

SYSTEM AND METHODS FOR VALIDATING AND PERFORMING OPERATIONS ON HOMOMORPHICALLY ENCRYPTED DATA

FIELD OF THE INVENTION

[0001] This disclosure relates to homomorphically encrypted data systems and methods, and more specifically, to validation of, and performing operations on, homomorphically encrypted confidential data without decryption of the confidential data.

BACKGROUND OF THE INVENTION

[0002] Privacy of sensitive personal information is an increasingly important topic as more personal data is transmitted and shared, particular via the use of wireless transmissions and cloud data services. Privacy issues arise due to the fear of having a security breach on cloud servers or due to the fear that the service providers themselves misuse this sensitive information. Standard encryption schemes try to address these concerns by devising encryption schemes that are harder to break, yet they do not solve the possible misuse of this sensitive data by the service providers.

[0003] Privacy of confidential and personal data, such as financial and medical data, is a paramount concern. However, access to this data for legitimate purposes, such as to execute a financial transaction, or providing access to medical data for conducting studies or addressing health emergencies, is also important.

[0004] Accordingly, there is a need for improvement in the art.

SUMMARY OF THE INVENTION

[0005] According to an embodiment of the invention, there is provided a method for securely transmitting and extracting information from encrypted data without fully decrypting the data, comprising: receiving a data request, with at least a portion of the data request encrypted according to a homomorphic encryption scheme, and the encrypted portion of the data request comprising at least a set of confidential data; retrieving one or more sets of encrypted comparison data from a database, each set of encrypted comparison data encrypted according to a homomorphic encryption scheme; comparing the encrypted set of confidential data from the data request with each set of encrypted comparison data using one or more homomorphic operations to determine which set of encrypted comparison data matches the encrypted set of confidential data and validating the set of confidential data upon a match; generating an

encrypted indicator indicating success or failure of validating the set of confidential data; and forwarding the encrypted indicator to a party associated with the data request, wherein the set of confidential data is never decrypted during the method.

[0006] According to a further embodiment of the invention, there is provided a data system comprising: a network interface for connecting to a network to receive homomorphically encrypted data from a plurality of remote data devices, wherein the homomorphically encrypted data is associated with users associated with the remote data devices; and a calculation engine connected to the network interface, the calculation engine configured to perform homomorphic calculations on the data to obtain calculation results that can be selectively decrypted.

[0007] According to a further embodiment of the invention, there is provided a device for performing homomorphic calculations, the device comprising a memory and at least one processor, the device configured to model a homomorphic calculation as a polynomial series and to compute a value of the polynomial series using encrypted data to obtain an encrypted result, the encrypted data being encrypted according to an encryption scheme, the device further configured to perform a homomorphic relational operation to compare the encrypted result with an encrypted condition, the encrypted condition being encrypted according to the encryption scheme, the device further configured to output a result of the comparison.

[0008] According to a further embodiment of the invention, there is provided a non-transient computer-readable medium containing computer-readable instructions which when executed by a computer processor perform the method of: receiving a data request to validate data with at least a portion of the data request encrypted according to a homomorphic encryption scheme, and the data request comprising at least a set of confidential data; retrieving one or more sets of encrypted comparison data from a database, each set of encrypted comparison data encrypted according to a homomorphic encryption scheme; comparing the set of confidential data to each set of the encrypted comparison data using one or more homomorphic operations to determine which set of encrypted comparison data matches the set of confidential data and validating the set of confidential data upon a match; generating an encrypted indicator indicating success or failure of validating the set of confidential data; and forwarding the encrypted indicator to a party associated with the data request, wherein the set of confidential data is never decrypted.

[0009] According to a further embodiment of the invention, there is provided a method of processing a secure financial transaction, comprising: receiving an at least partially homomorphically encrypted request to complete a financial transaction, which is homomorphically encrypted according to a lattice-based Fully Homomorphic Encryption (FHE) scheme, and the transaction request comprising confidential cardholder data including an account number, non-confidential cardholder data and transaction data; retrieving one or more sets of homomorphically encrypted comparison cardholder data encrypted according to a lattice-based FHE scheme; comparing the confidential cardholder data to each set of the comparison cardholder data using one or more homomorphic operations to determine which set of comparison cardholder data matches the confidential cardholder data and validating the confidential cardholder data; generating and encrypting an indicator indicating authorization or rejection of the request to complete the financial transaction based upon at least the validation of the confidential cardholder data; and forwarding the encrypted indicator indicating authorization or rejection of the request to complete the financial transaction to a party seeking authorization to complete the financial transaction, wherein the confidential cardholder data is never decrypted during the method.

[0010] According to a further embodiment of the invention, there is provided a non-transient computer-readable medium containing computer-readable instructions which when executed by a computer processor perform the method of: receiving an at least partially homomorphically encrypted request to complete a financial transaction, which is homomorphically encrypted according to a lattice-based Fully Homomorphic Encryption (FHE) scheme, and the transaction request comprising confidential cardholder data including an account number, non-confidential cardholder data and transaction data; retrieving one or more sets of homomorphically encrypted comparison cardholder data encrypted according to a lattice-based FHE scheme; comparing the confidential cardholder data to each set of the comparison cardholder data using one or more homomorphic operations to determine which set of comparison cardholder data matches the confidential cardholder data and validating the confidential cardholder data; generating and encrypting an indicator indicating authorization or rejection of the request to complete the financial transaction based upon at least the validation of the confidential cardholder data; and forwarding the encrypted indicator indicating authorization or rejection of the request to

complete the financial transaction to a party seeking authorization to complete the financial transaction, wherein the confidential cardholder data is never decrypted during the method.

[0011] According to a further embodiment of the invention, there is provided a method of performing a comparison with encrypted data, the method comprising: modeling a calculation as a polynomial series; computing a value of the polynomial series using encrypted data to obtain an encrypted result, the encrypted data being encrypted according to an encryption scheme; performing a homomorphic relational operation to compare the encrypted result with an encrypted/non-encrypted condition, the encrypted condition being encrypted according to the encryption scheme; and outputting a result of the comparison.

[0012] According to a further embodiment of the invention, there is provided a medical data system comprising: a network interface for connecting to a network to receive homomorphically encrypted data from a plurality of remote medical devices, wherein the homomorphically encrypted data is representative of measured physiological data of patients associated with the remote medical devices; and a calculation engine connected to the network interface, the calculation engine configured to perform homomorphic calculations on the data to obtain calculation results that can be selectively decrypted.

[0013] According to a further embodiment of the invention, there is provided a device for performing homomorphic calculations, the device comprising memory and at least one processor, the device configured to model a homomorphic calculation as a polynomial series and to compute a value of the polynomial series using encrypted data to obtain an encrypted result, the encrypted data being encrypted according to an encryption scheme, the device further configured to perform a homomorphic relational operation to compare the encrypted result with an encrypted condition, the encrypted condition being encrypted according to the encryption scheme, the device further configured to output a result of the comparison.

[0014] Other aspects and features according to the present application will become apparent to those ordinarily skilled in the art upon review of the following description of embodiments of the invention in conjunction with the accompanying figures.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] The drawings illustrate, by way of example only, embodiments of the present disclosure.

[0016] FIG. 1 is a diagram of a secure computing environment according to an embodiment of the present invention;

[0017] FIG. 2 is a diagram of a remote device according to an embodiment of the present invention;

[0018] FIG. 3 is a diagram of a data system according to an embodiment of the present invention;

[0019] FIG. 4 is a diagram of a key authority system according to an embodiment of the present invention;

[0020] FIG. 5 is a diagram of another data system according to an embodiment of the present invention;

[0021] FIGs. 6A – 6W show equations for performing computations according to an embodiment of the present invention;

[0022] FIGs. 7 – 10 are tables of data used in various examples for an embodiment of the present invention;

[0023] FIG. 11 is a listing of pseudocode for implementing a relational operation according to an embodiment of the present invention;

[0024] FIGs. 12 – 14 show testing parameters and results of tests conducted;

[0025] FIG. 15 is a diagram of a four-party credit card transaction system;

[0026] FIG. 16 is a diagram of a payment processing chain according to an embodiment of the present invention;

[0027] FIG. 17 is a diagram of a secure payment processing system according to an embodiment of the present invention;

[0028] FIG. 18 is a chart showing secret key size reduction according to an embodiment of the present invention;

[0029] FIG. 19 is a chart showing ciphertext size reduction and obviating a flatten function;

[0030] FIG. 20 shows pseudocode for a ciphertext multiplication operation;

[0031] FIG. 21 is a table of example parameter selection according to an embodiment of the present invention; and

[0032] FIGs. 22a – 22g show expressions/equations according to an embodiment of the present invention.

[0033] Like reference numerals indicate like or corresponding elements in the drawings.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0034] Although medical and financial applications form the basis of the inventive examples discussed herein, the inventive techniques discussed have application to other forms of confidential data. In this context, the different devices and parties involved in data acquisition, storage, and analysis may be reconsidered as appropriate for the type of confidential data being processed.

[0035] Homomorphic Encryption/Decryption

[0036] The homomorphic encryption/decryption operations executed on confidential data may be performed as shown in FIGs 1-5 and FIGs 6A-6E. Key generation may be performed at a secure system, such as the key authority system 36. Public keys 68 may be pre-loaded onto the remote devices 20 containing the confidential information (such as a medical device, credit card, remote sensor, etc.) prior to deployment of remote devices 20. Alternatively, or additionally, public keys 68 may be made available to the remote devices 20 via a network resource (such as a public cloud), which may be configured to require credential verification or authorization (e.g., user name and password) to access public keys 68. The encryption engines 62 on board the

remote devices 20, and in the data system 120, may be configured to implement the homomorphic encryption techniques discussed below. The decryption engine 104 at the key authority system 36 may be configured to implement the homomorphic decryption techniques discussed below. The calculation engine 88 at the data systems 22, 120 may be configured to implement the computational techniques discussed below, including the storage and processing of equations, calculation parameters, upper/lower bounds, and intermediate/accumulated results.

[0037] Regarding notation for the below discussion, for an odd prime number q , the ring Z/qZ (or Z_q) with the interval $(-q/2, q/2) \cap Z$ is identified. The notation $[x]_q$ denotes reducing x modulo q . The examples discussed herein use polynomial rings defined by the quotient polynomials $R = Z[X]/\Phi_m(X)$, where $\Phi_m(X) = x_n + 1$ is the irreducible m^{th} cyclotomic polynomial, in which n is a power of 2 and $m = 2n$. Let $R_q = R/qR$. Any type of multiplication including matrix and polynomial multiplication is denoted herein by the multiplication operator $\cdot$. Rounding up to the nearest integer is denoted by $|a|$. Matrices of rings are defined as $A_{M \times N}$, where $A_{ij} \in R_q$ and M, N are the matrix dimensions. $I_{\ell \times \ell}$ represents the identity matrix of rings. Row vectors are represented as $[a \ b]$, where a and b are the vector elements. Column vectors on the other hand are represented as $[a ; b]$.

[0038] The parameters of the cryptographic scheme are n , the degree of the number field; q , the modulus; σ_k and σ_c , the standard deviation of the discrete Gaussian error distribution in the key space and ciphertext space, respectively; $\ell \triangleq \lceil \log q \rceil$ that governs the number of ring elements in a ciphertext. The setting of these parameters depends on the security level λ (e.g., $\lambda = 80$ or 128 bits) as well as the complexity of functions contemplated to evaluate on ciphertexts.

[0039] A bit decompose function $\text{BD}(\text{integer})$ takes an ℓ -bit input integer, then outputs a row vector with size ℓ containing the bit decomposition of this integer. (Note that the letter “ ℓ ” referred to herein, including the figures, is a lowercase script letter “L”.) Similarly, $\text{BD}(\text{polynomial})$ takes an input polynomial of size n , where each coefficient is an ℓ -bit integer, then outputs an ℓ -sized row vector of polynomials (each of size n) containing the bit decomposition of each coefficient of the input polynomial, yielding a matrix of size $\ell \times n$. Finally, $\text{BD}(\text{Matrix of polynomials})$ takes an input matrix of polynomials of size $x \times y$ (each

polynomial is of size n with integer coefficients), then outputs a matrix of polynomials expanded by a factor ℓ in the column dimension, yielding a matrix of size $x \times y\ell$, where each consecutive ℓ elements along the row contain the bit representation of each coefficient of each of the input polynomials. For example, the bit decompose of the input polynomial matrix $B_{x \times y \times n}$ is $BD(B_{x \times y \times n}) = \beta_{x \times y\ell \times n}$. It is noted that despite the fact that the polynomial coefficients of matrix $\beta_{x \times y\ell \times n}$ are single bit values, the storage requirement of matrix β in CPU or GPU memory is not equal to $x \times y\ell \times n$ bits. This is due to the fact that the smallest addressable unit of memory is a byte (i.e., Byte Addressable). Hence, β requires $x \times y\ell \times n$ bytes of storage. This results in the further observation that the storage requirement of $\beta_{x \times y\ell \times n}$ is at least 8 times the storage requirement of $B_{x \times y \times n}$.

[0040] A bit decompose Inverse function $BDI()$ is the inverse of the bit decompose function $BD()$. The $BDI()$ function groups consecutive ℓ coefficients along a row (the coefficients don't need to be binary), and outputs the integer corresponding to those ℓ bits. Mathematically, the $BDI()$ function can be defined as multiplying the expanded matrix of polynomials $\beta_{x \times y\ell}$ from the right by the matrix $\alpha_{y\ell \times y}$ defined in the equation of FIG. 6A. (The polynomial dimension n is omitted from this point forward for clarity). Hence $B_{x \times y} = BDI(\beta_{x \times y\ell}) = \beta_{x \times y\ell} \cdot \alpha_{y\ell \times y}$.

[0041] Ring Learning With Errors (RLWE) based encryption scheme.

[0042] The method and system may employ a RLWE (Ring Learning With Errors) cryptographic scheme, and the general principles of such scheme will now be described. However, this scheme is not particularly limiting and other suitable polynomial-based fully homomorphic cryptographic schemes may be used. Moreover, any gaps in the below would be well understood by those skilled in cryptography in view of the known art.

[0043] The system and method is configured to generate keys, encrypt information, and decrypt information.

[0044] The key generator is configured to implement a $\text{Keygen}(1^\lambda)$ function as follows. A polynomial $t \leftarrow D_{R_q, \sigma_k}$ is chosen. The secret key becomes $sk = s_{2 \times 1} \leftarrow [1; -t] \in R_q^2$. The public key is $pk = A_{1 \times 2} = [b \ a]$, based on a uniform sample $a \leftarrow R_q$, $e \leftarrow D_{R_q, \sigma_k}$, set $b = a \cdot t + e$. It is noted that the expression in FIG. 22a holds.

[0045] As shown in FIG. 18, this is advantageous over a known secret key $sk = v = PO2(s)$ based on a powers-of-two expansion such as $PO2(x)$ defined as $[x, 2x, \dots, 2^{\ell-1}x]$. Hence, the key generator generates smaller secret keys by a theoretical factor of ℓ times.

[0046] The encryption engine 62 is configured to implement an $Enc(pk, \mu)$ function as follows. The message space is R_q . A uniform vector $r_{N \times 1}$ is sampled where each coefficient in the polynomials in r sampled from $\{0, 1\}$, $E_{N \times 2} \leftarrow D_{R_q, \sigma_c}^{N \times 2}$. The plaintext polynomial $\mu \in R_q$ is encrypted by calculating the expression in FIG. 22b. As shown in FIG. 19, this is advantageous over prior techniques that use $C_{N \times N}$, as the encryption engine 62 results in smaller ciphertext by a theoretical factor of ℓ times.

[0047] The decryption engine 104 is configured to implement a $Dec(sk, C)$ function as follows. Given the ciphertext C , the plaintext $\mu \in R_q$ is restored by multiplying C by the secret-key s according to the expression in FIG. 22c.

[0048] This is advantageous over prior techniques that implement $Dec(sk, C) = C_{N \times N} \cdot v_{N \times 1}$, as the decryption engine 104 requires the performance of fewer operations a theoretical factor of ℓ times.

[0049] It is noted that the first ℓ coefficients in the first term of the expression in FIG. 22c are in the form $\mu, 2\mu, \dots, 2^{\ell-1}\mu$. This means that the element at location $i \in [0, \ell - 1]$ is in the form $\mu \cdot 2^i + \text{error}$. That is, the most significant bit of each entry carries a single bit from the number μ assuming that $\text{error} < q/2$ and there is theoretically no wrap-around $\text{mod } q$ as may be found in prior techniques.

[0050] It is now possible to perform operations on ciphertext without first decrypting the ciphertext. For input ciphertexts $C_{N \times 2}$ and $D_{N \times 2} \in R_q^{N \times 2}$ encrypting μ_1 and μ_2 respectively, homomorphic operations are implemented as follows.

[0051] The addition operator implements an $ADD(C, D)$ function to add two ciphertexts $C_{N \times 2}$ and $D_{N \times 2}$ by performing the entry-wise addition $C_{N \times 2} + D_{N \times 2}$.

[0052] The multiplication operator and bitwise decomposition function implement a $MULT(C, D)$ function to multiply two ciphertexts $C_{N \times 2}$ and $D_{N \times 2}$ by performing the bitwise decomposition function (or BD) on one ciphertext and then executing the multiplication, as $BD(C_{N \times 2}) \cdot D_{N \times 2}$.

[0053] As shown in FIG. 19, this is advantageous over prior techniques that define $MULT(C, D) = FLATTEN(C_{N \times N} \cdot D_{N \times N})$, where $FLATTEN(A)$ is defined as $BD(BDI(A))$. The present technique requires fewer operations by a theoretical factor of at least ℓ times.

[0054] Correctness of the above homomorphic addition should readily apparent to those skilled in the art. The multiplication is asymmetric in the input ciphertexts C and D . That is, the components of D are treated as a whole, whereas the components of C are broken up into their bit-wise decompositions. The multiplication is correct, as discussed below, and gives a slow noise-growth rate.

[0055] The correctness of the multiplication operation should readily apparent to those skilled in the art in view of the expression in FIG. 22d, in which matrix dimensions are removed for clarity. In the last line of the manipulation of expression in FIG. 22d, it is apparent that the encryption of $\mu = \mu_2 \cdot \mu_1$.

[0056] Correct decryption depends on the ciphertext noise being bounded. Thus, it is important to understand how homomorphic operations increase ciphertext noise. Taking C as a fresh ciphertext, it is apparent that homomorphic addition of v ciphertexts increases the noise by a factor of v in the worst case. In various contemplated implementations, since the coefficients of the error polynomials are contemplated to follow a Gaussian distribution, the factor is closer to $O(\sqrt{v})$.

[0057] It is further apparent that homomorphic multiplication of two ciphertexts $C = Enc(\mu_1)$ and $D = Enc(\mu_2)$ with error magnitudes B_1 and B_2 , respectively, increases the error to $O(B_1 \cdot \|\mu_2\|_1 + B_2 \cdot n \log q)$ in the worst case, and $O(B_1 \cdot \|\mu_2\|_1 + B_2 \cdot \sqrt{n \log(q)})$ in various contemplated implementations. Here, $\|\mu\|_1$ denotes the ℓ_1 norm of the message polynomial μ . It is advantageous that error dependence on the two ciphertexts is asymmetric, as evident from the above.

[0058] To multiply v ciphertexts the order of multiplication is contemplated to play a role in error. In techniques described herein, input μ will typically be 0 or 1, meaning that the growth is simply additive with respect to B_1 . Thus, it is advantageous to multiply v ciphertexts with (the same) error level B through an accumulator-like function as shown in FIG. 20, rather than using a binary tree of multiplications, which tends to grow error at superpolynomial rates. The resulting error growth is $O(B \cdot v n \log(q))$ in the worst case, and $O(B \cdot \sqrt{v n \log(q)})$ in various contemplated implementations. Hence, the control logic may be configured to implement accumulative multiplications, as shown in FIG. 20 and as may be required by various contemplated implementations.

[0059] For example, reference is now made to the expression in FIG. 22e, in which $x_1, \dots, x_v$ are v -tuples of input encrypted bits, $y_1, \dots, y_v$ are v -tuples of bits in some set S , and operation $(x_i \oplus y_i)$ represents binary XNOR between bits x_i and y_i . Since the form of the expression in FIG. 25e stipulates that exactly one of the terms may survive ($F = 1$ when $x_1, \dots, x_v \in S$, otherwise $F = 0$), the small total error growth can result, even though the component computing based on the expression in FIG. 22e may not be able to determine precisely which term will survive.

[0060] It is apparent that noise grows to $O(B \cdot v n \log q \cdot |S|)$ in the worst case, or $O(B \cdot \sqrt{v n \log(q) |S|})$ in various contemplated implementations. This is in contrast to $O(B \cdot \sqrt{(n \log(q))^{\log(v)} |S|})$ when using the known Brakerski-Gentry-Vaikuntanathan encryption scheme, implemented in IBM HElib. Indeed, such expressions, as in the expression in FIG. 22e, are far from atypical, and they occur quite naturally in evaluating decision trees and PIR-like functions as will be discussed further below.

[0061] Another source of improvement afforded by the presently disclosed techniques is evident from the error term $B_1 \cdot \|\mu_2\|_1 + B_2 \cdot n \log q$. When multiplication is performed using an accumulator, as shown in FIG. 20, B_2 represents the smaller error in the fresh ciphertexts C_i , and B_1 represents the larger error in the accumulated ciphertext C_{accum} . If C_i encrypts $\mu_2 = 0$, then the larger error term B_1 disappears from the error expression.

[0062] This error reduction is also apparent from the expression in FIG. 22e. When evaluating each of the products in the expression in FIG. 22e, the error can be seen to grow proportional not to v , the total number of multiplications, but rather with k , the longest continuous chain of 1's starting from the end. It is contemplated that this is because the last time a zero is encountered in the multiplication chain, the error is reduced, by the observation above. Assuming that S is an expected set, the expected length of a continuous chain of trailing 1's is $\sum_{i=1}^v i \cdot 2^{-i} < 2$. In other words, the multiplicative factor of v disappears from the error expression as well, and error growth becomes close to $O(B \cdot \sqrt{n \log(q) |S|})$. This is substantially the same effect as if $|S|$ ciphertexts were added.

[0063] Further, when f is taken as a function to be evaluated, for example, the expression in FIG. 22e, the error $_{(B,n,q)}$ denotes how much the error grows when evaluating the function f on ciphertexts in R_q with an initial error of magnitude B . For correct decryption, it is expected that the expression in FIG. 22f holds. Since errors tend to grow slower using the present techniques, q can be set to be correspondingly smaller to meet a security level equivalent to that of prior techniques. Following the analysis of Lindner and Peikert, for a security level of λ bits, it is expected that the following expression holds:

$$n > \log q (\lambda + 110) / 7.2 \quad (7)$$

[0064] Because $\log q$ in the present techniques is smaller, n can be set to be smaller, for the same security level λ . In turn, a smaller n can result in a error $_{(B,n,q)}$ that is smaller, leading to an even smaller q , and so on. Suitable parameters are obtained by solving both the above inequalities in FIGs. 22f and 22g together. FIG. 21 summarizes an example of such a parameter selection.

[0065] NTRU based FHE scheme

[0066] As an alternative to the RLWE-based FHE scheme, in some examples of the invention, an NTRU variant of the encryption scheme is used to reduce computational complexity and to speedup operations, as will be detailed below. The encryption system works as follows.

[0067] A key generation function, $\text{Keygen}(1^\lambda)$, requires the choosing of two polynomials $f_{1 \times 1}, g_{(1 \times 1)} \leftarrow D_{R_q, \sigma_k}$ such that (a) f is invertible in the ring R_q ; and (b) $f \equiv 1 \pmod{2}$. This can be done by sampling the polynomial f from the distribution D_{R_q, σ_k} until it satisfies conditions (a) and (b).

[0068] The public key pk and the private (secret) key sk can be computed from the equation shown in FIG. 6B.

[0069] For the encryption function, $\text{Enc}(pk, m)$, the message space is R_q . A plain text polynomial $\mu \in R_q$ is encrypted by evaluating the equation of FIG. 6C, where $S_{\ell \times 1}, E_{\ell \times 1} \leftarrow D_{R_q, \sigma_e}^{\ell \times 1}$ are sampled from a discrete Gaussian distribution with standard deviation σ_e . This encryption function can be implemented at the encryption engine 62.

[0070] Concerning the decryption function, $\text{Dec}(sk, C)$, for a given the ciphertext C , the plaintext $\mu \in R_q$ is restored by multiplying C by the secret-key f using the equation of FIG. 6D. This decryption function can be implemented at the decryption engine 104.

[0071] With reference to FIG. 6D, the ℓ polynomials in the first term of the last line are in the form $\mu f, 2\mu f, \dots, 2^{\ell-1}\mu f$. This means that the element at location $i \in [0, \ell - 1]$ is in the form $\mu f \cdot 2^i + \text{error}$. That is, the most significant bit of each entry carries a single bit from each coefficient in μf assuming that $\text{error} < q/2$ and there is no wrap-around mod q as is found in the prior art. As such, “ μf ” can be fully recovered from C which can then be multiplied by f^{-1} to recover μ .

[0072] In addition to the ability to recover the message polynomial μ , the ℓ ring elements in the ciphertext facilitate and manage noise growth in homomorphic operations (in particular, homomorphic multiplication) as described below.

[0073] Regarding homomorphic operations, for input ciphertexts $C_{\ell \times 1}$ and $D_{\ell \times 1} \in R_q^{\ell \times 1}$ encrypting μ_1 and μ_2 respectively, homomorphic operations are defined as follows. For an addition operation, $\text{ADD}(C, D)$, to add two ciphertexts $C_{\ell \times 1}$ and $D_{\ell \times 1}$, the output is $C_{\ell \times 1} + D_{\ell \times 1}$, which is an entry-wise addition. For a multiplication operation, $\text{MULT}(C, D)$, to multiply two

ciphertexts $C_{\ell \times 1}$ and $D_{\ell \times 1}$, output is $BD(C_{\ell \times 1}) \cdot D_{\ell \times 1}$. The addition and multiplication operations can be implemented at the calculation engine 88.

[0074] The correctness of the above homomorphic addition and homomorphic multiplication should be apparent to those skilled in the art in view of this disclosure. It is clear that the multiplication operation is asymmetric in the input ciphertexts C and D. That is, the components of D are treated as a whole, whereas the components of C are broken up into their “bit-wise decompositions”. It is shown below that this multiplication method is correct and gives a slow noise-growth rate.

[0075] The correctness of the multiplication operation is evident from the decryption operation shown in FIG. 6E, in which matrix dimensions are removed for clarity. Note that the last line shown in FIG. 6E is the encryption of $\mu = \mu_2 \cdot \mu_1 \cdot f$. Note also that $BD(C_{\ell \times 1}) \cdot BDI(I_{\ell \times \ell}) = I_{\ell \times \ell} \cdot C_{\ell \times 1} = C_{\ell \times 1}$.

[0076] Concerning noise analysis, taking C as a fresh ciphertext, the following observations can be made in view of the operation shown in FIG. 6E. Homomorphic addition of v ciphertexts increases the noise by a factor of v, in the worst case. In some implementations, since the coefficients of the error polynomials follow a Gaussian distribution, the factor is closer to $O(\sqrt{v})$. In addition, homomorphic multiplication of two ciphertexts $C = \text{Enc}(\mu_1)$ and $D = \text{Enc}(\mu_2)$ with error magnitudes B1 and B2, respectively, increases the error to $O(B_1 \cdot \|\mu_2\|_1 + B_2 n \log(q))$ in the worst case, and $O(B_1 \cdot \|\mu_2\|_1 + B_2 \sqrt{n \log(q)})$ in various contemplated applications. Here, $\|\mu_2\|_1$ denotes the l_1 norm of the message polynomial μ . As can be seen, error dependence on the two ciphertexts is asymmetric.

[0077] Regarding the setting of parameters, taking f to be a function that is being evaluated and that computes the multiplication of v ciphertexts, $error_f(B, n, q)$ denotes how much the error grows when evaluating a function f on ciphertexts in R_q with an initial error of magnitude B. For correct decryption, the equation of FIG. 6F should be satisfied.

[0078] Since error in accordance with the present invention may grow slower than in some known schemes and since there may be no need for a chain of moduli to control the noise

growth, q can be set to be correspondingly smaller for the same security level afforded by such known schemes. For a security level of λ bits, the equation of FIG. 6G should be satisfied.

[0079] Because $\log q$ according to the present invention may be smaller relative to known schemes, n may be set smaller for the same security level λ . In turn, with a smaller n , the new $error_f(B, n, q)$ is smaller, leading to an even smaller q , and so on. Optimal parameters can be obtained by solving the above inequalities of FIGs. 6F and 6G together. The table of FIG. 7 summarizes an example parameter selection.

[0080] In addition, the encryption scheme according to the present embodiments does not use the flatten operation introduced by the GSW scheme. A flattened ciphertext (Ctxt) takes up a large memory space. It also needs considerable computation time to combine entries and decompose them back into bits (or even decompose them into groups of “ m ” bits for a higher radix Bit Decompose operator). Further, since the encryption scheme according to the present invention does not use a flatten operation, it does not have single bits representing ciphertexts. Rather, ciphertexts are represented as packed numbers mod q . To multiply ciphertexts, it is advantageous to use the fast NTT algorithm to speed up the ciphertext multiplication operation, as opposed to using regular polynomial circular convolution. The encryption scheme according to the present embodiment decrypts the most significant bit from all l polynomials in the ciphertext, as opposed to decrypting only a single bit from a single polynomial in the ciphertext. In this way l bits, one from each polynomial, can be decrypted. These bits are combined back into the encrypted polynomial using the formula shown in FIG. 6W.

[0081] Types of encrypted confidential data that can be collected and homomorphic calculations that can be performed thereon are now discussed. Specific data/calculations discussed below include analysis of medical data, such as encrypted genomic data, predictive analysis of disease, and evaluation of relational operations. Other data included herein is financial transaction data, particularly credit card data. Although some of the examples discussed pertain to specific data, it is noted that some computations can be used for other types of data (e.g., relational operations). Further, it is noted that the examples discussed below are not limiting and other examples within the scope of the present invention are contemplated.

[0082] Financial Transactions

[0083] Another area of application for an embodiment of the homomorphic encryption system is financial transactions, particularly credit card transactions. Attacks on credit cards have escalated tremendously in recent years, with major breaches resulting in millions of client record being exposed. Once exposed, the client information may be resold (e.g. on the dark web), used for fraudulent transactions, particularly card not present (CNP) transactions, or used for direct attacks on Point-of-Sale (POS) systems.

[0084] These attacks happen because credit card information, either in the databases or for the client, are present in a plain text format at some point in the credit card authorization process. According to an embodiment of the present invention, the system may encrypt the credit card information from the point at which it is acquired and never decrypt the ciphertext even at the authentication step.

[0085] Payment Processing

[0086] Major credit card companies, such as VISA™ and MasterCard™, structure bank card transactions in what is called the four-party model. The parties in this model are the cardholder 1500, the merchant 1510 (the service provider), the payment processor 1520 (the acquirer), and the card issuer 1540. In addition, there is potentially a fifth party which is the credit card company (payment brand) 1530 all as shown in Figure 15.

[0087] The transaction starts when the cardholder 1500, who wishes to purchase something, uses his credit card information on an online merchant or presents his credit card 1505 to the merchant 1510 at the store. The merchant 1510, online or using the point of sale system, acquires and encrypts the credit card information and sends it 1515 to the payment processor 1520 over the network. The encryption at the merchant may be done at the POS terminal or may be done using an application installed on the merchant's systems using the public encryption key. The payment processor 1520 then decrypts the credit card information and forwards it 1525 to the card issuer 1530 for authorization. The merchant 1510 charges the credit card and provides the service or product to the cardholder 1500 once the credit card authorization is received. The payment processor 1520 reimburses the merchant for the service, after that, the card issuer 1530 pays back the payment processor 1520 within 24 or 48 hours. A tokenized system may be utilized between the payment processor and the merchant to eliminate the need for the merchant

to store the cardholder information. The merchant stores only a token corresponding to either the cardholder account or to the individual transactions.

[0088] Information, such as cardholder and transaction data, as well as internal fees, such as network and interchange fees, may be passed 1545 between the payment processor 1520 and credit card company 1540, or between 1555 the card issuer 1530 and credit card company 1540, or both, either as required during the transaction, or after the transaction.

[0089] There are multiple points of vulnerability in this credit card system that may allow attackers to steal credit card information.

[0090] Point of Sale System (POS): When the cardholder presents his card at the merchant store, his/her card is swiped in a credit card machine or a card reader that transfers this information to a computer. There may be some points in the system where this card information is not encrypted for some time before sending it to the payment processor. At this point, malware installed in the system can gather this information and send it back to the attacker. To address this vulnerability, credit card information must be encrypted the moment it is read using the card reader. This way, malware will not be able to gather card information using this method. Currently, point to point encryption (P2PE) is used to securely transmit credit card information from the POS system to the payment processor, which then decrypts it to send to the card issuer for verification. P2PE uses 3DES or AES for encryption which may be considered safe for the time being but may not be secure against quantum computers in the future.

[0091] Payment Processor: When the payment processor receives the encrypted credit card information from the merchant, it decrypts it and sends it in plaintext to the card issuer for authentication. This is a clear point of weakness. Any malware installed in the payment processor system, or any attacker who broke the secure channel between the payment processor and the card issuer, can gather the credit card information while being in plaintext form. To address this vulnerability, credit card information should never be decrypted at any point. Additionally, in the case of a tokenized system, the presence of the secure vault which translates the transaction into a token and vice versa. This vault stores all the cardholder information and their corresponding tokens. If this vault is hacked, valuable information will be at risk.

[0092] The card issuer has two points of vulnerability: 1) it receives the client credit card information in plaintext. This exposes it to the possibility of attack; and 2) the card issuer database of credit card information for all its clients sits in a secure server in plaintext to compare it to the incoming card information. This is the largest security threat on credit card information. This is because an attack on this secure database will result in the loss of all credit card information.

[0093] These vulnerabilities may be solved if all the credit card information in the database and also the incoming card information in plaintext were encrypted and were compared while in ciphertext.

[0094] According to an embodiment as shown in Figure 16, there is presented an embodiment of the present invention which is a change in the credit card authorization system that may provide a much higher level of security than current systems. The embodied system may be based on encrypting the credit card information at the POS system using quantum secure Fully Homomorphic Encryption (FHE) which is based on lattice based cryptography. The need for FHE is based on the need for both homomorphic addition and multiplication operations on encrypted data. If only one type of operation is required, somewhat homomorphic encryption may be used. Once the card information is encrypted at the online website or at the POS, there may not be a need to decrypt at any point afterwards. The encrypted credit card information may be compared against credit card database data (e.g. sets of cardholder data for various cardholders held in a database) at the card issuer which may also be encrypted using the same FHE encryption.

[0095] Thus, the system may eliminate any possible attacks on credit card information, either in the network, or in the database servers because they may all be encrypted using a quantum secure FHE encryption scheme.

[0096] The cardholder data from the consumer 1610 (the primary account number (PAN)) is captured by the online transaction data entry (or at the POS), the data is encrypted using a public-key Fully Homomorphic Encryption (FHE) scheme. The credit card number (except for the first digit and the last 4 digits), month, year, and CVV numbers may be encrypted using the FHE public encryption key published by the credit card company. The name of the cardholder and the

first digit and the last 4 digits of the credit card number may be encrypted using the regular encryption schemes. The merchant 1620 then sends the encrypted PAN to the acquirer/payment processor 1630.

[0097] The payment processor 1630 forwards the encrypted PAN to the appropriate payment brand 1640 using the credit card first digit (e.g., Visa™, MasterCard™, American Express™, etc.). The payment brand 1640 then forwards the encrypted PAN to the card issuer 1650 (issuing bank) using the next five digits (BIN: Bank Identification Number).

[0098] The card issuer 1650 may use the cardholder name and last 4 credit card digits to narrow down the credit card entries in the database that match this information. Each entry in the narrowed down list may be enumerated, compared against the encrypted PAN using FHE algorithms, and the encrypted matching indicator will be multiplied by the number corresponding to each entry in the list. The encrypted result from each entry may all be added together to get a final result. The final result may encrypt the number of the matching account, the numbers corresponding to all other accounts will be multiplied by '0' (due to mismatch), which will nullify the result. The encrypted result is sent back to the payment brand 1640, which decrypts it and sends the final result, which is a single digit corresponding to the matching account, back to the card issuer 1650 to verify that the card is not reported lost or stolen, and that the account has the appropriate amount of credit/funds available to pay for the transaction. The card entries may be ordered by most frequently used and the rest of the entries may be ignored once a hit is found to reduce the verification time.

[0099] If approved, the issuer 1650 generates an authorization number and routes this number back along with a card-specific FHE encrypted PAN to the payment brand 1640. The payment brand 1640 then forwards the authorization code and the encrypted PAN back to the acquirer/processor 1630.

[00100] The acquirer/processor 1630 accesses a secure vault 1660 to retrieve/generate a token corresponding to the encrypted PAN. Note that the secure vault 1660 may also be FHE encrypted except for the tokens.

[00101] The acquirer/processor 1630 returns the token back to the merchant 1620. The merchant may retain the token long term in a merchant database 1670 for the processing of returns, retrieval requests or chargebacks, as well as for business intelligence reasons such as analysis of consumer buying behavior and creation of marketing programs.

[00102] This embodiment is an example of a centralized system where only one entity, the credit card company 1640, owns the secret decryption key and distributes public encryption keys. Another embodiment is a decentralized system where each card issuer 1650 holds its own secret decryption key and that the POS systems recognize which card issuer the credit card belongs to and encrypts the card's information using the appropriate public encryption key corresponding to this card issuer. This decentralized system may need another entity which is solely responsible for decrypting the encrypted comparison result. This may be required to decouple the card issuer servers holding the encrypted credit card information from the servers which hold the valuable secret decryption key.

[00103] Another embodiment is the use of FHE multiparty computation. Different secret keys will be generated for each user and the public key will be stored in the credit card itself. The cardholder data will be encrypted by the user's public key and may be sent to the system for matching against other accounts. Multiple secret decryption keys will be required to decrypt the final result. Additionally, with multiple keys, each party may be limited to decryption only of the information associated with their key, and not be capable of decrypting any other information, improving overall security of the data.

[00104] The system may be applied on multiple different levels: the first version is a full system where information is encrypted from the origin and never decrypted at any point in the system.

[00105] The second version is a partial system where, in order to accommodate the current payment systems, and to reduce the risk of data attacks, at the payment processor, after card information decryption, card information can be re-encrypted using FHE before sending to the card issuer. This may reduce the possibility of attacks but there is a brief moment in time where data is in the clear between information decryption and re-encryption where data can reside in

the device memory vulnerable to attacks. However, the credit card database at the card issuer may always remain encrypted using FHE.

[00106] The system also supports online transactions using online payment systems using credit cards or PayPal or other online transaction services where payment information may be encrypted on the client PC prior sending over the Internet.

[00107] According to an embodiment, credit cards consist of the following key information: Bank name (Front); Credit card number of 13-19 digits (Front); Credit card expiry date (Front); Cardholder name (Front); and Card Verification Value (CVV) (Back). The credit card number consists of a leading 6-digit Bank Identifier Number (BIN), also known as an Issuer Identifier Number (IIN), and a 6- to 12-digit account number, and a single digit checksum number. The terminating 3 digits of the account number and the checksum number are encrypted with the same FHE scheme but using a different key which may be given to specific parties for decryption. Other cards based on a BIN or IIN, such as debit cards, reward cards, and merchant-specific cards may also be used and have an equivalent numbering system.

[00108] In an embodiment, the following fields may be encrypted using quantum secure lattice-based Fully Homomorphic Encryption (FHE): the middle 3-9 digits of the credit card number (e.g. the secure portion of the account number), the expiry date and the CVV number. These fields may never be decrypted at any point in the verification process. They may always be in the ciphertext (Ctxt) form.

[00109] The remaining items in the credit card may be encrypted using the same encryption scheme but with one or more different public/secret key pairs. A proper key management system is crucial for the overall security of the system. The other keys may be used by intermediate parties to decrypt the needed fields. Though the remaining fields may be encrypted using an FHE scheme, no homomorphic operations may be applied to them. Every character of the remaining fields may be encoded in adjacent coefficients in the same polynomial. The remaining fields are: bank name, cardholder name, and the first 6 digits (BIN number) and last 4 digits of the credit card number.

[00110] These remaining fields may be in plaintext (Ptxt) form at some points in the verification process. They help identify the credit card company (first digit) and well as narrow down the credit card entries in the card issuer credit card database (last four digits + Cardholder name). This choice narrows down the probable matching cards to just a few. Additionally, by partitioning the fields, the system may accommodate future changes to the credit card number system, such as the 8-digit and alphanumerical BIN proposals developed by the International Organization for Standardization (ISO).

[00111] Additionally, further fields may be included in the transaction request and handled without disruption as, since the data is not decrypted, the actual source and content of the data is not significant, only the ability to validate and authenticate it. Thus, developments such as biometric (e.g. fingerprint) identification, rotating or variable CVV numbers, and one-time card numbers may all be validated and authentication via the present method and system, and, in some cases, without any substantive changes.

[00112] As mentioned earlier, some fields in the credit card information may never be decrypted at any time in the verification process. Performing the credit card authentication using these encrypted items uses the properties of the FHE encryption scheme.

[00113] Fully homomorphic encryption permits addition and multiplication operations on encrypted data to generate an encrypted result, which, when decrypted, gives the correct result if the same set of operations were applied on unencrypted data.

[00114] To perform blind matching between two single bits x and y , it is possible to apply the XNOR binary operation on these two bits $z = \overline{x \oplus y}$. To implement the XNOR operator homomorphically, it needs to be broken down into addition and multiplication operations, specifically, $z = \overline{x \oplus y} = xy + \bar{x}\bar{y}$.

[00115] To match multiple-bit inputs, $x = x_1x_2 \dots x_n$ and $y = y_1y_2 \dots y_n$, this can simply be done by matching the corresponding, same-order, individual bits, and multiplying the result of the individual bit matching together $z = z_1 \times z_2 \times \dots \times z_n$. Where the final result z is a single bit indicating if x is equal to y or not.

[00116] Implementing the XNOR operation using $\overline{x \oplus y} = xy + \bar{x}\bar{y}$ is used for other homomorphic encryption schemes which support multiplying multiple accumulator Ctxts (Ctxts that encrypts the result of previous Ctxt multiplications). This multiple accumulator multiplication is not suitable for the embodied encryption scheme since the parameter values are set to be as small as possible, while keeping the security level constant, that it supports multiplying an accumulator only with either fresh Ctxts, or Ctxts that are results from Ctxt addition/subtraction operations. This may reduce the memory and network usage of the Ctxts, and may also speedup Ctxt operations.

[00117] To address this, the XNOR operation can be re-written to make it suitable for the embodied encryption scheme. The XNOR is reformulated to be $z = \overline{x \oplus y} = (1 - x - y)(1 - x - y)$ and applying this formula on x and y will result in $z = 1$ only when $x = y$. This addresses the system requirements, since for multi-bit inputs x and y , the matching bit z equals $z = (1 - x_1 - y_1)(1 - x_1 - y_1)(1 - x_2 - y_2)(1 - x_2 - y_2) \dots (1 - x_n - y_n)(1 - x_n - y_n)$. Now each individual bracket consists only of Ctxt addition/subtraction operation which is implementable using the embodied encryption scheme.

[00118] Secure relational operation where $z = 1$ if $x > y$, and $z = 0$ otherwise is also possible using the FHE encryption scheme using the secure relational operations as discussed above in FIG. 6U and 6V. This can be very useful to also be able to check if the amount needed for the credit card transaction “ y ” exceeds the credit limit for this account “ x ” (or exceeds the account limit in case of a debit card).

[00119] According to an embodiment, in performing the secure credit card authentication, the middle digits (for this example, 6 digits for a 16-digit card number are assumed) of the credit card number are first converted into bits where each bit is then encrypted using FHE encryption. The same process is performed for the expiry date and the CVV number. These encrypted bits are then sent on the network along with the cardholder name, bank name, BIN number, and last 4 digits of the credit card to the payment processor which then forwards them to the appropriate card issuer. The card issuer will then receive this information, and using the cardholder name and last 4 digits of the card number, it will narrow down the search to just a few possible matching accounts (due to the match in the name and the last 4 digits) and the corresponding encrypted

information will be fetched. Secure authentication is then applied between the incoming encrypted information and the possible matching accounts. The authentication operation done on the non-matching accounts will result in an encrypted indicator which is $z=0$. The matching account, if all the bits of the credit card number, expiry date, and CVV numbers match (and possibly BIN number if used), and if the requested transaction amount is less than the account balance or limit, an encrypted indicator of $z=1$ will be generated. The encrypted indicator, either '0' or '1' can then be multiplied by a number corresponding to the order of this entry against the reduced list. To decrypt this final result and to get a confirmation for this transaction, this encrypted indicator, which encrypts the order of the matching account in the reduced list, needs to be decrypted using the secret decryption key available at the credit card company (or on a separate secure server owned by the card issuer in the case of decentralized scheme). After decryption, only a single digit is returned which corresponds to the order of the matching account in the reduced list.

[00120] In another embodiment, the sensitive bits for the input card and for each of the matching accounts are partitioned into "k" parts, where "k" corresponds to the number of processors (e.g. GPUs) available in each machine and each part is sent to a different GPU. Each GPU may be enumerated by a number ranging from 0 to k-1. The result from each GPU may be rotated by a number of polynomial slots corresponding to its GPU number by multiplying by a polynomial with a 1 shifted to the corresponding location. When the results are available from each GPU, all the results are added. This will produce a result where k slots of the encrypted polynomial each has a bit corresponding to a match or non-match for the corresponding bits. After decryption, if all the bits are non-zero, this means that this account is a match. When a matching account is found, it may be put at the top of the list in the database to make matching time faster in the next times, also the remaining accounts may be ignored.

[00121] The quantum-secure, lattice-based FHE scheme used for this system may be computationally intensive. To speed up the homomorphic operations, graphical processing units (GPUs) are used alongside the CPUs in the system. Each GPU is capable of supporting a number of transactions per second. To achieve high transactions per second required by large transaction companies like VISA™ and MasterCard™ (each may require up to 4000 transactions per

second), many GPU cards are required. To realize such a performance throughput, a multi-server environment may be implemented that can be scaled to serve any level of demand.

[00122] Furthermore, while the above embodiment presents a simple matching of the credit card in the transaction request against a list of account names and numbers to determine the presence of a valid account, other lists and types of validation may also take place. For example, there may be a separate list of stolen/lost cards and cardholder data against which the transaction request may be compared as well. In that case, the card number may be valid, but the authentication denied, if the card also appears on the stolen/lost list. A similar process may also be used for other validation and authentication lists, such as blacklisted countries for transaction, while remaining compliant with laws regarding disclosure of such information.

[00123] This may also be directed to the benefit of the user and/or merchant as well, such as by matching data in the request to a merchant rewards program, either by extracting the data from the request, or by matching it in a list. This reduces the need for multiple transactions and separate processing, potentially leading to greater transaction efficiency. Further extensions and variations may only be limited by data bandwidth and processing time requirements.

[00124] Effectively, any encrypted data may be validated and authenticated against any set of encrypted data without the need for decryption. Changes in the data formats, data structure, or even the data itself may be more easily accommodated, since decryption information need not be shared.

[00125] Homomorphic encryption does not natively provide data integrity. Hackers may be able to tamper with the encrypted data by applying their mathematical operations on the data while at rest or in motion. This may affect the result of the manipulated data and may change the result to something that may aid the hacker in an attack. For example, applying a homomorphic OR operation on the final encrypted flag with an encrypted “1” which will make the result always a “1” or “Transaction Granted”. The protection of data integrity from the modification of unauthorized parties may be provided by applying extra mathematical machinery to provide a tamper resistant credit card authentication system. As an example, without limitation, a Hash-based Message Authentication Code (HMAC) can be appended to the encrypted message to provide the required authentication of the message integrity.

[00126] In addition, as an extra security protection layer to the credit card transaction, the system may support encrypting a credit card CVV number that is periodically changing over time. The system may homomorphically encrypt the CVV number generator key and store it in the bank database. When a new customer credit card is to be verified and purchase authorized, mathematical operations are applied homomorphically on the CVV generator key to generate the new valid CVV number to be matched against the customer credit card to be verified.

[00127] For example, other forms of static or quasi-static private identification data which required both confidential/encrypted treatment and validation and authentication for a request, may be used within this system and method. Not only bank debit cards (similar data structure to credit cards), but other private identification which is typically found in a card or similar format may be used. Such as medical ID or insurance cards, which may contain a name, a group or plan number, and an individual identifier, along with the name of the insurance provider from which to retrieve the data for validation. Or passports, which may contain a number, passport number, date of issue and expiry, citizenship, and the country of issue, for validation. As with credit cards, the data deemed confidential and the data which may be shared as cleartext depends on the decisions of the governing body responsible for the identification data, e.g. the government in the case of a passport.

[00128] As a further example, this system and method may also be used in the field of authentication such as in Authentication as a Service (AaaS) software which operates as a cloud-based identity management service for securing access to data and applications in different locations which may be accessed from anywhere in the world by potentially a number of users using a variety of devices. Beyond identification and authentication, the present system and method may also be used to search government-maintained no-fly lists or other forms of watch lists or registries that contain confidential and/or sensitive personal information to, for example, confirm an individual's presence or absence on the list without revealing other information to the operator.

[00129] The full system according to an embodiment is shown in Figure 17. The authentication process may take place when the payment processor 1710 sends the credit card information to the card issuer. The card issuer may have a front-end server 1720 that accepts

incoming requests. This front-end server 1720 may be connected to a network of GPU machine farms 1730 that send frequent updates to the front-end server 1720 about its occupancy (if it is free or not). When the front-end server 1720 receives a request, it may check its server table for a free machine 1740, and fetch its corresponding IP address. This IP address is then sent back to the payment processor 1710. The payment processor 1710 then connects to this free computing node 1740 and sends the cardholder card information to it. When the computing node 1740 gets that information, it may fetch from the database the data of the credit cards which match the cardholder name and the last four digits. The authentication process may then take place on that GPU on this specific machine 1740. The result of the authentication operation may then be sent to the credit card company 1750 for decryption. The decrypted authentication digit may then be sent back to the card issuer 1710 to indicate the final result. This system may be implemented to ease the network connection requirements for the front-end server 1720 and to distribute it on all the computing nodes 1730.

[00130] Medical Data Collection

[00131] FIG. 1 shows a networked computer system embodying various aspects an embodiment of the present invention for use with confidential medical data. A plurality of remote medical devices 20 are connected to a medical data system 22 via a wide-area network 24. The remote medical devices 20 may include devices such as wearable devices 26, patient terminals 28, laboratory terminals 30, and the like. Wearable devices 26 may include devices such as special-purpose medical devices, smart-watches, wearable computers, wearable activity trackers, smart clothing, and similar. Wearable devices 26 may include, but are not limited to, sensors such as accelerometers, heart rate monitors, temperature monitors, respiration monitor, glucose monitors, global positioning system (GPS) receivers, barometric pressure sensors, combinations of such, and similar. Patient terminals 28 and laboratory terminals 30 may include devices such as smartphones, tablet computers, desktop/laptop computers, and similar. Features that mutually distinguish the remote medical devices 20 such as wearable devices 26, patient terminals 28, and laboratory terminals 30 are not contemplated to limit the present invention. Functionality of these devices 26 – 30 may overlap. Wearable devices 26 and patient terminals 28 collect data directly from patients and may do so without the need for manual input by patients. Wearable devices 26 and patient terminals 28 may allow for manual input by a patient

32 (e.g., typing or selecting) of data. Laboratory terminals 30 may require some form of manual input by medical personnel 34 (e.g., clinicians, lab technicians, nurses, physicians, etc.) to enter data related to medical tests (e.g., the results of blood work) and other medical procedures conducted on patients 32.

[00132] The remote medical devices 20 may be configured to collect data concerning patient medical status or health. Such data can be considered to fall into two general categories: medical data and non-medical data. These categories are not limiting and there may be overlap or various types of data may fall into both categories. Medical data may include, but are not limited to, physiological data and/or data that are directly indicative of the medical status or health of the person. Examples of medical data include heart rate, body temperature, respiration rate, and similar. Medical data may be directly measured by remote medical devices 20 (e.g., a heart rate monitor) or may be entered into a device 20 by the patient or a medical professional (e.g., entering a blood pressure measurement into a terminal).

[00133] Non-medical data are data that may be indirectly related to medical outcome/diagnosis and may include, but are not limited to, data related to environmental factors, genetic factors, health behaviours, social and community factors, socio-economic factors, socio-demographic factors, and government or state factors. Environmental factors may be physical environment factors with the potential to influence human health, e.g., water quality, air quality, sound pollution, working conditions. Genetic factors may be related to genetic factors outside those normally influenced by individual behaviours or by the social, economic or physical environment; genetic factors determine predisposition to certain conditions. Health behaviour factors may be factors related to the aspects of personal behaviour and risk factors that epidemiological studies have shown to influence health status, e.g., personal health practices, healthy child development. Social and community factors may be factors related to the measurement of the prevalence of social and community factors, such as social support, life stress, or social capital that epidemiological studies have shown to be related to health, e.g., school readiness, cultural background, social support, social environment, social status, housing affordability and literacy. Socio-economic factors may be factors related to the socioeconomic characteristics of the population that epidemiological studies have shown to be related to health, e.g., employment status, income, social status, level of education. Socio-demographic factors

may be factors related to the socio-demographic characteristics of the population. These indicators relate to or involve a combination of social and demographic factors included: gender, age, level of education, cultural background, employment status, profession, marital status, total number of persons living in the house and living arrangements. Government or state factors may be factors related to government policies, funding, infrastructures and programs, and similar. The skilled person will understand that the above factors are not definitive and some factors may be found in more than one category. Non-medical data may be entered into a device 20 by the patient or a medical professional (e.g., entering a patient's income group into a terminal).

[00134] The wide-area network 24 may include one or a combination of data networks, such as a local-area network, a wireless network, a cellular network, an intranet, a virtual private network (VPN), and the Internet.

[00135] The medical data system 22 may include one or more computers, which may be known as servers, configured with program code that is stored in memory and is executed by one or more processors to perform homomorphic calculations on data collected from the remote medical devices 20, as discussed in detail below.

[00136] The system may further include a key authority system 36 and analyst terminals 38.

[00137] The key authority system 36 stores one or more cryptographic keys, such as one or more private (secret) keys for a fully homomorphic asymmetric cryptographic scheme, such as a ring-learning with errors (RWLE) or NTRU homomorphic encryption scheme. The key authority system 36 includes one or more servers configured to store such private keys and restrict the use of the private keys to authorized users.

[00138] Each private key controlled by the key authority system 36 may correspond to a public key that is distributed to the remote medical devices 20. Data encrypted by the remote medical devices 20 could be decrypted by users with access to the respective private key, although this is not central to the present embodiment. According to the present embodiment, the results of homomorphic calculations performed on encrypted data may be decrypted by users with access to the respective private key. It is contemplated that a large number of remote medical devices 20 may share the same public key and thus form a large and continuous source

of data for homomorphic calculations that do not require the decryption of the data. Rather, the private key may only be needed for the decryption of calculation results.

[00139] Any number of public-private key pairs may be used. It may be advantageous to segment medical devices 20 into different sets, according to device type or other factor, by providing such devices with different public keys. For instance, wearable heart rate monitors may be given one public key while laboratory terminals 30 used for recording the results of blood work may be given a different public key. This may require additional processing at the medical data system 22 when performing calculations on heart rate and blood work data, but may offer the benefit of reducing exposure of patient data should one of the corresponding private keys be compromised. Different sets or types of medical devices 20 may be given different public keys based on other factors, such as the healthcare group/organization, insurer, device manufacturer, etc. Further, for a medical device 20 that collects multiple types of data (e.g., medical and/or non-medical data), each type of data may be assigned to a different public key for encryption by that public key. Again, this may reduce exposure of patient data. For sake of clarity, the examples discussed herein reference a single public-private key pair, but it should be noted that the present invention contemplates various public-private key pairs. Further, it is important to note that collaboration and computation using a set of data is limited to a set of encrypted data that is able to be decrypted by the same private key. Hence, to facilitate wide-ranging collaboration and computation, limiting the system to a single private key may be advantageous.

[00140] Analyst terminals 38 may include devices such as smartphones, tablet computers, desktop/laptop computers, and similar operable by analysts 40 such as researchers, clinicians, physicians, administrators, insurers, and the like. Analyst terminals 38 may initiate homomorphic calculations performed at the medical data system 22 and may have the encrypted results of such calculations decrypted by the private key held by the key authority system 36. Plaintext results of the calculations may then be outputted at the analyst terminals 38 for further calculation and study.

[00141] The private key can be provided to decrypt calculation results in various ways, depending on specific requirements of various implementations according to the present

invention. Encrypted results of calculations may be transmitted to the key authority system 36 for decryption at the key authority system 36, with the decrypted plaintext results (DR) 42 of the calculations being transmitted to one or more analyst terminals 38 for output via a secure channel 44, such as a secure subnetwork, a VPN operating through the wide-area network 24, or similar network that offers increased security. Additionally, or alternatively, such a secure channel 44 may be used to transmit the private key from the key authority system 36 to the analyst terminals 38 to decrypt calculation results at one or more analyst terminals 38. The secure channel 44 need not be limited to network communications. For example, analyst terminals 38 may be situated at secure locations, such as within physically secure areas of healthcare or research facilities, thereby offering a physical aspect to the secure channel 44, if the private key or encrypted data is to be transmitted over a network. Alternatively, the secure channel 44 may be mainly or exclusively physical and the private key or encrypted data can be copied onto physical key cards, memory sticks, or similar devices that can be used to manually convey the private key or encrypted data to the analyst terminals 38.

[00142] In operation, data may be continually collected by the remote medical devices 20, encrypted at the remote medical devices 20, and transmitted to the medical data system 22 as encrypted data (ED) 50. In general, any device transmitting data to the medical data system 22 may be configured to encrypt its data prior to transmission. The medical data system 22 may store the encrypted data 50 for as long as desired. At any time, an analyst terminal 38 may be used to select a set of data for analysis and to configure a calculation to be performed on the selected set of data. This information may be sent by the analyst terminal 38 to the medical data system 22 as a calculation command (CC) 52 that triggers the medical data system 22 to perform the calculation on the selected encrypted data, according to a homomorphic technique, without decrypting the data to obtain encrypted results 54. The encrypted results 54 of the calculation may then be transmitted to the analyst terminal 38. The analyst terminal 38 may then obtain decrypted results 42 using the secure channel 44 to communicate with the key authority system 36.

[00143] Advantageously, data may not be decrypted during the performance of calculations. Patient privacy may be improved and it is contemplated that more patients may volunteer their medical data to be used in medical studies knowing that their data is better protected. In addition,

opportunities for man-in-the-middle and other types of attacks may be mitigated due to the data and calculation results being transmitted in encrypted form and due to tight control of the private key.

[00144] In other applications, the remote devices 20 such as wearable devices 26, patient terminals 28, laboratory terminals 30, data system 22, terminals 38, remote devices 130 and other components may not be specifically medically oriented devices and may be general purposes devices or devices made specific to a chosen application.

[00145] FIG. 2 shows an example remote medical device 20. The remote medical device 20 may include a sensor 60 and/or input device, an encryption engine 62, and a network interface 64. The remote medical device 20 may further include memory (e.g., RAM, hard-drive, solid-state drive, etc.) for storing captured data 66, a public key 68, and encrypted medical data 50. The remote medical device 20 may also include a processor configured to execute the encryption engine 62 and control operations of the remote medical device 20.

[00146] The sensor and/or input device 60 may be configured to capture medical/physiological data and/or non-medical data 66 of an individual such as a patient. Example sensors include heart rate sensors, respiration sensors, blood sugar sensors, blood pressure sensors, and the like, with numerous other examples discussed elsewhere herein. Any sensor and/or input device 60 capable of measuring or capturing medical data is contemplated. Example input devices include a keyboard or touchscreen, for manual entry of medical or non-medical data. An input device may be used together with a sensor, such that the collected data includes manually inputted data as well as directly measured data. The type and nature of the data captured is not particularly limited.

[00147] The encryption engine 62 may be configured to apply the public key 68 to encrypt the captured data 66 to generate encrypted data 50. The encryption engine 62 may be configured to perform fully homomorphic encryption as discussed above.

[00148] The network interface 64 may be configured to communicate encrypted data 50 to the network 24 and specifically to the medical data system 22 (FIG. 1). After the encrypted data 50 has been transmitted to the medical data system 22, it may be deleted to save memory.

[00149] FIG. 3 shows an example of the data system 22, which, as mentioned, may be a medical data system 22. The medical data system 22 may include a network interface 80, a data accumulator 82, a query constructor 84, a patient/user interface 86, and a calculation engine 88. The medical data system 22 may further include memory (e.g., RAM, hard-drive, solid-state drive, etc.) for storing encrypted data 50, encrypted calculation results (ER) 54, patient data 90 (or other data in non-medical applications), and authorizations 92. The medical data system 22 may also include a processor configured to execute the data accumulator 82, the query constructor 84, the patient/user interface 86, and the calculation engine 88 and to control operations of the medical data system 22. It is noted that no encryption key need be stored at the medical data system 22, as the medical data system 22 performs its calculations on encrypted data.

[00150] The network interface 80 may be configured to receive data and commands from the network 24. The network interface 80 may be configured to receive encrypted data 50 and patient commands from the remote medical devices 20. The network interface 80 may be configured to receive calculation commands from analyst terminals 38.

[00151] The data accumulator 82 may be configured to control the capture of encrypted data 50 from the plurality of remote medical devices 20. The data accumulator 82 may be configured to periodically interrogate each remote medical device 20 for new encrypted data, receive such encrypted data in response, and store such encrypted data 50 in memory at the medical data system 22. The data accumulator 82 may be configured to reference the authorizations 92 as a condition for collecting data.

[00152] The query constructor 84 may be configured to receive calculation commands from analyst terminals 38. A calculation command triggers commencement of a calculation by the calculation engine 88. A calculation command may include parameters specifying a set of the encrypted data 50 on which to conduct a calculation as well as parameters specifying the nature of the calculation. The query constructor 84 may be configured to provide to the analyst terminals 38 a summary of the encrypted data 50 available for calculation. The query constructor 84 may be configured to reference the authorizations 92 as a condition for using elements of encrypted data 50 in calculations.

[00153] The patient/user interface 86 may be configured to receive patient commands and data from remote medical devices 20, analyst terminals 38, or other devices and to output information about the encrypted data 50. Specifically, the patient/user interface 86 may be configured to receive patient data 90 that is not necessarily encrypted. Patient data 90 may include data such as unique identifier, name, address, birthdate, general health status and indications, descriptions of the types of encrypted data 50 collected, and similar. Patient data 90 may include data useful in designing medical studies. Patient data 90 may also include associations to the encrypted data 50, such that elements of encrypted data 50 may be linked to the patient data 90 that is considered useful for designing medical studies. For example, an association of patient data 90 to encrypted data 50 may include a unique identifier, such as a hash or serial number, of the patient in both the encrypted data 50 and the patient data 90. It may be beneficial to include the unique identifier in the encrypted data 50 in plaintext form, such as via an unencrypted metadata field attached to the encrypted data 50, a file name, or an unencrypted database field in association with encrypted data 50 stored in a database.

[00154] The patient/user interface 86 may be configured to receive commands to control authorizations 92 that are granted by patients or other individuals to, for example, analysts at terminals 38. Authorizations 92 may include data indicative of the consent to collect and store encrypted data 50 and consent to make encrypted data 50 available to the calculation engine 88. In medical applications, authorizations 92 may include one or more many-to-many mappings that map patients to data and further to individuals or organizations, such that each patient (or his/her legal representative) can give consent to provide any type of data (e.g., blood pressure, heartrate, etc.) to any individual or organization (e.g., the patient's primary caregiver, an doctor in charge or a medical study, a hospital, etc.). Authorizations 92 may also include time windows for consent, such that consent is automatically withdrawn after expiry of a selected time.

[00155] The patient/user interface 86 may include an authentication system, such as a username and password log-in authentication system, for verifying that users who modify patient data 90 and authorizations 92 are authorized to make such changes.

[00156] The calculation engine 88 may be configured to perform homomorphic calculations on encrypted data 50 according to received parameters defining the set of encrypted data 50 and

the calculations to perform. The calculation engine 88 outputs encrypted results 54 that are transmitted via the network interface to the key authority system 36 or the analyst terminal 38. The calculation engine 88 can be configured to perform any suitable calculation in the encrypted domain.

[00157] Such calculations are contemplated to include addition, multiplication, discrete calculations, continuous calculations, comparisons using relational operations, combinations of such, and similar. Specific medical calculations may include atrial fibrillation stroke risk score calculations, cardiovascular risk score calculations, and genotype calculations, without limitation. Further, such calculations may be modeled as polynomial series. To achieve this, the calculation engine 88 may be configured as described below.

[00158] FIG. 4 shows an example key authority system 36. The key authority system 36 may include a network interface 100, an authorization processor 102, and a decryption engine 104. The key authority system 36 may further include memory (e.g., RAM, hard-drive, solid-state drive, etc.) for storing encrypted calculation results 54, a private key 106, and decrypted results 42. The private key 106 corresponds to the public key 68 (FIG. 2) stored in the remote medical devices 20. The key authority system 36 may also include a processor configured to execute the authorization processor 102 and the decryption engine 104, as well as to control operations of the key authority system 36.

[00159] The network interface 100 may be configured to receive encrypted results 54 from analyst terminals 38 via the secure channel 44 (FIG. 1). The network interface 100 may be further configured to transmit the decrypted results 42 to analyst terminals 38. Alternatively, or additionally, the network interface 100 may be configured to transmit the private key 106 to the analyst terminals 38 via the secure channel 44.

[00160] The authorization processor 102 may be configured to restrict access to the key authority system 36 to authorized users. The authorization processor 102 may include an authentication system, such as a username and password log-in authentication system or an electronic credential verification system, for verifying users who attempt to access the decrypted results 42 or the private key 106 or both.

[00161] The decryption engine 104 may be configured to apply the private key 106 to decrypt the encrypted calculation results 54 to generate the decrypted results 42. The decryption engine 104 may be configured to perform homomorphic decryption as discussed herein.

[00162] FIG. 5 shows a data system 120, which, as mentioned, may be a medical data system 120. The medical data system 120 may include components discussed herein, such as medical data system 22. The description for such components can be referenced with like reference numerals indicating like components. For sake of clarity, only differences will be discussed in detail. The medical data system 120 may replace the medical data system 22 in the system of FIG. 1.

[00163] The medical data system 120 may be configured to generate alerts based on calculations performed on encrypted data 50 received from patient remote medical devices 20. The medical data system 120 may transmit these alerts via the network 24 to remote devices 130 (FIG. 1) operated by medical professionals who may act on such alerts. It is contemplated that generated alerts remain in the encrypted domain. As such, alerts may be routed through the key authority system 36 for decryption prior to being transmitted to the relevant remote devices 130 in plaintext form. In such case, privacy may be enhanced by having the plaintext form of the alert be a general indication of the nature of the alert (*e.g.*, a text alert such as “blood pressure alert”) rather than specific numerical values, as it is contemplated that recipients of alerts may have background knowledge of the specific medical condition and perhaps knowledge of specific alert trigger values. Alternatively, remote devices 130 may be provided with the private key to decrypt received alerts.

[00164] The medical data system 120 may include alert triggers 122 executable by a processor and configurable by authorized users. The alert triggers 122 and the calculation engine 88 may perform comparisons between encrypted results 54 and encrypted alert conditions 124 stored in memory and configurable by authorized users. Encrypted alert conditions 124 may be initially received in unencrypted form via the network interface 80 before being encrypted by an encryption engine 62 using the same public key 68 that encrypts the data at the remote medical devices 20. Encrypted alert conditions 124 may be applied to data represented by the encrypted results 54 and alert triggers 122 issue alerts for data that meet the conditions. Such alerts may be

configured to be transmitted via the network interface 80 and the network 24 to remote devices 130 of selected authorized users, such as medical professionals. Alerts may be communicated via the key authority system 36 for centralized decryption prior to being forwarded to the selected authorized users in plaintext form. Alternatively, the remote devices 130 may store the private key 106 and may be configured with a decryption engine 104 to decrypt the alerts. For example, a doctor may set an alert condition 124 for a patient, identified by patient data 90, using a blood pressure equation homomorphically evaluated in the encrypted domain by the calculation engine 88 using encrypted raw blood pressure data continuously collected by a wearable medical device 26. The encrypted alert condition 124 may be a particular maximum, minimum, or interval of the blood pressure equation that, if met or exceeded, causes the alert trigger 122 to send an electronic alert message to remote devices 130 operated by the doctor and by a specialist who may be otherwise be unaware of the specific alert conditions. Advantageously, the specific evaluation and the values evaluated may remain in the encrypted domain, so as to improve patient privacy. That is, no entity except for the patient and the user setting an alert is necessarily aware of blood pressure equation, the condition(s), and the blood pressure data considered, yet an alert may be issued to any entities based on same.

[00165] The alert triggers 122 may be evaluated on a periodic basis or upon detecting new encrypted data 50. The alert triggers 122 may store information concerning delivery of the alert, such as network addresses (e.g., email addresses, phone numbers, etc.) of the destination remote devices 130 that are to receive the alerts.

[00166] Various components of the medical data systems 22, 120, and specifically the calculation engine 88, may be implemented as one or more hardware devices. Such hardware devices may be configured to implement the computational techniques discussed herein using only hardware or by using hardware that executes program code. A suitable hardware device may be configured to implement the computational techniques discussed herein using, for example, Chinese Remainder Theorem (CRT), Number Theoretic Transform (NTT), one or more memory blocks, one or more memory interfaces, matrix multiplications, matrix additions, or a combination of such. One such suitable hardware device to achieve this is a Graphics Processing Unit (GPU). Other examples of suitable hardware device include a field-programmable gate array (FPGA) and an application-specific integrated circuit (ASIC).

[00167] Embodiments of the invention may be used to implement statistical techniques used in genomic association studies. Examples of such techniques include the Pearson Goodness-of-Fit test and the Cochran-Armitage Test for Trend (CATT).

[00168] Genotypes and phenotypes may be encoded and encrypted as follows. In this example, a laboratory terminal 30 (FIG. 1) is configured to perform the encoding and encryption of genetic data obtained by medical personnel 34 from patients 32.

[00169] Regarding genotype encoding, a table containing genotype information is constructed in which each row corresponds to genotype information about a single person. For bi-allelic genes, each person's gene is encoded using three ciphertexts c_{AA} , c_{Aa} , c_{aa} . These ciphertexts will encrypt a "1" only in the case that the equality statement in FIG. 6H is satisfied, otherwise they encrypt "0". In case that the person's genotype at the specified locus is not known, all ciphertexts will encrypt "0".

[00170] Regarding genotype/phenotype correlation encoding, pre-computation time, and noise in the final encrypted result can be reduced, which will allow further computations on the resulting ciphertext if needed. A correlation matrix between genotypes {AA, Aa, aa} is generated and phenotypes {affected, unaffected} to generate a 3×2 correlation matrix with ciphertexts encrypting a "1" at a single location corresponding to this person's genotype/phenotype condition, "0" otherwise, as shown in FIG. 8. If the genotype or phenotype of this person is unknown, then all ciphertexts will be encrypting a "0".

[00171] Regarding the Pearson Goodness-of-Fit Test, this test is used to check if a gene is in Hardy-Weinberg Equilibrium (HWE). The HWE can be used for testing if gene allele frequencies are independent. Assume A and a are two alleles in a given gene, and that N_{AA} , N_{Aa} , and N_{aa} are the corresponding number of genotypes AA , Aa , and aa , respectively. Taking $N = N_{AA} + N_{Aa} + N_{aa}$ to be the total number of genotypes, the parameters in FIG. 6I can be computed homomorphically, where N_{ij} can be computed as in FIG. 6J. These parameters can then be used to compute the final deviation test statistic by the equation of FIG. 6K.

[00172] The Cochran-Armitage Test for Trend (CATT) is used to determine if an allele is associated to a disease or not. The test can be computed as shown by the equation of FIG. 6L

using the computed parameters of FIG. 6M, in which w_i are predetermined weights $\in \{0, 1, 2\}$, N_{ij} represents the number of individuals who are affected/unaffected with a certain disease and have genotype ij . The parameters N_{ij} , R_i , C_i are described in FIG. 9.

[00173] Predictive equations may be used to check for different diseases. In order to compute the predictive equation, patient medical data may be used. To protect such medical data, homomorphic encryption may be used to encrypt these data and computation of regression equations may be performed in a secure environment, such as by using the calculation engine 88 at the medical data system 22, 120. In one example, the predictive model for the likelihood of having a heart attack in an unspecified period is given by the logistic regression function shown in FIG. 6N, where x is calculated from the equation of FIG. 6O in which a is the age, sys is the systolic blood pressure, dia is the diastolic blood pressure, $chol$ is the cholesterol level, ht is the height, wt is the weight.

[00174] To implement the exponential expression in the equation of FIG. 6L, a Taylor series can be used up to an acceptable order to result in a sufficient accuracy (*e.g.*, degree 7) as shown in FIG. 6P. Since homomorphic encryption uses integer arithmetic, the equations of FIGs. 6O and 6P may be normalized with appropriate accuracy to the equations of FIGs. 6Q and 6R.

[00175] Inputs to the equations of FIGs. 6Q and 6R are represented as integer numbers and encoded in binary format in adjacent polynomial slots. When these binary polynomials are multiplied together, they result in correct output. Multiplication by constants may be implemented through sequential additions. To compute the correct output, the result of the equation of FIG. 6Q can be transmitted back to a device (*e.g.*, remote device 20) having the encryption engine 62 to be re-encrypted and applied to the equation of FIG. 6R to compute the final result.

[00176] With respect to relational operations, such as those discussed above with regard to the calculation engine 88, homomorphic relational operations, such as greater than and less than ($>$ and $<$), may be implemented. The relational operation $a > b$ results in one bit only, which is equal to “1” if $a > b$, and “0”, otherwise. Individual bits of each input a and b are encrypted and denoted as a_i and b_i , respectively. With k -bit numbers and starting with the least significant bit, the output is given by z_k as shown in FIG. 6S. If any of the two inputs were not encrypted, the

complexity of the calculation represented in FIG. 6S may be greatly reduced. For example, if b is not encrypted, the resulting equations will be as shown in FIG. 6T.

[00177] Regarding blood pressure tests, the calculation engine 88 may be configured to accept encrypted systolic and diastolic pressures and return a blood pressure classification, such as those listed in FIG. 10. To implement such a computation, the input blood pressure is compared to multiple ranges to determine the blood pressure classification. This may be done by the calculation engine 88 being configured with a function, such as a function according to the pseudocode of FIG. 11. The “Res_Sys” and “Res_Dia” input parameters for the pseudocode function each contain an encrypted number ranging from 0 to 5 indicating the classification of the input encrypted blood pressure measurements compared to the blood pressure range in the systolic “Sys_Ref” and diastolic “Dia_Ref” reference lists, respectively.

[00178] The blood pressure computation above is but one example and many similar examples of applying such relational operations are contemplated. For example, calculating the CHADS₂ score for atrial fibrillation stroke risk may be performed in a similar manner. Many other similar computations are within the scope of the invention.

[00179] In another example, a Framingham Coronary Heart Disease Risk Score (FCRS) is a computation used to estimate the 10-year cardiovascular risk of a person. The computation is based on several factors, including sex, age, cigarette smoking, total cholesterol, high-density lipoprotein (HDL) cholesterol, and systolic blood pressure. Each variable is compared against a range of numbers and a score is assigned for each range and added to the overall Framingham score. For example, if a man’s age is 50 years, 6 points are added to the total score.

[00180] To implement this computation, the relational operation concept described above may be extended to compare an input “ b ” against a lower bound “ a ” and against an upper bound “ c ”. A function $a < b < c$ is implemented to give only a “1” if “ b ” is within this range and “0” otherwise.

[00181] One way to implement $a < b < c$, is to implement it in two steps, namely $a < b$ and then $b < c$ and multiply the results. However, in this approach, as in many GSW-derived homomorphic-encryption schemes, homomorphic multiplication is asymmetric, which prevents

multiplying two non-fresh ciphertexts, a non-fresh ciphertext being a ciphertext that contains results of ciphertext multiplications. The equations for $a < b < c$ may advantageously be reformulated for implementation at the calculation engine 88 such that fresh ciphertexts may be multiplied by an accumulator.

[00182] To achieve such reformulation, for a k -bit input “ b ” with encrypted bits $[b_0, b_1, \dots, b_{k-1}]$ compared against “ a ” and “ c ” (encrypted or unencrypted) the function $a < b < c$ may be implemented as shown in the equations of FIGs. 6U and 6V. The equations of FIG. 6U define bit “0” and the equations of FIG. 6V define each bit “ i ”, where x_i, y_i, z_i are accumulators, b_i is encrypted bit i of input b , and a_i, c_i are the bits of the lower and upper bounds, respectively. The result of the relational operation is contained in z_k . In the case that a_i and c_i are plaintexts, the equations of FIG. 6V may be further simplified as described with respect to in FIG. 6T.

[00183] Tests were conducted on a system constructed according to the present invention. FIG. 12 details the test system used. FIG. 13 summarizes performance results of the homomorphic operations for the present invention compared to Khedr, A., Gulak, G., and Vaikuntanathan, V. SHIELD: Scalable Homomorphic Implementation of Encrypted Data-Classifiers, Cryptology ePrint Archive, Report 2014/838, 2014 (<http://eprint.iacr.org/2014/838.pdf>) shown in FIG. 13 at “Khedr 2014”; Lauter, K., Lopez-Alt, A., and Naehrig, M. Private Computation on Encrypted Genomic Data. Tech. Rep, MSR-TR-2014-93, June 2014 shown in FIG. 13 at “Lauter 2014”; and LBos, J. W., Lauter, K., and Naehrig, M. Private predictive analysis on encrypted medical data, In *Journal of biomedical informatics*, Elsevier Inc., 2014, pp. 234–243 shown in FIG. 13 at “LBos 2014”. The results reported for “Lauter 2014” are for larger parameters with larger circuit depth, as the circuit depth used to test the present invention is higher than in “Lauter 2014” with the chosen parameters. As can be seen, a CPU implementation of the present invention may achieve a 58 times speedup for a multiplication operation as compared to “LBos 2014”. By additionally exploring the parallelizable properties of the present invention, a 104 times speedup may be achieved by distributing computations on GPU cores. This resulted in an overall 6085 times speedup for the multiplication operation compared to “LBos 2014” and a 286 times speedup compared to “Lauter 2014”.

[00184] The present invention may be advantageously scalable across multiple GPU cards. Experiments were made using four GPU cards connected to the same computer to measure loss in performance due to cross-GPU communication. By partitioning large problems into small ones, computations can be scheduled among all four GPUs to obtain a speedup of 3.946 times, which indicates that communication overhead was reduced.

[00185] Performance results of the Pearson Goodness-of-fit test and the CAAT test, the predictive analysis, and the relational operations and blood pressure application described above are summarized in FIG. 14. Results are compared to “Lauter 2014”, “LBos 2014”, and Cheon, J. H., Kim, M., and Kim, M. Search-and-compute on Encrypted Data, Cryptology ePrint Archive, Report 2014/812, 2014 (<http://eprint.iacr.org/>) shown in FIG. 14 at “Cheon 2014”. It is noted that the reported results for “Cheon 2014” corresponds to the ring Z_{14} , which is because, in the blood pressure example of the present invention, more than two numbers are added to get the final correct result.

[00186] Medical data including physiological data are complex, voluminous and may be very noisy. Noise in data collection may arise, for example, from the techniques used to collect the data, human error, intrinsic stochasticity of the system itself, intra-individual variability of repeated observations and within-group/population variability or may be indicative of abnormality, disease or illness. Because of the noise, data mining includes data clustering and characterization applications that may be used for data exploration and to discover patterned diseases from stored patient data. Machine learning applications may be also useful in predicting individual patient outcome from the collected data from patients treated in the past for which the final diagnoses were verified. Both data mining and machine learning may help during screening, diagnosis, therapy, prognosis, monitoring, epidemiological studies, biomedical/biological analysis, hospital management, medical instruction and training.

[00187] Additionally, according to an embodiment, the encryption scheme described herein may be used in privacy-preserving machine learning applications, in privacy-preserving data mining including privacy-preserving data clustering applications, and in general secure computations on medical, financial or other confidential data. For example, the calculation engine 88 may be configured to perform homomorphic calculations related to privacy-preserving

machine learning applications, in privacy-preserving data mining including privacy-preserving data clustering applications on encrypted data.

[00188] Numerous advantages may be apparent from the above description of the present invention. Concerning wearable and portable devices and the “Internet of Things” (IoT), the present invention can be used to encrypt all data measured by wearable and portable devices prior to uploading such data to the cloud. This can be very useful to help researchers and clinicians conduct research on confidential data, in a manner that preserves privacy. As discussed, these devices can store public encryption keys produced by a centralized entity, which is also responsible for the control and distribution of private/secret keys to, for example, hospitals and other medical facilities where computation results/alerts are to be decrypted. Wearable/portable devices need only encrypt the captured data, and the modest processing power that is known in these kinds of devices is not a significant hindrance to implementation. Since performance of the encryption function is not necessarily time critical, embedded processors can encrypt measured data within seconds instead of milliseconds and still have acceptable performance.

[00189] The present invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. Certain adaptations and modifications of the invention will be obvious to those skilled in the art. Therefore, the presently discussed embodiments are considered to be illustrative and not restrictive, the scope of the invention being indicated by the appended claims rather than the foregoing description and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced therein.

WHAT IS CLAIMED IS:

1. A method for securely transmitting and extracting information from encrypted data without fully decrypting the data, comprising:

receiving a data request, with at least a portion of the data request encrypted according to a homomorphic encryption scheme, and the encrypted portion of the data request comprising at least one set of confidential data comprising a plurality of strings corresponding to a plurality of data fields in the confidential data;

retrieving one or more sets of encrypted comparison data from a database, each set of encrypted comparison data encrypted according to the homomorphic encryption scheme and comprising a plurality of data fields corresponding to the plurality of data fields in the set of confidential data;

comparing the encrypted set of confidential data from the data request with each set of encrypted comparison data using one or more homomorphic operations on the plurality of strings collectively to determine which set of encrypted comparison data matches the encrypted set of confidential data and validating the set of confidential data upon a match;

generating an encrypted indicator indicating success or failure of validating the set of confidential data; and

forwarding the encrypted indicator to a party associated with the data request,

wherein the set of confidential data is never decrypted during the method.

2. The method of claim 1, wherein the homomorphic encryption scheme is a fully homomorphic encryption scheme or a somewhat homomorphic encryption scheme.
3. The method of claim 1, wherein the data request further includes at least one set of non-confidential data.

4. The method of claim 1, wherein the data request includes a plurality of sets of confidential data.
5. The method of claim 4, further including executing a step of determining which set of confidential data is to be validated.
6. The method of claim 1, wherein a portion of the data request is compared to the set of comparison data to reduce the size of the set of comparison data prior to comparing the encrypted set of confidential data.
7. The method of claim 1, wherein the one or more homomorphic operations combine to form an XNOR operation.
8. The method of claim 1, wherein the set of confidential data is from one of: an identity card, a passport, a driver's license, a security card, a bank card, a bank account number, a credit card, and a health card.
9. The method of claim 1, wherein the encrypted indicator is sent to a party different from the party the data request was received from.

10. The method of claim 1, wherein the steps of:

receiving a data request comprises receiving an at least partially homomorphically encrypted request to complete a financial transaction, which is homomorphically encrypted according to a lattice-based Fully Homomorphic Encryption (FHE) scheme, and the plurality of strings corresponding to the plurality of data fields in the confidential data comprises confidential cardholder data including an account number, non-confidential cardholder data and transaction data;

retrieving one or more sets of encrypted comparison data from a database comprises retrieving homomorphically encrypted comparison cardholder data encrypted according to a lattice-based FHE scheme;

comparing the encrypted set of confidential data from the data request with each set of encrypted comparison data using one or more homomorphic operations on the plurality of strings collectively to determine which set of encrypted comparison data matches the encrypted set of confidential data and validating the set of confidential data upon a match comprises comparing the confidential cardholder data to each set of the comparison cardholder data using the one or more homomorphic operations to determine which set of comparison cardholder data matches the confidential cardholder data and validating the confidential cardholder data;

generating an encrypted indicator indicating success or failure of validating the set of confidential data comprises generating and encrypting an indicator indicating authorization or rejection of the request to complete the financial transaction based upon at least the validation of the confidential cardholder data; and

forwarding the encrypted indicator to a party associated with the data request comprises forwarding the encrypted indicator indicating authorization or rejection of the request to complete the financial transaction to a party seeking authorization to complete the financial transaction.

11. The method of claim 10, wherein the confidential cardholder data includes one or more of: a portion of the account number, the account number, an expiry date and a Card Verification Value (CVV) number.

12. The method of claim 10, wherein the non-confidential cardholder data includes one or more of: a bank name, a cardholder name, a Bank Identification Number (BIN) and the last four digits of the account number.

13. The method of claim 10, wherein the transaction data includes one or more of: a transaction amount, a transaction date and a merchant identifier.

14. The method of claim 10, further including executing a step of retrieving the transaction data and executing the step of comparing the transaction data using the one or more homomorphic operations to determine whether the transaction amount can be authorized.

15. The method of claim 10, wherein the non-confidential cardholder data is encrypted using a public key.

16. The method of claim 15, wherein the public key is held by a party making the request to complete the financial transaction, and a private key is held by a party receiving the request to complete the financial transaction.

17. The method of claim 10, wherein the one or more homomorphic operations combine to form an XNOR operation.
18. The method of claim 10, wherein the financial transaction is a credit card transaction.
19. The method of claim 10, wherein the request to complete the financial transaction is received by one or more of: a bank, a credit card company, a card issuer or a payment processor.