6.857 Computer and Network Security	October 11, 2001
Lecture Notes 10 : Mix-Nets	
Lecturer: Ron Rivest	*Scribe: Brunsman/Leon/Lin/Cheung*

1 Outline

- Zero-Knowledge proofs
  - Write-in votes with homomorphic encryption
  - General result
  - Design verifier
- Secret sharing
- Mix-Nets

2 Zero-Knowledge Proofs

The Paillier scheme has the additive homomorphic property such that $E(m_1) \cdot E(m_2) = E(m_1 + m_2)$

2.1 Original Paillier Voting Scheme

In the original Paillier voting scheme, the voter submits $E(m_1)$ and a NIZK (non-interactive zero-knowledge) proof that m_1 is proper (e.g., m_1 is either 0 or 1).

A Voter V casts his vote in an election with candidates A, B, and C. The election official records his vote in a table as shown below.

	A	B	C
V	$E(m_A)$	$E(m_B)$	$E(m_C)$
.	.	.	.
.	.	.	.
.	.	.	.
	$E(total_A)$	$E(total_B)$	$E(total_C)$

Multiplying all the $E(m_A)$'s gives $E(total_A)$ because of the additive homomorphic property.

2.2 Paillier Voting Scheme with Write-Ins

Recently, Moti Yung devised a voting scheme that handles write-ins. The election official records the votes as shown below.

	A	B	C	row sum	write-in
V	$E(m_A)$	$E(m_B)$	$E(m_C)$	$E(rowsum)$	$E(name)$
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

where $E(rowsum) = E(m_A + m_B + m_C)$

Question: What happens in a place like Cambridge where elections allow preferential ordering of write-ins?
Answer: This scheme only works with one write-in and does not support preferential ordering. Adapting this scheme to support multiple write-ins and preferential ordering might be an interesting term project.

There are two steps for this new voting scheme:

Step 1. Voter needs to prove that either $rowsum = 0$ or $name = 0$ (or both)

Question: How can you prove the complex statment: $rowsum = 0$ or $name = 0$?
Answer: There is a theorem, although the proof is not always efficient or practical.

Theorem 1 *Anything that you can prove (e.g., by revealing secrets), you can prove in zero-knowledge (without revealing secrets).*

Step 2. If write-ins could affect the results of the election, then the EO (election official) needs to:

- shuffle write-ins (by permuting randomly)
- decrypt write-ins
- adjust tallies

Question: Can we shuffle the rowsum and write-in as a pair, thus ensuring voter privacy, and then open the ballots to avoid using a complex ZK proof?
Answer: Yes, but Moti did not choose to do so.

Question: Do you prove that $rowsum$, M_a, M_b, and M_c are either 0 or 1 before Step 1?
Answer: Yes.

Example 1 *Software for paid customers only*

Given that M is some expensive or important piece of software, the creator of M, GigaSoft, may want to give M a unique signature for each of its customers in order to validate the software. GigaSoft provides to Company A, its customer, the ordered-triple (M, commit(x), proof), where the proof proves that x is either GigaSoft's special signature on M or x is the private key of Company A. In other words, for Company A, the proof, together with commit(x), proves that the software is authentic because nobody but Company A knows its secret key; the only alternative is that the signature must be that of GigaSoft. Company B, however, will not believe the software is authentic because the proof could be from GigaSoft or it could be from Company A.

Question: Company B can verify if the proof is for A's secret key, right?
Answer: No, remember that the commitment reveals nothing about its contents.

Question: When should we use ZK proofs? For software piracy issues?
Answer: This particular example is against software piracy because our assumption is that B rejects the software unless it is signed by Gigasoft.

3 Secret Sharing and Threshold Cryptography

Problem: In the past lectures, the role of the Administrator (A) has had complete power. The Administrator holds all the secrets and may abuse his/her power at will.

Solution: Split up the role of A among several (mutually suspicious) parties

$A \Rightarrow A_1, A_2, A_3, \ldots, A_n$

We require a subset t (where $t \leq n$) of the n parties to cooperate with each other in order to reconstruct the secret key. The secret key S is split into n shares, such that any t of the n administrators may reconstruct S. Note that administrators should not be able to obtain any information about S with fewer than t administrators participating.

3.1 A Secret Sharing Implementation Idea

So how do we implement secret sharing?

Theorem 2 *Given points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where the x_i's are distinct*

$y = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$

there is a unique curve of degree $n-1$.

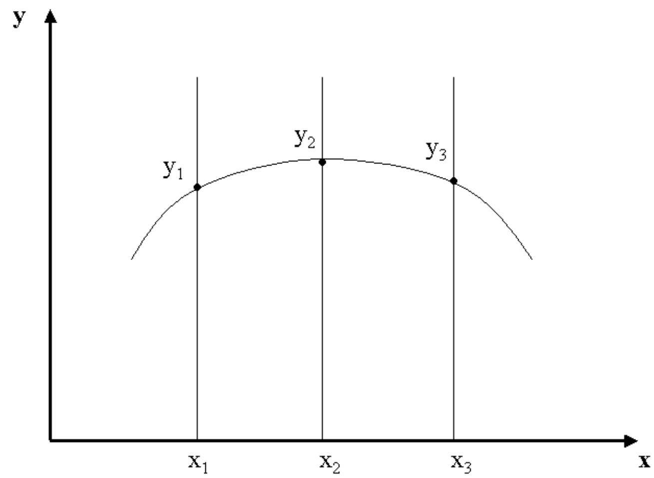

Figure 1: A quadratic curve determined by three points

Example 2 *Three unique points determine a quadratic curve* $y = a_0 + a_1x + a_2x^2$

Setup: $y = S + a_1x + a_2x^2$ (where S is the secret)
Party $i, 1 \leq i \leq n$ gets point $s_i = (i, f(i))$

Here, we see that any three parties may reconstruct the curve and then determine the secret, S.

Now, all we have to do is generalize to t unique points instead of using three unique points.

3.2 Threshold Cryptography and Voting

The concept of distributing a secret key S among several administrators is referred to as threshold cryptography because S may not be discovered unless at least a threshold number t of the n administrators are cooperating. Research has been done in order to allow S to be used by t administrators without ever having reconstructed it.[1] In this way, there is no trusted third party needed to reconstruct the key, and administrators may not discover S at the time of reconstruction.

Professor Rivest noted that one problem with threshold cryptography is that even if S is destroyed after it has been distributed among the administrators, it must have existed at a moment in time, and may have been discovered or seen (perhaps by a "dealer").

We can apply threshold cryptography to voting schemes. As we see in Figure 2, both public and private keys are generated, and the private key is distributed among n administrators. Once all the votes are in, the encrypted tallies are inserted into a central system and decrypted by S assuming that at least t administrators are in cooperation.

[1] http://citeseer.nj.nec.com/38096.html

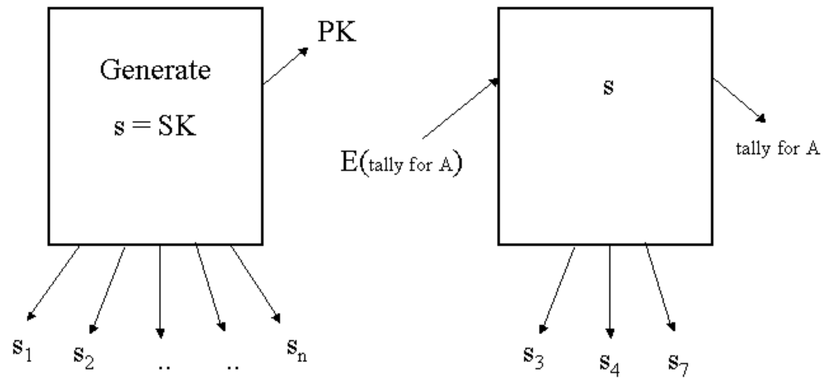

Figure 2: Threshold cryptography as applied to a voting scheme

4 Mix Nets (David Chaum)

The Mix Net voting scheme is the third voting scheme we have looked at in this class (the others were FOO and homomorphic). Recall that both the FOO and homomorphic voting schemes required anonymous channels to guarantee voter privacy.

The Mix Net scheme uses a series of permuting-devices called *mix-servers* that output a random permutation of their inputs. By using several such mix-servers in series, no single person can unscramble the order of the outputs to reveal the order of the inputs. This brings us one step closer to voter privacy.

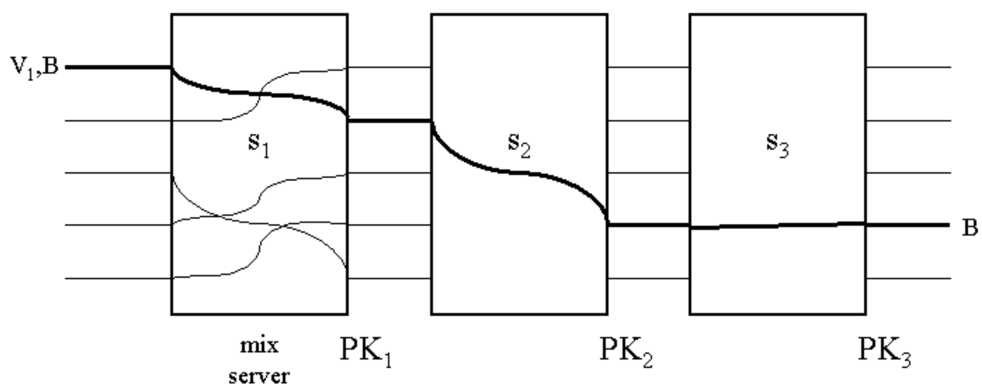

Figure 3: A Mix Next composed of three mix-servers

If the outputs are the same as the inputs, however, no amount of permutation will maintain voter anonymity. As such, we must either encrypt or decrypt the votes within the mix-servers so that the output appears different from the input.

David Chaum's approach decrypts the input within each mix-server:

$$\hat{\mathrm{B}} = PK_1(PK_2(PK_3(B)))$$

where $PK_a(X)$ is defined as the encryption of message X with public key PK_a. The voter submits $\hat{\mathrm{B}}$ in this scheme.

A variation of this approach uses each mix-server to encrypt its input, and subsequently decrypts the final result at the end of the mix-server series. The encryption process manipulates the ciphertext input such that the contents remain the same but the appearance is different.

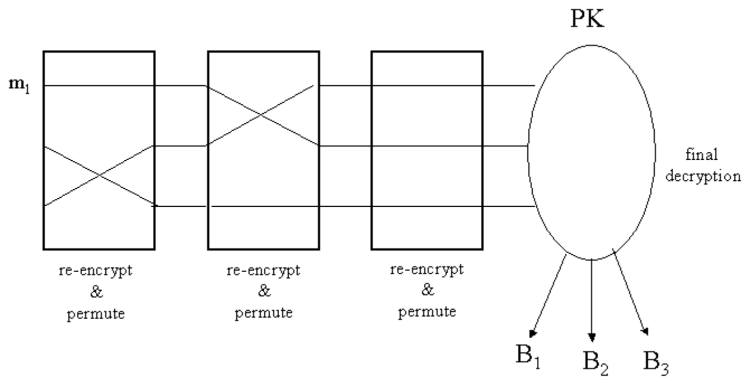

Figure 4: A variation of Chaum's Mix Next

In either approach, El Gamal can be used for encryption and decryption:

$$(PK, SK) = (y, x)$$
$$y = g^x \bmod p$$

Where g is a generator and p is a prime. The output of encryption is:

$$(g^r, m * y^r)$$

where r is a random integer and m is the message (in this case, the vote). Suppose, then, that a mix-server receives as input an encrypted ballot. The mix-server does not know the value of the secret key x, so to re-encrypt the input we simply choose a new random integer r' and multiply:

$$(g^r * g^{r'}, y^r * y^{r'} * m) = (g^{r+r'}, m * y^{r+r'})$$

Question: How do we decrypt the final output?
Answer: The same as we normally decrpyt: we simply treat $r + r'$ as the new r.

What are the risks to the Mix Net Voting Scheme? Substition attacks: we need proofs from each mix-server that the input votes are the same as the output votes. We also want these proofs to be ZK proofs to protect the election integrity.

What are the advantages of the Mix Net voting scheme? Voter anonymity: by permuting and encrypting the inputs, the output of a mix-server maintains voter privacy.

For additional reading about Mix Nets, see
http://www.inf.tu-dresden.de/~hf2/publ/2000/BeFK2000cfp2000.

For additional reading about David Chaum, see
http://www.chaum.com/welcome.html.