Applications of Number Theory in Enhancing Biometric Data Privacy through Hash Function

Satriadhikara Panji Yudhistira - 13522125[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522125@std.stei.itb.ac.id

***Abstract*— In this modern world, our unique identities such as fingerprints, iris patterns, and even facial contours are increasingly being used to identify and verify who we are. However, storing this sensitive information in plain text can pose a fatal security threat.**

This paper delves into the world of number theory, exploring how its tools - modular arithmetic and - can be leveraged to design efficient hash functions specifically for biometric data protection in computing technology.

***Keywords*— Biometric Data Privacy, Hash Function, Number Theory.**

I. INTRODUCTION

Do you unlock your phone using face? Or fingerprint? That is some examples of authentication using biometric. Why would we use biometric? What if we want to implement it as well to our local authentication like our home door or something like that.

Biometrics (ancient Greek: bios ="life", metron ="measure") refers to two very different fields of study and application. The first, which is the older and is used in biological studies, including forestry, is the collection, synthesis, analysis and management of quantitative data on biological communities such as forests. Biometrics in reference to biological sciences has been studied and applied for several generations and is somewhat simply viewed as "biological statistics" [1].

Our biometric data is one of the most important identification data. It is really needed that we secure our private data (the biometric data) in a safe way. With the application of number theory in hash function, we can securely save or store our biometric data.

II. THEORETICAL BASIS

A. Modular Arithmetic

Modular arithmetic plays an important role in integer computation, especially in application of cryptography. Operator that used in modulo arithmetic is $\boldsymbol{mod}$. Mod operator gives the remainder of a division. For example, 23 divided by 5 is 4 and remainder of 3, so we write $23\ \boldsymbol{mod}\ 5 = 3$. The definition of mod operator is stated as follows:

Notation: $a\ \boldsymbol{mod}\ m = r$ so that $a = mq + r$, with $0 \le r < m$

If $a\ \boldsymbol{mod}\ m = 0$, then we can say that a is multiple of m, that is a is divisible by m [1].

B. Hash Function

Data that saved on the computer memory need to place in a way that can be search fast. Every data in the form of record has a unique key field that differentiates from a record and another record. Hash function is used to place a record that have a key. Hash function is commonly used such as

$$h(k) = k\ \boldsymbol{mod}\ m$$

that in this case, m is a sum of memory location that available (for example if the memory is such as cell that given index of 0 until m – 1). The h function above place record with k key into certain memory that address of $h(k)$.

Suppose that $m = 11$, so we have cell of memory that given index of 0 to 10. We will store the record data that every each one has a key *15, 558, 32, 132, 102,* and *5*. In the beginning the cell of memory is empty.

Each these six data stored on the location that calculated as follows:

$$h(15) = 15\ \boldsymbol{mod}\ 11 = 4$$
$$h(558) = 558\ \boldsymbol{mod}\ 11 = 8$$
$$h(32) = 32\ \boldsymbol{mod}\ 11 = 10$$
$$h(132) = 132\ \boldsymbol{mod}\ 11 = 0$$
$$h(102) = 102\ \boldsymbol{mod}\ 11 = 3$$
$$h(5) = 5\ \boldsymbol{mod}\ 11 = 5$$

The state of the cells of memory after been stored the sixth record data is visualized as follows:

132			102	15	5			558		32
0	1	2	3	4	5	6	7	8	9	10

Because hash function is not one-to-one function, so there may be a collision in storing some record data. For example, we will store a record data with the key is 257. The calculated hash generates $h(257) = 257\ \boldsymbol{mod}\ 11 = 4$, whereas the cell memory index of 4 is already filled. So, we got a collision. To overcome a collision, we need to apply collision resolution policy. One collision resolution policy is finding the next highest cell that has not been filled. Then if we apply this policy, the data with the key 257 is stored at cell index of 6.

If we want to find the certain record data, then we use the hash function again. For example, we will find the record data with the key p, then we calculate $h(p) = p\ \boldsymbol{mod}\ 11$, for instance $h(p) = q$. If the record p is the same as the data in the location of q, we say record p location is found. Otherwise, if the record p is not the same as the data in the location of q, so we look at the next highest position; if record p is not on this position, we look at the next position and so on. If we reach an empty cell or come back to the beginning position, we conclude that there isn't any record p [2]. One of the most popular and most secure is *SHA-256*.

III. Methodology

Before I explain any of the method will be used, it has some requirement. To doing this experiment, we must have installed:

- Python (version 3.7 above)
- Pip (to install deepface)
- Virtual Environment (optional)

A. Store Biometric Data

In this case I will use face as my biometric data (there is some other examples like iris and fingerprint, but I don't have the necessary equipment to do so). I will be using python to store capture and store the data.

```
from deepface import DeepFace


def extract_facial_features_from_img(img):
    return DeepFace.represent(
        img_path=img, model_name="VGG-Face", detector_backend="retinaface"
    )
```

I used library from deepface [3], the function above will need an image path or embedded 64 images, and it will return a list of embedding. This way the biometric data is already got extracted to a data, but the data is still vulnerable.

B. Hashing with Hash Function

SHA-256 is one of the commonly used hash functions because of its effectiveness in securing data.

```
facial_features = extract_facial_features_from_img(img)
hashed_img = hashlib.sha256(str(facial_features).encode("utf-8")).hexdigest()
```

Using a highly optimized and secure built-in library, *hashlib*. We can use sha256 hash function easily but secure.

C. Hashing The Index to Store Data

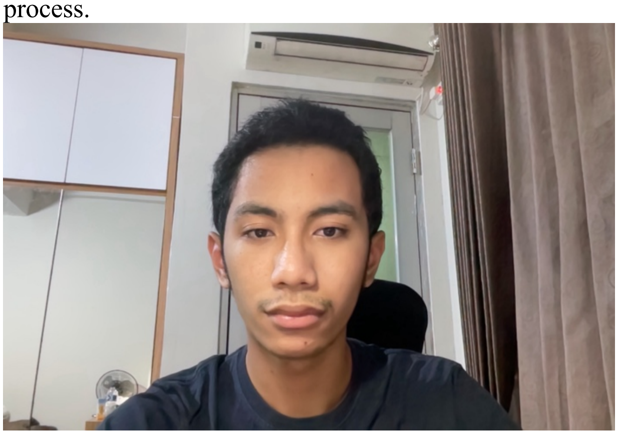
Since the data has been hashed, it is already secured valuely. Storing the data can also be protected by using hash function again. This time the hash function is for calculate the index we will store in the memory, the hash function we are using will be the modulo hashing method where the index is the result from the $data\ \mathbf{mod}\ size = index$. After we got the hashed index, we store it to the memory or databases (in real-world problem).

```
def modulo_hashing(facial_features, size=SIZE):
    return facial_features % size


def insert(facial_features, memory):
    facial_features_int = int(facial_features, 16)
    index = modulo_hashing(facial_features_int)

    while memory[index] is not None:
        index = (index + 1) % SIZE

    memory[index] = facial_features
```

Thus, the hashed biometric value is stored in the memory.

IV. Result

With all the process explained and built, we run all the process.



using this picture as a test case,

```
~/Developments/Kuliah/Semester-3/Makalah/faceRecognition                ✓ 5s  faceRecognition  base
└ python biometricHashing.py
b0b8abbcf35c6a9ac50b930fec91ff70f1affe9ad7bcd1dc580dae383d7d8cb3
```

b0b8abbcf35c6a9ac50b930fec91ff70f1affe9ad7bcd1dc580dae383d7d8cb3, that is the hashed of my biometric (face image) data, a hex digit. And the next step is storing the data into the memory (for this case I will store it to list).

```
~/Developments/Kuliah/Semester-3/Makalah/faceRecognition                ✓ 5s  faceRecognition  base
└ python biometricHashing.py
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, Non
e, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None,
 None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, Non
e, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None,
 None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, Non
e, 'b0b8abbcf35c6a9ac50b930fec91ff70f1affe9ad7bcd1dc580dae383d7d8cb3', None, None, None, None, None, None, None, None, None, None, Non
e, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None,
 None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
```

The data will be securely stored to the hashed index as above.

V. Conclusion

From the research we did, we can conclude that using biometric as an authentication in this modern world is nothing new. But we must aware that our biometric data is securely stored, because of the value of the biometric data that can access private things. Now we know how to securely store a biometric data, we can implement it to our local authentication as well or checking if our third-party authentication that use you biometric as the password is safe or no.

VI. Acknowledgement

The author is very grateful to God Almighty, Allah, who is the most merciful for providing strength, health, and support throughout this academic journey. And thanks to Dr. Ir. Rinaldi Munir, M.T. and Monterico Adrian, S.T., M.T., for their guidance, encouragement, and unwavering support throughout the entire process. My sincere thanks go to my friends and family for their understanding, patience, and encouragement during this demanding period. Your belief in me has been a constant source of motivation. And lastly, I would like to express my sincere gratitude to all those who have contributed to the completion of this paper. Your support and assistance have been invaluable, and I am truly thankful for your generosity.

REFERENCES

[1] Munir, R. (2020). Matematika Diskrit. Penerbit Informatika Bandung. Bandung. (pp. 192-193, 214-215).

[2] Bhattacharyya, D. (2009). Biometric Authentication: A Review. International Journal of u- and e- Service Science and Technology, 2(2), pp. 14.

ATTACHMENT

Source code used in this research: https://github.com/satriadhikara/biometric_hash

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, 11 Desember 2023

Satriadhikara Panji Yudhistira 13522125