot doar să verifice faptul că votul este asociat semnăturii trimise de alegător. Dacă totul este corect el semnează votul și îl trimite înapoi alegătorului. Un vot nu poate fi folosit decât dacă este semnat de administrator. La final, pentru a demonstra că nu au fost adăugate, modificate sau șterse voturi, administratorul face publice mesajele primite împreună cu numărul acestora.

Responsabilitatea revine la alegător. Acesta verifică dacă semnătura este corectă și, în caz contrar, poate demonstra că semnătura este greșită. Aceste greșeli semnalate pot fi ignorate în cazul în care diferența dintre alegeri este mare. În același timp administratorii ce dau dovadă că au produs multe greșeli vor fi înlocuiți. Voturile corecte sunt trimise mai departe către deschidere și numărare. Pentru a trimitere se folosește o rețea de mixare pentru a ascunde alegerea făcută.

Voturile sunt amestecate și adunate de o autoritate ce se ocupă de numărarea lor sau sunt postate într-un buletin public pentru verificarea integrității acestora. Dacă sunt considerate ca fiind corecte, acestea sunt numerotate și adăugate într-o listă.

Dat fiind că administratorii postează numărul de voturi validate se cunoaște numărul exact de voturi ce trebuie să fie afișate. Cel mai mare dezavantaj constă în faptul că doar alegătorii vor putea verifica corectitudinea rezultatelor în cazul în care un vot a fost înlocuit.

Odată ce s-a terminat perioada de validare fiecare alegător trebuie să își trimită cheia privată pentru a putea deschide votul. Acest lucru se realizează din nou printr-o rețea de mixare pentru a se ascunde legătura dintre vot și alegător.

De la început până la final se poate observa o interacțiune constantă între alegător și protocolul de vot. Asemenea sisteme necesită ca electoratul să fie matur și educat pentru ca procedura să ajungă la final cu succes. În același timp protocolul trebuie ajustat din punct de vedere al contextului în care se produce votul. Dacă votul este la distanță trebuie să avem un mediu sigur pentru producerea lui. Toate aceste proceduri nu sunt folositoare dacă, spre exemplu, alegătorii pot fi observați în timp ce votează. Posibilitatea existenței unui observator poate duce la ruperea anonimității sau la manipularea votului.

O observație ce poate fi făcută atunci când alegătorii controlează procesul de votare constă în faptul că devine dificilă realizarea proprietății de receipt-free. Dat fiind că numai alegătorul poate verifica și deschide votul acesta îl poate reproduce. Prin urmare un utilizator poate demonstra unei entități rău intenționată cum a votat.

Din aceste motive această schema întâmpină dificultăți pentru alegeri pe scală mare. În [26] este prezentat un protocol ce poate fi verificat public și este receipt-free. Un alt aspect important constă în ușurința cu care se votează. Un alegător trebuie doar să marcheze opțiunea dorită, să îl decupeze și să îl scaneze. Buletinul de vot este format din două părți. Pe o parte se află opțiunile amestecate iar pe cea de a doua parte se găsesc spațiile ce trebuie marcate pentru a alege opțiunea de pe prima parte împreună cu un mesaj criptat ce poate fi decriptat doar de o mulțime cinstită de autorități. Acest mesaj criptat ascunde opțiunile corespunzătoare căsuțelor. Prima parte este detașată cu ușurință și aruncată pentru a ascunde cum a votat alegătorul. Cea de a doua parte este păstrată pentru ca în cazul în care votul nu se regăsește în rezultatele finale acest lucru să fie contestat. Diferența stă în faptul că în afară de autoritățile oficiale nimeni nu poate decripta mesajul de pe buletin. Astfel, chiar dacă alegătorii au o adeverință, ei nu pot demonstra nimănui în afară de un grup format din autoritățile oficiale ce este capabil să decripteze mesajul. Prin urmare protocolul este receipt-free. Odată ce votul a fost realizat el este trimis printr-o rețea de mixare verificabilă către autoritățile ce se ocupă de decriptare și numărare. În [26] se folosește verificarea parțială aleatorie pentru rețeaua de mixare.

Protocolul este verificabil și de către expeditor. La etapa de votare alegătorii pot verifica mesajul criptat pentru a se asigura că votul este înregistrat cum trebuie. Mesajul criptat este scanat și decriptat pe loc după care se verifică dacă acesta

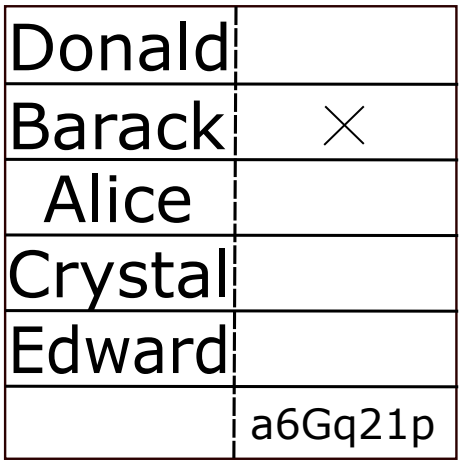


Figura 3.1: Buletin de vot folosit în Prêt à voter.

corespunde jumătății cu opțiunile scrise în clar. Este important totuși ca buletinul de vot să nu mai fie folosit pentru votare dacă este decriptat pentru verificare. Verificarea poate fi realizată de mai multe ori de către un alegător sceptic pentru a se asigura că totul este în regulă. Acest lucru descurajează fabricarea de buletine de vot greșite și creează încredere în protocol. Odată ce alegătorul decide că mesajele sunt corecte, el ia un nou buletin și votează. Protocolul este în continuare ușor de verificat de către alegător prin jumătatea de buletin criptată ce este păstrată de acesta.

3.3 Licitație electronică

Licitația electronică este un proces prin care se realizează vânzarea sau cumpărarea de bunuri sau servicii pentru unul sau mai mulți participanți în urma unor oferte făcute de aceștia. Licitațiile se bazează pe principiul economic al concurenței pentru a oferi cumpărătorilor bunuri sau servicii ce se potrivesc cerințelor și bugetelor acestora. În același timp vânzătorii obțin adesea prețul potrivit ofertei lor dar în același timp aceștia pot hotărî un preț minim și pot refuza tranzacția pentru a evita cazuri nefavorabile.

Odată cu apariția internetului licitațiile au devenit mult mai des folosite dat fiind publicul larg la care un vânzător poate ajunge. Acestea au ajuns să fie folosite des în vânzările în timp real de reclame unde agenții de publicitate cumpără spațiile pentru reclame bazat pe datele despre utilizator. Licitațiile mai sunt populare și în cadrul vânzărilor de proprietăți imobiliare sau cele de automobile.

Există patru tipuri tradiționale de licitații des întâlnite: licitație engleză, licitație olandeză, licitație în plic închis și licitație Vickrey. Pe lângă cele patru mai există și alte tipuri precum licitațiile de tip M+1 în care se scot la vânzare mai multe obiecte de același fel iar cumpărătorii fac o ofertă secretă pentru una sau mai multe copii al acestui obiect. Nu toate tipurile de licitație necesită anonimitate, licitația engleză spre exemplu începe de la o sumă de pornire la care un cumpărător poate face o

ofertă mai bună. Cumpărătorii fac astfel oferte din ce în ce mai mari până când se ajunge la o sumă ce este nefavorabilă. În fiecare punct al licitației se cunosc astfel cea mai mare sumă și cine a făcut oferta.

În licitația în plic închis fiecare licitant face o singură ofertă într-un plic închis iar, după ce perioada de ofertare se termină, organizatorul licitației deschide plicurile pentru a descoperi sumele și declară câștigător ofertantul cu cea mai mare sumă. Acest tip de licitație este folosit atunci când participanții doresc să nu divulge suma pe care o oferă în cazul în care nu câștigă. În același timp, dat fiind că se poate realiza o singură ofertă ce nu este cunoscută, licitanții nu își pot ajusta sumele oferite în funcție de ofertele celorlalți participanți. Acest tip de licitații se folosesc în vânzarea de contracte guvernamentale. În licitațiile electronice plicul este reprezentat prin criptare. Păstrarea anonimității este realizată prin folosirea rețelelor de mixare pentru comunicare și prin utilizarea unei a treia entități ce nu va colabora cu vânzătorii.

La fel ca în votul electronic există cazuri în care cumpărătorii sunt constrânși să liciteze într-un anumit mod. De exemplu un participant le cere celorlalți să ofere sume foarte mici pentru ca acesta să câștige licitația la un preț mai mic decât cel normal. Aceste cazuri trebuie evitate pentru a avea licitații corecte.

Dorința de păstrare a anonimității și confidențialității provine din mai multe motive. Dorința cumpărătorilor de a își ascunde modul de licitare este unul din motivele principale. Aceștia nu doresc ca felul în care au apreciat prețul să fie înregistrat pentru ca acest lucru să nu fie folosit împotriva lor în viitor. Din același motiv vânzătorul nu trebuie să cunoască ofertele sau distribuția lor pentru a nu avea un avantaj în viitor când va scoate la licitație un lucru asemănător.

În continuare vor fi prezentate protocoalele de licitații în plic închis dată fiind utilitatea rețelelor de mixare din cadrul acestora.

3.3.1 Specificații

Licitațiile în plic închis necesită atât siguranță cât și eficiență pentru a fi practice. În continuare vor fi prezentate o serie de proprietăți folositoare pentru atingerea acestui scop:

- **Corectitudine** - Un preț câștigător corect și câștigatorul/câștigătorii licitației, potrivit regulilor, pot fi determinați doar atunci când toate entitățile participante acționează onest. Astfel toate ofertele trebuie luate în considerare iar odată ce perioada de licitare se termină nimeni nu poate schimba ofertele.

- **Anonimitate** - Odată ce licitația se termină cât și în timpul acesteia nimeni nu cunoaște identitatea celor ce au făcut ofertele necâștigătoare.

- **Confidențialitate** - Toate sumele licitate rămân secrete până la finalul perioadei de licitare.

- **Non-repudiere** - Câștigătorul nu poate nega oferta pe care a făcut-o. Astfel câștigătorul trebuie să plătească prețul hotărât în urma regulilor licitației.

- **Verificabilă public** - Oricine poate verifica dacă oferta câștigătoare are cea mai mare valoare și poate confirma public dacă câștigătorul este valid sau nu.
- **Nefalsificabilitate** - Nimeni nu se poate da drept alt licitant.
- **Robustețe** - Comportamentul rău voitor sau greșit al unui participant nu poate afecta desfășurarea licitației și nu poate duce la un rezultat incorect.
- **Eficiență** - Protocolul folosit trebuie să fie rezonabil din punct de vedere al puterii de calcul și costului de comunicare necesare.

Aceste proprietăți sunt suficiente pentru a avea o licitație în plic închis practică și sigură. Mai există o serie de proprietăți opționale folosite de unele protocoale de licitație ce aduc beneficii suplimentare:

- **Receipt-free** - Nimeni, nici măcar licitantul însuși, nu poate fi capabil să dovedească vreo informație despre ofertele făcute unei entități din exterior.
- **Oferte flexibile** - Sumele licitate nu sunt limitate la o mulțime de prețuri predefinită.
- **Reguli flexibile** - Protocolul folosit este independent de regulile folosite de o licitație anume. Prin urmare protocolul poate folosi orice tip de reguli fără să întâmpine probleme.

Pe lângă aceste proprietăți protocoalele au adesea nevoie de instrumente suplimentare pentru funcționarea corectă. Cele mai întâlnite sunt:

- **Bulletin board.** Adesea pentru verificarea corectitudinii se folosește un spațiu pentru afișarea informațiilor publice. Oricine poate citi informațiile afișate și nimeni nu poate șterge vreo informație. În același timp numai entitățile autorizate pot adăuga informații. Aceste entități pot avea un identificator care poate ajuta la identificare în cazuri de repudierie.
- **Canal sigur de comunicare.** Aceste canale transmit informația într-un singur sens pastrând conținutul secret față de cei din afară. Aceste canale de comunicare sunt folosite pentru a împiedica ascultarea informațiilor transmise precum chei criptografice sau alte date sensibile.
- **Rețele de mixare.** La fel ca în cazul canalelor sigure rețelele se folosesc pentru a ascunde informațiile transmise. Un lucru pe care îl aduc în plus aceste rețele este anonimitatea transmițătorului. Astfel destinatarul nu cunoaște identitatea transmițătorului cât timp aceasta nu poate fi obținută din mesajul primit.

3.3.2 Structura

După cum am văzut în proprietățile prezentate anterior sumele oferite pot fi de două feluri:

- **Prețuri fixe.** Cumpărătorii trebuie să aleagă dintr-o mulțime prestabilită de oferte. Odată cu dimensiunea mulțimii crește astfel și precizia de care beneficiează aceștia.
- **Prețuri flexibile.** Prețurile nu depind de o mulțime fixă. Cumpărătorii pot licita oricât de precis doresc.

Când vine vorba de încredere există diferite feluri de structuri bazate pe tipul entităților participante. Avem prin urmare:

- **Modelul cu prag de încredere.** În aceste licitații se folosesc un număr de m organizatori din care o parte sunt considerați ca fiind de încredere. Aceștia evaluează împreună cine este câștigătorul licitației. Rezultatul este corect dacă numărul de organizatori din partea considerată onestă nu se află sub un prag stabilit de protocol. Acesta este considerat ca fiind cel mai slab model, dată fiind necesitatea unui număr mai mare de participanți. Odată cu creșterea acestui număr scade și eficiența protocolului. Cumpărătorul trebuie să aibă încredere într-un număr ridicat de entități, lucru ce nu este dorit în practică.
- **Modelul cu o a treia autoritate.** O a treia entitate ce nu este interesată în a colabora cu cumpărătorii sau vânzătorii este folosită. Această entitate mediază licitația și asigură faptul că anonimitatea, confidențialitatea și non-repudierea sunt respectate.
- **Modelul fără organizatori.** În acest tip de protocol licitația este realizată doar de cumpărători. Dat fiind aspectul competitiv al licitațiilor este presupus că aceștia nu vor colabora. Cât timp cel puțin un cumpărător nu colaborează informațiile despre licitație nu vor fi divulgate. Există însă două probleme cu acest model. Atunci când un participant constrânge restul cumpărătorilor se poate ajunge la ruperea confidențialității. A doua problemă constă în eficiența scăzută a protocolului.

Când vine vorba de organizarea etapelor licitației putem vorbi de patru stagii. Acestea nu se intersectează și sunt dependente unul față de celălalt. Ele sunt în ordine după cum urmează:

1. **Etapa de pregătire.** În această etapă se hotărăsc regulile, bunurile sau serviciile licitate și componentele criptografice folosite. Acestea sunt făcute publice împreună cu organizatorii ce se vor ocupa de licitație și orice alt detaliu necesar. Aici este anunțat și intervalul de timp pentru etapa de licitare.
2. **Etapa de licitare.** În această etapă fiecare participant trimite o singură ofertă "sigilată" (e.g. criptată). În unele protocoale se trimite mai întâi o dovadă prin care cumpărătorul garantează faptul că oferta nu poate fi schimbată

după deschidere. Această angajare a responsabilității înainte de trimiterea ofertei este singura metodă de asigurare a corectitudinii fără a avea încredere în organizatorul licitației.

3. **Etapa de deschidere.** La finalul perioadei de licitare organizatorii deschid ofertele primite cu ajutorul celorlalți participanți. Este anunțată suma/sumele câștigătoare iar organizatorii află identitatea câștigătorului/câștigătorilor. Este important ca identitățile pierzătorilor cât și sumele oferite să nu fie cunoscute de nimeni, nici măcar de organizatorii licitației. Există protocoale ce fac publice identitatea câștigătorului dar pentru siguranța acestuia este adesea preferat ca acest lucru să nu se întâmple. În această etapă se fac și pașii necesari în cazul unei egalități e.g. printr-o nouă licitație sau prin negocieri între vânzător și cumpărători.

4. **Etapa de tranzacționare.** Câștigătorul cumpără bunul sau serviciul cu prețul hotărât în urma aplicării regulilor licitației. Dacă acesta neagă faptul că oferta îi aparține, organizatorii licitației parcurg pașii de dovedire a identității câștigătorului.

3.3.3 Modele de scheme de licitație

În [24] se discută importanța confidențialității ofertelor după etapa de licitare. Relaxarea din acest punct de vedere duce la beneficii precum prețuri și reguli de licitație flexibile într-un mod eficient. Această relaxare se realizează prin afișarea tuturor sumelor oferite. Este important de notat că doar sumele sunt făcute publice, nu și cine a făcut oferta. De asemenea acest protocol folosește o rețea de mixare proprie pentru a evidenția importanța acestora când vine vorba de un protocol eficient. Se trimit dovezi ale ofertei înainte de a trimite oferta sigilată pentru a asigura faptul că licitația este corectă. Aceste dovezi sunt folosite pentru a evita cazuri în care organizatorii licitației pot colabora cu alți participanți pentru a obține un preț mai bun. În cazurile în care un participant nu colaborează, organizatorii licitației ce dețin serverele de mixare dezvăluie identitatea acestora facând public drumul parcurs de mesajele dăunătoare. Această schemă este simplă și eficientă însă nu poate proteja utilizatorii de cazuri de constrângere. Utilizatorii pot dovedi unui participant rău intenționat ce sumă au oferit. În același timp anonimitatea este păstrată atât timp cât cel puțin un server de mixare este onest. În cazul în care toate aceste servere sunt controlate de o singură autoritate anonimitatea dispare.

Când vine vorba de licitații cu un număr mic de cumpărători anumite protocoale precum [13] devin eficiente. Prin folosirea semnăturilor circulare (Ring signatures) se obține non-repudiere și confidențialitate atunci când utilizatorii nu cooperează. Prin semnătură circulară ne referim la o semnătură digitală deținută de o mulțime de posibili semnatari. Semnatarii sunt cunoscuți public însă nu se cunoaște care dintre ei folosește semnătura. În același timp prețurile sunt fixe pentru a asigura faptul că numai cea mai mare ofertă este deschisă. Acesta este singurul mod prin care un organizator poate afla cel mai mare preț fără a deschide toate ofertele. Non-repudierea

este rezolvată de către o a treia autoritate de încredere. Această autoritate colaborează cu organizatorii licitației pentru a identifica câștigătorul în acest caz. În același mod poate fi identificat și un participant ce împiedică rularea corectă a protocolului. Odată identificat acesta este exclus din lista celor participanți. Dată fiind utilizarea unui bulletin board toate datele necesare pentru verificarea rezultatului sunt publice. Oricine poate reproduce rezultatul.

Capitolul 4

Aplicație de vot electronic

4.1 Arhitectura aplicației

În continuare va fi prezentată structura unei aplicații de vot bazată pe lucrarea lui Fujioka et al.[10] prezentată anterior. Limbajul folosit pentru creare aplicației este Python, pentru bazele de date se folosește SQLite. Pentru realizarea comunicării în rețea se folosesc librării precum flask și requests. SQLAlchemy este folosit pentru comunicarea aplicațiilor cu bazele de date. Partea de operații criptografice este realizată cu ajutorul librăriei Pycryptodome și cu o implementare proprie pentru sistemul de criptare Elgamal homomorfic folosit de rețeaua de mixare la criptare, recriptare respectiv decriptare. Aplicația este organizată în mai multe pachete fiecare având un rol bine definit în cadrul aplicației. Avem astfel pachete ce implementează:

- **VotingBooth** - clientul folosit de alegători pentru trimiterea voturilor.
- **VSS** - serverul ce se ocupă de înregistrarea și inițializarea rețelelor de mixare
- **PoolServer** - serverul ce se ocupă de gestionarea mesajelor.
- **MixServer** - structura unui server de mixare.
- **BulletinBoard** - structura unui panou public pentru afișarea mesajelor
- **Administrator** - serverul ce se ocupă de semnarea voturilor.
- **Counter** - implementarea autorității ce se ocupă de numărarea voturilor.

În continuare vor fi prezentate pachetele enumerate mai sus.

4.1.1 Structura Aplicației

VotingBooth

Partea interactivă a aplicației constă într-un client folosit pentru votare. Interfața este realizată cu ajutorul librăriei grafice standard din python, tkinter. Întrebarea și răspunsurile sunt obținute de la autoritatea care se ocupă de procesul de votare.

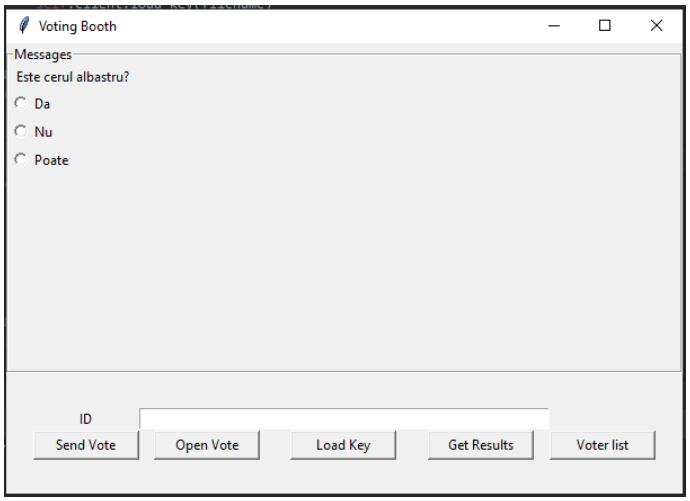


Figura 4.1: Fereastra folosită de alegători pentru votare

Comunicarea dintre aplicații atât aici cât și în cele ce urmează se realizează prin protocolul HTTP. Utilizatorul introduce ID-ul înregistrat de administrator și deschide fișierul ce conține cheia criptografică asociată ID-ului cu butonul *LoadKey*. Se selectează răspunsul dorit și se apasă butonul de *Send*. Mesajul este trimis către administrator iar apoi este postat pe panoul pentru mesaje. În același timp detaliile despre vot împreună cu semnătura administratorului sunt stocate într-un fișier local. Odată ce perioada de votare se termină utilizatorii obțin lista de voturi prin *VoterList*. Dacă se regăsește în listă acesta poate deschide votul folosind fișierul stocat anterior prin apăsarea butonului *OpenVote*. Odată ce voturile au fost calculate butonul *GetResults* este folosit pentru a afișa rezultatele.

Administrator

Serverul administrator este un server tip REST ce se ocupă de verificarea și semnarea mesajelor.

```
@app.route('/sign', methods=['POST'])
def sign_message():
    # request={ID:str,e:hex,s:str}
    if app.config["OPEN"] is False:
        abort(403)
    req_json = json.loads(request.get_data())
    ID = req_json["ID"]
    voter = db.get_by_voter_id(ID)
    if voter is None:
        print("Voter not registered")
        abort(401)
    if voter["voted"] == "True":
        print("Already Voted")
        abort(403)
    voter_key_str = voter["key"]
```

```
voter_key_bytes = voter_key_str.encode()
voter_key = RSA.import_key(voter_key_bytes)
# Hashul Votului
e = req_json["e"]
# Semnatura Votului
s = req_json["s"]
if verify(voter_key, int_to_bytes(e), bytes.fromhex(s)) is False:
    print("Wrong signature")
    abort(403)
db.update_voter(ID, hex(e), s)
pv_key = app.config["SECRET_KEY"]
signature = sign(pv_key, e)
return dict(sig=signature)
```

În cazul în care etapa de votare s-a terminat administratorul refuză să semneze alte voturi. În caz contrar se verifică legitimitatea cererii comparând datele furnizate cu cele înregistrate în baza de date. Dacă cererea are dreptul de a vota administratorul continuă prin a verifica dacă semnătura este corectă. În cazul în care semnătura este corectă se continuă prin semnarea votului, semnătură ce va fi trimisă înapoi ca răspuns. În același timp se salvează în baza de date informațiile furnizate de alegător împreună cu faptul că acesta a votat. Această bază de date poate fi verificată mai târziu în cazul în care apar probleme majore cu procesul de vot.

VSS

Rețeaua de mixare este inițializată folosind protocolul de criptografie distribuită prezentat în [23]. Rețeaua obține o pereche de chei Elgamal folosite pentru procesul de mixare. Fiecare server deține o bucată din cheia privată folosită pentru decriptarea mesajelor. Astfel, pentru a decripta un mesaj trebuie ca jumătate plus 1 servere oneste să colaboreze. Pentru procesul de inițializare este necesar acest server ce se ocupă de înregistrarea rețelelor de mixare și postarea lor. Fiecare server se înregistrează la acest server și primește o listă de servere cu care va colabora pentru a crea un mixnet. După ce mixnetul a fost creat, acest server este notificat și în cazul în care toate serverele rețelei sunt de comun acord, informațiile precum cheia publică a rețelei și lista de servere ce constituie drumul pe care un mesaj îl ia în mixnet sunt făcute publice.

PoolServer

Odată ce rețeaua este inițializată aceasta are o cheie publică unică. Cu această cheie se realizează o singură criptare de către utilizatori iar mesajul rezultat este postat pe un panou de mesaje. Un pool server este folosit pentru adunarea mesajelor de pe panoul de mesaje. Mesajele ce au fost deja mixate nu sunt luate în calcul pentru a împiedica atacuri de tip replay. Acest server rupe legătura dintre mesajul postat și

momentul în care acesta a fost postat selectând aleator mesaje pentru a crea grupuri de mesaje numite batchuri. Odată ce suficiente mesaje au fost adunate un grup de mesaje este creat și trimis către mixnet. Detaliile despre mixnet sunt obținute de la serverul ce se ocupă de înregistrarea lor.

MixServer

Pool serverul postează un batch de mesaje pe panoul public al rețelei de mixare și trimite id-ul postării către toate serverele. Aceste servere au un drum prestabilit pentru mixare, verificare și decriptare. Odată ce serverele primesc id-ul de la serverul pool acestea verifică dacă batchul există în baza de date. Dacă nu există se creează o noua intrare cu datele necesare pentru mixare. În același timp în tabela de mesaje se adaugă mesajele primite împreună cu secretul necesar pentru recriptarea fiecărui mesaj. În același timp fiecare server generează un challenge folosit mai târziu pentru verificarea corectitudinii mixării și postează un hash al challengeului pe panoul public al rețelei. Procesul de mixare poate începe cu primul server din rețea.

Odată ce un server trebuie să mixeze un batch, acesta ia mesajele asociate batchului, le permută folosind permutarea asociata batchului și recriptează fiecare mesaj folosind cheia publică a rețelei. Acest server continua să demonstreze egalitatea logarimilor folosind protocolul ZKP descris anterior.

Procesul de dovedire a corectitudinii începe prin postarea pe panoul public a unui tuplu (a, b) generat aleator. Următorul server oferă challengeul împreună cu secretul folosit pentru generarea hashului de pe panoul public. Acest secret va fi folosit drept valoarea aleatoare c. Challengeul constă în jumătate din intrările serverului ce a mixat mesajele. Acest server trebuie să facă public mesajele recriptate asociate acestor intrări. Serverul verifică hashul și continuă prin a posta permutarea inversă asociată challengeului împreună cu z. Pentru a verifica faptul că mixarea a fost realizată corect se dovedește folosind datele furnizate de server faptul că $log_g(c1' * c1^{-1}) = log_h(c2' * c2^{-1})$ unde $c1$ și $c2$ reprezintă produsul intrărilor asociate challengeului iar $c1'$ și $c2'$ sunt produsele ieșirilor asociate acestor intrări. Dacă serverele acceptă această dovadă procesul de mixare continuă în mod normal, altfel mesajele mixate de serverul în cauză nu sunt luate în considerare.

Odată ce toate serverele au mixat mesajele începe procesul de decriptare. Primul pas constă în verificarea faptului că cel puțin jumătate plus 1 din servere au mixat corect mesajele. În cazul în care nu au fost suficiente servere, mesajele nu sunt decriptate. Pentru a decripta un batch serverele încep prin a decripta parțial ultimele mesaje valide până când jumătate din servere au facut această operație. Ultimul server realizează o interpolare bazată pe serverele ce au participat la decriptare și folosește rezultatul pentru a finaliza decriptarea așa cum este prezentat și în [23]. Odată ce mesajele sunt decriptate, acestea sunt postate pe panoul public folosind cheia publică a rețelei drept identitate.

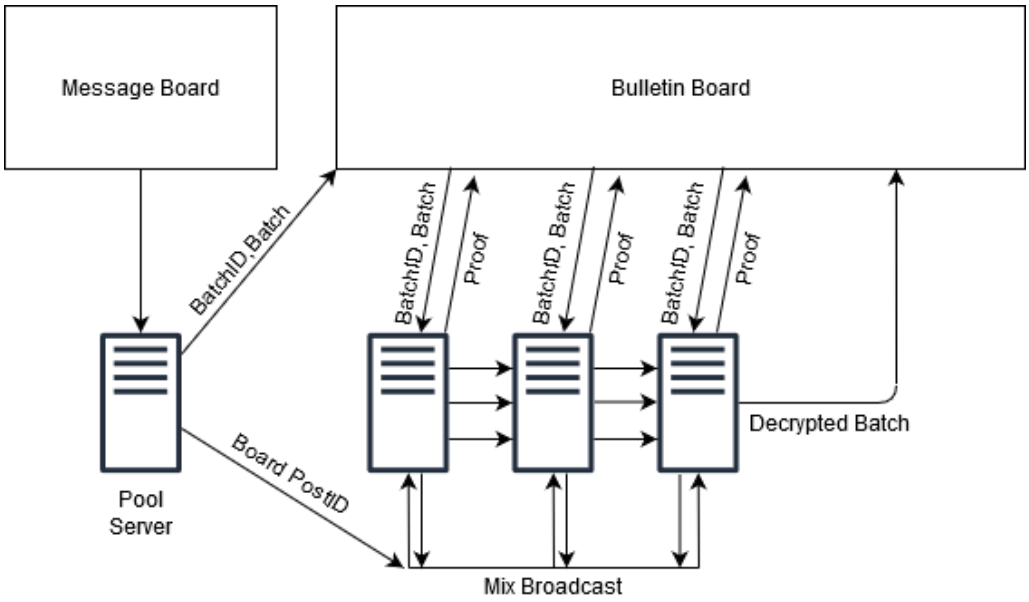


Figura 4.2: Procesarea mesajelor de către rețeaua de mixare.

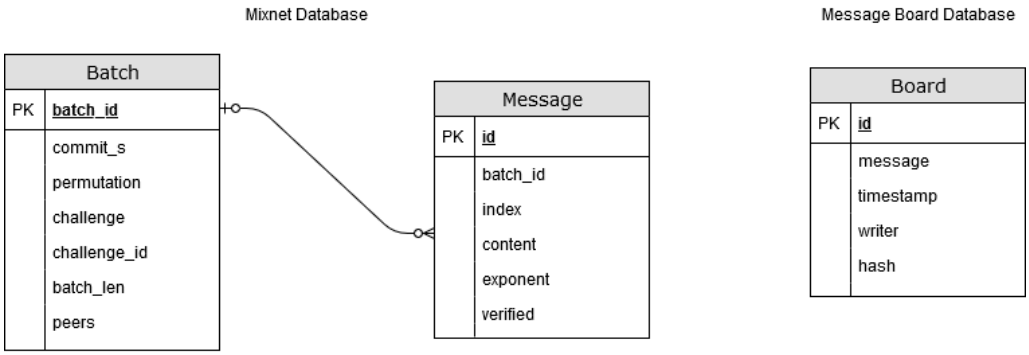


Figura 4.3: Structura bazelor de date folosite.

BulletinBoard

Serverele folosite drept panouri publice și panouri pentru mesaje sunt implementate folosind o arhitectură de tip REST. Acest server înregistrează mesajul împreună cu momentul de timp în care a fost înscris, identitatea scriitorului și un hash pentru verificare. Identitatea scriitorului se bazează pe chei criptografice și este reprezentată prin cheia publică a scriitorului.

Counter

Serverul ce se ocupă de numărarea voturilor și informații precum lista de voturi, întrebarea folosită pentru votare, perioadele de votare, numărare și afișarea rezultatelor. Acest server este separat în două părți. O parte se ocupă de informațiile cerute de alegători iar a doua parte numără și verifică voturile primite.

Partea de informare este realizată printr-o arhitectură de tip REST. În timpul procesului de votare aceasta oferă întrebarea și răspunsurile asociate. Odată ce timpul pentru votare se termină aceste cereri vor fi înlocuite de o notificare a acestui

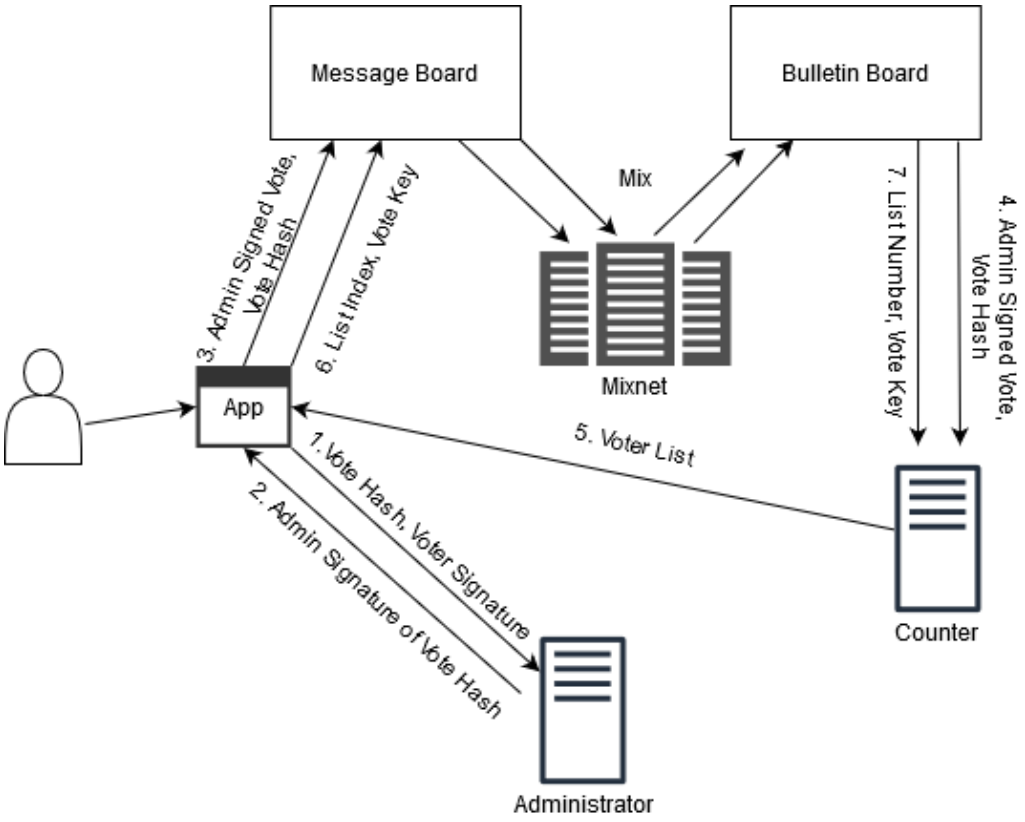


Figura 4.4: Procesul de votare bazat pe Fujioka et al.

lucru. Serverul stochează toate voturile valide primite într-o listă care nu este făcută publică decât după ce perioada de votare s-a terminat. Lista este stocată într-o bază de date. În cazul în care lista este cerută înainte aceste cereri sunt refuzate. Odată ce rezultatul este calculat serverul acceptă cereri pentru obținerea acestuia, în caz contrar cererile sunt refuzate.

Cea de a doua parte se ocupă de partea de procesare a voturilor. Acest server începe un temporizator și continuă prin a aduna voturile postate de rețeaua de mixare. În această primă etapă se obțin hashurile voturilor alegătorilor împreuna cu o semnătură a hashului realizată de administrator. Folosind cheia publică a administratorului serverul verifică faptul că semnătura este validă și salvează hashurile. În caz contrar mesajele nu sunt salvate. Odată ce temporizatorul anunță faptul că perioada de votare s-a terminat serverul trimite cereri de oprire împreună cu o parolă secretă către administrator și partea care se ocupă de informare. Acestea la rândul lor vor refuza cereri pentru crearea de voturi noi. În continuare serverul află de la administrator numărul de voturi semnate și citește ieșirea mixnetului pentru a obține voturile finale. Acesta primește de la alegători secretul folosit pentru realizarea hashului votului împreună cu identificatorul votului corespunzător. Votul în cauză este luat și testat folosind secretul. Dacă hashul reprezintă un răspuns valid pentru întrebare votul este considerat ca fiind valid. În cazul în care acesta corespunde unui răspuns rezultatul este actualizat. Odată ce toate voturile au fost primite conform administratorului numărarea voturilor se termină și rezultatul obținut este trimis către serverul ce se ocupă de informare.

Aplicația realizează astfel o numărare sigură și validă a tuturor voturilor și poate demonstra corectitudinea procesului.

Concluzii

Scopul principal al aceste lucrări a fost de a prezenta importanța anonimității și de a implementa o soluție pentru realizarea acesteia. S-au folosit diferite mecanisme și protocoale criptografice în acest scop și o aplicare practică a acestora.

În prima parte au fost prezentate instrumente criptografice folosite pentru realizarea confidențialității mesajelor și pentru verificarea integrității și autenticității acestora. În continuare a fost detaliat conceptul de mixnet și modurile în care poate fi implementat. Au fost enumerate posibile vulnerabilități ale rețelelor de bază și sugerate moduri de a împiedica manifestarea acestora. A urmat protocolul folosit în aplicația propriu-zisă împreună cu specificații atât pentru această aplicație cât și pentru aplicații ce se folosesc de mixneturi în general. Aceste detalii tehnice s-au făcut cunoscute în aplicația de vot electronic implementată. Aceasta a reușit să realizeze un proces de vot sigur, verificabil și anonim într-un mod eficient și scalabil.

Pentru viitor această soluție poate fi îmbunătățită prin implementarea unor panouri publice mult mai sigure folosind mecanisme precum cele din [12] cât și prin implementarea unor servere avansate de monitorizare și gestionare a reputației mixneturilor. Optimizări se pot face și la bazele de date pentru a spori siguranța și integritatea informațiilor.

Sper ca această lucrare să încurajeze pe viitor dezvoltarea și extinderea principiilor anonimității în diferite aplicații.

Bibliografie

[1] Masayuki Abe. "Universally verifiable mix-net with verification work independent of the number of mix-servers". In: *Advances in Cryptology — EUROCRYPT'98*. Ed. by Kaisa Nyberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 437–447. ISBN: 978-3-540-69795-4.

[2] Josh Cohen Benaloh. "Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret (Extended Abstract)". In: *Advances in Cryptology — CRYPTO' 86*. Ed. by Andrew M. Odlyzko. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 251–260. ISBN: 978-3-540-47721-1.

[3] Dan Boneh and Philippe Golle. *Almost Entirely Correct Mixing with Applications to Voting*. 2002.

[4] David Chaum. "The dining cryptographers problem: Unconditional sender and recipient untraceability". In: *Journal of Cryptology* 1.1 (Jan. 1988), pp. 65–75. ISSN: 1432-1378. DOI: 10.1007/BF00206326. URL: https://doi.org/10.1007/BF00206326.

[5] David L. Chaum. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms". In: *Commun. ACM* 24.2 (Feb. 1981), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/358549.358563. URL: http://doi.acm.org/10.1145/358549.358563.

[6] David Chaum and Torben Pryds Pedersen. "Wallet Databases with Observers". In: *Advances in Cryptology — CRYPTO' 92*. Ed. by Ernest F. Brickell. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 89–105. ISBN: 978-3-540-48071-6.

[7] Yvo Desmedt and Kaoru Kurosawa. "How to Break a Practical MIX and Design a New One". In: *Advances in Cryptology — EUROCRYPT 2000*. Ed. by Bart Preneel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 557–572. ISBN: 978-3-540-45539-4.

[8] Roger Dingledine et al. "A Reputation System to Increase MIX-Net Reliability". In: *Information Hiding*. Ed. by Ira S. Moskowitz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 126–141. ISBN: 978-3-540-45496-0.

[9] T. Elgamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *IEEE Transactions on Information Theory* 31.4 (July 1985), pp. 469–472. DOI: 10.1109/TIT.1985.1057074.

[10] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. "A practical secret voting scheme for large scale elections". In: *Advances in Cryptology — AUSCRYPT '92*. Ed. by Jennifer Seberry and Yuliang Zheng. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 244–251. ISBN: 978-3-540-47976-5.

[11] Philippe Golle et al. "Optimistic Mixing for Exit-Polls". In: *Advances in Cryptology — ASIACRYPT 2002*. Ed. by Yuliang Zheng. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 451–465. ISBN: 978-3-540-36178-7.

[12] James Heather and David Lundin. "The Append-Only Web Bulletin Board". In: *Formal Aspects in Security and Trust*. Ed. by Pierpaolo Degano, Joshua Guttman, and Fabio Martinelli. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 242–256. ISBN: 978-3-642-01465-9.

[13] Xiong Hu, Qin Zhiguang, and Li Fagen. "An Anonymous Sealed-bid Electronic Auction Based on Ring Signature". In: *International Journal of Network Security* 62 (Jan. 2009).

[14] Markus Jakobsson. "A practical mix". In: *Advances in Cryptology — EUROCRYPT'98*. Ed. by Kaisa Nyberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 448–461. ISBN: 978-3-540-69795-4.

[15] Markus Jakobsson. "Flash Mixing". In: *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing*. PODC '99. Atlanta, Georgia, USA: ACM, 1999, pp. 83–89. ISBN: 1-58113-099-6. DOI: 10.1145/301308.301333. URL: http://doi.acm.org/10.1145/301308.301333.

[16] Markus Jakobsson and Ari Juels. "An Optimally Robust Hybrid Mix Network". In: *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing*. PODC '01. Newport, Rhode Island, USA: ACM, 2001, pp. 284–292. ISBN: 1-58113-383-9. DOI: 10.1145/383962.384046. URL: http://doi.acm.org/10.1145/383962.384046.

[17] Markus Jakobsson and Ari Juels. "An Optimally Robust Hybrid Mix Network". In: *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing*. PODC '01. Newport, Rhode Island, USA: ACM, 2001, pp. 284–292. ISBN: 1-58113-383-9. DOI: 10.1145/383962.384046. URL: http://doi.acm.org/10.1145/383962.384046.

[18] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. "Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking". In: *Proceedings of the 11th USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2002, pp. 339–353. ISBN: 1-931971-00-5. URL: http://dl.acm.org/citation.cfm?id=647253.720294.

[19] Masashi Mitomo and Kaoru Kurosawa. "Attack for Flash MIX". In: *Advances in Cryptology — ASIACRYPT 2000*. Ed. by Tatsuaki Okamoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 192–204. ISBN: 978-3-540-44448-0.

[20] Wakaha Ogata et al. “Fault tolerant anonymous channel”. In: *Information and Communications Security*. Ed. by Yongfei Han, Tatsuaki Okamoto, and Sihan Qing. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 440–444. ISBN: 978-3-540-69628-5.

[21] Miyako Ohkubo and Masayuki Abe. “A Length-Invariant Hybrid Mix”. In: *Advances in Cryptology — ASIACRYPT 2000*. Ed. by Tatsuaki Okamoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 178–191. ISBN: 978-3-540-44448-0.

[22] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. “Efficient Anonymous Channel and All/Nothing Election Scheme”. In: *Advances in Cryptology — EUROCRYPT ’93*. Ed. by Tor Helleseth. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 248–259. ISBN: 978-3-540-48285-7.

[23] Torben Pryds Pedersen. “A Threshold Cryptosystem without a Trusted Party”. In: *Advances in Cryptology — EUROCRYPT ’91*. Ed. by Donald W. Davies. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 522–526. ISBN: 978-3-540-46416-7.

[24] Kun Peng et al. “Efficient Implementation of Relative Bid Privacy in Sealed-Bid Auction”. In: vol. 2908. Aug. 2003, pp. 244–256. DOI: 10.1007/978-3-540-24591-9_19.

[25] R.L. Rivest, A. Shamir, and L. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. In: *Communications of the ACM* 21 (1978), pp. 120–126.

[26] Peter Ryan et al. “PrÊt À Voter: a Voter-Verifiable Voting System”. In: *Information Forensics and Security, IEEE Transactions on* 4 (Jan. 2010), pp. 662–673. DOI: 10.1109/TIFS.2009.2033233.

[27] Kazue Sako and Joe Kilian. “Receipt-Free Mix-Type Voting Scheme”. In: *Advances in Cryptology — EUROCRYPT ’95*. Ed. by Louis C. Guillou and Jean-Jacques Quisquater. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 393–403. ISBN: 978-3-540-49264-1.

[28] Adi Shamir. “How to Share a Secret”. In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176. URL: http://doi.acm.org/10.1145/359168.359176.