6.857 Computer and Network Security	October 4, 2001
Lecture Notes 9 : Homomorphic Encryption	
Lecturer: Ron Rivest	*Scribe: Chan/Kottahachchi/Samaranayake/Wee*

1 Introduction

2 Outline

- Voting schemes based on homomorphic encryption
- Zero-Knowledge proofs

2.1 Homomorphisms

Homomorphic properties of Public-Key Systems:

1. RSA is multiplicative:
$$E(m_1) \cdot E(m_2) = E(m_1 \cdot m_2)$$
2. Pallier has the additive homomorphic property:
$$E(m_1) \cdot E(m_2) = E(m_1 + m_2)$$

	X	Y	Z
V_1	1	0	0
V_2	0	1	0
V_3	1	0	0
Total	2	1	0

Usually, 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.

Question	:	*Does encryption have to be randomized?*
Answer	:	Yes. We want encryptions of 0s to 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. The same applies to encryptions of 1's too. RSA is therefore not suitable for this application.

2.2 Issues

In the rest of this lecture, we would like to address some of the following issues:

1. *Pallier?* Do schemes that satisfy the additive homomorphic property exist?
2. *Are ballots well-formed?* We want only 0s and 1s, no double votes or negative votes.
3. *Write-in Ballots?* How should write-in ballots be encrypted and processed? [next lecture]
4. *Protect against abuse of decryption power (that is, breaching voter privacy)?* Separation of power, threshold keys.

2.3 Pallier Scheme

1. Key setup:
   (a) $N = pq$, p, q large primes, work mod N^2. (any element has order that divides $\phi(N^2) = pq(p-1)(q-1)$).
   (b) Let g be an element whose order is a multiple of N.
   (c) $PK = (N, g), SK = \lambda(N) = \text{lcm}(p-1, q-1)$.
2. Encryption:
   (a) $M \in \mathbb{Z}_N, x \in_R \mathbb{Z}_N^*$
   (b) $E(m) = c = g^M x^N (\text{mod} N^2)$
3. Homomorphic:

$$\begin{array}{l} E(M_1) = g^{M_1} x_1^N (\text{mod} N^2) \\ E(M_2) = g^{M_2} x_2^N (\text{mod} N^2) \\ \hline E(M_1) \cdot E(M_2) = g^{M_1+M_2} (x_1 x_2)^N (\text{mod} N^2) \end{array}$$

4. Decryption:
   (a) $M = \frac{L(c^{\lambda(N)} \text{mod} N^2)}{L(g^{\lambda(N)} \text{mod} N^2)} \bmod N$, where $L(u) = \frac{u-1}{N}$ (domain: $\{u | u \equiv 1 (\text{mod} N)\}$).
   (b) Refer to [1] for correctness.

Security of this Scheme

1. Depends on distinguishing between N^{th} residues from non N^{th} residues:

$$c \text{ is encryption of } m \leftrightarrow \frac{c}{g^m} \bmod N^2 \text{ is an } N^{th} residue$$

2. Scheme is randomized and semantically secure. Cannot distinguish between $E(0)$ from $E(1)$ with probability better than 50%.

Are ballots well-formed?

We want to append to the signature $E(m)$ a convincing "proof" that $m \in \{0,1\}$. At the same time, this "proof" must not reveal what m is. This may seem like a set of contradictory requirements, but as we do often see in cryptography, it is possible to fulfill this two requirements at the same time through a construct known as Zero-Knowledge Proofs developed at MIT by Micali, et al [2].

Note that in the Pallier Scheme, it is easy to copy a vote, or even to invert a vote: if $c = g^M x^N (\bmod N^2)$ is a valid encryption of m, then $gc^{-1} = g^{1-M}(x^{-1})^N (\bmod N^2)$ is a valid encryption of $1-m$. [1]

Question	:	*Can we just do a running count at regular intervals, and if the totals don't jump too much in relation to the number of voters, assume that the ballots are well-formed?*
Answer	:	This scheme will not work if the adversary encrypts a small number like 2 in his ballot or if someone abstains.

3 Zero-Knowledge Proofs

3.1 The General Set-up

(In our discussion of ZK proofs, we will use the RSA scheme rather than the Paillier scheme in our examples, because it is simpler. For the Paillier version of these ZK proofs, see the paper handed out.)

Suppose we have a ciphertext from the RSA encryption of a message , $c = E(m) = m^e (\bmod n)$, and the Prover wants to convince the Verifier that he knows m. A two-party protocol [2] that achieves this goal looks like this:

P = Prover		V = Verifier
wants to prove he knows m		wants to learn if P knows m
	$\xrightarrow{w=\text{witness}}$	
	$\xleftarrow{x=\text{random challenge}}$	
	$\xrightarrow{y=\text{response}}$	test (c, w, x, y).
		Accept if OK, reject otherwise.

We want the protocol to satisfy the following properties:

1. If P does know m, then V always accepts. (Completeness)

[1]The zero-knowledge proof addresses this issue too, as the duplicated or modified vote will need to include a response to a different challenge in zero-knowledge proof.

[2]There is another nice layman exposition of zero-knowledge proofs on RSA's Web site at `http://www.rsa.com/rsalabs/faq/2-1-8.html`

2. If P can answer successfully many challenges, then can compute m.
3. If P does not know m, then can succeed on only a few challenges. (Soundness)

Question	:	*Isn't "If V accepts, P always knows m" preferable?*
Answer	:	These protocols are normally randomized, and we therefore need to at least allow room for lucky guessing on the part of the Prover, so that even when V accepts, there is still a very small chance that P actually does not know m.

3.2 Interactive Zero-Knowledge Proof for RSA

$$
\begin{array}{ccl}
\text{Prover} & c = E(m) = m^e & \text{Verifier} \\
r \in_R \mathbb{Z}_n^*\{0,1\}^n & \xrightarrow{\quad w = r^e (\operatorname{mod} n) \quad} & \\
 & \xleftarrow{\quad\quad\quad x \quad\quad\quad} & x \in_R \{0, 1, \ldots, 2^t - 1\} \\
 & & \text{random } t\text{-bit string} \\
 & \xrightarrow{\quad\quad y = m^x r \quad\quad} & \text{test } y^e \stackrel{?}{=} c^x w
\end{array}
$$

Completeness. If the Prover knows m, then he will send y such that $y^e = (m^x r)^e = m^{ex} r^e = c^x w (\operatorname{mod} n)$, so the Verifier will accept.

Soundness. Suppose the Prover is able to respond correctly to two different challenges, yielding two proofs (w, x, y) and $(w, \overline{x}, \overline{y})$. We want to show that he will then be able to compute m with high probability.

$$
\begin{aligned}
y = m^x r, \overline{y} &= m^{\overline{x}} r \\
y / \overline{y} &= m^{x - \overline{x}}
\end{aligned}
$$

Given m^a and m^b, he can compute $m^{\gcd(a,b)}$.

$$
\begin{aligned}
& y = m^x r, \overline{y} = m^{\overline{x}} r \\
& y / \overline{\overline{y}} = m^{x - \overline{\overline{x}}} \Rightarrow m^{\gcd(x - \overline{x}, x - \overline{\overline{x}})} = m \text{ with probability } \tfrac{6}{\pi^2} = 0.60793
\end{aligned}
$$

where the last step follows from the following result in number theory[4].

Theorem 1 *(Dirichlet) If we choose u and v at random, then* $\operatorname{Prob}[\gcd(u, v) = 1] = \frac{6}{\pi^2}$.

Question	:	*Is it possible to combine the first and third rounds, that is, begin with the Verifier sending x, and then the Prover responds with (w, y)?*
Answer	:	No, because this would allow the Prover to "cheat", by picking y at random, and then computing $w = y^e / c^x$, and the Prover could then make the Verifier accept although he does not know what m is.

Zero-Knowledge. The triples (w, x, y) can be generated efficiently by V:

$$\text{Pick } x \text{ random, } y \text{ random, and compute } w = y^e/c^x.$$

These triples have the same distribution as the ones in the proof [3]. Therefore, V learns nothing else from the proof apart from the fact that P does know m, since he could have generated the transcript of the protocol himself.

Question	:	*What if the Verifier does not pick w randomly, but instead sends a special w based on some transaction records? In that case, the simulation in proving zero-knowledge fails.*
Answer	:	Indeed, and therefore this protocol isn't strictly zero-knowledge (it is only "honest-verifier zero-knowledge"). This is partly why we replace w with a hash function in the non-interactive protocol in the next section. (And since the hash function is "honest", we are OK...

3.3 Non-Interactive Protocol for RSA

To make non-interactive, let $x = h(c, w)$ where h is a hash function that serves as a random oracle (Fiat-Shamir Trick [3]). h provides a source of random bits, which we could use to generate the random challenge, thereby eliminating the second round. Therefore, the zero-knowledge proof that I know the message m for a ciphertext c is a triplet $(w, x = h(c, w), y)$.

The proof for showing that $m \in \{0, 1\}$ is in [1]. Proving that the row sum $\in \{0, 1\}$ is easy.

References

[1] O. Baudron, P.-A. Fouque, D. Pointcheval, G. Poupard, and J. Stern. Practical Multi-Candidate Election System, *Proceedings of the ACM Conference on Principles of Distributed Computing, 2001* from `ftp://ftp.di.ens.fr/pub/users/pointche/Papers/2001_podc.pdf`

[2] S. Goldwasser, S. Micali, C Rackoff. The Knowledge Complexity of Interactive Proof Systems, *SIAM J Computing, Vol. 18, No. 1, Feb 1989*

[3] A. Fiat and A. Shamir, How to prove yourself: Practical solutions to identification and signature problems, *Advances in Cryptology - Crypto '86, Springer-Verlag (1987), 186-194.*

[4] D. E. Knuth, *The Art of Computer Programming, Vol 2., Seminumerical Algorithms*, 2nd edition, Addision-Wesley, Reading, MA 1981.

[3]provided both the prover and the verifier behave honestly and adhere to the protocol; in particular, V must choose w randomly.