Abstract—We will look into privacy preserving biometrics using the example of a fingerprint reader and partially homomorphic encryption. Therefore we will cover the basics necessary to understand the discussed subject, partially homomorphic encryption and fingerprint based authentication, as well as showing a concrete protocol and its implications on performance and security of the system. While security will remain the same, speed is heavily reduced and bandwidth increased compared to the unprotected protocol. We will see to which problems homomorphic encryption presents a solution and which have to be solved differently.

1 INTRODUCTION

Biometric authentication has become more popular than ever. Nearly every new notebook is equipped with a fingerprint reader and in the future more service providers and governments want to use those unique characteristics of humans for their authentication and identification processes. Biometric identifiers are usually unique, it is thought to be impossible for two humans to share the same biometric traits, and permanent, which reduces the need to update data. However, those characteristics carry a risk. Uniqueness makes you identifiable and permanent characteristics are themselves unrevocable. A user might want to prevent theft of his biometric data, while a service provider wants to prevent a user from learning anything about the database other than the users presence in it.

Processing of biometric data is done with thresholds, since reading a biometric traits rarely give the exact same results, due to sensor noise, bodily functions or posture - the image can be rotated, wrinkles, scrapes or sweat can distort the image. Therefore a threshold is defined as the maximum acceptable error on comparing the read and stored samples.

The system can not always be entirely trusted or the processing or storage environment is untrusted. It is desirable that no unprotected data enters an untrusted environment, where unprotected describes private data in any ways that can be processed in a not specified way for other purposes than the desired authentication.

Authentication has to work fast and with few bandwidth to be accepted by the user and applicable for a system. If there is too much load on server side, it is expensive and slow to use for lots of users. A system that needs big bandwidth needs an expensive and reliable underling network.

These still unsolved constraints make solutions for the problem hard to be found and accepted.

In Section 2 we will discuss the basics for the discussed topic, especially the basic scenario, the functionality of fingerprint based authentication and homomorphic encryption. As an implementation of homomorphic encryption, we will look at the Paillier encryption mechanism. In Section 3 we will introduce the protocol of Barni et al. [2] and look into its mechanics and implications on performance and security. Subsequently we will look at the limitations and alternatives. Finally Section 5 will summarize this term paper and give an outlook.

2 BASICS

In this Section, we will have a look at the necessary basics for privacy preserving biometrics using homomorphic encryption.

2.1 Notation

Throughout this term paper, we will use the notations shown in Table 1.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>({a}_{sk})</td>
<td>The encryption of a using key sk.</td>
</tr>
<tr>
<td>({a})</td>
<td>The encryption of a when the used key can be known from context.</td>
</tr>
<tr>
<td>(a/b)</td>
<td>a is a divisor of b.</td>
</tr>
<tr>
<td>(a/b)</td>
<td>a is not a divisor of b.</td>
</tr>
<tr>
<td>(\phi(n))</td>
<td>The euler phi function.</td>
</tr>
<tr>
<td>(\mathbb{Z}/n\mathbb{Z})</td>
<td>A cyclic group.</td>
</tr>
<tr>
<td>(\mathbb{Z}/n\mathbb{Z})</td>
<td>The multiplicative group of integers modulo n, that has (\phi(n)) elements.</td>
</tr>
<tr>
<td>(x_1,\ldots,x_n)</td>
<td>The elements of an n dimensional vector x, with (x = (x_1,\ldots,x_n)).</td>
</tr>
</tbody>
</table>
2.2 Scenario

The basic scenario we will look at is a client server model, where the client is equipped with a biometric reader and the server is entrusted with a database of identities and corresponding biometric templates. The goal of the protocol we discuss in Section 3 is to authenticate a user using the users fingerprint, without revealing any additional information. We want to perform identification (called single authentication by Barni et al. [1]), knowing if the user is present in the database or not, or authentication, gaining every identifier matching the fingerprint of the user. The view of the client is shown in Figure 1. The client, fingerprint reader and their communication are trusted by the user, whereas the network and the server are untrusted. We want the server to learn nothing on the clients fingerprint, not even if it is in the database or not. We do not want the client to learn anything about the database, besides the information needed for the authentication. The protocol should fulfill certain criteria:

- **The server must not learn the fingerprint of a user:** We want the server to remain ignorant on the users fingerprint. The server is not necessarily the service provider.

- **The server must not learn about the identification:** We also want the server to remain ignorant on the authentication and matching process while authentication or identification.

- **The client must not learn anything unnecessary about the database:** The client should only be able to learn if the user is present in the database (identification) or every associated identifier for the given fingerprint (authentication). Beyond that we want the client to remain ignorant.

- **Listening to the network should be futile:** A passive listener should not gain any information on the fingerprint or the authentication process.

- **A compromised fingerprint should be revocable:** If the fingerprint should get compromised, we want to revoke the old registration and generate a new one.

- **The authentication process should react fast and use little bandwidth.**

2.3 Adversary Model

Due to the many ways of attacking a complex system we will assume an honest-but-curious setting for the protocol by Barni et al. [2]. For this scenario we make a lot of assumptions:

- **Both participants follow the protocol:** Both will compute every formula correct and will not insert or manipulate data or encryption keys. This allows us to conduct an analysis for core faults of the protocol, we could enforce this assumption by using trusted hardware or different communication approaches.

- **Encryption is considered secure:** No party can gain any information, besides the size, on a plain-text just from its cipher-text.

- **Transmissions are not altered:** Every message sent by a party will be received unchanged and unmanipulated by the other participant. We can ensure this by integrity protection mechanisms already known.

- **An eavesdropper can not learn anything:** Since the encryption is secure, an eavesdropper can not gain any useful information on the sent data.

- **The public key can be trusted:** The clients public key is needed for the server, we will assume the public key is transmitted securely and certified, so the server can check its authenticity.

Therefore, further attacking strategies as, for example forging database entries, replay attacks or man-in-the-middle attacks, will not be covered.

2.4 Fingerprint Based Authentication

There are two main ways of storing fingerprint data: minutiae and fingercode. Minutiae are certain special points of interest in human fingerprints such as sudden endpoints, branches or bridges. Minutiae can be thought of as anomalies in a locally unified field of parallel ridges and valleys. Figure 2 shows important examples of minutiae. [9] shows how minutiae triplets are saved for authentication and, if not protected otherwise, can be used to recreate an imprecise form of the original fingerprint. We will have a deeper look at fingercode based authentication, because it is used in the article proposing the protocol we will discuss in Section 3 and is widely used in computer science articles.

2.4.1 Fingercode based authentication

[6] presents a method for processing, storing and testing fingerprint data called fingercode. Minutiae points are interpreted as local anomalies in locally parallel ridges. The goal of fingercode is to extract this information, but not locate and process minutiae. Minutiae are causing the deviation in the last step. Transforming a fingercode has four basic steps:
Figure 2: Important examples of minutiae: a) A ridge ending, b) a ridge bifurcation, c) a short ridge and d) a bridge.

1) Determine an area of interest through a reference point: The point with maximum curvature is determined through an algorithm given by [6], that estimates the orientation field of ridges.

2) Tesselate the area of interest: The round area of interest is split in $45^\circ$ brackets. Every bracket is split in sectors. We want one ridge and valley pair or one detectable minutia per sector.

3) Filter the image with gabor filters: Every sector is normalized to a constant mean and variance, then banks of gabor filters are used in eight directions, for edge detection and noise reduction in the sectors.

4) Compute the feature values in every sector: The feature values are computed as the average absolute deviation from the mean (called AAD) of gray values and can be interpreted as gray values themselves.

The process is depicted in simplified form in Figure 3, the tessellated area of interest is marked blue, the point of reference red. The edge detection used in this figure is only exemplary and not done using gabor filters. Instead the inkscape implementation of the algorithm for edge detection by Canny [3] is used. The feature values are again exemplary and shown as gray disks.

When matching for authentication, we use a Euclidean distance and measured against a threshold $\tau$. Usually a system stores the same fingerprint five times at different angles to compensate for rotations or movement. If one of the five stored samples compares to the measured sample as smaller than $\tau$, the measured sample is accepted as belonging to the same fingerprint. To further increase the security, minutiae and filter based authentication can be combined.

Figure 3: Simplification of the Fingercode generation from [6], Fingerprint depiction from [10].
2.5 Homomorphic Encryption

According to [5], a partially homomorphic cryptosystem for an operation ⊙ is one, that fulfills for cipher texts \( \{a\} \) and \( \{b\} \) the following equation:

\[
\{a \odot b\} = \{a\} \odot \{b\}
\]

with an efficiently computable operation \( \odot \).

The intent of homomorphic encryption is to compute operations on encrypted data, without the need for decryption. We take addition as an example and try to compute

\[
\{3 + 7\} = \{3\} \odot \{7\}.
\]

We will use a basic encryption that can be computed as

\[
\{a\}_k = k \cdot a.
\]

Decryption can be done by division. We will use \( k = 7 \), it is obvious that \( \odot \) is also addition. We can now compute

\[
\{3\}_7 = 3 \cdot 7 = 21,
\]

\[
\{7\}_7 = 7 \cdot 7 = 49,
\]

\[
\{3\} \odot \{7\}_7 = \{3\}_7 + \{7\}_7 = 21 + 49 = 70.
\]

Thus the additive homomorphic property is shown. The multiplicative homomorphic property follows trivially as

\[
\{a\}^c = \{ca\},
\]

for a constant \( c \) from the additive homomorphic property:

\[
\{ca\} = \{a + \ldots + a\} = \{a\} \cdot \ldots \cdot \{a\} = \{a\}^c.
\]

3 Protocol by Barni et al.

In this Section we will present the protocol by Barni et al. presented in [2].

3.1 Function

For this algorithm we assume certain prerequisites:

- **Ids** are stored as \( i_d = 2^i \): The limitation on quantity of identifiers can be handled by using sets of identifiers. The steps in this Section are performed on those sets instead of the complete set of identifiers.
- The client is equipped with a fingerprint reader.
- The client has generated a Paillier key pair.
- The server knows the public key of the client: Secure key transmission is not part of this paper.

The process of authentication is briefly depicted in Figure 4.

3.1.1 Initialization and distance computation

First the client reads the fingerprint and performs a vector extraction, like the fingercode transformation, to receive the vector \( x \). He then computes \( \sum_{j=1}^{n} x_j^2 \), which is necessary for the server later on, and encrypts this sum as well as every component. Finally the client transmits \( \{x_1\}, \ldots, \{x_n\}, \{\sum_{j=1}^{n} x_j^2\} \) to the server.

2.6 Paillier encryption

Since we will use the Paillier cryptosystem [3] in Section 3 this paragraph will give a short introduction. The Paillier cryptosystem is an asymmetric and probabilistic cryptosystem. Let \( p, q \) be prime numbers that fulfill the equation

\[
gcd(pq, (p-1)(q-1)) = 1
\]

and \( n := pq \) is sufficiently big. Further set

\[
\lambda := \text{lcm}(p-1, q-1)
\]

and pick \( g \in (\mathbb{Z}/n^2\mathbb{Z})^* \) at random. The public key is \((n, g)\) and the private key is \(\langle p, q \rangle\).

2.6.1 Encryption

Let \( a \) be a message, chose \( r \) at random and calculate

\[
\{a\}_g = g^a r^n \mod n^2.
\]

1. gcd stands for the greatest common divisor.
2. lcm stands for least common multiple.

2.6.2 Decryption

A cipher-text \( \{a\} \) can be decrypted using the formula

\[
a = \{a\} \cdot \lambda \mod n^2.
\]

Whereas \( \lambda := (\frac{a^2 \mod n^2 \mod n}{n})^{-1} \). For proof of correctness refer to [8].

2.6.3 Homomorphic Property of the Paillier Cryptosystem

Let \((n, g)\) be the public Paillier key and \( r_1, r_2 \) be two random numbers. For texts \( a, b \) we calculate:

\[
\{a\} \odot \{b\} = g^{a r_1^n} n^2 g^{b r_2^n} n^2 = g^{(a + b)(r_1 r_2)^n} n^2 = \{a + b\}.
\]

This is by no means a secure system and only for demonstration purposes.

It is important to note, that this is done for certain operations, like addition or multiplication. A scheme that is homorphic for every operation is called fully homomorphic. The possibility of such a fully homomorphic scheme is shown in [5].
Let $m$ be the number of identities the server has stored and $n$ the number of elements in fingercode vectors. The server will now compute

$$\{D_i\} = \{\sum_{j=1}^{n} (x_j - y_{ji})^2\} = \{\sum_{j=1}^{n} x_j^2\} - 2\{\sum_{j=1}^{n} x_j y_{ji}\} + \{\sum_{j=1}^{n} y_{ji}^2\} = \{\sum_{j=1}^{n} x_j^2\} \prod_{j=0}^{n} (x_j)^{-2y_{ji}} \{\sum_{j=1}^{n} y_{ji}^2\}
$$

for every $i \in 1 \ldots m$. We have now encrypted every necessary distance.

### 3.1.2 The Bit-MIN Function

After distance computation the server invokes the bit-MIN protocol. Let $\{X\}, \{Y\}$ be vectors and $\{b\} = \text{bitMIN}(\{X\}, \{Y\})$, the bitMIN protocol ensures that the following equation is true:

$$b = 0 \iff X < Y.
$$

The bitMIN function calls another sub protocol called DGK, a protocol to compute the predicate bit of $d < r$ of two $n$ bit integers. As an auxiliary function of the protocol, the bit-MIN function is critical: It is important to keep the client from gaining knowledge on $X$ and $Y$ in this computation. Both, bit-MIN and DGK, require additional transmissions between client and server, which again are encrypted with the clients public key. In Figure 4 this is condensed to the “bitMIN” and “bitMIN helper” blocks. For further knowledge on those two sub protocols, refer to [1].

### 3.1.3 Authentication

Let $r$ be a random number. As a final step for single authentication (the information on presence of the fingerprint in the database) the server computes

$$\{R\} = r \{\sum_{i=1}^{m} b_i\} = \left(\prod_{i=1}^{m} \{b_i\}\right)^r.$$

$\{R\}$ is now transmitted to the client, which can decrypt it and check

$$f(R) = \begin{cases} \text{success} & \text{if } R \neq 0 \\ \text{failure} & \text{if } R = 0 \end{cases}$$

to gain knowledge on the success or failure of the identification.

### 3.1.4 Identification

To retrieve all identities a different approach is necessary for the final step. For this purpose the server computes

$$\{R\} = \{\sum_{i=1}^{m} b_i id_i\} = \prod_{i=1}^{m} \{b_i\}^{id_i}.
$$

and sends $\{R\}$ to the client. The client can again check $f(R)$ for simple authentication or check for an $id = 2^h$ with

$$g(R,h) = \begin{cases} \text{success} & \text{if } R_h = 1 \\ \text{failure} & \text{if } R_h = 0 \end{cases}$$

where $R_h$ is the $i$'th bit of $R$. Retrieving identities from $R$ can be done more efficiently by using de Bruijn sequences to index ones in a computer word, as shown by [7].

### 3.2 Efficiency

It is apparent that the complexity of a system using this protocol will be reduced, since former additions are transmorphed into multiplications and former multiplications times $a$ are transmorphed to raising the power to $a$. A short analysis of the protocol notation in [1] gives the estimated numbers of operations shown in Table 2. We can see, that the authentication process is linear in the number of identities in the database. But the protocol does need a lot more transmissions between

<table>
<thead>
<tr>
<th>Operation</th>
<th>Estimated count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplication</td>
<td>$O(mn \log n)$</td>
</tr>
<tr>
<td>Modulo</td>
<td>$O(mn)$</td>
</tr>
<tr>
<td>Addition</td>
<td>$O(mn)$</td>
</tr>
<tr>
<td>Exponentation</td>
<td>$O(mn)$</td>
</tr>
</tbody>
</table>
client and server; especially the transmissions for the bitMIN sub protocol are unnecessary while not using homomorphic encryption. Since Paillier encrypted texts are twice as long as its corresponding plain-text, there is a lot of transmission overhead. Let $n$ be the length of a fingercode vector and $m$ the number of database entries. The DGK sub protocol transmits $2n+1$ encrypted bits and the bitMIN protocol transmits one $101 + n$ bit long, encrypted integer. Since the bitMIN protocol calls the DGK protocol exactly once, there is an overhead of at least $m(6n + 202)$ bits on the bitMIN computation. Special interest is given to the complexity and load for the client. A more detailed list of operations done by the client is shown in Table 3. We can see the protocol causing noticable workload for the client.

### 3.3 Security

Since the Euclidean distance is still accurately computed and compared to the threshold $\tau$ the accuracy of the system remains the same. Assuming the cryptosystem is semantically secure, the server can not retrieve any information about the fingerprint transmitted. The server will not even know if the fingerprint is present in the database, since every message the server receives is encrypted by the client. The case for the client is harder to ascertain, since he can decrypt every message he gets. The client gets to know the values $b_i$ without knowing the $i_i$, since the server scrubbed the identities randomly. In an honest-but-curious setting of single authentication, the client might gain knowledge on authentication early, but can not get any information on the identities. The sub protocols used are secure in an information theoretical way: Any data sent is indistinguishable from uniformly distributed random data, according to [2]. Both variants, the one for single authentication and the one for retrieving all identities, enjoy the same level of security.

### 4 Discussion

We saw that the requirements we stated in Section 2.2 are not all fulfilled by the Barni et al. protocol. Even though partially homomorphic encryption can address certain privacy problems in biometrics, it is in itself not a solution to every problem. Especially the weaknesses of the protocol shown in Section 3 need to be considered, before using it.

### 4.1 Limitations

The server does still need to store an unencrypted representation of the fingerprint for this protocol. The unencrypted representation can still be used to combine information from different systems and track registered users, even though their authentication can not be tracked. In general the user has no control over the usage of his biometric data, so the server can use the data in unauthorized and unwanted ways. Another issue not addressed by this protocol is revocability: If the fingerprint gets compromised or stolen, a valid user can not mark it as such and register a new fingerprint, but some ways of revocable biometrics are compatible with this protocol, as long as the Euclidean distance is not impaired. In addition this protocol needs a public key infrastructure using Paillier keys. This infrastructure makes clients complex and the user responsible to have a usable client, since a public tool can hardly be entrusted with a private key; otherwise the clients has its own key pair and has to be trusted by the user as a system. A possible solution might be one-time keys, that are generated when needed. The attacking model honest-but-curious is a massive simplification of real world attacking scenarios and the protocols have to be checked for further weaknesses through replays or other attacks.

### 4.2 Alternatives

We need different protocols for biometric data, since this protocol only supports Euclidean distance authentication, which is not used in all fields of biometry authentication, for example face recognition [4].

#### 4.2.1 Biohash

Another way of preserving privacy is biohashing. Biohashing is a two-factor authentication: It tries to transform biometric data using a token or password. The transformation is then stored on the server. If compromised, the token is changed and the server data is renewed. This has certain advantages: The server has no direct access to the true biometric traits and can not connect them over different services (assuming different passwords or tokens). If the transformation is compatible to a Euclidean distance measurement, biohashes can be combined with the presented protocol by Barni et al. and thus enhance privacy for users. This would add another fulfilled requirement.

### 5 Conclusion

Privacy preserving in biometrics is still a problematic topic. The shown protocol can solve some of the problems, for example keeping the knowledge for both sides at a minimum, but brings in new ones, like a public key infrastructure. There is still a heavy need for further research, especially further checks of security, but partial homomorphic encryption and biohashing are important approaches. On its own, those approaches can not
cover every problem, e.g. non revocability or untrusted matching environment. But combinations might make protocols too complex and slow to use.

REFERENCES


