HOMOMORPHIC ENCRYPTION

Ψηφιακή ασφάλεια και ιδιωτικότητα

Γεώργιος Σπαθούλας

Msc Πληροφορική και υπολογιστική βιοιατρική

Πανεπιστήμιο Θεσσαλίας

IDEA OF HOMOMORHPIC ENCRYPTION

PLAYING WITH ENCRYPTED DATA

- Suppose Alice gives Bob a **securely encrypted computer file** and asks him to **sum a list of numbers** she has put inside
- Without the decryption key, this task also seems impossible
- The encrypted file is just as opaque and impenetrable as a locked suitcase
- **“Can’t be done”** Bob concludes again.
- But Bob is **wrong**

IT CAN BE DONE

- Because Alice has chosen a **very special encryption scheme**, Bob **can carry out her request**
- **He can compute with data he can't inspect**
- The numbers in the file remain encrypted at all times, so Bob cannot learn anything about them
- Nevertheless, he can run computer programs on the encrypted data, performing operations such as summation
- The output of the programs is also encrypted; Bob can't read it
- But when he gives the results back to Alice, she can extract the answer with her decryption key

FULLY HOMOMORPHIC ENCRYPTION

- The technique that makes this magic trick possible is called **fully homomorphic encryption**, or **FHE**
- It’s not exactly a new idea, but for many years it was viewed as a fantasy that would never come true
- That changed in 2009, with a breakthrough discovery by Craig Gentry, who was then a graduate student at Stanford University (He is now at IBM Research)
- Since then, further refinements and more new ideas have been coming at a rapid pace
- Homomorphic encryption is not quite ready for everyday use
- The methods have been shown to work in principle, but they still impose a **heavy penalty of inefficiency**

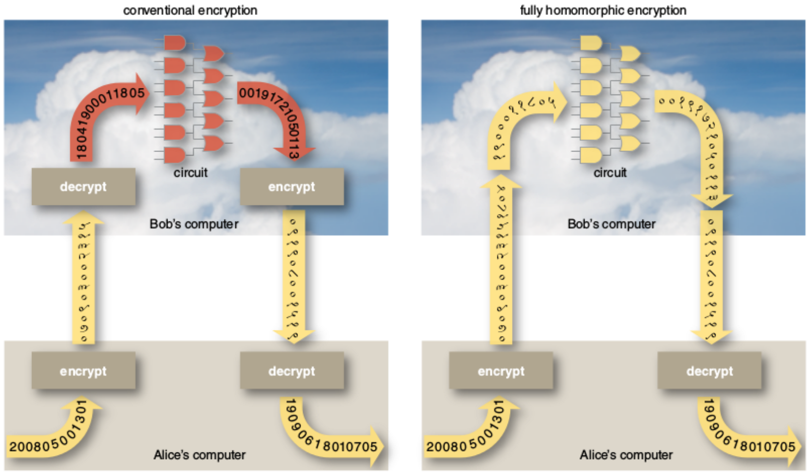
LOOKING TO THE FUTURE

- If the system can be made more practical, however, there are applications ready and waiting for it
- Many organizations are eager to outsource computation: Instead of maintaining their own hardware and software, they would like to run programs on servers "in the cloud," a phrase meant to suggest that physical location is unimportant
- But letting sensitive data float around in the cloud raises concerns about security and privacy
- Practical homomorphic encryption would address those worries, protecting the data against eavesdroppers and intruders and even hiding it from the operators of the cloud service

FHE USABILITY

- Cryptographic technology has become a routine part of life on the Internet
- When you check your bank balance on the Web, or make an online purchase encryption is behind the scenes
- Even Google searches are encrypted
- These measures are meant to protect your messages while they are in transit
- On the other hand, the cryptographic protocols conceal nothing from the recipients of your messages, who have the keys to decipher them
- Usually, that’s just fine, because the intended recipient is a trusted party
- Homomorphic encryption is the tool for those occasions when you don’t trust anyone, not even Bob

FHE / CONVENTIONAL CRYPTOGRAPHY

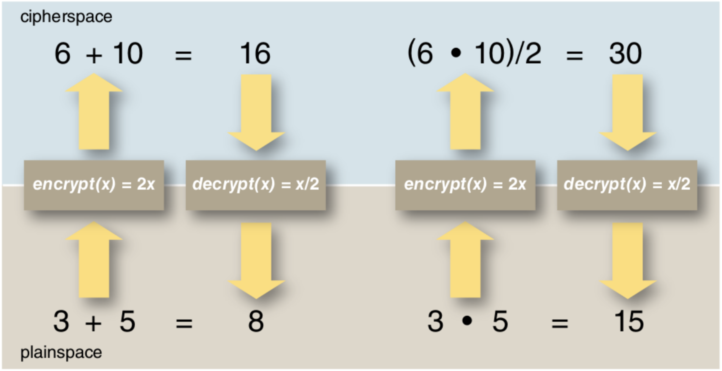


Alice has confidential data she wants to process on Bob's computer, which is a server "in the cloud." But she wants to make sure no one else gains access to the data—not even Bob. Conventional encryption *(left)* protects her information while it is in transit but not while the computation is underway on Bob's computer *(red portion of pathway)*. Homomorphic encryption *(right)* offers security from the moment the data stream leaves Alice's computer until it returns. The strategy requires that all the arithmetical and logical operations needed in the computation (symbolized here by a circuit of Boolean gates) be applied to the encrypted form of the data. In this diagram the distinction between encrypted and unencrypted data—between ciphertext and plaintext—is suggested by a typographic convention: The ciphertext is shown in numerals of the Devanagari alphabet, which have the sequence ०१२३४५६७८९.

IDEA OF HOMOMORPHISM

- The Greek world **homomorphic** translate as **same shape** or **same form**, and the underlying idea is that of **a transformation that has the same effect on two different sets of objects**
- Homomorphic cryptography offers a similar pair of pathways :
  - We can do arithmetic directly on the plaintext in puts x and y
  - Or we can encrypt x and y, apply a series of operations to the ciphertext values, then decrypt the result to arrive at the same final answer
- Among the many operations on numbers we might consider, it turns out that adding and multiplying are all we really need to do; other computations can be expressed in terms of these primitives

EXAMPLE



The concept of homomorphism describes a parallel linkage between operations on two sets of objects. In this toy example the sets of objects are the set of all integers *(lower panel)* and the set of even integers *(upper panel)*. The operations on the objects are addition and multiplication. Going back and forth between the two sets is just a matter of doubling or halving a number. Addition works the same way in both sets. In the case of multiplication, an adjustment is needed: For even numbers, the product of x and y is defined as $(x \cdot y)/2$. These sets and operations can be pressed into service as a rudimentary homomorphic cryptosystem. Plaintext integers are encrypted by doubling; then any sequence of additions and multiplications can be carried out; finally the result is decrypted by halving.

FULLY HOMOMORPHIC ENCRYPTION

HISTORY

- The idea of computing with encrypted data was first proposed in **1978** by **Ron Rivest**, **Len Adleman** and **Michael Dertouzos** at MIT
- Just a few months before, **Rivest** and **Adleman**, along with **Adi Shamir**, had introduced the first implementation of a public-key crypto system, which came to be known as **RSA** after their initials
- The basic **RSA** scheme is **partially homomorphic**: It allows multiplication of ciphertexts but not addition
- In the next 30 years there were occasional advances on this front
- For example, in **2005 Boneh, Goh** and **Nissim** devised a homomorphic system that allowed an unlimited number of additions on the ciphertext, followed by a single multiplication
- In spite of such incremental progress, however, **Gentry's** announcement of a **fully homomorphic scheme** came as a total surprise in **2009**

GENTRY'S FHE OUTLINE

- He creates a crypto system with the usual **encrypt** and **decrypt** functions, which convert bits from plaintext to ciphertext and back
- He also builds an **evaluate function** that accepts a description of a computation to be performed on the ciphertext
- The computation is specified not as a sequential program but as a **circuit or network**, where input signals pass through a cascade of logic gates
- Such circuits are most often assembled from **Boolean gates** (and, or, not, etc.), but they can also be specified in terms of **addition and multiplication steps**

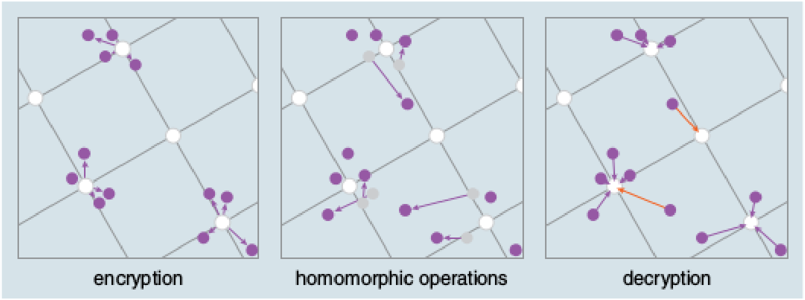
BARRIER

- The evaluate function amounts to a complete computer embedded in the cryptosystem
- In principle, it can calculate any computable function, provided that the circuit representing the function is allowed to extend to arbitrary depth
- The depth of a circuit is the number of gates on the longest path from input to output and a full-powered computer must be able to handle circuits of arbitrary depth
- Here the homomorphic system runs into a **barrier**
- The problem is that ciphertext data are contaminated with numerical **“noise”—slight discrepancies** from their ideal values
- Every arithmetic operation **amplifies the noise**, until eventually it **overwhelms the signal**

NOISE

- The origin of the noise lies in the **probabilistic encryption process**
- Think of each ciphertext **value** as a **point in space**
- The probabilistic encrypt function **injects** a smidgen of **randomness** into each of the point's coordinates, **displacing it slightly** from the position it would occupy in a deterministic cryptosystem
- The **decrypt** function **filters out the noise** by treating each point as if it were located at the nearest unperturbed position
- When the noise is **amplified** by homomorphic computations, however, the point **wanders farther from its correct position**, until finally the decrypt function will associate it with an **incorrect plaintext value**

NOISE AMPLIFICATION



Random "noise" in a secure cryptosystem is the principal impediment to homomorphic operation. Encrypted data can be envisioned as points *(purple disks)* that are given small random displacements from a finite set of lattice points *(white disks)*. On decryption, each purple disk is attracted to the nearest white lattice point. Homomorphic operations amplify the random displacements. If the noise level exceeds a threshold, some of the disks gravitate to the wrong lattice point, leading to an incorrect decryption *(red arrows)*. Without some means of noise control, the system can support only a limited number of homomorphic operations.

AND IF WE COULD RESET RANDOMENESS...

- Roughly speaking, **each homomorphic addition doubles the noise**, and **each multiplication squares it**
- Hence the number of operations must be limited or errors will accumulate
- Because of the **limit on circuit depth**, this version of the cryptosystem cannot be called **fully homomorphic** but only **somewhat homomorphic**
- The depth limit could be evaded in the following way: **Whenever the noise begins to approach the critical threshold, decrypt the data and then re-encrypt it, thereby resetting the noise to its original low level**
- The **trouble** is, **decryption requires the secret key**, and the whole point of FHE is to allow computation in a context where that key is unavailable

ENCRYPTING THE ENCRYPTION KEY

- This is where the story gets wacky and wonderful
- The evaluate function built into the cryptosystem is **capable of performing any computation, provided it does not exceed the noise limit on circuit depth**
- So we can ask evaluate to run the **decrypt function**
- Evaluate is designed to work with encrypted data, so it is supplied with an **encrypted version of the normal key**
- Specifically, the secret key supplied to evaluate is the **ciphertext produced when encrypt is applied to the plaintext of the secret key**
- When decrypt is run with this enciphered key, **the result is not plaintext but a new encryption of the ciphertext, with reduced noise**

THE KEY IS LOCKED BY ITSELF

- In effect, Alice is giving Bob a **copy of the key needed to unlock the data**, but the key is inside a **securely locked box** and can only be used within that box
- As a matter of fact, the box is locked with the very key that is locked inside the box!
- The pause to re-encrypt and refresh the noisy ciphertext can be repeated as needed
- In this way the computer **can handle a circuit of any finite depth**, and the system becomes fully homomorphic
- It can carry out arbitrarily complex computations on encrypted data

LIMITING THE DEAPTH OF DECRYPT

- An essential assumption in this scheme is that **the decrypt circuit is it self shallow enough** to run without exceeding the noise threshold
- Indeed, its depth needs to be a little less than the limit, or else the computer will spend all its time refreshing the data and will never accomplish any useful work
- When Gentry first formulated his FHE scheme, he found that **this condition was not met**
- The evaluate function could not run the decrypt routine without accumulating excessive noise
- The remedy was **a technique for "squashing" decrypt, at the cost of making the key larger and more complicated**
- With this last innovation, the problem was solved

LETS TAKE A STEP BACK… RSA … PAILLIER

RSA MULTIPLICATIVELY HOMOMORHPIC

Ο αλγόριθμος RSA παρουσιάζει την εξής ιδιότητα:

$$D(E(m_1) * E(m_2) mod n^2) = m_1 * m_2 mod n \tag{1}$$

- Έστω Public key (e,N) = (66617,76201) και Private key (d,N) = (4553,76201)
- Εάν $m_1 = 66624$ και $m_2 = 18532$
- $m_1 * m_2 mod n = 67366$
- $E(m_1) = 64959$ και $E(m_2) = 6778$
- $c_1 * c_2 mod n^2 = 440292102$
- $D(c_1 * c_2 mod n^2) = 67366$

PAILLIER ADDITIVELY HOMOMORHPIC

Ο αλγόριθμος Paillier παρουσιάζει τις εξής ιδιότητες:

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

$$D(E(m)^k mod n^2) = k * m mod n \tag{3}$$

$$D(E(m_1) * g^{m_2} mod n^2) = m_1 + m_2 mod n \tag{4}$$

ΣΥΣΤΗΜΑ ΨΗΦΟΦΟΡΙΑΣ

Βάσει της ιδιότητας 1 της προηγούμενης διαφάνειας λειτουργεί το εξής σύστημα ψηφοφορίας

- Σε μία εκλογική διαδικασία συμμετέχουν N_c υποψήφιοι
- Έστω ότι υπάρχουν N_v ψηφοφόροι
- Ο κάθε ψηφοφόρος μπορεί να ψηφίσει όσους θέλει από τους υποψήφιους ή και κανέναν

ΕΠΙΛΟΓΗ ΒΑΣΗΣ

Επιλέγουμε έναν αριθμό βάση b έτσι ώστε $b > N_v$ και αναπαριστούμε την ψήφο σε κάθε υποψήφιο με τον εξής αριθμό :

1ος υποψ. : b^0
2ος υποψ. : b^1
....
N_c υποψ. : b^{N_c-1}

Ο ψηφοφόρος αθροίζει τους αριθμούς που σχετίζονται με τους υποψηφίους της επιλογής του

Εάν κάποιος επιλέξει όλους τους υποψηφίους τότε ο αριθμός στον οποίο αθροίζονται οι προτιμήσεις του είναι

$$m_{max} = \sum_{i=1}^{N_c} b^{i-1} \tag{5}$$

ΠΑΡΑΔΕΙΓΜΑ

- Έστω ότι έχουμε N_c = 3 υποψηφίους και N_v = 9 ψηφοφόρους
- Υπολογίζουμε την βάση του συστήματος ως $b = N_v + 1 = 10$
    - Η ψήφος στον 1ο γίνεται $10^0 = 1$
    - Η ψήφος στον 2ο γίνεται $10^1 = 10$
    - Η ψήφος στον 3ο γίνεται $10^2 = 100$

ΠΑΡΑΔΕΙΓΜΑ

- Έστω ότι οι 9 ψηφοφόροι ψήφισαν ως εξής :
    - Ο 1ος ψήφισε τον 2ο υποψήφιο $\rightarrow$ 10
    - Ο 2ος ψήφισε τον 1ο υποψήφιο $\rightarrow$ 1
    - Ο 3ος ψήφισε τον 3ο υποψήφιο $\rightarrow$ 100
    - Ο 4ος ψήφισε τον 1ο υποψήφιο $\rightarrow$ 1
    - Ο 5ος ψήφισε τον 3ο υποψήφιο $\rightarrow$ 100
    - Ο 6ος ψήφισε τον 3ο υποψήφιο $\rightarrow$ 100
    - Ο 7ος ψήφισε τον 1ο υποψήφιο $\rightarrow$ 1
    - Ο 8ος ψήφισε τον 2ο υποψήφιο $\rightarrow$ 10
    - Ο 9ος ψήφισε τον 3ο υποψήφιο $\rightarrow$ 100
- Αθροίζοντας τις ψήφους προκύπτει το 423
- Τα ψηφία 3,2,4 αντιστοιχούν στις ψήφους των τριών υποψηφίων

ΜΕΓΙΣΤΟ ΑΘΡΟΙΣΜΑ

- Πρόκειται να αθροίσουμε τις ψήφους όλων των ψηφοφόρων
- Μας ενδιαφέρει το μέγιστο δυνατό άθροισμα

$$T_{max} = N_v * m_{max} \tag{6}$$

Το άθροισμα αυτό αντιστοιχεί στο ενδεχόμενο να ψηφίσουν όλοι οι ψηφοφόροι όλους τους υποψηφίους

ΕΠΙΛΟΓΗ P,Q

- Για να είναι εφικτή η λειτουργία του αλγόριθμος Paillier θα πρέπει το n να είναι μεγαλύτερο από το μέγιστο δυνατό απλό κείμενο οπότε
$$n > T_{max}, n = p * q \tag{7}$$
- Οπότε επιλέγουμε κατάλληλα τα p,q π.χ.
$$p, q \geq \sqrt{T_{max} + 1} \tag{8}$$
- Στην συνέχεια επιλέγονται/υπολογίζονται οι τιμές για τις υπόλοιπες παραμέτρους του αλγορίθμου

ΚΡΥΠΤΟΓΡΑΦΗΣΗ

- Ο κάθε ψηφοφόρος κρυπτογραφεί τον αριθμό που υπολόγισε αθροίζοντας τις ψήφους του
- Επιλέγει ένα τυχαίο $r \in Z_n^*$ και υπολογίζει το κρυπτοκείμενο

$$E(m_i) = c_i = g^{m_i} * r_i^n mod n^2 \tag{9}$$

- Στην συνέχεια στέλνει το κρυπτόγραμμα c_i στην αρχή που διενεργεί την ψηφοφρία

ΥΠΟΛΟΓΙΣΜΟΣ ΑΠΟΤΕΛΕΣΜΑΤΟΣ

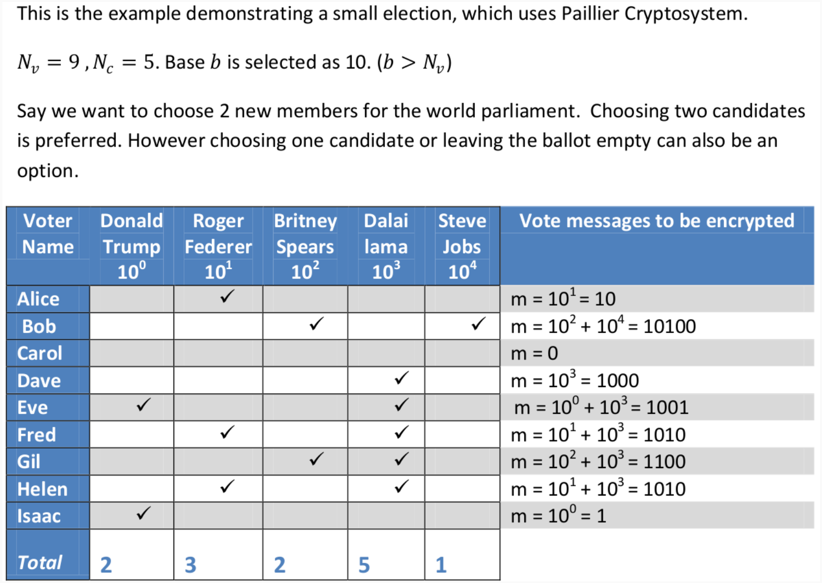
- Στην συνέχεια η αρχή που διενεργεί τις εκλογές πολλαπλασιάζει όλα τα c_i που έλαβε από τους ψηφοφόρους
- Ο πολλαπλασιασμός αυτός αντιστοιχεί στην πράξη της πρόσθεσης στο πεδίο των απλών μυνηματών

$$T = \prod_{i=1}^{N_v} c_i \bmod n^2 \tag{10}$$

- στην συνέχεια αποκρυπτογραφείται το αποτέλεσμα και θα πρέπει να προκύψει το άθροισμα των αρχικών ψήφων

$$D(T) = \sum_{i=1}^{N_v} m_i \bmod n \tag{11}$$

TALLYING EXAMPLE



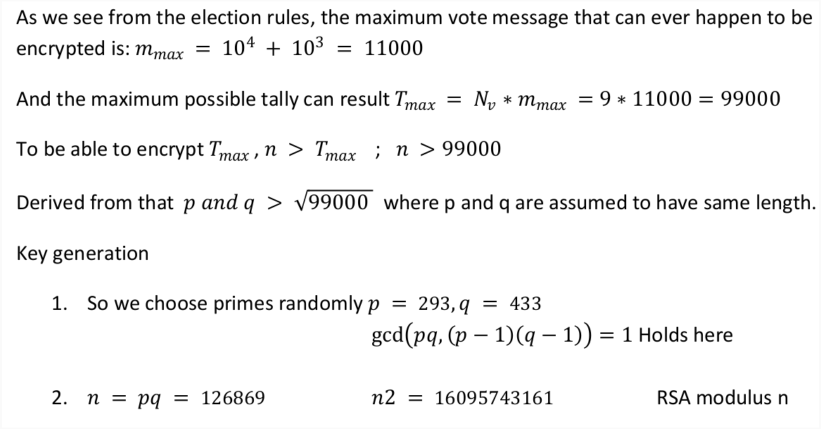
This is the example demonstrating a small election, which uses Paillier Cryptosystem.

$N_v = 9\,, N_c = 5$. 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	

TALLYING EXAMPLE



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} , $n > T_{max} \;\; ; \;\; n > 99000$

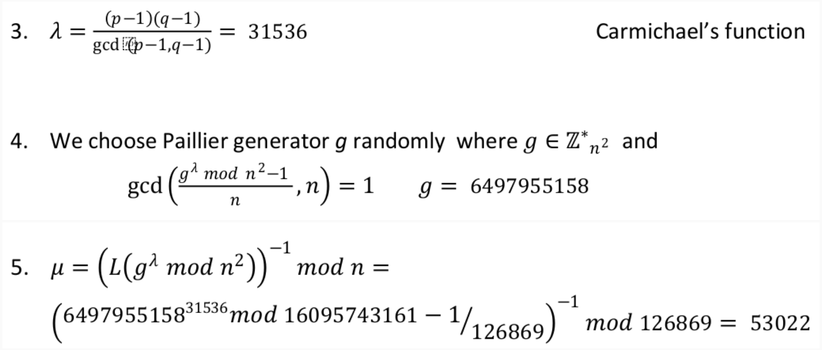
Derived from that $p \; and \; q > \sqrt{99000}$ where p and q are assumed to have same length.

Key generation

1. So we choose primes randomly $p = 293, q = 433$
   $\gcd\big(pq, (p-1)(q-1)\big) = 1$ Holds here

2. $n = pq = 126869$ $\quad$ $n2 = 16095743161$ $\quad$ RSA modulus n

TALLYING EXAMPLE



3. $\lambda = \frac{(p-1)(q-1)}{\gcd(p-1,q-1)} = 31536$ Carmichael's function

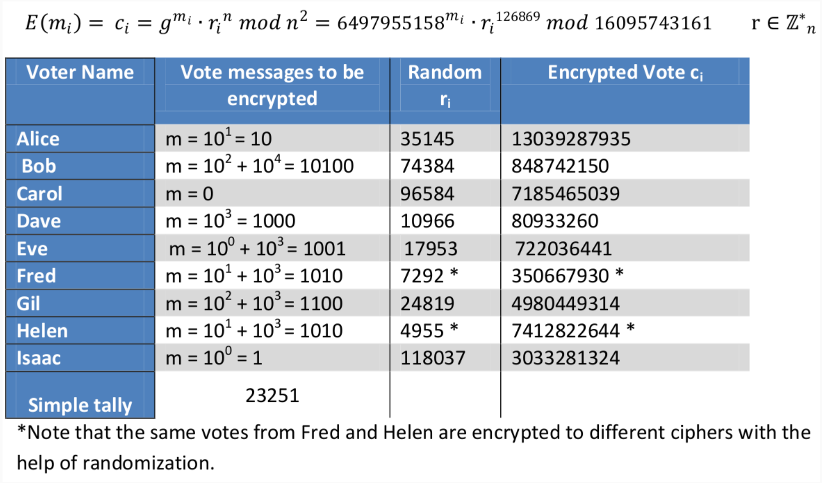
4. We choose Paillier generator g randomly where $g \in \mathbb{Z}^*{}_{n^2}$ and

$$\gcd\left(\frac{g^{\lambda}\ mod\ n^2-1}{n}, n\right) = 1 \qquad g = 6497955158$$

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

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

TALLYING EXAMPLE

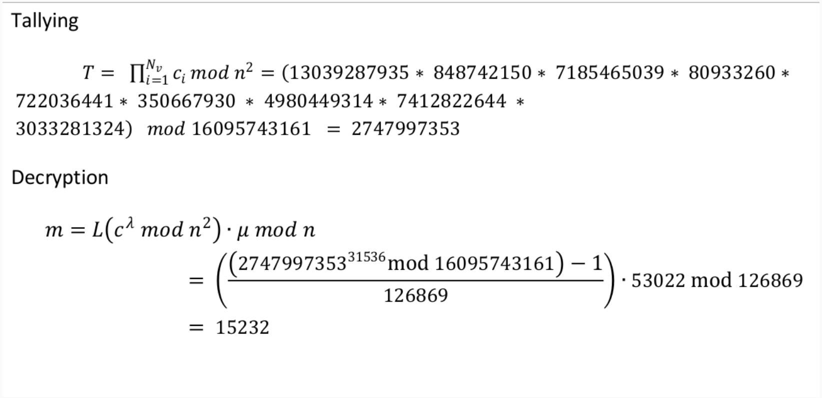


$$E(m_i) = c_i = g^{m_i} \cdot {r_i}^n \; mod \; n^2 = 6497955158^{m_i} \cdot {r_i}^{126869} \; mod \; 16095743161 \qquad r \in \mathbb{Z}^*{}_n$$

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 EXAMPLE



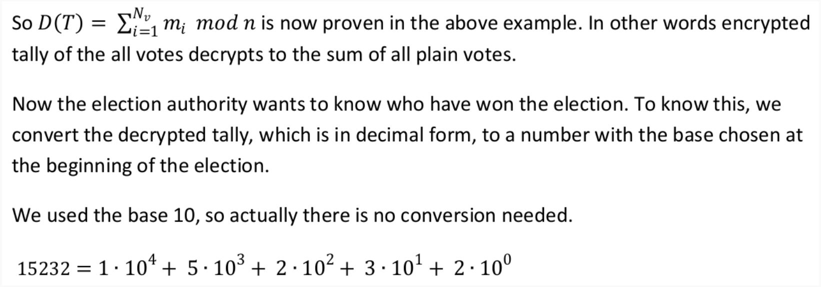
Tallying

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

Decryption

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

TALLYING EXAMPLE



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

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

BIBLIOGRAPHY

- **Alice and Bob in Cipherspace**, Brian Hayes
- **Homomorphic Tallying with Paillier Cryptosystem**, Sansar Choinyambuu
- **The development of homomorphic cryptography**, Sigrun Goluch