Analyzing Hashing Algorithms and One-Time Passwords in Pseudonymous Signature and Account Verification System for a “Campus Digital Town Hall” Twitter Auto Base

Zhillan Attarizal Rezyarifin (18220008)
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
zhilanar@gmail.com

***Abstract*—Twitter “auto bases” has become the preeminent town hall infrastructure of Indonesia’s digital space with up to millions of users, including university students. This phenomenon caused serious risks and problems in anonymity and privacy, data security, auto base governance, and excessive admin workload. We propose and evaluate hashing algorithms and one-time password (OTP) implementation designs that might provide simple but reliable solutions to these issues. We found that an automated verification system using TOTP and a 7-digit base64 pseudonymous signature generated by SHA-3 satisfy our evaluation process. We also demonstrated their feasibility through conceptual prototypes in Python.**

Keywords—Hashing, SHA-3, OTP, Twitter, Auto Base, Digital Town Hall, Anonymity, Pseudonymity, Data Security, Cybersecurity

I. Introduction

In any defined and sufficiently large community, a “town hall” is a vital infrastructure necessary to communicate, exchange, and market information, in the realization of a “marketplace of ideas” based on the principle that free speech must be tolerated because it will lead towards the truth [1]. A town hall of a community possesses the following characteristics: (1) universal and exclusive access for members, with access to external actors granted through community agreement, (2) ease of use and access in process and time, and (3) freedom from threat and coercion.

Throughout the development of Indonesia’s modern social media culture, “auto bases” on the Twitter microblogging social media platform, from AskrlFess to ITBfess, have emerged as the preeminent de facto town halls of the Indonesian national digital space. An auto base receives input from a Twitter account registered as a “member of a community” and then tweets that input as a “menfess”. Indonesian auto bases come in varying sizes, from a few thousand users to millions. The operations of these bases are often funded through donations or paid promotion marketing. Recent dynamics have increased Twitter’s value in this aspect, with Twitter owner Elon Musk intending to turn Twitter as a whole into the “world’s digital town hall.” [2].

The public nature, perceived anonymity, ease of access, and freedom of speech of these auto bases are the immutable prerequisites for their popularity. As an already widely used social media, Twitter naturally drives significant traffic to auto bases, which internal confession boards cannot possibly compete. Furthermore, many admins maintain ideological commitments to the principles of internet anonymity and freedom, influenced by the culture of the early internet posting boards. These prerequisites inform the design of auto bases, whether developed independently or in the form of a commercial-off-the-shelf (COTS) from a third-party provider.

In general, Twitter direct messaging (DM) is the primary interface for an auto base system, although some auto bases employ external input systems, such as a custom website. Membership registration for these auto bases is mostly implemented through a combination of a manual account verification system and account follow-back. Backend systems, API, server hosting, and databases are either custom-developed by individual developers or handled by third-party services that provide COTS auto base.

Four problem areas emerged from these implementations: (1) anonymity and privacy, (2) data security, (3) auto base governance, and (4) excessive admin workload. More serious-themed auto bases, such as campus auto bases, often forces candidate users to hand over their personal information, such as their college profile and email, to be used for manual verification. There currently exists no convention or commonly used service to manage this process, turning this necessary process into a serious data protection and privacy risk potential. A particular concern is in the transfer of auto base ownership, as there is no guarantee on the intentions of the new owner. The data and legal governance situation of third-party providers, most of them small enterprises, are either amateurish or nonexistent; for example, there is presently no way to know if a data breach has happened. Furthermore, the pool of acquired data gives auto base admins the ability to crosscheck the identity of *menfess* senders with their real-life identity, severely undermining the anonymity principle. A false perception of anonymity is particularly problematic in the context of sensitive-themed auto bases, such as those that allow political, mental health, suicidal, sexual, or love confessions, as compromised anonymity can lead to serious real-life repercussions.

A related problem is the enforcement of auto base terms and conditions (ToS). In social media, most ToS-breaching behaviors are committed by a minuscule percentage of users, but they can single-handedly determine auto base convention of use, culture, and reputation [3]. Admins need a way to identify and effectively ban serial trolls without compromising anonymity, whether in perception or in the system. Another major cybersecurity and cyber-governance problem is the sheer volume of manual work required by the present verification system.

Contributions. In this paper, we answer two seminal questions on auto base cybersecurity and governance.

1. How can we streamline and automate a campus auto base user registration process and secure its user data privacy by removing unnecessary data requirements?
2. How can we simultaneously implement an anonymity guarantee and data privacy system with a ToS policing system in a campus auto base, given that these goals contradict each other?

To answer the former part, we described in Sect. III C a verification system using a combination of campus email and one-time password (OTP). For the latter part, we described in Sect III D a pseudonymity system using hash functions. We then evaluated these answers in Sect. IV A and B respectively.

Outline. The rest of this paper is organized as follows. In Sect. II, we describe this paper's research methodology and limitations. Sect. III covers theoretical frameworks used in the evaluation process and explores conceptual solutions. We then design and conduct an evaluation of algorithms and designs (Sect IV), and provide some general conclusions (Sect. V).

II. Research Methodology

A. Methodology

We initiated this research based on our prior knowledge and experience of the researcher in setting up, running, and managing a large auto base in the last 6 months. We then collect relevant algorithms and designs through literature reviews of journals, documentation, and other sources. Following that, we defined parameters to evaluate which algorithms and designs best satisfy the needs of a Twitter auto base, and conduct a general review. These forms the basis for a demonstrative conceptual prototype.

B. Limitations

We limited our research to the exploration and evaluation of conceptual algorithms and designs, complemented with supporting prototype implementations to demonstrate their feasibility. Problem areas are limited to (1) a user registration and verification system and (2) a pseudonymity signature system, both in the context of a university-focused auto base.

III. Theories and Conceptual Explorations

A. Services of a Cryptographic and Cybersecurity System

A cryptographic and cybersecurity system provides some of all of the following services: (1) confidentiality, (2) data integrity, (3) authentication, and (4) non-repudiation [10]. We cover three of these services as follows:

1. *Confidentiality*. We aim for a system that hides the identity of a *menfess* sender from anyone, including the admins.
2. *Authentication*. We aim for a registration system that reliably verifies whether a registering user is an active college student of a specific campus or not.
3. *Non-repudiation*. We aim for a *menfess* broadcast system where the sender cannot deny that they have sent a ToS-breaching *menfess*, enabling admins to take the necessary actions against them.

Items covered in this segment will be further defined and explained in the rest of this section.

B. Auto Base

An auto base is a Twitter bot account that can receive text and media input from another Twitter account, or "sender", and then automatically tweet that input. All tweets from an auto base can be viewed by the general public, driving traffic and increasing the auto base's popularity and user base growth. To prevent spam and support other custom goals, such as gatekeeping auto base use to just a certain community, there exists some method to verify whether the sender's account has been verified. Auto bases are either self-hosted by individual developers or sold as a COTS service by a third party. The following figure shows an auto base flow process.

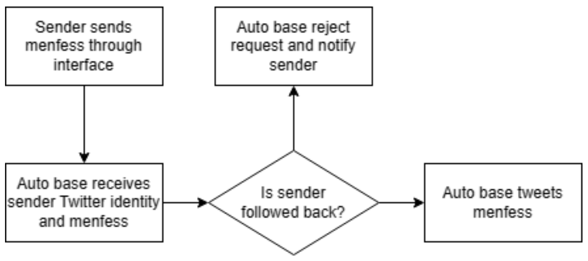

Sender sends menfess through interface
Auto base reject request and notify sender
Auto base receives sender Twitter identity and menfess
Is sender followed back?
Auto base tweets menfess

Fig. 3.1. Auto base flow process

Twitter DM is generally used as the user interface to send inputs to auto bases, and admins can read everything sent through DMs and what account sent them; other popular interface options are Telegram and custom-developed web apps. This admin ability creates a serious privacy risk and compromises the anonymity principle. On the other hand, admins rely on this ability to police auto base ToS and ban serial or exceptional offenders. An improved auto base should satisfy both of these contradictory needs.

C. Twitter API

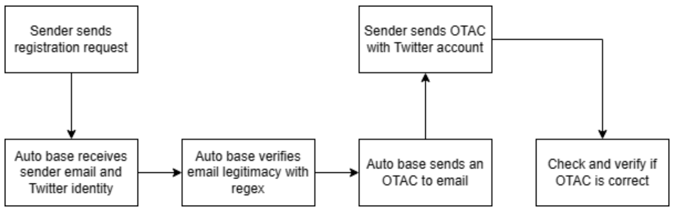
Flow of data from a sender account or a third-party application to an auto base is controlled by the Twitter API. Twitter API comes in three tiers: (1) basic, which provides free-of-charge write tweeting services (limited to 1500 tweets per month) as well as read and write DM services; (2)

premium, which increases the quota to 3000 tweets (along with other services), and (3) enterprise. Tweet lengths should generally be a non-issue in the context of these quotas, given most auto base accounts are verified under Twitter Blue.

D. Verification and One-Time Authorization Code

A key characteristic of a digital town hall is the limitation of access to just the members of a defined community. This paper covers auto bases meant for active students of a particular university. Currently, the most common registration method is a mass manual verification system under an event known as open follow back, or "opfol", where new users submit whatever personal identity is required by the admins. These *opfol* are regularly conducted in public reply spaces, usually as a part of a paid promotion event, causing a personal user data security disaster.

We propose that campus emails provide a simpler, automated, and far more secure alternative as an identity verifier, thanks to its domain format of <campus name>.ac.id enabling easy checking with regular expression. In this proposition, we assume that access to such emails is limited to active members of the campus. The automated nature of an email-based verification allows for regular purges of inactive users and alumni, the latter having lost access to their campus email. The following figure describes the flow process of a sender's Twitter account verification using an active campus email to verify that a registration requester is the owner of the provided campus email address.



Sender sends registration request
Auto base receives sender email and Twitter identity
Auto base verifies email legitimacy with regex
Auto base sends an OTAC to email
Sender sends OTAC with Twitter account
Check and verify if OTAC is correct

Fig. 3.2. Account verification flow process

To implement this process, we propose the use of a One-Time Password (OTP, not to be confused with One-Time Pad). OTP allows easy and secure verification of email-based log-in processes. Instead of sending login IDs, passwords, or personal information such as college profile screenshots, only a single-use password with a limited session lifespan crosses the network, thus protecting users from eavesdropping attacks. If an OTP is later compromised, it will not affect the security of other sessions, as new OTPs will have automatically been generated [4, 5].

Time-Based One-Time Password Algorithm (TOTP) is the most common functional implementation of OTP, consisting of the following formula [6]:

$$TOTP = HOTP(K, T) \tag{1}$$

Where:

- K is the Key.
- T0 is the initial counter time, with 0 being its default value
- T is defined as current Unix time - T0
- HOTP is an HMAC-Based One-Time Password Algorithm

Moreover, the HOTP function itself is defined as the following:[7]

$$HOTP(K, C) = Truncate(HMAC\text{-}SHA\text{-}1(K, C)) \tag{2}$$

Where:

- Truncate function extracts a 4-byte dynamic binary code from a 160-bit (20-byte) HMAC-SHA-1 result
- C is counter
- HMAC-SHA-1 is Keyed-Hashing for Message Authentication algorithm that uses SHA-1 hash algorithm

Let HS = HMAC-SHA-1(K, C). The truncation function is defined as follows:

$$D = StToNum(DT(HS)) \bmod 10^n \tag{3}$$

Where:

- D is a number in the range of $[0..10^n\text{-}1]$
- StToNum converts a 31-bit string to a number in the range $[0..2^{31}\text{-}1]$
- n is the number of digits of the OTP
- DT is the dynamic truncation algorithm

For a string with the length range of String = String[0]...String[19] and Offset in the range of [0..15], DT function is defined as follows:

```
DT(String)
    OffsetBits = low-order 4 bits of String[19]
    Offset = StToNum(OffsetBits)
    P = String[OffSet]...String[OffSet+3]
    Return the last 31 bits of P
```

(4)

Finally, HMAC-SHA-1 function is defined as the following [8]:

$$H(K\ XOR\ opad, H(K\ XOR\ ipad, text)) \tag{5}$$

Where:

- H is the SHA-1 hash algorithm (other algorithms like MD-5 are also possible, although they suffer from poor quality against collision attacks).
- SHA-1 is a commonly used hash function that generates a 160-bit digest

- ipad is “inner pad”
- opad is “outer pad”

The inclusion of Unix time in the OTP generation process ensures that each OTP is both unique and uniformly distributed.

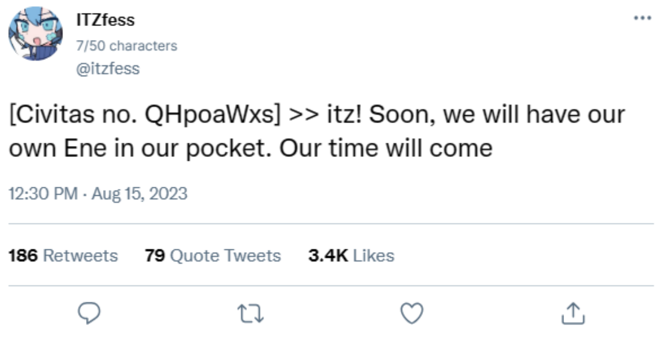
E. *Pseudonymity and Hashing Algorithms*

The following problems are the focus of this segment:

1. To safeguard anonymity, admins must not be able to know the identity behind a *menfess* sender.
2. To enforce terms and conditions, admins must be able to know the identity behind a *menfess* sender.

We propose that a pseudonymous message signature system efficiently solves both of these problems. Early internet boards and even present-day Twitter extensively implemented such a system. Instead of real identity, users posts behind a pseudonymous username. In this system, admins can simply ban usernames that serially breached the ToS, whether temporarily or permanently. A banned user will then have to wait out the ban, or create a new account.

In the context of auto bases, this system will cryptographically generate a pseudonym for each user based on their Twitter handle (e.g. “@itzfess”), and then put that pseudonym as a signature in each *menfess* tweet in a stylized fashion.


ITZfess
7/50 characters
@itzfess

[Civitas no. QHpoaWxs] >> itz! Soon, we will have our own Ene in our pocket. Our time will come

12:30 PM · Aug 15, 2023

186 Retweets 79 Quote Tweets 3.4K Likes

Fig. 3.3. Example of a pseudonymous signature system, with “QHpoaWxs” as the sender signature

We furthermore propose that a pseudonymous signature provides three significant advantages only available in the context of a campus auto base.

1. Admins can limit one email account to one Twitter user account by storing registered college email accounts in a database. This prevents banned users from simply registering with a new Twitter account. In general, college email accounts are based on student identity numbers, which are already public information. This negates the worries about a potential data leak.
2. A signature system shifts the responsibility of content creation and offloads burdens of controversies to its senders. A scandal involving the tweet in Figure 3.3., for example, will see public anger directed at “Civitas No.QHpoaWxs”, instead of the auto base or the admins. This gives admins significantly more political capital to maintain extreme liberal levels of free speech and expression.

From these explanations, we define three goals for this pseudonymous signature system:

1. To give a unique signature for all users. As a benchmark, this system should at minimum be effective for a user base size of 15,000 people, which is the total number of all students in ITB. There should not be any two Twitter handles (T1 and T2) with the same signature.
2. To make sure that the signature is reasonably compact, short, and pleasing to human eyes. A *menfess* should not, for example, be dominated by a 64-character-long signature.
3. To enable a ban system. Admins store a list of banned signatures on a database. Whenever a user sends a *menfess*, the system first generates a signature and then checks whether that signature is banned. This database can be reset periodically according to the admin’s governance policy.

To implement this pseudonymous system, we propose the use of hash functions. A hash function h mathematically transforms a message M of arbitrary length into a digest h of a fixed length, as follows [9].

$$h = H(M) \tag{6}$$

The purpose of a hash function is to make h a reasonably unique signature for M. As such, a hash function H has the following characteristics [9, 11]:

1. *Fast computation*. Given M, it is trivial to find h.
2. *Preimage resistance*. Given h, it is difficult to find M such that h = H(M).
3. *Second preimage resistance*. Given H(M1) = h, it is difficult to find M2 such that H(M2) = h..
4. *Collision resistance*. It is difficult to find arbitrary M1 and M2 such that H(M1) = H(M2).

According to the birthday paradox theory, we will likely find two arbitrary matching outputs for N bits of output in just $2^{n/2}$ hash operations [12]. Elaborating on this principle yields the following collision resistance formulas [13]:

$$n(p;\ H) \approx \sqrt{2H \ln \frac{1}{1-p}} \tag{7}$$

$$Q(H) \approx \sqrt{\frac{\pi}{2} H} \tag{8}$$

Where:

- n(p; H) is the number of brute force attacks needed to raise the probability of one arbitrary collision happening throughout the entire process to p
- Q(H) is the likely number of values that must be brute-forced until one collision is found.

- H is the number of all theoretically possible hash combinations. This being 2^n, with n the number of bits in the hash digest.

Pre-imaging resistance, meanwhile, is measured using simple brute force analysis [14].

Two of the most common hashing function family is Secure Hash Algorithm (SHA) and Message Digest (MD), with the latest SHA iteration, SHA-3, being the most secure. SHA-3 uses Keccak Algorithm, which uses a sponge construction to randomly permutate an arbitrary amount of data, or "absorbing", and then output a variable length digest, or "squeeze", with a mechanism as follows [15].

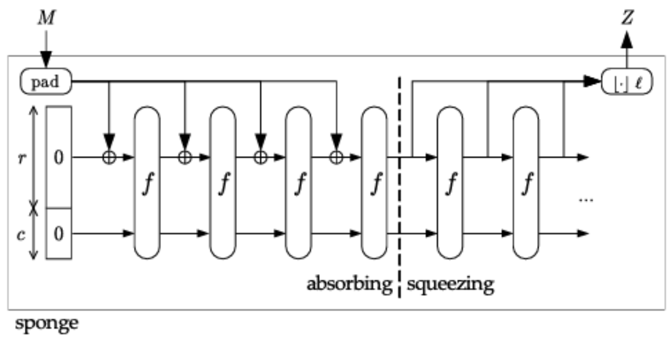


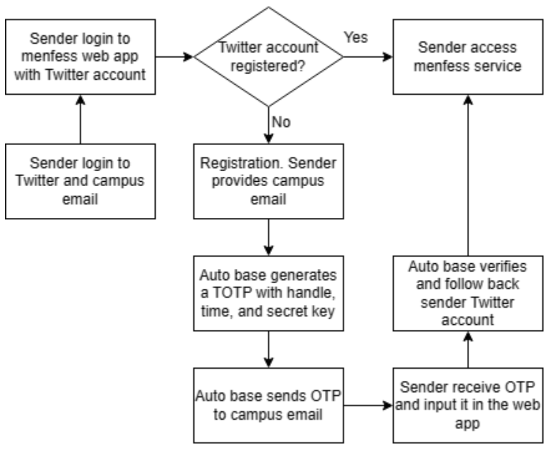
Fig. 3.4. Sponge construction in the Keccak algorithm

SHA-3 outputs bits that must be converted into a string data type first before it is usable for signature use. We propose using Base 64 to do this job, given its efficient ability to represent 6 bits for every digit (compared to hexadecimal and its 4 bits per digit) and its binary-safe nature [16].

IV. Solution Design, Evaluation, Demonstration

A. *TOTP in account verification*

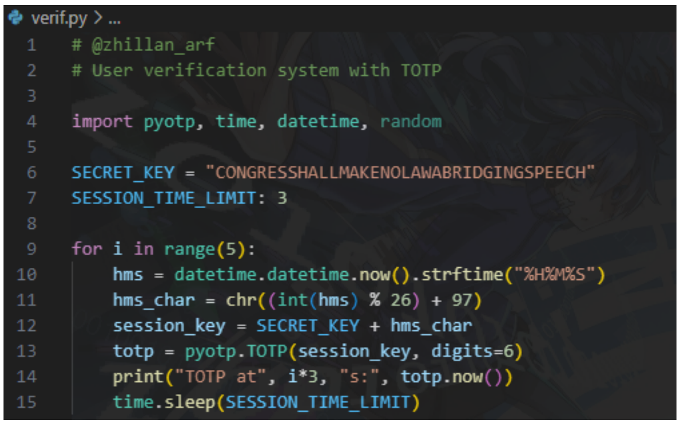
From the aforementioned algorithms and designs, we propose the following account verification design that utilizes TOTP. We assume that this design is implemented as a custom-developed, self-hosted web application.



Sender login to menfess web app with Twitter account
Twitter account registered?
Yes
Sender access menfess service
No
Sender login to Twitter and campus email
Registration. Sender provides campus email
Auto base generates a TOTP with handle, time, and secret key
Auto base verifies and follow back sender Twitter account
Auto base sends OTP to campus email
Sender receive OTP and input it in the web app

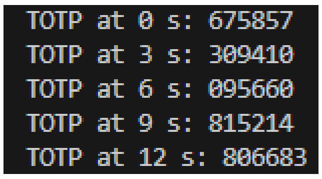
Fig.4.1. New admin-less user verification system with TOTP

The following Python code and its execution demonstrate the TOTP part of the verification system.



```python
# @zhillan_arf
# User verification system with TOTP

import pyotp, time, datetime, random

SECRET_KEY = "CONGRESSHALLMAKENOLAWABRIDGINGSPEECH"
SESSION_TIME_LIMIT: 3

for i in range(5):
    hms = datetime.datetime.now().strftime("%H%M%S")
    hms_char = chr((int(hms) % 26) + 97)
    session_key = SECRET_KEY + hms_char
    totp = pyotp.TOTP(session_key, digits=6)
    print("TOTP at", i*3, "s:", totp.now())
    time.sleep(SESSION_TIME_LIMIT)
```

Fig.4.2. TOTP implementation in Python



```
TOTP at 0 s: 675857
TOTP at 3 s: 309410
TOTP at 6 s: 095660
TOTP at 9 s: 815214
TOTP at 12 s: 806683
```

Fig.4.3. TOTP Python implementation result

B. *SHA-3 Collision and Pre-Imaging Resistance Analysis*

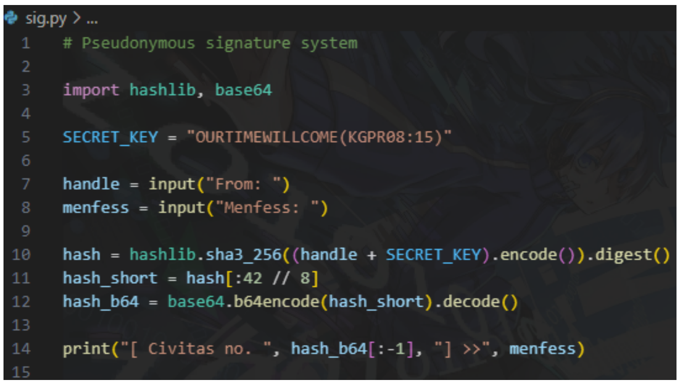
Using equations (7) and (8), we assemble the following table of birthday attack resistance. Note that SHA-3 outputs a variable bit length of either 224, 256, 384, or 512 bits. As a benchmark, we take the lowest bound, 224. We define a collision chance of 0.1%, given a population, to be safe".

TABLE I. Collision resistance for various bit lengths

bits	base64 str length	H	~Q(H)	~n(p; H) = 0.1%
6	1	64	1	0
12	2	4.10×10^3	8	3
24	4	1.68×10^7	5,000	183
30	5	1.68×10^9	41,000	1,465
36	6	1.07×10^{10}	328,000	11,726
42	7	6.87×10^{12}	2,628,000	93,810
48	8	4.40×10^{14}	21,000,000	750,000
60	10	1.15×10^{18}	1.35×10^9	4.80×10^7
72	12	4.72×10^{21}	8.61×10^{10}	3.07×10^9
96	16	7.92×10^{28}	3.53×10^{14}	1.26×10^{13}
228	37	4.31×10^{68}	2.60×10^{34}	9.29×10^{32}

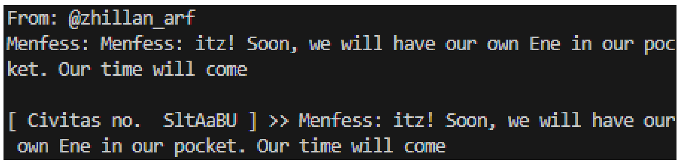
As previously stated, we use ITB's 15,000 students as a benchmark. For such a population size, a bit length of 42, represented by 7 digits of base 64, satisfies the collision chance bottom threshold requirement.

The following Python code demonstrates a pseudonymous signature generator system.



```python
# Pseudonymous signature system

import hashlib, base64

SECRET_KEY = "OURTIMEWILLCOME(KGPR08:15)"

handle = input("From: ")
menfess = input("Menfess: ")

hash = hashlib.sha3_256((handle + SECRET_KEY).encode()).digest()
hash_short = hash[:42 // 8]
hash_b64 = base64.b64encode(hash_short).decode()

print("[ Civitas no. ", hash_b64[:-1], "] >>", menfess)
```

Fig.4.4. Pseudonymous signature implementation in Python



```
From: @zhillan_arf
Menfess: Menfess: itz! Soon, we will have our own Ene in our poc
ket. Our time will come

[ Civitas no.  SltAaBU ] >> Menfess: itz! Soon, we will have our
 own Ene in our pocket. Our time will come
```

Fig.4.5. Pseudonymous signature implementation result

V. Conclusion

In this paper, we reviewed the issues of Twitter auto base, propose conceptual solutions using hashing algorithms and OTPs, evaluate design solutions, and demonstrate a few conceptual protoypes. We found that an automated verification system using TOTP and a 7-digit base64 pseudonymous signature generated by SHA-3 provides satisfactory solutions to the problems of anonymity and privacy, data security, auto base governance, and excessive admin workload.

Video Link at YouTube

We provide a complementary explanation video on YouTube demonstrating the conceptual prototypes, with the link as follows: https://youtu.be/2N-lZ0MsGq0

Acknowledgments

We give a special acknowledgment to God, who provided the mathematical laws and human brain architecture critical for the completion of this paper, along with other things. We would also thank Dr. Rinaldi Munir, M.T. whose tireless life of service has enabled us to master the necessary cryptographic science and materials. Finally, we give a special shout-out to Hiroyuki Nishimura, whose seminal work and sheer gut bravery first inspired us to delve into our previous works that now serve as the foundation of this paper.

References

[1] John Milton (1644), Areopagitica, in Areopagitica and Of Education 1, 50 (Harlan Davidson, Inc. 1951)

[2] Bush, S. (2022, October 31). Musk is right: we do need a digital town square. Financial Times. https://www.ft.com/content/4d9f7a9f-5e1e-4c0d-8b5b-8c6a3a7f0b6d

[3] Chandrasekharan, E., Pavalanathan, U., Srinivasan, A., Glynn, A., Eisenstein, J., & Gilbert, E. (2017). You can't stay here: The efficacy of Reddit's 2015 ban examined through hate speech. Proceedings of the ACM on Human-Computer Interaction, 1(CSCW), 1-22. https://doi.org/10.1145/3134666

[4] Haller, N.: The S/KEY one-time password system. RFC 1760 (February 1995) http://www.ietf.org/rfc/rfc1760.txt

[5] Paterson, K.G., Stebila, D. (2010). One-Time-Password-Authenticated Key Exchange. In: Steinfeld, R., Hawkes, P. (eds) Information Security and Privacy. ACISP 2010. Lecture Notes in Computer Science, vol 6168. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14081-5_17

[6] M'Raihi, D., Machani, S., Pei, M., & Rydell, J. (2011). TOTP: Time-Based One-Time Password Algorithm. Internet Engineering Task Force (IETF). https://tools.ietf.org/html/rfc6238

[7] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[8] Krawczyk, H., Bellare, M., & Canetti, R. (1997). HMAC: Keyed-Hashing for Message Authentication. Request for Comments: 2104. Informational. IBM and UCSD. https://tools.ietf.org/html/rfc2104

[9] K. Aggarwal and H. K. Verma, "Hash_RC6 — Variable length Hash algorithm using RC6," 2015 International Conference on Advances in Computer Engineering and Applications, Ghaziabad, India, 2015, pp. 450-456, doi: 10.1109/ICACEA.2015.7164747.

[10] Munir, Rinaldi. (2023). Pengantar Kriptografi. Sistem dan Teknologi Informasi Sekolah Teknik Informatika Institut Teknologi Bandung. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/01-Pengantar-Kriptografi-(2023).pdf

[11] Munir, Rinaldi (2023). Fungsi Hash. Sistem dan Teknologi Informasi Sekolah Teknik Informatika Institut Teknologi Bandung. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/13-Fungsi-hash-2023.pdf

[12] Patarin, J., & Montreuil, A. (2005). Benes and Buttery schemes revisited. IACR Cryptology ePrint Archive, 2005(004). https://eprint.iacr.org/2005/004.pdf

[13] Bellare, Mihir; Rogaway, Phillip (2005). "The Birthday Problem". Introduction to Modern Cryptography. pp. 273–274. https://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf

[14] Hatzivasilis, G. (2017). Password-Hashing Status. Cryptography, 2(2), 10. https://doi.org/10.3390/cryptography1020010

[15] Bertoni, G., Daemen, J., Peeters, M., & Assche, G. V. (2011). Cryptographic sponge functions. In Proceedings of the 2011 IEEE Symposium on Security and Privacy (pp. 115-130). IEEE. https://keccak.team/files/CSF-0.1.pdf

[16] base64. Python Software Foundation. Python Language Reference, version 3.11. https://docs.python.org/3/library/base64.html

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, 22 Mei 2023

[illegible]

Zhillan Attarizal Rezyarifin
NIM: 18220008