Finding Minimum Optimal Path Securely Using Homomorphic Encryption Schemes in Computer Networks

Levent Ertaul
Department of Mathematics & Computer Science
California State University, East Bay
Hayward, CA, USA.
levent.ertaul@csueastbay.edu

Vaidehi
Department of Mathematics & Computer Science
California State University, East Bay
Hayward, CA, USA.
vaidehikedlaya@gmail.com

Abstract - *In this paper we find a secure routing protocol for computer networks, which finds minimum optimum path using homomorphic encryption schemes. We briefly look into the existing homomorphic encryption algorithms. We make use of ElGamal encryption, Elliptic Curve encryption and a privacy homomorphism, which exhibits the property of homomorphism in our new routing protocol. Elliptic curve exhibits the property of additive homomorphism and is computationally faster than ElGamal and RSA. However, the privacy homomorphism using mod operation is computationally much faster than both ElGamal and Elliptic Curve. Using the homomorphic property of these encryption algorithms, we propose three new protocols, which are ElGamal, Elliptic Curve and Privacy Homomorphism to find the minimum optimal path securely. These protocols provide confidentiality.*

Keywords - ElGamal Encryption, Elliptic Curve Encryption, Privacy Homomorphism, Determining minimum optimal path.

1. Introduction

The routing algorithm decides which line the packet should be transmitted to. In a wireless environment the route keeps on changing, so we should dynamically select the route to transmit the packet. Using homomorphic encryption scheme we can securely find a minimum path in these networks.

Homomorphic [1], [2], [3] encryption scheme can be implemented in routing protocols to enhance security. Using homomorphic encryption, operations can be performed by the intermediate nodes on the ciphertext as if performed on the plaintext without actually knowing the plaintext [1] [2] [3]. This enhances security of the protocol as the intermediate nodes if malicious cannot determine the plaintext. Homomorphism allows operation to be performed on the encrypted data (ciphertext) as if the operation is performed on the plaintext. Homomorphism has the property of additive, multiplicative and mixed multiplicative [1]. In *additive homomorphism*, decrypting the sum of two ciphertext is same as addition of two plaintext, represented as $E(x+y) = E(x) + E(y)$. In *multiplicative homomorphism*, decrypting the product of two ciphertext is same as multiplication of the two plaintexts. Multiplicative homomorphism is mathematically represented as $E(x*y) = E(x) * E(y)$. In *mixed multiplicative homomorphism*, decrypting the product of one ciphertext and plaintext is same as multiplication of two plaintext, represented as $E(x*y) = E(x) * y$.

In this paper we briefly describe the encryption schemes having the property of homomorphism. We then aim to find the minimum optimal path by using ElGamal, Elliptic Curve and Privacy Homomorphism encryption schemes.

The paper is organized as follows. In section 2, we briefly describe the overview of homomorphic encryption schemes. In section 3, we briefly describe the protocol, which determines maximum optimal path dynamically. In section 4, we propose new routing protocols, which use the homomorphic property of ElGamal encryption, Elliptic Curve Encryption and Privacy Homomorphism. Finally conclusions are given.

2. Encryption Schemes Exhibiting the Property of Homomorphism

In this section, we give an overview of cyptosystem using *mod* operation [1], Privacy Homomorphism [2], ElGamal encryption [3],[4] and Elliptic Curve [8],[9],[10],[11],[12] which exhibits the property of homomorphism.

2.1 Encryption Functions using Mod Operations

In this section, we focus on encryption schemes using *mod* operations, which are *cryptosystem using mod operation* and *privacy homomorphism* exhibiting the property of homomorphism.

The *cryptosystem using mod operation* is introduced in [1]. This cryptosystem uses large number n, where $n= p* q$. Here p and q are large prime numbers, which are kept secret. The set of original plaintext messages is in $Z_p =\{ x|x <= p \}$, $Z_n = \{ x|x <n \}$ has the set of ciphertext messages and $Q_p = \{ a|a \notin Z_p \}$ has the set of encryption clues.

The *encryption algorithm* is performed by choosing a plaintext 'x' belonging to Z_p and a random number 'a' in Q_p

such that $x = a \bmod p$. Here p is kept secret. The ciphertext y is calculated as $y = E_p(x) = a \bmod n$.

In *decryption algorithm* the plaintext x is recovered as $x = D_p(y) = y \bmod p$, where p is the secret key.

This cryptosystem has the property of additive, multiplicative and mixed multiplicative homomorphism. The proposed protocol, though exhibits the property of homomorphism is not very secure against known plaintext attacks, but secure against known ciphertext attacks [1].

We now look into a privacy homomorphism protocol, which is relatively secure against known plaintext attacks and completely secure against known ciphertext attacks.

The *privacy homomorphism* is introduced in [2] which is a homomorphic encryption scheme not vulnerable to known ciphertext attacks.

Let us look into the protocol in detail. In this protocol n and m are the public parameters. Here $m = p * q$, where p and q are large prime numbers. To increase security, m can be kept secret. The number 'n' represents the split of the plaintext. The secret keys are p, q, x_p, x_q. Here, $x_p \in Z_p$ and $x_q \in Z_q$

Encryption operation is performed by selecting the plaintext $a \in Z_m$. We then split a into secret numbers $a_1, a_2 \dots a_n$, such that $a = (a_1 + a_2 \dots + a_i + \dots a_n) \bmod m$ and $a_i \in Z_m$.

$E_k(a) = (a_1 x_p \bmod p,\ a_1 x_q \bmod q), (a_2 x^2_p \bmod p,\ a_2 x^2_q \bmod q) \dots (a_n x^n_p \bmod p,\ a_n x^n_q \bmod q)$

Decryption operation is performed by computing scalar product of the i^{th} pair $[\bmod p, \bmod q]$ by $[x^{-i}_p \bmod p,\ x^{-i}_q \bmod q]$ to get $[a_i \bmod p,\ a_i \bmod q]$. The pairs are then added up to get $[a \bmod p,\ a \bmod q]$. Finally, Chinese remainder theorem (CRT) [5] is performed to get $a \bmod m$.

Let us illustrate an *example* to explain the protocol in more detail. Consider the example of 2 multiplication and 1 addition such that $(x1 * x2) + (x3 * x4)$.

Let $n = 2$, which implies that the plaintext is split into 2. Consider $p = 11,\ q = 7,\ x_p = 2,\ x_q = 3$ as secret keys. Let $(x1, x2, x3, x4) = (-1, 1, 2, 3)$

Encryption operation is performed as shown below. First the plaintext is split then encryption is done,

$Ek(x1) = Ek(-1) = Ek(2, -3)$
$= [2 x_p \bmod p,\ 2 x_q \bmod q],$
$[-3 x^2_p \bmod p,\ -3 x^2_q \bmod q]$
$= [4, 6], [10,1]$

$Ek(x2) = Ek(1) = Ek(4, -3)$
$= [4 x_p \bmod p,\ 4 x_q \bmod q],$
$[-3 x^2_p \bmod p,\ -3 x^2_q \bmod q]$
$= [8, 5], [10,1]$

$Ek(x3) = Ek(2) = Ek(3, -1)$
$= [3 x_p \bmod p,\ 3 x_q \bmod q],$
$[-1 x^2_p \bmod p,\ -1 x^2_q \bmod q]$
$= [6, 2], [7,5]$

$Ek(x4) = Ek(3) = Ek(4, -1)$
$= [4 x_p \bmod p,\ 4 x_q \bmod q],$
$[-1 x^2_p \bmod p,\ -1 x^2_q \bmod q]$
$= [8, 5], [7,5]$

$Ek(x1) * Ek(x2) = ([4, 6]\ [10, 1]) * ([8, 5]\ [10, 1])$
$= [0, 0]\ [4*8, 6*5]\ [4*10, 6*1]$
$[10*8, 1*5]\ [10*10, 1*1]$
$= [0, 0]\ [32, 30]\ [40, 6]\ [80, 5]$
$[100,1]$

$Ek(x3) * Ek(x4) = ([6, 2]\ [7, 5]) * ([8, 5]\ [7, 5])$
$= [0, 0]\ [6*8, 2*5]\ [6*7, 2*5]$
$[7*8, 5*5]\ [7*7, 5*5]$
$= [0,0]\ [48,10][42,10][56,25][49,25]$

Performing $(Ek(x1) * Ek(x2)) + (Ek(x3) * Ek(x4))$ we get, $[0,0]\ [80, 40]\ [218, 46]\ [149, 26]$

Decryption operation is performed as,

$[0 * x^{-1}_p \bmod p,\ 0 * x^{-1}_q \bmod q], [80 * x^{-2}_p \bmod p,\ 40 * x^{-2}_q \bmod q], [218 * x^{-3}_p \bmod p,\ 46 * x^{-3}_q \bmod q], [149 * x^{-4}_p \bmod p,\ 26 * x^{-4}_q \bmod q]$

$[0 \times 6 \bmod 11,\ 0 \times 5 \bmod 7], [80 * 6^2 \bmod 11,\ 40 * 5^2 \bmod 7], [218 * 6^3 \bmod 11,\ 46 * 5^3 \bmod 7], [149 * 6^4 \bmod 11,\ 26 * 5^4 \bmod 7]$

$[0, 0], [9, 6], [8, 3], [10, 3]$

Add up all the terms over $Zp \times Zq$ to get,
$[0 + 9 + 8 + 10 \bmod 11,\ 0 + 6 + 3 + 3 \bmod 7]$
$[5, 5]$

By applying Chinese remainder theorem (CRT) on $[5, 5]$ we get 5 (decrypted ciphertext).

We know that $(x1 * x2) + (x3 * x4) = 5$ which is the plaintext. This shows that the plaintext and the decrypted ciphertext are the same. The protocol is said to be secure against known plaintext attacks and ciphertext attacks [2], but there is a possibility to break this protocol with great difficulty by using known plaintext ciphertext pairs as explained in [6]. When this protocol is used, one has to consider this weakness.

2.2 ElGamal Encryption and Elliptic Curve Encryption

In this section we briefly look into public key encryption schemes like ElGamal and Elliptic Curve, which have the property of homomorphism.

ElGamal encryption [3] [4] is a public key encryption, which requires large prime numbers, p and q where $p = 2q+1$. The public keys are p, g and y, where g is the cyclic group which is the subset of Z_p, $y = g^x$, x being the secret key belonging to Z_q. Here Z_p is set of integers from 0 to $p-1$ and Z_q is the set of integers from 0 to $q-1$.

During *encryption* the message M is encrypted using public keys. $E(M) = (A = g^r,\ B = y^r M)$ where r is the random number.

Decryption is computed with the private key x, $D(E(M)) = B/A^x$.

ElGamal encryption has the property of multiplicative homomorphism [3].

Elliptic curve encryption [8], [9], [10], [11], [12] is also a public key encryption scheme, which requires a point G and an elliptic group $E(a,b)$ to perform encryption and decryption. The node S, selects a private key n_S and generates public key $P_S = n_S \times G$. The *encryption operation* is performed by choosing a random number k and the ciphertext is generated as, $C = \{ kG, m+kP_D \}$ where P_D is the public key of node D.

Decryption is performed by node D using the private key n_D. Node D computes the multiplication of the 1st point of the ciphertext with it's private key and then subtracts the result from the 2nd point of the ciphertext to recover the plain text.

$m+kP_D - (n_D kG) = m + k(n_D G) - (n_D kG) = m$

Elliptic Curve has the property of additive homomorphism [15], [16].

3. A Dynamic Programming based on Homomorphic Encryption

The dynamic programming based on homomorphic encryption is introduced in [3] which find a maximum optimal path using ElGamal Homomorphic encryption.

The protocol chooses a weight w such that $(1 <= w <= n)$ and chooses n such that it is large enough to represent the longest path.

$$e(w) = (e_1, e_2, \ldots e_n)$$
$$= \underbrace{E(z),\ldots,E(z)}_{w},\ \underbrace{E(1),\ldots,E(1)}_{n-w}$$

Here $e(w)$ is the encryption of weight w, $E(1)$ is the encryption of 1, $E(z)$ is the encryption of z and z is a public number not equal to 1. z is chosen such that $z^k \bmod p \neq 1$ for $0 < k < q$.

Let us consider an example where $w=2$ and $n=4$

$e(w) = E(z), E(z), E(1), E(1)$. Here z can be any number not equal to 1.

A constant f is added to encrypted function $e(w)$ by shifting $e(w)$ to the right f times.

$$e(w+f) = \underbrace{E(z),\ldots,E(z)}_{f},\ e_1, \ldots e_{n-f}$$

let $e(w) = E(z), E(z), E(z), E(1),E(1),E(1)$ and $f=2$, then, $e(w+f) = E(z), E(z), E(z), E(z), E(z), E(1)$

This operation can be performed without decrypting $e(w)$. If we compare $e(w)$ and $e(w+f)$ we cannot know the amount of shift. The shifting and the encryption can be masked by multiplying with $E(1)$ and using different random numbers for encryption. The value of z can be different for different encryption.

The maximum of the two weights can be found without decrypting the entire encrypted weights. This can be achieved by multiplying the two encrypted weights and decrypting the resulting product from e_n to e_1, for $i = 1$ *to* n until $D(E(xi)) \neq 1$ and i determines highest of the two weights.

Consider two weights $e(w+f)$ and $e(v)$

$e(w+f) = E(z), E(z), E(z), E(z),E(z), E(1)$

$e(v) = E(z), E(z), E(1), E(1),E(1), E(1)$

$e(w+f) * e(v) = E(z^2), E(z^2), E(z), E(z),E(z), E(1)$

Decrypting the *6th* element we get $D(E(x) = 1$, decrypting the *5th* element we get $D(E(x)) = z \neq 1$, therefore the maximum of the two weight is 5. Using this scheme a maximum optimal path is determined dynamically in [3].

In the next section we propose new protocols, which determine the shortest path using ElGamal, Elliptic Curve and Privacy Homomorphism.

4. Finding Minimum Optimal Path Using Homomorphic Encryption Schemes

In this section we propose three new routing protocols in computer networks, which finds minimum optimal path using ElGamal Encryption, Elliptic Curve Encryption and Privacy Homomorphism. As these encryption schemes are additive and multiplicative we reduce the computational power and increase security by doing operations on encrypted data. Furthermore the proposed protocols encrypt the weight in a particular fashion, which make it difficult for an intruder to determine the weight.

4.1 Use of ElGamal Homomorphic Encryption and Its Application to Find the Shortest Optimal Path

We first find the minimum optimal path using ElGamal homomorphic encryption scheme. Encryption scheme in ElGamal [3] [4] requires two exponentiation, which could be computed ahead of time, as it is independent of the message. Decryption requires one exponentiation and one division, which is computationally much faster. To find the minimum optimal path, we assume that all the nodes know the weight to its neighboring nodes. We choose a number n, which is large enough to represent the length of the longest path. The weight w is encrypted with the public key and $w<=n$

$$e(w) = (e_1, e_2, \ldots e_n)$$
$$= \underbrace{E(1),\ldots,E(1)}_{w},\ \underbrace{E(z),\ldots,E(z)}_{n-w}$$

Here $e(w)$ is the encryption of weight w, $E(1)$ is the encryption of 1, $E(z)$ is the encryption of z and z is a public number not equal to 1. z is chosen such that $z^k \bmod p \neq 1$ for $0 < k < q$.

Consider an example where $w=2, n=4$ and $z \neq 1$.

$e(w) = E(1), E(1), E(z), E(z)$

In this protocol, the two paths are combined by shifting the encrypted path $e(w)$ to the right by the weight of the other path.

Let $e(w)$ be the encrypted weight of one path and f be the weight of other path, then

$$e(w+f) = \underbrace{E(1),...,E(1)}_{f},\ e1, ..., en\text{-}f$$

Consider $f=2$

let $e(w) = E(1), E(1), E(1), E(z),E(z), E(z)$

$e(w+f) = E(1), E(1), E(1), E(1),E(1), E(z)$

This operation can be performed without decrypting $e(w)$. If we compare $e(w)$ and $e(w+f)$ we cannot know the amount of shift. The shifting and the encryption can be masked by multiplying with $E(1)$. Further different random numbers can be used for different encryption.

The minimum of the two weights can be found without decrypting the entire encrypted weight. This can be achieved by multiplying the two encrypted weights and decrypting the resultant product from right to left until we get a value $z=1$. The position at which we get a value $z=1$, determines the minimum weight.

The optimal of the two paths are determined by decrypting the encrypted paths at *minimum weight+1* position. The path, which decrypts to a value $z \neq 1$ at the *minimum weight+1* position, is the minimum optimal path.

Consider two weights $e(w+f)$ and $e(v)$

$e(w+f) = E(1), E(1), E(1), E(1),E(1), E(z)$

$e(v) = E(1), E(1), E(z), E(z),E(z), E(z)$

$e(w+f) * e(v) = E(1), E(1), E(z), E(z),E(z), E(z^2)$

Decrypting the 6th, 5th, 4th and 3rd element of the resultant product, we get $D(E(x)) \neq 1$, decrypting the 2nd element we get $D(E(x)) = 1$, therefore the minimum of the two weight is *2*.

To find the optimal path between $e(w+f)$ and $e(v)$ decrypt both these paths at the 2nd +1 position. The path which decrypts to $z \neq 1$ at the 2nd + 1 position is the optimal path with the minimum weight.

Let us look into Figure 1, to illustrate the protocol in detail. $w(0,1) = 1$, $w(0, 2) = 2$, $w(1, 2) = 1$, $w(1, 3) = 2$, $w(2, 3) = 0$ are the weights. Let $n = 6$ and z can be any number not equal to *1*.

$e(0,1)\ (w(0,1)) = E_0(1),E_0(z),E_0(z),E_0(z),E_0(z),E_0(z)$: encrypted with the public key of node *0*.

$e(0,2)\ (w(0,2)) = E_0(1),E_0(1),E_0(z),E_0(z),E_0(z),E_0(z)$: encrypted with the public key of node *0*.

$e(1,2)\ (w(1,2)) = E_1(1),E_1(z),E_1(z),E_1(z),E_1(z),E_1(z)$: encrypted with the public key of node *1*.

$e(1,3)\ (w(1,3)) = E_1(1),E_1(1),E_1(z),E_1(z),E_1(z),E_1(z)$: encrypted with the public key of node *1*.

$e(2,3)\ (w(2,3)) = E_2(z),E_2(z),E_2(z),E_2(z),E_2(z),E_2(z)$: encrypted with the public key of node *2*.

By decrypting path *2-3* at node *2*, we find the distance to be *0*. At node *2* we compute path *1-2-3* by shifting the encrypted path of *1-2* to the right by the weight of path *2-3*, which is *0*.

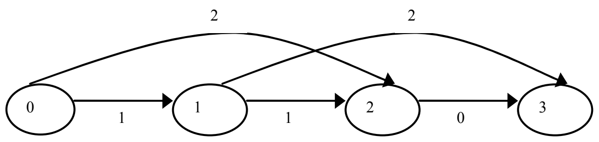
$e(1,2,3)\ (w(1,2,3)) = E_1(1),E_1(z),E_1(z),E_1(z),E_1(z),E_1\ (z)$

To find the minimum of two paths *1-2-3* and *1-3* with weights $w(1,2,3)$ & $w(1,3)$ respectively, at node *1* we multiply the two weights.

$e(1,2,3)(w(1,2,3)) * e(1,3)\ (w(1,3))$

$= E_1(1),E_1(z),E_1(z^2),E_1(z^2),E_1(z^2),E_1(z^2)$

By decrypting the 6th, 5th, 4th, 3rd and 2nd element we get $D(E(x)) \neq 1$ and decrypting the 1st element we get $D(E(x)) = 1$. So the minimum weight is *1*.



The circled numbers represent nodes
The rest of the numbers represent published weights.

Figure 1: Example of a directed graph

To find the optimal path between *1-2-3* and *1-3*, decrypt *1-2-3* and *1-3* at the 1st+1 (2nd) position and the path, which decrypts to the value not equal to *1* at the 2nd position, is the shortest optimal path. Here path *1-2-3* is the optimal path, as it decrypts to $z \neq 1$ at the 2nd position.

To combine two paths *0-1* and *1-2-3* to get *0-1-2-3*, we shift the encrypted path of *0-1* to the right by *1* at node *1*, which is the weight of the path *1-2-3*.

$e(0,1,2,3)(w(0,1,2,3))=E_0(1),E_0(1),E_0(z),E_0(z),E_0(z), E_0(z)$

At node *0* we also obtain an encrypted path *0-2-3* from node 2, which is obtained by shifting the encrypted path *0,2* to the right by *0* (weight of path *2-3*).

$e(0,2,3)(w(0,2,3)) = E_0(1),E_0(1),E_0(z),E_0(z),E_0(z), E_0(z)$

Node *0* has two encrypted paths $e(0,2,3)$ and $e(0,1,2,3)$, and we determine the minimum path by multiplying the two encrypted paths to get,

$e(0,1,2,3)(w(0,1,2,3)) * e(0,2,3)(w(0,2,3))$

$= E_0(1), E_0(1), E_0(z^2), E_0(z^2), E_0(z^2), E_0(z^2)$

By decrypting the resultant path at node *0* we find the 2nd position decrypts to value *1* and so the minimum weight is *2*. By decrypting encrypted paths $e(0,2,3)$ and $e(0,1,2,3)$ at the 2nd +1 position we find that both the paths decrypts to a value z ≠ 1. The optimal path from *0* to *3* has the minimum weight *2* and the optimal path is either *0-1-2-3* or *0-2-3* determined at node *0* which is the source.

In the next section we find the minimum optimal path using Elliptic Curve encryption.

4.2 Use of Elliptic Curve Homomorphic Encryption and Its Application to Find the Shortest Optimal Path

Elliptic Curve Public Key Cryptosystem [8], [9], [10], [11], [12] is a relatively new public key cryptography which uses relatively small key sizes compared to ElGamal and RSA. Reduction in key sizes brings the advantage of less storage area and less required bandwidth, which are important requirements of wireless network architectures. In addition, Elliptic Curve permits the implementation of high-speed and efficient network security protocols requiring less power and smaller code sizes as compared to classical public key techniques such as ElGamal, RSA and Diffie-Hellman [7], [12], [13], [14].

We now look at how Elliptic Curve encryption scheme can be used to find the minimum optimal path. We use a similar scheme as in ElGamal encryption, but the only difference is that the value of z can be any number other than 0 and the minimum path is obtained by adding the two paths as Elliptic Curve Cryptosystem has only additive homomorphism. That's why z is chosen such that $z \neq 0$. The weight w is encrypted as follows

$$e(w) = (e_1, e_2, \dots e_n)$$
$$= \underbrace{E(0),\dots,E(0)}_{w},\ \underbrace{E(z),\dots,E(z)}_{n-w}$$

Here $E(0)$ is the encryption of weight 0 and $E(z)$ is the encryption of weight z. Here z can be any number not equal to 0, n is large enough to represent the length of the longest path and weight $w<=n$. The encrypted weights are randomized by adding with $E(0)$.

As in ElGamal, shifting the encrypted weight of one path to the right by the weight of the other path combines the two paths. The paths are compared for optimality by adding the two paths. The *minimum weight* of the two paths is the position at which the resultant sum decrypts to a value $z=0$, when the decryption is carried out from right to left. The *minimum optimal path* is the one, which decrypts to a value $z \neq 0$ at the *minimum weight +1* position.

Let us consider the Figure 1, to explain this in detail

$w(0,1) = 1, w(0, 2) = 2, w(1, 2) = 1, w(1, 3) = 2, w(2, 3) = 0$, are the weights. Let $n = 6$ and z can be any number not equal to 0.

$e(0,1)\ (w(0,1)) = E_0(0),E_0(z),E_0(z),E_0(z),E_0(z),E_0(z)$: encrypted with the public key of node 0.

$e(0,2)\ (w(0,2)) = E_0(0),E_0(0),E_0(z),E_0(z),E_0(z),E_0(z)$: encrypted with the public key of node 0.

$e(1,2)\ (w(1,2)) = E_1(0),E_1(z),E_1(z),E_1(z),E_1(z),E_1(z)$: encrypted with the public key of node 1.

$e(1,3)\ (w(1,3)) = E_1(0),E_1(0),E_1(z),E_1(z),E_1(z),E_1(z)$: encrypted with the public key of node 1.

$e(2,3)\ (w(2,3)) = E_2(z),E_2(z),E_2(z),E_2(z),E_2(z),E_2(z)$: encrypted with the public key of node 2.

By decrypting path *2-3* at node 2, we find the distance to be 0. At node 2 we compute path *1-2-3* by shifting the encrypted path of *1-2* to the right by the weight of path *2-3*, which is 0.

$e(1,2,3)\ (w(1,2,3) = E_1(1),E_1(z),E_1(z),E_1(z),E_1(z),E_1(z)$

To find the minimum of two paths *1-2-3* and *1-3* with weights $w(1,2,3)$ & $w(1,3)$ respectively, at node 1 we add the two weights to get,

$e(0,1,2)(w(0,1,2)) + e(0,2)\ (w(0,2))$
$= E_0(0), E_0(z), E_0(2z), E_0(2z), E_0(2z), E_0(2z)$

By decrypting the 6th, 5th, 4th, 3rd and 2nd element, we get $D(E(x)) \neq 0$ and decrypting the 1st element we get $D(E(x)) = 0$. So the minimum weight is 1.

To find the optimal path between *1-2-3* and *1-3*, we decrypt *1-2-3* and *1-3* at the 1st + 1 (2nd) position and the path, which decrypts to the value not equal to 0 at the 2nd position, is the shortest optimal path. Here path *1-2-3* is the optimal path, as it decrypts to $z \neq 0$ at the 2nd position.

To combine two paths *1-2-3* and *0-1* to get *0-1-2-3*, at node 1 we shift the encrypted path of *0-1* to the right by 1, which is the weight of the path *1-2-3*.

$e(0,1,2,3)(w(0,1,2,3))=E_0(0),E_0(0),E_0(z),E_0(z),E_0(z), E_0(z)$

At node 0 we also obtain an encrypted path *0-2-3* from node 2, which is obtained by shifting the encrypted path *0,2* to the right by 0 (weight of path *2-3*).

$e(0,2,3)(w(0,2,3)) = E_0(0),E_0(0),E_0(z),E_0(z),E_0(z), E_0(z)$

Node 0 has two encrypted paths $e(0,2,3)$ and $e(0,1,2,3)$, and we determine the minimum path by adding the two encrypted weights to get,

$e(0,1,2,3)(w(0,1,2,3)) + e(0,2,3)(w(0,2,3))$
$= E_0(0), E_0(0), E_0(2z),E_0(2z), E_0(2z), E_0(2z)$

By decrypting the resultant path we find the 2nd position decrypts to value 0 and so the minimum weight is 2. The optimal path from 0 to 3 has the minimum weight 2 and the optimal path is either *0-1-2-3* or *0-2-3* as both these paths decrypt to a value $z \neq 0$ at the *minimum weight +1* position.

In the next section we find the minimum optimal path using a privacy homomorphism.

4.3 Use of Privacy Homomorphism and Its Applications to Find the Shortest Optimal Path

In this section we look into our routing protocol using privacy homomorphism encryption scheme. In this protocol, a weight w is chosen such that $w <= n$ and n is chosen such that it is large enough to represent the length of the longest path as in ElGamal and Elliptic Curve encryption. The encryption of the weight in this protocol is performed as follows:

$$e(w) = (e_1, e_2, \dots e_n)$$
$$= \underbrace{E(0),\dots,E(0)}_{S^*w},\ \underbrace{E(z),\dots,E(z)}_{S^*(n-w)}$$

Here $e(w)$ is the encryption of weight w, $E(0)$ is the encryption of 0, $E(z)$ is the encryption of z, n is large enough to represent the longest path, S represents the number of split of the plaintext and z is any number not equal to 0. In this encryption randomness is achieved by splitting each plaintext differently, and adding $E(0)$ which are split differently.

Here the paths are compared for optimality by adding the two encrypted paths instead of multiplying the two paths as multiplication increases the vectors in squares.

Consider an example where $w=2, z=5, n=4, S=2$

$e(w) = E(0)\ E(0), E(0)\ E(0), E(z)\ E(z), E(z)\ E(z)$

In this protocol, the two paths are combined by shifting the encrypted path $e(w)$ to the right by S^*f, f being the weight of the other path.

$$e(w+f) = \underbrace{E(0),\dots,E(0)}_{S^*f},\ e_1\ , \dots, e_{n-f}$$

Consider $f=2$, $S=2$ and $n=4$

let *e(w) = E(0) E(0), E(0) E(0), E(z) E(z), E(z) E(z)*
e(w+f) = E(0) E(0), E(0) E(0), E(0) E(0), E(0) E(0)

This operation can be performed without the decryption of *e(w).* If we compare *e(w)* and *e(w+f)* we cannot know the amount of shift. The shifting and the encryption can be masked by adding *E(0).*

The minimum of the two weights can be found without decrypting the entire encrypted weights. This can be achieved by adding the two encrypted weights and decrypting the resultant sum from right to left in pairs of *S* until we get a value *z=0*. The position at which we get a value *z=0*, determines the minimum weight.

The minimum optimal path can be determined by decrypting the two paths at *minimum weight + 1* position and the path, which decrypts to a value $z \neq 0$ is the minimum optimal path. Decryption is performed in groups of *S* as *S* is the split of the plaintext.

Consider two weights *e(w)* and *e(v)*

e(w) = E(0) E(0), E(0) E(0), E(z) E(z)
e(v) = E(0) E(0), E(z) E(z),E(z) E(z)
e(w) + e(v) = E(0) E(0), E(z) E(z), E(z) E(z)

Decrypting the 6th and 5th element together of the resultant sum we get *D(E(x)) = z,* decrypting the 2nd and 1st element together we get *D(E(x)) = 0*, therefore the minimum of the two weight is *1*.

To find the optimal path between *e(w)* and *e(v)* decrypt both these paths at the *3rd and 4th* position. The path, which decrypts to $z \neq 0$ at the 3rd and 4th position, is the optimal path with the minimum weight.

Let us consider Figure 1 to illustrate this scheme in detail. Let *w(0,1) = 1, w(0, 2) = 2, w(1, 2) = 1, w(1, 3) = 2, w(2, 3) = 0* , be the weights. Let *n = 6* and *S=2,* then

e(0,1) (w(0,1)) = E(0) E(0), E(z) E(z), E(z) E(z), E(z) E(z), E(z) E(z), E(z) E(z)
e(0,2) (w(0,2)) = E(0) E(0), E(0) E(0), E(z) E(z), E(z) E(z), E(z) E(z), E(z) E(z)
e(1,2) (w(1,2)) = E(0) E(0), E(z) E(z), E(z) E(z), E(z) E(z), E(z) E(z), E(z) E(z)
e(1,3) (w(1,3)) = E(0) E(0), E(0) E(0), E(z) E(z), E(z) E(z), E(z) E(z), E(z) E(z)
e(2,3) (w(2,3)) = E(z) E(z), E(z) E(z), E(z) E(z), E(z) E(z), E(z) E(z), E(z) E(z)

By decrypting path *2-3* at node *2,* we find the weight of path *2-3* to be *0*. The path *1-2-3* is obtained by shifting the encrypted path *1-2* to the right by the weight of *2-3* in pairs of *S* at node *2.*

e(1,2,3) = E(0) E(0), E(z) E(z), E(z) E(z), E(z,) E(z), E(z), E(z), E(z) E(z)

To find the minimum of two paths *1-2-3* and *1-3,* at node *1* we add the two encrypted paths to get the resultant path,

e(1-2-3) + e(1-3) = E(0) E(0), E(z) E(z), E(z) E(z), E(z) E(z), E(z) E(z), E(z) E(z).

By decrypting the *12th* and *11th* element together,..., *4th* and *3rd* element together, we get $D(E(x)) \neq 0$ and by decrypting the 2nd and 1st element together we get *D(E(x)) = 0.* So the minimum weight is *1.*

To find the optimal path between *1-3* and *1-2-3,* we decrypt *1-3* and *1-2-3* at the 3rd and the 4th position together and the path, which decrypts to a value $z \neq 0$ is the optimal path. We find that path *1-2-3* decrypts to a value $z \neq 0$. Hence, path *1-2-3* is the optimal path.

We know that the path *1-2-3* has the weight *1.* To combine two paths *0-1* and *1-2-3* to get *0-1-2-3,* we shift the encrypted path *0-1* to the right by *1* in pairs of *S.*

e(0-1-2-3) = E(0) E(0), E(0) E(0), E(z) E(z), E(z) E(z), E(z) E(z), E(z) E(z).

At node *0* we also obtain an encrypted path *0-2-3* from node *2*.

e(0-2-3) = E(0) E(0), E(0) E(0), E(z) E(z), E(z)E(z), E(z) E(z), E(z) E(z).

Node *0* has two encrypted paths *e(0,2,3)* and *e(0,1,2,3)*, and we determine the minimum path by adding the two encrypted weights to get,

e(0,1,2,3)(w(0,1,2,3)) + e(0,2,3)(w(0,2,3))
= E(0) E(0), E(0) E(0), E(z) E(z), E(z) E(z), E(z) E(z),E(z) E(z).

By decrypting we find that the minimum weight is *2*. The optimal path from *0* to *3* has the minimum weight *2* and the optimal path is either *0-1-2-3* or *0-2-3,* determined at node *0,* as both these paths decrypt to a value $z \neq 0$ at the *minimum weight +1* position.

In this section, we have proposed three new routing protocols, which use ElGamal Encryption, Elliptic Curve Encryption and Privacy Homomorphism. These schemes ensure that the minimum optimal path is finally determined by the source. None of the nodes are aware of what the optimal path is going to be. We consider that Elliptic Curve Encryption scheme is better over ElGamal as it requires less key space and is computationally faster than ElGamal Encryption [12]. However, we consider that the Privacy Homomorphism is computationally much faster than both ElGamal and Elliptic Curve Encryption. But the problem with Privacy homomorphism is that the encrypted weight is *'S'* times longer than that required for ElGamal and Elliptic Curve Encryption. This results in more storage space.

5. Conclusions

In this paper we have shown that a minimum optimal path can be found securely to route the packets in computer networks by using homomorphic encryption schemes. Using these homomorphic encryption schemes, minimum weight of the two paths can be found without actually decrypting the entire resultant path. The minimum of the two paths can be found by decrypting the two paths at the minimum weight +1 position, thus reducing the computational power by not decrypting the entire path. These proposed protocols provide confidentiality as the minimum path is found securely. Using ElGamal and Elliptic Curve encryption schemes, confidentiality is achieved, as the intermediate nodes can neither determine the encrypted weight of the other nodes nor the minimum optimal path. The minimum optimal path is chosen by the source. An intruder can neither determine the encrypted weights nor the minimum optimal path without the knowledge of ElGamal or Elliptic Curve secret keys of every node. Using privacy homomorphism, confidentiality is also achieved, as an intruder cannot determine the weight or the optimal path without the knowledge of the secret key.

In the future, implementation problems of these newly proposed protocols will be addressed and performance comparison of these schemes will be given.

6. References

[1] Hyungjick Lee, Jim Alves-Foss,Scott Harrison, " The use of Encrypted Functions for Mobile Agent Security",Proceedings of the 37th Hawaii International Conference on System Sciences – 2004.

[2] Josep Domingo i Ferrer, "A new Privacy Homomorphism and Applications", Elsevier North-Holland, Inc, 1996.

[3] Makoto Yokoo, Koutarou Suzuki, "Secure Multi-agent Dynamic Programming based on Homomorphic Encryption and its Application to Combinatorial Auctions", Proceedings of the First International joint Conference on Autonomous Agents and Multiagent systems(AAMAS), 2002.

[4] T. Elgamal," A public key cryptosystem and a signature scheme based on discrete logarithm. IEEE Trans. On Information Theory, 1986.

[5] William Stallings "Cryptography and Network Security", Third Edition, Chinese Remainder Theorem (CRT), pp. 245-47.

[6] Jung Hee Cheon, Hyun Soon Nam,"A Cryptanalysis of the Original Domingo-Ferrer's Algebraic Privacy Homorphism", http://eprint.iacr.org/2003/221.pdf

[7] W. Diffie, M. Hellman, "New Directions in Cryptography", IEEE Trans., on IT, Nov., 1976, pp. 644-654.

[8] N. Koblitz, "ECC", Math. of Computation, v. 48, 1987, pp. 203-209.

[9] A.J. Menezes, D. B. Johnson, "EC-DSA: An Enhanced DSA", Invited Talks – 7th Usenix Sec., Symp., Jan., 1998, pp. 33-43.

[10] Certicom Corp., "Certicom ECC Tutorials".

[11] Certicom Corp., "Remarks on the Security of the ECC systems", ECC White Papers, July 2000.

[12] K. Lauter, "The Advantages of Elliptic Curve Cryptography for Wireless Security," *IEEE Wireless Communications,* vol. 11, no. 1, pp. 62-67, February 2004.

[13] M. Aydos, T. Tanık, Ç. K. Koç, "High-Speed Implementation of an ECC-based Wireless Authentication Protocol on an ARM Microprocessor", IEE Pro.: Comms, Oct., 2001, pp 273-279.

[14] R. L. Rivest, A. Shamir, L.M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Comms of the ACM, v. 21-n.2, February 1978, pp. 120-126.

[15] L. Ertaul, W. Lu, "ECC Based Threshold Cryptography for Secure Data Forwarding and Secure Key Exchange in MANET (I)", Proc. of the Networking 2005 International Conf., May 2005, University of Waterloo, Ontario, CA

[16] L. Ertaul, "Cryptography Lecture Notes", California State University, East Bay, http://www.mcs.csueastbay.edu/~lertaul/