HSR Hochschule für Technik Rapperswil

Homomorphic Tallying with Paillier Cryptosystem

E-Voting seminar

Sansar Choinyambuu - MSE student
12/6/2009

An Introduction

This paper introduces a Paillier's cryptosystem in general, and explains how it can be used in Cryptographic voting system given its Homomorphic property. The paper mainly focuses on the tallying part of the election process, using Paillier Cryptosystem, which gives us the ability to sum up votes even though they have been encrypted.

Outline

- Homomorphic encryption
- Mathematical functions and notations
- Paillier's Cryptosystem
- Using Paillier encryption's additively homomorphic property for vote tallying
- Example
- Literature

Homomorphic Encryption

Definition: The encryption algorithm E() is homomorphic if given E(x) and E(y), one can obtain E(x ¬ y) without decrypting x; y for some operation ¬.

Examples

✓ RSA is a multiplicative homomorphic algorithm

$c_i = E(m_i) = m_i^e \bmod N$ public key is modulus N and exponent e

$c_1 \cdot c_2 = m_1^e \cdot m_2^e \bmod N = (m_1 \cdot m_2)^e \bmod N \rightarrow E(m_1) \cdot E(m_2) = E(m_1 \cdot m_2)$

✓ ElGamal is multiplicative homomorphic algorithm as well

In a group G, if the public key is (G,q,g,h), where $h = g^x$, and x is the secret key

$c_i = E(m_i) = (g^r, m_i \cdot h^r)$ for some $r \in \{0, \dots, q - 1\}$

$c_1 \cdot c_2 = E(m_1) \cdot E(m_2) = (g^{r1}, m_1 \cdot h^{r1})\,(g^{r2}, m_2 \cdot h^{r2}) = (g^{r1+r2}, (m_1 \cdot m_2)\, h^{r1+r2}) = E(m_1 \cdot m_2)$

Mathematical functions and notations

To be able to understand how Paillier Cryptosystem works, one should have knowledge on the following basic mathematical concepts.

General common divisor (gcd) of two or more non zero integers is the largest positive integer that divides the numbers without a remainder. The greatest common divisor of *a* and *b* is written as gcd(*a*, *b*), or sometimes simply as (*a*, *b*). For example $gcd(4,\ 6)\ =\ 2, gcd(4,\ 14)\ =\ 2$. Two numbers are called *coprime* or *relatively prime* if their greatest common divisor equals to 1. For example, 9 and 28 are relatively prime.

Least common multiple (lcm) of two or more non zero integers is the smallest integer that is divisible by every member of a set of numbers without a remainder. For example,

$$lcm(4,\ 6)\ =\ 12, lcm(4,\ 14)\ =\ 28.$$

It's a remarkable fact that $ab\ =\ lcm(a,b)\ *\ gcd(a,b)$, thus $lcm(a,b)\ =\ \frac{ab}{gcd\,(a,b)}$

This fact can be easily seen that $gcd(a,b)$ is the product of the common prime factors of ab, and the remaining factors would result $lcm(a,b)$.

Euler's totient function (phi function) The totient of a positive integer *n* is defined to be the number of positive integers less than or equal to *n* that are coprime to *n*. For example

ϕ (9) = 6 since the six numbers 1, 2, 4, 5, 7 and 8 are coprime to 9. If *n* can be factorized to distinct prime numbers *p* and *q*, then $\varphi(n)\ =\ (p-1)(q-1)$. For example

$$\phi(15)\ =\ \phi(3*5)\ =\ (3-1)(5-1)\ =\ 8$$

Carmichael's function (λ function) is given by the least common multiple (lcm) of all the factors of the totient function ϕ(n). If n can be factorized to prime number p and q

$$\lambda(n)\ =\ lcm\ (p-1, q-1)$$

Modular multiplicative inverse of an integer *a* modulo *m* is an integer *x* such that

$a^{-1}\ \equiv x\ mod\ m$ This is equivalent to $ax\ \equiv 1\ mod\ m$

The multiplicative inverse of *a* modulo *m* exists if and only if *a* and *m* are coprime (i.e., if $gcd(a,\ m)\ =\ 1$).

Converting a decimal Number to any base number The remainders that we get when we sequentially divide the decimal number by the base end up being the digits of the result, which are read from bottom to top. Example: convert 190_{10} to base 3.

$190 = 2*3^4 + 1*3^3 + 0*3^2 + 0*3^1 + 1*3^0$

The following notations are used frequently in Paillier Cryptosystem explanation:

$\mathbb{Z}_n$ - set of integers n

$\mathbb{Z}_n{}^*$ - set of integers coprime to n - this set consists of $\phi(n)$ number of integers

$\mathbb{Z}_{n^2}{}^*$ - set of integers coprime to n^2 - this set consists of $n\phi(n)$ number of integers

Paillier cryptosystem

The Paillier Cryptosystem named after and invented by French researcher Pascal Paillier in 1999 is an algorithm for public key cryptography.

The distinguishing technique used in public key cryptography is the use of asymmetric key algorithms, where the key used to encrypt a message is not the same as the key used to decrypt it. Each user has a pair of cryptographic keys — a *public* key and a *private* key. The private key is kept secret, whilst the public key may be widely distributed. Messages are encrypted with the recipient's public key and can only be decrypted with the corresponding private key. The keys are related mathematically, but the private key cannot be feasibly (ie, in actual or projected practice) derived from the public key.

The Paillier Cryptosystem's scheme works as follows:

Key generation

1. Choose two large prime numbers *p* and *q* randomly and independently of each other such that $\gcd(pq, (p-1)(q-1)) = 1$

   This property is assured if both primes are of equivalent length, i.e., $p, q \in 1 \parallel \{0,1\}^{s-1}$ for security parameter *s*.

2. Compute RSA modulus *n* = *pq* and

   Carmichael's function $\lambda = lcm(p-1, q-1)$ it can be computed using $\lambda = \frac{(p-1)(q-1)}{\gcd(p-1,q-1)}$

3. Select generator *g* where $g \in \mathbb{Z}^*_{n^2}$ There are two ways of selecting the g.
   a. Randomly select g from a set $\mathbb{Z}^*_{n^2}$ where
   $$\gcd\left(\frac{g^{\lambda} \bmod n^2-1}{n}, n\right) = 1$$

   There are $\phi(n) * \phi(n)$ number of valid generators, therefore the probability of choosing them out of $n\phi(n)$ elements of $\mathbb{Z}^*_{n^2}$ set is relatively high for big n.

   b. Select α and β randomly from a set $\mathbb{Z}_n^*$ then calculate
   $g = (\alpha n + 1)\beta^n \bmod n^2$
   In this case the selected generator always meets the condition above.
4. Calculate the following modular multiplicative inverse
   $\mu = \left(L\left(g^{\lambda} \bmod n^2\right)\right)^{-1} \bmod n$
   Where the function *L* is defined as $L(u) = \frac{u-1}{n}$
   This multiplicative inverse exists if and only if valid generator was selected in previous step.

- The public (encryption) key is (*n*,*g*).
- The private (decryption) key is (λ,μ).

A a simpler variant of the above key generation steps would be to set $g = n + 1,\ \lambda = \varphi(n)$ and $\mu = \varphi(n)^{-1}\ mod\ n$, where $\varphi(n) = (p-1)(q-1)$.

Encryption

1. Let *m* be a message to be encrypted where $m \in \mathbb{Z}_n$
2. Select random *r* where $r \in \mathbb{Z}^*{}_n$
3. Compute ciphertext as: $c = g^m \cdot r^n\ mod\ n^2$

Decryption

1. Ciphertext $c \in \mathbb{Z}^*{}_{n^2}$
2. Compute message: $m = L\left(c^\lambda\ mod\ n^2\right) \cdot \mu\ mod\ n$

Homomorphic properties

A notable feature of the Paillier cryptosystem is its homomorphic properties. As the encryption function is additively homomorphic, the following identities can be described:

- **Homomorphic addition of plaintexts**

  The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts,

  $$D(E(m_1, r_1) * E(m_2, r_2)\ mod\ n^2) = m_1 + m_2\ mod\ n$$

  The product of a ciphertext with a plaintext raising g will decrypt to the sum of the corresponding plaintexts,

  $$D(E(m_1, r_1) * g^{m_2}\ mod\ n^2) = m_1 + m_2\ mod\ n$$

Practically, this leads to the following identities: Where $\forall\ \boldsymbol{m_1, m_2}\ \in\ \mathbb{Z}_{\boldsymbol{n}}\ \boldsymbol{and}\ \boldsymbol{k} \in \mathbb{N}$

$$D(E(m_1)E(m_2)\ mod\ n^2) = m_1 + m_2\ mod\ n$$

$$D\left(E(m)^k\ mod\ n^2\right) = km\ mod\ n$$

$$D(E(m_1)g^{m_2}\ mod\ n^2) = m_1 + m_2\ mod\ n$$

$$\left.\begin{matrix} D(E(m_1)^{m_2}\ mod\ n^2) \\ D(E(m_2)^{m_1}\ mod\ n^2) \end{matrix}\right\} = m_1 m_2 \bmod n\ .$$

Applications

- **Electronic Voting**

The above homomorphic properties can be utilized by secure electronic voting systems. Consider a simple binary ("for" or "against") vote. Let *m* voters cast a vote of either *1* (for) or *0* (against). Each voter encrypts their choice before casting their vote. The election official takes the product of the *m* encrypted votes and then decrypts the result and obtains the

value *n*, which is the sum of all the votes. The election official then knows that *n* people voted *for* and *m-n* people voted *against*. The role of the random *r* ensures that two equivalent votes will encrypt to the same value only with negligible likelihood, hence ensuring voter privacy.

Using Paillier encryption's additively homomorphic property for vote tallying

For the voting application one of the two possible additive homomorphic encryption algorithms are usually employed: Paillier encryption or modified ElGamal encryption. Paillier encryption is inherently additive homomorphic and more frequently applied. The original ElGamal encryption scheme can be simply modified to be additive homomorphic: a message is used as an exponent in an exponentiation computation, and then the exponentiation is encrypted using the original ElGamal encryption. A passive result of this modification is that search for a logarithm must be performed in the decryption function, which becomes inefficient when the searching space is big. As the number of the voters is often large in voting applications, Paillier cryptosystem is usually preferred.

Let's assume the election will take place with:

N_v– Number of voters

N_c – Number of candidates

We have to present each vote in numeric form and encrypt them with Paillier's encryption. Each Paillier encryption requires a random number, so that the same votes will be encrypted to different ciphers.

If we denote the base, which we use to encrypt the messages, as b then the following condition must be held. Namely, the base must be greater than the number of voters.

$$b > N_v$$

Base serves in the same way as 2 for binary system, and 10 for decimal system.

Vote messages for candidates will be seen as:

1st candidate: b^0

2nd candidate: b^1

…

N_c-th candidate: b^{N_c-1}

Then maximum possible number representing single vote m_{max} can exceed to:

$$m_{max} = \sum_{i=1}^{N_c} b^{i-1}$$ (A voter has selected all candidates)

The maximum possible tally of the all votes can exceed to:

$$T_{max} = N_v * m_{max}$$ (All voters have selected all candidates)

Key generation

Public Key

To be able to encrypt T_{max} , RSA modulus n must hold the following:

$$n \geq T_{max} + 1 \qquad n = p\,q$$

Where $p\ and\ q$ are large primes and $\gcd(pq, (p-1)(q-1)) = 1$

Select random integer *g* where $g \in \mathbb{Z}^*_{n^2}$ and $\gcd\left(\frac{g^{\lambda}\ mod\ n^2-1}{n}, n\right) = 1$

The election authorities should choose the prime numbers p and q considering the number of voters and candidates as described above.

Private Key

λ = *lcm*(*p* − 1,*q* − 1) with *λ(n)* being the Carmichael function

Modular multiplicative inverse: $\mu = \left(L\left(g^{\lambda}\ mod\ n^2\right)\right)^{-1}\ mod\ n$

where function *L* is defined as $L(u) = \frac{u-1}{n}$.

Encryption

$$E(m_i) = c_i = g^{m_i} \cdot r_i^{\,n}\ mod\ n^2$$ Where $r \in \mathbb{Z}^*_n$

Tallying

At the end of the election authorities would have at most N_v of encrypted votes. Then authorities can calculate the encrypted tally which is the product of all encrypted votes modulo n^2.

$$T = \prod_{i=1}^{N_v} c_i\ mod\ n^2$$ T – Tally

Decryption

As described in homomorphic properties of Paillier encryption:

$$m = L\left(c^{\lambda}\ mod\ n^2\right) \cdot \mu\ mod\ n$$

$$D(T) = \sum_{i=1}^{N_v} m_i\ \ mod\ n$$

As the result of this decryption function, one gets simple tallying of all votes. To know how many votes there were for every candidate we can use “Division remainder” method with number of the voters as base. Please refer to the following example for the use of Paillier’s cryptosystem in voting.

Example

This is the example demonstrating a small election, which uses Paillier Cryptosystem.

$N_v = 9$, $N_c = 5$. Base b is selected as 10. ($b > N_v$)

Say we want to choose 2 new members for the world parliament. Choosing two candidates is preferred. However choosing one candidate or leaving the ballot empty can also be an option.

Voter Name	Donald Trump 10^0	Roger Federer 10^1	Britney Spears 10^2	Dalai lama 10^3	Steve Jobs 10^4	Vote messages to be encrypted
Alice		✓				m = 10^1 = 10
Bob			✓		✓	m = 10^2 + 10^4 = 10100
Carol						m = 0
Dave				✓		m = 10^3 = 1000
Eve	✓			✓		m = 10^0 + 10^3 = 1001
Fred		✓		✓		m = 10^1 + 10^3 = 1010
Gil			✓	✓		m = 10^2 + 10^3 = 1100
Helen		✓		✓		m = 10^1 + 10^3 = 1010
Isaac	✓					m = 10^0 = 1
Total	2	3	2	5	1	

As we see from the election rules, the maximum vote message that can ever happen to be encrypted is: $m_{max} = 10^4 + 10^3 = 11000$

And the maximum possible tally can result $T_{max} = N_v * m_{max} = 9 * 11000 = 99000$

To be able to encrypt T_{max} , $n > T_{max}$; $n > 99000$

Derived from that $p \; and \; q > \sqrt{99000}$ where p and q are assumed to have same length.

Key generation

1. So we choose primes randomly $p = 293, q = 433$
   $\gcd(pq, (p-1)(q-1)) = 1$ Holds here
2. $n = pq = 126869$ $n2 = 16095743161$ RSA modulus n
3. $\lambda = \frac{(p-1)(q-1)}{\gcd(p-1,q-1)} = 31536$ Carmichael's function
4. We choose Paillier generator g randomly where $g \in \mathbb{Z}^*_{n^2}$ and
   $\gcd\left(\frac{g^\lambda \bmod n^2-1}{n}, n\right) = 1$ $g = 6497955158$

5. $\mu = \left(L\left(g^{\lambda} \bmod n^2\right)\right)^{-1} \bmod n =$

$\left(6497955158^{31536} mod\ 16095743161 - 1/_{126869}\right)^{-1} mod\ 126869 = 53022$

Encryption of the vote messages

$E(m_i) = c_i = g^{m_i} \cdot r_i^{n} \bmod n^2 = 6497955158^{m_i} \cdot r_i^{126869} \bmod 16095743161 \qquad r \in \mathbb{Z}^*_n$

Voter Name	Vote messages to be encrypted	Random r_i	Encrypted Vote c_i
Alice	m = 10^1 = 10	35145	13039287935
Bob	m = 10^2 + 10^4 = 10100	74384	848742150
Carol	m = 0	96584	7185465039
Dave	m = 10^3 = 1000	10966	80933260
Eve	m = 10^0 + 10^3 = 1001	17953	722036441
Fred	m = 10^1 + 10^3 = 1010	7292 *	350667930 *
Gil	m = 10^2 + 10^3 = 1100	24819	4980449314
Helen	m = 10^1 + 10^3 = 1010	4955 *	7412822644 *
Isaac	m = 10^0 = 1	118037	3033281324
Simple tally	23251		

*Note that the same votes from Fred and Helen are encrypted to different ciphers with the help of randomization.

Tallying

$T = \prod_{i=1}^{N_v} c_i \bmod n^2 = (13039287935 * 848742150 * 7185465039 * 80933260 * 722036441 * 350667930 * 4980449314 * 7412822644 * 3033281324) \bmod 16095743161 = 2747997353$

Decryption

$$m = L\left(c^{\lambda} \bmod n^2\right) \cdot \mu \bmod n$$
$$= \left(\frac{\left(2747997353^{31536} \bmod 16095743161\right) - 1}{126869}\right) \cdot 53022 \bmod 126869$$
$$= 15232$$

So $D(T) = \sum_{i=1}^{N_v} m_i \bmod n$ is now proven in the above example. In other words encrypted tally of the all votes decrypts to the sum of all plain votes.

Now the election authority wants to know who have won the election. To know this, we convert the decrypted tally, which is in decimal form, to a number with the base chosen at the beginning of the election.

We used the base 10, so actually there is no conversion needed.

$15232 = 1 \cdot 10^4 + 5 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 2 \cdot 10^0$

As the result of this election Dalai Lama and Roger Federer would be sitting in world parliament.

Literature

1. E-Voting Simulator based on the Paillier Cryptosystem, Andreas Steffen, HSR Hochschule für Technik Rapperswil
2. Interactive demonstration of Paillier Cryptosystem, Omar Hasan
3. Code for interactive demonstration of Paillier Cryptosystem, Omar Hasan
4. E-Voting Simulator Glossary, Andreas Steffen, HSR Hochschule für Technik Rapperswil
5. GoogleTechTalk: Verifying Elections with Cryptography (2007), Ben Adida
6. Paillier Cryptosystem, Wikipedia
7. Public-key cryptosystems based on composite degree residuosity classes (1999), Pascal Paillier
8. A Generalisation, a Simplification and some Applications of Paillier's Probabilistic Public-Key System (2001), Ivan Damgard, Mads Jurik
9. A Generalization of Paillier's Public-Key System with Applications to Electronic Voting (2003), Ivan Damgard, Mads Jurik, Jesper Buus Nielsen
10. End-to-end auditable voting systems Wikipedia
11. Paillier's Cryptosystem Università di Catania
12. Homomorphic encryption Wikipedia
13. Lecture Notes 15 : Voting, Homomorphic Encryption Ron Rivest
14. Progress in cryptology: INDOCRYPT 2004 : 5th International Conference Anne Canteaut, Kapaleeswaran Viswanathan