POSTER: Can We Use Biometric Authentication on Cloud?: Fingerprint Authentication Using Homomorphic Encryption

Taeyun Kim
Sungkyunkwan University
Suwon, Republic of Korea
taeyun1010@skku.edu

Hyoungshick Kim
Sungkyunkwan University
Suwon, Republic of Korea
hyoung@skku.edu

ABSTRACT

Even though biometric authentication such as fingerprint authentication is popularly used, there are few network services supporting biometric authentication because many users have serious privacy concerns about leakage of the biometric data on a server. For example, in fingerprint authentication, a user's raw fingerprint is typically stored in plaintext form. Unlike conventional text passwords, we cannot use a cryptographic hash function such as SHA-256 because it is very hard to obtain the same fingerprint features even when the exactly same finger is scanned multiple times. In this paper, we present a fingerprint authentication mechanism using the Fastest Homomorphic Encryption in the West (FHEW) library for Learning With Errors (LWE) scheme. Our implementation allows us to securely store fingerprint data on a network server using homomorphic encryption to calculate the distance between two encrypted fingerprint data without decrypting them. To show the efficiency of our implementation, we used "NIST special database 9" containing 4,000 fingerprint samples. Our results show that two fingerprints can be efficiently compared with about 209 seconds on average even when they are securely stored in encrypted form.

CCS CONCEPTS

• **Security and privacy → Cryptography**; *Biometrics*; *Privacy-preserving protocols*;

KEYWORDS

Homomorphic encryption, FHEW, fingerprints, user authentication

ACM Reference Format:
Taeyun Kim and Hyoungshick Kim. 2018. POSTER: Can We Use Biometric Authentication on Cloud?: Fingerprint Authentication Using Homomorphic Encryption. In *ASIA CCS '18: 2018 ACM Asia Conference on Computer and Communications Security, June 4–8, 2018, Incheon, Republic of Korea.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3196494.3201585

1 INTRODUCTION

Fingerprint authentication is popularly used for mobile devices only because storing a raw fingerprint or biometric template in plaintext form is generally required for fingerprint matching algorithms. To overcome this limitation, we apply a homomorphic encryption algorithm to fingerprint authentication.

Homomorphic encryption [2] is an encryption scheme which allows computations to be held on encrypted data. When one decrypts the result of such computations, the result is guaranteed to be the same as the result of performing the same computations on plaintext data. Both partially homomorphic encryption scheme and fully homomorphic encryption scheme exist. A partially homomorphic encryption scheme only allows some particular computations to be held on encrypted domain, whereas a fully homomorphic encryption scheme allows any arbitrary computations to be held on encrypted domain.

In this paper, we introduce an implementation of fingerprint authentication using a fully homomorphic encryption scheme to securely protect users' biometric data stored on a remote server. We used the Fastest Homomorphic Encryption in the West (FHEW) homomorphic encryption library since it is well-known for efficient bootstrapping that can be performed within 0.1 seconds [1]. For fingerprint authentication, we used the filterbank-based fingerprint matching algorithm [3]. We found that this algorithm can be working efficiently because it requires the simple calculation of the Euclidean distance between the two corresponding fingercodes.

To implement the filterbank-based fingerprint matching algorithm using the fully homomorphic encryption scheme, we extend the FHEW library to support the encryption of integers of arbitrary bit length. The existing FHEW library can support 5 basic logic gate operations (AND, OR, NAND, NOR and NOT) only. To show the feasibility of our implementation, we used "NIST special database 9" containing 4,000 fingerprint samples. The experiment results show that two fingerprints can be efficiently compared within 209 seconds on average even for the encrypted fingerprint data.

2 METHODOLOGY

2.1 Fingerprint matching algorithm

The filterbank-based fingerprint matching, or fingercode fingerprint matching algorithm, works as follows. First, a reference point of a given fingerprint is located. Here, a reference point is defined as a point of maximum curvature of the concave ridges in the fingerprint image [3]. Next, fingerprint image pixels around the reference point are divided. This removes any translation variant that given fingerprint image might have since region of interest then becomes pixels that are relative to the reference point. Then, the region of interest around reference point is Gabor-filtered and the deviation from the mean gray values are calculated for all sectors. Typically, multiple fingercodes having the values from rotated fingerprints are stored for each fingerprint to get rid of any rotation variant given

ASIA CCS '18, June 4–8, 2018, Incheon, Republic of Korea

ACM ISBN 978-1-4503-5576-6/18/06.
https://doi.org/10.1145/3196494.3201585

fingerprint might have. These fingercodes, or feature vectors of a fingerprint, can be used to determine if two fingerprints originated from the same person. Readers who are interested in more detail should refer to [3].

2.2 Protocol that uses encrypted fingerprints

To utilize the fully homomorphic encryption scheme in fingerprint authentication, the server must store encrypted form of fingerprint and must be able to decide whether to authenticate the user or not when given only two encrypted fingerprints. The input fingerprint the user who would like to be authenticated provides must be homomorphically encrypted and sent to the server. Since homomorphic encryption uses secret key to encrypt and decrypt plaintexts and uses evaluation key (which is not the same as secret key) to do operations on encrypted ciphertexts, the fingerprint data that user inputs can be encrypted using his secret key at the client side. The server, who does not have client's secret key, can only perform calculations on encrypted data using evaluation key. After performing the desired calculations (in this case calculations that would result in the encrypted form of distance between two encrypted fingerprint data), the server can send the encrypted distance back to the client side, where the result can be decrypted to decide whether or not to authenticate the user. The protocol is presented in Figure 1.

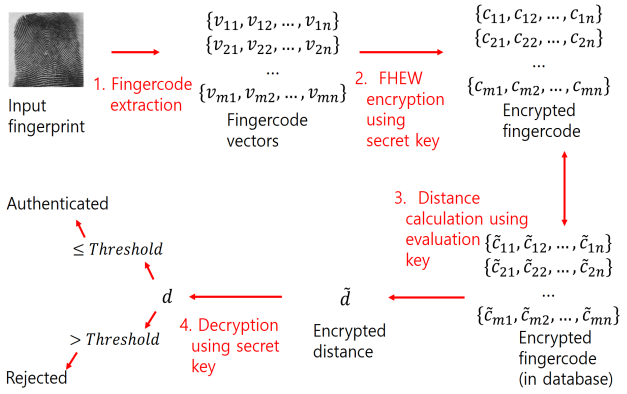

Figure 1: FHEW embedded in filterbank-based fingerprint matching.

Algorithm 1: addition

```
Input: EvalKey, CipherTexts1[], CipherTexts2[],Initialcarry
Output: Output[]
for int i : 0 to (numberOfBits-1) do
    if i==0 then
        [SumOfBits, Carry] = addBit(EvalKey, CipherTexts1[i],
          CipherTexts2[i],Initialcarry);
    else
        [SumOfBits, Carry] = addBit(EvalKey, CipherTexts1[i],
          CipherTexts2[i], Carry);
    end
    Output[i] = SumOfBits;
end
```

Algorithm 2: addBit

```
Input: EvalKey, EncryptedBit1, EncryptedBit2,CarryIn
Output: SumOfBits, CarryOut
temp1 = HomAND(EvalKey, EncryptedBit1, EncryptedBit2);
temp2 = HomXOR(EvalKey, EcnryptedBit1, EncryptedBit2);
temp3 = HomAND(EvalKey, temp2, CarryIn);
SumOfBits = HomXOR(EvalKey, temp2, CarryIn);
CarryOut = HomOR(EvalKey, temp1, temp3);
```

2.3 Extension of FHEW library

FHEW library, which is an open source library written in C++, is available on Github.[1] It provides an implementation of fully homomorphic encryption presented in [1]. The library only allows encryption of individual bit. Everytime one bit is encrypted, the code results in C++ struct named CipherText which contains an array of 500 integers and 1 integer. Encryption of one bit requires a secret key which is an array of 500 integers. Performing gate computations on encrypted bits requires an evaluation key which consists of a bootstrapping key and a switching key. Bootstrapping key is a 3-dimensional array of 500 * 23 * 2 integers, and switching key is a 3-dimensional array of 1024 * 25 * 7 integers. We extended FHEW library to enable encryption of integers of arbitrary bit length. The library also enables computation to take place on encrypted bits, although it provides support for only 5 basic logic gates – AND, OR, NAND, NOR and NOT. To calculate the distance between homomorphically encrypted fingercode templates, we implemented more complex calculations on encrypted domain based on these gates. Since we were required to compute a norm distance between vectors, we implemented a support for XOR-gate operation, addition, subtraction, and absolute value calculation on encrypted domain. Implementing these operations using 5 provided logic gates is straightforward, and pseudocode for addition of integers and bitwise-addition are shown in Algorithms 1 and 2. We omit how we implemented other XOR-gate operation, subtraction and absolute value calculation used for the Euclidean distance calculation but they can also be easily implemented in a similar manner. Since calculating a two-norm distance on encrypted domain turned out to be too computationally heavy and resulted long authentication time that would make the scheme unusable, we relaxed the filterbank-based fingerprint matching algorithm and calculated 1-norm distance instead of 2-norm distance. We also rounded fingercode vectors of doubles to vectors of integers to further save computation time.

2.4 Fingerprint verification on encrypted domain

We used an open source implementation of filterbank-based fingerprint matching which was available on Github[2]. We used 8 number of bands and 8 number of sectors (see [3] for details) which resulted in fingercode vectors having 16 doubles each. Since we rounded doubles to integers and used 10 bits for representing an integer, we ended up with 16*10*(500+1) = 80160 integers representing each

[1] https://github.com/lducas/FHEW
[2] https://github.com/hbhdytf/fingercode

fingercode vector. For the fingerprint dataset, we used "NIST special database 9", which contains fingerprint images of 2000 different individuals. The image size is 832 * 768 pixels, and the images are scanned at 500 dpi. The dataset contains 2 different fingerprint instances of the same finger per individual, resulting in a total of 4000 fingerprints. In order to calculate an appropriate threshold for distance that can be used to decide whether or not to authenticate a user given his fingerprint, we measured a 1-norm distance between two different instances of fingerprint of the same finger. To check how much distance gets larger when fingerprint of wrong individual is used for verification, we also measured a 1-norm distance between i-th person's fingerprint and (i+1)th person's fingerprint. Using these 2 sets of 2000 different distances, we could determine a suitable threshold for "NIST special database 9" that can maximize true positive rate while minimizing false positive rate. We then compared true positive rates and false positive rates of 1-norm distance encrypted fingerprint authentication scheme to those of 2-norm distance scheme (the original filterbank-based fingerprint matching algorithm). While we mainly tested our scheme on the specified setting, we also tested on different settings such as fingerprint vectors of length 640 doubles.

3 RESULTS

3.1 Matching performance

We used an Intel Core i5-7500 CPU with 3.40GHz * 4 and 64-bit Ubuntu 16.04 LTS. As expected, our extension of FHEW library calculates exact 1-norm distance given two encrypted fingercode vectors since FHEW library is fully homomorphic. Hence, if any degrade in performance was introduced, it must came from the relaxation that we introduced by rounding vectors of doubles to vectors of integers and calculating 1-norm distance instead of 2-norm distance. However, under the setting we tested, the result shows that such degrade in performance is negligible. We plotted FAR (False Acceptance Rate) and FRR (False Rejection Rate) versus threshold for both our relaxed version of 1-norm distance and original version of 2-norm distance. The plots are shown in Figures 2 and 3. By carefully setting the threshold value, we could achieve comparable performance to the original 2-norm distance algorithm. For example, setting 1-norm distance threshold to 100 gave false negative of 1621, true positive of 379, false positive of 134, and true negative of 1865. Setting 2-norm distance threshold to 32.14 gave false negative of 1633, true positive of 367, false positive of 134, and true negative of 1865.

3.2 Time performance

Under our setting, executing one of 5 gate computations that FHEW library provides took an average of 0.1934 seconds. Homomorphic XOR gate computation took an average of 0.5594 seconds, just about 3 times longer than one of 5 basic gate computations since 3 gate computations were required to execute XOR gate computation. Encrypting fingercode vector was almost instantaneous. Encryption of 640-integer length fingerprint vector took an average of 0.02393 seconds. Given two encrypted fingercode vectors of length 16, time to calculate 1-norm of difference between two vectors required an average of 209 seconds, meaning an average of 209 seconds before the user could be authenticated. We have also verified that

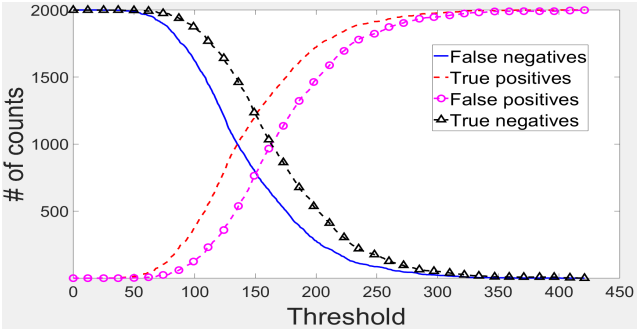

Figure 2: FARs and FRRs versus threshold when 1-norm distance is used, rounded doubles to integers.

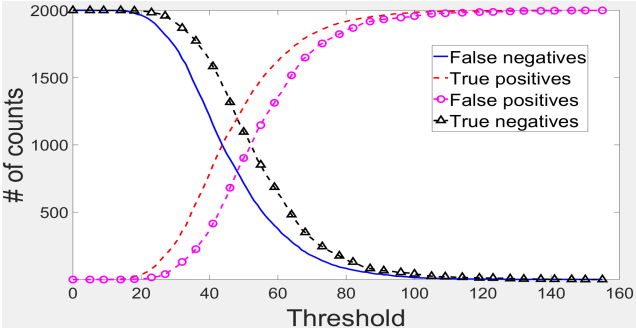

Figure 3: FARs and FRRs versus threshold when 2-norm distance is used.

time taken to compute 1-norm distance between two vectors grows linearly as vector length increased.

4 CONCLUSION

In this paper, we extended FHEW library to calculate a 1-norm distance between two encrypted fingercodes. We then applied extended library to the filterbank-based fingerprint matching algorithm. Even though it takes relatively longer time (about 209 seconds) than conventional matching algorithms without encryption, this authentication time could be acceptable for some applications such as user registration and user identification services without requiring real-time processing. For future work, we will extend our implementation based on FHEW to cover various algorithms for biometric data.

ACKNOWLEDGMENTS

This research was supported in part by the MIST (2015-0-00914) and the ICT R&D program (No.2017-0-00545).

REFERENCES

[1] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. 2016. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 3–33.

[2] Craig Gentry. 2009. *A Fully Homomorphic Encryption Scheme.* Ph.D. Dissertation. Stanford, CA, USA. Advisor(s) Boneh, Dan. AAI3382729.

[3] Anil K Jain, Salil Prabhakar, Lin Hong, and Sharath Pankanti. 2000. Filterbank-based fingerprint matching. *IEEE Transactions on Image Processing* 9, 5 (2000), 846–859.