Homomorphic Voting Based on Paillier Cryptosystem

Panagiotis Grontas

$\mu\Pi\lambda\forall$ - CoReLab Crypto Group

26/06/2013

Motivation

- Homomorphic property $E(v1) \otimes E(v2) = E(v1 \oplus v2)$
- In voting: Count the votes while maintaining secrecy
- Exponential ElGamal: $E(v, r) = (g^r, g^m h^r)$ instead of (g^r, mh^r)
- Decryption has to solve discrete log problem
  - Not as bad as it sounds
  - Difficulty depends on the message space
  - Problem on elections with multiple candidates
- Solutions based on quadratic or higher residuosity

Key Generation

- Choose two large primes p, q randomly and independently such that $gcd(pq, (p-1)(q-1)) = 1$
- Calculate RSA modulus $n = pq$
- Calculate $\boldsymbol{\lambda} = lcm(p-1, q-1) = \frac{(p-1)(q-1)}{gcd(p-1,q-1)}$ (Carmichael's Function)
  - Easy to calculate if we know p, q
  - $\forall x \in \mathbb{Z}_{n^2} : x^{\lambda(n)} = 1 \quad modn$
  - $\forall x \in \mathbb{Z}_{n^2} : x^{n\lambda(n)} = 1 \quad modn^2$
- Select generator $g \in \mathbb{Z}^*_{n^2}$
  - The order of g must be a non zero multiple of n
- Calculate inverse $\boldsymbol{\mu} = L(g^{\lambda} \,(mod\, n^2))^{-1} \,mod\, n$ where $L(x) = \frac{x-1}{n}$
  - $L()$ is given elements that are equal to $1 \quad modn$
  - $L()$ *'solves'* the discrete log problem and *'decrypts'*
  - Inverse always exists if g is a valid generator
- Public Key is (n, g) and private Key (λ, μ)
  - We can always select $g = n + 1$ so public key becomes n

Operation

Encryption

- Encode message m into $\mathbb{Z}_n$
- Select random $r \in \mathbb{Z}_n^*$
- Return $c = E_g(m, r) = g^m r^n \,(mod\, n^2)$

Decryption

- Ciphertext $c \in \mathbb{Z}_{n^2}^*$
- Return $m = L(c^\lambda \,mod\, n^2)\mu \,(mod\, n) = \frac{L(c^\lambda \,mod\, n^2)}{L(g^\lambda \,mod\, n^2)}(modn)$

Security

The composite residuosity problem

- Given $n = pq$ and $z \in \mathbb{Z}^*_{n^2}$ decide if z is $n - residue$ module n^2
- Does there exist $y \in \mathbb{Z}^*_{n^2}$ st: $z = y^n (mod n^2)$

Decisional composite residuosity assumption (DCRA): There is no polynomial time algorithm to decide the composite residuosity problem.

Remark: If there was an algorithm to decide if $z \in \mathbb{Z}^*_{n^2}$ is the encryption of message 0 then we could solve the composite residuosity problem

Correctness I

Target

Prove that for $c = E_g(m, r) = g^m r^n (mod n^2)$
the decryption operation $\frac{L(c^\lambda \, mod \, n^2)}{L(g^\lambda \, mod \, n^2)} (mod n)$ yields m.

Main Lemma

$\forall \, w \in Z^*_{n^2} : \boldsymbol{L(w^\lambda \, mod \, n^2) = \lambda [w]_{n+1} \, mod \, n}$

Notation

- $w = E_{n+1}([w]_{n+1}, r)$ which means w is the ciphertext and $[w]_{n+1}$ is the plaintext for $g = n + 1$
- $\lambda = lcm(p - 1, q - 1)$
- $L(x) = \frac{x-1}{n}$

Correctness II

Helper Lemma 1

$\forall x \in \mathbb{Z}_n : (1+n)^x = 1 + nx \ (\mathit{mod}\, n^2)$

Proof.

$$
\begin{aligned}
(1+n)^x &= \\
1 + \binom{x}{1} n + \binom{x}{2} n^2 + s + n^x \ (\mathit{mod}\, n^2) &= \\
1 + xn \ (\mathit{mod}\, n^2)
\end{aligned}
$$

$\square$

Correctness III

Helper Lemma 2

$\forall c \in \mathbb{Z}^*_{n^2}$, and proper generators g_1, g_2 : $\quad [c]_{g_1} = [c]_{g_2}[g_2]_{g_1}$

Proof.

$$
\begin{aligned}
g_2 = g_1^y b^n &\quad \sim \quad y = [g_2]_{g_1} \\
c = g_2^z d^n &\quad \sim \quad z = [c]_{g_2} \\
c = g_2^z d^n = (g_1^y b^n)^z d^n = g_1^{zy}(b^z d)^n &\quad \sim \quad yz = [c]_{g_1} \\
&\quad [c]_{g_1} = [c]_{g_2}[g_2]_{g_1}
\end{aligned}
$$

$\square$

Correctness IV

Main Lemma Proof

$\forall w \in Z^*_{n^2} : \boldsymbol{L(w^\lambda \, mod \, n^2) = \lambda[w]_{n+1} \, mod \, n}$

$n+1$ is a proper generator g

$\forall w \in Z^*_{n^2}$:

$$
\begin{aligned}
w = E_{n+1}([w]_{n+1}, r) = (n+1)^{[w]_{n+1}} r^n \, (mod n^2) \Rightarrow \\
w^\lambda = (n+1)^{\lambda[w]_{n+1}} r^{\lambda n} \, (mod n^2) \\
= \boldsymbol{(1 + \lambda[w]_{n+1} n)} r^{\boldsymbol{k\phi(n)} n} \, (mod n^2) \\
= (1 + \lambda[w]_{n+1} n) \\
L(w^\lambda) = \frac{w^\lambda - 1}{n} = \lambda[w]_{n+1}
\end{aligned}
$$

Correctness V

Decryption operation:

$$\frac{L(c^\lambda \, mod \, n^2)}{L(g^\lambda \, mod \, n^2)} =$$

$$\frac{\lambda[c]_{n+1}}{\lambda[g]_{n+1}} =$$

$$\frac{[c]_{n+1}}{[g]_{n+1}} = \frac{[c]_g [g]_{n+1}}{[g]_{n+1}} = [c]_g = m$$

Homomorphic Properties

- $D(E_g(m_1, r_1)E_g(m_2, r_2)) = m_1 + m_2 \quad mod\, n$
- $D(E_g(m_1, r_1)g^{m_2}) = m_1 + m_2 \quad mod\, n$ (a full encryption of the second message is not necessary)
- $D(E_g(m_1, r_1)g^{nx}) = m_1 + nx \quad mod\, n = m_1$ (self blinding)
- $D(E_g(m_1, r_1)^k) = km_1$

A Generalisation [DJ01] I

For each $s \geq 1$ we can define a cryptosystem CS_s:

Key Generation

Input: security parameter k

- Select admissible $n = pq$ with length k bits
- Choose random j with $gcd(j, n) = 1$ and random $x \in \mathbb{Z}_{n^s}$ and calculate $g = (1+n)^j x \quad mod n^{s+1}$
- Calculate $\lambda = lcm(p-1, q-1)$
- Select d such that
  - $d \, mod \, n \in \mathbb{Z}^*_{n^s}$
  - $d = 0 \quad (mod \lambda)$

Output: Public key $= (n, g)$ Private key $= d$

Remark: Paillier Cryptosystem is the special case $s = 1$

In Paillier $d = \lambda$ but larger values are preferred for threshold version to be secure.

A Generalisation [DJ01] II

Encryption

- Encode message m into $\mathbb{Z}_{n^s}$
- Select random $r \in \mathbb{Z}_n^*$
- Return $c = E_g(m, r) = g^m r^{n^s} \; (mod \, n^{s+1})$

A Generalisation [DJ01] III

Decryption

- Ciphertext $c \in \mathbb{Z}^*_{n^{s+1}}$
- Calculate $c^d \quad mod n^s = (1+n)^{mjd} \quad mod n^s$
- Extract mjd
- Calculate $g^d \quad mod n^s = (1+n)^{jd} \quad mod n^s$
- Extract jd
- $m = \frac{mjd}{jd}$

Security

$\forall s$ CS_s is one way if Paillier (CS_1) is one way and semantically secure iff the DCRA is true

A simplification I

Key Generation

- Public key is $n = pq$
- Private key is $\lambda = lcm(p-1, q-1)$

$g = (1+n)$ and s can be selected at any point in time as long as $m < n^s$

Encryption

- Encode message m into $\mathbb{Z}_n$
- Select random $r \in \mathbb{Z}_n^*$
- Return $c = E(m, r) = (1+n)^m r^{n^s} \pmod{n^{s+1}}$

A simplification II

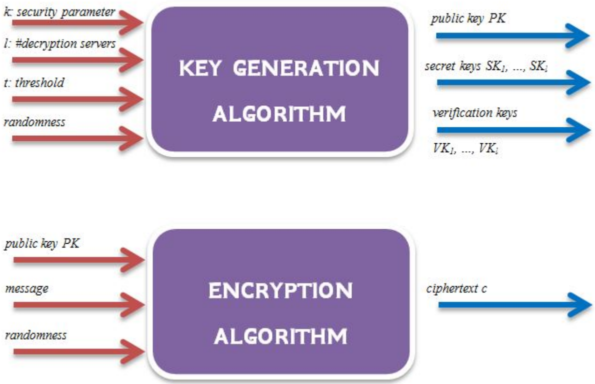
Decryption

- Ciphertext $c \in \mathbb{Z}^*_{n^2}$
- Calculate $c^\lambda mod n^{s+1} = (1+n)^{m\lambda \, mod n^s} r^{\lambda n^s) \, mod \lambda} \quad mod n^{s+1} = (1+n)^{m\lambda} \quad mod n^{s+1}$
- Extract $m\lambda$
- $m = \frac{m\lambda}{\lambda}$

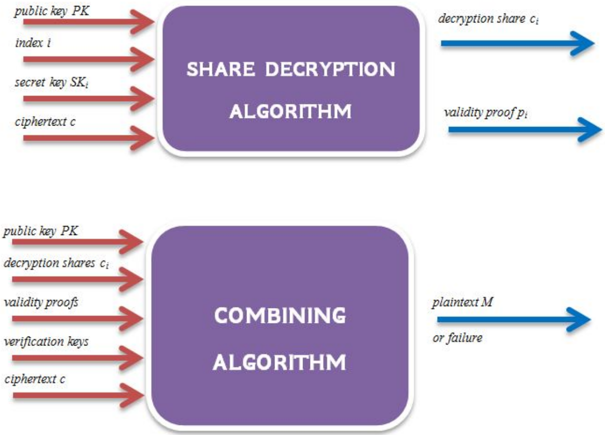
Security

$\forall s$ the simplified version is one way if Paillier (CS_1) is one way and semantically secure iff the DCRA is true.

Threshold decryption: a reminder I



Threshold decryption: a reminder II



Reminder:Shamir secret sharing

Objective

Share a secret element s between l players so that any $t+1$ subset can recover it, but no t element subset can.

- Main:Idea Lagrange Interpolation
- A polynomial P of degree t can be reconstructed from $t+1$ distinct elements $(x_i, y_i)_{i=1}^{t+1}$
- $P(x) = \sum_{i=1}^{t+1} \prod_{j=1, j\neq i}^{t+1} \frac{x-x_j}{x_i-x_j} y_i$
- To share a secret:
    - Dealer chooses a random polynomial of degree t so that $P(0) = s$
    - Distribute l pairs $(x_i, P(x_i)) x_i \neq 0$
    - $t+1$ players can reconstruct the polynomial (and recover s), but t players cannot

Reminder:Proof of Discrete Log Equality

Proof(α, b, A, B)

α, b are elements of a cyclic group G with generator g,
Prove that A, B have the same logarithm s in bases α, b

How *(Non Interactive Version)*

- Select a random element $r \in G$
- Calculate $x_\alpha = \alpha^r$ and $x_b = b^r$
- Generate $e = hash(\alpha, b, A, B, x_\alpha, x_b)$
- Calculate $t = r + es$
- Calculate $e' = hash(\alpha, b, A, B, \frac{\alpha^t}{A^e}, \frac{b^t}{B^e})$
- Check if $e = e'$

Threshold RSA [Sho00] I

- **Key Generation**
  - Calculate RSA modulus $n = pq$ where $p = 2p' + 1, q = 2q' + 1$ and $m = p'q'$. Notice that $4m = \phi(n)$
  - Choose prime $e > l$
  - Choose $d \in \mathbb{Z}_m$ st: $ed = 1 \quad mod m$
  - Share d using Shamir Secret Sharing
    - Polynomial $f(x) = \sum_{i=0}^{t} f_i x^i$
    - $f_0 = d$
    - $f_i \in_R \mathbb{Z}_m$
  - Secret Shares: $SK_i = d_i = f(i) \quad mod m$
  - Verification Keys:
    - $Q_n = \{x \in \mathbb{Z}_n^* | x = y^2 \pmod n\}$
    - $VK = v \in_R Q_n$ and
    - $VK_i = v^{d_i} \pmod n$

Threshold RSA [Sho00] II

- **Encryption**
  - $E(M) = M^e \pmod n$
- **Decryption shares**
  - Calculate $\Delta = l!$
  - Decryption shares: $c_i = c^{2\Delta d_i}$
  - Validity Proof $Proof(c^{4\Delta}, v, c_i^2 = (c^{4\Delta})^{d_i}, v^{d_i})$
- **Combination Preliminaries**
  - Validate proofs of decryption shares.
  - If t shares are valid at most then fail.
  - Set S a set of $t+1$ valid decryption shares
  - Lagrange coefficients multiplied by Δ: $\mu_{i,j}^S = \Delta \frac{\prod_{j'} (i-j')}{\prod_{j'} (j-j')}$ $\mu_{0,j}^S = \Delta \frac{\prod_{j'} (-j')}{\prod_{j'} (j-j')}$
  - $\Delta f(i) = \sum_j \mu_{i,j}^S f(j) \pmod m$

Threshold RSA [Sho00] III

- **Combination Algorithm**:Retrieve $d = f(0)$ and decrypt
  - Raise squares of shares c_j to $\mu^S_{0,j}$ for $j \in S$
  - Create product of above

$$w = \prod_j c_j^{2\mu^S_{0,j}} = \prod_j c^{(2\Delta d_j)(2\mu^S_{0,j})} = \prod_j (c^{4\Delta})^{d_j \mu^S_{0,j}} = (c^{4\Delta})^{\sum_j d_j \mu^S_{0,j}} = (c^d)^{4\Delta^2} = M^{4\Delta^2}$$

  - Using EGCD calculate α, b st: $\alpha 4\Delta^2 + be = 1$
  - Calculate $w^\alpha = M^{4\Delta^2 \alpha}$
  - Calculate $c^b = M^{eb}$
  - $M^{4\Delta^2} M^{eb} = M$

Threshold Paillier [FPS01] I

- **Key Generation**
  - Calculate RSA modulus $n = pq$ with $gcd(n, \phi(n)) = 1$ where $p = 2p' + 1, q = 2q' + 1$ and $m = p'q' = \frac{p-1}{2}\frac{q-1}{2}$
  - Generate g
    - Randomly choose $(\alpha, b) \in \mathbb{Z}_n^* \times \mathbb{Z}_n^*$
    - Set $g = (1+n)^{\alpha} b^n \pmod{n^2}$
  - Choose random element $\beta \in \mathbb{Z}_n^*$
  - Set secret key $SK = \beta m$
  - Shamir secret key sharing
    - $f_0 = SK$
    - Coefficients $f_i \in_R \{0, s, nm - 1\}$
    - Polynomial $f(x) = \sum_{i=0}^{t} f_i x^i \pmod{nm}$
    - Shares $s_i = f(i) \pmod{nm}$
  - Public Key
    - $\theta = L(g^{m\beta}) = \alpha m \beta \pmod{n}$
    - (n, g, θ)

Threshold Paillier [FPS01] II

- Verification Keys:
  - $Q_n = \{x \in \mathbb{Z}_n^* | x = y^2 \pmod n\}$
  - $VK = v \in_R Q_n$ and
  - $VK_i = v^{d_i} \pmod n$
- **Encryption**
  - $r \in_r \mathbb{Z}_n^*$
  - $E(M) = g^M r^N \pmod{n^2}$
- **Share Decryption**
  - $\Delta = l!$
  - Decryption shares: $c_i = c^{2\Delta s_i} \pmod{n^2}$
  - Validity Proof $Proof(c^{4\Delta}, v^{\Delta}, c_i^2 = (c^{4\Delta})^{s_i}, v^{s_i})$

Threshold Paillier [FPS01] III

- **Combination Preliminaries**
  - Validate proofs of decryption shares.
  - If t shares are valid at most then fail.
  - Set S a set of $t+1$ valid decryption shares
  - Lagrange coefficients multiplied by Δ:
  - $\mu^S_{i,j} = \Delta \frac{\prod_{j'}(i-j')}{\prod_{j'}(j-j')}$
  - $\mu^S_{0,j} = \Delta \frac{\prod_{j'} j'}{\prod_{j'}(j-j')}$
  - $\Delta f(0) = \sum_j \mu^S_{0,j} f(j) mod m$

Threshold Paillier [FPS01] IV

- **Combination Algorithm**
  - Raise squares of shares c_j to $\mu^S_{0,j}$ for $j \in S$
  - Create product of above
    - $w = \prod_j c_j^{2\mu^S_{0,j}} = \prod_j c^{(2\Delta s_j)(2\mu^S_{0,j})} = \prod_j (c^{4\Delta})^{s_j \mu^S_{0,j}} = (c^{4\Delta})^{\sum_j s_j \mu^S_{0,j}} = (c^{m\beta})^{4\Delta^2}$
  - But c is a Paillier encryption of message M. $c = g^M r^N$
    $(c^{m\beta})^{4\Delta^2} = ((g^M r^n)^{m\beta})^{4\Delta^2} = (1+n)^{\alpha 4\Delta^2 m\beta M}(br)^{nm\beta 4\Delta^2} \pmod{n^2} = (1+n)^{\alpha 4\Delta^2 m\beta M} \pmod{n^2} = 1 + n\alpha 4\Delta^2 m\beta M \pmod{n^2}$
  - Apply L function: $L(1 + n\alpha 4\Delta^2 m\beta M) = \alpha 4\Delta^2 m\beta M = M 4\Delta^2 \theta$
  - Divide by $4\Delta^2\theta$ (public information) and retrieve plaintext M.

Theorem

If the original Paillier cryptosystem is semantically secure then the threshold version is secure as well.

Proof of Knowledge Of Randomness I

Objective

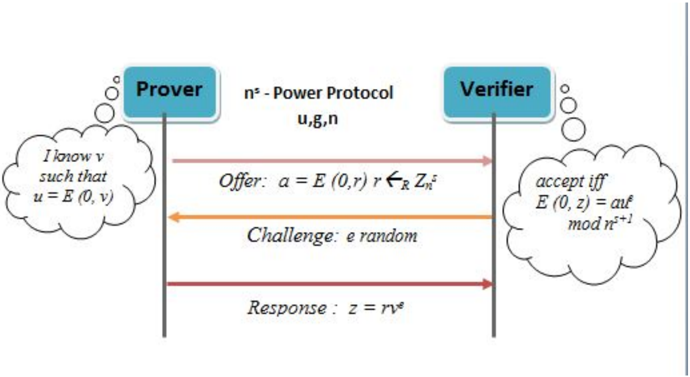
The prover presents a ciphertext c to the verifier and claims that it encrypts the message m, which means that the prover possesses randomness r st: $c = E(m, r)$

$$
\begin{aligned}
c = E(m, r) = g^m r^{n^s} \quad (mod\, n^{s+1}) &\Rightarrow \\
cg^{-m} = r^{n^s} \quad (mod\, n^{s+1}) &\Rightarrow \\
cg^{-m} = E(0, r)&
\end{aligned}
$$

We must prove that cg^{-m} is a n^s power

The protocol

Proof of Knowledge Of Randomness II



Completeness:

$$E(0,z) = E(0,rv^e) = E(0,r)E(0,v^e) = E(0,r)E(0,v)^e = \alpha u^e \quad (mod\, n^{s+1})$$

Homomorphic Tallying [DJN03] I

Yes-No Voting

- There are M voters
- Each voter decides on his vote v_i and calculate $E_i = E(v_i, r_i)$
- Compute ZK Proof of validity
- The authority(-ies) filter out the votes with invalid proofs
- Compute $E = \prod_i E_i = E(\sum_i v_i \quad mod n^2, \prod_i r_i \mod n)$
- The authority decrypts and receives the number of yes-votes $\sum_i v_i$
- The number of no-votes can be computed by subtracting from the total-number of valid votes.
- Remark: The tally must be less than n^2
- There are generalisations of Paillier for $n^s, s \geq 2$. One can choose s such that $M < n^s$

Homomorphic Tallying [DJN03] II

$L > 2$ candidates - A simple solution

- L parallel yes/no votes v_{ij}
- $v_{ij} = 1$ for the preferred candidates
- Proof of validity must include that the voter voted for exactly t candidates
- L parallel sums
- **Remarks**:
    - The vote size is large $O(Llog_2n)$
    - Many decryptions are needed

Homomorphic Tallying [DJN03] III

$L > 2$ candidates - A better solution [DJN03]

- Vote for candidate j - Encryption of M^j
- Vote for t candidates: Submit many encryptions
- $M^L < n^s$
- Tallying: All encrypted votes are multiplied
- The result is of the form $\alpha = \sum \alpha_j M^j$ where α_j is the number of votes cast for candidate j
- The result is a number in $M - \alpha ry$ notation
  - The vote size is $O(log_2 L log_2 n)$
  - One decryption is needed
  - An extra proof must be employed to deter voting for the same candidate t times

References I

Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard.

Practical multi-candidate election system.

In *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, PODC '01, pages 274–283, New York, NY, USA, 2001. ACM.

Sansar Choinyambuu.

Homomorphic tallying with paillier cryptosystem, 2009.

Ivan Damgård and Mats Jurik.

A generalisation, a simplification and some applications of paillier's probabilistic public-key system.

In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*, PKC '01, pages 119–136, London, UK, UK, 2001. Springer-Verlag.

References II

Ivan Damgard, Mads Jurik, and Jesper Buus Nielsen.
A generalization of paillier's public-key system with applications to electronic voting.
P Y A RYAN, page 3, 2003.

Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern.
Sharing decryption in the context of voting or lotteries.
In *Financial Cryptography*, pages 90–104. Springer, 2001.

Michael O�Keeffe.
The paillier cryptosystem - a look into the cryptosystem and its potential application, 2008.

References III

Pascal Paillier.

Public-key cryptosystems based on composite degree residuosity classes.

In *Proceedings of the 17th international conference on Theory and application of cryptographic techniques*, EUROCRYPT'99, pages 223–238, Berlin, Heidelberg, 1999. Springer-Verlag.

Victor Shoup.

Practical threshold signatures.

In *Proceedings of the 19th international conference on Theory and application of cryptographic techniques*, EUROCRYPT'00, pages 207–220, Berlin, Heidelberg, 2000. Springer-Verlag.

Tobias Volkhausen.

Paillier cryptosystem: A mathematical introduction, 2006.