

Enhancing Account Recovery for Low-Income Users: A Comparative Analysis of MFA with Shamir's Secret Sharing and Secure Multi-Party Computation with Garbled Circuit

Helen Gelagay Emiru
helengelagay3@gmail.com

ABSTRACT

Based on various research, people in the world don't give emphasis to their phone's security, which leads them to cyberattacks. There are different reasons for their negligence. Starting from the security system's complexity, not having the money to buy advanced security materials, not having access to advanced smartphones or data plans, to simply not thinking they can be a target of malicious hackers. This research talks about how we can use Multi-Factor Authentication integrated with Shamir's Secret Sharing to provide secure and low-cost recovery for low-income users in urban areas. Shamir's Secret Sharing ensures that the hacker will not learn the full secret unless he/she is able to identify all the shares. This research will also compare different methods like Secure Multi-Party Computation integrated with Garbled Circuit and see which way is more secure, user-friendly, and cost-effective, allowing users to use it with access to a limited infrastructure.

Index Terms: Multi-Factor Authentication, Secret Sharing Scheme, Secure Multi-Party Computation, Garbled Circuit

INTRODUCTION

In this fast-paced world, using cash to buy goods and pay cabs has become increasingly rare. Instead, we are doing most of our things online, even our jobs. The first thing we check right after we wake up is our phone or electronic devices. We use our electronic devices for calling, texting, emails, and basically to run our day-to-day life, and that's why our data online should always be protected. Based on [1], there were 2,365 cyberattacks in 2023, with 343,338,964 victims.

From the real-life story published by [2] during mid-October in 2020, the Hackney Council in East London, which is a local government authority that provides services like housing, social care, transport,

infrastructure, and so on, was hit by a cyberattack. What happened was the hackers deployed a ransomware that terribly affected their system. The hackers published the data they got from the council. Even after two years, the Council is still trying to clean up the mess that happened back then. As a result, the systems were not functioning properly, and the people's data was exposed without their consent.

Most of the Multi-Factor Authentication (MFA) security methods are not cost-effective and they are a bit complex. Because of this, many people choose not to use it at all; as a result, they are more vulnerable to a cyberattack. Based on [3], most security failures do not result from technical issues; instead, they are from economic problems because every individual and company makes their decision based on the money they can invest in their security.

Based on [4], the traditional account recovery methods like email-based recovery, SMS/call recovery, and Knowledge-Based Authentication (KBA) or Security questions are not very reliable, but integrating MFA with Shamir's Secret Sharing (SSS) will improve this. The main aim of the research is to answer this question, How can the Multi-Factor Authentication system be improved with secret sharing schemes to provide secure and low-cost recovery for low-income users in urban areas?

SSS will be added to MFA to make it more secure and accessible. The research will also compare MFA + SSS with Secure Multi-Party Computation (SMPC) with Garbled Circuit (GC) to see which one is better.

To support the thesis, first we will see the literature review that is based on previous research about similar topics to see what the world already knows about this topic. Next we will see how MFA and SSS work, and we will evaluate their efficiency. We will also see how SMPC and GC compare with MFA and SSS.

Problem Statement

Low-income urban mobile users face severe account lockout risks when primary Multi-Factor Authentication (MFA) devices are lost, stolen, or broken. Existing account recovery paradigms, such as email resets, SMS OTPs, Knowledge-Based Authentication (KBA), or dedicated hardware keys, fail for this demographic due to strict operational constraints: limited digital literacy, reliance on outdated or shared hardware, intermittent cellular/data connectivity, and an inability to afford specialized security materials.

The core problem addressed by this paper is the lack of an account recovery framework that guarantees information-theoretic security and fault tolerance while operating under zero-cost, low-bandwidth, and offline-capable constraints.

To evaluate this problem, we establish the following parameters:

- User Profile: Low-income urban users relying on legacy or second-hand smartphones without constant data plans or hardware security tokens.

- **Attacker Model:** An adversary who may compromise an individual recovery channel (e.g., intercepting an SMS or stealing an offline physical backup) or attempt unauthorized account takeovers. The adversary must be unable to learn any information about the secret unless they control at least t shares.
- **Recovery Scenario:** A user loses access to their primary authentication device and must reconstruct their master account key to regain access.

Key Requirements:

1. *Accessibility & Low Cost:* Requires zero financial outlay and functions over basic SMS or offline channels.
2. *Cryptographic Resilience:* The recovery secret remains completely unrecoverable if an attacker obtains fewer than t shares.
3. *Fault Tolerance:* Account recovery succeeds even if up to $n - t$ shares are permanently lost or inaccessible.

Literature Review

Problem

Low-income users face different barriers when it comes to using their digital accounts. Research shows that most of the challenges arise because of economic limitations. Kostan et al. (2024) found out that people in poverty use old, broken, or second-hand devices and free untrusted cloud storage due to cost constraints. Users also said that they sense a false sense of security, meaning that hackers will not target them because they don't have anything valuable. Moreover, the inability to afford up-to-date hardware, antivirus tools, or any secure online services forces them to settle with less secure digital habits.

Multi-Factor Authentication (MFA) itself has its own difficulties. Reese et al. (2019) shows that while two-factor authentication methods like SMS, time-based one-time passwords (TOTP), and push notifications can improve security, they also have their own side effects. For instance, SMS depends on the consistency of the mobile service. TOTP requires smartphones, and not everyone can afford them. Push notifications require internet access.

Cause

Low-income users face different challenges when it comes to using multi-factor authentication. These challenges arise from two main causes. The first one is the socioeconomic limitations. As Kostan et al. (2024) mentioned, people with low income rely on old devices, and these devices may lack support for the advanced MFA tools like push notifications. This leads to users not being able to adopt secure methods even if they want to. The other thing is SMS or push notifications depend on reliable internet connections, and low-income users may live in weak infrastructures or may not be able to afford constant data plans. The other thing is digital security requires a basic level of digital literacy. Low-income communities may have less exposure to formal digital literacy training, making MFA even harder to adopt. The second reason is MFA design. As Das et al. (2019) mentioned, most of MFA research and design is conducted with participants that are developed and relatively experienced. For this reason the

MFA system works well for modern smartphones with constant internet access, but it is not inclusive for the underserved groups. The other thing is that only 9.1% of MFA research includes user studies, and the studies rarely focus on low-income users. MFA often adds extra steps that make the users feel overwhelmed and make the system more complex. Das et al. (2019) found that adoption rates decrease when MFA is mandatory by organizations or services because users see it as more troublesome than it is worth.

Solution

Given the different challenges faced by low-income users that limit MFA adaptation, we can see a clear need for an alternative method that is both accessible and secure. One of the approaches is integrating the Secret Sharing Scheme (SSS) into MFA systems. SSS is a cryptographic technique that divides a secret into different shares. A user doesn't need all the shares to construct the secret, but a certain number of shares is needed (threshold). If the shares found are less than the required threshold, then the user will not be able to reconstruct the secret or know pieces of information about the secret. This system is flexible and more inclusive. SSS can distribute the shares across trusted contacts, devices, or different storage methods. The other advantage is since low-income users often depend on old devices, losing one device does not mean losing access to the account; with SSS, they can still reconstruct the secret with the remaining shares. The other point is by allowing shares to be split among different places, SSS can accommodate different user contexts. Someone can reconstruct their secret if they have enough offline shares. The other is in order to simplify recovery, users only need to change a few shares, and it is way better than remembering multiple complex passwords or carrying a costly hardware token.

Background on SMPC and GC

There is also an advanced method. According to [5], "SMPC is a generic cryptographic primitive that enables jointly computing in a privacy-preserving manner." In other words, it is a way for people to compute something together without revealing their share to one another. There are different protocols we can use for SMPC, like Garbled Circuits, Secret Sharing-based protocols like the Goldreich-Micali-Wigderson Protocol (GMW), the Ben-Or, Goldwasser, and Wigderson Protocol (BGW), the Somewhat Practical Distributed Zero-Knowledge Protocol (SPDZ), Oblivious Transfer protocols, Homomorphic Encryption-based approaches and others, but for this research we will talk about the Garbled Circuit.

According to [6], Yao's Garbled Circuit is a cryptographic protocol that enables two-party secure computation in which two mistrusting parties can jointly evaluate a function over their private inputs without the presence of a trusted third party. The classic example used is the Yao's Millionaires Problem. In this problem the millionaires want to know who is the richest without revealing their wealth to each other, and in this situation we can use SMPC to determine who is the richest.

Justification of SMPC + Garbled Circuits as a Comparison Baseline

While Secure Multi-Party Computation (SMPC) via Yao's Garbled Circuits (GC) is conventionally designed for privacy-preserving function evaluation rather than static credential recovery, it serves as an important cryptographic baseline for this study for three reasons:

- **Distributed Trust Architecture:** Both SSS and SMPC eliminate single points of failure by distributing trust across multiple independent entities. Comparing SSS against SMPC allows us to evaluate whether interactive, multi-party cryptographic protocols offer meaningful security advantages over passive threshold secret reconstruction during credential retrieval.
- **Interactive Zero-Knowledge Recovery Model:** In advanced security contexts, an account recovery mechanism could hypothetically require two untrusted recovery agents (such as distinct service providers or trusted contacts) to compute an authentication function jointly: verifying the user's identity without ever exposing the underlying secret credential to either party. SMPC+GC represents the theoretical upper bound for this interactive model.
- **Benchmarking Architectural Trade-offs for Resource-Constrained Environments:** SMPC+GC provides an ideal counter-baseline to demonstrate performance trade-offs. By benchmarking the low operational cost of SSS against the high communication complexity and computational overhead of GC, this study formally proves why complex interactive cryptographic primitives are unviable for low-income, resource-constrained infrastructure.

METHODOLOGY

This research is mainly to evaluate the effectiveness of MFA integrated with SSS and do a comparison with Secure Multi-Party Computation (SMPC) integrated with Garbled Circuit (GC). After the comparison we will see which one is better for secure and cost-effective account recovery among low-income users.

- Data

To evaluate the secret-sharing architecture, a reference implementation of Shamir's Secret Sharing (SSS) was developed in Python following the foundational framework established by Shamir (1979) and supported by existing literature on MFA adoptions and limitations. Instead of collecting user data directly, the evaluation focuses on evaluating MFA with SSS and analyzing its performance by comparing its performance against alternative methods such as Secure Multi-Party Computation with Garbled Circuits.

- Tools

Python was chosen as the code language due to its simplicity and cryptographic libraries in SSS demonstration and algorithms in general. The code we chose uses built-in libraries such as random for polynomial coefficient generation and modular arithmetic for secret reconstruction.

- Process

First we obtained a demonstration code of the python implementation of SSS from [7]. The code was studied and modified to understand how it generates and reconstructs secrets. Specifically, core functions such as `generate_coefficients()`, `evaluate_polynomial()`, `generate_shares()`, and `reconstruct_secret()`, which uses Lagrange interpolation to recover the original secret by using the threshold number of shares, were identified.

Then we introduce MFA into the scheme. The main goal is to evaluate whether combining MFA with SSS strengthens the overall security model. This was not done by code modification; instead, we did it in a theoretical way where MFA requirements are applied on top of the secret construction.

Then the performance of SSS integrated with MFA is analyzed in terms of time complexity, user-friendliness, recovery, and other criteria, which we will further discuss later. Finally, we will compare SSS integrated with MFA with SMPC based on GC. For this comparison, we will rely on the description of the Yao's Garbled Circuit protocol, which enables two parties to compute a function without revealing their input to each other. The comparison will be done by using the same criteria we used to evaluate SSS integrated with MFA.

You can refer to the Python code repository demo that shows how SSS works in the appendix. The code has four functions; the `generate_coefficients()` function generates random polynomial coefficients and embeds the secret as a constant term. The `evaluate_polynomial()` function calculates the polynomial value at a given x , which is used to produce shares. The `generate_shares()` function generates n shares represented as (x,y) in the polynomial. The `reconstruct_shares()` function uses the number of shares to reconstruct the secret by using the Lagrange Interpolation.

The code will first call the `generate_shares()`. In the `generate_shares()`, the coefficient will call the `generate_coefficient()` function, which will return a list with the secret found at the 0 index and a random integer picked from 0 to the FIELD SIZE. It will return to the `generate_shares()` function, and it will call `evaluate_polynomial()`, which will return the value of a polynomial evaluated at a specific input and the polynomial's coefficients given in the list `coeffs` and it will go back to the `generate_shares` and return a list of shares with x and y values (x : the share's position and y : the value at that position). Then, we will go to the last function, which is the `reconstruct_secret()` function with the parameter of shares, and it reconstructs the secret by using the Lagrange interpolation and it will return the reconstructed secret.

Lagrange Interpolation finds a polynomial called Lagrange Polynomial that takes on certain values at arbitrary points and finds the new data points within the range of a discrete set of known data points [8]. In simple terms, it is a mathematical method that is used to find a polynomial that passes through the given set of points. For instance, let's say we have two points, (1,2) and (2,3), and if we are asked to find $f(4)$, what we are going to do is this.

$$f(x) = (2) * (x-2) / 1-2, \text{ replace } x \text{ with } 4$$

$$f(x) = -4$$

$$f(x) = (3) * (x-1) / 2-1, \text{ replace } x \text{ with } 4$$

$$f(x) = 9$$

$$f(4) = -4 + 9 = 5$$

RESULTS

Evaluation of MFA + SSS

Security Strength

Combining MFA with SSS will significantly improve the security strength. We can use MFA during the login by requiring multiple authentication factors like fingerprint, Face ID, password, etc., and we can use SSS to make sure the account recovery is secure even if some of the shares are compromised. The traditional recovery system relies on one single device or email, but SSS can split the shares into different places. The setup is information-theoretically secure, meaning an attacker with unlimited computing power cannot reconstruct the original secret without finding the threshold (t) number of shares. Integrating SSS with MFA will layer the defense and protect against attackers.

Cost Effectiveness

SSS integrated with MFA is more cost-effective than relaying of MFA-only hardware tokens for recovery. As we have discussed, MFA software may have subscription fees and high-security tokens requiring IT setup and maintenance, but SSS is more flexible in distributing and storing shares. The shares can be sent via low-cost methods or can be stored offline. Compared to the previous expensive MFA-only approach, integrating SSS with MFA will make it more cost-effective which is suitable for low-income users.

Usability

SSS is a backend algorithm system, meaning that users will not have a direct interaction with SSS, so the usability is highly dependent on the type of MFA used. Fingerprint and Face ID are usually fast, while push-based protocols are slower and require internet access, which most low-income users don't have.

Recovery

With SSS integrated with MFA, recovery will be a lot easier than with the traditional MFA, where most things are centered on the same device. With SSS added, even if the user lost their device, they can still reconstruct the secret from the remaining shares, up to $n-t$ shares lost. By distributing the shares to different places, the system increases reliability and prevents the risk of permanent account loss.

Speed

Sep 2026

Vol 11. No 4.

SSS's speed depends on factors like the finite field and the number of shares and threshold; it is still a fast algorithm with the time complexity of share generation of $O(n \cdot t)$ and secret reconstruction of $O(t^2)$. The speed of MFA depends on the type of MFA used. For instance, biometric and push notifications are almost instantaneous, so we can say that MFA+SSS is a responsive system.

Evaluation of SMPC + GC

Security Strength

According to research, Garbled Circuit integrated with SMPC is secure in the Universal Composability (UC) model, ensuring a strong privacy guarantee. The UC model is a formal framework in cryptography for analyzing the security of protocols.

Cost Effectiveness

Regarding efficiency, whether SMPC+GC is cost-effective or not depends on the garbler's computations, the evaluator's computation and the communication size of the garbled program. SMPC + GC is cost-effective for shorter integer computations; if the modulo is big, then it will be more expensive.

User-friendly

SMPC with Garbled Circuit is secure but not user-friendly because of its complexity and high communication costs.

Recovery

GC-based SMPC is fundamentally engineered for interactive, privacy-preserving multi-party computations rather than static credential retrieval. Unlike MFA + SSS, which natively incorporates threshold secret reconstruction to restore lost keys, SMPC + GC does not provide a direct mechanism for account recovery. It is evaluated here primarily as an interactive zero-knowledge baseline.

Speed

The speed depends on the circuit size, number of parties, and other factors, and the time complexity of GC-based SMPC is $O(n+m \cdot k)$, where n is the number of non XOR gates, m is the number of input bits from the evaluator, and k is the key length.

Criteria	SSS with MFA	Garbled Circuit with SMPC (Interactive Baseline)
Security Strength	Very high	Very high
Cost Effective	Moderate	Moderate to low
User-friendly	High	Low

Recovery	High	Low (not designed for recovery)
Speed	High	Moderate

DISCUSSION

The theoretical evaluation of MFA + SSS addresses the main weakness of the traditional MFA system, like account recovery. The system creates secure, low-cost and user-friendly recovery mechanisms especially for low-income users that use second-hand or older devices with limited infrastructure. MFA is used for secure authentication, while SSS can protect from account loss. This combined system balances security, usability, cost, and recovery.

In contrast, SMPC with GC has a different design. Based on what was discussed above, we can see that SMPC + GC is secure under the Universal Composability (UC) model, meaning it guarantees strong security. This makes SMPC + GC highly recommended when security and privacy are needed while computation is done between untrusted parties. It is not necessarily made for recovery. Its main goal is to preserve privacy during computation. Regarding speed, GC's performance has a time complexity of $O(n+m*k)$; while it works well for secure computations, it is still slower than MFA + SSS.

Overall, MFA + SSS balances security, usability, cost, and recovery, while SMPC + GC prioritizes security and privacy but at the cost of usability and recovery.

Future Work

Future studies could focus on real-world implementation and testing of MFA + SSS among low-income urban communities to measure adaptation rates, usability, and performance. Additionally, further research on how hybrid systems can combine the privacy strength of SMPC with the recovery benefit of Shamir's Secret Sharing to see if it can create a better framework.

CONCLUSION

The research is done to address the problem of digital account safety for low-income users, who often face problems such as outdated devices, inconsistent internet access, and limited resources. The challenges make traditional MFA systems hard to adopt and leave users vulnerable to cyberattacks. The integration of SSS with MFA provides a strong solution. While MFA makes sure the authentication is secure, SSS will give a cost-effective and reliable account recovery method. This approach prevents permanent account loss, provides high security even if some shares are compromised, and remains accessible to users with limited infrastructures. In comparison, Secure Multi-Party Computation (SMPC) with Garbled Circuits (GC) gives high security and privacy, but it is not designed for recovery and is not

user-friendly due to its complexity. Based on the research, MFA + SSS gives the best balance between security, usability, cost-effectiveness, and recovery, making it a better option for low-income urban users. On the other hand, SMPC + GC is better in scenarios where security and privacy are the main priorities.

REFERENCES

- [1] Sufi, F. (2024). A New Time Series Dataset for Cyber-Threat Correlation, Regression and Neural-Network-Based Forecasting. *Information*, 15(4), 199. <https://doi.org/10.3390/info15040199>
- [2] Pattnaik, N., Turner, J., Mott, G., MacColl, J., Huesch, P., & Sullivan, J. (2023). It's more than just money: The real-world harms from ransomware attacks. *IFIP Advances in Information and Communication Technology*, 261–274. https://doi.org/10.1007/978-3-031-38530-8_21
- [3] Anderson, R., & Moore, T. (2006). The economics of information security. *Science*, 314(5799), 610–613. <https://doi.org/10.1126/science.1130992>
- [4] Bonneau, J., Bursztein, E., Caron, I., Jackson, R., & Williamson, M. (2015). Secrets, lies, and account recovery: Lessons from the use of personal knowledge questions at Google. *Proceedings of the 22nd International Conference on World Wide Web (WWW '15)*. <https://research.google.com/pubs/archive/43783.pdf>
- [5] Zhou, J., & Zhang, L. (2018). Secure multi-party computation: Theory, practice and applications. *Journal of Computational and Applied Mathematics*, 344, 1–12. <https://doi.org/10.1016/j.cam.2018.01.033>
- [6] Bellare, M., Hoang, V. T., & Rogaway, P. (2012). Foundations of garbled circuits. *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, 784–796. <https://doi.org/10.1145/2382196.2382279>
- [7] Shamir, A. (1979). How to share a secret. *Communications of the ACM*, 22(11), 612–613. <https://doi.org/10.1145/359168.359176>
- [8] Das, B., & Chakrabarty, D. (2016). Lagrange's interpolation formula: Representation of numerical data by a polynomial curve. *International Journal of Mathematics Trends and Technology*, 34(2), 64–72. <https://doi.org/10.14445/22315373/ijmtt-v34p514>
- [9] Büttner, A., & Gruschka, N. (2024, March 22). *Evaluating the influence of multi-factor authentication and recovery settings on the security and accessibility of user accounts*. *arXiv*. <https://arxiv.org/pdf/2403.15080>

- [10] Mohassel, P., Rosulek, M., & Zhang, Y. (2015). Fast and secure three-party computation: The garbled circuit approach. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*, 591–602. <https://doi.org/10.1145/2810103.2813705>
- [11] Heath, D. (2024). Efficient arithmetic in garbled circuits. *Cryptology ePrint Archive*. <https://eprint.iacr.org/2024/139.pdf>
- [12] Feng, D., & Yang, K. (2022). Concretely efficient secure multi-party computation protocols: Survey and more. *SANDs*, 9(1), 1–23. <https://sands.edpsciences.org/articles/sands/pdf/2022/01/sands20210001.pdf>

APPENDIX

```
import random
# Large prime (must be greater than secret)
FIELD_SIZE =
208351617316091241234326746312124448251235562226470491514186331217050270460481

def reconstruct_secret(shares):
    sums = 0

    for j, (xj, yj) in enumerate(shares):
        prod = 1

        for i, (xi, _) in enumerate(shares):
            if i != j:
                prod *= (xi * pow(xi - xj, -1, FIELD_SIZE)) % FIELD_SIZE
            prod %= FIELD_SIZE

        prod *= yj
        prod %= FIELD_SIZE
        sums += prod
        sums %= FIELD_SIZE

    return sums

def evaluate_polynomial(x, coefficients):
    result = 0
    for power, coeff in enumerate(coefficients):
        result += coeff * pow(x, power, FIELD_SIZE)
```

```
    result %= FIELD_SIZE
return result

def generate_coefficients(threshold, secret):
    coeffs = [secret] # constant term
    for _ in range(threshold - 1):
        coeffs.append(random.randrange(0, FIELD_SIZE))
    return coeffs

def generate_shares(n, threshold, secret):
    coeffs = generate_coefficients(threshold, secret)
    shares = []

    for i in range(1, n + 1):
        x = random.randrange(1, FIELD_SIZE)
        y = evaluate_polynomial(x, coeffs)
        shares.append((x, y))

    return shares

# Example usage (calling the functions)
if __name__ == "__main__":
    secret = 1234
    n = 5 # total shares
    t = 3 # threshold

    print("Original secret:", secret)
    shares = generate_shares(n, t, secret)
    print("Shares:", shares)

    selected_shares = random.sample(shares, t)
    print("Selected shares for recovery:", selected_shares)
    recovered = reconstruct_secret(selected_shares)
    print("Recovered secret:", recovered)
```