6.915 Computer and Network Security	Fall Term, 1995
Lecture 24 : December 5, 1995	
Lecturer: Rosario Gennaro	*Scribe: Navneet Singh*

1 Overview

The topic of today's lecture is **Electronic Election Schemes**. When dealing with such protocols, one should consider the following issues:

- Trusted Government
- Untrusted Government
- Partially Trusted Government
- Coercion

In general, we would like our Election Protocols to have the following properties:

1. Only authorized voters can vote.
2. No one can vote more than once.
3. Secrecy of votes is maintained.
4. No one can duplicate anyone else's vote.
5. The tally is computed correctly.
6. Anybody should be able to check 5.

2 The Merritt Election Protocol

Consider the following scheme by Michael Merritt in which Alice (A), Bob (B), Carl (C), and David (D) are voters:

In order to cast a vote[1], each voter chooses a random number r and computes,

$$E_A(E_B(E_C(E_D(v,r)))) = y \tag{1}$$

[1] We assume binary votes for now

The voter then chooses random numbers r_1, r_2, r_3, r_4, and computes

$$E_A(E_B(E_C(E_D(y, r_1), r_2), r_3), r_4) \tag{2}$$

At this point, the verification cycle begins. It consists of two rounds of decryption in the order $A \longrightarrow B \longrightarrow C \longrightarrow D$.

The biggest and most obvious drawback of the above protocol is that it gets all voters involved. In a variation of this scheme, A, B, C and D are government agencies. However, this would not be a fault-tolerant scheme at all. Even if one of the government agencies corrupts[2], the entire system falls apart. For this reason, we consider the following scheme.

3 A fault-tolerant Election Protocol

3.1 Homomorphic Encryption Algorithms

Since the protocol presented in this section assumes that the encryption algorithm used is homomorphic, we first discuss the properties of such methods and modify the ElGamal algorithm so that it is homomorphic.

A $(+, \times)$-homomorphic encryption algorithm E has the property $E(X + Y) = E(X)E(Y)$. RSA and ElGamal encryptions are $(\times, \times)$-homomorphic since they both have the property $E(XY) = E(X)E(Y)$. Let us show a way to modify ElGamal to a $(+, \times)$ scheme.

Consider the regular ElGamal algorithm in which x is the secret key and $y = g^x \bmod p$ is the public key. Then,

$$E_r(m) \ = \ (a = g^r, \ b = y^r \cdot m). \tag{3}$$

for a randomly chosen r.

In order to make it $(+, \times)$-homomorphic we choose an element $h \in Z_p$ of order ℓ (that is $h^i \neq 1$ if $1 \leq i < \ell$ and $h^\ell = 1$). If we assume the message space to be $\{0, \ldots, M\}$ for $M < \ell$ we can modify the scheme as follows. The modified ElGamal, MEG, is

$$MEG_r(m) \ = \ (a = g^r, \ b = y^r \cdot h^m). \tag{4}$$

[2]we take corruption to mean any kind of failure

In order to see that MEG is homomorphic, let $MEG_{r_1}(m_1) = (a_1,\ b_1)$ and $MEG_{r_2}(m_2) = (a_2,\ b_2)$, and consider the ciphertext (a_1a_2, b_1b_2). We see that

$$a_1a_2 = g^{r_1}g^{r_2} = g^{r_1+r_2}, \text{ and}$$

$$b_1b_2 = y^{r_1}h^{m_1}y^{r_2}h^{m_2} = y^{r_1+r_2} \cdot h^{m_1+m_2}$$

Therefore, $MEG(X+Y) = MEG(X)MEG(Y)$.

In order to decrypt however we have to compute discrete log in base h. However if M is not too big we can afford to do this with a exhaustive search in M steps. Also there are more efficient methopds that allow this computation in only $\sqrt{M}$.

So it is crucial to keep the message space small. In our case as we will see shortly M is at most the number of voters.

3.2 The Protocol

Let there be m voters, $V_1, ..., V_m$ and n centers or government agencies, $C_1, ..., C_n$. We give two versions of the protocol – one that assumes that all centers are trusted parties (Trusted version), and then relax this condition to look at the case when at most t are corrupted (Threshold version). Furthermore, we divide the protocol into two parts – vote casting and tally computation.

Vote Casting

In both the Trusted and Threshold versions, votes are cast as follows:

Assuming the votes are either 0 or 1, each voter prepares ballots $B_1, ..., B_{100}$, where

$$B_i = (MEG(v_i),\ MEG(\bar{v_i})) \tag{5}$$

where $v_i = random(0,1)$ and $\bar{v_i}$ is its complement. The government then asks the voter to reveal 99 of these ballots (notice the similarity to blinding coins). At the end the government is convinced that the unopened ballot contains regular votes and let the voter choose on the unopened ballot components as his/her vote.

Tally Computation

Let x be the private key of the vote casting encryption scheme and $(p, g, y = g^x)$ the corresponding public key.

Assume for a moment that none of the centers are corrupted and that there is a trusted party that acts only at the beginning once for all.

The trusted party gives center C_i a share x_i of the secret key x such that

$$x_1 + x_2 + \ldots + x_n = x$$

and publishes $y_1 = g^{x_1}, \ldots, y_n = g^{x_n}$

Remember that each voter casted $MEG(v_i) = (a_i, b_i)$. If we let $a = \prod_{i=1}^m a_i$ and $b = \prod_{i=1}^m b_i$ then because of the homomorphic properties of MEG we know that (a, b) is an encryption of the tally. So the tally is going to be

$$tally = Dlog_h \frac{b}{a^x}$$

so the only opeeration[3] in which the secret key is involved is computing a^x.

Each center C_i publishes $\alpha_i = a^{x_i}$. Clearly $a^x = \prod_{i=1}^n \alpha_i$. However we are in trouble if the center publishes $\alpha_i \neq a^{x_i}$.

In order to verify that center C_i is not publishing a false α_i, a verifier (all the other centers for example) requires him to produce a zero-knowledge proof that

$$Dlog_g(y_i) = Dlog_a(\alpha_i)$$

in the following manner.

Center	$\longrightarrow a^r g^r \longrightarrow$	Verifier
Center	$\longleftarrow b = 0, 1 \longleftarrow$	Verifier
Center	$\longrightarrow$r(0), $r + x_i(1) \longrightarrow$	Verifier

The above is repeated several times. This is an extension of the zk proof of knowledge of discrete logs that we saw in problem set 3.

[3]Notice also as discrete logs in base h are computed on numbers at most the size of the tally and so smaller than the number of voters

Notice that if even one center is corrupted, although we can detect this using the zk proof, we will have to restart the whole election from scratch.

Let us assume now that at most $t = \frac{n}{2} - 1$ of the centers are corrupted, and that we still have a trusted party. Let f be a polynomial of degree $t = n/2$

$$f = \gamma_0 + \gamma_1 x_1 + ... + \gamma_t x_t \tag{6}$$

where $f(0) = \gamma_0 = x$. The trusted party now distributes $x_i = f(i)$ to center C_i. The rest of the protocol is the same as the Trusted version, except for the computation of a^x.

In order to compute x we need only $t+1$ shares (instead of all n as before). Indeed given $t+1$ shares, say $x_1, \ldots, x_{t+1}$, one can interpolate the polynomial f as follows.

Let $\vec{\gamma} = [\gamma_0, \gamma_1, ..., \gamma_t]$, $\vec{x} = [x_1, x_2, ..., x_{t+1}]$, and A be the matrix whose ith row is $[1, i, i^2, ..., i^t]$. We know that

$$A\vec{\gamma}^T = \vec{x}^T$$

Since A is non-singular,

$$\vec{\gamma}^T = A^{-1}\vec{x}^T$$

. Now x is simply γ_0, meaning that if $A_1, \ldots, A_{t+1}$ are the elements in the first row of A^{-1} we have that

$$x = A_1 x_1 + A_2 x_2 + \ldots + A_{t+1} x_{t+1}$$

.

Now notice that we don't want to reconstruct x, otherwise we would be able to learn everybody's vote. Instead we want to reconstruct only a^x. But this is easy from the values $\alpha_i = a^{x_i}$ that are published by each center (after being checked for correctness with the zk proof). We have that

$$a^x = (a^{x_1})^{A_1} \cdot (a^{x_2})^{A_2} \cdot \ldots (a^{x_{t+1}})^{A_{t+1}}$$

Both the Trusted and Threshold versions are disussed in detail in the readings for this Lecture in the paper, *A receipt-free election scheme tolerating a coercer* by Rosario Gennaro. Gennaro's paper also discusses the case when there is no trusted party.