

OPTIMIZED PRIVATE BIOMETRIC MATCHING

Cross Reference To Related Applications

[0001] This application claims priority to U.S. Patent Application No. 16/664,530, filed October 25, 2019, the disclosure of which is incorporated by reference herein.

Background

[0002] The background description provided herein is for the purpose of generally presenting the context of the disclosure. Work of the presently named inventors, to the extent it is described in this background section, as well as aspects of the description that may not otherwise qualify as prior art at the time of filing, are neither expressly nor impliedly admitted as prior art against the present disclosure.

[0003] The use of biometrics as a mechanism of authentication between devices or between a device and a person has been in use for some time. However, templates of a person's biometric data may be shared and subsequently stored by devices outside a person's control. Should one or more of those biometric templates be compromised by that third party device, that person may have no recovery path, since the person cannot generate new fingerprints, face scans, iris images, etc.

Summary

[0004] In an embodiment, a biometric confirmation device may use a homomorphic encryption technique to compare a biometric template with a biometric scan to determine whether a match exists without actually disclosing the biometric template to the third party device. Further, the homomorphic process allows the terminal to send current biometric scan data to the terminal without leaking scan information. In order to optimize the encryption process, a term packing process allows a dramatic reduction in computing overhead during the verification stage.

Brief Description of the Drawings

[0005] The figures depict a preferred embodiment for purposes of illustration only. One skilled in the art may readily recognize from the following discussion that

alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles described herein.

[0006] Fig. 1 is a system diagram illustrating elements for use in private biometric matching in accordance with the current disclosure;

[0007] Fig. 2 is a bounce diagram illustrating a first technique for private biometric matching;

[0008] Fig. 3 is a bounce diagram illustrating a modification of the technique illustrated in Fig. 2;

[0009] Fig. 4 is a bounce diagram illustrating another technique for private biometric matching; and

[0010] Fig. 5 is a bounce diagram illustrating another aspect of the technique of Fig. 4.

Detailed Description

[0011] A terminal may wish to confirm that a person is associated with a particular device, such as a smartphone. For example, a point of sale device (POS), such as at a retailer, may be capable of processing a transaction using a sight-unseen smartphone. That is, the POS may be able to complete a transaction using Wi-Fi or Bluetooth with a smartphone using the same protocol as would have been used if the smartphone was presented to the POS via Near Field Communication (NFC) by tapping the smartphone to the POS. However, the POS would first need to determine which smartphone in the vicinity is the one with the proper credential corresponding to the current purchase and purchaser.

[0012] The POS may send a request to nearby smartphones requesting biometric templates of images for their respective users. The POS may also capture an image of the user engaged in the current transaction. Once the POS finds a smartphone with the image biometric template of the user matching the image just captured, the transaction processing can proceed using a normal payment flow. That is, in an exemplary embodiment, the smartphone can establish a secure channel with the POS, the transaction data can be transferred and signed by the smartphone's payment application along with the personal account number (PAN) or token data related to the

user's payment card. This information may be passed by the POS to the retailer's acquiring bank for approval and processing.

[0013] However, in this process, the POS should not simply be able to request biometric templates from every nearby smartphone simply so that matching the image to the correct user can be performed. This would expose the biometric data of every nearby smartphone user, including those devices/users who are not involved in the transaction in any way. Conversely, the image of the user or its biometric template, should not be shared with each nearby smartphone so that the user's information remains protected from unwanted distribution. The techniques for accomplishing this are discussed below. In this example, there is little or no threat of the smartphone attempting to spoof the POS into thinking it is someone else, because the payment is made via a separate purchase transaction using a different credential. This type of identification may be thought of as passive.

[0014] In a similar process for a different transaction type, for example, one involving receiving cash at an automated teller machine (ATM), the benefit of a deliberate attempt to deceive the identification process at the ATM could result in fraudulent receipt of funds. In this case, an active identification may be used to further identify the user. In this case, instead of just matching the biometric template, the user may provide a copy of the template signed by an authority that also includes user identity information. In this case, the approval process results in not only establishing a relationship between a user and a smartphone, but the user and the user's identity as previously established with the authority.

[0015] A homomorphic cryptography algorithm, such as a Paillier encryption among others, may be used. The basic premise is that an operation performed on encrypted data has the same result as that operation being performed on unencrypted data. As will be discussed in greater detail below, this allows verification operations to be performed on encrypted data so that while the matching process may occur on the terminal, the biometric template never leaves the smartphone.

[0016] While the advantages of such a process are apparent, homomorphic cryptographic algorithms, such as Paillier, can be very compute-intensive at certain phases of the process. For example, key generation and data encryption can be very

time consuming, so long that including these steps in a normal cryptographic transaction, such as a purchase transaction may be prohibitive. The following description discloses an approach to address these shortcomings. Further, the homomorphic algorithms as applied to biometrics can yield datasets that are large enough to adversely affect transmission time between the user device and the terminal (for example, the POS or ATM). An embodiment described below for packing data reduces the data required for transmission by a factor of 8, although other reduction factors are possible. Bit masking of non-essential terms also helps to reduce the potential for data leakage during the data reduction process.

[0017] In an embodiment using current technology, the matching process takes about 300 milliseconds with an initial communication payload of 55 Mbytes of data and about 1 Mbyte for each communication after that.

[0018] Fig. 1 illustrates an exemplary system 100 that supports the homomorphic matching process. A terminal 102 may be a POS device, such as a register at a retailer, an ATM, or any other device for which a biometric authentication may be useful including access control, record keeping, or other personal transactions. The terminal 102 may include a processor 104 coupled by a data bus 105 to a memory 106, a biometric capture device 112, a cryptographic module 114 and a wireless interface 116. The memory 106 may include an operating system and utilities 108 used in the startup and basic operation of the terminal 102 including communication and user interface, as is known. A transaction processing module 110 may handle interactions between purchasers and/or payment instruments, bar code scanners or card acceptance devices, host systems, and acquirers.

[0019] The homomorphic module 114 may include cryptographic routines for encryption and decryption, as well as routines for generating templates from biometric data received from the biometric capture device 112. These data may include facial images, fingerprint scans, iris scans, retinal scans, palm outline data, voice prints, etc. More specific functions associated with the homomorphic module 114 are discussed below. The wireless interface 116 may include a Wi-Fi module, a Bluetooth module, or any wireless connection hardware and associated protocol that allows relatively high speed data communication between the terminal 102 and external entities.

[0020] The system 100 may also include or be coupled with a plurality of mobile devices such as smartphones 118a, 118b, 118c, and 118d. The mobile devices may also include laptops, tablets, wearable devices or other electronic gear capable of storing data and two-way communication.

[0021] In operation, a user 120 may approach the terminal 102 to complete a transaction. The terminal 102 may wish to determine which, if any, of the smartphones 118 in the vicinity are associated with the user. Several different scenarios for this process are discussed with the help of Figs. 2-5.

[0022] Turning to Fig. 2, an interaction 200 that only identifies the device, that is, a passive transaction, is shown. At step 201, the user device 118a may generate an asymmetric key pair pk , sk (public key and secret key). This may be done one time only at any time prior to the transaction. A biometric x may be collected and stored as a series of data x_1 - x_n , also referred to as a biometric template, or more simply, a template. The template may be encrypted into a set of values represented by $[[x_n]]$. The collection, generation, and encryption of the biometric data may also be done only one time prior to its use. As the key generation, template generation, and encryption are the most compute-intensive steps associated with the use of Paillier (homomorphic) encryption, the one-time pre-computing of these data may be performed when the device is inactive, saving time during a transaction.

[0023] At step 202, at the beginning of a potential transaction, the terminal 102 may collect a biometric measurement, such as an image, of the user 120 and use the raw data to generate a template y . Because the terminal 102 may have fewer limitations than the mobile device including access to power and a larger processor, the generation of the template via a neural network may be performed virtually in real time. In other embodiments, the terminal 102 and a mobile device may begin this initial step before the transaction is even initiated, such as when the user enters a store. Step 202 may continue by requesting a template match from surrounding mobile devices 118a, 118b, 118c, 118d. While the following process may be repeated for each mobile device, only a single response is followed for the sake of clarity and simplicity.

[0024] In response to the request, the smartphone's public key and the encrypted template may be sent to the terminal 102 at step 204. At block 206, the terminal 102

may multiply each term of the encrypted template with the corresponding term of the local template. As discussed more below, a characteristic of Paillier encryption or other additive homomorphic encryption schemes is that operations performed on encrypted data yield the same result as the same operation on unencrypted data.

[0025] At this point, the terminal has no information about the user's biometric template, other than that it exists. If the terminal 102 returns the result matrix z , that is, the inner product of x and y , to the device 118a, the device 118a may be able to determine the template taken at the terminal. To avoid leaking this information to the device 118a, the terminal 102 may add a random number, B , to the result of the operation. With the addition of the random value, the device 118a cannot discover any information about the template from the terminal 102.

[0026] The encrypted value z which includes the added value B , is sent at step 208 from the terminal 102 to the device 118a. There, at step 210, the device 118a may decrypt the z to reveal the plaintext value z . The decryption process is much more efficient than the encryption process, so the device 118a can perform the decryption in a timely manner compared to expected transaction processing time. Further, the data transferred at step 208 (and at step 212 below) is significantly smaller than the original transfer at block 204. This helps to speed the process.

[0027] This value may be returned at step 212 to the terminal 102. By subtracting the value B from z , the inner product of x and y can be developed. If this inner product is less than a threshold value, or in the case of cosine similarity match, close enough to a value of 1, the match is confirmed. As discussed above, since the biometric match is not actually used to complete the transaction, this passive technique is usable. The match merely confirms that the payment credentials on the device 118a can be used to make the payment. If the device 118a were to take steps to spoof or deceive the terminal 102 into thinking it is someone else, the result would simply be that the device 118a completes the payment transaction, perhaps for goods or services that in the end are taken by another user. In other words, there is no incentive to deceive in this scenario.

[0028] While the size of the data sent in the example of Fig. 2 may be manageable, Fig. 3 illustrates a packing technique for reducing the data size of at least the initial data

transfer by a factor of up to eight. While larger and smaller size reductions are possible, for current system configurations the factor of eight may provide an optimized balance between processing time and data transmission time. At step 221, similar to the previous example, the device 118a may collect biometric data and generate the key pair. In this case, only every eighth value of the template may be encrypted. In various embodiments, the value of the number of skips may be a function of the level of encryption (key size). This is discussed more below.

[0029] The new encrypted vector, $[[x'_j]]$, may therefore be, in this example, eight times smaller than the similar encrypted value of the example of Fig. 2. The value may be sent to the terminal at step 224 responsive to a request following generation of the template at step 222. At step 226, the terminal 102 will perform a similar grouping of the terms $(y_i, \dots, y_{i+7})$ to yield y'_j . The respective groupings are multiplied together and the resultant value sent back to the device 118a at step 228.

[0030] At step 230, a random value grouping $r=(r_0, \dots, r_7)$ may be generated and added to the decrypted values to prevent overflows, as discussed in more detail below. The decrypted grouping z' and value $r'=r_0+\dots+r_7$ may be returned to the terminal at step 232. At step 234, the value p is the difference between z' and B , while the inner product is the sum of the terms less r' .

[0031] Fig. 4 illustrates a process 240 that illustrates the case where the user/device 118a may be highly incentivized to deceive the terminal 102, such as when making a withdrawal at an ATM. In this case, the device 118a may register the encrypted template with an authority, such as a financial institution, to bind the encrypted template, an authority key, and an identity of the user 120 in a signed object σ . After step 242, step 243 and step 244 follow as above. Step 246 not only performs the operation on the local template and the received encrypted template and adds the value B but also multiplies the result of the operation by a second random value A .

[0032] As above, the value z is sent to the device 118a at step 248. At step 250, the private key, sk , is used to decrypt the value z , which is returned to the terminal 102 at step 252. At step 254, the value B is subtracted from z and the result divided by A to determine the inner product value.

[0033] Fig. 5 illustrates the process 240 of Fig. 3 showing the effect of the value A when the device 118a attempts to deceive the terminal 102 into thinking there is a match. At step 250, the device 118a may attempt to add (or subtract) a value e to the resulting decryption value so that the terminal 102 will reach an incorrect conclusion about the value. However, the value of A may be selected so that any attempt to alter the value z will result in the returned $z+e$ term, when multiplied by A^{-1} will be well above the threshold value t due to the plaintext space of the being large. The use of modulus math ensures that even though the terminal 102 does not know the value of e , any attempt to alter the returned value will be detectable.

[0034] Once the match is made, for example, at Fig. 4, the user identifier from the encrypted token may be extracted using the authority public key. This essentially creates a two-factor authentication, the first being the match of the biometric and the second being the possession of the token/object σ .

[0035] The following describes details of the encryption and padding processes.

[0036] 1 Introduction

[0037] Our goal is to enable a card holder's phone to store their biometric data/template r and to securely enable an untrusted payment terminal to determine if a biometric measurement taken by it matches the template v . For various reasons this must happen without revealing v to the terminal or the measurement to the phone. The communication channel will be Bluetooth or Wi-Fi. The current protocol implementation achieves this in roughly 0.3 seconds with 55MB of communication for one comparison and 1MB of communication for each comparison after that.

[0038] 1.1 Biometrics

[0039] The field of biometric authentication is quite well explored under some use case. For facial and fingerprint biometrics, many of the most of effective methods follow a similar framework which build on machine learning and neural networks. Beforehand, given many raw biometric measurements (e.g. photos of people's faces or fingerprint scans) a neural network can be trained to "embed" a new biometric measurement into the vector space $\mathbb{R}^n$ such that two embeddings can easily be compared to determine if they are from the same source, i.e. person. In particular, we will consider the embedding where the "comparison" function is the *cosine similarity* function. The

function outputs a value in $[-1,1]$ which is proportional to the angle between two vectors in $\mathbb{R}^n$. That is, when two embeddings have a small angle between them they are considered a match. More specifically, for a threshold $t \in (0,1]$ we say that two embeddings $u, v \in \mathbb{R}^n$ match each other if

$$\begin{aligned} t &\leq \text{CosineSim}(u, v) \\ &= \frac{u \cdot v}{\|u\|_2 \|v\|_2} \\ &= \frac{\sum_i u_i v_i}{\sqrt{\sum_i u_i^2} \sqrt{\sum_i v_i^2}} \\ &= \cos(\Theta) \end{aligned}$$

[0040] Putting it together, let NeuralNet be the trained neural network which outputs embeddings. The two measurements m_1, m_2 are said to be a match if

[0041] $t < \text{CosineSim}(\text{NeuralNet}(m_1), \text{NeuralNet}(m_2))$

[0042] For the remainder of the paper we will refer to these embeddings in $\mathbb{R}^n$ as a biometric template.

[0043] 1.2 Setting

[0044] Broadly speaking, our goal is to allow a user to authenticate to terminal (e.g. Visa card terminal, ATM) using their biometric data. From a pure functionality point of view we wish the terminal to take a measurement (e.g. photo) m^* of the user/customer, and then should learn if there exists a pre-registered measurement m_{user} of a user that matches, i.e. learn user such that

[0045] $t < \text{CosineSim}(\text{NeuralNet}(m^*), \text{NeuralNet}(m_{user}))$

[0046] A naive solution to this problem is to maintain a large database containing the measurement m_{user} for each user. When a terminal takes a measurement m^* of the customer, the terminal sends m^* to a central server which computes if a matching m_{user} exists. One issue is that comparing all m_{user} against m^* could be expensive due to having many users. Secondly, the threshold t would have to be very strict to ensure that m_{user} is unique. This could lead to a bad user experience due to the increased probability of rejecting valid customer measurements (photos). Finally, Visa does not wish to maintain a database of user biometric data due to privacy concerns and the risk

of leaking this data during a security breach. For example, a rouge terminal could simply start asking the central server if a random/probable template v is contained in the database and the matching user ID if it exists.

[0047] 1.3 Homomorphic Encryption

[0048] The solution we build relies on techniques known as (additively) homomorphic encryption. Classical encryption allows an entity to sample a random key k and then encrypt data as $c = \text{Enc}_k(m)$. We will also use the notation that $\llbracket m \rrbracket = \text{Enc}_k(m)$ to denote that $m \in \mathbb{Z}$ is encrypted under some key. Homomorphic encryption extends this idea to allow an entity with $\llbracket m_1 \rrbracket, \llbracket m_2 \rrbracket$ to “add” the ciphertexts to get a ciphertext of the underlying values added together. That is

$$[0049] \quad \llbracket m_1 \rrbracket + \llbracket m_2 \rrbracket = \llbracket m_1 + m_2 \rrbracket$$

[0050] The ability to add two ciphertexts means that the encryption scheme is additively homomorphic. The computation of $m_1 + m_2$ is performed modulo some integer p . An example of such a scheme is known as Paillier encryption. Additively homomorphic encryption also has the property that you can multiply a ciphertext $\llbracket m_1 \rrbracket$ with a plaintext m_2 by performing repeated additions. That is

$$\llbracket m_1 \rrbracket * m_2 = \sum_{i=1}^{m_2} \llbracket m_1 \rrbracket = \llbracket m_1 m_2 \rrbracket$$

[0051] Some encryption are multiplicatively homomorphic which means ciphertexts can be multiplied together. That is

$$[0052] \quad \llbracket m_1 \rrbracket * \llbracket m_2 \rrbracket = \llbracket m_1 m_2 \rrbracket$$

[0053] For example, ElGamal encryption is multiplicatively homomorphic. If a scheme is both additive and multiplicatively homomorphic then it is called a fully homomorphic encryption scheme.

[0054] Unless otherwise stated, the protocols here are only required to be additively homomorphic. Therefore fully homomorphic encryptions schemes would also be compatible with our protocols.

[0055] 2 Overview

[0056] This solution takes a different approach by having the user store their biometric data m_{user} on their own phone. This approach has the advantage that the overall collection of data has no single point of failure. Moreover, storing biometric data only on

the phone follows the existing FIDO standard. A breach of the central server or even a phone does not reveal the biometric data of all users.

[0057] We consider several entities for this application. A *user* with their mobile device and pre-registered biometric measurement m_{user} . This user is running a banking app that will perform the following protocol(s). A central *server* will help in the authentication but will never store plain biometric data. Finally, a *terminal* which the user will interact with. This terminal can be in communication with both parties and trusts the server to be honest. Once set up, the terminal should not need to talk to the server to authenticate a user. The terminal is equipped with a biometric reader (e.g. fingerprint read or camera).

[0058] The overall flow of the protocol is given in Figure 1. In a registration phase the user encrypts their biometric data and sends it to the server. The biometric data is then tied to the user's identity using cryptographic methods (e.g. a signature) to generate a payment/identity token. This token is returned to the user and is only valid for this user and in the context of a biometric payment with this exact encrypted template.

[0059] In the comparison phase the terminal will set up channels with nearby devices/users (e.g. over Bluetooth or Wi-Fi) and run the comparison protocol with each of them. The terminal has one (or more) measurement and wishes to identify which of the users have registered a matching biometric. Each user will begin by sending their token σ_{user} and encrypted template $\llbracket u' \rrbracket$ to the terminal.

[0060] When the terminal takes a measurement m_i^* , it first computes the template $v_i = \text{NeuralNet}(m_i)$ and normalizes it as $v'_i = v_i / \|v_i\|_2$. It is then the case that $\text{CosineSim}(u, v_i) = \sum_j u'_j v'_{i,j}$. Using additively homomorphic encryption, an encryption of $\text{CosineSim}(u, v_i)$ can be computed by the terminal as $\llbracket \text{CosineSim}(u, v_i) \rrbracket = \sum_j \llbracket u'_j \rrbracket v'_{i,j}$. This encryption can then be sent to the user to be decrypted. However, this would reveal the result to the user instead of the terminal. This is rectified by having the terminal one-time mask the result with a random value r_i as $\llbracket x_i \rrbracket = r_i + \sum_j \llbracket u'_j \rrbracket v'_{i,j}$. The user decrypts and sends back x_i which allows the terminal to compute $\text{CosineSim}(u, v_i) = x_i + r_i$.

[0061] This protocol has several notable improvements over a “standard” approach to solving this program. First is that this protocol is optimized for the case where the user’s template is fixed and will be compared with server measurements $m_1^*, \dots, m_\ell^*$. In particular, the user only sends the large vector $\llbracket u' \rrbracket$ once which is used to compute the ℓ small results $\llbracket x_1 \rrbracket, \dots, \llbracket x_\ell \rrbracket$. This greatly improves the efficiency of the protocol for the intended use case.

[0062] For example, when a user walks into a store, their device can send $(\sigma_{user}, \llbracket u' \rrbracket)$ to the terminal. This will consist of the bulk of the communication between the devices and the terminal, e.g. 55KB. This message is large due to $\llbracket u' \rrbracket$ being the vector of roughly $n = 512$ elements. Later, when the terminal takes a measurement m_i^* , they can compute the single element $\llbracket x_i \rrbracket$, which is n times smaller than $\llbracket u' \rrbracket$, and send this to the user device. Not only does this decrease overall communication by allowing the terminal to reuse $\llbracket u' \rrbracket$, it also decreases latency due to $\llbracket u' \rrbracket$ being sent before a measurement has ever been taken. In practice with the use of Paillier encryption, sending $\llbracket u' \rrbracket$ currently requires roughly 55KB of communication while $\llbracket x_i \rrbracket$ and x_i combined require less than 1KB.

[0063] The second advantage of this approach is that the user only generates the encryption $\llbracket u' \rrbracket$ once during the registration phase. This is of particular concern due to Paillier encryption being somewhat inefficient while computation and decryption is more efficient. For example, encrypting u'

[0064]

[Registration] A protocol between the user and server.

- (a) The user captures their raw biometric data m_{user} , e.g. a photo of their face.
- (b) The user computes their biometric template $u = \text{NeuralNet}(m_{user})$.
- (c) The user generates an encryption key pair (sk, pk) and computes the encryption $\llbracket u'_i \rrbracket = \text{Enc}_{pk}(u_i / \|u\|_2)$.
- (d) The user sends $\llbracket u' \rrbracket$ to the server which generates an identity token $\sigma_{user} = \text{Token}_k(user, \llbracket u' \rrbracket)$ and returns σ_{user} to the user.

[Compare] A protocol between the user and terminal. The terminal has ℓ measurements $m_1^*, \dots, m_\ell^*$. The terminal will learn which m_i^* match m_{user} .

- (a) The user sends the terminal $(\sigma_{user}, \llbracket u' \rrbracket)$ who checks that they are “valid”.
- (b) For each $i \in [\ell]$, the terminal computes $v_i = \text{NeuralNet}(m_i^*)$ and $v'_i = v_i / \|v_i\|_2$.
- (c) For each $i \in [\ell]$, the terminal computes $\llbracket x_i \rrbracket = r_i + \sum_j \llbracket u_j \rrbracket v'_{i,j}$ where r_i is a random value. Note $x_i = \text{CosineSim}(u, v_i) + r_i$.
- (d) The terminal sends $(\llbracket x_1 \rrbracket, \dots, \llbracket x_\ell \rrbracket)$ to the user which decrypts and returns $(x_1, \dots, x_\ell)$.
- (e) For each $i \in [\ell]$, the terminal compute $\text{CosineSim}(u, v_i) = x_i - r_i$. If $\text{CosineSim}(u, v_i) \geq t$ then let $\tau_i = \sigma_{user}$ and $\perp$ otherwise.
- (f) The terminal outputs $(\tau_1, \dots, \tau_\ell)$.

Figure I: Overview of the bask protocol.

[0065] can require many seconds while the rest of the protocol requires much less than a single second. As such, if the user had to freshly encrypt u' for each measurement, then the use of Paillier encryption could be prohibitively inefficient/power intensive for the practicality of this application.

[0066] An advantage of this approach over comparing the measure m_i^* with a central database containing all registered measurements m_{user} is that the terminal can significantly narrow down the search to only the measurements of nearby users. For example, there could be 20 nearby users while the total number of users could be a hundred million or more. This could allow us to set the threshold t to be much more relaxed and decrease the probability of rejecting a pair of matching measurements.

[0067] 3 Batching & Paillier Encryption

[0068] We now introduce a new optimization for Paillier encryption which reduces the communication overhead of the bask protocol by roughly 4 to 8 times. Paillier is additively homomorphic with a plaintext space $\mathbb{Z}_{pq}$ where p, q are large primes. Typical sizes of p, q is 1024 bits each which means that a single Paillier ciphertext can contain 2048 bits of plaintext data. The security level would then be similar to RSA with a 2048 bit key. In the biometric application we must encrypt a vector $u = (u_1, \dots, u_n)$. The standard method for doing this to encrypt each u_i on its own to give you n ciphertexts $\llbracket u_1 \rrbracket, \dots, \llbracket u_n \rrbracket$. However, this is very wasteful since each element u'_i can typically be

represented using at most 32 bits while a ciphertext is large enough to hold 2048 bits. The size of a Paillier ciphertext cannot be made smaller due to security constraints.

[0069] 3.1 Encoding

[0070] A method known as packing has long been known to allow you to encode several values into a ciphertext. The idea is that you place multiple elements into a single ciphertext with zero padding between them. For example, the packing of $u_1, \dots, u_n \in \mathbb{Z}_{2^m}$ can be computed as $u^* = \sum_{i=1}^n 2^{(i-1)(m+d)} u_i \in \mathbb{Z}_{n(m+d)}$ where d is the number of padding bits between each element. Given two such packed values, u^*, v^* , adding them results in

$$\llbracket m_1 \rrbracket * m_2 = \sum_{i=1}^{m_2} \llbracket m_1 \rrbracket = \llbracket m_1 m_2 \rrbracket$$

[0071] i.e. the addition of each element in the encoded format. The d padding bits are used here to ensure that if $u_i + v_i$ generates a carry bit, i.e. $u_i + v_i \geq 2m$, then this carry will be stored in the padding bits. In this case $d = 1$ would be sufficient. More generally, this approach works so long as no position ever requires more than $m' = m + d$ bits to be represented.

[0072] This encoding also works when multiplying a single value $x \in \mathbb{Z}_{2^m}$ with the encoded vector v^* . In particular,

$$xv^* = \sum_{i=1}^n 2^{(i-1)m'} m'(xv_i)$$

[0073] Again we require d to be large enough to make sure xv_i can be represented with $m' = m + d$ bits. In this case $d = m$ would be sufficient. However, multiplying two vectors component-wise (Hadamard Product) does not work in this format. For example, if we have $n = 2$ then

$$[0074] \quad v^* u^* = 2^{0m'} v_1 u_1 + 2^{1m'} (v_1 u_2 + v_2 u_1) + 2^{2m'} v_2 u_2$$

[0075] Although we do not get $2^{0m'} v_1 u_1 + 2^{m'} v_2 u_2$ as desired, observe that the values of interest $(v_1 u_1, v_2 u_2)$ are stored in the result just with $v_2 u_2$ at a different positions than in the input. We call this its “level 2” position. It is easy to verify that for any n ,

[0076] $v^*u^* = v_1u_1 + 2^{m'}(\dots) + 2^{2m'}(\dots) + \dots + 2^{(2n-2)m'}v_nu_n$

[0077] v_1u_1 and u_nu_n can be recovered but the other (...) terms will consist of various cross terms between u_i and v_j . We now show a new encoding strategy which allows us to recover all of the product terms v_iu_i from various positions v^*u^* . Let $n = 3$ and let us now encode $u_1, \dots, u_n$ as

[0078] $u^* = u_1 + 2^{m'}u_2 + 2^{3m'}u_3$

[0079] Note that the position for $2^{2m'}$ is set to zero. If we then multiply v^*u^* using these encodings we obtain

[0080] $v_1u_1 + 2^{m'}(v_1u_2 + v_2u_1) + 2^{2m'}v_2u_2 + 2^{3m'}(v_1u_3 + v_3u_1) +$

[0081] $2^{4m'}(v_2u_3 + v_3u_2) + 2^{5m'}(0) + 2^{6m'}v_3u_3.$

[0082] The “level 2” positions here are 0, 2, 6, i.e. the first bit of each v_iu_i starts at bit index 0, $2m'$, $6m'$. Then general ideal can be extended for an arbitrary value of n . In particular, given a suitable encoding for vectors of $n - 1$ elements, an encoding for n elements can be performed by adding $2^{n'm'}u_n$ to the existing encoding where n' is the smallest integer such that no previous element is encoded at position n' and the new cross terms introduced do not collide with previous output positions. For example with $n = 3$ then $n' = 3$ instead of 2 (the next available input position) because the output position for v_2u_2 would have been obscured with the cross terms $(v_1u_3 + v_3u_1)$.

Define I index these values such that

[0083] $u^* = \sum_i 2^{m'I[i]} u_i$

[0084] is the encoding.

[0085] For this encoding strategy, we must additionally ensure the d is sufficiently large such that any overflow from any of the cross terms does not exceed $m' = m + d$ bits.

[0086] 3.2 Masking

[0087] One issue with the encoding strategy is that if we reveal v^*u^* to a party, then they will learn $\{v_1u_1 \mid i \in [n]\}$ and the value of all of the other positions. For example, with $n = 2$ learning v^*u^* also reveals $v_1u_2 + v_2u_1$ which is undesirable. This issue is overcome by adding an additional random value to each of these position. When encoding a vector, we now make every slot of size $m' + \times$ bits, i.e.

[0088] $u^* = \sum_{i=1}^n 2^{l[i](m'+\lambda)} u_i$

[0089] and when we decrypt / reveal $v^* u^*$ we now reveal

[0090] $x = v^* u^* + \sum_{i \in \bar{I}} 2^{i(m'+\lambda)} r_i$

[0091] instead where r_i is uniform value in $\mathbb{Z}_{2^{m'+\lambda}}$ and $\bar{I}$ indexes the positions with the cross terms we wish to hide. To ensure that adding the r_i terms do not corrupt the other positions, we require that the addition between the cross terms and r_i does not overflow the $m' + \lambda$ bits of the position. Recall that we guarantee that m' is large enough to hold the cross terms. Therefore the probability of an overflow is that probability that the top λ bits of r_i is all ones. This happens with probability $2^{-\lambda}$ which can be made arbitrarily small so that the overall error probability is acceptable.

[0092] 3.3 Signed Values

[0093] Signed values can also be supported. This is achieved by using a twos complement encoding of the values. Given an m bit signed value, encode it as a twos complement $2m$ bit number¹. This requirement for this expansion is to ensure that after multiplying two of these values, the first $2m$ bits of the product. One security ramification of this approach is that we must extend the masking approach described above to hide the top $2m$ bits of the $4m$ bit product. If these bits were not masked then when the product is reveal the decrypted could distinguish multiply two positive values and two negative values which both result in the same logical value, i.e. the product of negative numbers would have at least one 1 in the top $2m$ bits while the other would not.

[0094] Regardless, these top $2m$ bits can be masked in a straightforward extension of the masking technique described above.

3.4 Inner Product

[0095] Up to this point we have only discussed computing component wise multiplication. However, we desire computing inner product. The high level ideal is as follows where we are encoding 8 values per ciphertext.

[0096] 1. Each party computes several encodings

¹ Actually, this must be sign extended to the size of the final result in bits (when not twos complement encoded). For inner products this could be as large as $2m + \log_2 n$ bits. See below.

[0097] (a) $u_1^*, \dots, u_\ell^* = E(u_1, \dots, u_8), \dots, E(u_{n-7}, \dots, u_n)$

[0098] (b) $v_1^*, \dots, v_\ell^* = E(v_1, \dots, v_8), \dots, E(v_{n-7}, \dots, v_n)$

[0099] 2. The user encrypts $[[u_1^*]], \dots, [[u_\ell^*]]$ and sends them to the terminal.

[00100] 3. The terminal computes $[[Z^*]] = \sum_i [[u_i^*]] v_i^* + B = [[\sum_i u_i^* v_i^* + B]]$ and sends it to the user.

[00101] 4. The user decrypts Z^* and compute $s^* = Z^* + r^*$ where r^* is an encoding at “level 2” of the random values $r_1, \dots, r_8 \in \mathbb{Z}_{2^{m'}}$ and m' is the number of “unmasked” bits at each position. The user sends s^* and $r' = \sum_i r_i \bmod 2^{m'}$.

[00102] 5. The terminal computes $x^* = s^* - B$ and decodes it to $x_1, \dots, x_8$ and computes $\sum_i u_i v_i = \sum_i x_i - r'$.

[00103] Additional care here must be taken to ensure that enough padding bits are placed between each value to guarantee that no overflows will happen. This can be done simply by computing the maximum value any position can take.

[00104] As an optimization, if a priori it is known that any that the inner product will not exceed some given value then fewer padding bits may be needed. This is the case of cosine similarity where the inner product can be at most 1 (times some scaling factor) while each product $u_i v_i$ can also be at most 1. This observation can be used to prove that $\log_2 n$ fewer padding bits are needed. This is contrasted with the case where the only constraint is that $u_i v_i$ can be at most 1 which would imply that the inner product could be as large as n .

[00105] Fully Homomorphic Encryption (FHE) can be used in a similar way as Paillier encryption. It is actually more computationally efficient but requires an increase in communication overhead. With careful optimization this added overhead may be minimized. An advantage of FHE is that it could be used to give stronger guarantees that the user’s encrypted template $u_1, \dots, u_n$ is of correct form. Further investigations are required.

[00106] A technical effect of the current concept is that a user’s biometric template never leaves the user’s possession even though the matching process occurs at the terminal device at which the user is requesting authentication. This is a technical solution to a technology problem, that of preserving the integrity of a person’s biometric

data from compromise during the act of using that biometric with a foreign and unknown device. Similarly, the process ensures that a terminal device capturing a person's biometric data will not leak that data to other devices while attempting to discover the correct device with which to interact.

[00107] Further, the bit packing process disclosed overcomes previous carry bit problems to allow significant data reductions in the size of data transferred between devices, resulting in significant speed improvements.

[00108] A system and method in accordance with the current disclosure benefits both users and merchants. The biometric matching may occur with little or no interaction from the user 120, and the transaction accomplished so that the user does not even need to remove his or her device 118a from a purse or pocket. This streamlines the interaction while reducing the inconvenience and threat posed by removing the device 118a from a purse or pocket. More importantly, the biometric matching occurs without the terminal 102 or any other server from even temporarily handling the user's biometric template. That information is stored solely on the user's device 118a. The terminal 102 may gather biometric data directly from the user for the period of the transaction but without the means to register that information, that template has little to no value. As a rule, a merchant does not want to be responsible for that kind of user data in the event of a breach and will systematically discard such information after a transaction.

[00109] The figures depict preferred embodiments for purposes of illustration only. One skilled in the art will readily recognize from the following discussion that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles described herein.

[00110] Upon reading this disclosure, those of skill in the art will appreciate still additional alternative structural and functional designs for the systems and methods described herein through the disclosed principles herein. Thus, while particular embodiments and applications have been illustrated and described, it is to be understood that the disclosed embodiments are not limited to the precise construction and components disclosed herein. Various modifications, changes and variations, which will be apparent to those skilled in the art, may be made in the arrangement,

operation and details of the systems and methods disclosed herein without departing from the spirit and scope defined in any appended claims.