Efficient Methods for Practical Fully-Homomorphic Symmetric-key Encryption, Randomization, and Verification

Aviad Kipnis akipnis@nds.com;
Eli Hibshoosh ehibshoo@nds.com

Abstract

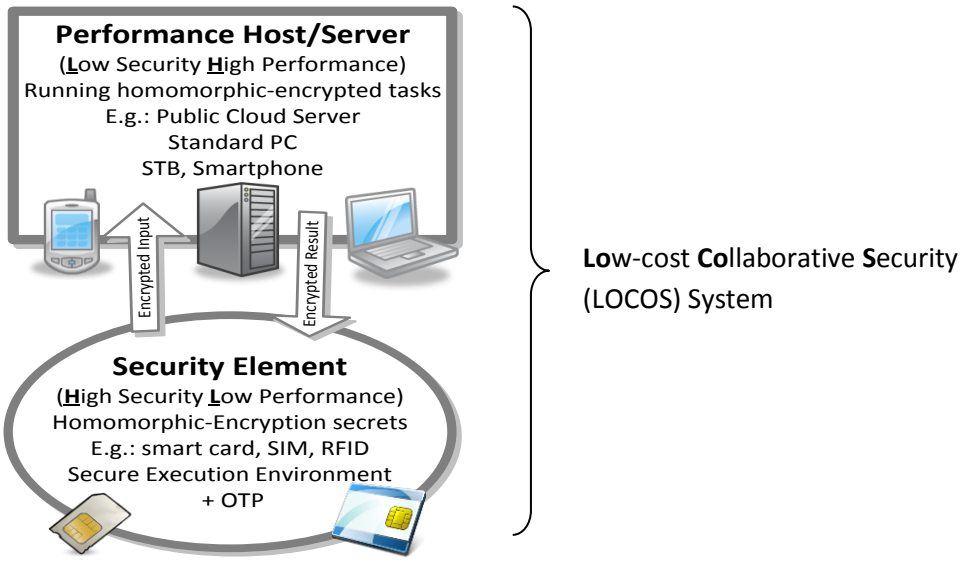
We present high performance non-deterministic fully-homomorphic methods for practical randomization of data (over commutative ring), and symmetric-key encryption of random mod-N data over ring Z_N well suited for crypto applications. These methods secure, for example, the multivariate input or the coefficients of a polynomial function running in an open untrusted environment. We show that random plaintext is the sufficient condition for proof of security for the homomorphic encryption. The efficient nature of the methods - one large-numbers multiplication per encryption and six for the product of two encrypted values - motivates and enables the use of low cost collaborative security platforms for crypto applications such as keyed-hash or private key derivation algorithms. Such a platform is comprised of a low-cost and low performance security element supported by an untrusted high performance server running the homomorpic algorithms. The methods employed may also provide enhanced protection for some existing crypto algorithms against certain attacks. Specifically, it is shown how to secure OSS public-key signature against Pollard attack. Further, we demonstrate how the homomorphic randomization of data can offer protection for an AES-key against side-channel attacks. Finally, the methods provide both fault detection and verification of computed-data integrity.

1. Introduction

We present non-deterministic, highly-efficient methods that provide a **F**ully-homomorphic **S**ymmetric-key **E**ncryption and **R**andomization **F**unction (FSERF). The methods are invariant under encryption or randomization. For each of the two (isomorphic) methods we define two domains; one domain for secure encryption and the other for randomization. For the symmetric-key encryption the plain text is an element of Z_N, and all computations are done modulo N; N being a product of two (large) primes. For security reasons the plaintext and random values used in the (non-deterministic) encryption operation are large random numbers (e.g., 1k bits or higher). For randomization where there is no security constraint the input data and random values used are in some commutative ring (CR). Throughout the paper, we focus on the methods' internal operation with the underlying understanding that use of different domains, CR or Z_N , will randomize or encrypt the input data, respectively. Another useful inherent property of the methods are coupled calculations where one calculation can attest to the validity of another and thus verify the integrity of a computed function with encrypted inputs. Fault

detection is yet another use of the above.

We present several applications where the utility of FSERF is evident. In particular, FSERF drives down the cost of a **Lo**w-cost **Co**llaborative **S**ecurity (LOCOS) system. In various real world applications we face the following problem: there is a strong machine that processes the data but this machine is usually not secure, on the other hand the security module in the system is computationally weak and can't process large amount of data fast enough. Such platform is depicted below



To provide a quick and intuitive grasp we present the methods and their properties in their simplest manifestation — a single plaintext element and a single random per method application; (a more generalized description of the methods is given in the Appendix A). In subsequent sections we also address related work, sample applications, and performance.

2. Related Work

The problem of homomorphic encryption (privacy) was introduced by Rivest, etal [5]. The problem of fully homomorphic encryption (FHE) supporting both addition and multiplication with an unlimited number of operations had essentially remained open (only partial solutions had been known) until Gentry's work in 2009. Gentry in 2009 [3] showed the first fully-homomorphic encryption scheme based on ideal lattices. Gentry developed a two steps approach. In the first step he presented a "Somewhat Homomorphic encryption scheme" — an encryption scheme that satisfies the homomorphism requirement. However, algebraic operation on encrypted data accumulates errors; thus decryption of the encrypted data remains valid only if limited number of algebraic operations on the encrypted data were performed. In the second step Gentry developed a general "bootstrap" method that cleans the accumulated errors. A combination of these two steps enables a construction of Fully Homomorphic Encryption scheme (FHE). Few other Homomorphic Encryption Scheme were proposed following Gentry's success – in 2011 Brakersky and Vikuntanathan published an FHE scheme based on the Learning With Errors (LWE) problem, [2] . The computational complexity of the above schemes

motivated research aimed at Improving their efficiency, e.g., [4]. An approach to demonstrate efficient homomorphic encryption with practical application to real-world problems is described in [1]. The approach is based on 'Somewhat'-homomorphic encryption supporting a **limited** number of homomorphic operations. The practical results in [1] are based on the use of the somewhat-homomorphic scheme proposed in [2]. This approach first set out to satisfy the requirement for efficiency (practicality), and then found the solution in the 'Somewhat'-homomorphic.

In this paper we present fully homomorphic encryption (FHE) methods which are practical albeit mainly applicable to cryptographic and security algorithms. The schemes are based on non-deterministic linear transformations. In general, linear schemes are inherently ill-suited, for encryption. However, the linear transformations presented in this paper enable homomorphic computation which is efficient. To provide the necessary security it is sufficient to constrain the plaintext such that it contains only random large-numbers in Z_N. This renders the scheme suitable for the domain of some security applications.

3. The Methods: Practical FSERFs Simplified (single input)

Each of the isomorphic methods presented is an efficient non-deterministic Fully-homomorphic Symmetric Encryption and Randomization Function (FSERF). The input or message, IN, is comprised of k elements in a commutative ring (CR) or in ring Z_N. Randomization is defined over a general commutative ring (CR), e.g., F_{256}, R (real numbers), or sub ring of commutative matrices. Encryption is defined over the ring Z_N (all operations are mod N, where N is a product of two (large secret) primes). The non-deterministic method operates on an input and a random parameter (in Z_N or CR); the random parameter changes per each execution of the method. For encryption both the plaintext input and the random parameter associated with it are random large-numbers in Z_N.

Two fundamental isomorphic methods are defined below; each randomizes or encrypts a single input element X_i . The basic (isomorphic) methods are a matrix-based method, MORE (Matrix Operation for Randomization or Encryption), and a polynomial-based method, PORE (Polynomial Operation for Randomization or Encryption). Additionally, 'compound' methods can be constructed by successively applying the basic methods, (see Appendix).

To simplify the description in this section we shall assume that the input is to be encrypted.

3.1 The matrix method, MORE.

Symmetric-Key Generation

Alice randomly selects a secret 2x2 invertible matrix, S, in Z_N to be the symmetric key for encryption.

Encryption

For each plaintext input element X_i Alice selects a random large-number Y_i, in Z_N.

X_i and Y_i are placed in order on the diagonal of a 2x2 diagonal matrix. We denote MORE's output matrix as Ai and define the encryption of X_i as:

$$MORE(X_i) = A_i = S \begin{pmatrix} X_i & 0 \\ 0 & Y_i \end{pmatrix} S^{-1} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

The cipher text of X_i can be regarded as the four values given in matrix Ai. However an alternative economical representation is possible. For a given S, MORE's matrix space is defined by two large numbers, and the cipher text of a Xi is defined by any pair of large numbers in A_i except for a_{12}, a_{21}.

Decryption

One who know S can decrypt the cipher text matrix A and recover a plaintext X. He eliminates S and S^{-1} by simple matrix multiplication, $X = (S^{-1}AS)_{11}$.

Alternatively, let the vector (1, e) be an eigenvector of the matrix $A = S\begin{pmatrix} X & 0 \\ 0 & Y \end{pmatrix}S^{-1}$

satisfying: (1 , e)A=(X , e·X)

The decryption of A is defined as follows:

$X = a_{11} + e \cdot a_{21}$ (Mod N) where $e = (-S_{12}/S_{22})$ mod N, and S_{ij} , a_{ij} a are the ij elements of matrices S and A, respectively.

Computation of multivariate function of encrypted values

We define below the operations of addition, multiplication and division of encrypted values.

We let A1 and A2 be the encrypted values of X_1 and X_2, respectively.

The Ai's comprise the input of a function in which they are added, multiplied, or divided. We thus have for:

Addition

MORE (X_1) + MORE (X_2) = $A_1 + A_2$; and

Multiplication

MORE (X_1) · MORE (X_2) = $A_1 \cdot A_2$; and

Division (for Det MORE(X2) ≠ 0)

1/ MORE (X_2) = A_2^{-1} and MORE (X_1) / MORE (X_2)= $A_1 \cdot A_2^{-1}$;

It can be easily shown that under the above definitions MORE is fully homomorphic.

3.2 The polynomial method, PORE.

Again, the simplest yet useful case to consider is a single variable encryption with the minimum degree of the public polynomial.

Symmetric-Key Generation

Alice selects two (mod N) secret random large-numbers, v_1 and v_2, for the symmetric-key.

Alice computes the public polynomial $PP(v) = (v-v_1) \cdot (v-v_2)$ mod$N = v^2+b \cdot v + c$

Encryption

Encryption of plain text X_i, Enc (Xi), is **any** linear function in variable v of the form $a_i \cdot v + d_i$ satisfying $a_i \cdot v_1 + d_i = X_i$.

Let the pair (a_i, d_i) define Enc (Xi).

Alice selects a large-number mod N random R_i, for a_i and solves the linear equation $R_i \cdot v_1 + d_i = X_i$ for d_i; thus $d_i = X_i - R_i \cdot v_1$.

The **cipher text of Xi, consists of the pair (a_i , d_i).**

Alternatively, Alice can pick a random large-number mod N, R_i , for the given X_i and solve the simultaneous equations below for the unknowns a_i and d_i:

a. $a_i \cdot v_1 + d_i = X_i$, and

b. $a_i \cdot v_2 + d_i = R_i$

resulting in: $a_i = (X_i - R_i) / (v1 – v2)$, and $d_i = X_i - a_i \cdot v_1 = (R_i \cdot v_1 – X_i \cdot v_2)/(v_1 – v_2)$.

This alternative, (computationally heavier), is useful in certain applications of verification.

Decryption

Given that an encrypted variable, (or a computed function of encrypted variables) is represented by a pair (a,d), anyone who knows the secret roots can decrypt by simply computing $a \cdot v_1 + d$.

Computation of multivariate function

For Bob to compute a function with the encrypted Xi, the public coefficients, b and c, (defined above under key generation) are needed; note that b and c do not change for encryption of different variables; they are given once only (per some predefined period) by Alice to Bob. We also note that only one large-number multiplication is needed for encryption.

When computing multivariate functions with the encrypted variables we need to consider the addition, multiplication and division of two variables. Addition and multiplication of encrypted values are defined by the addition and multiplication, respectively, of the corresponding linear functions in $Z_N[v]/PP(v)$

Given $PORE(X_1)= (a_1 , d_1)$, $PORE(X_2)= (a_2 , d_2)$ and $PP(v) = v^2+bv + c$, addition, multiplication and division are performed as below.

Addition

$PORE(X_1) + PORE(X_2) = (a_1+a_2 , d_1+d_2)$,

Multiplication:

$PORE(X_1) \cdot PORE(X_2) = ((a_1 + d_1) \cdot (a_2 + d_2) - a_1 \cdot a_2 \cdot (1 + b) - d_1 \cdot d_2 , (d_1 \cdot d_2 - a_1 \cdot a_2 \cdot c))$

This particular form aims at minimizing the number of multiplications of large numbers, i.e. five.

Note, for squaring a variable, $(PORE(X_1))^2 = (a_1 \cdot (2d_1 - b \cdot a_1) , (d_1 + \sqrt{c} \cdot a_1) \cdot (d_1 - \sqrt{c} \cdot a_1))$

Division:

Let $D = d_2 \cdot (a_2 \cdot b - d_2) - (c \cdot a_2) \cdot a_2$

$PORE(X_1)/PORE(X_2) = ((a_2 \cdot d_1 - a_1 \cdot d_2) /D , (d_1 \cdot (a_2 \cdot b - d_2) - (c \cdot a_2) \cdot a_1)/D)$

It can be shown easily that, under the above definitions, the PORE scheme is fully homomorphic.

3.3 MORE and PORE are isomorphic

Given the above definitions of MORE and PORE where operations under MORE are over the commutative ring C1 = $\{SMS^{-1} \mid M$ a diagonal matrix comprised of X and Y$\}$, and operations under PORE are in the commutative ring C2 = $Z_N[v]$ mod PP(v), it can be shown that:

1. The mapping T: C1→C2 defined by $T(SMS^{-1}) = av +d$ where $av_1+d=X$ and $av_2 + d =Y$, is an isomorphism, and
2. For a given element in C1 finding its isomorphic image by someone who knows PP(v) is as difficult as factoring N.

3.4 Verification

The methods provide a mechanism to verify that a returned result of a calculation performed by a third party on encrypted multivariate input is valid. We designate the returned result as R_M^* where MORE is used. The result is a homomorphic calculation denoted f performed on $A_1,A_2,\dots,A_k$ wherein A_i is equal to one of $MORE(X_i)$; $f(A_1,A_2,\dots,A_k)$ is equal to $MORE (f(X_1,X_2,\dots,X_k))= f(MORE (X_1), MORE (X_2),\dots,MORE (X_k))$.

The verifier receives the result of $f(A_1,A_2,\dots,A_k)$ in the form:

$$\mathrm{MORE}\big(f(X_1, X_2, \dots, X_K)\big) = S \begin{pmatrix} f(X_1, X_2, \dots, X_K) & 0 \\ 0 & f(Y_1, Y_2, \dots, Y_K) \end{pmatrix} S^{-1} = R_M^*$$

It decrypts the result to yield $f(X_1, X_2, \dots, X_K)$ and $f(Y_1, Y_2, \dots, Y_K)$

The verifier has precomputed $f(Y_1, Y_2, \ldots, Y_K)$. This may be done well before computing the encrypted input. The verifier compares the decrypted value $f(Y_1, Y_2, \ldots, Y_K)$ with the precomputed one, and deems the result of the computation of $f(X_1, X_2, \ldots, X_k)$ verified if a match occurs.

4. Applications

In this section applications are proposed that exploit the inherent properties of the methods.

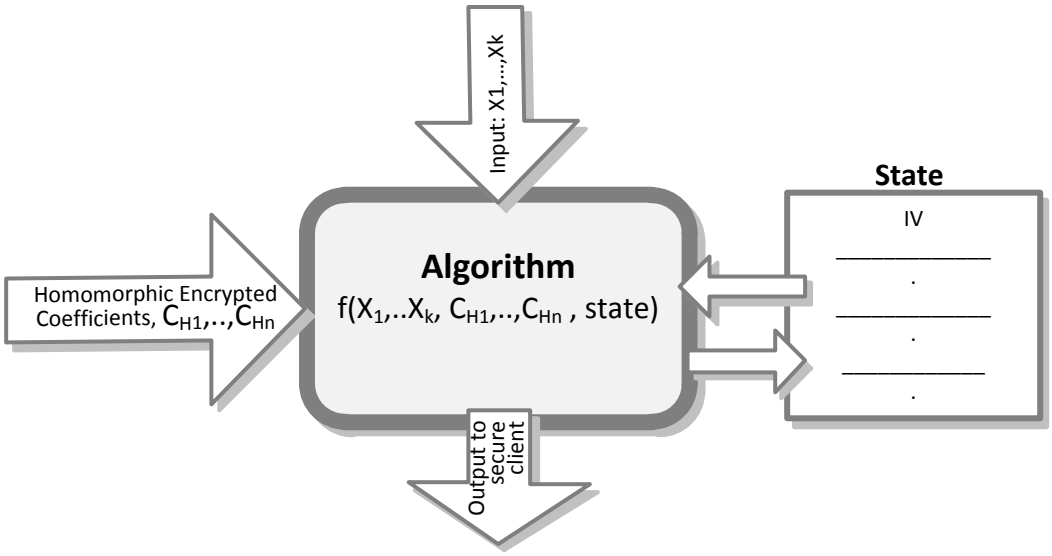
4.1 Securing for LOCOS and untrusted open cloud

For remote computing in an untrusted platform we consider three cases that define the secrets in a model of an **input** and **function** (algorithm):

a) **Only the input is secret** (not the algorithm).
The algorithm is characterized as a secret polynomial whose input is homomorphically-encrypted data as in a keyed-hash where the only the key is secret. In general, the computed polynomial may have a multivariate input.

b) **Only the algorithm is secret** (not the input).

The algorithm is characterized by a polynomial some of whose coefficients are secret large-numbers (mod N) and the input is public. An example is a Secret Function for Key Generation (SFKG) whose input is a public ECM (used in broadcast CA systems). Yet another similar example is a secret HMAC function depicted below. The HMAC runs securely on an open machine; the last step in its process is delivered to the secure client for verification.



c) **Both the algorithm and input are secret**.
Both the algorithm and its input are secret — not known to the processing entity. (note that both the algorithm and its input must be encrypted with the same homomorphic-encryption key.)

Such an untrusted platform could be a public (open) cloud server, an STB host, a mobile phone ACPU, etc.; the decryption of the result is done by the secure client, e.g., smart card, SIM, RFID, or secure execution environment with secure OTP of a low performance device.

4.2 Securing OSS Public Signature against Pollard Attack, HoMOSS (HomoMorphic OSS).

The OSS public signature scheme described in [6] is very efficient both for signing and verifying; signing and verifying requires few modular multiplications.

Below we summarize its highlights.

A signature of a message is a pair (X,Y) satisfying the quadratic bivariate modular (QBM) equation $X^2 + \mu Y^2 = M \pmod N$, where M is the hash of the message.

The public key is the pair (μ, N), and the private key is ε , where $\varepsilon^2 = -\mu \pmod N$.

The signer calculates M and uses the private key ε to find a solution, (X,Y), for the above QBM equation as follows: he selects a random value r modulo N and computes X and Y

$X = \frac{1}{2}(r + \frac{M}{r})$; $Y = \frac{1}{2\varepsilon}(r - \frac{M}{r})$. It is easily seen that X and Y solve the above QBM equation.

The problem of finding a solution for a QBM equation was thought to be hard, and it was therefore assumed that it is hard for an attacker who has no knowledge of ε to forge a signature.

The OSS signature scheme was broken by Pollard and Schnorr [9]. They presented an algorithm for finding a solution of the bivariate quadratic equation of the form $X^2 + \mu Y^2 = M \ \ mod\ N$ where the factorization of N or the square root of $\mu \ mod\ N$ are not known.

We present a modification of the original OSS scheme to protect against the Pollard Schnorr attack.

HoMOSS — Homomorphic-Modified OSS

For U a 2x2 matrix over Z_N we define a commutative ring of 2x2 matrices over Z_N, $C_U = \{B \mid BU = UB \mid B \in M_{2\times 2}(Z_N)\}$, the subring of 2x2 matrices over Z_N that commute with U.

It can be verified that C_U is a module with dimension 2 over Z_N, and can be represented with two parameters t_1, t_2: $C_U = \{t_1 U + t_2 I \mid t_1, t_2 \epsilon Z_N\}$.

The homomorphic non-deterministic encryption is induced by a one-way mapping: $Z_N \oplus Z_N \rightarrow C_U$.

The HoMOSS scheme presented in this section is based on equations of commutative matrices that are obtained from the consideration of two coupled OSS (QBM) equations' private keys.

We choose to illustrate HoMOSS by the use of the MORE method.

The Public key is a 2x2 matrix U over Z_N . The characteristic polynomial of U should not be a square of a linear polynomial and its root resides in Z_N.

The Private key is a matrix S , and the values $\varepsilon_1, \varepsilon_2$ (square roots of μ_1, μ_2 ,respectively), satisfying

$$U = S\begin{pmatrix} \mu_1 & 0 \\ 0 & \mu_2 \end{pmatrix} S^{-1}$$

Signing a message:

1. The message is hashed to produce two Z_N parameters: t_1, t_2 .
2. A matrix M in C_U is calculated: $M = t_1 U + t_2 I$.
3. The matrix M has a representation $M = S\begin{pmatrix} M_1 & 0 \\ 0 & M_2 \end{pmatrix} S^{-1}$, the signer calculates M_1 M_2 from M.
4. The signer knows $\varepsilon_1, \varepsilon_2$ the square roots of μ_1, μ_2, respectively. He calculates a solution X_1, Y_1 of the equation $X_1^{\,2} - \mu_1 Y_1^{\,2} = M_1 \pmod N$ and a solution X_2, Y_2 of the equation $X_2^{\,2} - \mu_2 Y_2^{\,2} = M_2 \pmod N$ as in the original OSS scheme.
5. The signer computes the following two matrices in C_U :
   $A = S\begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix} S^{-1},\ B = S\begin{pmatrix} Y_1 & 0 \\ 0 & Y_2 \end{pmatrix} S^{-1}.$
6. Each of the matrices A and B in C_U can be represented with two modulo N parameters. Thus the signature is comprised of four modulo N numbers ($a_{11}, a_{12}, b_{11}, b_{12}$; from the first row of A and B, respectively).

Verifying a signature:

The verifier receives the message and signature — 4 modulo N numbers, ($a_{11}, a_{12}, b_{11}, b_{12}$).

1. The verifier recovers the missing elements of A and B (each is a known linear combination of two elements of the signature).
2. The verifier calculates the hash of the message and produces the Z_N elements: t_1, t_2 .
3. The verifier calculates matrix M in C_U : $M = t_1 U + t_2 I$. And,
4. Verifies that the matrix equation: $A^2 + U \cdot B^2 = M$ holds to accept the signature as valid.

Homomorphic Modified OSS - simplified version

A simplified version of the scheme is presented

Signing a message:

1. The message is hashed to produce a number m in Z_N .
2. The signer knows $\varepsilon_1, \varepsilon_2$ the square roots of μ_1, μ_2, he calculates a solution X_1, Y_1 of the equation $X_1^{\,2} - \mu_1 Y_1^{\,2} = m \pmod N$ and a solution X_2, Y_2 of the equation $X_2^{\,2} - \mu_2 Y_2^{\,2} = m \pmod N$ as in the original OSS scheme.
3. The signer computes the following two matrices in C_U :
   $A = S\begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix} S^{-1},\ B = S\begin{pmatrix} Y_1 & 0 \\ 0 & Y_2 \end{pmatrix} S^{-1}.$

4. Each of the matrices A and B in C_U can be represented using two modulo N parameters. Thus the signature is comprised of four modulo N numbers ($a_{11}, a_{12}, b_{11}, b_{12}$; from the first row of A and B, respectively).

Verifying a signature:

The verifier receives the message and signature (4 modulo N numbers).

1. The verifier recovers the missing elements of A and B (each is a known linear combination of two elements of the signature).
2. The verifier calculates the hash of the message and produces the number m. And
3. Verifies that the matrix equation: $A^2 + U \cdot B^2 = m \cdot I$ holds to accept the signature.

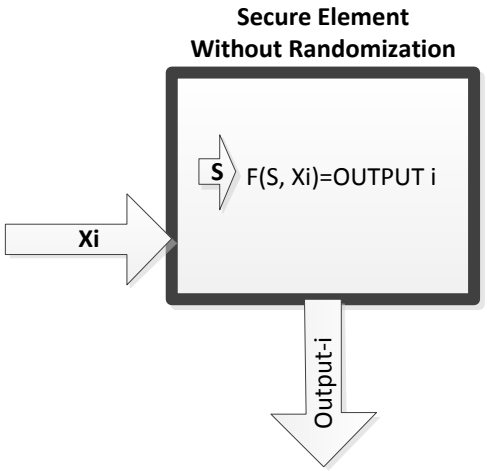
It can be readily shown that a valid signature will satisfy the above matrix equation.

HoMOSS Resistance against Pollard and Schnorr attack

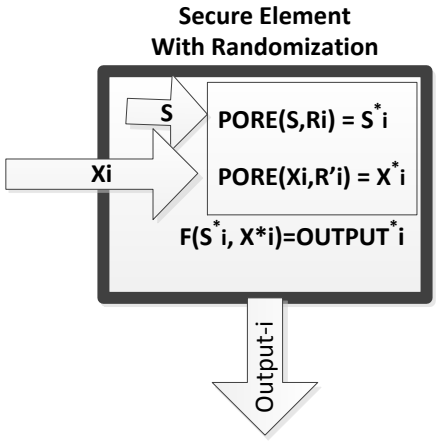
The Pollard and Schnorr attack [9] is predicated on the fact that the OSS QBM equation over Z_N can be viewed as a particular multiplicative norm in Z_N , Z_L or Z (L being a prime). The particular multiplicative norm enables Pollard to start by finding a solution over Z_L and then define a recursive process to yield a solution of the OSS QBM over Z that is transformable to Z_N .
The proposed protection of the OSS scheme stems from the use of matrices in the commutative ring C_U instead of elements in Z_N. We presume that it is hard to find a norm over matrices that can be utilized in Pollard's attack.

4.3 Practical Fully Homomomorphic Randomization of Arbitrary Data

The randomization of arbitrary data is presumed to take place in a secure environment; otherwise one could deduce the randomization key from given randomized arbitrary data. A potential application is to protect against side-channel attacks on algorithm execution aimed at getting the algorithm's sensitive data. Side-channel attacks are mounted on secure execution environment (e.g., secure smart cards) where leakage information (e.g., timing or power consumption), is gathered to glean secret information. For example, in the AES key attack, the attacker runs the algorithm repeatedly with varying plaintexts but with the same AES key while gathering the side-channel information in order to reveal the AES key. However, in this non-deterministic approach, the randomized key and plaintext change per cipher run thereby foiling such attack, (see Figure XX below). Note, AES inherently lends itself to an efficient representation as a sequence of algebraic operations over the F_{256} that can be efficiently run.



Repeat for different Xi and analyze the data (e.g. DPA) or other behavior of Output-i to deduce bits of secret(s) S



Randomization (of inputs S and Xi) prevents such attack as every run of F presents different OUTPUT*i corresponding to randomized inputs. The Homomorphic property allows F to be computed from beginning to end without correction for the randomization effect per step.

The randomization can provide protection against fault attacks. An induced fault is detectable since the computations' validity can be verified; see Appendix for discussion on verification.

5. Performance and Throughput

Referring to section 3.2 (PORE) we see that the cost of a single input encryption / decryption (or randomization) is one multiplication of large numbers. The throughput is two large numbers in the steady state (two additional PPV coefficients are passed only once as the PPV does not change per encryption). For the result of a single homomorphic computation only two large numbers are returned.

Cost of calculating a product of two distinct variables is five multiplications of large numbers.
Cost of squaring a variable is four multiplication, and computing x^{2n} results in 4n multiplications.

Division can be done in ten multiplications plus one division.

6. Security

The **fully**-homomorphic (symmetric) encryption schemes presented in this paper are essentially secret linear transformations that are utilized to construct a non-deterministic encryption schemes. In general, secret linear schemes are inherently ill-suited for cryptographic applications as the secret transformation is easily recovered using linear algebra techniques once sufficient number of plain text and cipher text pairs are known. Thus such transformations are not used for traditional data encryption where it is usually assumed the adversary knows the plain text. However, the specific linear transformations presented in this paper are deemed secure under specific conditions, and has benefits for practical (efficient) homomorphic computation. The proposed linear transformations provide sound

security in various applications under two conditions: 1. the plaintext is not known and 2. The plain text can be regarded as "random data".

In this section we claim that the homomorphic encryption scheme, MORE, as presented in section 3.1, ensures high degree of security under the assumption that the plaintext is comprised of unknown random large-numbers in Z_N.

Theorem: One who knows the encrypted values of random plain text values $X_1 \ldots X_k$ in Z_N, and is capable of calculating any one of the plain text values (inputs) can factor N.
Proof: without loss of generality we show that if an adversary can calculate the first input, X_1, he can factor N. We first prove the theorem for k=1, we denote by A_1 the encrypted value of X_1:

$$A_1 = Enc(X_1)$$

Lemma 1: An adversary who gets A_1 does not get more information about X_1 than an adversary who only knows the characteristic polynomial of A_1.

Proof: The process encrypting X_1 requires an independent selection of a random secret value Y_1 in Z_N and a secret random invertible matrix S over Z_N. We show below (a) that the distribution of the encrypted matrix A_1 is indistinguishable from the uniform distribution over all matrices whose characteristic polynomial is given by f(x)= (x - X1)(x - Y1). We then show (b) that one who knows $f(x)$ is capable of uniformly selecting a matrix among all matrices with characteristic polynomial equals to $f(x)$. As a result we conclude that when one is given A_1 the information one gets about X_1 is the same information provided by knowledge of the characteristic polynomial of A_1.

a. The distribution of A_1 is indistinguishable from uniform distribution over all matrices whose characteristic polynomial is $f(x)$, $f(x) = (x - X_1)(x - Y_1)$. A matrix B having $f(x)$ as a characteristic polynomial has the following representation $B = T\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix} T^{-1}$ for some regular matrix $T \in GL_{2x2}(Z_N)$. We denote by $S_{f(x)}$ the set of all 2x2 matrices over Z_N having $f(x)$ as a characteristic polynomial. One can define a partitioning of the regular 2x2 matrices over Z_N ($GL_{2x2}(Z_N)$). For any matrix $B \in S_{f(x)}$ we define the set of regular matrices S_B to be: $S_B = \{S | S \in GL_{2x2}(Z_N)\ ;\ S\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix} S^{-1} = B\}$. We show that the size of S_B is independent of B, X_1 , Y_1 as long as the inequalities $X_1 \neq Y_1$, $X_1 \neq 0$, $Y_1 \neq 0 \bmod P$ and Q are satisfied*. Let T_1, T_2 be matrices in S_B iff

$T_1\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix} T_1^{-1} = B$; $T_2\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix} {T_2}^{-1} = B \rightarrow T_1\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix} T_1^{-1} = T_2\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix} {T_2}^{-1}$ iff $\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix} T_1^{-1} \cdot T_2 = T_1^{-1} \cdot T_2 \begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix}$ i.e., $E \triangleq T_1^{-1} T_2$ commutes with $\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix}$. We conclude that any matrix T_2 in S_B can be represented as $T_2 = T_1 \cdot E$ for some invertible matrix E that commutes with $\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix}$. A matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ commutes with $\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix}$ iff

$b \cdot X_1 = b \cdot Y_1 \ mod\ N$ and $c \cdot X_1 = c \cdot Y_1 mod\ N$; since $X_1 \neq Y_1$ both mod P and mod Q, we get $c = 0\ mod\ N$ and $d = 0\ mod\ N$, and E has the form $\begin{pmatrix} a & 0 \\ 0 & d \end{pmatrix}$ $a, d \in Z_N$. E is an invertible matrix, i.e., $(a, N) = 1$ and $(d, N) = 1$. There are $\Phi(N)^2 = (P-1)^2(Q-1)^2$ such matrices. QED

*We note that the events: $(\mathrm{X}_1, N) \neq 1$, $(\mathrm{X}_1 - Y_1, N) \neq 1$ or $(\mathrm{Y}_1, N) \neq 1$, are of negligible probability.

b. One who knows the characteristic polynomial of A_1, $f(x)$, is capable of selecting a uniformly distributed matrix among all matrices with characteristic polynomial $f(x)$.
Let $f(x) = x^2 + a \cdot x + b$, we define the matrix $B = \begin{pmatrix} 0 & 1 \\ -b & -a \end{pmatrix}$ with characteristic polynomial $f(x)$. For $\mathrm{X}_1 \neq Y_1 \bmod \mathrm{P}$ and Q , there is an invertible 2x2 matrix U over Z_N such that $B = U\begin{pmatrix} \mathrm{X}_1 & 0 \\ 0 & \mathrm{Y}_1 \end{pmatrix} U^{-1}$. We select a random invertible 2x2 matrix R over Z_N, and define $D = RBR^{-1}$; $D \in S_{f(x)}$. We claim that D is uniformly distributed among all matrices with characteristic polynomial $f(x)$. Above it is shown that
for $\mathrm{X}_1 \neq Y_1$, $\mathrm{X}_1 \neq 0$, $\mathrm{Y}_1 \neq 0 \bmod \mathrm{P}$ and Q the number of matrices Tgenerating the same matrix $T\begin{pmatrix} \mathrm{X}_1 & 0 \\ 0 & \mathrm{Y}_1 \end{pmatrix} T^{-1}$ is constant. Since $D = RU\begin{pmatrix} \mathrm{X}_1 & 0 \\ 0 & \mathrm{Y}_1 \end{pmatrix} (RU)^{-1}$ and RU is a uniformly distributed regular random matrix, we conclude that the matrix D is distributed uniformly among the matrices in $S_{f(x)}$. QED

We conclude the proof of the theorem for k=1, by observing that if one is capable of finding the plain text $\mathrm{X}_1$from its encrypted value, A_1, then he is capable of finding a root of a random quadratic polynomial modulo N, $f(x)$; this problem is known to be as hard as factoring N.

The general case: One who gets the encrypted values of random plain text values $\mathrm{X}_1 \ldots \mathrm{X}_k$ in $\mathrm{Z_N}$, and is capable of calculating the plain text value X_1 can factor N.

Lemma 2: The information about X_1 provided by **t**he encrypted values: $A_1 = Enc(X_1), A_2 = Enc(X_2), \ldots, A_k = Enc(X_k)$, is the same as the information that A_1 provides about X_1.

Proof: The proof of the lemma has two steps. In step a. we show that the matrices $A_2, A_3, \ldots A_k$ are randomly selected matrices from the set of all matrices that commute with A_1; denoted as C_{A1}. In step b. we show that one who gets A_1 can randomly select k-1 matrices of C_{A1}.
As a result of these steps we conclude that one who knows the sequence of matrices $A_1, A_2, \ldots A_k$ doesn't get more information about X_1 than one who only sees A_1. Given above proof the special case k=1 we can state that finding X_1 is as hard as factoring N.

a. Define the set of all matrices that commute with A_1: $C_{A1} = \{B | BA = AB\ ; B \in M_{2x2}(Z_N)\}$.
We have $A_1 = S\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix} S^{-1}$, for B that commutes with A_1 we have
$B \cdot S\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix} S^{-1} = S\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix} S^{-1} \cdot B \rightarrow S^{-1}BS\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix} = \begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix} S^{-1}BS$, i.e., $\begin{pmatrix} X_1 & 0 \\ 0 & Y_1 \end{pmatrix}$ commutes with $S^{-1}BS$. We showed in above lemma 1 that such matrices are of the form $\begin{pmatrix} \mathrm{a} & 0 \\ 0 & \mathrm{d} \end{pmatrix}$ for $a, d \in Z_N$ where, $\mathrm{X}_1 \neq Y_1 \bmod \mathrm{P}$ and Q . Therefore, $B = S\begin{pmatrix} \mathrm{a} & 0 \\ 0 & \mathrm{d} \end{pmatrix} S^{-1}$, i.e., C_{A1} is a 2 dimensional sub-module of the $\mathrm{Z_N}$ module of the 2x2 matrices over $\mathrm{Z_N}$. The matrices A_i for $2 \leq i \leq k$ have the form $S\begin{pmatrix} \mathrm{X}_i & 0 \\ 0 & \mathrm{Y}_i \end{pmatrix} S^{-1}$ for $\mathrm{X}_i, \mathrm{Y}_i$ random elements in Z_N. We therefore conclude that one can regard $A_2, \ldots, A_k$ as an independent random elements of C_{A1}.

b. We show that one who knows A_1 can select random elements from C_{A1} by using a different representation of C_{A1}. Let $H_{A1} = \{a \cdot A_1 + b \cdot I | a, b \in Z_N\}$; it is easily verified that matrices of the form $aA_1 + bI$ commute with A_1 and also form a sub module of dimension 2 of the Z_N module of the 2x2 matrices over Z_N. Therefore, $H_{A1} = C_{A1}$. One randomly selects parameters a_i, b_i in Z_N to yield a random choice of matrix B_i in C_{A1}: $B_i = a_i \cdot A_1 + b_i \cdot I$.

Q.E.D.

A similar proof exists for a non-trivial rational function of the inputs.

7. Conclusion

Fully homomorphic efficient methods were presented for domain-specific applications. For symmetric key FHE the applications domain is security algorithms whose input is random large-numbers in Z_N. For randomization the input data is arbitrary elements of a commutative ring without any constraints of size. Fault detection capability is inherent allowing for verification of computed data. The cost of computation of encrypted data is reflected in the cost of product of two variables – five modular large-number multiplications. Encryption or decryption requires only one modular large-number multiplication. In addition, the methods employ operations in an algebraic domain of rings of commutative matrices which are likely new to cryptographic applications. We hope that this can be used to augment security in some algorithms as has been presented above for OSS, and to improve protection against some side-channel attacks.

A challenging area for search is finding new algorithms that will allow us to relax the current constraints of encryption for the proposed methods, e.g., large random values in ZN, such that arbitrary data can be efficiently encrypted for FHE application or for Somewhat homomorphic applications. Further investigation may shed light on our presumption that it is hard to find a norm over matrices that can be utilized in Pollard's attack.

8. Appendix: The Methods – Practical FSERFs - Generalized

Each of the isomorphic methods presented is an efficient non-deterministic Fully-homomorphic Symmetric Encryption and Randomization Function (FSERF). The input, IN, of the functions is comprised of k input elements in a commutative ring (CR) or in ring Z_N. Randomization is defined over a commutative ring (CR), and encryption is over ring Z_N (all operations are mod N, where N is a product of two (secret) primes). Use of large random numbers mod N for the input IN and for the random parameters employed in the methods will provide a sufficient condition for security of the encryption, (see section 6).

Two fundamental isomorphic methods are defined below; each randomizes or encrypts a set i of any m distinct input elements of IN (where 0<m<k+1 and m<n) denoted as input X_{im} (= (X_1, X_2, …,X_m)). The basic methods are a matrix-based method, MORE (Matrix Operation for Randomization or Encryption), and a

polynomial-based method, PORE (Polynomial Operation for Randomization or Encryption). Additionally, a 'compound' method can be constructed by successively applying the basic methods.

Frequently, m equals 1; a method uses one input element (m=1, $X_{i1} = X_j$ (j=(1..k)) per method application to allow for computation of (arbitrary) multivariate input in functions whose operations are addition, multiplication and division. However, if, for example, there are two known multivariate functions with two distinct sets of inputs, e.g., ($X_1, X_2, \ldots, X_L$) and ($X_{L+1}, X_{L+2}, \ldots, X_s$), respectively, then no pair of variables in each of the input sets should be jointly encrypted (or randomized), i.e., with a single application of the method.

A compound method can be constructed by the successive application of the fundamental MORE and PORE.

8.1 The matrix method, MORE.

A randomly chosen secret nxn invertible matrix, S, in CR (or Z_N), is used as a symmetric key for randomization (or encryption). For each set X_{im} of m distinct input elements selected from IN and denoted as $X_1, X_2, \ldots, X_m$ that are to be jointly randomized (or encrypted) we select n-m random numbers $Y_1, Y_2, \ldots Y_{n-m}$, where the input elements in X_{im} and $Y_1, Y_2, \ldots Y_{n-m}$ are placed in order on the diagonal of a nxn diagonal matrix. We denote MORE's output matrix as Aim and define it as:

$$MORE(X_{im}) = A_{im} = S \begin{pmatrix} X_1 & 0 & & & \cdots & \cdots & 0 \\ 0 & X_2 & 0 & & & \cdots & \vdots \\ \vdots & 0 & \ddots & 0 & \cdots & & \\ & & \ddots & X_m & 0 & \cdots & \\ & & & 0 & Y_1 & \ddots & 0 \\ & \vdots & & \vdots & 0 & Y_2 & 0 \\ & & & & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & & & 0 & Y_{n-m} \end{pmatrix} S^{-1} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$$

Where, A_{im} is the randomized value or cipher text for plaintext $X_1, X_2, \ldots, X_m$, depending on the use of CR, or Z_N (modulo N), respectively.

We note, again, that no pair of input variables represented by $X_1, X_2, \ldots$, or X_m is used in a single multivariate-input function, i.e., each of the encrypted (or randomized) elements (variables) must be used by a distinct function.

8.2 The Polynomial method, PORE.

PORE is a polynomial-based method where $PORE(X_{im}) = A_{im} = (a_{i0}, a_{i1}, \ldots, a_{i\,n-1})$ in CR or Z_N.
We select n random numbers, $v_1, v_2, \ldots, v_n$, in CR or Z_N, to define a public polynomial PP(v)= $\prod_{i=1}^{n}(v -$

$v_i) = \sum_{j=0}^{n} C_j \cdot v^j$ (C_n=1). We let $P(v)$ be any function in variable v of the form $\sum_{j=0}^{n-1} a_{ij} \cdot v^j$ which satisfies the equations: $\sum_{j=0}^{n-1} a_{ij} \cdot {v_1}^j = X_1$, $\sum_{j=0}^{n-1} a_{ij} \cdot {v_2}^j = X_2$, …. , $\sum_{j=0}^{n-1} a_{ij} \cdot {v_m}^j = X_m$.

To define $PORE(\quad)$ for a particular X_{im} we assign n-m chosen random values in CR or Z_N , to $a_{i,m}, a_{i,m+1} \ldots, a_{i,n-1}$, that would yield a solution for the unknown coefficients $a_{i,0}, a_{i,1} \ldots, a_{i,m-1}$ in the above equations; (else, absent a solution, a new set of n-m random values is selected in another attempt to solve the above equations for the unknown coefficients). Or, alternatively, we select n-m random values R_1, …, R_{n-m} in CR or Z_N that would solve the following n simultaneous equations:

$\sum_{j=0}^{n-1} a_{ij} \cdot {v_1}^j = X_1$, $\sum_{j=0}^{n-1} a_{ij} \cdot {v_2}^j = X_2$, …. , $\sum_{j=0}^{n-1} a_{ij} \cdot {v_m}^j = X_m$, $\sum_{j=0}^{n-1} a_{ij} \cdot {v_{m+1}}^j = R_1$, $\sum_{j=0}^{n-1} a_{ij} \cdot {v_{m+2}}^j = R_2$,…., $\sum_{j=0}^{n-1} a_{ij} \cdot {v_n}^j = R_{n-m}$ for unknowns $a_{i0}, a_{i1}, \ldots, a_{i\,n-1}$, thereby producing, as above, for X_1 , X_2,… X_m a random text comprising the set $(a_{i0}, a_{i1}, \ldots, a_{i\,n-1})$ and public (C_0, C_1, …, C_{n-1}, C_n) of PP(v).

This produces the joint randomization (or encryption) output , A_{im}, for plaintext X_1, X_2,… ,X_m, in the form of the set $(a_{i0}, a_{i1}, \ldots, a_{i\,n-1})$ and a public set of coefficients (C_0, C_1, …, C_{n-1}, C_n) of P(v). Note, that the set of coefficients (C_0, C_1, …, C_{n-1}, C_n) is required for performing arithmetic operations with randomized (or encrypted) elements.

8.3 Compound Methods

We note that other methods can result from the successive application of a mix of functions PORE(X_i) and MORE(X_i), as in:

$$\text{MORE(PORE}(X_i)) = (\, MORE(\boldsymbol{a_{i0}}),\ \text{MORE}(\boldsymbol{a_{i1}}), \ldots, MORE(\boldsymbol{a_{i\,n-1}})) =$$

$$\left(\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}_{\boldsymbol{i0}}, \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}_{\boldsymbol{i1}}, \ldots., \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}_{\boldsymbol{i\,n-1}}\right);$$ and conversely

$$\text{PORE(MORE}(X_i)) = \text{PORE}\left(\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}\right) \text{ where } \text{PORE}(a_{mj}) = (a_0, a_1, \ldots, a_{n-1})_{mj}, \text{ therefore}$$

$$\text{PORE(MORE}(X_i)) = \begin{pmatrix} (a_0, a_1, \ldots, a_{n-1})_{11} & \cdots & (a_{i0}, a_{i1}, \ldots, a_{i\,n-1})_{1n} \\ \vdots & \ddots & \vdots \\ (a_0, a_1, \ldots, a_{n-1})_{n1} & \cdots & (a_0, a_1, \ldots, a_{n-1})_{nn} \end{pmatrix}.$$

8.4 Homomorphic operations for multivariate functions

It can be readily shown that the methods above are fully homomorphic (addition and multiplication) and, moreover, are homomorphic for division.

We define below the operation of addition, multiplication and division of encrypted (or randomized) values subject to the above constraints of CR and Z_N, respectively.

We let A1 and A2 be the encrypted (or randomized) values of X_1 and X_2, respectively, under MORE or PORE, s.t. MORE(X_i) = A_i matrix, and PORE(X_i) = $A_i = a_{i0}, a_{i1}, \ldots, a_{i\,n-1}$ under C_0, C_1, …, C_{n-1}, C_n $of\ P(v) = \sum_{j=0}^{n} C_j \cdot v^j$ (C_n=1).

The Ai's comprise the input of a homomorphic function in which they are added, multiplied, or divided. We thus have for:

Addition

MORE (X_1) + MORE (X_2) = A_1 +A_2 ; and

PORE (X_1) + PORE (X_2) = $a_{10} + a_{20}, a_{11} + a_{21}, \dots, a_{1\,n-1} + a_{2\,n-1}$

Multiplication

MORE (X_1) · MORE (X_2) = A_1 ·A_2 ; and

PORE (X_1) · PORE (X_2) = $r_0, r_1, \dots, r_{n-1}$, the coefficients of the resulting n-1 order polynomial r(v) , where r(v)= $((\sum_{j=0}^{n-1} a_{1j} \cdot v^j) \cdot (\sum_{j=0}^{n-1} a_{2j} \cdot v^j)) \bmod \sum_{j=0}^{n} C_j \cdot v^j$.

Division

1/ MORE (X_2) = A_2^{-1} and MORE (X_1) / MORE (X_2)= $A_1 \cdot A_2^{-1}$; and

Let PORE (X_1) = $a_{10}, a_{11}, \dots, a_{1\,n-1}$ and PORE (X_2) = $a_{20}, a_{21}, \dots, a_{2\,n-1}$and
let 1/PRHT(X_2) = $u_{20}, u_{21}, \dots, u_{2\,n-1}$ be derived by solving:
$(\sum_{j=0}^{n-1} u_{2j} \cdot v^j) \cdot (\sum_{j=0}^{n-1} a_{2j} \cdot v^j)) \bmod \sum_{j=0}^{n} C_j \cdot v^j$ =1 in terms of n unknown coefficients u_{2j} , $(j = 0, \dots, n-1)$, thereby determining

$$\sum_{j=0}^{n-1} g_{2j}(u_{20}, u_{21}, \dots, u_{2\,n-1}, \quad a_{10}, a_{21}, \dots, a_{2\,n-1,}\, c_{10}, c_{21}, \dots, c_{2\,n-1,}) \cdot v^j)$$

where, $g_{2j}(\ )$ is a linear combination of the n unknowns u_{2j}; and

solving n derived equations $g_{20}(\ )$ =1, and $g_{2j}(\ )$ = 0 for j=1,..,n-1 for all u_{2j} j=0,…,n-1, for the n unknowns $u_{20}, u_{21}, \dots, u_{2\,n-1}$; and thus

PORE (X_1) / PORE (X_2) = PRHT (X_1) · (1/PORE(X_2)) =

$((\sum_{j=0}^{n-1} a_{1j} \cdot v^j) \cdot (\sum_{j=0}^{n-1} u_{2j} \cdot v^j)) \bmod \sum_{j=0}^{n} C_j \cdot v^j$.

8.5 Decryption or Derandomization

Once a homomorphic function, f, is computed over the encrypted (or randomized) multivariate input, the returned result has to be decrypted (or derandomized). Let A_i denote the encrypted or randomized X_i input; we receive the result of such a function $f(A_1, A_2, \dots, A_k)$ in one of the following forms:

$$f(A_1, A_2, \dots, A_k) = \text{MORE}\big(\text{f}(\text{X}_1, \text{X}_2, \dots., \text{X}_\text{K})\big) =$$

$$S\begin{pmatrix} f(X_1, X_2, \ldots, X_K) & 0 & 0 \\ 0 & f(Y_{m1}, Y_{m2}, \ldots, Y_{mk}) & 0 \\ 0 & 0 & f(Y_{n-1,1}, Y_{n-1,2}, \ldots, Y_{n-1,k}) \end{pmatrix} S^{-1}$$

$$= \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} = R_M^*;$$

Or,

$f(A_1, A_2, \ldots, A_k)$ is in the form of $a^*_0, a^*_1 \ldots, a^*_{n-1} = R_P^*$;

In the case where the result is in the matrix form of R_M^*, the decryption yields:

$$f(X_1, X_2, \ldots, X_k) = S^{-1} R_M^* S;$$

or, alternatively, using S to determine $S^{-1} = \begin{pmatrix} s'_{11} & \cdots & s'_{1n} \\ \vdots & \ddots & \vdots \\ s'_{n1} & \cdots & s'_{nn} \end{pmatrix}$

$$f(X_1, X_2, \ldots, X_k) = a_{11} + (1/s'_{11}) \cdot \sum_{j=2}^{n} a_{j1} \cdot s'_{1j} = (1/s'_{11}) \cdot \sum_{j=1}^{n} a_{j1} \cdot s'_{1j}$$

And for PORE() we use R_P^* and v_1 to determine:

$$f(X_1, X_2, \ldots, X_k) = \sum_{j=0}^{n-1} a_j^* \cdot v_1^j.$$

8.6 Verification of integrity of computations

The methods provide a mechanism to verify that a returned result of a calculation performed by a third party on encrypted multivariate input is valid. We designate the returned result as R_M^* , or R_P^* depending upon whether MORE (denoted below as M) or PORE (denoted below as P) is used. The result is of a homomorphic calculation denoted f performed on $A_1, A_2, \ldots, A_k$ wherein A_i is equal to one of $M(X_i)$ and $P(X_i)$; $f(A_1, A_2, \ldots, A_k)$ is equal to either $M(f(X_1, X_2, \ldots, X_k)) = f(M(X_1), M(X_2), \ldots, M(X_k))$, or to $P(f(X_1, X_2, \ldots, X_k)) = f(P(X_1), P(X_2), \ldots, P(X_k))$.

The verifier receives the result of $f(A_1, A_2, \ldots, A_k)$ in one of the following forms:

$$M(f(X_1, X_2, \ldots, X_K)) =$$

$$S \begin{pmatrix} f(X_1, X_2, \ldots, X_K) & 0 & 0 \\ 0 & f(Y_{m1}, Y_{m2}, \ldots, Y_{mk}) & 0 \\ 0 & 0 & f(Y_{n-1,1}, Y_{n-1,2}, \ldots, Y_{n-1,k}) \end{pmatrix} S^{-1}$$

$$= R_M^*; \quad \text{and}$$

$f(A_1, A_2, \ldots, A_k)$ in the form of $a^*_0, a^*_1 \ldots, a^*_{n-1}, = R_P^*$;

Where the result is in the form of R_M^*:

The verifier has precomputed $f(Y_{m1}, Y_{m2}, \ldots, Y_{mk})$, for some m , in (1,2,…,n-1) where Y_{mj} the m[th] random used in encrypting X_j. This may be done well before computing the encrypted input.

It then decrypts R^*_M to determine $f(Y_{m1}, Y_{m2}, \ldots, Y_{mk})$, from the resulting matrix diagonal of the m+1 row. Finally it compares the decrypted $f(Y_{m1}, Y_{m2}, \ldots, Y_{mk})$ with the precomputed one, and deems the result of the computation of f(Xi) verified if a match occurs.

Where the result is in the form of R_P^*:

The verifier has precomputed $f(R_{m1}, R_{m2}, \ldots, R_{mk})$, for some m , in 1,2,…,n-1 where R_{mj} is the m[th] random used in encrypting X_j. This may be done well before computing the encrypted input.

It uses $(v_1, v_2, \ldots, v_n)$ and picks the same m in (1,..,n-1) as above to determine: $\sum_{j=0}^{n-1} a_j^* \cdot v_{m+1}{}^j = f(R_{m1}, R_{m2}, \ldots, R_{mk})$, and it compares the determined $f(R_{m1}, R_{m2}, \ldots, R_{mk})$ with the pre-computed, and deems the result the of the computation of $f(X_1, X_2, \ldots, X_K)$ verified if a match occurs.

9. References

1. Can Homomorphic Encryption be Practical? Kristin Lauter, Micahel Naehring, Vinod Vikuntanathan
2. Zvika Brakerski and Vinod Vaikuntanathan . Efficient fully homomorphic encryption from (standard) LWE. FOCS, 2011.
3. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, STOC , pages 169–178. ACM, 2009
4. Implementing Gentry's fully-homomorphic encryption scheme, Gentry, S Halevi - Advances in Cryptology–EUROCRYPT 2011,
5. R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, 1978
6. Ong, Schnorr, Shamir An Efficient signature based on quadratic equations, proceedings of the 16'th symposium on theory of computing pp.208-216 1984
7. Rivest, R.; A. Shamir; L. Adleman (1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". *Communications of the ACM* **21** (2): 120–126 .
8. FIPS -186-3 The third and current revision to the official DSA specification.

9. J.Pollard & C.Schnorr "An efficient solution of the congruence $x^2 + ky^2 = m$ modulo n "IEEE transactions on Information Theory, vol. IT-33 no.5., September 1987 pp 208-216