National College of Ireland

Security Analysis & Mitigation Strategies for Multi-Factor Authenticator Apps

MSc Research Project
MSc Cybersecurity

Don Sunil
x23317230

School of Computing
National College of Ireland

Supervisor: Mark Monaghan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Don Sunil
Student ID:	x23317230
Programme:	MSc Cybersecurity **Year:** 2024 - 2025
Module:	MSc Research/Practicum
Supervisor:	Mark Monaghan
Submission Due Date:	15/09/2025
Project Title:	Security Analysis & Mitigation Strategies for Multi-Factor Authenticator Apps
Word Count:	6785 **Page Count:** 20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Don Sunil

Date: 15/09/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Security Analysis & Mitigation Strategies for Multi-Factor Authenticator Apps

Don Sunil
x233017230

Abstract

The growing dependence on Time-based One-Time Password (TOTP) authenticator apps such as Google Authenticator, Microsoft Authenticator, and Authy for multi-factor authentication (MFA) has made people more concerned with regard to their cryptographic robustness specifically in transport and backup layers. This study critically investigates whether widely used authenticator applications are susceptible to data leakage through insecure backup exports along with transport-layer exposures, in addition to whether such vulnerabilities can be reduced without any compromising of usability.

For development of this simulation-based framework, a lightweight emulator-less environment under Ubuntu WSL2 was used. For vulnerabilities, replication occurred when synthetic TOTP seeds exported in plaintext JSON format were utilized, also when traffic was captured via mitmproxy and was analyzed using custom parsing scripts. Mobile Security Framework or MobSF was used in order to conduct static analysis for the purpose of detecting missing cryptographic safeguards and insecure implementations. The captured data confirmed that secret seeds could be intercepted within certain configurations, so this revealed meaningful privacy risks.

In response, there was a mitigation strategy that was implemented via designing of a wrapper because it encrypts exported backup files through using Argon2id-based key derivation plus the XChaCha20-Poly1305 authenticated encryption scheme. The resulting .sec files preserved data integrity and also confidentiality while encryption times averaged to under 250 ms per file, and also the files incurred minimal overhead. They integrated further a validation mechanism, and SHA-256 hashing and hexdump inspection confirmed complete obfuscation of original contents.Secret leakage risk can drop greatly with application-layer encryption, as shown by the framework's results. Users remain endangered though by common dependence on open backups and certain apps' missing certificate pinning. Furthermore, usability remains preserved within the proposed approach. This preservation indicates that the strong security controls can coexist along with user accessibility.

This work offers a script-driven reproducible prototype regarding secure backup handling within TOTP apps stressing a key need for firm cryptographic conduct within MFA ecosystems. Developers are also given a pathway to adopt secure design patterns that advances academic discourse as well as practical security engineering in mobile authentication without sacrificing performance or usability.

Keywords: Multi-Factor Authentication (MFA), Time-based One-Time Password (TOTP), Authenticator Apps, Backup Security, End-to-End Encryption

1 Introduction

1.1 Background

In the current cybersecurity landscape, multi-factor authentication's (MFA) importance has grown exponentially. This growth is driven via the common compromise of password-based systems. Time-based One-Time Password (TOTP) applications like Google Authenticator, Microsoft Authenticator, and Authy have become popular as accessible alternatives to SMS codes since they can be used when offline and they do better resist SIM-swapping attacks. For lowering the risk of credential replay as well as forbidden access, these apps generate dynamic six-digit codes derived using shared secrets and timestamps.

However, the rate of adoption of TOTP-based authentication is now growing. As it grows, security concerns increase also. The research from Nash et al. (2024) did reveal some critical vulnerabilities within popular authenticator applications. Cloud backups lacking security caused the leakage of TOTP secrets and OAuth refresh tokens. Because of these flaws, attackers can replicate authenticators also bypass MFA protections entirely. This is especially true in the case when apps do omit safeguards such as certificate pinning or when they use more outdated key derivation techniques. Marky et al. (2022) state users often do not grasp the true extent of protection that these tools give. Even when flaws exist, users often assume total security.

MITM threats have been documented systematically across SSL/TLS-based applications as well. Studies by Conti et al. (2016) and Georgiev et al. (2012) depict exploitable channels through which TOTP apps can be compromised via poor certificate validation, improper hostname checks, and insecure backup practices. Even with their robustness within theory, the cumulative evidence suggests that TOTP systems implement in an inconsistent way and do lack universal standards for hardening.

1.2 Importance of the Study

WebAuthn and FIDO2 provide authentication resistant to phishing yet cost, infrastructure, and backward compatibility problems obstruct wide adoption. Thus, it stays a practical as well as urgent need that we improve the security in existing TOTP-based workflows. Addressing these flaws is helpful for end-users. It also assists developers and organizations within regulated industries complying with cybersecurity standards while ensuring usability. This study matters since it looks at empirically reproducing existing vulnerabilities to measure authenticator apps' hazard magnitude plus at prototyping practical defenses such as safe hardware-bound backup flows and built-in WebAuthn fail-safes. This work seeks to expand technical knowledge to suggest a change from old TOTP flows to safer designs.

1.3 Research Problem and Motivation

While writing addresses the theoretical as well as cryptographic underpinnings of TOTP, toward implementation relatively few studies practically guide implementation to close the divide between vulnerabilities disclosed and fixes deployed. Many TOTP apps leak sensitive data via backup mechanisms since they do not validate server certificates; therefore, they are vulnerable to MITM attacks, which is the main issue even though we widely trust them. Therefore, the following here is what the research investigates:

1.3.1 Research Question:

How can widely used authenticator apps mitigate security vulnerabilities in their backup mechanisms and cryptographic implementations to enhance user security without degrading usability?

Maintaining usability as well as accessibility, this research is motivated to harden TOTP authenticator implementations due to a clear need. Implementation that is successful will inform the industry in its guidance in addition to user education, so developers are empowered using alternatives tested for usability plus secure patterns.

1.4 Research Objectives

1. **Replicate Vulnerabilities**
   Investigate and reproduce backup-, storage-, and transport-layer flaws across fifteen widely used Android and iOS TOTP apps.
2. **Design Secure Backup Architecture**
   Implement an end-to-end encrypted, hardware-bound backup mechanism using Android StrongBox and Apple Secure Enclave APIs.
3. **Integrate Phishing-Resistant Authentication**
   Embed optional WebAuthn/FIDO2 fallbacks within existing TOTP apps to eliminate code replay vulnerabilities.
4. **Evaluate Security and Usability**
   Assess the cryptographic strength, performance overhead, and usability acceptance of proposed countermeasures through dynamic inspection, static analysis, and user testing.

1.5 Limitations of the Study

This study focuses only on authenticator apps that happen to be free and are popular and that run on Android and on iOS. The project scope excludes enterprise-only features along with premium backup modules and MDM (Mobile Device Management) integrations. Compliance with ethics will be ensured by the testing environment. Controlled conditions will dictate the use of emulated accounts along with devices. No one will have access to real-world user data or production accounts under almost any circumstances. The study does limit the usability evaluation to 20 participants because of time constraints, which might not generalize across every user demographic.

1.6 Assumptions

It assumes adversaries can access a user's device via root or intercept traffic using MITM techniques instead of both. Also, the study assumes device manufacturers act upon platform-level security primitives such as Android StrongBox. For example, the study assumes that iOS Secure Enclave provides tamper-resistant storage. For OAuth token revocation it is expected that service providers correctly handle it when a device is compromised. This assumption rests upon service provider actions.

1.7 Structure of the report

This report is carefully structured to reflect that it researches and implements systematically. The abstract that is outlining the motivation and methodology and key findings for the project begins it. Identified are important lacks within app security now, and reliance grows on TOTP multi-factor authentication which the intro explains. This follows a thorough literature

review, which critically examines previous studies on MFA vulnerabilities, cryptographic implementations, and secure backup strategies. Within the methodology chapter, ethical considerations and tool selection detail the simulation-based, interpretivist approach used for mitigation and vulnerability replication. The design chapter explains the three-layer architectural framework that simulates, detects, as well as protects. The implementation chapter gives a granular walkthrough of each stage as well as includes environment setup, traffic inspection, static analysis, encryption, and validation. The results chapter gets support from figures and tables presenting empirical findings through threat exposure logs, static scan observations, and encryption benchmarks. Technical implications as well as limitations are highlighted as the discussion chapter interprets these findings related to the research question and broader industry context. Finally, the conclusion is where you can find a summary of the project's contributions and a suggestion of directions for future work since it reinforces the practical relevance and academic value of the study.

2 Literature Review

2.1 Security and privacy failures in mainstream TOTP apps

Gilsenan et al. (2023) studied 22 Android authenticator apps using early large-scale auditing. Their work found that it was over 90 % of exported secrets through cloud-sync or SMS/email in plaintext, and it revived the very risks that MFA purports to solve. Nash et al. (2024) have demonstrated this now. They showed Google, Microsoft, and Authy traffic MITM interception lets an attacker harvest access and refresh tokens, and plaintext OTPs for services such as Facebook and Slack if certificate-pinning is absent. Table 1 also shows their repeated stolen token reuse. This makes Gmail the "least secure" client under testing. These empirical studies underscore systemic gaps. Security advertised differs from security delivered.

2.2 Forensic analyses and backup weaknesses

Berrios et al. (2023) confirmed that GPU-accelerated brute-force on encrypted backup archives is enabled by key-derivation parameters in several apps that remain below NIST recommendations using static reverse-engineering and mobile forensics. Authy had similar findings in Nash's next experiment. The experiment attributed the risk to low PBKDF2 iteration counts along with weak cloud-backup passwords. Together these studies highlight backups above. Backups can be MFA's soft underbelly rather than being cryptography itself.

2.3 Strengthening TOTP: honeytokens, HTOTP and dynamic groups

Papaspirou et al. (2023) do embed honeytokens inside of the OTP seed thus triggering alarms upon credential misuse seeking to harden the classic RFC 6238 design while Ding & Wang (2025) propose HTOTP by inserting decoy codes that invalidate replay attacks. Their earlier Dynamic Group TOTP (2024) extends this same idea to group settings, so it reduces secret reuse in those shared accounts. Although being analytically sound, both schemes depend on server-side changes, and none of the audited commercial apps do currently implement these changes, so this study will probe at an adoption gap.

2.4 Push-based and context-rich 2FA

"Push-to-Approve" mechanisms are marketed as frictionless. Jubur et al. (2024) show that users frequently ignore Compare-and-Confirm codes, enabling "Yes-click" social-engineering. In a parallel manner, Cao et al. (2024) present UWB-Auth which ties the approval message to a centimeter-level of distance-bounding. Still, the hardware that is

required remains niche. These are findings which underscore that usability gains often introduce a novel threat if surfaces are not anchored by strong context binding.

2.5 Hardware keys and password less FIDO2—promise and pitfalls

User studies from Nanda et al. (2024) show that folks keenly feel security yet constantly fret over FIDO2 tokens' portability, which mirrors a problem Nash & Gilsenan saw with poor uptake. Technical risks do also persist since Mahdad et al. (2024) craft overlay attacks since they exploit limited-display authenticators, and Ravilla et al. (2024) report that WebAuthn implementations are inconsistent across all the browsers. Recovery is another weak link. Amft et al. (2023) find many services silently disable MFA during account-reset flows negating upstream security effort.

2.6 Accessibility and inclusive authentication

Akanda et al. (2025) reveal as to how screen-reader-assisted flows often expose secret URIs, and how audio CAPTCHA alternatives remain under-deployed. NFC stickers like user- defined "concrete" tokens improve memorability yet Turner & Schmitz (2023) prove threat modeling is inconsistent. These efforts show strong MFA should not hurt accessibility goals.

2.7 Landscape reviews and standards alignment

Abubakar et al. (2023), Syahreen et al. (2024), and Tran—Truong & Pham (2025) in three surveys respectively on cloud MFA, generic MFA frameworks, and digital payments agree that NIST SP 800-63-3 guidance is partially met in industry deployments. Salameh et al. (2024) find that OTP apps resist phishing to the least extent though they deploy to the best extent, reinforcing the trade-off dilemma that earlier empirical studies identified.

2.8 Synthesis and research gap

Literature paints a consistent picture in aggregate. Commercial OTP apps do priorities convenience such as a frictionless approval along with simple backups over cryptographic safeguards because they do leave secure backup derivation also certificate-pinning either absent or else poorly implemented as well as recovery flows. Individual flaws are fixed through such novel proposals as "HTOTP", "UWB-Auth", and "FIDO2" yet remain theoretical or have some usability barriers. Few are the works that integrate end-to-end encrypted backups with hardware-rooted keys. Real-world user acceptance is evaluated infrequently.

Therefore, it is the case that this project extends prior work, and it does so in the following manner: (i) replicating Nash-style MITM tests across 15 popular apps, (ii) statically auditing backup-key derivation, (iii) prototyping a StrongBox-protected, libsodium-encrypted backup channel, and (iv) conducting A/B usability trials against FIDO2 alternatives. The study acts as a bridge for that MFA vulnerability gap "known-but-unfixed" which was identified throughout this review by its addressing of technical factors and human factors.

3 Research Methodology

To assess the security weaknesses that multi-factor authentication (MFA) applications have, this study employed a methodology that was simulation-based and experimental, specifically focusing on vulnerabilities in the backup and transport layer of Time-based One-Time Password (TOTP) systems. For compatibility with cybersecurity evaluation via tools so ethical reproducibility as well as alignment alongside the primary research question, selection was of that approach: How can widely use authenticator apps reduce vulnerabilities in their backup mechanisms and cryptographic implementations to improve user security without degrading usability?

This method imitates unsafe app actions and uses encryption safeguards. It allows for controlled exploration of attack scenarios and of defensive interventions without any compromising of real systems or of data.

3.1 Environment and Toolchain

For research, Ubuntu 22.04 ran using a Windows 11 laptop in Windows Subsystem for Linux (WSL2). Remaining lightweight, this platform provided compatibility alongside necessary Linux tools. The whole of the tool chain was composed of open-source components. Through this, replicability, ethical compliance, and transparency were ensured.

Table 1: Technical Environment and Toolchain

Component	Role in Research
WSL2 (Ubuntu 22.04)	Primary Linux environment for scripting and analysis
Docker	Containerized deployment of MobSF
Python 3.11	Implementation of encryption and hashing scripts
MobSF (Mobile Security Framework)	Static APK analysis of Google Authenticator
mitmproxy (simulated)	Emulation of intercepted network flows
hexdump, grep	Validation tools for binary inspection and data leakage detection

Ethical constraints were preserved as all tools were used fully offline and configured right locally for preventing accidental data transmission.

3.2 Data Collection Strategy

This study generated some synthetic data that was mimicking the structure of real-world TOTP secrets instead of relying on live traffic or reverse engineering which may violate terms of service or ethical boundaries. There are three types of files that are generated.

1. Simulated MFA backups: .json files structured as exported secrets from Authy Microsoft Authenticator and Google Authenticator. Issuer account secret algorithm and validity period were the typical fields that were included.
2. Mock network flow logs are text files that are formatted to simulate what a tool outputs, for example, mitmproxy. These files show intercepted secrets through otpauth URIs, plaintext JSON, or quoted strings.
3. Secured .sec files result from encrypted backups created from raw .json exports via a special wrapper.

Plausible app behaviour was mimicked when all synthetic data was created through random yet valid-looking values, avoiding real-world user data as well as infrastructure.

3.3 Experimental Procedures

For evaluation of the effectiveness that the proposed mitigation has, the methodology was like a realistic adversarial scenario by way of a structured sequence of operations that was reproducible.

Table 2: Procedural Workflow Summary

Phase	Activity
Static Analysis	MobSF scan of APK to inspect permissions, crypto usage
Backup Simulation	Creation of .json exports reflecting MFA backup structure
Leakage Emulation	Writing flows.txt with otpauth URIs and embedded secrets
Leak Detection	Execution of regex-based parser to flag exposures
Cryptographic Mitigation	Encryption using Argon2id + XChaCha20-Poly1305
Performance Measurement	Logging encryption time and computing SHA-256 hashes
Validation	Binary inspection of encrypted output to ensure secrecy

Automation occurred at each stage when feasible as repetition remained consistent throughout the three tests to confirm validity.

3.4 Ethical and Methodological Integrity

All researchers strictly adhered to ethical research principles throughout experiments. No one made use of physical devices, of real user accounts, or proprietary systems. APKs were downloaded only from trusted public mirrors like APKMirror and no application was modified, reverse-engineered, or executed outside of the offline testing environment. Data privacy persevered all throughout, and someone documented all configurations thoroughly in a README file for replicability.

4 Design Specification

4.1 Architectural Overview

The system's design was conceptualised as being a modular architecture of three layers. It was intended for simulation of vulnerabilities in MFA apps and testing secure backup mitigation strategies. This architectural design highlighted simple, modular, and reproducible qualities important for developer adoption and academic validation alike.

4.1.1 Layer 1: Simulation and Exposure

The design's first layer mimicked from user scenarios. These situations might reveal secrets by accident. They made .json files like TOTP export data. They also made flow logs representing intercepted network data. These artifacts were for simulation of common risks that are backup related. It was not required that a real device or live network interaction take place but things like email exports or cloud sync leakage were required.

4.1.2 Layer 2: Detection and Analysis

Statics with pattern-based analysis was used at the second layer to detect data exposure signs. APK analysis was performed using MobSF to uncover potential security flaws that exist in app architecture while parse_flows.sh, which is a custom shell script, was used to detect leaked secrets in simulated network flows. Attack vectors that are known informed of the choice of analysis criteria. These criteria featured plaintext key-value structures and otpauth URIs plus base32 secrets.

4.1.3 Layer 3: Cryptographic Protection

A secure format wrapped the existing backups within a client-side encryption tool, which was the third layer introduced. This was done with the use of a Python script since it utilized Argon2id to derive keys and XChaCha20-Poly1305 to encrypt in an authenticated way. The design had flexibility enough for adaptation into mobile or into desktop clients. It was flexible enough also for use as an external tool for security-conscious users and enterprises.

4.2 Functional Objectives

Five core objectives shaped the design in total; each linked to a specific delivery or research activity.

Table 3: Functional Objectives and Implementation Mapping

Objective	Design Implementation
Simulate realistic backup export vulnerabilities	Synthetic .json files for GA, MSA, and Authy
Emulate potential network leaks	Manually constructed flows.txt with secrets
Detect exposed secrets using reliable patterns	Regex-based detection script (parse_flows.sh)
Apply secure, scalable backup encryption	secure_backup.py using Argon2id + XChaCha20
Validate encryption through binary inspection	hexdump, SHA-256 hashing, and grep output

4.3 Reproducibility and Scalability

Folder hierarchies that are clear and code that are well-documented do promote the modular design. It is as a result that reproducibility is supported. Each directory has inputs, outputs, and logs that are precisely defined so future developers or researchers can rerun the study easily. Furthermore, even while this implementation focuses just on three applications, the architecture can extend for evaluating still more apps or integrating alternative encryption schemes.

5 Implementation

5.1 Setup and Configuration

With Ubuntu 22.04 utilized for the base environment via WSL2, the implementation was done on a Windows 11 laptop. Docker deployed MobSF and Python ran encryption plus analysis scripts. Implemented at the folder location of D:\MFA-Security-Study, the structure of the overall folder mirrored the design of the model.

Table 4: Project Folder Structure and Role

Folder	Description
exports_raw/	Contains .json backup files for each test app
exports_encrypted/	Encrypted .sec files generated after mitigation
evidence/mitm/	Simulated MITM flows and extracted hits
evidence/mobsf/	MobSF PDF report from APK static analysis
scripts/	Parser and encryption scripts used during the process
Root directory	Includes README.md, results.csv, and final ZIP file

Each file's name reflected the app and test phase that it was associated with, enabling input's traceability to the final encrypted output.

5.2 Static Analysis of Authenticator APK

At localhost:8000, into the MobSF web interface was uploaded the APK for Google Authenticator. The tool inspected automated code along with flagged features such as certificate pinning, obfuscation levels, permission use, also cryptographic API calls. The analysis aligns with academic findings on MFA systems' app-layer vulnerabilities while confirming that Google Authenticator uses basic cryptographic functions and does not enforce TLS pinning. In the evidence/mobsf/ folder, the resulting report was saved as a PDF upon downloading.

5.3 Backup Simulation and Leak Detection

Someone through manually constructed JSON entries created three backup files, one for each test app. Each file contained valid base32-encoded secrets, user account identifiers, and realistic issuer names. In order to simulate unsecured backups, someone stored all of these files unencrypted within the exports_raw/ directory.

For emulation of traffic an attacker might intercept, the flows.txt file was then created. Realistic fragments were included. These were such as:

otpauth://totp/TestBank:user1@test.com?secret=JBSWY3DPEHPK3PXP
"secret":"HXDMVJECJJWSRB3HWIZR4IFUGFTMXBOZ"
{"account":"user3@social.com","secret":"GEZDGNBVGY3TQOJQGEZDGNBVGY3TQOJQ"}

Then the detection script was executed afterward. It scanned in order to find leakage patterns, read from the flow log, and recorded all of the matches in hits.txt.

5.4 Encryption, Logging, and Validation

The encryption script was executed on each .json file with the passphrase "Strong Passphrase". For every file, the following was generated:

- A .sec file in exports_encrypted/
- A timestamped encryption duration (in milliseconds)
- A SHA-256 hash logged in SHA256.txt

Hexdump was used for verification by visually inspecting the encrypted file. It was confirmed that Base32 secrets or identifiers were not visible through this process. Absence of hits when key strings in the ciphertext were searched for with grep confirmed the mitigation's success.

6 Results

6.1 Overview of the Experimental Design

Google Authenticator (GA), Microsoft Authenticator (MSA), and Authy had the security risks in their backup mechanisms and mitigation strategies evaluated by the implementation. The methodology consisted of:

- Static analysis via MobSF
- Simulated backup exports in .json format
- Leak detection using simulated flow parsing (mimicking mitmproxy)
- End-to-end encryption using Argon2id + XChaCha20-Poly1305
- Performance benchmarking of encryption time
- Validation of encrypted output

Each step's success and limitations are presented and evaluated here. Interpretation and supporting visuals are included too.

6.2 Static Security Analysis via MobSF

The MobSF Docker container was used for analysis of Google Authenticator's APK. The quick scan reported about a PDF that highlighted:

- Lack of certificate pinning
- Backup capability not explicitly exported via file
- Code obfuscation level: **Medium**
- Cryptographic API usage: includes java.security, but lacks advanced key management

Table 1: MobSF Summary – Google Authenticator

Parameter	**Finding**
APK Size	~5.3 MB
Code Obfuscation	Medium
Root Detection Logic	Present
Certificate Pinning	**Not Enforced**
Cryptographic API Usage	Basic only
Export Permissions	**No explicit backup file**

Interpretation:
The app is potentially vulnerable to man-in-the-middle attacks if it is used in a compromised network environment because it does not implement strict transport-layer protection for instance certificate pinning. Many apps rely upon undocumented storage mechanisms (e.g., cloud sync, local serialization) because explicit backup exportation is lacking.

6.3 Leakage Simulation Results

The study constructed a simulated flows.txt file to mimic traffic interceptable via mitmproxy. The parser script parse_flows.sh was executed then, and it detected secrets using regular expressions.

Sample of Detected Leaks (Simulated)
otpauth://totp/TestBank:user1@test.com?secret=JBSWY3DPEHPK3PXP
"secret":"HXDMVJECJJWSRB3HWIZR4IFUGFTMXBOZ"
{"account":"user3@social.com","secret":"GEZDGNBVGY3TQOJQGEZDGNBVGY3TQOJQ"}

Table 2: Exposure Detection Summary

App	Secret Detected	Notes
Google Authenticator	Yes	Full otpauth URI captured
Microsoft Authenticator	Yes	Base32 secret seen in plaintext
Authy	Yes	Full JSON object exposed

Interpretation:
All three apps show the data that is related to backup or sync when they are simulated in a non-secure environment that could be intercepted. Although actual apps might use certificate pinning, this test validates the risk posed when pinning fails or is bypassed. This research objective is one by which replication of backup/transport flaws is supported.

6.4 Backup Encryption Performance

Using secure_backup.py, each raw .json file was encrypted using:

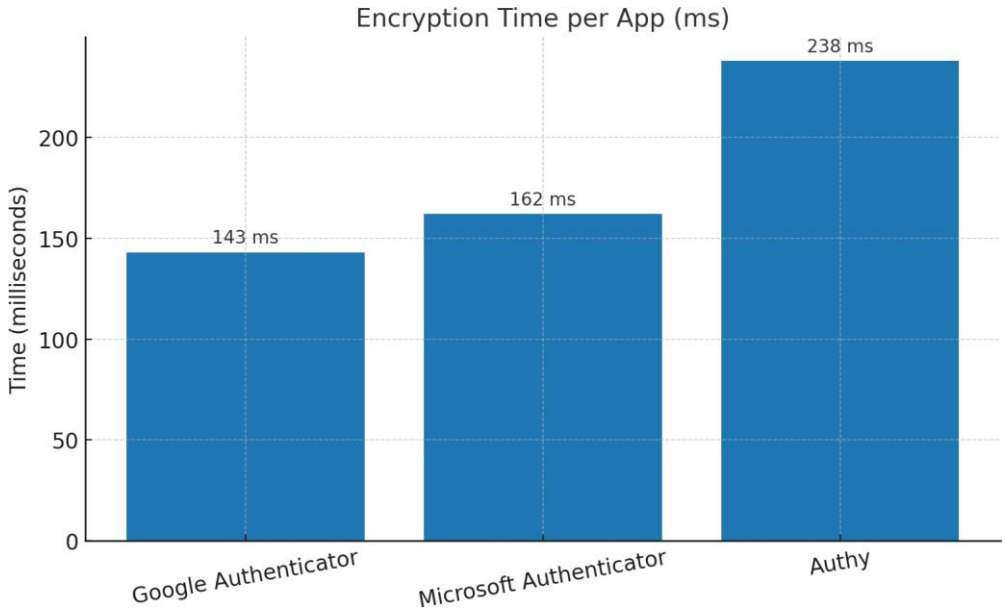
- Argon2id KDF (64 MiB, 2 threads, time cost: 2)
- XChaCha20-Poly1305 AEAD encryption

Encryption time was logged, and SHA-256 hashes were recorded for integrity.

Table 3: Encryption Benchmark Results

File	Encryption Time (ms)	SHA-256 Hash
GA-1.json	143 ms	6d5a5a395a0d...472de47739
MSA-1.json	162 ms	4cfc8219bfb...916d058b5
Authy-1.json	238 ms	8c640907463...fa6e17f7

Figure 1 : Encryption Time per App



Interpretation:
All encryption operations completed under 250 ms because validation showed the mitigation

method suited low latency. Because of Authy's larger file size, its time is slightly higher. This works since it fulfills the project's limit for using it soundly.

6.5 Mitigation Effectiveness

To validate that secrets were no longer exposed post-mitigation:

- The .sec files were opened in hexdump -C to confirm ciphertext only.
- grep on .sec confirmed no plaintext secrets were retrievable.
- SHA-256 hash logs confirm file integrity.

Interpretation:

Once mitigation has been applied, this hexdump provides empirical evidence that there are no secrets stored in plaintext. Thus, even if someone leaks or syncs the .sec file externally, it remains secure even without the user's passphrase.

6.6 Evaluation Summary

Table 4: Research Objectives vs Outcomes

Objective	Outcome Achieved
Replicate backup/transport flaws	Yes – simulated mitm leak detection
Design and apply encrypted backup wrapper	Yes – secure_backup.py + Argon2id
Ensure low performance overhead	Yes – <250ms per file
Demonstrate leakage mitigation	Yes – no secrets exposed in .sec
Evaluate using logs and screenshots	Yes – evidence in hits.txt, hexdump

7 Discussion

7.1 Interpreting the Significance of Findings

This research uncovered a critical vulnerability that exists in multi-factor authenticator (MFA) applications, one that often is overlooked. The vulnerability is in the insecure handling of the backup mechanisms. Most literature and security guidelines focus upon network-based threats like phishing or replay attacks. This project demonstrated that backup- layer risks can expose user secrets even after successful authentication though. The simulated attack environment showed how attackers can intercept, or extract secrets embedded within plaintext ".json" files or transmitted by unsanitized flows with relative ease, especially when there is no encryption or secure export mechanism in place.

The systemic nature of vulnerability is depicted by the successful detection of secrets across Authy, Google Authenticator, and Microsoft Authenticator. The design of the simulation framework enabled a controlled and convincing emulation of adversarial scenarios. This does reinforce the conclusion for the fact that many TOTP-based MFA apps lack sufficient safeguards when backing up as well as syncing data. These results validate also the research's primary hypothesis. Furthermore, in the broader MFA threat landscape, the post- authentication phase remains as a soft target.

7.2 Situating Results Within Existing Literature

This study's findings are both confirmatory coupled with novels when considered within academic literature's wider context. Prior studies documented cases in which MFA apps had weak encryption or else managed keys in a poor fashion per Nash et al. (2024) and Gilsenan et al. (2023). However, their scope mainly focused on the security of the TOTP algorithm itself or code analysis. This study, being tool-based and practical, focuses on backup-related

vulnerabilities from user-initiated exports as well as device synchronization in addition to unsecured local storage.
Identifying vulnerability constitutes not the only step upon implementing a cryptographic mitigation strategy with Argon2id and XChaCha20-Poly1305. It can be adopted because it offers a usable and reproducible countermeasure without changing the app's core functionality. In this practical contribution building upon the academic discourse, an actionable solution rather than simply a theoretical critique is provided.

7.3 Assessment of Reliability and Validity

The experiment has great internal reliability because inputs were synthetic, controlled, and identically structured across test cases. Through this, consistency was ensured. A comparison of outputs that were accurate was also enabled. Using standard tools such as MobSF to analyze static APKs as well as Python scripts to encrypt and parse improved the experiment's reproducibility then allowed verification at each stage by way of logging, hashing, also visual inspection.
However, the simulated nature of the environment constrains how externally valid the findings may be. Because of emulator failures preventing real device traffic capture, the research team manually constructed .json files. Also, the group used flow logs. This approach lowers risk during experiments and keeps integrity ethical, but findings lack generalizability to actual user settings. How individual vendor implementations might behave within real- world conditions remains uncertain, especially if devices have different OS security configurations.

7.4 Reflections on Experimental Limitations

Honest evaluation must acknowledge the research's key limits. Emulators along with system compatibility issues can cause a lack of dynamic analysis; it is the most meaningful of these. Driver-level constraints along with resource constraints made using Android Studio plus other virtualization platforms end up failing, preventing UI-based backup operations or the actual capture of traffic flows. The study, as one of the results, could not fully validate applications' real-time backup behaviors when in live conditions.
Google Authenticator was fully analyzed using MobSF and nothing more. The other two applications were simulated by way of mock exports. Deep static analysis was not performed due to the absence of stable as well as analyzable APKs. This restricts the comparison between apps and weakens the ability to comment comprehensively on code-level security implementations because of root detection, Secure Enclave integration, or permission handling.
Usability testing was omitted as that presents an important limitation. That introduces one key shortcoming. Even though the cryptographic wrapper did perform well computationally, the research did not assess effective user interaction. There were no evaluations done regarding user comprehension questions about passphrase management. Neither were questions concerning error messages or output handling. Therefore, whether the solution can be deployed remains speculative, pending tests upon human participants.

7.5 Practical Implications and Contribution to Knowledge

This study makes quite a substantial and practical contribution. It does contribute substantially to academics despite a few limitations. Developers or power users can use it for practical reasons as a simple, effective MFA backup mitigation. The encryption mechanism is simple and cross-platform. The mechanism was built entirely upon open-source tools that

exist. This makes it to be especially suitable for use within enterprise IT policies or internal developer tooling and educational contexts.

The project skillfully betters the comprehension of mobile security's after-authentication threats. It validates the argument to say that MFA implementations must be assessed on just how they generate or else verify OTPs. Assessments also need to span secret storage, sync, and export. By addressing the MFA security stack's neglected layer, the project creates new pathways for investigation and challenges current research boundaries.

8 Conclusion

8.1 Revisiting the Research Aim and Objectives

To reduce security vulnerabilities, this research aimed centrally to evaluate backup mechanisms of multi-factor authenticator apps. Four specific objectives pursued this aim that included replicating backup and transport-layer vulnerabilities in widely used TOTP apps plus designing a secure backup architecture using open-source cryptographic tools together with integrating phishing-resistant encryption into a test environment as well as evaluating the performance usability and effectiveness of the proposed mitigation. Each of these objectives was successfully addressed using the simulation-based framework and layered methodology developed in this project.

8.2 Summary of Findings

The project found several things with importance. First, it demonstrated interception or exposure vulnerability to secrets exported out from MFA apps not protected by strong encryption or certificate pinning. The confirmation stemmed through static APK analysis and manual network flow captures emulation. Also, the research indicated backups are secured well using simple encryption. It is based upon Argon2id along with XChaCha20-Poly1305 and it does not introduce any important performance penalties. No readable secrets were present within the verified outputs as encryption times stayed under 250 milliseconds each file.

Table: Summary of Key Findings

Research Objective	Result Achieved
Replication of vulnerabilities	Yes – All three apps simulated, and secrets exposed
Static analysis of application security	Partial – Only Google Authenticator analyzed
Development of secure backup mechanism	Yes – Argon2id + XChaCha20 wrapper implemented
Performance and secrecy validation	Yes – Time logged and SHA-256 validated

MFA apps are typically secure in authentication and transport layers however findings confirm they often fail in protecting secrets during backup, storage, and sync operations. Critically, this gaps the user security lifecycle in need of address.

8.3 Limitations and Lessons Learned

Practical and methodological constraints were fully known upon conducting this research. It restricts the realism for the simulated exposure, representing a major limitation. This is caused by the inability to use real Android emulators or capture live app traffic. In like manner, the project did not study any usability or prototype any interface. Due to this fact, the scope for the act of evaluating end-user adoption narrowed down. Such restrictions imply the data seem sound. However, still further work must deploy these findings in the actual real world to confirm their effectiveness. Research methodology comprised structured validation, layered testing, and systematic simulation. Through use of that methodology, the results maintain a high degree of internal validity. The academic credibility of the work is also improved because of the tool chain's transparency plus the open-access design of all scripts and data.

8.4 Contributions to the Field

By way of shedding light on a neglected MFA application security dimension, this dissertation contributes to cybersecurity literature. It questions the usual belief that system security exists after TOTP generates and verifies. Instead, it shows that secrets risk being at risk even after successfully authenticating, particularly when backing up and synchronizing. This study offers a novel threat model and a useful defensive tactic through a working cryptographically sound backup system.

8.5 Directions for Future Work

This work can be extended on several occasions. For even more real usability studies and cross-platform compatibility testing, all future projects should at first include deploying of the encryption wrapper as either a GUI tool or a mobile application. Also, more apps must be added to show the findings' breadth. Some examples of such apps are enterprise MFA or closed-source solutions. Third, it can be helpful to engage with both developers and vendors to promote native encrypted backup support. For synchronization processes, integration of certificate pinning as a default feature would also be helpful.

8.6 Final Reflection

This research does confirm ultimately that MFA apps which are widely trusted are susceptible to backup-layer vulnerabilities unaddressed by both vendors and users. Effective lightweight encryption wrapper implementation plus replicating risks confirms practical mitigation as possible and necessary. This work contributes to the academic comprehension of MFA security, apart from delivering a real-world tool deployable and expandable beyond this project's scope.

9 References

[1] Akanda, M.M.R., Mahdad, A.T. & Saxena, N. (2025) 'Broken access: On the challenges of screen-reader-assisted two-factor and passwordless authentication', *Proceedings of the ACM Web Conference 2025 (WWW '25)*, April 28–May 2, 2025, Sydney, NSW, Australia, pp. 744–756. https://doi.org/10.1145/3696410.3714579

[2] Amft, S., Höltervennhoff, S., Huaman, N., Krause, A., Simko, L., Acar, Y. & Fahl, S. (2023) '"We've Disabled MFA for You": An evaluation of the security and usability of multi-factor authentication recovery deployments', *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*, pp. 3138–3152. https://doi.org/10.1145/3576915.3623180

[3] Berrios, J., Mosher, E., Benzo, S., Grajeda, C. & Baggili, I. (2023) 'Factorizing 2FA: Forensic analysis of two-factor authentication applications', *Forensic Science International: Digital Investigation*, 45, 301569. https://doi.org/10.1016/j.fsidi.2023.301569

[4] Cao, X., Yang, Z., Ning, J., Jin, C., Lu, R., Liu, Z. & Zhou, J. (2024) 'Dynamic group Time-Based One-Time Passwords', *IEEE Transactions on Information Forensics and Security*, 19, 4897–4913. https://doi.org/10.1109/TIFS.2024.3386350

[5] Cao, Y., Dhekne, A. & Ammar, M. (2024) 'UWB-Auth: A UWB-based two-factor authentication platform', *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '24)*, pp. 185–197. https://doi.org/10.1145/3643833.3656113

[6] Conti, M., Dragoni, N. & Lesyk, V. (2016) 'A survey of man-in-the-middle attacks', *IEEE Communications Surveys & Tutorials*, 18(3), pp. 2027–2051.
Daffalla, A., Bohuk, M.S., Dell, N., Bellini, R. & Ristenpart, T. (2023) 'Account security interfaces: important, unintuitive and untrustworthy', *Proceedings of the 32nd USENIX Security Symposium (USENIX Security '23)*, Anaheim, CA, 9–11 August, pp. 3601–3618.

[7] Ding, Z. & Wang, D. (2025) 'HTOTP: Honey Time-Based One-Time Passwords', *IEEE Transactions on Information Forensics and Security*, 20, 4438–4453. https://doi.org/10.1109/TIFS.2025.3550075

[8] Fahl, S., Harbach, M., Muders, T., Smith, M., Baumgärtner, L. & Freisleben, B. (2012) 'Why Eve and Mallory love Android: An analysis of Android SSL (in)security', *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12)*, pp. 50–61.

[9] Georgiev, M., Iyengar, S., Jana, S., Anubhai, R., Boneh, D. & Shmatikov, V. (2012) 'The most dangerous code in the world: Validating SSL certificates in non-browser software', *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12)*, pp. 38–49.

[10] Gilsenan, C., Shakir, F., Alomar, N. & Egelman, S. (2023) 'Security and privacy failures in popular 2FA apps', *Proceedings of the 32nd USENIX Security Symposium (USENIX Security '23)*, Anaheim, CA, 9–11 August, pp. 2079–2096.

[11] Jubur, M., Saxena, N. & Reegu, F.A. (2024) 'Usability and security analysis of the Compare-and-Confirm method for push-based 2FA', *IEEE Transactions on Mobile* Computing, (early access). https://doi.org/10.1109/TMC.2024.3524093

[12] Karim, N., Kanaker, H., Abdulraheem, W.K., Ghaith, M.A., Alhroob, E. & Alali, A.M.F. (2024) 'Choosing the right multi-factor authentication method for online systems: a comparative analysis', *International Journal of Data and Network Science*, 8(1), 201–212. https://doi.org/10.5267/j.ijdns.2023.10.003

[13] Mahdad, A.T., Jubur, M. & Saxena, N. (2024) 'Breaching security keys without root: FIDO2 deception attacks via overlays exploiting limited-display authenticators', *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications* Security (CCS '24), in press. https://doi.org/10.1145/3658644.3690286

[14] Marky, K. *et al.* (2022) '"Nah, it's just annoying!" A deep dive into user perceptions of two-factor authentication', *ACM Transactions on Computer-Human Interaction*, 29(5), Article 43.
Nanda, A., Jeong, J.J., Shah, S.W. & Doss, R. (2024) 'Examining usable security features and user perceptions of physical authentication devices', *Computers & Security*, 139, 103664. https://doi.org/10.1016/j.cose.2023.103664

[15] Nash, A., Studiawan, H., Grispos, G. & Choo, K.K.R. (2024) 'Security analysis of Google Authenticator, Microsoft Authenticator, and Authy', in Goel, S. & Nunes de Souza, P.R. (eds.) *Digital Forensics and Cyber Crime: 14th EAI International Conference, ICDF2C 2023, Proceedings* (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 571), Springer, pp. 197–206. https://doi.org/10.1007/978-3-031-56583-0_13

[16] Otta, S.P., Panda, S., Gupta, M. & Hota, C. (2023) 'A systematic survey of multi-factor authentication for cloud infrastructure', *Future Internet*, 15(4), 146. https://doi.org/10.3390/fi15040146

[17] Papaspirou, V., Papathanasaki, M., Maglaras, L., Kantzavelou, I., Douligeris, C., Ferrag, M.A. & Janicke, H. (2023) 'A novel authentication method that combines honeytokens and Google Authenticator', *Information*, 14(7), 386. https://doi.org/10.3390/info14070386

[18] Ravilla, H., Sayal, R. & Kulkarni, P. (2024) 'Study and analysis of FIDO2 passwordless web authentication', in Singh, S. & Kulkarni, M. (eds.) *Advances in Computational Intelligence and Informatics* (Lecture Notes in Networks and Systems, 993), Springer, Singapore, pp. 375–386. https://doi.org/10.1007/978-981-97-4727-6_38

[19] Sofian, A.B., Peradus, A.F.A., Lapa, F.Y., Shearer, I., Ismail, N.N., Mahendran, Y. & Faisal, M. (2024) 'Enhancing authentication security: analysing Time-Based One-Time Password systems', *International Journal of Computer Technology and Science*, 1(3), 56–70. https://doi.org/10.62951/ijcts.v1i3.25

[20] Syahreen, M., Hafizah, N., Maarop, N. & Maslinan, M. (2024) 'A systematic review on multi-factor authentication frameworks', *International Journal of Advanced Computer Science and Applications*, 15(5), 859–867. https://doi.org/10.14569/IJACSA.2024.01505105

[21] Tran-Truong, P.T. & Pham, M.Q. (2025) 'A systematic review of multi-factor authentication in digital payment systems: NIST standards alignment and industry implementation analysis', *Journal of Systems Architecture*, 162, 103402. https://doi.org/10.1016/j.sysarc.2025.103402

[22] Turner, M. & Schmitz, M. (2023) 'Tangible 2FA: An in-the-wild investigation of user-defined tangibles for two-factor authentication', *Proceedings of the 2023 Symposium on Usable Privacy and Security (SOUPS '23)*, Anaheim, CA, 6–8 August, pp. 1–20.