A Comparison of El Gamal and Paillier Cryptosystems

Nicholas Chen
nicholasbchen@umail.ucsb.edu
Department of Computer Science
University of California Santa Barbara

June 2018

Abstract

This paper will examine the concepts behind both the El Gamal and Paillier style of public key cryptography. I will compare and contrast the size and complexity of both algorithms. I will focus on Pailliers method of encryption, in particular, its additive homomorphism and the benefits this style can bring to practice.

1 El Gamal Encryption

The discrete exponential function is the basis of El Gamal's cryptosystem:

$$\text{Exp:} Z_{p-1} \rightarrow Z_p^*, x \rightarrow g^x$$

There is no efficient algorithm for computing the inverse function, that is computing x given $y = g^x$. This is called the discrete logarithm assumption. The encryption scheme is described as such:

Key Generation:
1. Choose a large prime p
2. Choose a random $x \in Z_p^*$
3. Pick a generator $g \in Z_p^*$
4. Compute $y = g^x \bmod p$

The public key of this encryption is (p, g, y) and x is kept as the secret key.

Encryption:

1. Prepare a message $m \in Z_p$
2. Choose a random k that is relatively prime to $p-1$
3. $a = g^k \bmod p$
4. $b = y^k \cdot M \bmod p$
5. Send ciphertext $c = (a, b)$

Decryption:

1. $(a, b) = c$
2. $M(\text{plaintext}) = b/a^x \bmod p$

The security of this scheme depends on the assumption that it is impossible to compute g^{xk} from g^x and g^k, and hence g^{-xy} and M. This is known as the Diffie-Hellman problem. An efficient algorithm to compute discrete logarithms would solve this problem but no algorithms exist thus far.

2 Paillier Encryption

Paillier's public key encryption scheme is based on the Composite Residuosity Class Problem. We begin with the following definition:

Definition 1. A number z is said to be the n-th residue modulo n^2 if there exists a number $y \in Z^*_{n^2}$ such that

$$z = y^n \bmod n^2.$$

The problem of deciding n-th residuosity, i.e. distinguishing n-th residues from non n-th residues will be denoted by CR[n]. Deciding n-th residuosity is believed to be computationally hard and the hypothesis that there exists no polynomial time distinguisher for n-th residues modulo n^2 is referred to as the Decisional Composite Residuosity Assumption (DCRA).

We then begin to describe the numberic-theory framework underlying this cryptosystem. Let us first denote Carmichael's function on n as $\lambda(n) = \text{lcm}(p-1, q-1)$ or just λ for shorthand. Let g be some element of $Z^*_{n^2}$ and denote by $\mathcal{E}_g$ the integer-valued function defined by:

$$\begin{aligned} Z_n \times Z^*_n &\to Z^*_{n^2} \\ (x, y) &\to g^x \cdot y^n \bmod n^2 \end{aligned}$$

We denote $\mathcal{B}_\alpha \subset Z^*_{n^2}$ the set of elements of order $n\alpha$ and by $\mathcal{B}$ their disjoint union for $\alpha = 1, ..., \lambda$.

Definition 2. Assume that $g \in \mathcal{B}$. For $w \in Z_{n^2}^*$, we call n-th residuosity class of w with respect to g the unique integer $x \in Z_n$ for which there exists $y \in Z_n^*$ such that

$$\mathcal{E}_g(x, y) = w.$$

We denote the class of w as $[\![w]\!]_g$.

Definition 3. We call Composite Residuosity Class Problem the computational problem Class[n] defined as follows : given $w \in Z_{n^2}^*$ and $g \in \mathcal{B}$, compute $[\![w]\!]_g$.

In order to find the computational complexity of this definition, we state the following theorems that are proved in [2].

Theorem 1 . $Class[n] \Leftarrow Fact[n]$.

Theorem 2 . $Class[n] \Leftarrow RSA[n, n]$.

Theorem 3 . Let D-Class[n] be the decisional problem associated to Class[n] i.e. given $w \in Z_{n^2}^*, g \in \mathcal{B}$ and $x \in Z^*$, decide whether $x = [\![w]\!]_g$ or not. Then

$$CR[n] \equiv D\text{-}Class[n] \Leftarrow Class[n]$$

To conclude the computational hierarchy:

$$CR[n] \equiv D\text{-}Class[n] \Leftarrow Class[n] \Leftarrow RSA[n, n] \Leftarrow Fact[n]$$

We can then state that there exists no probabilistic polynomial time algorithm that solves the Composite Residuosity Class Problem, i.e. Class[n] is intractable. This conjecture is referred to as the Computational Composite Residuosity Assumption (CCRA). The proofs for the definitions, theorems and identities in this section can be found in Paillier's paper, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*. We shall now describe the encryption scheme based on this problem.

Key Generation:
1. Choose a large $n = pq$ that is the product of two large primes
2. Randomly select a base $g \in \mathcal{B}$

The public key of this encryption is (n, g) and the secret key is (p, q).

Encryption:
1. Prepare a message $m < n$
2. Choose a random $r < n$
3. Ciphertext $c = g^m \cdot r^n \bmod n^2$

Decryption:
1. Ciphertext $c < n^2$
2. $m(\text{plaintext}) = \frac{\mathrm{L}(c^\lambda \bmod n^2)}{\mathrm{L}(g^\lambda \bmod n^2)} \bmod n$

Where $\mathrm{L}(u) = \frac{u-1}{n}$ and λ is Carmichael's function.

This scheme is semantically secure if and only if the Decisional Composite Residuosity Assumption holds. This scheme is one-way if and only if the Computational Composite Residuosity Assumption holds.

Paillier's scheme has additive homomorphic properties as the two encryption functions, $m \to g^m r^n \bmod n^2$ and $m \to g^{nr+m} \bmod n^2$ are additively homomorphic on Z_n. This leads to the following identities:

$$\begin{aligned} \forall m_1, m_2 \in Z_n and k \in N \\ Dec(Enc(m_1)Enc(m_2) \bmod n^2) &= m_1 + m_2 \bmod n \\ Dec(Enc(m)^k \bmod n^2) &= km \bmod n \\ De(Enc(m_1)g^{m_2} \bmod n^2) &= m_1 + m_2 \bmod n \\ \left.\begin{array}{r} Dec(Enc(m_1)^{m_2} \bmod n^2) \\ Dec(Enc(m_2)^{m_1} \bmod n^2) \end{array}\right\} &= m_1 m_2 \bmod n \end{aligned}$$

3 Comparisons

Both El Gamal and Paillier encryption require the generation of a large element prime element. For El Gamal, the element p needs to be selected. This is done by choosing a large number x at random. If the number is even, replace x with $x + 1$ and apply the probabilistic test to check whether it is prime or not. If x is not a prime, replace x with $x+2$ until it is prime. There is expected to be $\mathcal{O}(\ln(x))$ numbers that are tested until a prime number is found. For Paillier encryption, this process occurs twice for both p and q to compute n.

El Gamal's complexity arises from the selection of the random element k that must be prime relative to $p - 1$. This can be achieved by choosing a large prime factor of the form $p = 2kq + 1$ where q is a large prime. The same process in choosing p is used to choose the number q. Then, a random k is generated that is of appropriate bit length. Apply probabilistic primality testing to $z = 2kq + 1$ and increment k by 1 until primality is achieved. This process is expected to test $\mathcal{O}(\ln(z))$ numbers.

Paillier's encryption complexity stems from the exponentiation of base g. Choosing a small g will expedite computations drastically. Computation of r^n or $g^{nr} \bmod n^2$ can be computed in advance. Computing $\mathrm{L}(u)$ for $u \in S_n$ during decryption can occur at very low cost by precomputing $n^{-1} \bmod 2^{|n|}$. The constant parameter $\mathrm{L}(g^\lambda \bmod n^2)^{-}1 \bmod n$ needs to only be precomputed once.

The tables below compare the security, size, and complexity of these two encryption schemes.

Schemes	Paillier Scheme	El Gamal Scheme
One-wayness	Class[n]	DH[p]
Semantic Sec	CR[n]	D-DH[p]
Plaintext Size	$\|n\|$	$\|p\|$
Ciphertext Size	$2\|n\|$	$2\|p\|$

Encryption	Paillier Scheme	El Gamal Scheme
$\|n\|, \|p\| = 512$	5120	1536
$\|n\|, \|p\| = 768$	7680	2304
$\|n\|, \|p\| = 1024$	10240	3072
$\|n\|, \|p\| = 1536$	15360	4680
$\|n\|, \|p\| = 2048$	20480	6144

Decryption	Paillier Scheme	El Gamal Scheme
$\|n\|, \|p\| = 512$	768	768
$\|n\|, \|p\| = 768$	1152	1152
$\|n\|, \|p\| = 1024$	1536	1536
$\|n\|, \|p\| = 1536$	2304	2304
$\|n\|, \|p\| = 2048$	3072	3072

Observing the tables reveal the similarity between decryption complexity and ciphertext size. The complexity of Paillier's encryption is significantly larger than that of El Gamal's.

4 Prêt á Voter

The additive homomorphic properties allow this scheme to be useful in voting protocols, threshold cryptosystems, watermarking, secret sharing schemes and more. In the next sections, we will discuss the application of this cryptosystem to the Prêt á Voter scheme. The key innovation of the Prêt á Voter scheme is that the vote is encoded in a random frame of reference or better represented as a randomized candidate list using a threshold scheme. In this way, the voter's choice does not need to be encoded and any communication with an encyrption device is completely avoided. Each ballot has a right-hand side with a unique ballot number that marks the selection of the voter. The left-hand side lists the candidates for the reader. The right-hand side is

casted when the voter turns in his or her ballot. The value printed at the bottom of this form is key to its evaluation and is called the "onion".

Each form has a unique, random secret seed value ρ drawn from some seed space S. The candidate permutation, π, is computed using the seed over a publicly agreed function $\sigma : S \to \Pi_C$ where C is the set of candidates and Π_C is a permutation over this set. Each ballot is a tuple with $(\pi, \{\rho\}_{PK_T})$ where PK_T is the threshold public key and π is the candidate permutation. The value ρ is kept secret. A receipt ballot has the form $(i, \{\rho\}_{PK_T})$ where i is the index value of the cell the voter placed their vote in.

5 Prêt á Voter with Paillier Encryption

We assume that the public key of the tellers, $PK_T = (n, g)$, is certified and publicised and that there is a publicly agreed function, σ, from the seed space into the set of permutations of the candidates. Each booth device creates a public key pair PK_b, SK_b and publishes PK_b. The form is given a random serial number ξ. The booth device is then fed this serial number in which the signing key SK_b is applied to generate a random string η that will be used to derive randomisation and the random seed. The device then computes π, the candidate order, and the onion value, θ, by encrypting the seed value, ρ, with the PK_T using the randomization ζ.

The seed and randomization are computed from: $< \rho, \zeta >:= \{\xi\}_{SK_b}$.
The candidate order as: $\pi := \sigma(\rho)$.
The onion value by: $\theta := \{\rho, \zeta\}_{PK_T}$.

The ballot forms will be generated by a set of l clerks in such a way that each contributes to the cryptographic values from which the candidate list is derived. We again assume that there is a set of tellers with key shares for a threshold Paillier algorithm with public key $PK_T : (g, n)$ and that booths have public keys $PK_B = (q, m)$ in order to distinguish the two. In order to generate w ballots at a given booth, the j-th clerk generates w sub-onion pairs: $\theta^T_{j,i}; \theta^B_{j,i}$ where

$\theta^T_{j,i} := \{s_{j,i}, x_{j,i}\}_{PK_T} = g^{s_{j,i}} \cdot (x_{j,i})^n \pmod{n^2}$ and
$\theta^B_{j,i} := \{s_{j,i}, y_{j,i}\}_{PK_B} = q^{s_{j,i}} \cdot (y_{j,i})^m \pmod{m^2}$.

The first term is the encryption of the j, i-th seed under the public teller's key and the second is the seed for the booth's public key. The randomisations of x and y should be independent. The full onions are formed by using the remaining un-audited rows of pairs, denoted as $\overline{A}_i$. The onions for the row i are then computed as:

$\Theta^T_i := \prod_{j \in \overline{A}_i} \theta^T_{j,i}$
$\Theta^B_i := \prod_{j \in \overline{A}_i} \theta^B_{j,i}$

This results in onions for which the seed value is: $s_i = \sum_{j \in \overline{A}_i} s_{j,i}$ and the randomisation by:
$x_i = \prod_{j \in \overline{A}_i} x_{j,i} \pmod{n^2}$
$y_i = \prod_{j \in \overline{A}_i} y_{j,i} \pmod{m^2}$.

The seed values and therefore, the candidate orders, remain encrypted and the ballots can be distributed in this form to be used in conventional pre-printed form. We introduce two new processes P_L and P_T. P_T takes a batch of forms and for each form, looks up the corresponding Θ_i^T where i is the index on the form. It re-encyrpts this and prints it on the right hand side of the form. Once P_T is finished, the ballots are shuffled, and passed to process P_L that holds the secret booth key. For a form carrying the j-th index, P_L looks up the appropriate Θ_j^B, decrypts this, and prints out the candidate order on the left hand side of the ballot. The resulting ballots are then sealed in individual envelopes for later use.

6 Conclusion

In this paper I overviewed the key generation, encryption, and decryption algorithms for the El Gamal and Paillier cryptosytems. El Gamal's security stems from the Diffie-Hellman problem and the underlying discrete logarithm assumption. Paillier's security stems from n-th residuosity's computational hardness that leads to the Decisional Composity Residuosity Assumption. We then examined the Composite Residuosity Class Problem that creates the Computational Composite Residuosity Assumption that adds extra security to Paillier's scheme. Comparing the two systems revealed the many similarities between the two but the difference in complexity when it comes to encrypting messages. The Prêt á Voter system for voting has many practical benefits and we explored how this style can be employed using Paillier's system. The additive homomorphic properiteis of Paillier's system can prove beneficial in many other practices and this paper only examines one of these situations.

References

[1] Hans Delfs and Helmut Knebl. *Introduction to Cryptography, Principles and Applications [Second Edition]*. Springer-Verlag, 2007.

[2] Pascal Paillier. *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*. Springer-Verlag, 1999.

[3] P.Y.A. Ryan, *Pret a Voter with Paillier encryption*. Mathematical and Computer Modelling, Volume 48, pp. 1646-1662. Elsevier, 2008.