**6.857 Computer and Network Security** October 29, 2002

# Lecture Notes 15 : Voting, Homomorphic Encryption

*Lecturer: Ron Rivest* *Scribe: Ledlie/Ortiz/Paskalev/Zhao*

## 1 Introduction

The big picture — and where we end up at the end of the lecture — is a voting scheme where (a) each voter casts exactly one ballot and (b) voting is anonymous. We delve into two areas on our way to this goal: Blind Signatures, which allow for anonymous voting, and Paillier Cryptosystem, which gives us the ability to sum up votes even though they have been encrypted. From there, we try out a voting scheme without privacy and show how privacy without cheating can be added to this pretty simple scheme.

## 2 Outline

- Homomorphic Encryption
- Blind Signatures
- Paillier Cryptosystem
- Electronic Voting with Paillier Cryptosystem and Blind Signatures

Historical Note: Bush signed the "Help America Vote Act of 2002"[1] today. This act will give states $3.9 billion to replace outdated punch-card and lever voting machines, and to improve voter education and poll-worker training.

## 3 Homomorphic Encryption

**• Homomorphic Encryption**

The encryption algorithm E() is homomorphic if given E($x$) and E($y$), one can obtain E(x $\perp$ y) without decrypting $x, y$ for some operation $\perp$.

**• RSA (Multiplicative Homomorphism)**

Given $c_i = E(m_i) = m_i^e \bmod N$

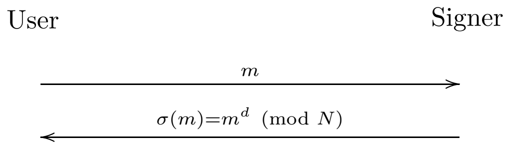
$$\begin{array}{ccccccc}
c_1 & = & m_1^e & \bmod N & & & \\
c_2 & = & m_2^e & \bmod N & & & \\
\hline
c_1 \cdot c_2 & = & m_1^e \cdot m_2^e & \bmod N & = & (m_1 \cdot m_2)^e & \bmod N
\end{array}$$

[1] http://www.cnn.com/2002/ALLPOLITICS/10/29/elec02.bush.changes.ap/index.html

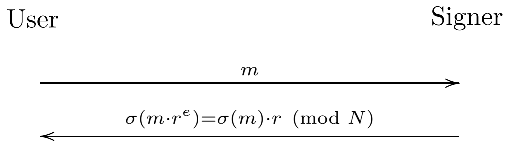
**RSA demonstrates a multiplicative homomorphic property:** $E(m_1) \cdot E(m_2) = E(m_1 \cdot m_2)$

# 4 Blind Signatures

Before we cover Paillier Cryptosystem, let's think about how a vote can be anonymized. To achieve this, we use Blind Signatures.



The diagram above shows how ordinary RSA signatures work.
In order to sign, the signer needs to know what the message $m$ is.



With blind signatures, the signer does not know what the message $m$ is but is still able to sign the message.

**Calculation performed by Signer on last sending:**
$\sigma(m \cdot r^e) = (m \cdot r^e)^d = m^d \cdot r^{ed} = m^d \cdot r = \sigma(m) \cdot r \pmod N$

**Note** The symbols $e$ and $d$ refer to the encryption and decryption keys, respectively. Also note, when you divide the response by $r$, you get the signature of the message $\sigma(m)$ : $\frac{\sigma(m) \cdot r}{r} = \sigma(m)$.

- **Physical Example of Blind Signature Scheme**

Materials: Envelope, White paper, Carbon paper

1. Place the carbon paper on top of the white paper
2. Place both items in the envelope and seal
3. Sign on the envelope

In the end, the signer does not know what he has signed.

- **Examples**

In the top figure $x$ represents a coin in the special form $''Bank...r''$, where $r$ is a random number. $\sigma(x)$ is a coin signed by the bank. Hence, the bank recognizes its signature but cannot trace the

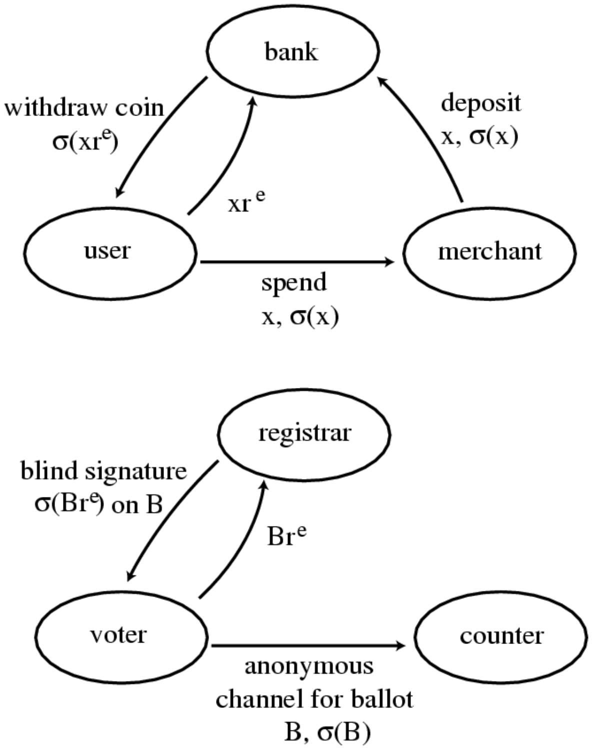


Figure 1: Two examples of Blind Signatures.

coin. The initial signing is information theoretically secure; however, problems arise when user tries to spend a coin multiple times.

Similarly, in the bottom figure, the registrar signs the voter's sealed vote, which the voter then passes on to the counter.

# 5 Paillier Cryptosystem

## 5.1 Key Generation

Like RSA, pick two primes $p$, $q$ and let $N = p \cdot q$ — but here we are going to work mod $N^2$. Note that $\varphi(N^2) = N \cdot \varphi(N) = N \cdot \varphi(p) \cdot \varphi(q)$ and that all elements have order dividing $\varphi(N^2)$. Create $PK = (N, g)$ where $g$ has order a multiple of $N$ and $SK = (\lambda(n))$ where $\lambda(n) =$lcm$(p-1, q-1)$ (where "lcm" denotes lowest common multiple).

### 5.2 Encryption

To encrypt a message $m \in Z_N$:

- Choose $x \in_R Z_N^*$.
- Produce an encryption $E(m) = g^m x^N \bmod N^2$.

Doing some arithmetic, here we can show that Paillier has the following useful property:

$$\begin{array}{rcll} E(m_1) & = & g^{m_1} x_1^N & \bmod N^2 \\ E(m_2) & = & g^{m_2} x_2^N & \bmod N^2 \\ \hline E(m_1) \cdot E(m_2) & = & g^{m_1+m_2} (x_1 x_2)^N & \bmod N^2 \quad = E(m_1 + m_2) \end{array}$$

**Paillier demonstrates an additive homomorphic property.**

### 5.3 Decryption

Use $L(u) = \frac{(u-1)}{N}$ for $u \equiv 1 \pmod N$. Note that the formula for $L(u)$ is not modulo anything.

If $c = E(m)$, then

$$m = \left( \frac{L(c^{\lambda(n)} \bmod N^2)}{L(c^{\lambda(n)} \bmod N^2)} \right) \bmod N$$

Proof is given in [1].

### 5.4 Benefits of Paillier Cryptography

- It gives the homomorphic property we want for voting.
- It is *semantically secure*, assuming that it is hard to distinguish $N^{th}$ residues from non-$N^{th}$ residues mod $N^2$ (in addition to the DLP). *Semantically secure* means that you cannot distinguish $E(0)$ from $E(1)$ better than 50% as $N \to \infty$.

## 6 Voting with Paillier Cryptosystem and Blind Signatures

### 6.1 Motivation: Bulletin Board Voting

We are aiming to use $E(m_1) \cdot E(m_2) = E(m_1 + m_2)$ so that we can add up the votes.

Imagine that we have a large public bulletin board with candidates X, Y, Z and voters $V_1, \ldots, V_n$.

| | X | Y | Z |
|---|---|---|---|
| $V_1$ | 1 | 0 | 0 |
| $V_2$ | 0 | 1 | 0 |
| $V_3$ | 1 | 0 | 0 |
| Total | 2 | 1 | 0 |

With a regular bulletin board we have the ballots in plaintext. X, Y, and Z denote candidates. The $V_i$ are voters. The entry 1 represents a vote for a candidate.

| | X | Y | Z |
|---|---|---|---|
| $V_1$ | $C_{1X}$ | $C_{1Y}$ | $C_{1Z}$ |
| $V_2$ | $C_{2X}$ | $C_{2Y}$ | $C_{2Z}$ |
| $V_3$ | $C_{3X}$ | $C_{3Y}$ | $C_{3Z}$ |
| Total | $C_X$ | $C_Y$ | $C_Z$ |

$C_{1X}$, etc., denote votes encrypted using some public key scheme, with the private keys owned by election officials.

If we have the additive homomorphic property, we only need to decrypt the figures in the row of totals, thereby providing voter privacy. We want to find $E(T_j) = \prod_i E_{ij}$.

Question : *Does encryption have to be randomized?*

Answer : Yes. Encryptions of 0s must look different, so that an adversary will not be able to tell whether two people made the same vote by simply checking whether the ciphertexts are the same. Similarly encryptions of 1s must also look different. Hence, simple RSA is therefore not suitable for this application.

## 6.2 Correctness

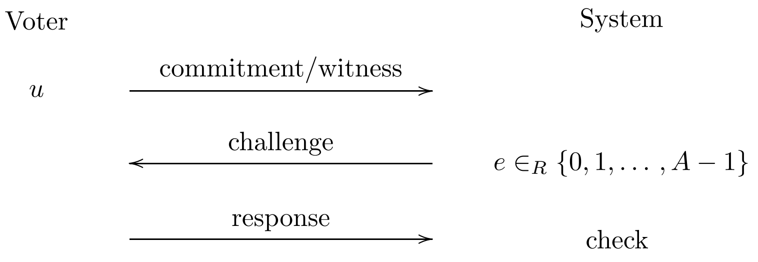
Now that we have these two tools, Paillier Cryptosystem and Blind Signatures, we can return to our public bulletin board of votes.

- To stop a voter from submitting multiple votes for one candidate (or negative votes) we have to make sure that $m_{ij} \in \{0,1\}$ and that the row subtotal $\in \{0,1\}$.
- We also need to enforce non-malleability. For example if Alice's vote were $m_{ij} \in \{0,1\}$, Bob could negate Alice's ballot in this fashion by computing:

$$\frac{E(1)}{E(m_{ij})} = E(1) \cdot E(-m_{ij}) = E(1 - m_{ij})$$

The solution is to use zero knowledge proofs.

- To prove the vote is correct:



This round is performed $t$ times.

**Fiat-Shamir trick:** replace $e \in_R \{0, 1, \ldots, A-1\}$ with $e = \text{hash(commitment)}$ to make it a non-interactive ZK proof. Hence, *commitment/witness* and *response* can be reduced to one message.

- We now show how the voter can prove that he knows how he is voting (knows $m$), when using Paillier encryption.

$$c = g^m x^N \bmod N^2$$

$$\begin{tikzcd}
\text{Voter} & & \text{System} \\
\begin{matrix} {\scriptstyle r \in_R Z_N} \\ s \in_R Z_N^* \end{matrix} \arrow[rr, "u=g^r\, s^N \bmod N^2"] & & \\
& & e \in_R \{0, 1, \ldots, A-1\} \arrow[ll, "e"'] \\
\begin{matrix} {\scriptstyle v=r-em} \\ w = s\, x^{-e}\, g^{(r-em) \div N} \end{matrix} \arrow[rr, "v{,}w"] & & \text{check } g^v\, c^e\, w^N \overset{?}{=} u
\end{tikzcd}$$

The check becomes

$$g^{r-em}\, (g^m\, x^N)^e\, (s\, x^{-e}\, g^{(r-em) \div N})^N \overset{?}{=} u \Longleftrightarrow g^r\, s^N \overset{?}{=} u$$

After $t$ successful challenges, the chance of forgery is $\simeq \frac{1}{A^t}$.

Question : *How can you stop decryption of individual entries with the secret key $d$? Who should control the secret key?*

Answer : Use secret sharing to divide $d$ such that any $t$ shares are to be used to recreate $d$. Unfortunately, this still leaves key generation and decryption (recombination of shares) as vulnerable points.

# References

[1] O. Baudron, P. Fouque, D. Pointcheval, G. Poupard, and J. Stern. Practical multi-candidate election system. In *Proc. of the ACM Symp. on Principles of Distributed Computing*, Philadelphia, 2001.