6.897: Advanced Topics in Cryptography Apr 9, 2004

Lecture 18: Mix-net Voting Systems

Lecturer: Ron Rivest *Scribed by: Yael Tauman Kalai*

1 Introduction

In the previous lecture, we defined the notion of an electronic voting system, and specified the requirements from such a system. In particular, we required an electronic voting system to be verifiable and robust. Loosely speaking, a voting system is said to be verifiable if any individual can verify that his vote was counted. A voting system is said to be robust if there does not exist any small set of servers that can disrupt the election.

The voting systems that appear in the literature can be roughly categorized into three groups: one based on mix-nets, one based on homomorphic encryptions, and one based on blind signatures. In this lecture we concentrate on mix-net protocols. We describe two types of mix-net protocols: *decryption mix-nets* and *re-encryption mix-nets*.

The general structure of mix-nets was illustrated in the previous lecture. They begin with an initial encryption phase E, whose outputs are posted on a bulletin board, in order to achieve verifiability. The initial encryption phase is followed by several mix phases $mix_1, \ldots, mix_k$. The reason we need *several* of them is to achieve robustness. In decryption mix-nets, the mix phases mix and partially decrypt, whereas in re-encryption mix-nets, the mix phases mix and re-encrypt. In re-encryption mix-nets a final decryption phase D is added.

2 Decryption Mix-Net

A decryption mix-net does not have a final decryption phase. Rather, the initial encryption phase E encrypts its inputs by applying a concatenation of k encryption operations to each input; each mix peels off one of these encryptions by applying a corresponding decryption algorithm; it then mixes all its decrypted inputs by applying a secret random permutation to them. Thus, this scheme has the structure of an onion; E builds the onion, and each mix peels off one layer of the onion.

More specifically, each mix has its own pair of keys. We denote the keys of mix_i by (SK_i, PK_i). mix_i decrypts its inputs using its keys (SK_i, PK_i); it then secretly permutes all its decrypted inputs.

The initial encryption E has the public keys of all the mixes $(PK_1, \ldots, PK_k)$; it encrypts each input by first encrypting it with PK_k, then encrypting the result with PK_{k-1}, then encrypting the result with PK_{k-2}, and so on. Thus, if we denote the ballots by $B_1, \ldots, B_n$, then for each $i = 1, \ldots, n$,

$$C_i = E(B_i) = E(PK_1 \ldots E(PK_{k-1}, E(PK_k, B_i)) \ldots).$$

There are some issues that need to be addressed:

1. Note that secure encryption schemes do not hide the length of the plain texts. Since the outputs of E appear publicly on a bulletin board, in order to preserve secrecy, we must require all the cipher-texts to be of the same length.

2. Note that that mix_k (the last mix) generates the final output of the vote. Thus, if he doesn't like the output he may abort. One way of preventing mix_k from aborting, is by making his secret shared. This arouses further issues, such as key management.

3. It seems like semantic security is enough, assuming the encrypted ballots are publicized only after all the voters have voted. Otherwise, we need a stronger security notion, such as CCA2 security, in order to achieve non-malleability.

4. The above protocol, as described, is neither verifiable nor robust. In order to achieve these two desired properties, we need to add some ingredients to the protocol. These ingredients will be added following the description of re-encryption mix-nets.

3 Re-encryption Mix-nets

As opposed to a mix phase in a decryption mix-net, whose role is both to mix and to partially decrypt, the role of a mix phase in a re-encryption mix-net is only to mix. Note, however, that a mix which merely scrambles the inputs is not good enough. This is so, since by merely scrambling, the resulting set of ciphertexts does not change, and thus for each resulting ciphertext it is easy to recover the voter associated with it. Thus, an extra operation is needed in order to mix in an unrecoverable way. In a re-encryption mix-net, the extra operation added to each mix phase is a re-encryption operation.

In total, a re-encryption mix-net consists of an initial encryption phase E, several mix phases $mix_i, \ldots, mix_k$, who mix by scrambling and re-encrypting, and a final decryption phase D. Typically, the encryption scheme used in a re-encryption mix-net is the El-Gamal encryption scheme, which has a nice re-encryption property. In what follows, we describe in more detail an El-Gamal based re-encryption mix-net.

3.1 El-Gamal Based Re-encryption Mix-nets

Recall that in the El-Gamal encryption scheme, an encryption of a message m, with respect to a public key (p, g, y), consists of a pair (g^r, my^r), where all the operations are done modulo p, and $r \in_R \mathbb{Z}_q$ where q is a large prime dividing $p - 1$, where g is a generator of the subgroup of elements whose order divides q, and m is in this subgroup. The secret key corresponding to (p, g, y) is x such that $g^x = y(mod\ p)$.

The El-Gamal encryption scheme has the following nice re-encrypting property: any encrypted message $(a, b) = (g^r, my^r)$ can be re-encrypted by choosing a random $s \in_R \mathbb{Z}_q$ and computing $(ag^s, by^s) = (g^{r+s}, my^{r+s})$. Note that this re-encrypting operation results with a random ciphertext for the same message m.

We are now ready to define the El-Gamal based re-encryption mix net:

1. An El-Gamal public-key (p, g, y) is generated (in some distributed manner).

2. The initial encryption phase E simply encrypts all the ballots $B_1, \ldots, B_n$ by applying the El-Gamal encryption algorithm with the public-key (p, g, y). It then posts all the resulting ciphertexts $(C_{1,0}, \ldots, C_{n,0})$ on a bulletin board.
3. The i'th mix phase, on input a set of ciphertexts $(C_{1,i-1}, \ldots, C_{n,i-1})$, re-encrypts each ciphertext and permutes the resulting ciphertexts using a secretly chosen random permutation.
4. The final decryption phase D, given a set of ciphertexts $(C_{1,k}, \ldots, C_{n,k})$, simply decrypts all the ciphertexts in some distributed manner (in order to achieve robustness).

3.2 Verifiability and Robustness

Recall that a voting system is said to be verifiable if all voters can verify that their vote was counted. A voting system is said to be robust is a small set of servers cannot disrupt the election. Note that the above mix-net protocol is neither verifiable nor robust. In order to obtain these two properties several ingredients must be added to the protocol.

In particular, one ingredient which may be added is the requirement that each *mix* server prove that he has indeed done the correct operation. Namely, each mix_i will be required to prove that there exists a permutation π such that $C_{j,i}$ is a re-encryption of $C_{\pi(j),i-1}$, for $j = 1, \ldots, n$.

In what follows we consider the simpler task of merely proving that one ciphertext is a re-encryption of another. Let $c_1 = (\alpha_1, \beta_1) = (g^t, m_1 y^t)$ and $c_2 = (\alpha_2, \beta_2) = (g^u, m_2 y^u)$ be any two ciphertexts. Note that c_2 is a re-encryption of c_1 if and only if c_1 and c_2 are both encryptions of the same message. Consider the tuple

$$(g, y, \frac{\alpha_2}{\alpha_1}, \frac{\beta_2}{\beta_1}) = (g, y, g^{u-t}, \frac{m_2}{m_1} y^{u-t}).$$

Thus, c_2 is a re-encryption of c_1 if and only if $(g, y, \frac{\alpha_2}{\alpha_1}, \frac{\beta_2}{\beta_1})$ is a DDH tuple, i.e., tuple of the form (g, y, g^r, y^r), which is equivalent to being a tuple of the form (g, g^x, g^r, g^{rx}). Thus, proving that c_2 is a re-encryption of c_1 boils down to proving that $(g, y, g^r, y^r) \in DDH$.

In what follows we describe the Chaum-Pederson protocol [CP92] for proving that a tuple $(g, y, w, u) = (g, g^x, g^r, g^{rx})$ is a DDH tuple.

$$\begin{array}{ccc}
P & & V \\
s \in \mathbb{Z}_q & \xrightarrow[(a,b) = (g^s, y^s)]{\qquad\qquad\qquad} & \\
 & \xleftarrow[c]{\qquad\qquad\qquad} & c \in \mathbb{Z}_q \\
 & \xrightarrow[t = s + cr]{\qquad\qquad\qquad} & \begin{array}{c}\text{accept if and only if} \\ g^t = aw^c \wedge y^t = bu^c\end{array}
\end{array}$$

It is easy to verify that the above protocol is an honest verifier zero-knowledge proof-of-knowledge protocol.

Remarks:

1. Neff proposed a slightly different re-encryption mix-net, also based on El-Gamal. In Neff's protocol a re-encryption operation consists in part of taking a ciphertext (a, b) and generating another ciphertext (a^c, b^c), for a randomly chosen $c \in_R \mathbb{Z}_q$. Note that this operation does change the encrypted message from m to m^c. The motivation behind Neff's scheme is that he manages to give efficient zero-knowledge proofs, which involve only a linear (in n) number of exponentiations.

2. There are faster protocols that are not zero-knowledge, such as the one proposed by Boneh and Golle [BG02] and the one proposed by Jacobsson, Juels and Rivest [JJR02]. Both use new techniques to verify correctness. In [BG02], for each mix server, the product of a random subset of its inputs is computed, and the mix server is required to produce a subset of outputs of equal products. In [JJR02], a new technique is used, called *randomized partial checking*, in which each server provides strong evidence of its correct operation by revealing a pseudo-randomly selected subset of its input/output relations.

3.3 An overview of an El-Gamal based Re-encryption Mix-net

1. Voters vote.

2. An El-Gamal public key (p, g, y) is produced (in a distributed manner)

3. The initial encryption phase is performed.

4. All the mix phases are performed.

5. Each mix phase produces a proof. The proof includes a non-interactive version of the Chaum-Pederson proof, obtained by applying the following Fiat-Shamir type step: the challenge is computed by applying some pseudo-random function to the first message and to the content of the bulletin board; the seed to the pseudo-random function is chosen in a distributed manner.

6. All the proofs are checked, and if they are correct, then the decryption phase is performed by applying a threshold decryption. If a proof of mix_i fails, then the bad server mix_i is skipped and all the mix phases $mix_{i+1}, \ldots, mix_k$ are redone.

Note that so far we only showed how to prove that one ciphertext is a re-encryption of another ciphertext. We didn't show how to fully prove that a *mix* operated correctly.

References

[BG02] D. Boneh and P. Golle. Almost entirely correct mixing with applications to voting. ACM Conference on Computer and Communications Security 2002: 68-77.

[CP92] D. Chaum and T. P. Pedersen. Wallet Databases with Observers. CRYPTO 1992: 89-105.

[JJR02] M. Jakobsson, A. Juels, and R. Rivest. Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking. In D. Boneh, ed., USENIX Security '02, pp. 339-353. 2002. (Also available as IACR eprint 2002/025.)