Multiplexing schemes for homomorphic cryptosystems

Juan Camilo Corena*, Jaime Andrés Posada**

Fecha de recepción: 18 de mayo de 2010
Fecha de aprobación: 12 de junio de 2010

Abstract. We present in this article two secure multiplexing and demultiplexing schemes that use homomorphic properties from known public key cryptosystems. One scheme is suitable for cryptosystems with additive homomorphic properties such as Paillier and Benaloh cryptosystems. The proposed scheme employs a modification of Hadamard codes to generate a set of orthogonal codes over $\mathbb{Z}_3$. The other one is suitable for cryptosystems with multiplicative homomorphic properties such as RSA and ElGamal. Both schemes might be used in voting and auction systems where anonymity of the individuals is crucial.

Resumen. Presentamos en este artículo dos esquemas seguros de multiplexación y demultiplexación que utilizan propiedades homomórficas de algunos sistemas de cifrado de llave pública. Un esquema es adecuado para sistemas con propiedades aditivas como Paillier y Benaloh. El esquema propuesto utiliza una modificación de los códigos de Hadamard para generar un conjunto de códigos ortogonales sobre $\mathbb{Z}_3$. El otro esquema es adecuado para sistemas con homomorfismos multiplicativos como RSA y ElGamal. Ambos esquemas podrían ser utilizados en sistemas de votación y en subastas donde el anonimato de los individuos es crucial.

Keywords: homomorphic encryption, orthogonal vectors, secure auctions.

Palabras Clave: cifrado homomórfico, vectores ortogonales, subastas seguras.

* Juan Camilo Corena. M. Sc. Ingeniería de Sistemas, Universidad de los Andes. investigacion@juancamilocorena.com

** Jaime Posada, M.A. en Matemáticas UW-Madison, se desempeña como docente de tiempo completo del Departamento de Matemáticas del Politécnico Grancolombiano desde el año 2006. japosada@poli.edu.co.

1. Introduction

Vector orthogonality has been widely used in communications to allow senders to multiplex their messages simultaneously and receivers to demultiplex the desired data. One of these methods is synchronous CDMA, which exploits the orthogonality between vectors representing information. To achieve orthogonality, synchronous CDMA uses binary orthogonal vectors based on Hadamard codes, but this is not suitable for most public key cryptosystems since those cryptosystems use positive integer and not vectors as a way to represent data. Even though one could cipher these vectors in each of their components, this approach might be vulnerable to chosen ciphertext attacks. Thus it would be desirable to balance these two aspects to achieve orthogonality in order to allow several individuals to add encrypted information into a single data stream, and being able to retrieve it later.

In order to allow these individuals to add information to a secure data stream, we considered two case scenarios in this article. One is intended to be used in algorithms with multiplicative homomorphisms in which case the proposed solution is straightforward using unique factoring into primes. The second scenario deals with cryptosystems with additive homomorphisms in which case we propose the employment of modified Hadamard codes over $\mathbb{Z}_3$. The proposed schemes can be used in elections, auctions and in any scenario where sensitive information is to be provided by several individuals and a third party is to process the individual components.

Although homomorphic encryption has been used in many cryptographic protocols applied to elections and auctions such as [1,4,6,7], and later succeeded by protocols such as [2] as a mechanism for protecting voters privacy and the integrity of the election. To check the validity of encrypted information a proof of knowledge is issued with a vote in a ballot. This proof will confirm that a given ballot adds one and only one vote to the final tally. These proofs are outside the tallying itself, which allows election officials to remove malicious ballots if they are cast. Our scheme allows one to verify the correctness of the tally without such proofs, at the cost of reduced number of ballots and larger key sizes. Our proposal achieves similar results as the afore mentioned scheme with a different approach, that could be used to simplify hardware implementations by joining two stages of the process.

The article is organized in four main topics. In the first one we summarize several cryptosystems with homomorphic properties along with some basic principles about CDMA multiplexing. In the second we introduce the set of codes used by our scheme and we present it and finally we discuss computational viability in Paillier's cryptosystem and some possible applications aimed at auction systems.

2. Homomorphic encryption

Homomorphic encryption is a form of encryption where it is possible to perform an algebraic operation on the plaintext by performing a possibly different operation

on the ciphertext. This form of encryption has been used widely in many contexts, and in the next section we introduce four different cryptosystems that have two different kinds of homomorphisms: multiplicative or additive.

2.1. Multiplicative homomorphisms

RSA Cryptosystem. Let $n = pq$ where p, q are two distinct prime numbers. Then compute $\phi(n) = (p-1)(q-1)$. Now choose $e \in \mathbb{Z}^*_{\phi(n)}$ and $e > 0$ such that $\gcd(e, \phi(n)) = 1$, the next step involves determining e^{-1} in $\mathbb{Z}^*_{\phi(n)}$, we will call this number d which is a number that satisfies $ed \equiv 1 \pmod{\phi(n)}$. The public key is (n, e) and the private one is (n, d). To encrypt a plaintext m we compute

$$c \equiv m^e \pmod{n} \tag{1}$$

We call the resulting encryption function E. To decrypt, calculate

$$m \equiv c^d \pmod{n} \tag{2}$$

This algorithm's homomorphism can be deduced as follows:

$$\begin{aligned} E(m_1) \cdot E(m_2) = m_1^e m_2^e \pmod{n} &= (m_1 m_2)^e \pmod{n} \\ &= E(m_1 \cdot m_2) \end{aligned} \tag{3}$$

which can be rephrased as: given the product of two ciphertexts encrypted with the same key, once they are decrypted the result will be the product of the plain texts generating the ciphertexts modulo n. See [11] for more details regarding this algorithm.

ElGamal Cryptosystem. Let G be a cyclic group of prime order q with generator g. Then a random $x \in \mathbb{Z}_q$ is chosen to be the secret key. The public key is (G, q, g, h), where $h = g^x$. Given a plain text $m \in G$ the encryption function E is:

$$E(m) = (g^r, mh^r) \tag{4}$$

where $r \in \mathbb{Z}_q$ is randomly chosen. To decrypt a message of type (c_1, c_2) using the secret key x, first calculate $s = c_1^x$ and then recover the original message m as follows:

$$m = \frac{c_2}{s} \tag{5}$$

This algorithm's homomorphism can be deduced as follows:

$$\begin{aligned} E(m_1) \cdot E(m_2) &= (g^{r_1}, m_1 h^{r_1})\,(g^{r_2}, m_2 h^{r_2}) \\ &= \left(g^{r_1+r_2}, m_1 m_2 h^{r_1+r_2}\right) = E(m_1 \cdot m_2) \end{aligned} \tag{6}$$

which can interpreted as: product of two ciphertexts encrypted with the same key, once they are decrypted the result will be the product of the plain texts generating the ciphertexts. See [8] for more details regarding this algorithm.

2.2. Additive homomorphisms

Benaloh Cryptosystem. Choose a blocksize r and choose primes p and q such that r divides $(p-1)$ and $\gcd(q-1,r)=1$. Let $n=pq$, and choose $y \in \mathbb{Z}^*_{\phi(n)}$ such that $y^{\frac{\phi(n)}{r}} \not\equiv 1 \pmod{n}$. The public key is (y,n,r) and the private key (p,q). Given a plain text $m \in \mathbb{Z}_r$ the encryption function E is:

$$E(m) = y^m u^r \pmod{n} \tag{7}$$

where $u \in \mathbb{Z}^*_n$ is randomly chosen. To decrypt a message $c = y^m u^r \pmod{n}$ note that

$$c^{\frac{\phi(n)}{r}} = y^{m\frac{\phi(n)}{r}} \pmod{n} \tag{8}$$

and so in order to find m, one can to solve the discrete logarithm by exhaustive search, or using the Baby-step, Giant-step method because m is small in practice.

This algorithm's homomorphism can be deduced as follows:

$$\begin{aligned} E(m_1)\cdot E(m_2) &= \left(y^{m_1}{u_1}^r\right)\left(y^{m_2}{u_2}^r\right) = y^{m_1+m_2}(u_1u_2)^r \\ &= E\left(m_1+m_2 \pmod{\phi(n)/r}\right) \end{aligned} \tag{9}$$

which can be rephrased as: given the product of two ciphertexts encrypted with the same key, once they are decrypted the result will be the sum of the plain texts generating the ciphertexts modulo $\phi(n)/r$. See [5] for more details regarding this algorithm.

Paillier Cryptosystem. Let $n=pq$ where p, q are two distinct prime numbers. Then compute $\lambda = \operatorname{lcm}(p-1, q-1)$. Choose $g \in \mathbb{Z}^*_{n^2}$ such that n divides the order of g. The public key is (n,g) and the private key (p,q). Given a plain text $m \in \mathbb{Z}_n$ the encryption function E is:

$$E(m) = g^m r^n \pmod{n^2} \tag{10}$$

where $r \in \mathbb{Z}^*_n$ is randomly chosen. To decrypt a message $c \in \mathbb{Z}^*_{n^2}$, compute:

$$m = \left(L\left(c^\lambda \pmod{n^2}\right)\right)\left(L\left(g^\lambda \pmod{n^2}\right)\right)^{-1} \pmod{n} \tag{11}$$

where $L(u) = (u-1)/n$. This algorithm's homomorphism can be deduced as follows:

$$\begin{aligned} E(m_1)\cdot E(m_2) &= \left(g^{m_1}{r_1}^n\right)\left(g^{m_2}{r_2}^n\right) = g^{m_1+m_2}(r_1r_2)^n \\ &= E\left(m_1+m_2 \pmod{n}\right) \end{aligned} \tag{12}$$

which can be rephrased as: given the product of two ciphertexts encrypted with the same key, once they are decrypted the result will be the sum of the plain texts generating the ciphertexts modulo n. See [10] for more details regarding this algorithm.

3. CDMA multiplexing

3.1. Hadamard codes

Hadamard codes are used for signal error detection and correction. They were successfully used in space probes such as Mariner, Viking or Voyager to send pictures from other planets back to earth. Valid codewords are the rows of H and $-H$, where H is a Hadamard matrix (a square matrix whose entries are either 1 or -1 and whose rows are pairwise orthogonal) and where each -1 is replaced by 0. To illustrate this, given the following Hadamard matrix,

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{13}$$

the codewords are $(1,1,1,1)$, $(1,0,1,0)$, $(1,1,0,0)$, $(1,0,0,1)$, $(0,0,0,0)$, $(0,1,0,1)$, $(0,0,1,1)$ and $(0,1,1,0)$.

3.2. Codes over $\mathbb{Z}_3$

We modify the construction of Hadamard codes to generate recursively the following set of codes over $\mathbb{Z}_3$:

1. (1) is a valid code.
2. If $\boldsymbol{v}$ is a valid code, then $\boldsymbol{v}^\frown\boldsymbol{v}$ and $\boldsymbol{v}^\frown\neg\boldsymbol{v}$ are valid codes, where $^\frown$ denotes concatenation of vectors, and $\neg\boldsymbol{v}$ denotes the vector obtained from $\boldsymbol{v}$ by interchanging the digits 1 and 2, or by calculating additive inverses over $\mathbb{Z}_3$.

We have then that (1), $(1,1)$, $(1,2)$, $(1,1,1,1)$, $(1,1,2,2)$, $(1,2,1,2)$, $(1,2,2,1)$ are all valid codes, and in general a full binary tree T is constructed such that in its n^{th} level T_n there are 2^n codes of length 2^n. These codes have the following properties used in the next section to describe our variation of the synchronous CDMA technique.

Theorem 1. *If $n \geq 1$, then all codes of T_n are orthogonal over $\mathbb{Z}_3$. If n is odd, and $\boldsymbol{v} \in T_n$, then $\|\boldsymbol{v}\| \equiv 2 \pmod 3$. If n is even, and $\boldsymbol{v} \in T_n$, then $\|\boldsymbol{v}\| \equiv 1 \pmod 3$.*

Proof. By induction: $(1,1)$ and $(1,2)$ are orthogonal since $(1,1) \bullet (1,2) = 3 \equiv 0 \pmod 3$. Let $\boldsymbol{v}, \boldsymbol{w} \in T_{n+1}$, $n \geq 1$. There are four cases:

1. $\boldsymbol{v} = \boldsymbol{x}^\frown\boldsymbol{x}$, $\boldsymbol{w} = \boldsymbol{y}^\frown\boldsymbol{y}$, where $\boldsymbol{x}, \boldsymbol{y} \in T_n$.
   In this case $\boldsymbol{v} \bullet \boldsymbol{w} = \boldsymbol{x} \bullet \boldsymbol{y} + \boldsymbol{x} \bullet \boldsymbol{y} = 0 + 0 \equiv 0 \pmod 3$
2. $\boldsymbol{v} = \boldsymbol{x}^\frown\boldsymbol{x}$, $\boldsymbol{w} = \boldsymbol{y}^\frown\neg\boldsymbol{y}$, where $\boldsymbol{x}, \boldsymbol{y} \in T_n$.
   In this case $\boldsymbol{v} \bullet \boldsymbol{w} = \boldsymbol{x} \bullet \boldsymbol{y} + \boldsymbol{x} \bullet \neg\boldsymbol{y} = \boldsymbol{x} \bullet \boldsymbol{y} - \boldsymbol{x} \bullet \boldsymbol{y} = 0 - 0 \equiv 0 \pmod 3$
3. $\boldsymbol{v} = \boldsymbol{x}^\frown\neg\boldsymbol{x}$, $\boldsymbol{w} = \boldsymbol{y}^\frown\boldsymbol{y}$, where $\boldsymbol{x}, \boldsymbol{y} \in T_n$.
   In this case $\boldsymbol{v} \bullet \boldsymbol{w} = \boldsymbol{x} \bullet \boldsymbol{y} + \neg\boldsymbol{x} \bullet \boldsymbol{y} = \boldsymbol{x} \bullet \boldsymbol{y} - \boldsymbol{x} \bullet \boldsymbol{y} = 0 - 0 \equiv 0 \pmod 3$

4. $\boldsymbol{v} = \boldsymbol{x}^\frown\neg\boldsymbol{x}$, $\boldsymbol{w} = \boldsymbol{y}^\frown\neg\boldsymbol{y}$, where $\boldsymbol{x}, \boldsymbol{y} \in T_n$.
   In this case $\boldsymbol{v} \bullet \boldsymbol{w} = \boldsymbol{x} \bullet \boldsymbol{y} + \neg\boldsymbol{x} \bullet \neg\boldsymbol{y} = \boldsymbol{x} \bullet \boldsymbol{y} + \boldsymbol{x} \bullet \boldsymbol{y} = 0 + 0 \equiv 0 \pmod 3$

On the other hand, since $1^2 = 2^2 \equiv 1 \pmod 3$, then any given code $\boldsymbol{v} \in T_n$ of length 2^n satisfies the following:

$$\|\boldsymbol{v}\| \equiv 2^n \pmod 3 \tag{14}$$

Using another induction argument we have that

$$2^n \equiv \begin{cases} 2 \pmod 3 & n \text{ odd} \\ 1 \pmod 3 & n \text{ even} \end{cases} \tag{15}$$

and so it follows the second part of the theorem.

3.3. Synchronous CDMA

If several transmitters want to send information simultaneously over a single channel, there are various techniques used to accomplish that, but we focus ourselves in synchronous CDMA, which exploits the orthogonality between vectors representing information. Our technique is basically synchronous CDMA, but we use the codes constructed in section 3.2. More details about CMDA can be found in [12].

Suppose that k transmitters $t_1, t_2, \ldots, t_k$ want to send binary vectors s_i, $1 \leq i \leq k$, of length m over a single channel. Let n be such that $k \leq 2^n$, and to each transmitter t_i, a unique code $\boldsymbol{v}_i \in T_n$ is assigned. If the data to be transmitted in the vector s_i is a one, then the vector $\boldsymbol{v}_i$ is transmitted, and if a zero is to be transmitted, then $\neg\boldsymbol{v}_i$ is transmitted.

For example, if t_i is assigned code $\boldsymbol{v}_i = (1, 2)$, and wishes to send the binary vector $(1, 0, 1)$, the actual transmission is $(\boldsymbol{v}_i, \neg\boldsymbol{v}_i, \boldsymbol{v}_i) = ((1, 2), (2, 1), (1, 2))$

In this way each transmitter t_i generates a signal $\sigma_i = (x_1^i, x_2^i, \ldots, x_m^i)$ where each x_l^i is either $\boldsymbol{v}_i$ or $\neg\boldsymbol{v}_i$, to be transmitted over a single channel. To accomplish that purpose, the signals σ_i, $1 \leq i \leq k$, are to be added over $\mathbb{Z}_3$ to create the following multiplexed signal:

$$\Lambda_M = \sum_{i=1}^{k} \sigma_i \pmod 3 \tag{16}$$

To demultiplex the signal Λ_M and recover a particular signal σ_i, the orthogonality of the codes is used: Since the codes $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are orthogonal, then

$$(x_1^j \bullet \boldsymbol{v}_i, x_2^j \bullet \boldsymbol{v}_i, \ldots, x_m^j \bullet \boldsymbol{v}_i) = (0, 0, \ldots, 0) \tag{17}$$

and so when we calculate the dot product of the code $\boldsymbol{v}_i$ with each component of the signal $\Lambda_M = (x_1^1, x_2^1, \ldots, x_m^1) + \cdots + (x_1^i, x_2^i, \ldots, x_m^i) + \cdots + (x_1^k, x_2^k, \ldots, x_m^k)$ we obtain

$$(x_1^i \bullet \boldsymbol{v}_i, x_2^i \bullet \boldsymbol{v}_i, \ldots, x_m^i \bullet \boldsymbol{v}_i) \tag{18}$$

Each x_l^i is either $\boldsymbol{v}_i$ or $\neg\boldsymbol{v}_i$, so each component of the previous vector is either $\boldsymbol{v}_i \bullet \boldsymbol{v}_i = \|\boldsymbol{v}_i\|$ or $\boldsymbol{v}_i \bullet \neg\boldsymbol{v}_i = -\|\boldsymbol{v}_i\|$. Thus a vector with digits $1, 2$ is obtained. Since the magnitude of a code is based on the parity of n, there are two procedures to recover the binary vector s_i:

1. n is odd. In this case by theorem 1, $\|\boldsymbol{v}_i\| = 2$ and so the r^{th} component $x_r^i \bullet \boldsymbol{v}_i$ of the vector

$$(x_1^i \bullet \boldsymbol{v}_i, x_2^i \bullet \boldsymbol{v}_i, \ldots, x_m^i \bullet \boldsymbol{v}_i) \tag{19}$$

   is 2 if and only if a one was to be transmitted in the r^{th} position of the vector s_i.
2. n is even. In this case by theorem 1, $\|\boldsymbol{v}_i\| = 1$ and so the r^{th} component $x_r^i \bullet \boldsymbol{v}_i$ of the vector

$$(x_1^i \bullet \boldsymbol{v}_i, x_2^i \bullet \boldsymbol{v}_i, \ldots, x_m^i \bullet \boldsymbol{v}_i) \tag{20}$$

   is 1 if and only if a one was to be transmitted in the r^{th} position of the vector s_i.

To give an example of this process, suppose transmitter α is assigned code $(1, 1)$ and wishes to send message $(1, 0, 1)$, and transmitter β is assigned code $(1, 2)$ and wishes to send message $(1, 1, 0)$. Then

$$\sigma_\alpha = ((1, 1), (2, 2), (1, 1)) \quad \sigma_\beta = ((1, 2), (1, 2), (2, 1)) \tag{21}$$

and so

$$\Lambda_M = ((2, 3), (3, 4), (3, 2)) = ((2, 0), (0, 1), (0, 2)) \pmod 3 \tag{22}$$

To recover σ_α we calculate the dot product of the code $(1, 1)$ with each component of the signal Λ_M to get $(2, 1, 2)$. In this case $n = 1$, so a 2 means that a one was to be transmitted. The recovered message is $(1, 0, 1)$. To recover σ_β, the dot product of the code $(1, 2)$ with each component of the signal Λ_M is $(2, 2, 4) = (2, 2, 1) \pmod 3$. The recovered message is $(1, 1, 0)$.

4. Proposed schemes

We propose the schemes based on the following requirements:

1. The scheme is able to multiplex a set of data identified by specific codes.
2. The multiplexed information can be sent over a secure channel without leaking any information about the individual components.
3. It is possible for external agent having a proper cryptographic key and a valid code to verify if a given information was sent over the secure channel.

4.1. Multiplicative homomorphisms

Given a set of individuals $S = \{s_1, s_2, \ldots, s_n\}$, each s_i wishing to send a message m_i in the set $M = \{m_1, \ldots, m_n\}$ we assign a prime number p_i to each sender s_i. These prime numbers will serve as codes for identifying each s_i within the multiplexed message. To multiplex the information we compute

$$\Lambda = \prod_{i=1}^{n} E\left(p_i^{m_i}\right) \tag{23}$$

where E is an encryption function with multiplicative homomorphic properties such as RSA or ElGamal. To demultiplex the information we revert the encryption function by computing

$$\Gamma = E^{-1}\left(\prod_{i=1}^{n} E(p_i^{m_i})\right) = \prod_{i=1}^{n} p_i^{m_i} \tag{24}$$

Then, to recover the message m_i sent by individual s_i one has to find the largest power of p_i dividing Γ.

Regarding the requirements we intended for our scheme, the first requirement is true based on the fundamental theorem of arithmetic. The second requirement is satisfied since we used E which is assumed to be a secure cryptographic algorithm such as RSA or ElGamal. For the last requirement one has to check if p_i divides Γ, this is an easy task for an individual knowing the proper private key and a code p_i as long as m_i is bounded.

4.2. Additive homomorphisms

Given a set of individuals $S = \{s_1, s_2, \ldots, s_n\}$, each s_i wishing to send a message m_i in the set $M = \{m_1, \ldots, m_n\}$ we assign a unique code $\boldsymbol{v}_i \in T_k$ as explained in section 3.2, where k is such that $n \leq 2^k$.

Given a signal σ_i, which is a vector of vectors, each component being $\boldsymbol{v}_i$ or $\neg\boldsymbol{v}_i$, then $\bar{\sigma}_i$ denotes the vector obtained from σ_i by concatenating its components into a single vector. For example, if $\sigma_\alpha = ((1,1),(2,2),(1,1))$ then $\bar{\sigma}_\alpha = (1,1,2,2,1,1)$.

Let B be a positive integer base such that $B \geq 2^{k+1} + 1$. B is chosen this way since the largest value one can have in a given position when adding numbers in base B with only digits 1 and 2 is $2n \leq 2^{k+1}$. Let $[\sigma_i]_B$ be the following number in base B

$$[\sigma_i]_B = \sum_{i=1}^{l} \gamma_{l+1-i} B^{l+1-i} \tag{25}$$

where γ_i is the i^{th} component of the vector $\bar{\sigma}_i$, and l is its length. For example, if $\sigma_\alpha = ((1,1),(2,2),(1,1))$ then $[\sigma_\alpha]_B = 1B^5 + 1B^4 + 2B^3 + 2B^2 + 1B + 1$.

To multiplex the information we compute

$$\Lambda = \prod_{i=1}^{n} E\left([\sigma_i]_B\right) \tag{26}$$

where E is an encryption function with additive homomorphic properties such as Benaloh or Paillier. To demultiplex, first compute

$$\Gamma = E^{-1}\left(\prod_{i=1}^{n} E\left([\sigma_i]_B\right)\right) = \sum_{i=1}^{n} [\sigma_i]_B \tag{27}$$

Since the base B is large enough, then $\sum_{i=1}^{n} [\sigma_i]_B$ will behave as vector sum, and so this will allow us to reassemble the original CDMA signal. To accomplish that, one has to compute $\sum_{i=1}^{n} \sigma_i$ by reversing the process that constructed $[\sigma_i]_B$ from σ_i as follows: first think of $[\sigma_i]_B$ as a vector $\bar{\sigma}_i$ and then split this vector into a vector of vectors of length 2^k to get σ_i.

For example, if

$$\sigma_\alpha = ((1,1),(2,2),(1,1)) \quad \sigma_\beta = ((1,2),(1,2),(2,1)) \tag{28}$$

then $B = 5$, $[\sigma_\alpha]_B = 1B^5 + 1B^4 + 2B^3 + 2B^2 + 1B + 1$ and $[\sigma_\beta]_B = 1B^5 + 2B^4 + 1B^3 + 2B^2 + 2B + 1$. In this case the recovered Γ is

$$\Gamma = 2B^5 + 3B^4 + 3B^3 + 4B^2 + 3B + 2 \tag{29}$$

To reassemble the original CDMA signal, first we think of Γ as the vector $(2,3,3,4,3,2)$, and then we split this vector into the vector $((2,3),(3,4),(3,2))$. Finally, this vector modulo 3 is $((2,0),(0,1),(0,2))$. To recover each individual messages $(1,0,1)$, $(1,1,0)$ sent by α, and β respectively, one proceeds as the final example in section 3.2.

Regarding the requirements we intended for our scheme, the first requirement is true based on the results obtained in theorem 1. The second requirement is satisfied since we used E which is assumed to be a secure cryptographic algorithm such as Paillier. For the last requirement one has to check if $[\sigma_i]_B$ is one of the components of Γ, this is an easy task for an individual knowing the proper private key and a code $[\sigma_i]_B$.

5. Computational viability

Given a set of codes $C \subseteq T_k$, a base B is chosen such that $B \geq 2^{k+1} + 1$ as in section 4.2. To estimate the size of the keys p, q needed in Paillier's cryptosystem one proceeds as follows. If each one of the individuals s_i want to send one bit message, the worst case scenario is given by the input $\boldsymbol{v}_m = (2, 2, \ldots, 2)$ which in base B is

$$[\boldsymbol{v}_m]_B = 2\frac{B^{2^k} - 1}{B - 1} \tag{30}$$

for example: if $k = 7$ there are 128 possible codes and so B might be 257. In this case

$$\log_2\left([\boldsymbol{v}_m]_B\right) \approx 1018 \tag{31}$$

so two primes p, q of length 509 (in bits) will be needed to guarantee $[\boldsymbol{v}_m]_B < pq$. Considering that a key size of 1024 bits is the recommended security standard for algorithms such as RSA, the proposed scheme can be used in practice with reasonable scalability.

For messages consisting of more than 1 bit, key sizes become rather large, so the scheme would not be practical to carry more information. Despite this limitation, the scheme is suitable for elections and auctions situations since interaction among participants can be easily modeled by binary responses. One such application is presented in the next section.

6. Possible applications

The proposed scheme has several applications in the field of secure voting and secure auctions. In this section we describe one possible way to implement anonymous auctions based on Paillier's cryptosystem.

Let S be the seller and $B = \{b_1, b_2, \ldots, b_m\}$ be the set of bidders, each bidder b_i generates the following set of pairs of codes $C_i = \{(a_{ij}, d_{ij}) : 1 \leq j \leq r\} \subseteq T_n$ where n is large enough to accommodate all the bidders, r is the maximum number of rounds in the auction and the sets C_i are pairwise disjoint. In the auction context, a_{ij} is meant to serve as an acceptance of the offer to a given price set by the seller and d_{ij} as declining the offer. Each set C_i is digitally signed with b_i's public key and sent to the seller in a secure way so the seller can keep track of the auction process.

The auction will have several rounds; in the j^{th} round S fixes a price and the bidders are required to submit an answer accepting or declining the offer. To this purpose a bidder b_i sends $(E_{P_S}(a_{ij}), \Sigma_i(a_{ij}))$ if he/she is willing to pay for the fixed price and sends $(E_{P_S}(d_{ij}), \Sigma_i(d_{ij}))$ otherwise. Here E_{P_S} is Paillier's cryptosystem encrypting function with a public key belonging to S and Σ_i is a digital signature function for b_i.

The seller verifies the digital signatures for every bidder against the data sent during the setup stage to confirm the identity of the bidder, then computes

$$E^{-1}\left(\prod_{j=1}^{m} E_{P_S}(x_{ij})\right) = \sum_{j=1}^{m} x_{ij} \tag{32}$$

where x_{ij} is the answer issued by b_i. Then the seller recovers the individual components x_{ij} as in section 4.2; the winner is decided when there is only one positive response x_{ij}. Since dot products usually have lower computational complexity than decryption, the advantage of this approach lies in the fact that a single decryption operation is needed.

This scheme guarantees privacy of the bidders, anonymity of them except for the seller, non repudiation for bids and validity in the process, conditions for secure auctions as stated in [9].

7. Related work

Other applications where homomorphic additive cryptosystems are central, include elections or voting systems. Another technique for counting votes is known as a multi-counter [3], which consists of generating a counter for each available candidate. To every candidate a segment of n continuous bits is assigned, so the segment is able to accommodate at most $2^n - 1$ votes. If there are m candidates, then mn bits would be needed to keep track of all the votes. When a voter casts a vote for a particular candidate, his choice is added to the segment assigned to that candidate. Arithmetic within each segment is performed without interfering with the other segments. For example, in an election with 3 candidates A, B, C, and 3 potential voters, each candidate is assigned with 2 bits. In case there were 28 voters instead of just 3, the number of bits assigned to each candidate has to be incremented from 2 to 5. The aggregated result is shown in figure 1 where two votes were cast for A, none for B and one for C.

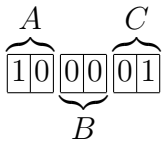

Fig. 1. An example tally with two votes for A, none for B, and one for C.

Attacks to systems based on the previous counting method are possible, these include: adding a vote several times to increase the count for a particular candidate and subtracting votes via additive inverses. To thwart these kind of attacks, proofs of the value contained are generated to check for the validity of a ballot. However this kinds of integrity checking routines are performed as an additional procedure. Regarding this, our system can be used to check the validity of an election, since introducing a value outside the assigned set for a given voting place, immediately invalidates the count, thus revealing the presence of ballot tampering. This is achieved by adding another layer of error checking based on the orthogonality of the chosen codes.

8. Conclusions

This article presented an alternative way to multiplex information in several known cryptosystems with homomorphic properties. Even though the proposed scheme for multiplicative homomorphism is not practical, the scheme presented for additive homomorphisms is practical and presents advantages over other known ways to achieve the same result in terms of computational complexity to detect fraud in elections or auctions.

References

1. Abe, M. and Suzuki, K.: M+1-st price auction using homomorphic encryption. Public Key Cryptography, pages 115–124, (2002)
2. Adida, B. and Rivest, R. L.: Scratch & vote: self-contained paper-based cryptographic voting. WPES '06: Proceedings of the 5th ACM workshop on Privacy in electronic society, pages 29–40, New York, NY, USA, ACM. (2006)
3. Baudron O., Fouque P. A., Pointcheval, D., Stern, J. and Poupard G.: Practical multi-candidate election system. PODC '01: Proceedings of the twentieth annual ACM symposium on Principles of distributed computing, pages 274–283, New York, NY, USA, ACM. (2001)
4. Benaloh, J. C. and Tuinstra, D.: Receipt-free secret-ballot elections (extended abstract). In *STOC*, pages 544–553, (1994)
5. Clarkson, J. B.: Dense probabilistic encryption. Proceedings of the Workshop on Selected Areas of Cryptography, pages 120–128, (1994)
6. Cramer, R., Gennaro, R. and Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. EUROCRYPT, pages 103–118, (1997)
7. Cramer, R. J., Franklin, M., Schoenmakers, L. A. and Yung, M.: Multi-authority secret-ballot elections with linear work. Technical report, Amsterdam, The Netherlands, (1995)
8. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, **31**(4):469–472, (1985)
9. Kikuchi, H., Harkavy, M. and Tygar, J. D.: Multi-round anonymous auction protocols. Proceedings of the First IEEE Workshop on Dependable and Real-Time E-Commerce Systems, pages 62–69. Springer-Verlag, (1999)
10. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. EUROCRYPT, pages 223–238, (1999)
11. Rivest, R. L., Shamir A., and Adleman, L. M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM, **21**(2):120–126, (1978)
12. Viterbi, A. J.: CDMA: principles of spread spectrum communication. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, (1995)