Biometric Information in Digital Signing and Non-Repudiation

Nathanael Santoso – 13520129
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: 13520129@std.stei.itb.ac.id

***Abstract*—In the dynamic landscape of digital security, the imperative need for robust and dependable authentication mechanisms has surged. Traditional forms of authentication, such as passwords and PINs, susceptible to prevalent cybersecurity threats like social engineering, have catalyzed a shift towards biometric authentication, leveraging intrinsic aspects of individuals for heightened security. Among biometric methods, fingerprint recognition is the most popular due to its blend of convenience and security. This paper explores the fusion of fingerprint minutiae with cryptographic techniques, particularly the Schnorr Scheme, which presents challenges due to the precise key requirements conflicting with the variability inherent in fingerprint minutiae. However, recent advancements like Fuzzy Extraction and Secure Sketches offer promising solutions, enabling the generation of stable cryptographic keys despite minute variations in samples. Fuzzy Extraction in particular piques interest as it is reusable for known public parameters thus allowing non-repudiation by virtue of public data availability. This paper proposes a pipeline combining Fuzzy Extraction with the Schnorr Digital Signature Scheme, to develop a reusable, noise-tolerant verification scheme, advancing the frontier of secure digital authentication.**

***Keywords*—Biometric Signing, Fingerprint Authentication, Fuzzy Extraction, Schnorr Scheme**

I. Introduction

In the evolving landscape of digital security, the need for robust and reliable authentication mechanisms has become increasingly important. Traditional passwords, PINs, and passkeys, while widely used, are susceptible to the most common cybersecurity threat, social engineering, prompting a shift towards alternatives which leverage other facets of our being such as biometric authentication. Unlike the aforementioned, which is something we possess, biometric authentication is something that we are and is harder for an adversary to acquire non-destructively [2]. Among the various biometric techniques available currently, fingerprint recognition is the most widely adopted due to its high convenience factor in addition to the security provided. In this paper, we will discuss advancements surrounding digital signatures, non-repudiation, and minutiae-based Cryptography.

The integration of fingerprint minutiae with cryptographic techniques such as the Schnorr Scheme comes with a few challenges. The Schnorr Scheme, for example, requires a set of asymmetric keys that can be computed with absolute certainty which prerequisites that part of the scheme be constant [3]. Fingerprint minutiae, however, are difficult to accurately measure, and slight differences often appear between different samples of the same fingerprint. When combined with the uniqueness of the patterns of fingerprint minutiae, this significantly exacerbates the difficulty in obtaining stable cryptographic keys. However, despite this challenge, various methods have sprung up in recent years such as Fuzzy Extractors and Secure Sketches which can generate unique keys despite minute differences in samples [4].

Fuzzy extraction, in particular, allows us to obtain a certain key given that the Hamming Distance between samples does not exceed a certain threshold. Furthermore, this algorithm can obfuscate the relationship between the sample and the key making it quite difficult for an adversary to break. This particular property results in a low likelihood of forgery thus enabling non-repudiation to be established [1]. In this paper, we will be combining the use of Fuzzy Extraction and the Schnorr Digital Signature Scheme to construct a reusable noise-tolerant verification scheme.

II. Preliminaries

A. Biometric Authentication

Biometric authentication is a key solution in digital security, utilizing unique physiological or behavioral traits like fingerprints, facial features, iris patterns, and voice recognition to verify identities. The authentication method itself is based on the fact that there are very certain, extractable, unique differences between each individual, enough so that identification becomes possible. In the case of Fingerprint authentication. A scanner will collect multiple samples of your fingerprint and construct a list of common minutiae that are unique to each individual [2].

A typical authentication scheme will often request a sample of your fingerprint which can then be compared to the pre-established minutiae. If a certain threshold of matches is met, the algorithm will authenticate the user. This method of authentication is acceptable in most use cases, however digital signing requires more than just the approximation of sampled minutiae to reference minutiae, but also requires a generated cryptographic key. Then numbers used to generate these cryptographic keys are often constant and do not change (i.e. Read Only) contrasting biometric data that can often change slightly depending on real life situations.

B. Non-Repudiation

Non-repudiation is a crucial security principle that ensures a party in a digital communication cannot deny the authenticity of their signature or the sending of a message, thereby providing undeniable proof of the origin and integrity of the data. This concept is fundamental in various applications, including legal contracts, financial transactions, and secure communications, as it prevents individuals from reneging on their commitments or disputing the validity of their actions.

C. Digital Signing

Digital Signing is a cryptographic process that ensures the authenticity and integrity of digital messages or documents, making it an essential component of modern cybersecurity. This technique employs a pair of keys, one public and one private, to create a unique digital signature that verifies the origin and confirms that the content has not been altered during transmission. This also prevents the signer from reneging on the signed document since the public key is proof that the document has been undeniably signed by the signer assuming the private key used to sign the document is not compromised. However, if the private key is indeed compromised, non-repudiation is broken and cannot be asserted. The application of biometric data in digital signing on the other hand guarantees without a doubt, save serious bodily harm, the authenticity of a document since it cannot be stolen. Extracting a constant value from a person's being remains the only challenge left in idealizing the aforementioned.

D. Minutiae

Fingerprint minutiae are the detailed ridge characteristics found in a fingerprint used to distinguish one finger from another. These minutiae points include ridge endings, where a ridge abruptly terminates, and bifurcations, where a single ridge splits into two, as well as other features like dots, islands, and lakes shown by Fig. 1.

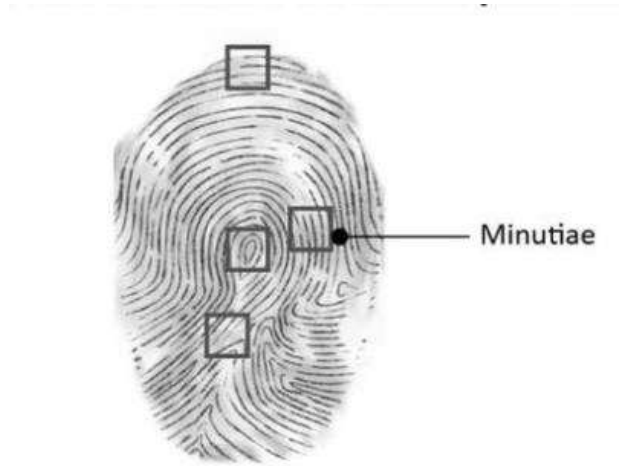


Fig. 1. Minutiae (source: adapted from wikimedia.org)

The uniqueness of a fingerprint is largely derived from the specific pattern and distribution of these minutiae points, which remain consistent throughout a person's life despite minor injuries or changes in the skin. Technologically, capturing and analyzing fingerprint minutiae involves advanced image processing techniques that convert the analog patterns into digital data, enabling precise matching and identification. The robustness of fingerprint recognition systems depends on the accuracy with which these minutiae can be detected and compared, making them a critical component in biometric authentication systems.

E. Fuzzy Extraction

Fuzzy extraction is a cryptographic technique designed to securely generate reproducible cryptographic keys from noisy or imprecise biometric data. This method addresses the inherent variability in biometric inputs, such as fingerprints or voice patterns, which can differ slightly with each capture due to factors like environmental conditions or sensor inconsistencies.

Fuzzy extraction operates by first transforming the biometric input into a stable, error-tolerant format, often involving error correction codes that allow for slight deviations in the input data. It then extracts a cryptographic key from this stabilized representation, ensuring that the same key can be reproduced from different but similar biometric readings.

This process, illustrated by Canetti Et. Al. in Fig. 2 involves two steps, generation and reproduction where generation produces a key from biometric information and an error correction helper. This helper is then reused by the reproduction function each time the key is to be derived [1].

Gen

1. *Input:* $w = w_1, ..., w_n$
2. *Sample* $r \xleftarrow{\$} \{0,1\}^\kappa$.
3. *For* $i = 1, ..., \ell$:
   (i) *Choose random* $1 \le j_{i,1}, ..., j_{i,k} \le n$
   (ii) *Set* $v_i = w_{j_{i,1}}, ..., w_{j_{i,k}}$.
   (iii) *Set* $c_i = \mathsf{lock}(v_i, r)$.
   (iv) *Set* $p_i = c_i, (j_{i,1}, ..., j_{i,k})$.
4. *Output* (r, p), *where* $p = p_1 \dots p_\ell$.

Rep

1. *Input:* $(w' = w'_1, ..., w'_n, p = p_1 \cdots p_\ell)$
2. *For* $i = 1, ..., \ell$:
   (i) *Parse* $p_i = c_i, (j_{i,1}, ..., j_{i,k})$.
   (ii) *Set* $v'_i = w'_{j_{i,1}}, ..., w'_{j_{i,k}}$.
   (iii) *Set* $r_i = \mathsf{unlock}(v'_i, c_i)$. *If* $r_i \neq \perp$ *output* r_i.
3. *Output* $\perp$.

Fig. 2. Generate and Reproduce Construction [1]

F. Schnorr Digital Signing Scheme

The Schnorr Digital Signature Scheme is a cryptographic protocol used to generate secure digital signatures, characterized by its efficiency and simplicity. It operates within the framework of modular arithmetic and relies on the hardness of the discrete logarithm problem for its security. In this scheme, the signer generates a key pair consisting of a private key and a corresponding public key [3]. To create a signature, the signer computes a commitment using a randomly chosen nonce and then generates a challenge by hashing the message and the commitment together. The final signature is produced by combining the nonce, the private key, and the challenge in a specific manner. Verification of the signature involves checking the integrity of the commitment and ensuring that it aligns with the public key and the hashed message. Due to its compactness and reduced computational overhead, the Schnorr Digital Signature Scheme is well-suited for applications requiring efficient and secure authentication, such as blockchain technologies and secure communications.

III. IMPLEMENTATION

A. Signature Pipeline Design

This paper aims to create a fully functioning pipeline which can take in a document and a fingerprint and successfully generate a signature. To achieve the criteria above, the following functions must be present within the pipeline in sequential order:

1. The ability to parse a fingerprint sample into its minutiae.
2. The ability to encode the minutiae into a form of data that can be used by the Fuzzy Extraction algorithm.
3. The ability to obtain a private key by means of Fuzzy Extraction
4. The ability to generate a Schnorr Scheme and a public key from said private key.
5. The ability to parse, sign, and verify a document with the public and private key scheme.

B. Concretion

1. Fingerprint Minutiae Parsing

Fingerprint Minutiae Parsing is done with the help of an image processing library. The library will take in images and find a frame in which the fingerprint is fitted. Then the minutiae are characterized based on the required fields in the ISO 19794-2 fingerprint template format. These fields are as follows:

- Type: Which is 2 bits determining whether it is a ridge ending, valley bifurcation, ridge bifurcation or unknown.
- X Coordinate: Which is 14 bits relative to the start of the fingerprint frame from the X-axis. (The frame which is determined by the bounding box that can surround the entire fingerprint)
- Y Coordinate: Which is 14 bits relative to the fingerprint frame from the Y-axis.
- Angle: Which is 8 bits representing direction which can be calculated by dividing the degree angle from the cardinal direction by 360 degrees and multiplying it by 256 before subtracting by 1 and vice versa for conversion to degrees. This results in an error rate within 3-degree margins.

2. Minutiae 32-bit Encoding

Minutiae are then parsed to fit into 32 bits based on the struct illustrated in Fig. 3., the X and Y coordinates are shortened by 3 bits as most fingerprint images are within the bounds of 2048 x 2048 pixels (the upper limit of 2^{11}).

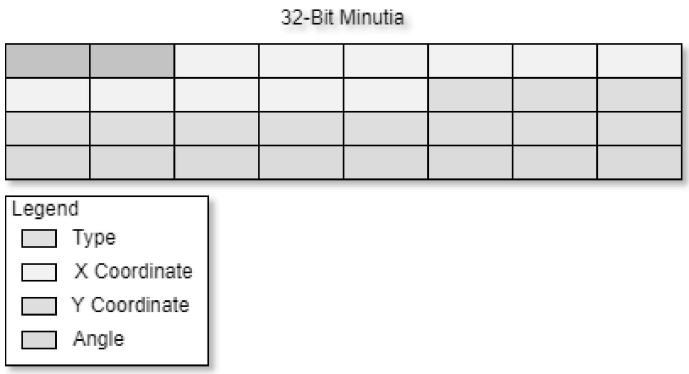


Fig. 3. Diagram of Minutia Encoding

3. Fuzzy Extraction Algorithm

The parsed minutiae are then fed sequentially into the Fuzzy Extraction Algorithm that can be seen in the code snippet below. This is a direct implementation of the work of Canetti Et. Al. in the paper "*Reusable Fuzzy Extractors for Low-Entropy Distributions*". The Definitions for both functions can be found in Fig. 2. The general idea behind these functions is:

- Gen: For an input string, we compute a random key such that the key can be used to lock the input string using a hash function. The lock uses a hash function that utilizes nonces and security bits to obfuscate correlation. The input value is masked and passed through the hash function to result in a digest which is then encrypted by the key. The key obtained is the Private Key, while the ciphers, masks, and nonces are the Reproduction helpers.

```
func (fz *fuzzy32extractor) Gen(value string)
(Key, *Helpers[uint32], error) {
val, err := hex.DecodeString(value)
if err != nil {
return "", nil, err
}

if len(val) != fz.blockLength * 4 {
return "", nil,
errors.New("gofze/lib/fuzzy32.go: invalid
value length")
}

val32 := make([]uint32, fz.blockLength)
binary.Read(bytes.NewReader(val),
binary.BigEndian, &val32)

key := make([]byte, fz.blockLength * 4)
key32 := make([]uint32, fz.blockLength)
rand.Read(key)
binary.Read(bytes.NewReader(key),
binary.BigEndian, &key32)

pad := make([]uint32, fz.securityLength)
keyPad := append(key32, pad...)

nonces  := make([][]uint32, fz.numHelpers)
masks   := make([][]uint32, fz.numHelpers)
vectors := make([][]uint32, fz.numHelpers)
digests := make([][]uint32, fz.numHelpers)
ciphers := make([][]uint32, fz.numHelpers)
```

```
for i := range fz.numHelpers {
nonce8 := make([]byte, fz.nonceLength * 4)
nonces[i] = make([]uint32, fz.nonceLength)
rand.Read(nonce8)
binary.Read(bytes.NewReader(nonce8),
binary.BigEndian, &nonces[i])

mask8 := make([]byte, fz.blockLength * 4)
masks[i] = make([]uint32, fz.blockLength)
rand.Read(mask8)
binary.Read(bytes.NewReader(mask8),
binary.BigEndian, &masks[i])

vectors[i] = make([]uint32, fz.blockLength)
for j := range fz.blockLength {
vectors[i][j] = val32[j] & masks[i][j]
}
vector8 := new(bytes.Buffer)
binary.Write(vector8, binary.BigEndian,
&vectors[i])

digest8 := pbkdf2.Key(vector8.Bytes(),
nonce8, 1, (fz.blockLength +
fz.securityLength) * 4, fz.hash)
digests[i] = make([]uint32, fz.blockLength +
fz.securityLength)
binary.Read(bytes.NewReader(digest8),
binary.BigEndian, &digests[i])

ciphers[i] = make([]uint32, fz.blockLength +
fz.securityLength)
for j := range fz.blockLength +
fz.securityLength {
ciphers[i][j] = digests[i][j] ^ keyPad[j]
}
}

return Key(hex.EncodeToString(key)),
&Helpers[uint32]{
ciphers: ciphers,
masks: masks,
nonces: nonces,
}, nil
}
```

- Rep: For an input string that is different slightly from the initial input string and a precomputed helper, we can unlock the key by reversing the gen process and making it such that at least one helper approximates the initial input string. This results in the key being revealed when we perform the XOR to the ciphers. The initial key is recoverable if at least one helper function approximates it by bit-masking such that the more helpers there are, the more fault tolerant the algorithm is. Choosing the correct number of helpers is thus crucial for the security of this algorithm.

```
func (fz *fuzzy32extractor) Rep(value
string, helper *Helpers[uint32]) (Key,
error) {
val, err := hex.DecodeString(value)
if err != nil {
return "", err
}

if len(val) != fz.blockLength * 4 {
return "",
errors.New("gofze/lib/fuzzy32.go: invalid
value length")
}

val32 := make([]uint32, fz.blockLength)
binary.Read(bytes.NewReader(val),
binary.BigEndian, &val32)

ciphers := helper.ciphers
masks := helper.masks
nonces := helper.nonces
vectors := make([][]uint32, fz.numHelpers)
digests := make([][]uint32, fz.numHelpers)
plains := make([][]uint32, fz.numHelpers)

for i := range fz.numHelpers {
vectors[i] = make([]uint32, fz.blockLength)
for j := range fz.blockLength {
vectors[i][j] = masks[i][j] & val32[j]
}
vector8 := new(bytes.Buffer)
binary.Write(vector8, binary.BigEndian,
&vectors[i])

nonce8 := new(bytes.Buffer)
binary.Write(nonce8, binary.BigEndian,
&nonces[i])

digest8 := pbkdf2.Key(vector8.Bytes(),
nonce8.Bytes(), 1, (fz.blockLength +
fz.securityLength) * 4, fz.hash)
digests[i] = make([]uint32, fz.blockLength
+ fz.securityLength)
binary.Read(bytes.NewReader(digest8),
binary.BigEndian, &digests[i])

plains[i] = make([]uint32, fz.blockLength +
fz.securityLength)
for j := range fz.blockLength +
fz.securityLength {
plains[i][j] = digests[i][j] ^
ciphers[i][j]
}
if sum(plains[i][fz.blockLength:]...) == 0
{
plain8 := new(bytes.Buffer)
binary.Write(plain8, binary.BigEndian,
plains[i][:fz.blockLength])
return
Key(hex.EncodeToString(plain8.Bytes())),
nil
}
}

return "",
errors.New("gofze/lib/fuzzy32.go: unable to
reproduce key")
}
```

4. Schnorr Scheme Generation

The Schnorr Scheme Generation Library used is taken from the author's past work. However, this work modifies it slightly in that the Private key is no longer randomly generated but passed on by the Extraction Algorithm. The public key can be derived from Primes P, Q, and generator G if the private key is less than Q.

5. Document Signing and Verification

We will then read the document into a byte array and pass it into the Schnorr Signing scheme with the Private Key which will generate a Signature and a Hash which can be used alongside the public key to verify authenticity and intent.

C. Caveats

A few caveats are present within this implementation which are:

- Fingerprint feature extraction only uses a single picture as a reference resulting in incomplete minutiae.
- The minutiae extraction library used is immature and features are often missing, however due to time constraints, it is impossible to implement independently. There is no other option for the minutiae extraction library for the chosen language.
- Minutiae are fed sequentially into the fuzzy extraction algorithm, as a result, typically the order of the fingerprints matters in determining the outcome of the fuzzy extraction. The downside regarding this is that the correctness is heavily influenced by the angle of the reference picture.

IV. Evaluation

A. Fingerprint Noise Addition

To test this against small noise, we used a smudging factor on the reference case minutiae where each case would be smudged by a certain number of bits where smudging refers to the act of flipping random bits up to n times. Table 1. Below shows the breaking point until a certain amount of bits n have been fudged.

Table 1. N to Failure Table

N	Verification Status
1	Success
2	Success
3	Success
4	Success
5	Success
6	Success
7	Success
8	Success
9	Failed

This shows that the algorithm is safe up to 8 bits of tolerance, depending on how many helpers there are, tolerance can grow however, since each helper is very large, the spatial growth of this algorithm is also very large and can require a lot of computational time depending on the number of minutiae and helpers such that there is a tradeoff between error tolerance and spatial growth of helpers.

In addition to the above, since there is randomness in choosing the masks in the helpers, results may vary between trials with some trials having a lot of success while others not so much.

B. Real World Cases

Test cases using real world metrics have been proven unsuccessful using this pipeline. Using Test Cases Taken from the Fingerprint Verification Competition (FVC) 2002 Dataset Packet B_1 (source: http://bias.csr.unibo.it/fvc2002/Downloads/DB1_B.zip) in Table 2. it shows that most attempts to verify the signature failed.

Table 2. Real World Case Evaluation Result

Reference Case	Case 3_4
Case 3_1	Not Enough Minutiae
Case 3_2	Not Enough Minutiae
Case 3_3	Not Enough Minutiae
Case 3_5	Unsuccessful
Case 3_6	Not Enough Minutiae
Case 3_7	Not Enough Minutiae
Case 3_8	Not Enough Minutiae

C. Further Development

Based on the tests done, it is recommended that the following be done in extension to this paper:

- Test the impact of the change in parameters of the Fuzzy Extractor.
- Devise a better distance metric than the Hamming Distance for Point Data such as minutiae.
- Devise a proper minutiae extraction and set difference inference in order to increase success rate in real world scenarios.

Source Code

The source code for the various projects that contributed to the creation of this pipeline can be found below:

- Pipeline:
  https://github.com/nart4hire/gofze
- Schnorr Scheme:
  https://github.com/nart4hire/goschnorr
- Minutiae Extraction:
  https://github.com/nart4hire/fingerprints

REFERENCES

[1] Canetti, R., Fuller, B., Paneth, O., Reyzin, L., Smith, A.D.: Reusable fuzzy extractors for low-entropy distributions. In: Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 117–146 (2016). doi:10.1007/978-3-662-49890-3 5

[2] Jain, Anil K.; Ross, Arun (2008). "Introduction to Biometrics". In Jain, AK; Flynn; Ross, A (eds.). Handbook of Biometrics. Springer. pp. 1–22. ISBN 978-0-387-71040-2

[3] Schnorr, C. P. (1990). "Efficient Identification and Signatures for Smart Cards". In Gilles Brassard (ed.). Advances in Cryptology. Conference on the Theory and Application of Cryptographic Techniques. Proceedings of CRYPTO '89. Lecture Notes in Computer Science. Vol. 435. pp. 239–252. doi:10.1007/0-387-34805-0_22

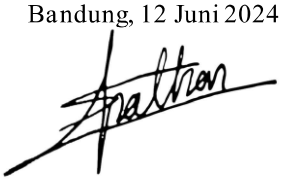
[4] Dodis, Y., Reyzin, L., Smith, A.D.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, pp. 523–540 (2004). doi:10.1007/978-3-540-24676-3 31.

[5] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024



Nathanael Santoso (13520129)