Chapter 9

Generation of Cryptographic Keys from Personal Biometrics: An Illustration Based on Fingerprints

Bon K. Sy and Arun P. Kumara Krishnan

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/51372

1. Introduction

Biometric approach for authentication is appealing because of its convenience and possibility to offer security with non-repudiation. However, additional hardware such as biometric scanners and complex software for feature extraction and biometric template matching are required if biometric approach is to provide security for protecting sensitive data such as personal health information.

Cryptographic approach, on the other hand, ties data protection mathematically to the *Key* that is utilized to protect it. This allows a data owner to have complete control over one's personal information without relying on, or relinquishing control to, a third party authority. The protection of personal sensitive information is also not tied to complex software and hardware systems that may need constant patches.

Biometric authentication and authorization for data protection could be thought of as enabling security based on "what one is." The lynchpin of biometric security is the use of sufficiently unique, but often imprecise, physiological or behavioral traits to characterize an individual for authentication and identification purposes. The characterization is expressed in form of some biometric signature, which often can be reduced to some feature vector or matrix representation. For example, a biometric face could be expressed in terms of a linearized vector of color distribution [1], EigenMap [2], or Eigen Face components [3]. In our research a fingerprint is expressed in terms of a 320x1 vector of integers containing minutia point information, and a voice signature is expressed in terms of a 20x1 mean vector and a 20x20 covariance matrix of Mel cepstrum characterizing the multi-variant Gaussian distribution of an individual's voiceprint [4]. The security parameter for assessing the strength of a biometrically based approach is typically related to the size of the underlying feature vector (or matrix) and the number of bits for representing a value, as well as the biometric data dis-

tribution leading to inter and intra variability --- a main source of false negative or false positive alarms when applying biometric approach for security.

On the other hand, cryptographically based security could be thought of as a security approach based on "what one knows." The lynchpin of cryptographic security is the secret key for decrypting a cipher text that is the encrypted from sensitive data. The security parameter for assessing the strength of a cryptographic approach is typically the key size in terms of the number of bits, and information leakage which can be measured by the information gain on the sensitive data given its corresponding cipher text and the mathematical structure of the cryptographic mechanism for encryption/decryption. In order to mitigate the risk of information leakage, semantic security is desirable. In brief, we say it is semantically secure with IND-CPA property (INDistinguishability under Chosen Plaintext Attack) [5] if one is given the cipher texts of some encryption, one could not tell whether the cipher texts correspond to the encryption of the same text; i.e., the given cipher texts are indistinguishable, thus thwarting a Chosen Plaintext Attack.

In theory, the size of a biometric signature or the size of a secret key in cryptography could be increased indefinitely to increase the security strength. However, in practice, the limitation in the resolution of biometric sensors, among other factors, does not allow the security strength to be scaled proportionally. On the other hand, cryptographic approach has its own drawback too. Since the confidentiality of sensitive data is protected through encryption, one must keep the decryption key as a secret. Generally the secret key is generated and withheld by the party that handles the decryption of the sensitive data. If the secret key is compromised, the confidentiality of the sensitive data is compromised.

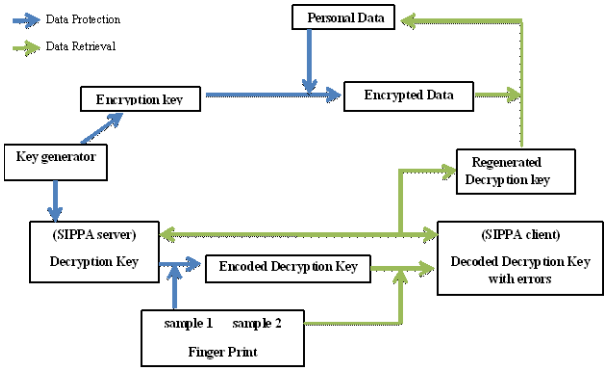

Figure 1. Cryptographic key (re)generation protocol.

The objective of this research is to investigate a secure computation protocol for (re)generating cryptographic key based on biometric signatures. More specifically, we want to protect sensitive personal data based on strong cryptographic approach (e.g., AES256) [6]. But we do not store the secret key anywhere. Instead, we (re)generate the secret key based on an individual's biometric signatures; e.g., fingerprints or voiceprints. In this research we focus

http://dx.doi.org/10.5772/51372

on fingerprints. In other words, we use a (cryptographic) key generator to generate the encryption/decryption key pair, and use the key to encrypt the sensitive personal information. Afterward, we discard the key generator and the encryption key. Then, we encode the decryption key using an individual's biometric signatures. The encrypted personal data and the encoded decryption key are given to the individual while the original decryption key will be kept in a separated server without the sensitive personal data. During the retrieval process, the data owner will use his/her biometric signature to *subtract* from the encoded decryption key, and then use it to reconstruct the exact decryption key through SIPPA-2.0 for decrypting the personal information.

The contribution of this research is a cryptographic key (re)generation scheme based on biometrics. The scheme is comprised of the following components:

1. A client/server model is developed for privacy preserving cryptographic key regeneration; where the decryption key --- no (encrypted) personal data --- is stored in the server. In this case, the security and the privacy of the data are still preserved even if the server is compromised. The client party, on the other hand, holds a storage device containing the encrypted sensitive data and the decryption key encoded with personal biometrics. Without the individual's biometrics, decryption key cannot be decoded, thus the confidentiality of the encrypted sensitive data is still preserved even if the storage device is compromised or stolen.

2. A 2-party secure computation protocol, referred to as *SLSSP*(Secure Linear System Solution Protocol), is developed for privacy preserving data comparison based on solving linear equations that yields a solution; and this solution can be used to estimate the similarity or the closeness between the source (server side) data and the target (client side) data.

3. A secure information processing protocol, referred to as SIPPA-2.0 (Secure Information Processing with Privacy Assurance Version 2.0), is developed to realize a 2-party message exchange protocol through which the client and the server could determine how similar their data are to each other by using SLSSP --- without ever revealing their own private data. Furthermore, the client can perfectly reconstruct the server side data if the server provides helper data after the server side determines that the client data is sufficiently similar; whereas the perfect reconstruction is based on a geometrical relationship in the vector space among the helper data and the Eigen components of the client and source data.

4. For proof-of-concept, we implement the cryptographic key regeneration scheme as an Android application for privacy preserving health data management. More specifically, the Android application stores in the memory of a smart phone the encrypted health data and the decryption key encoded by personal fingerprints. During the health data retrieval process, the Android application receives a fingerprint sample (from a fingerprint scanner via a Bluetooth channel) and uses it as an offset to the encoded decryption key to arrive at an estimated decryption key; and then acts as a client to invoke SLSSP and SIPPA for reconstructing a perfect decryption key based on the helper data of the server when the estimated decryption key is sufficiently similar to the actual decryption key.

2. Problem formulation and Related Work

2.1. SLSSP and SIPPA formulation

The concept of SLSSP (formerly referred to as PPCSC) and SIPPA was first introduced in the book chapter of *Biometrics* [7] (INTECH 2011). SLSSP aims at solving the following problem:

Let P1 and P2 be two parties and each has private data (Ai, bi) for i=1,2; whereas Ai is a matrix and bi is a vector. Both parties wish to compute x in (A1+ A2)x = (b1 + b2) without revealing its own private data (Ai, bi) to each other.

In section 6 we will present the protocol design of SLSSP --- which is an improved version of PPCSC. The security and privacy analysis of SLSSP under realistic and reasonable model assumptions will be presented in the appendix. Specifically, the security and privacy of SLSSP will be analyzed under these model assumptions; authenticated communication channels with computational security, the behavior of the participants being rational, and the computational power of the adversary is polynomial bounded[1]. Our SLSSP possess the following two desirable security properties:

(a) The malicious behavior of a party deviating from semi-honest behavior is detectable through verifiable correctness of the private message exchange.

(b) The practical implementation of the homomorphic encryption for SLSSP realizes semantic security with IND-CPA property.

SIPPA-2.0 is the second generation of our secure computation protocol for the following 2-party scenario; where a client party can reconstruct source data of a server party under the following conditions:

(a) The client party must possess some client data that is a "sufficiently good approximation" of the source data, in order to initiate the SIPPA process.

(b) Rather than revealing the source data of the server party to the client party, only some helper data related to the Eigen components of the source data is provided (by the server party) to the client party for reconstructing the source data.

2-party SIPPA-2.0 is a secure process for private data reconstruction with the following desirable security and privacy properties:

(a) The server party retains complete control over the sharing of helper data – thus keeping private the source data - based on the "closeness" between the client and server data.

(b) The server party can determine the "closeness" between the client and server data without the client party ever disclosing it's data to the server party – thus the privacy of both client and server data is respected.

---

1 The analysis presented in the appendix is part of our other paper entitled "SIPPA-2.0 – Secure Information Processing with Privacy Assurance (version 2.0)" appeared in the proceeding of the PST 2012, Paris, France. It is included to make this chapter self-sufficient.

http://dx.doi.org/10.5772/51372

(c) Only if the client and server data are sufficiently close and the server sends the client the helper data, the client can perfectly reconstruct the source data.

The major improvement of SIPPA-2.0 over the previous one is the use of a newly discovered computational geometry that allows perfect data reconstruction from the solution x securely computed in SLSSP. Before we discuss the details of SIPPA-2.0, we first present a survey on related work.

2.2. Related Work

Our research draws from various other works in the literature of biometrics and secure computation. We first discuss the major works in biometrics on which we draw upon, and then the advances in secure computation to which SIPPA-2.0 is related.

Hao et al. [8] [9] were among the pioneers in successfully melding biometrics with cryptography. Iris codes typically contain 10-20% error between samples of the same eye. By utilizing a two tier approach of Hadamard and Reed Solomon Codes to correct both random and burst errors; they achieved a successful *Key* retrieval rate of over 99%. Our research draws from their overall approach of appending a random *Key* with biometrics and then negating this appended biometrics with another biometric sample to retrieve the original cryptographic *Key*. However, SIPPA-2.0 allows us to achieve perfect key reconstruction without using any error correction codes such as Hadamard or Reed Solomon.

Barni et al. [10] utilize a finger-code approach to represent a fingerprint partitioned into sectors from a reference point. Their approach is to utilize these finger-codes to perform fingerprint matching over homomorphic encryption; essentially computing a Euclidean distance between two sets of finger-codes over homomorphic encryption. In contrast to traditional matching methods, such an approach is more privacy aware because the matching process does not expose the biometric sample being matched to the matching agent. Our research draws a few ideas from them including the use of a reference point and sector segmentation to represent fingerprints. However, we notice that any approach applying homomorphic encryption for matching comparison is restricted by using only the matching functions that can be privately computed via the multiplicative or additive properties of the specific homomorphic encryption. In our research, homomorphic encryption is applied in the (SIPPA-2.0) protocol layer which allows not only private data comparison, but source data reconstruction; thus providing additional flexibility on the choice of matching criteria and functions.

Clancy et al. [11] pioneered the use of fingerprints to generate cryptographic Keys. Their essential approach is to use Minutia points as the feature set and Reed-Solomon error correction to handle noise inherent in biometric samples. To describe their approach in their own words:

"Fingerprint minutiae coordinates m_i *are encoded as elements in a Finite Field F and the secret Key is encoded in a polynomial f(x) over F[x]. The polynomial is evaluated at the minutiae locations, and the pairs* $(m_i; f(m_i))$ *are stored along with random* $(c_i; d_i)$ *chaff points such that* $d_i != f(c_i)$*. Given a matching Fingerprint, a valid user can separate out enough true points from the chaff points to reconstruct f(x), and hence the original secret Key."*

Clancy et al. in their pioneering work achieved a *Key* size of 69 bits, with an EER (Equal Error Rate) at about 30%. Our research also draws inspiration from Clancy et al. in their overall approach of melding a proven biometric modality such as fingerprints with the field of cryptography; bringing forth desirable properties such as non-repudiation into the field of cryptography. In our research we are able to show a much improved result for a similar approach that can accommodate an arbitrary large key size and a much better EER. We will show one such result in the experimental study section of this book chapter.

Recently, non-minutiae methods as features for fingerprint are proposed in addition to the minutiae methods [39, 40], these methods are viable alternatives to consider in trying to improve the performance of many cryptographic key generation protocols.In this research we have primarily focused on minutiae features as data points for our cryptographic key generation protocol. On the other hand, Lalithamani et al. [12] proposed a method utilizing cancellable biometrics to transform a fingerprint into a revocable form; this revocable form is then transformed into an irrevocable viable cryptographic *Key*. The essence of this approach relies on image processing to transform noisy biometric samples into standard representations, from which a cryptographic *Key* is derived. Unfortunately, no practical experimental data was presented utilizing this approach. Nonetheless, this is another indication on the potential interest in combining biometrics with cryptography.

Various secure computation protocols based on cryptographic approach have been developed in the past for privacy preserving data comparison [13] [14]. One main thrust is to rely on Oblivious Transfer (OT) protocols [15] for private joint table lookup. Privacy protection is achieved by transmitting encrypted versions of the entire table using pre-computed public keys. However, even with significant progress in reducing the computational complexity of OT protocols, the relative enormity of the table size coupled with the complexity of encrypting large amounts of data with public key cryptography makes OT based Secure Computation protocols impractical for certain viable privacy preserving applications such as biometrics.

Recent advances in cryptography has led to various cryptosystems that preserve certain operations such as addition [16], multiplication [17], and XOR [18], in the encrypted domain. Such cryptosystems are classified into either Partially Homomorphic (PH) [19] [20] or Fully Homomorphic (FH) [21]. A PH cryptosystem does not allow the evaluation of both addition and multiplication in encrypted domain; and a FH allows the evaluation of unlimited addition and multiplication operations in encrypted domain. A FH cryptosystem would allow [13] the secure evaluation of any function in the encrypted domain where the privacy of the original data is guaranteed if knowing only the function output is not a privacy concern. However the only known semantically secure FH cryptosystems are extremely inefficient for practical use. To put it in context, recently, one of the leading inventors of such a FH cryptosystem Craig Gentry, states that performing a single search query utilizing the best available FH cryptosystem would increase computational complexity by about a trillion [22].

To balance the practical usability and the complexity inherited in a fully homomorphic cryptosytem, we develop SIPPA-2.0 that allows parallelization while relying only on cryptosystems that are Partially Homomorphic for cryptographic key generation from biometric data

http://dx.doi.org/10.5772/51372

such as fingerprints. Over the past few years, privacy preserving protocols based on homomorphic cryptosystems have also been developed; e.g., protocol for secure scalar/dot product between vectors [23], protocol for secure Euclidean distance derivation between vectors [24], secure Hamming distance protocol [25], secure evaluation of linear systems [26], and k-mean clustering [27]. These protocols are practically usable in biometric applications and utilize semantically secure Partially Homomorphic (PH) cryptosystems to achieve metric specific privacy preserving biometric authentication. SIPPA-2.0 takes the current state-of-the-art one step further by facilitating not just private data comparison which can be used for privacy preserving biometric authentication, but also private data reconstruction for biometric information retrieval.

3. Theoretical Foundation of SIPPA-2.0

The theoretical foundation of SIPPA-2.0 is built upon two main theorems. We first summarize the important findings of the theorems, and then present the rigorous formulation and proof, followed by the use of the theorems to realize the SIPPA-2.0 protocol for private reconstruction of the server source data by the client.

Let P1 and P2 be the SIPPA-2.0 server and client respectively. Let **de** and **dv** be the column vector representing private data of P1 and P2 respectively. Let (λ_{De} $\mathbf{v}_{\mathbf{de}}$) and (λ_{Dv} $\mathbf{v}_{\mathbf{dv}}$) be the 2-tuples of the most significant Eigen value and the corresponding unity normalized Eigen vector of the matrices $\mathbf{de} \bullet \mathbf{de}^{T}$ and $\mathbf{dv} \bullet \mathbf{dv}^{T}$ respectively.

If the deviation between the most significant eigenvector of $\mathbf{de} \bullet \mathbf{de}^{T}$ and $\mathbf{dv} \bullet \mathbf{dv}^{T}$ (similarity score) correlates to the distribution of the instances of **de** and **dv** as classified by being from the same or different biometric source, this provides a basis for a good threshold function. SIPPA-2.0 can then be utilized to provide secure and private derivation on this deviation, without each party revealing their private biometric data to each other.

We will show in theorem 1 below that there is an algebraic relationship between the symmetric matrix representation of the client and the server data in the Eigen vector space; and the algebraic relationship guarantees the existence of a bisector vector that allows each party to use it to determine whether the client and server data are sufficiently similar. Second, the source data can be perfectly reconstructed by the client if a carefully scaled Eigen value of the symmetric matrix representation of the source data is given. In other words, this scaled Eigen value serves as the helper data --- and the only data --- that the server needs to share; thus the privacy of the server side source data is preserved.

Theorem 1: Consider $(\mathbf{de} \bullet \mathbf{de}^{T} + \mathbf{dv} \bullet \mathbf{dv}^{T})x = \lambda_{de}\ \mathbf{v}_{\mathbf{de}} + \lambda_{dv}\ \mathbf{v}_{\mathbf{dv}}$, the solution x = v satisfying $(\mathbf{de} \bullet \mathbf{de}^{T} + \mathbf{dv} \bullet \mathbf{dv}^{T})\mathbf{v} = \lambda_{de}\ \mathbf{v}_{\mathbf{de}} + \lambda_{dv}\ \mathbf{v}_{\mathbf{dv}}$ has a unity scalar projection onto the unity normalized $\mathbf{v}_{\mathbf{de}}$ and $\mathbf{v}_{\mathbf{dv}}$, and is a bisector for the interior angle between $\mathbf{v}_{\mathbf{de}}$ and $\mathbf{v}_{\mathbf{dv}}$.

Proof: By the definition of Eigen vectors and values, $\mathbf{de}\bullet\mathbf{de}^T\bullet\mathbf{v_{de}} = \lambda_{de}\,\mathbf{v_{de}}$. Since $\mathbf{de}^T\bullet\mathbf{v_{de}}$ is a scalar, **de** and $\mathbf{v_{de}}$ has the same directionality. Furthermore, $\mathbf{de}/|\mathbf{de}| = \mathbf{v_{de}}$ because $\mathbf{v_{de}}$ is a unity normalized vector. Similarly, $\mathbf{dv}/|\mathbf{dv}| = \mathbf{v_{dv}}$, and the following results can be established:

$$\mathbf{de}/|\mathbf{de}| = \mathbf{v_{de}}\mathbf{de}\bullet\mathbf{de}^T\bullet\mathbf{v_{de}} = \lambda_{de}\mathbf{v_{de}} \;\blacksquare\; (\mathbf{de}/|\mathbf{de}|)\bullet(|\mathbf{de}|\bullet\mathbf{de}^T\bullet\mathbf{v_{de}}) = (\mathbf{v_{de}})(\lambda_{de}) \;\blacksquare\; \lambda_{de} = \mathbf{de}^T\bullet\mathbf{de}$$

$$\mathbf{dv}/|\mathbf{dv}| = \mathbf{v_{dv}}\mathbf{dv}\bullet\mathbf{dv}^T\bullet\mathbf{v_{dv}} = \lambda_{dv}\mathbf{v_{dv}} \;\blacksquare\; (\mathbf{dv}/|\mathbf{dv}|)\bullet(|\mathbf{dv}|\bullet\mathbf{dv}^T\bullet\mathbf{v_{dv}}) = (\mathbf{v_{dv}})(\lambda_{dv}) \;\blacksquare\; \lambda_{dv} = \mathbf{dv}^T\bullet\mathbf{dv}$$

To prove theorem 1, we need to prove that (1) **v** has a unity scalar projection onto the unity normalized **de** and **dv**, and (2) **v** is a bisector.

To prove **v** has a unity scalar projection onto the unity normalized $\mathbf{v_{de}}$ and $\mathbf{v_{dv}}$, it is sufficient to show $\mathbf{v_{de}}\bullet\,\mathbf{v} = \mathbf{v_{dv}}\bullet\mathbf{v} = 1$, or $(\mathbf{de}/|\mathbf{de}|)\bullet\mathbf{v} = (\mathbf{dv}/|\mathbf{dv}|)\bullet\mathbf{v} = 1$. Since **v** is a solution to $(\mathbf{de}\bullet\mathbf{de}^T + \mathbf{dv}\bullet\mathbf{dv}^T)\mathbf{x} = \lambda_{de}\,\mathbf{v_{de}} + \lambda_{dv}\,\mathbf{v_{dv}}$, we re-write the RHS and the LHS as below:

$$\text{LHS} = (\mathbf{de}\bullet\mathbf{de}^T + \mathbf{dv}\bullet\mathbf{dv}^T)\mathbf{v} = \mathbf{de}\bullet(\mathbf{de}^T\bullet\mathbf{v}) + \mathbf{dv}\bullet(\mathbf{dv}^T\bullet\mathbf{v})$$
$$= |\mathbf{de}|\bullet\mathbf{de}\bullet(\mathbf{de}^T\bullet\mathbf{v}/|\mathbf{de}|) + |\mathbf{dv}|\bullet\mathbf{dv}\bullet(\mathbf{dv}^T\bullet\mathbf{v}/|\mathbf{dv}|)$$

$$\text{RHS} = \lambda_{de}\,\mathbf{v_{de}} + \lambda_{dv}\,\mathbf{v_{dv}} = \mathbf{de}^T\bullet\mathbf{de}\bullet(\mathbf{de}/|\mathbf{de}|) + \mathbf{dv}^T\bullet\mathbf{dv}\bullet(\mathbf{dv}/|\mathbf{dv}|)$$

$$= |\mathbf{de}|\bullet\mathbf{de} + |\mathbf{dv}|\bullet\mathbf{dv} \text{ because } \mathbf{de}^T\bullet\mathbf{de} = |\mathbf{de}|^2 \text{ and } \mathbf{dv}^T\bullet\mathbf{dv} = |\mathbf{dv}|^2$$

Comparing the terms on the RHS and LHS, when de and dv are linearly independent, $\mathbf{de}^T\bullet\mathbf{v}/|\mathbf{de}| = 1 \;\blacksquare\; (\mathbf{de}/|\mathbf{de}|)\bullet\mathbf{v} = \mathbf{v_{de}}\bullet\mathbf{v} = 1$ and $\mathbf{dv}^T\bullet\mathbf{v}/|\mathbf{dv}| = 1 \;\blacksquare\; (\mathbf{dv}/|\mathbf{dv}|)\bullet\mathbf{v} = \mathbf{v_{dv}}\bullet\mathbf{v} = 1$. Therefore, v has a unity scalar projection onto the unity normalized **de** and **dv**. This completes the proof for (1).

The scalar projection of **v** onto $\mathbf{v_{de}}$ is one, and so as the scalar projection of **v** onto $\mathbf{v_{dv}}$. By the theorem of bisector, v is the bisector of the interior angle of $\mathbf{v_{de}}$ and $\mathbf{v_{dv}}$. This completes the proof for (2).

Theorem 2: Consider $(\mathbf{de}\bullet\mathbf{de}^T + \mathbf{dv}\bullet\mathbf{dv}^T)\mathbf{x} = \lambda_{de}\,\mathbf{v_{de}} + \lambda_{dv}\,\mathbf{v_{dv}}$, **de** can be efficiently reconstructed − with an accuracy proportional to the closeness between $\mathbf{v_{de}}$ and $\mathbf{v_{dv}}$ − by a party with **dv**, $\lambda_{dv,,}$ and $\mathbf{v_{dv}}$ when (i) the interior angle between $\mathbf{v_{de}}$ and $\mathbf{v_{dv}}$ is less than 90 degree and (ii) the party is given **x** and $\lambda_{de}/\,\mathbf{de}^T\bullet\mathbf{x}$. Specifically, $\mathbf{de} = (\mathbf{est\_v}_{\mathbf{de}}/|\,\mathbf{est\_v}_{\mathbf{de}}\,|)(\lambda_{de}/\,\mathbf{de}^T\,\mathbf{x})$; where

$$\mathbf{est\_v_{de}} = \mathbf{v_{dv}} + [|\mathbf{v_{dv}}|\bullet\tan(2\cos^{-1}(\mathbf{v_{dv}}\bullet\mathbf{x}/(|\mathbf{v}_{\mathbf{dv}}|\bullet|\mathbf{x}|)))]\bullet[(\mathbf{x}-\mathbf{v}_{\mathbf{dv}})/|\mathbf{x}-\mathbf{v_{dv}}|]$$

Proof: Let $\mathbf{x} = \mathbf{v_{de}} + \mathbf{e1}$ and $\mathbf{x} = \mathbf{v_{dv}} + \mathbf{e2}$. We can derive the length of **te** (as shown in Fig. 2), which is a vector with the same directionality as that of the vector **e2** when the interior angle between $\mathbf{v_{de}}$ and $\mathbf{v_{dv}}$ is less than 90 degree. Specifically, $\mathbf{v_{dv}}$ and **e2** are orthogonal (i.e., they are perpendicular of each other). The length of $\mathbf{te}=|\mathbf{te}|=[|\mathbf{v}_{\mathbf{dv}}|\bullet\tan(2\cos^{-1}(\mathbf{v_{dv}}\bullet\mathbf{x}/(|\mathbf{v_{dv}}|\bullet|\mathbf{x}|)))]$ because $\mathbf{e1} = \mathbf{e2}$ and the angle between $\mathbf{v_{dv}}$ and $(\mathbf{v_{dv}} + \mathbf{te})$ is twice the angle between $\mathbf{v_{dv}}$ and **x** (theorem 1). Therefore, $\mathbf{te}=|\mathbf{te}|\bullet[\mathbf{e2}/|\mathbf{e2}|]=[|\mathbf{v_{dv}}|\bullet\tan(2\cos^{-1}(\mathbf{v_{dv}}\bullet\mathbf{x}/(|\mathbf{v_{dv}}|\bullet|\mathbf{x}|)))]\bullet[\mathbf{e2}/|\mathbf{e2}|]$. Since $\mathbf{v_{dv}} + \mathbf{te}$ $(=\mathbf{est\_v_{de}})$ produces a vector with the same directionality as $\mathbf{v_{de}}$, and $\mathbf{v_{de}}$ is a unity normalized vector, we can conveniently derive $\mathbf{v_{de}}$ by normalizing $\mathbf{est\_v_{de}}$; i.e., $\mathbf{v_{de}} = \mathbf{est\_v_{de}}/|\,\mathbf{est\_v_{de}}\,|$. Finally, since $\mathbf{de}\bullet\mathbf{de}^T\mathbf{x} \approx \lambda_{de}\,\mathbf{v_{de}}$, we can derive **de** from $(\lambda_{de}/\,\mathbf{de}^T\bullet\mathbf{x})\bullet\mathbf{v_{de}}$ or $(\lambda_{de}/\,\mathbf{de}^T\bullet\mathbf{x})\bullet(\,\mathbf{est\_v_{de}}/|\mathbf{est\_v_{de}}\,|)$ with an approximation error proportional to the closeness between $\mathbf{v_{de}}$ and $\mathbf{v_{dv}}$. Q.E.D.

http://dx.doi.org/10.5772/51372

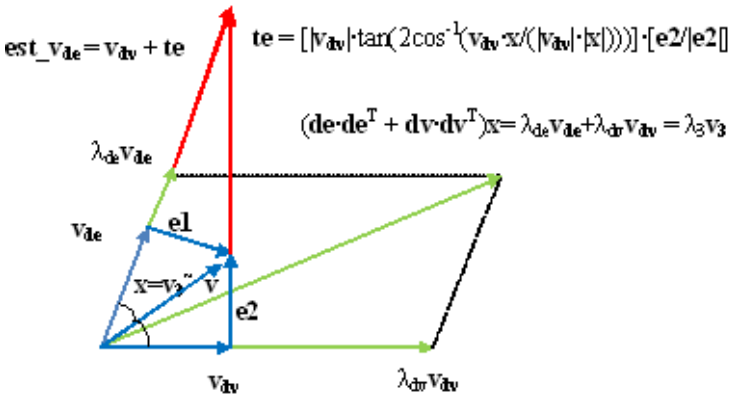

Figure 2. Geometric relationship between x and eigenvector of data.

3.1. Secure Computation of Angular Deviation

In SIPPA-2.0, private reconstruction of the server source data by a client relies on the helper data. The server provides the helper data only if the target data of the client is sufficiently close to the source data. But how could the server determine the closeness between the target data and the source data while the client can keep the target data private? As shown in figure3, v_{de}, v_{dv} and x all converge to each other when **de** and **dv** converge to each other. In addition, the matrices $\mathbf{de} \bullet \mathbf{de}^T$ and $\mathbf{dv} \bullet \mathbf{dv}^T$ can be thought of as the mapping functions for the eigenvectors and x. The difference between **de** and **dv** proportionality affects the difference in the mapping functions, which subsequently introduces an angular deviation on the angle between the unity normalized Eigen vectors as well as the magnitude deviation as measured by the Euclidean distance between the two Eigen vectors scaled by the corresponding Eigen values. Therefore, angular deviation and magnitude deviation between the client and server Eigen vectors can be used to obtain the information about the closeness between the target and the source data.

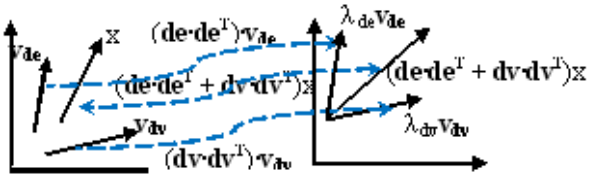

Figure 3. Relationship between source and target data in Eigen space.

In SIPPA-2.0, if angular deviation is used as the metric to determine closeness, then both the server and the client can privately and precisely determine whether the data of the other party is sufficiently similar without revealing one's own private data. Recall from theorem 2 the angular deviation can be derived from $2\cos^{-1}(\mathbf{v_{dv}} \bullet \mathbf{x}/(|\mathbf{v_{dv}}| \bullet |\mathbf{x}|))$ or $2\cos^{-1}(\mathbf{v_{de}} \bullet \mathbf{x}/(|\mathbf{v_{de}}| \bullet |\mathbf{x}|))$. If x is known, each party with one's own Eigen vector can derive the angular deviation.

Therefore, a key aspect to realize SIPPA-2.0 is a 2-party secure computation protocol through which the client and the server can collaborate to solve for x in the algebraic system $(\mathbf{de}\bullet\mathbf{de}^{T} + \mathbf{dv}\bullet\mathbf{dv}^{T})\mathbf{x} = \lambda_{de}\,\mathbf{v_{de}} + \lambda_{dv}\,\mathbf{v_{dv}}$ while each party will keep one's data, Eigen value and Eigen vector private. Furthermore, the security and privacy of SIPPA-2.0 will defer to the security and privacy of the 2-party secure computation protocol for solving the algebraic system. In this research we show one such protocol, referred to as SLSSP (Secure Linear System Solution Protocol), that we developed. As formulated in section 3, SLSSP is a novel and general two-party secure computation protocol to solve for x in (A1+ A2)x = (b1 + b2) without each party revealing their private data (Ai, bi) to each other.

There are two noteworthy points about SLSSP. First SLSSP is readily available to realize SIPPA-2.0 with (A1, b1) and (A2, b2) being $(\mathbf{de}\bullet\mathbf{de}^{T}, \lambda_{de}\,\mathbf{v_{de}})$ and $(\mathbf{dv}\bullet\mathbf{dv}^{T}, \lambda_{dv}\,\mathbf{v_{dv}})$ respectively. Second, SLSSP has the same problem formulation as that of PPCSC (Privacy Preserving Collaborative Scientific Computation). Our focus on SLSSP is to also achieve desirable security properties so that SLSSP is secure and private under realistic and reasonable assumptions discussed in section 3.

4. Procotol design for SLSSP and SIPPA

4.1. SIPPA2.0 Protocol

There are three major aspects of SIPPA-2.0: (1) derivation of the eigenvalues and the corresponding unity normalized eigenvectors of the symmetric matrix representation of the data; (2) derivation of a vector x which provides for the two parties to determine the deviation between their respective eigenvectors, which is formulated as a two-party secure computation SLSSP [28]; and if required: (3) reconstruction of the source data based on *helper data* composed of a scalar eigenvalue combined with a scalar derived from the vector product between the transpose of the linearized source data vector and the vector x. The steps for the SIPPA2.0 protocol are detailed below using the notation introduced in the previous section:

Step 1: Derive, by the respective party, the most significant eigenvalue and its corresponding unity-normalized eigenvector of $\mathbf{de}\bullet\mathbf{de}^{T}$ and $\mathbf{dv}\bullet\mathbf{dv}^{T}$. This step yields $(\lambda_{de}\,\mathbf{v_{de}})$ for SIPPA2.0 server and $(\lambda_{dv}\,\mathbf{v_{dv}})$ for SIPPA2.0 client.

Step 2: Compute $\mathbf{x}$ such that $(\mathbf{de}\bullet\mathbf{de}^{T} + \mathbf{dv}\bullet\mathbf{dv}^{T})\mathbf{x} = \lambda_{de}\,\mathbf{v_{de}} + \lambda_{dv}\,\mathbf{v_{dv}}$ utilizing SLSSP. The vector x is known to both parties following SLSSP. The details on SLSSP will be presented later in this section.

Step 3: The party that wishes to determine the deviation between its eigenvector and the other party's eigenvector can do so utilizing $\boldsymbol{x}$ (derived in step 2). Suppose that the party with $\mathbf{v_{de}}$ wishes to determine the angular deviation between $\mathbf{v_{de}}$ and $\mathbf{v_{dv}}$, this can be done by obtaining the angle between $\mathbf{v_{de}}$ and $\boldsymbol{x}$. i.e. $\cos^{-1}(\mathbf{v_{de}}\bullet\mathbf{x}/(|\mathbf{v_{de}}|\bullet|\mathbf{x}|))$. The angular deviation between $\mathbf{v_{de}}$ and $\mathbf{v_{dv}}$ is then

$2\cos^{-1}(\mathbf{v_{de}}\bullet\mathbf{x}/(|\mathbf{v_{de}}|\bullet|\mathbf{x}|))$.

http://dx.doi.org/10.5772/51372

Step 4: If **de** and **dv** are sufficiently similar as determined by either the angular distance or the Euclidean distance between vectors $\mathbf{v_{de}}$ and $\mathbf{v_{dv}}$ as measured by some pre-defined threshold, proceed to send the following helper data: $\lambda_{de}/\mathbf{de}^{T} \bullet x$.

Remark: We can also only send $(\lambda_{de})^{0.5}$ as the helper data to allow perfect reconstruction of $\mathbf{de}= (\mathbf{est\_v_{de}} / |\mathbf{est\_v_{de}}|)(\lambda_{de})^{0.5}$ because (1) $\lambda_{de}=\mathbf{de}^{T} \bullet \mathbf{de} = |\mathbf{de}|^2$ (from Theorem 1), (2) $\mathbf{de}/|\mathbf{de}| = \mathbf{v_{de}}$ or $\mathbf{de}= |\mathbf{de}| \bullet \mathbf{v_{de,}}$ (from Theorem 1) and (3)$\mathbf{est\_v_{de}} / |\mathbf{est\_v_{de}}| = \mathbf{v_{de}}$ (from Theorem 2) if we are to realize unconditional source data reconstruction.

Step 5: Derive estimated $\mathbf{v_{de}}$**- est_**$\mathbf{v_{de}}$ as stated in theorem 2, and then derive $\mathbf{de} = (\mathbf{est\_v_{de}} / |\mathbf{est\_v_{de}}|)(\lambda_{de}/\mathbf{de}^{T}x)$.

SIPPA-2.0 relies on the protocol SLSSP to solve for x in step 2. To provide semantic security, Pailler encryption is adopted in the protocol design for SLSSP. Pailler encryption is operated over Integer domain on scalar values. Yet SIPPA-2.0 deals with Real Number domain and matrix operations. Therefore, prior to presenting the details on the protocol for SLSSP, we will first discuss two related issues: (i) homomorphic matrix addition and multiplication with encrypted matrices, and (ii) the fixed point representation of real numbers.

4.2. Homomorphic Matrix addition and Multiplication with Encrypted Matrices.

All matrix operations described in this section require knowledge of only the Pailler public-key pair(g, n). Encrypted matrices and vectors are denoted$[[M]]^{P(g,n)}$, $[[v]]^{P(g,n)}$respectively, where each element of the matrix or vector is an element encrypted utilizing the Paillier public-key pair (g, n). Specifically, the decryption of any element $[[M]]_{i,j}^{P(g,n)}$ equals $M_{i,j}$ i.e. $M_{i,j} = PD\left([[M]]_{i,j}^{P(g,n)}\right)$. The operator "[+]" denotes homomorphic addition of matrices or vectors; whereas the operator "[X]" represents multiplication of matrices, where one of the two matrices are encrypted.

4.2.1. Paillier Cryptosystem:

The Paillier encryption scheme is a probabilistic, asymmetric, public-key cryptosystem whose security is based on the hypothetical intractability of the Decisional Composite Residuosity Assumption (DCRA). The Paillier encryption function PE(m, r), a bijection $\left(Z_n x Z_n^* \rightarrow Z_{n^2}^*\right)$ encrypts a message mby raising a basis gto the power m, then multiplying g^m with a random $r^n$and reducing the product $\left(g^m.r^n\right)$ modulo $n^2$where nis the public modulus. An important consequence of this approach is that the Paillier Cryptosystem is additively homomorphic, specifically the following properties hold:

1. $\left(\left(PE(m_1, r_1) . PE(m_2, r_2)\right) \bmod n^2\right) = PE(m_1 + m_2, \; r_1 . r_2)$
2. $\left(\left(PE(m_1, r_1) . g^{m2}\right) \bmod n^2\right) = PE(m_1 + m_2, \; r_1)$
3. $\left(\left(PE(m_1, r_1)^{m2}\right) \bmod n^2\right) = PE(m_1 . m_2, \; r_1)$

Paillier Key Generation:

1. Choose a modulus $n = p \cdot q$. The modulus is chosen in accordance with the RSAES-OAEP [2] specification, where nhas the properties of a well-chosen RSA modulus.
2. Choose a random $g \in Z_{n^2}^*$ensure that the order of gin modulo $n^2$is a multiple ofn, if not choose anotherg until the requirement is met.
3. Compute $\lambda = \lambda(n) = \text{lcm}((p - 1), (q - 1))$, where $\lambda(n)$ is the Carmichael function.
4. Let$L(u) = \frac{(u - 1)}{n}$, compute $\mu = (L(g^{\lambda} \bmod n^2))^{-1} \bmod n$.
5. The Paillier public-key pair is (g, n).
6. The Paillier private-key pair is (λ, μ).

The Paillier Encryption function PE(m, r):

Given a Paillier public-key pair, choose a message to be encrypted $m \in Z_n$, and a randomr chosen uniformly fromZ_n^*, then the Paillier encryption function is defined as $PE(m, r) = (g^m \cdot r^n) \bmod n^2$.PE(m, r)is a bijection $(Z_n x Z_n^* \rightarrow Z_{n^2}^*)$which produces a ciphertext $(c \in Z_{n^2}^*)$.

The Paillier decryption function PD(c):

Given a Paillier public-key, private –key pair and a Paillier ciphertext $c \in Z_{n^2}^*$, then the Paillier decryption function is defined as:

$$PD(c) = (L\ (c^{\lambda} mod\ \ n^2).\mu)\ \ mod\ \ n \tag{1}$$

4.2.2. Homomorphic addition of two matrices:

1. Given two encrypted *mXn*matrices $[[A]]^{P(g,n)}$ and$[[B]]^{P(g,n)}$, their homomorphic addition $([[A]]^{P(g,n)}[+\ ][[B]]^{P(g,n)}) = [[A + B]]^{P(g,n)} = [[C]]^{P(g,n)})$ is carried out by multiplying each element of *A*with its matching element in *B*i.e.

$$[[C]]_{i,j}^{P(g,n)} = \left([[A]]_{i,j}^{P(g,n)} \cdot [[B]]_{i,j}^{P(g,n)}\right)\ \ mod\ \ n^2 \tag{2}$$

4.2.3. Multiplication with encrypted matrices.

1. Given an encrypted *mXb*matrix $[[A]]^{P(g,n)}$ and a *bXn*plain-text matrix*B*, $([[A]]^{P(g,n)}[X]B) = [[AB]]^{P(g,n)} = [[C]]^{P(g,n)}$, where $[[C]]^{P(g,n)}$ is an *mXn*matrix, which can be computed in the following manner:

$$[[C]]_{i,j}^{P(g,n)} = \left(\prod_{k=1}^{b} \left([[A]]_{i,k}^{P(g,n)}\right)^{B_{k,j}}\right)\ \ mod\ \ n^2 \tag{3}$$

http://dx.doi.org/10.5772/51372

2. Given an plain-text mXbmatrix A and an encrypted bXnmatrix $[[B]]^{P(g,n)}\left(A[X][[B]]^{P(g,n)}\right)=[[AB]]^{P(g,n)}=[[C]]^{P(g,n)}$, where $[[C]]^{P(g,n)}$ is an mXnmatrix, which can be computed in the following manner:

$$[[C]]_{i,j}^{P(g,n)}=\left(\prod_{k=1}^{b}\left([[B]]_{i,k}^{P(g,n)}\right)^{A_{k,j}}\right)\ mod\ n^2 \tag{4}$$

4.3. Fixed Point Representation.

The Paillier cryptosystem operates over a finite field Z_n, we extend the cryptosystem to operate over the reals utilizing a simple Fixed Point representation scheme. Let $s\in\mathsf{Z}$be some exponent of 10, then for every $r\in\mathsf{R}$, $r$is represented as$(\lfloor 10^s r\rfloor)\in\mathsf{Z}$. An approximation of r, can be obtained by$\tilde{r}=\frac{\lfloor 10^s r\rfloor}{10^s}\in\mathsf{R}$, specifically:

1. For any$r\in\mathsf{R}^+$, a Paillier ciphertext is obtained by $PE(\lfloor 10^s r\rfloor, x)$, where $x$is some random and$\tilde{r}=\frac{PD(PE(\lfloor 10^s r\rfloor, x))}{10^s}$.

2. For any$r\in\mathsf{R}^-$, a Paillier ciphertext is obtained by $PE(n+\lfloor 10^s r\rfloor, x)$, where $n$is the Paillier modulus and $\tilde{r}=\frac{PD(PE(n+\lfloor 10^s r\rfloor, x))\ -\ n}{10^s}$

It is to be noted that representing reals with a fixed point representation introduces errors due to truncation, which is directly proportional to the size of $s$chosen. The domain of the encryption function is also truncated from Z_n to$\left\{0,1,\ \ldots.\left\lfloor\frac{n-1}{10^s}\right\rfloor\right\}$, whereas extending the fixed point scheme to include negative reals further reduces the encryption domain to $\left\{0,1,\ \ldots.\left\lfloor\frac{n-1}{2*10^s}\right\rfloor\right\}$. Since division operations are not properly defined in Paillier, we hold off on downscaling operations in the encrypted domain. A record of the change in scale is kept after each operation in the encrypted domain; this record is utilized to obtain $\tilde{r}$ upon decryption of the result.

4.4. SLSSP protocol details

Step #	PARTY α Private Data: *m*X*m* Matrix A1, *m*X*1* vector b1	Step #	PARTY β Private Data: *m*X*m* Matrix A2, *m*X*1* vector b2
α1	Generate a random *m*X*m* matrix**P1**. Generate a random 1Xn vector $\mathbf{v^T}$. Obtain a Paillier Public Key and Private Key pair i.e. (*g*α,*n*α) and (λα,μα) respectively.	β1	Generate a random *m*X*m* matrix **P2**. Obtain a Paillier Public Key and Private Key pair i.e. (*g*β,*n*β) and (λβ,μβ) respectively.

α2	Compute and Send $([[A1]]^{P(g\alpha,n\alpha)})$and $([[b1]]^{P(g\alpha,n\alpha)})$ to PARTY β.	β2	Receive $([[A1]]^{P(g\alpha,n\alpha)})$ and $([[b1]]^{P(g\alpha,n\alpha)})$from PARTY α.
α3	Receive$([[A2]]^{P(g\beta,n\beta)})$ and $([[b2]]^{P(g\beta,n\beta)})$ from PARTY β.	β3	Compute and Send $([[A2]]^{P(g\beta,n\beta)})$ and $([[b2]]^{P(g\beta,n\beta)})$ to PARTY α.
α4	Compute and send to PARTY β: $([[P1(A1+A2)]]^{P(g\beta,n\beta)})$	β4	Receive and decrypt matrix obtained from step α4 to obtain a mXm matrix: $(P1(A1+A2))$
α5	Compute $([[(P1(b1+b2)v^T)]]^{P(g\beta,n\beta)})$	β5	Compute the Moore–Penrose Pseudoinverse of $((P1(A1+A2)))$ to obtain $R=((P1(A1+A2)))^{-1}$
α6	Send the following to party β: $[[Y]]^{P(g\beta,n\beta)}=([[((P1(b1+b2)v^T))]]^{P(g\beta,n\beta)})$	β6	Receive from PARTY α (Step α6), and decrypt to obtain $Y=P1(b1+b2)v^T$
α7	Send $([[[v^{-1}]]]^{P(g\alpha,n\alpha)})$to PARTY β, where v^{-1} is the conjugate transpose of$v$ divided by its magnitude squared.	β7	Receive $[[[v^{-1}]]]^{P(g\alpha,n\alpha)}$ from step α7 and compute $X1$, utilizing **Y.** Send $X1$ to PARTY α. $X1=[[\mathbf{R}^*(\mathbf{Y})]]^{P(g\beta,n\beta)}$
α8	Receive $X1$from partyβ, (step β7) and compute the solution $[[x]]^{P(g\beta,n\beta)}$ by homomorphically multiplying $v^{-1}$to $X1$	β8	Compute the solution $[[x]]^{P(g\alpha,n\alpha)}$ by homomorphically multiplying (**R*Y**)to $([[[v^{-1}]]]^{P(g\alpha,n\alpha)})$
α9	Send $[[x]]^{P(g\beta,n\beta)}$ to PARTY β	β9	Receive $[[x]]^{P(g\beta,n\beta)}$ from PARTY α and decrypt to Obtain the solution x
α10	Receive$[[x]]^{P(g\alpha,n\alpha)}$ from PARTY β and decrypt to Obtain the solution x	β10	Send $[[x]]^{P(g\alpha,n\alpha)}$ to PARTY α.
VERIFICATION PHASE:			
α10	Compute the mx1 vector using the information obtained in step α3: $c\boldsymbol{\alpha}=([[((A1+A2)x)-(b1+b2)]]^{P(g\beta,n\beta)})$	β10	Compute the mx1 vector using the information obtained in step β2: $c\boldsymbol{\beta}=([[((A1+A2)x)-(b1+b2)]]^{P(g\alpha,n\alpha)})$
α11	For each element in $\mathbf{c\alpha}_{i,1}$, utilizing the zero-knowledge proof protocol specified in the appendix, verify with party β that $\mathbf{c\alpha}_{i,1}$is an encryption of some element within a setγ, where the set γcontains a finite list of fixed point elements close to zero. For each verification request from PARTY β, require that one PARTY α's verification request is processed reciprocally. Abort at any $\mathbf{c\alpha}_{i,1}$ if PARTY β is unable to prove that the decryption of $\mathbf{c\alpha}_{i,1}\in\gamma$	β11	For each element in $\mathbf{c\beta}_{i,1}$, utilizing the zero-knowledge proof protocol specified in the appendix, verify with party α that $\mathbf{c\beta}_{i,1}$is an encryption of some element within a setγ, where the set γcontains a finite list of fixed point elements close to zero. For each verification request from PARTY α, require that one PARTY β's verification request is processed reciprocally. Abort at any $\mathbf{c\beta}_{i,1}$ ifPARTY α is unable to prove that the decryption of $\mathbf{c\beta}_{i,1}\in\gamma$

Table 1. SLSSP protocol details

http://dx.doi.org/10.5772/51372

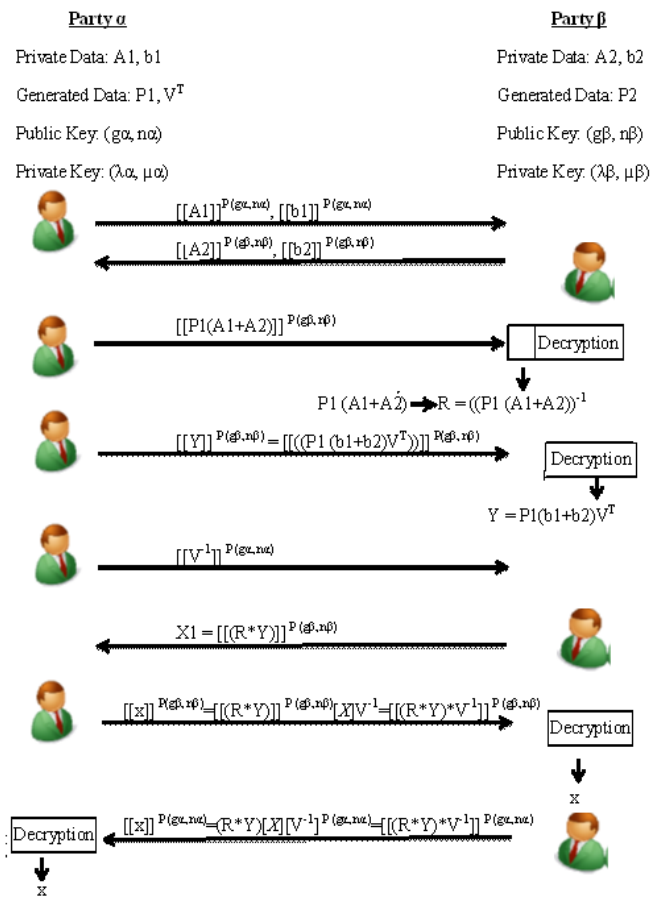

Figure 4. SLSSP protocol illustration.

5. Biometric Crypto Key Experiment & Application Showcase

5.1. Generation and retrieval of cryptographic keys from fingerprints

A modified finger-code approach is used to represent a fingerprint as an *attribute* vector. Several concentric circles are extended from a chosen core point; these concentric circles are further divided into sectors. Each of the sectors forms the boundary of one coding region representing the fingerprint. The Euclidean distance of the farthest and closest minutia points within each coding region in relation to the core point is normalized to a value between 0 and 255. These values make up the above described *attribute* vector. The length of the attribute vector; i.e., the number and area of each coding region is a variable chosen for optimal performance.

In this proposed method, generation of a cryptographic *Key* utilizable with strong symmetric encryption algorithms such as AES256 is straightforward. The *Key* generation phase essentially involves generation of a vector called the *k-vector,* whose length exactly equals the *attribute* vector. The *k-vector* consists of a series of random integers between 0 and 255. A fingerprint template attribute vector (T) is obtained to lock the *k-vector* (K); elementary addition of the two vectors (K + T) produces the locked vector (K_L). The unlocking process begins by deriving an error laden version of K. This is done by procuring a fingerprint sample attribute vector (S), and elementary subtraction (K_L - S) to obtain an error laden k-vector (K_E). K_E typically is not exactly identical to K. It cannot be directly utilized for the decryption of data encrypted with K. Measuring any physical object produces an error between measurements. Hence it is unlikely that matching minutia points in T and S will completely cancel each other during the locking and unlocking process. Our secure computation protocol, SIPPA is utilized to determine the deviation between K_E and K. If the party with K_E deems sufficient similar, it will send *helper data* (as described in the SIPPA section 6) which allows the party with K_E to derive K.

A perfect copy of K is retained at a 3rd party called the SIPPA server, and the SIPPA client engages with the SIPPA server utilizing K_E to obtain a reconstruction of K, if the server deems similarity. SIPPA also guarantees that no information that each of the parties possesses will leak to the other party in the case where T & S are dissimilar.

Figure 5 details the exact parameters utilized by us in our Key generation and retrieval algorithm, A Futronic FS88 [28] fingerprint scanner was utilized to capture fingerprints and a commercial algorithm (Neurotechnology's Verifinger v6.3 [29]) was used to extract minutia and core point coordinates from fingerprints.

5.2. Secure Computation of Angular Deviation

In order to assess SIPPA-2.0's usability in privacy preserving fingerprint biometrics, we conducted our performance analysis with the publicly availalble CASIA-FingerprintV5 database [38]. The database contains five diffrent digital impressions of each finger from 500 subjects. Each of these fingerprint images were converted to our custom fingercode format (as described in figure 8), which yields a 320x1 vector. NEUROtechnology's Verifinger SDK was utilized to orient and extract minutia, corepoints from the fingerprint images.For each session, a random key of size 320x1 of integers in the range 0 and 255 (i.e., 320x256=81920 bits) was generated (R), to which the fingercode vector (T) is added to obtain the vector (R +T). Party A possesses the key vector (R), whereas Party B possesses the vector (R+T). An impostor session is generated by subtracting the fingercode vector of a finger other than (T) from (R+T) to yield (IM), whreas a true user session is generated by subtracting the fingercode vector of a diffrent impression of (T) to yield (TU). SIPPA-2.0 is utilized as the matching function which compares [(R) vs (IM)] or [(R) vs (TU)] where the similarity score can either be the angle between the securely computed vector (X) (vector X is the final output of SIPPA-2.0) and (R), or the euclidean distance between the vector (X) and (R).

http://dx.doi.org/10.5772/51372

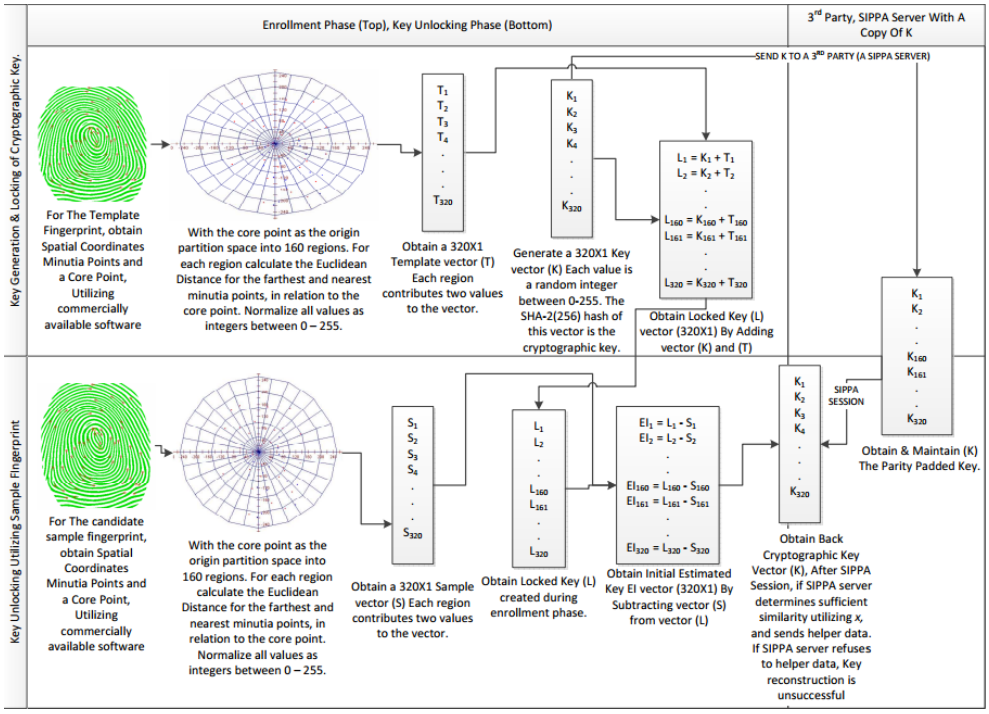

Figure 5. Cryptographic locking and unlocking illustration.

Because of the overheads involved in the matrix encryption and decryption process within our secure computation protocol (SLSSP), it becomes impractical to directly engage in SIPPA-2.0 with a 320x1 vector. An experiment was conducted to determine the performance of SIPPA-2.0 with various vector sizes, i.e., the 320x1 vector was split into 10x1, 16x1 and 20x1 vectors. Each of these splits yields a vector X_i, 100 such experiments were conducted (dual-core 3Ghz machine with 4Gb RAM) with actual fingerprint data at each split size(10,16,20). The average time to process the 320x1 vector at each of the three diffrent split parameters is reported in figure 6.

SIPPA Split Dimension	Average # of seconds
10x1	450
16x1	700
20x1	1120

Figure 6. SIPPA-2.0 complexity performance.

Due to SIPPA-2.0's complexity constraints, an extensive experiment directly utilizing SIPPA-2.0 was ruled out. Since SLSSP is theoretically guaranteed to produce a correct solution $\boldsymbol{x}$, we conducted our experiment by replacing SLSSP with a standard linear algebra package (EJML) to solve for $\boldsymbol{x}$in the algebraic system $(\mathbf{de} \bullet \mathbf{de}^{T} + \mathbf{dv} \bullet \mathbf{dv}^{T})\boldsymbol{x} = \lambda_{de}\, \mathbf{v}_{de} + \lambda_{dv}\, \mathbf{v}_{dv}$. The error between SLSSP's solution $\boldsymbol{x}$, and the solution produced EJML was determined experimentally to be always less than 4.77E-8 in over 6800 trials.

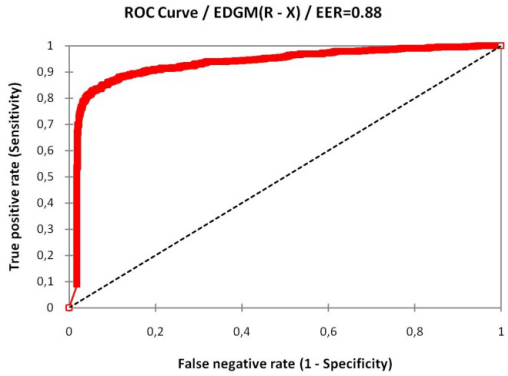

Figure 7. SIPPA-2.0 Performance (Split 10x1), ED(R vs. X).

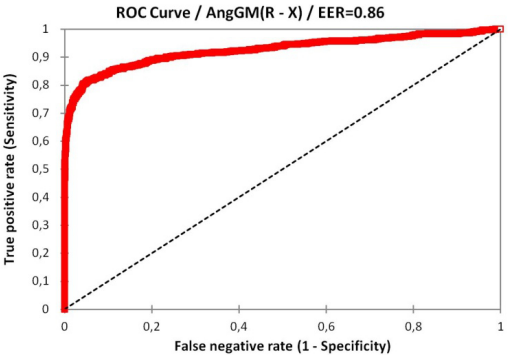

Figure 8. SIPPA-2.0 Performance (Split 10x1), Ang(R vs. X).

To assess SIPPA-2.0's ability to securely distinguish between True Users and Impostors we obtained over 20,000 unique True User sessions (20k-TU) and 20,000 unique Impostor sessions (20k-IM) from the extensive CASIA-FingerprintV5 database as described in the previous paragraphs. The ROC plot for (20k-TU) with (20k-IM) is provided in figure 7, a SIPPA-2.0 split dimension of 10X1 was utilized, and the 32 Euclidean distance (ED) scores (ED(R vs. $\boldsymbol{x}$)) per session was aggregated into one similarity score by obtaining their Geometric Mean. The ROC plot (20k-TU) with (20k-IM) where a SIPPA-2.0 split dimension of 10X1 was utilized, and the 32 dot product(Ang) scores(Ang(R vs. $\boldsymbol{x}$)) per session was aggregated into one similarity score by obtaining their Geometric Mean is provided in figure 8. Experiments were

http://dx.doi.org/10.5772/51372

also conducted at other split dimensions i.e. 15, 20; however they produced inferior or similar results with an exponentially increased processing times. No obvious methods (addition, scaling etc.) of combining the ED and Ang scores yielded better results suggesting possible dependency (between the two) that will deserve further research. Of the over 50,000 instances where helper data was sent to reconstruct the random key R, R was always reconstructed successfully except 6 instances out of the over 50,000 instances. We postulate that this is due to the sensitivities in finite precision arithmetic. In these 6 instances, the number of errors in the 320 integers constituting the random key R ranges between one and four. To safeguard against this remote possibility, R can be encoded in error correction codes, allowing for correcting R when the reconstruction fails due to a few errors.

5.3. Android app prototype for emergency health data retrieval

For proof of concept, we show a use case of SIPPA-2.0 for emergency health data management on an Android device. The diagram in figure 9 shows a scenario on an emergency responder using his/her Android phone to retrieve the emergency health data from the Android phone of an unconscious individual. In the enrolment process, the emergency health data of the individual is first encrypted and stored in the Shared Preference of his/her Android device. Furthermore, the fingerprint of the individual, together with a random noise vector generated by a third party, are used to encode the decryption key, and the encoded decryption key is also stored in the Shared Preference.

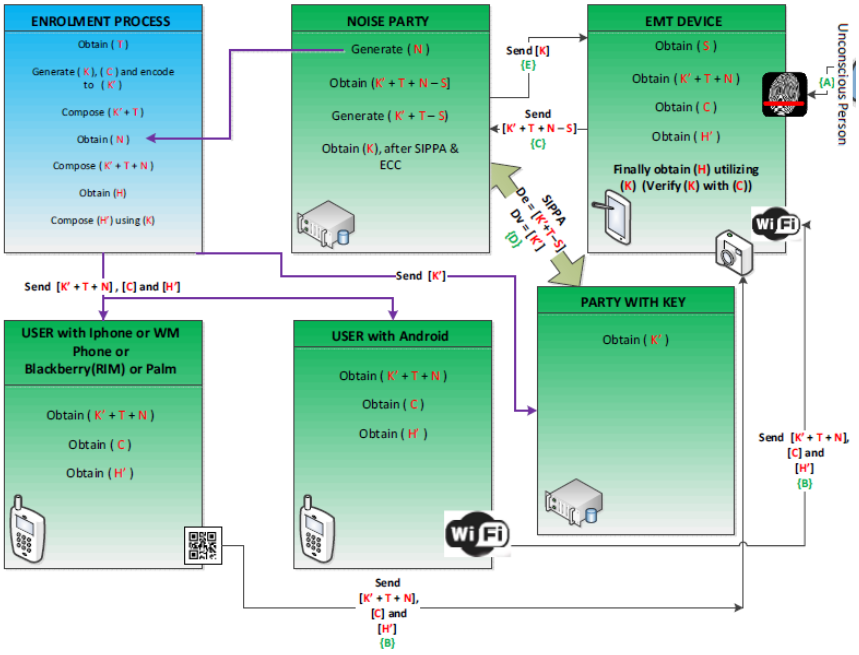

Figure 9. Use Case on applying SIPPA for emergency health data on an Andriod device.

LEGEND	Reed Solomon Encoding
(T) = User's Fingerprint Template. [320 X 1] Vector. Each element is an Integer, Range [0 to 255]. (S) = User's Fingerprint Sample. [320 X 1] Vector. Each element is an integer Range [0 to 255]. (H) = User's Health Information. (K) = [160 X 1] Vector. Each element is a random integer Range [0 to 255] (C) = Hash of (K) (K') = (K) encoded with Reed Solomon – 320 blocks 80 Correctable Symbols. (N) = [320 X1] Vector. Each element is a random integer Range [0 to 255] (H') = Encrypted Health Information using (K)	Let Be = Number of Blocks prior to Encoding. Let Ae = Number of Blocks after Encoding. Let Ce = Number of desired Correctable Blocks. By Berlekamp Massey Algorithm : Ae – Be = 2Ce Desired Ce = 80 (At EER SIPPA DIM 4) Therefore since Ae is fixed at 320, Be = 160. Be is K, whose hash can be utilized as a key for cryptography. Blue Arrows Represent Enrollment Phase. Sequence In Field, In case of an unconscious person {A}->{B}->{C}->{D}->{E}

Figure 9.1. Figure 9 Legend.

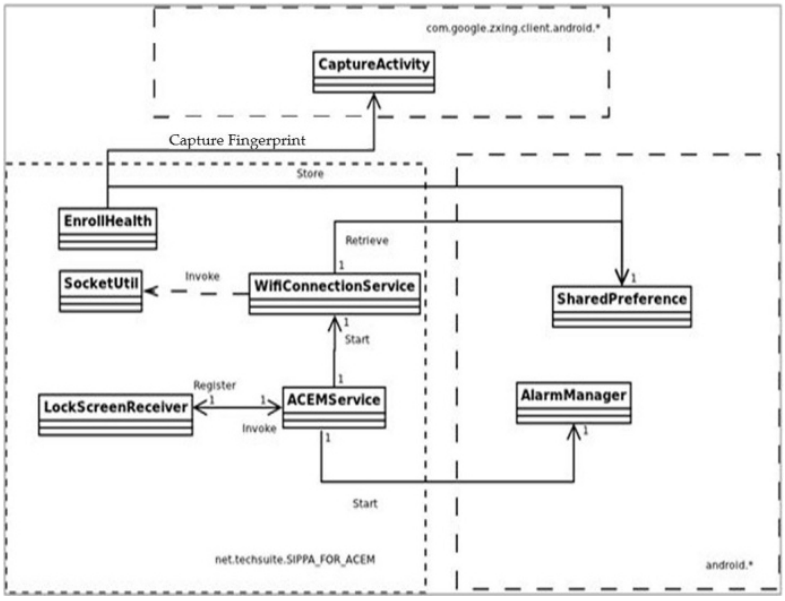

Figure 10. UML class diagram of the overall structural design.

http://dx.doi.org/10.5772/51372

During the retrieval process, the emergency responder will first establish a WiFi connection with the individual's Android device and retrieve the encrypted personal emergency health data and the encoded decryption key. The emergency responder will then use a built-in or wireless bluetooth fingerprint scanner to capture the indvidual's fingerpint for decoding the decryption key. The decoded fingerprint is then transmitted to the third party for noise removal to arrive at a sufficently close decrpytion key. The third party then acts as a SIPPA-2.0 client to interact with the SIPPA-2.0 server, and reconstruct the orginal decryption key based on the helper data provided by the SIPPA-2.0 server. The reconstrcted original key is sent back to the emergency responder for decrypting the personal emergency health data. The UML class diagram in figure 10 illustrates the overall structural design.

6. Conclusion

In this book chapter we present a novel mechanism for generating and retrieving cryptographic *keys* from fingerprints. Our approach brings together the worlds of biometrics and cryptography bytackling the problem of generating revocable repeatable keys (binary strings) from biometric channels which are by their nature noisy. Our scheme efficiently generates strong keys (256 bits) from fingerprint biometrics where there is no direct mapping of a key to the fingerprint from which it was generated. The entire key space is utilized where different keys can be produced from the same fingerprint for varying applications. Our scheme makes possible various new applications where there is a requirement for the strong coupling of an identity to cryptographic applications. For instance, data can be encrypted utilizing biometrics such as fingerprints where the key is linked to a person's physiology, oran individual's identity can be verified without the need for a central database of fingerprints.Such approaches allow for users to retain complete control of their data. It is not necessary for them to store their private data at a third party, thus alleviating privacy concerns. A use case utilizing our cryptographic key generation protocol is also presented; where health data is encrypted with a user's fingerprint and privately stored in a user's smartphone for secure retrieval during emergency scenarios. We also present our newly developed secure computation protocol called SIPPA-2.0, which is central to the cryptographic key generation protocol.Additionally, SIPPA-2.0 is shown to be provably correct, with an included security analysis proving SIPPA-2.0 to be secure under the semi-honest and semi-malicious models. Experiments revealing acceptable computational performance including ROC's indicating usability in practical scenarios are also presented.

Our future plans for this research include exploiting the parallelizability of our protocol to increase performance through a parallel architecture, and exploring recent non-minutia based fingerprint representation/comparison developments to improve segregation performance.

7. Appendix: Security Analysis

We approach the security analysis of SLSSP through the following steps. We first define the notion of perfect secrecy. We then analyze the cryptographic primitives employed by SLSSP and the protocol itself using the notion of perfect secrecy. To show that SLSSP is secure under semi-honest and semi-malicious models, we adopt a zero-knowledge proof protocol that can verify a given cipher text in SLSSP is indeed a Paillier encryption of its corresponding message; thus providing a mechanism to detect malicious behavior and to guarantee correctness with perfect secrecy.

7.1. General Assumptions

Throughout this section the security analysis is carried out in the Random Oracle model [36]; i.e., every party is assumed to have access to a Random Oracle where ideal hash functions and true random generators are available.

Perfect Secrecy:

Given a Cryptosystem, Plaintext (**M**), a Key (**K**) and Ciphertext (**C**); *Perfect Secrecy* [30] or *Information Theoretic Security* is an attribute assigned to Cryptosystems where knowledge of the Ciphertext provides no additional information about the original Plaintext independent of the computational power of an adversary. More specifically, a Cryptosystem has *Perfect Secrecy* if uncertainty within the Plaintext equals uncertainty within the Plaintext given the Ciphertext; i.e. utilizing Shannon Entropy, *Perfect Secrecy* can be defined as $H(M \mid C) = H(M)$.

7.2. Security Analysis of cryptographic primitives in SLSSP

Multiplication of candidate matrix with a random matrix.

At time $\mathbf{t_1}$, Alice wants to securely send an **nXn** matrix $A \in GL_n(Z_p)$(p is a prime)[31] to Bob over an authenticated channel [32], where Eve can intercept and store any messages over the channel. Eve is assumed to possess unbounded computational power. Alice wants to be assured that the matrix A is being sent with *Perfect Secrecy;* i.e., Eve should not be able to obtain any information about A due to knowledge of data sent over the channel.

Alice also has access at some time $\mathbf{t_0}$ ($\mathbf{t_0} < \mathbf{t_1}$) to a secure channel [32] with Bob. Alice shares a random nXn matrix Rchosen uniformly from $GL_n(Z_p)$ with Bob. At time $\mathbf{t_1}$, through the authenticated channel Alice sends $C = (A*R)$to Bob. To decrypt and obtain matrix$A$, Bob calculates$A = (C*R^{-1})$. Alice and Bob, discard $R$and do not use it to encrypt any other messages. Assuming that Eve only has knowledge of$C$, to prove *Perfect Secrecy* it will be sufficient to show that$H(A \mid C) = H(A)$. Let $(\mathbf{A} \in GL_n(Z_p))$, be the set of possible plaintexts, i.e, the sample space for$A$. Let $(\mathbf{C} \in GL_n(Z_p))$, be the set of possible ciphertexts, i.e. the sample space

http://dx.doi.org/10.5772/51372

for$C$. Let $(\mathbf{R} \in GL_n(Z_p))$, be the set of possible keys, i.e., the sample space for$R$. Note that $s = |GL_n(Z_p)| = \prod_{i=0}^{i=n-1} (p^n - p^i)$[33].

1. Let $(k \in \mathbf{R})$be a possible key, then $P_R(k) = 1/s$

2. Let $(l \in \mathbf{C})$be a possible ciphertext, and given the independence of $A$and Rthen:

$$P_C(l) = \sum_{\substack{m \in A,\ k \in R \\ l=(m*k)}} P_A(m) P_R(k) \tag{5}$$

3. Since

$$P_R(k) = 1/s: \tag{6}$$

$$P_C(l) = 1/s * \left(\sum_{\substack{m \in \mathbf{A}, k \in \mathcal{R} \\ l=(m*k)}} P_A(m) \right) \tag{7}$$

4. Because$l$, $m$uniquely determine k(given $l$and$m$ there exists exactly one $k$that satisfies the encryption equation $(l = (m*k))$, every $m \in \mathbf{A}$occurs exactly once in the above summation; therefore $P_C(l) = 1/s$

5. Since $P_C(l) = P_R(k)$ for all $l$and$k$, $H(C) = H(R)$

6. Since knowledge of (A, R) or(A, C) is sufficient to completely know (A, C, R), $H(A, R, C) = H(A, R) = H(A, C)$

7. Since $M$and $K$are independent, $H(A, R) = H(A) + H(R)$

8. By the chain rule for conditional entropy, $H(A, C) = H(A \mid C) + H(C)$

9. From (5),(7),(8) we obtain$H(A \mid C) = H(A)$; Proving *Perfect Secrecy.*

7.3. Security Analysis of the SLSSP protocol:

In an ideal Secure Computation protocol, where there are no assumptions made about maliciousness or computational capabilities of the parties involved; no participant learns anything about any other's private input(s) apart from any advantage obtained from the malleability of a legitimate output. In the semi-honest model[37], the parties involved are limited to PP [34] complexity, are assumed to follow the defined protocol honestly but are allowed to retain a transcript of all communication, which they can utilize to obtain information about other party's private input(s). In the semi-malicious model[37], parties are still restricted to PP complexity, whereas they are allowed to manipulate the secure computation

protocol, retain a transcript of all communication, which they can utilize to influence the outcome of the protocol and to obtain information about other party's private input(s). It is assumed in all these models that secure channels are utilized for inter-party communication.

The SLSSP protocol is secure under the semi-honest model, since through the entire protocol, each party only observes encrypted versions of each other's private input(s). The encryption primitives utilized in SLSSP have been shown in this section to be at least IND-CPA. Even though both parties have access to the encryption oracle, and SLSSP is a multi-step protocol, they do not gain any advantage by observing multiple versions of each other's encrypted private input(s). The SLSSP protocol can further be made secure under the semi-malicious model by utilizing the following Zero-Knowledge protocol (ZKP) which guarantees correctness of SLSSP's output.

Zero-Knowledge proof that a Paillier encrypted message encrypts a particular plaintext:

The Paillier encryption function $P(m, r)$is a Bijection$\left(Z_n x Z_n^* \rightarrow Z_{n^2}^*\right)$, which maps every plaintext $\left(m \in Z_n\right)$, random $\left(r \in Z_n^*\right)$ pair to a unique Ciphertext$\left(c \in Z_{n^2}^*\right)$. The prover **P** could therefore send $r$toa verifier **V**, which would be sufficient to establish that$c$ encrypts a particular message$m$. Revealing$r$ may however jeopardize certain desirable qualities of the Paillier Cryptosystemincluding its indistinguishability under a chosen plain text attack (IND-CPA). However, since Paillier encryption is semantically secure (IND-CPA), the verifier V will not be able to tell whether a given cipher text c is indeed the encryption of a message m. To solve this problem without leaking additional information, we adapt from [35] an honest verifier Zero-Knowledge protocol which can be formulated as below:

The honest Verifier **V** knows **m** and **possesses** a Paillier Ciphertext$c = g^{m'} . r^n \bmod n^2$, and knows only the Paillier public-key pair (g, n). **V**, not knowing ***r*** or the corresponding private-key pair$(\boldsymbol{\lambda}, \boldsymbol{\mu})$, would like to verify that$c$ is the encryption of a particular message$m$, where the prover **P** possesses (g, n) and random$r$.

1. **V** homomorphically subtracts $m$from$c$; i.e., computes z as an encryption of $(\boldsymbol{m'\text{-}m}) \leftrightarrow z = \left(c.g^{-m}\right) \bmod n^2$). Note that if$m' = m$, then $z$is an encryption of Zero; i.e.,$z = r^n \bmod n^2$

Remark: This step may require using the fixed point representation discussed in section 6.3 to compute$z = \left(c.g^{-m}\right) \bmod n^2$).

2. **P** chooses a random$q \in Z_n^*$ and computes an encryption of zero:$h = g^0 . q^n \bmod n^2$. **P** sends h to **V**.

3. **V** chooses a random e $(e \neq an, \quad a \in Z) \in Z$and sends $e$to **P**. (This step stages a commitment from **P** through his chosen q that defines an encrypted zero so that **V** can tie it to a random **e** to verify the **r** used in encrypting **m**.)

4. **P** computes$k = qr^e \bmod n$. **P** sends $k$to V.

Remark: Even V knows h and n, q cannot be derived under DCRA (Decisional Composite Residuosity Assumption).

http://dx.doi.org/10.5772/51372

5. **V** checks and accepts that $m' = m$ if

$\gcd(h, n)=1$, $\gcd(k, n)=1$ and $(g^0 . k^n \bmod n^2)=(h . z^e \bmod n^2)$ i.e., $(g^0 . k^n \bmod n^2)= (q^n r^{en}) \bmod n^2$

*Completeness –Prover **P** and verifier **V** are assumed to be honest:*

1. Assume that $m' = m$, then z as computed by **V** in Step-1 of the Zero-Knowledge protocol is an encryption of zero, i.e. $z = r^n \bmod n^2$.

2. $(g^0 . k^n \bmod n^2)$, from the verification in Step-5 of the protocol can be rewritten as $(q^n . r^{en} \bmod n^2)$.

3. Due to assumption in (1), $(h . z^e \bmod n^2)$, from the verification in Step-5 can be rewritten as $(q^n . r^{en} \bmod n^2)$.

4. From (1),(2),(3), we get that, $if\ (m' = m)$, $then\ ((g^0 . k^n \bmod n^2)=(h . z^e \bmod n^2))$

5. Assume that $m' \neq m$, then z as computed by **V** in Step-1 of the protocol equals $(g^{m'-m} . r^n \bmod n^2)$ where $(g^{m'-m} \neq 1)$.

6. $(g^0 . k^n \bmod n^2)$, from the verification in Step-5 of the protocol can be rewritten as $(q^n . r^{en} \bmod n^2)$.

7. Due to assumption in (5) $(h . z^e \bmod n^2)$, from the verification in Step-5 can be rewritten as $(q^n . (g^{m'-m} . r^n)^e \bmod n^2)$.

8. From (5), (6), (7) we get that: $if\ (\neg(m' = m))$, $then\ (\neg((g^0 . k^n \bmod n^2)=(h . z^e \bmod n^2)))$.

9. Since the verifier **V**, checks the equality in Step-5 of the protocol as a condition for acceptance, from (4), (8), it is shown that the verifier **V** is correct in his conviction i.e. it is indeed true that: $(m' = m)$ *if and only if* $((g^0 . k^n \bmod n^2)=(h . z^e \bmod n^2))$.

*Soundness – Assuming a cheating prover **P*** and honest verifier **V**:*

1. Assume that a cheating prover **P*** is trying to convince **V** that $m' = m$, where in actuality, $(m' - m) \neq 0$. Therefore **P*** would try to show that $(g^0 . k^n \bmod n^2)=(h . z^e \bmod n^2)$, where **P*** has control over k and h and no knowledge of z since $(m' - m) \neq 0$.

2. Any choice of k sent by party **P*** in Step-4 of the protocol will yield the **n** th power of k in modulo n^2 for the left hand side of the verification equation $(g^0 . k^n \bmod n^2)$.

3. Since the **P*** has to show that $(g^0 . k^n \bmod n^2)=(h . z^e \bmod n^2)$ and has no control over z^e; either h must equal $(k^n . z^{-e} \bmod n^2)$ or k^n must equal $(h . z^e)$.

4. (3) presumes knowledge of z and e. It is to be noted that h must be committed before **V** sends e to **P***. Hence **P*** can at best guess **z** and **e**, then computes and sends $h = k^n . z^{-e} \bmod n^2$, before knowledge of e.

5. When computing k, **P*** has knowledge of e, and no knowledge of z.Since $(m' \neq m)$, $z = (g^{m'-m} \cdot r^n \bmod n^2)$ where $(g^{m'-m} \neq 1)$, $(h \cdot z^e)$ contains a term $((g^{m'-m})^e)$ which is not a power of n, $(g^{(m'-m)})$ cannot be a power of n because of domain restrictions).

6. From (4),(5) **P*** cannot successfully convince **V** that $m' = m$, when in actuality $m' \neq m$.

Acknowledgements

This work was supported in part by a grant from PSC-CUNY Collaborative Research Award and a NSF DUE Award.

Author details

Bon K. Sy and Arun P. Kumara Krishnan*

*Address all correspondence to: bon@bunny.cs.qc.cuny.edu

Computer Science Dept., Queens College/CUNY, Flushing NY, USA

References

[1] Tian, Yingli, & Tsui, Hungtat. (1995). Estimating Shape and Reflectance of Surfaces by Color Image Analysis. *Lecture Notes in Computer Science 1024, "Image Analysis Applications and Computer graphics"*, 266-273.

[2] Belkin, Mikhail, & Niyogi, Partha. (2003). Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, MIT Press, June, 15(6), 1373-1396.

[3] Turk, M., & Pentland, A. (1991a). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1).

[4] Sy, Bon K. (1911). Secure Computation for Biometric Data Security Application to Speaker Verification. *IEEE Systems Journal*, 3(4).

[5] Katz, J., & Lindell, Y. (2007). Introduction to Modern Cryptography: Principles and Protocols, Chapman & Hall.

[6] http://en.wikipedia.org/wiki/Advanced_Encryption_Standard.

[7] Kumara Krishnan, Arun P., Ramirez, Adam, & Sy, Bon K. (2011). Parallel Secure Computation Scheme for Biometric Security and Privacy in Standard-Based BioAPI Framework. *Biometrics*, INTECH 2011, http://www.intechopen.com/source/pdfs/14648/InTech-Parallel_secure_computation_scheme_for_biometric_security_and_privacy_in_standard_based_bioapi_framework.pdf.

http://dx.doi.org/10.5772/51372

[8] http://en.wikipedia.org/wiki/Reed%E2%80%93Solomon_error_correction.

[9] Hao, F., Anderson, R., & Daugman, J. (2005). Combining cryptography with biometrics effectively. University of Cambridge, Tech. Rep. UCAMCL-TR-640.

[10] Barni, M., Bianchi, T., Catalano, D., Raimondo, M. D., Labati, R. D., & Faillia, P. (2010). Privacy-preserving finger code authentication. In MM&Sec', Roma, Italy. ACM.

[11] Clancy, T. C., Kiyavash, N., & Lin, D. J. (2003). Secure smart card-based finger print authentication. *Proc. ACMSIGMM 2003 Multimedia, Biometrics Methods and Applications Workshop*, 45-52.

[12] Lalithamani, N., & Soman, K. P. (2009). Irrevocable Cryptographic Key Generation from Cancelable Fingerprint Templates: An Enhanced and Effective Scheme. *European Journal of Scientific Research*, 31(3), 372-387.

[13] http://en.wikipedia.org/wiki/Secure_computation.

[14] Du, W., & Atallah, M. (2001). Privacy-Preserving Statistical Analysis, in ACSAC.

[15] Yao, A. C. (1982). Protocols for secure computations. *Proceeding of 23rd IEEE Sym. On Foundations of Computer Science*.

[16] Goldreich, O., Micali, S., & Wigderson, A. (1987). How to play ANY mental game. *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, 218-229, January, New York.

[17] Goldwasser, S. (1987). Multi party computations: past and present. *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, 1-6, August 21-24, 1997, California.

[18] Goldwasser, S., & Micali, S. (1984). Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2), 270-299.

[19] Paillier, P. (1999). Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, in EUROCRYPT.

[20] Elgamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4), 469-472.

[21] Goldwasser, S., & Micali, S. (1984). Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2), 270-299.

[22] Gentry, C. (2009). Fully Homomorphic Encryption Using Ideal Lattices. *41st ACM Symposium on Theory of Computing*.

[23] Cooney, M. (2009). IBM touts encryption innovation. *Computerworld*, 25 June 2009. [Online]. Available, http://www.computerworld.com/s/article/9134823/IBM_touts_encryption_innovation, Accessed 15.03.2012.

[24] Goethals, B., Laur, S., Lipmaa, H., & Mielikainen, T. (2004). On Private Scalar Product Computation for Privacy-Preserving Data Mining. *ICISC*.

[25] Erkin, Z., Franz, M., Guajardo, J., Katzenbeisser, S., Lagendijk, I., & Toft, T. (2009). Privacy-Preserving Face Recognition. *Privacy Enhancing Technologies*, Springer Berlin / Heidelberg, 235-253.

[26] Osadchy, M., Pinkas, B., Jarrous, A., & Moskovich, B. (2010). SCiFI- A System for Secure Face Identification. *IEEE Symposium on Security Privacy*.

[27] Kumara Krishnan, Arun P., & Sy, Bon K. (2012). SIPPA-2.0- Secure Information Processing with Privacy Assurance (version 2.0). *Proceeding of the PST 2012*, Paris, France.

[28] http://www.futronic-tech.com/product_fs88.html.

[29] http://www.neurotechnology.com/verifinger.html.

[30] Shannon, C. (1949). Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28(4), 656-715.

[31] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/General_linear_group. [Accessed 16 3 2012].

[32] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Secure_channel. [Accessed 16 3 2012].

[33] Overbey, J., Traves, W., & Wojdylo, J. (2005). On the Keyspace of the Hill Cipher. *Cryptologia*, 29(1), 59-72.

[34] "Wikipedia PP Complexity," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/PP_(complexity). [Accessed 20 3 2012].

[35] Damgard, M. Jurik, & Nielsen, J. (2010). A generalization of Paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, 9(6), 1615-5262.

[36] Fiat & Shamir, A. (1986). How to Prove Yourself: Practical Solutions to Identification and Signature Problems. *CRYPTO*.

[37] Katz, J., & Lindell, Y. (2007). Introduction to Modern Cryptography, Chapman & Hall.

[38] http://biometrics.idealtest.org/downloadDB.do?id=7 [Accessed 30 3 2012].

[39] Yang, J. C. (2011). Non-minutiae based fingerprint descriptor. book chapter, *Biometrics*, Intech, Vienna, Austria, June, 978-9-53307-618-8.

[40] Yang, J. C., & Park, D. S. (2008). A Fingerprint Verification Algorithm Using Tessellated Invariant Moment Features. *Neurocomputing*, 71(10-12), 1939-1946.