Secure E-Voting

Li Chen

lichen.xd at gmail.com

Xidian University

May 15, 2014

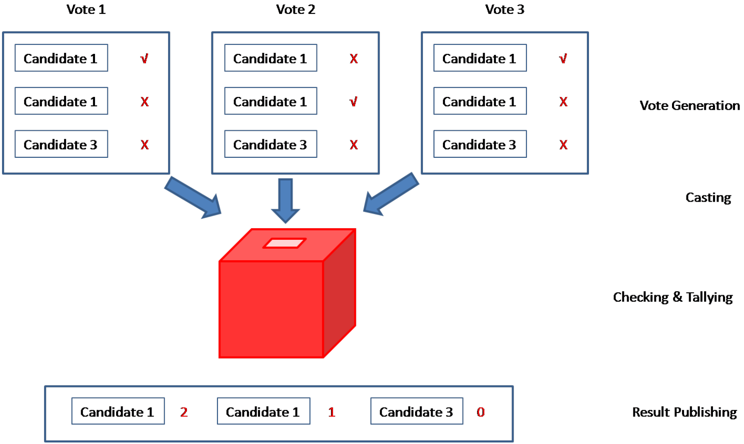
Outline

Designed-Verifier Re-encryption Proof

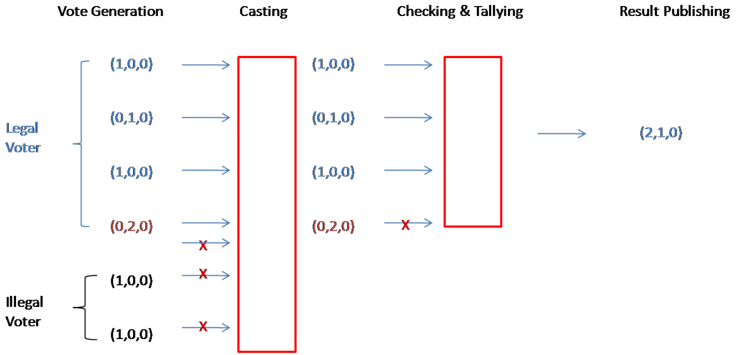
References

[1] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *Advances in CryptologyąłEUROCRYPT 2000*, pages 539–556. Springer, 2000.

1 Traditional Voting



2 E-Voting



All these steps are completed via a network.

3 Security requirements for E-Voting

- Authentication: only legal voters can cast a vote.
- Privacy (ballot secrecy): vote content cannot be related to voter identities.
- universal verifiability: correctness of the voting process can be checked.
- Receipt-freeness: voter can not prove the vote content to others.

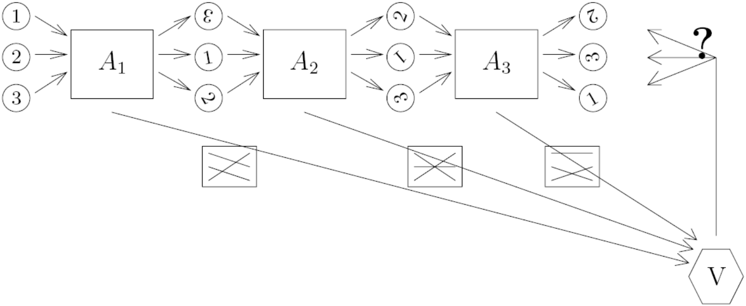
4 E-Voting Scheme I [1]

4.1 Voting Model

Entities

- Authorities $A_1, \cdots, A_N$.
- Voters $U_1, \cdots, U_M$.
- Votes $v_1, \cdots, v_L$.
- Ballot Board.

E-Voting Scheme I



4.2 Scheme

Vote Generation

Let v_i be a valid vote, $e_i^{(0)}$ be the corresponding *standard encryption*. In turn, for each $A_k(k = 1, \cdots, N)$:

1. A_k picks the encrypted valid votes list $e_1^{(k-1)}, \cdots, e_L^{(k-1)}$, *shuffles* it randomly, and hand it to the next authority.

   Shuffle means to re-encrypt each $e_i^{(k-1)}$, and randomly permute the order of the list. Precisely, randomly choose a permutation $\pi_k : \{1, \cdots, L\} \to \{1, \cdots, L\}$, and computes $e_i^{(k)} \leftarrow e_{\pi(i)}^{(k-1)} \bigotimes E(0, r_i^{(k-1)}), r_i^{(k-1)} \stackrel{\$}{\leftarrow}$.

2. A_k publicly proves that she honestly shuffled, namely by proving for each i there exists a re-encryption of $e_i^{(k-1)}$ in the list $e_1^{(k)}, \cdots, e_L^{(k)}$, without revealing which.

3. A_k secretly conveys to the voters the permutation π_k she used, with a privately proof of its correctness.

4. If the voter does not accept the proof, he publicly complains about the authority. If he does so, we set $e_i^{(k)} \leftarrow e_i^{(k-1)}, i = 1, \cdots, L$, i.e. the shuffling of this authority is ignored. Voter may at most complain against $N - t$ authorities.

Vote Casting

The voter drives the position i of the encrypted vote $e_i^{(N)}$ of his choice, and publicly announces it.

Tallying

Those encrypted vote are summed to obtain an encryption $E(T)$ of the sum T of the votes. Authorities decrypt $E(T)$ to output T, with a proof of its correctness.

4.3 Component

Additive Homomorphic ELGamal Encryption

$$E(v) = (g^{\alpha}, \gamma^{v} h^{\alpha})$$

where g, γ are two generator of a commutative group G, with order $|G| = q$, a larger prime. $h = g^{z}$, z is the secret key, (g, γ, h) is the public key. A standard encryption of v is denoted as $e_v^{(0)} = (1, \gamma^{v})$.

- Homomorphic Property: Let $e_1 = (x_1, y_1), e_2 = (x_2, y_2)$, the addition is defined as $e_1 \bigoplus e_2 = (x_1 x_2, y_1 y_2)$.
- Re-encryptability: Let $e = (x, y)$ it's re-encryption is $e' = (x', y') = (g^{\xi} x, h^{\xi} y)$.

- Verifiable Decryption: To decrypt T from $e = (x, y)$, the authorities first compute, reveal and prove $\hat{x} = x^z$. This can be obtain by having each authority A_i compute $\hat{x_i} = x_i^{z_i}$, where z_i is A_i's share of the secret key z, and then compute $\hat{x}$ from $\hat{x_i}$. Once $\hat{x}$ is obtain with proof of correctness, one can compute

$$\frac{y}{\hat{x}} = \frac{\gamma^T \cdot h^\alpha}{g^{\alpha z}} = \gamma^T$$

1-out-of-L Re-encryption Proof

Suppose a prover want to prove that for an encrypted vote (x, y), there exist a re-encryption in the L encrypted votes list $(x_1, y_1), \cdots, (x_L, y_L)$. Assume that (x_t, y_t) is a re-encryption of (x, y), and the witness is ξ, i.e. $(x_t, y_t) = (g^\xi x, h^\xi y)$.

1. The prover selects $d_1, \cdots, d_L$ and $r_1, \cdots, r_L$ at random, record $w = \xi d_t + r_t \pmod q$, and computes
$$a_i = \Big(\frac{x_i}{x}\Big)^{d_i} \cdot g^{r_i}, \text{ and } b_i = \Big(\frac{y_i}{y}\Big)^{d_i} \cdot g^{r_i}, \text{ (for } i = 1, \cdots, L).$$
and sends a_i, b_i to verifier. (These values commit the prover to d_i and r_i for all $i = 1, \cdots, L$ except for $i = t$.)
2. The verifier picks a random challenge $c \xleftarrow{\$} \mathbb{Z}_q$, and sends it to the prover.
3. The prover modifies d_t, s.t. $c = \sum_{i=1}^{L} d_i \pmod q$, modifies r_t to, s.t. $w = \xi d_t + r_t \pmod q$, and sends $d_1, \cdots, d_L$ and $r_1, \cdots, r_L$ to the verifier.

4. the verifier tests wether

$$c \stackrel{?}{=} \sum_{i=1}^{L} d_i \pmod q$$

$$a_i \stackrel{?}{=} \left(\frac{x_i}{x}\right)^{d_i} \cdot g^{r_i}, \text{ (for } i = 1, \cdots, L).$$

$$b_i \stackrel{?}{=} \left(\frac{y_i}{y}\right)^{d_i} \cdot g^{r_i}, \text{ (for } i = 1, \cdots, L).$$

Note: $a_t = g^{\xi d_t + r_t}$ and $b_t = h^{\xi d_t + r_t}$.

Designed-Verifier Re-encryption Proof

Suppose the prover want to privately prove that (x', y') is a re-encryption of (x, y), where ξ is the witness, i.e. $(x', y') = (g^{\xi} x, h^{\xi} y)$. The voter's secret key is denoted as z_v and the public key is given by $h_v = g^{z_v}$.

1. The prover selects d, w and r at random, computes

   $$a = g^d, \ b = h^d, \text{ and } s = g^w h_v^r.$$

   and sends it to the verifier. These values commit the power to d, w and r. However, s is a chameleon commitment for w and r, and the verifier can use his knowledge of z_v to open s to arbitrary values w' and r' satisfying $w' + z_v r' = w + z_v r$.

2. the verifier picks a random challenge $c \xleftarrow{\$} \mathbb{Z}_q$ and sends it to the prover.

3. prover computes $u = d + \xi(c + w)$ and sends w, r, u to the verifier.

4. The verifier tests whether

$$
\begin{aligned}
s &\stackrel{?}{=} g^w h_v^r \\
g^u &\stackrel{?}{=} \left(\frac{x'}{x}\right)^{c+w} \cdot a \\
h^u &\stackrel{?}{=} \left(\frac{y'}{y}\right)^{c+w} \cdot b
\end{aligned}
$$

Thanks! & Questions?