Code Verification E-Voting

Anthi Orfanou

UoA

March 23, 2012
E-Voting

- 2 types of E-Voting
  - Kiosk Voting
  - Internet Voting

- Internet Voting - 3 phases
  - Registration/Key Generation
  - Elections
    - **Voter**: Casts a vote
    - **Personal Computers**: Vote Encryption, Vote Submission,
    - **Vote Collectors**: Vote Storing, Forward to Tallier
  - Tallying
    - **Tallier**: Vote Decryption, Counting
E-Voting Protocol Properties

- Correctness
- Robustness
- Privacy
- Integrity
- Coercion Resistance
- Verifiability
- Usability

In practice many of them conflict!
Code Verification Protocols

- Provide **security codes** as a receipt for voting
- Guarantee **vote integrity**
- Additional entities:
  - **Security Code Generators**
  - **Messengers**
  - **Out-of-band communication channels**

The Norwegian E-Voting System
The Norwegian E-Voting System

PC

Vote Collector

after elections

Tallier

before elections

Code Generator

Creates and sends the security codes

Voter

Cnd : Code

Messenger
The Norwegian E-Voting System

PC

Cnd

Submits a vote

Cnd : Code

Voter

Vote Collector

Messenger

Tallier

before elections

after elections

Code Generator

Creates and sends the security code
The Norwegian E-Voting System

- **PC**
  - Encrypts and sends the vote

- **Vote Collector**
  - Stores the vote
  - Forwards to the tallier

- **Cnd**
  - Submits a vote

- **Messenger**
  - Participates in code reconstruction

- **Voter**
  - Submits a vote

- **Tallier**

- **Code Generator**
  - Creates and sends the security code

- **Cnd : Code**
The Norwegian E-Voting System

PC

Vote Collector
Stores the vote
Forwards to the tallier

Voter
Cnd: Code
Submits a vote

Messenger
Participates in code reconstruction

Cnd: Code
Reconstructs and sends the code

Tallier

Code Generator
Creates and sends the security code

after elections

before elections
The Norwegian E-Voting System

Cnd: the voter submits a vote.

PC: the voter encrypts and sends the vote.

Vote Collector: stores the vote and forwards it to the tallier.

Tallier: participates in code reconstruction.

Code Generator: creates and sends the security codes.

Messenger: helps in the process of code reconstruction.

After elections, the tallier finalizes the vote count.
Coalition of malicious PC and messenger can submit forged ballots!
The Norwegian E-Voting System

2 main approaches to generate and transfer the security codes

- **The Proxy Oblivious Transfer** [Heiberg, Lipmaa, Van Laenen 2010 [1]]
  - Random security codes
  - Easy setup phase
  - Linear communication complexity in the number of Candidates

- **The Pseudo-random Composition** [Gjøsteen 2010-11 [2],[3]]
  - Pseudo-random security codes
  - More complex setup phase
  - Communication complexity independent of the number of Candidates

Both share similar security issues!
Proxy Oblivious Transfer

3 entities involved:

- **Chooser**: chooses an index \( x \)
- **Sender**: Stores a database \( f = (f_0, f_1, \ldots, f_{n-1}) \)
- **Proxy**: Retrieves a single item \( f_x \) without knowing \( x \)
  - Strong POT: the proxy retrieves the correct time by performing certain computations
  - Weak POT: the proxy stores additional supporting data that do no leak information
The Proxy Oblivious Transfer E-Voting [HLV’10]

ElGamal PKC $\langle G, q, g \rangle$, $N$ candidates, $(pkm, skm)$ messenger’s keys, $(pkt, skt)$ tallier’s keys, $H$ random oracle

∀ voter $V$
∀ candidate $i$
$f_i^V \leftarrow Z_q$
$Code_V[i] = H(g^{f_i^V})$
The Proxy Oblivious Transfer E-Voting [HLV’10]

ElGamal PKC \( \langle G, q, g \rangle \), \( N \) candidates, \( (pkm, skm) \) messenger’s keys, \( (pkt, skt) \) tallier’s keys, \( H \) random oracle

\[ \{ g^{f_0^V}, g^{f_1^V}, \ldots, g^{f_{N-1}^V} \} \]

\[ \langle f_0^V, f_1^V, \ldots, f_{N-1}^V \rangle \]

Vote Collector

PC

Messenger

Tallier

before elections

Code Generator

after elections

Voter

\[ Code_V[i] = H(g^{f_i^V}) \]

\[ i \leftrightarrow Code_V[i] \]
The Proxy Oblivious Transfer E-Voting [HLV'10]

ElGamal PKC $\langle G, q, g \rangle$, $N$ candidates, $(pkm, skm)$ messenger's keys, $(pkt, skt)$ tallier's keys, $H$ random oracle
The Proxy Oblivious Transfer E-Voting [HLV’10]

ElGamal PKC $\langle G, q, g \rangle$, $N$ candidates, $(pkm, skm)$ messenger’s keys, $(pkt, skt)$ tallier’s keys, $H$ random oracle

$E_m = Enc_{pkm}(g^x)$
The Proxy Oblivious Transfer E-Voting \[\text{[HLV'10]}\]

ElGamal PKC \(\langle G, q, g \rangle\), \(N\) candidates, \((pkm, skm)\) messenger’s keys, \((pkt, skt)\) tallier’s keys, \(H\) random oracle

\[
E_m = Enc_{pkm}(g^x)
\]

\[
\langle f_0^V, f_1^V, \ldots, f_{N-1}^V \rangle
\]

\[
\forall i, r_i \leftarrow Z_q
\]

\[
e_i = (Enc_{pkm}(g_i^x))^{r_i} \cdot Enc_{pkm}(g_i^{f_i^V})
\]
The Proxy Oblivious Transfer E-Voting [HLV’10]

ElGamal PKC $\langle G, q, g \rangle$, $N$ candidates, $(pkm, skm)$ messenger’s keys, $(pkt, skt)$ tallier’s keys, $H$ random oracle

Vote Collector

$E_m = Enc_{pkm}(g^x)$

$\{f_0^V, f_1^V, \ldots, f_{N-1}^V\}$

$\forall i$

$r_i \leftarrow Z_q$

$e_i = (Enc_{pkm}(g_i^x))^{r_i} \cdot Enc_{pkm}(g_i^x)$

Voter

PC

$\{e_0, e_1, \ldots, e_{N-1}\}$

after elections

Tallier

before elections

Code Generator

Messenger

$\langle pkm, skm \rangle$
The Proxy Oblivious Transfer E-Voting [HLV’10]

ElGamal PKC \(\langle G, q, g \rangle\), \(N\) candidates, \((pk_m, sk_m)\) messenger’s keys, \((pkt, sk_t)\) tallier’s keys, \(H\) random oracle

\[ E_m = Enc_{pk_m}(g^x) \]

\[ \{f_0^V, f_1^V, \ldots, f_{N-1}^V\} \]

\[ \forall i \quad r_i \leftarrow Z_q \quad e_i = (Enc_{pk_m}(g_i^{f_i^V}))^{r_i} \cdot Enc_{pk_m}(g_{f_i^V}) \]

\[ \{e_0, e_1, \ldots, e_{N-1}\} \]

PC

Voter

Message

Vote Collector

\(x\)

\(r_i \leftarrow Z_q\)

\(e_i = (Enc_{pk_m}(g_i^{f_i^V}))^{r_i} \cdot Enc_{pk_m}(g_{f_i^V})\)

\(\{e_0, e_1, \ldots, e_{N-1}\}\)

\(\langle f_0^V, f_1^V, \ldots, f_{N-1}^V\rangle\)

\((pk_m, sk_m)\)

Tallier

Code Generator

before elections

after elections
The Proxy Oblivious Transfer E-Voting [HLV’10]

ElGamal PKC $\langle G, q, g \rangle$, $N$ candidates, ($pkm$, $skm$) messenger’s keys, ($pkt$, $skt$) tallier’s keys, $H$ random oracle

\[ E_m = Enc_{pkm}(g^x) \]

\[ \forall i \]
\[ r_i \leftarrow Z_q \]
\[ e_i = (Enc_{pkm}(\frac{g^i}{E_m}))^{r_i} \cdot Enc_{pkm}(g^{f_i^V}) \]

before elections

Code Generator

Voter

\[ g^{(i-x)r_i+f_i^V} \]

\[ \{ g^{f_0^V}, g^{f_1^V}, \ldots, g^{f_{N-1}^V} \} \]

after elections

Tallier

Message

Vote Collector

PC
ElGamal PKC $\langle G, q, g \rangle$, $N$ candidates, $(pkm, skm)$ messenger’s keys, $(pkt, skt)$ tallier’s keys, $H$ random oracle
ElGamal PKC \( \langle G, q, g \rangle \), \( N \) candidates, \((pkm, skm)\) messenger’s keys, \((pkt, skt)\) tallier’s keys, \( H \) random oracle

\[
E_m = Enc_{pkm}(g^x) \quad E_t = Enc_{pkt}(g^x)
\]

\( \{f^V_0, f^V_1, \ldots, f^V_{N-1}\} \)

\( \{e_0, e_1, \ldots, e_{N-1}\} \)

Vote Collector

PC

Voter

\( x \)

Code Generator

Anthi Orfanou (UoA)

Code Verification E-Voting

March 23, 2012
The Proxy Oblivious Transfer Security

- Malicious PC: Sees the vote (no privacy), cannot alter it without being detected (integrity)
- Malicious VC: Sees encryptions (privacy), cannot change signed votes (integrity)
- Malicious Messenger: Gets no information about the indexes (privacy)

⚠️ Weakness: Collaboration among Vote Collector and Messenger breaks the protocol’s privacy!

- Compare the value $a = g^{fx}$ of the messenger with the database of VC $i \leftrightarrow f_i$
Security Codes are constructed through 3 different pseudo-random functions
For each voter $V$ let:

- $f$ be a **global** encoding function $f : \text{Candidates} \rightarrow G$
- $s_V$ be a secret exponent
- $h_V$ be a pseudo-random function, selected from prf family $\mathcal{F}$
- $\text{Code}_V[cnd] = h_V((f(cnd))^{s_V})$
ElGamal PKC $\langle G, q, g \rangle$, $f$ encoding function, $F$ prf function family

$$s_V \leftarrow Z_q$$
$$h_V \leftarrow F$$

$$\forall \text{ candidate } i$$
$$\text{Code}_{V[i]} = h_V((f(cnd))^{s_V})$$

$$i \leftrightarrow \text{Code}_{V[i]}$$
Sharing the Keys

Key Generation:
- Values $a_1, a_2, a_3 \in \mathbb{Z}_q$ are selected such that $a_1 = a_2 + a_3 \mod q$
- Let $skt = a_1$, $skv = a_2$, $skm = a_3$ be the secret keys and $pkt = g^{a_1}$, $pkv = g^{a_2}$, $pkm = g^{a_3}$ be the public keys
- $pkt = g^{a_1} = g^{a_2} g^{a_3} = pkm \cdot pkv$

+ No need to encrypt the submitted votes with different keys
- A Coalition of the VC and the Messenger can reconstruct the decryption key
The Pseudo-random E-Voting

ElGamal PKC \( \langle G, q, g \rangle \), \( f \) encoding function, \( a_1 = a_2 + a_3 \mod q \), \( h_V \) prf function, \( s_V \) secret exponent
The Pseudo-random E-Voting [Gjò ’10–’11]

ElGamal PKC \((G, q, g)\), \(f\) encoding function, \(a_1 = a_2 + a_3 \mod q\), \(h_V\) prf function, \(s_V\) secret exponent
The Pseudo-random E-Voting

ElGamal PKC $\langle G, q, g \rangle$, $f$ encoding function, $a_1 = a_2 + a_3 \mod q$, $h_V$ prf function, $s_V$ secret exponent

$e = \text{Enc}_{pkt}(f(x))$

$\left( g^x, g^{a_1 r} f(x) \right)$

Vote Collector

$a_2$

$a_1$

after elections

$a_3$

before elections

Tallier

Code Generator

Voter

PC

$X$
The Pseudo-random E-Voting [Gjò ’10-’11]

ElGamal PKC \(\langle G, q, g \rangle\), \(f\) encoding function, \(a_1 = a_2 + a_3 \mod q\), \(h_v\) prf function, \(s_v\) secret exponent

\[
e = Enc_{pkv}(f(x)) \quad (g^r, g^{a_1r}f(x))
\]

\[
\hat{e} = Dec_{skv}(e) \quad (g^r, \frac{g^{a_1r}f(x)}{(g^r)^a_2})
\]

\[
Enc_{pkm}(f(x))
\]
ElGamal PKC $\langle G, q, g \rangle$, $f$ encoding function, $a_1 = a_2 + a_3 \mod q$, $h_V$ prf function, $s_V$ secret exponent

\begin{align*}
e &= \text{Enc}_\text{pkt}(f(x)) \left(g^r, g^{a_1r}f(x)\right) \\
\hat{e} &= \text{Dec}_\text{skv}(e) \left(g^r, \frac{g^{a_1r}f(x)}{(g^r)^{a_2}}\right) \\
\text{Enc}_\text{pkm}(f(x)) &= e^* = (\hat{e})^{s_V} \\
\text{Enc}_\text{pkm}(f(x))^{s_V}
\end{align*}
The Pseudo-random E-Voting [Gjò ’10-’11]

ElGamal PKC $(G, q, g)$, $f$ encoding function, $a_1 = a_2 + a_3$ mod $q$, $h_V$ prf function, $s_V$ secret exponent

$e = Enc_{pkt}(f(x))$,

$e = Enc_{pkt}(f(x)) = (g^r, g^{a_1r} f(x))$

$\hat{e} = Dec_{sk_V}(e)$

$\hat{e} = Dec_{sk_V}(e) = (\hat{e})^{s_V}$

$\hat{e} = Dec_{sk_V}(e) = (\hat{e})^{s_V}$

Vote Collector

Voter

PC

$x$

before elections

Code Generator

after elections

Tallier

Messenger

$h_V$

$a_1$

$a_2$

$a_3$
The Pseudo-random E-Voting [Gjò ’10-’11]

ElGamal PKC $\langle G, q, g \rangle$, $f$ encoding function, $a_1 = a_2 + a_3 \mod q$, $h_V$ prf function, $s_V$ secret exponent

$e = \text{Enc}_{pkr}(f(x))$

$\langle g^r, g^{a_1r} f(x) \rangle$

$\hat{e} = \text{Dec}_{skv}(e)$

$e^* = (\hat{e})^{s_V}$

$a^* = f(x)^{s_V}$

$\text{Code} = h_V(a^*)$

Vote Collector

PC

Voter

before elections

Code Generator

Tallier

after elections

encapsulation

Messenger

$\text{Enc}_{pkm}((f(x))^{s_V})$

Anthi Orfanou (UoA)
The Pseudo-random E-Voting [Gjø '10-'11]

ElGamal PKC $\langle G, q, g \rangle$, $f$ encoding function, $a_1 = a_2 + a_3 \mod q$, $h_V$ prf function, $s_V$ secret exponent

\[ e = \text{Enc}_{pkt}(f(x)) \]
\[ (g^r, g^{a_1} f(x)) \]
\[ s_V \]
\[ a_2 \]

Vote Collector

$e = \text{Enc}_{pkt}(f(x))$

$\text{Enc}_{pkm}((f(x))^{s_V})$

ZKP correct computation

Messenger

$h_V$

$tallier$

before elections

Code Generator

Voter

$\text{Code} = h_V(a^*)$

$\text{Enc}_{pkt}(f(x))$

$a_1$

$a_3$
The Pseudo-random Superposition Security

- Same guarantees with the POT approach, for each individual entity
- Collaboration among Vote Collector and Messenger breaks the protocol’s privacy!
  - Shared key reconstruction
  - Separate the keys? Given $s_V$ the Messenger can break privacy by comparisons: $\forall c^* \ f^{s_V}(c^*)$
The ballot format

- A ballot has multiple options: $k_{max}$ out of $n$ candidates ($x_1, \ldots, x_{k_{max}}$) (possibly padded)
- Order not important, no write-in option
- **Encrypted ballot:** tuple of multiple ElGamal cipher-texts of fixed size $c = (Enc(f(x_1)), Enc(f(x_2)), \ldots, Enc(f(x_{k_{max}})))$
- **Code verification:** Different security code for each cipher-text
- **Decryption:** Depends on the encoding function used
Choosing the encoding function

How to choose $f$?

- Random injection $f : \text{Candidates} \to G$
- Decryption: Each option separately
- Special Structure $f : \text{Candidates} \to \text{SpecialSet}$
- Decryption: compress and recover cipher-texts efficiently!
A new approach

- $p, q$ primes, $p = 2q + 1$ and $G$ be the cyclic group of quadratic residues of $\mathbb{F}_p$
- $\mathcal{L}$ the set of the smallest primes $\{l_1, l_2, ..., l_L\}$, $l_i \in G$
- Factoring of products of small primes can be solved efficiently

**Definition**

**Prime DDH** Given $(l_1, ..., l_L) \in G^n$ decide if $(x_1, ..., x_L) \in G^n$ was sampled uniformly from the powers $\{l_1^s, ..., l_L^s\}$ or uniformly from $G^L$.

⚠️ Unknown hardness! Only weaker, special cases have been proved equivalent to the DDH problem
A new approach

- $f$: random injection $\text{Candidates} \rightarrow \mathcal{L}$

**Vote Collector**
- Input $c = (\text{Enc}_p\text{kt}(f(x_1)), ..., \text{Enc}_p\text{kt}(f(x_{k_{max}}))$
- Compression: $\tilde{c} = \prod_{i=1}^{k_{max}} \text{Enc}_p\text{kt}(f(x_i))$

**Tallier**
- Decrypts $\tilde{c}$: $\prod_{i=1}^{k_{max}} f(x_i) = l_1...l_{k_{max}}$
- Recovers the votes $l_1, ..., l_{k_{max}}$
Open Problems

- Preserve privacy against Vote Collector and Messenger Coalitions
- Preserve integrity against PC and Messenger Coalitions
- Study further the conjectured hardness of the Prime DDH problem
- Improve the current protocol’s performance (encryption, ZKPs)


The End...

Thank you!