Science, Engineering & Education, 10, (1), 2025, 19-23

Algorithmic Approach to Secure Two-Factor Authentication Using Time-Based One-Time Passwords in Desktop Environments

Ivaylo Atanasov*

University of Chemical Technology and Metallurgy, 8 Kliment Ohridski Blvd., Sofia 1797, Bulgaria

Received 05 December 2024, Accepted 17 March 2025
DOI: 10.59957/see.v10.i1.2025.3

ABSTRACT

Time-Based One-Time Password (TOTP) systems are frequently utilized in modern security protocols in addition to two-factor authentication (2FA) because of their simplicity and strong security measures. Although many TOTP options target mobile devices, there is a noteworthy demand for solutions tailored to desktops systems because of factors such as user preferences, improved security measures, disaster recovery, and convenience for desktop-oriented tasks. This article presents a sophisticated algorithmic method for handling TOTP on desktop environments, meeting these requirements while upholding strict security and user friendliness criteria. The suggested method may guarantee the safe and easy creation of TOTP through the incorporation of complex algorithms and techniques, offering adaptability and strong disaster recovery measures. This approach boosts the connection with desktop security protocols, enhances usability for users without mobile devices, and simplifies the authentication process for those who mainly use desktop systems or laptop setups.

Keywords: 2FA, multi-factor authentication, TOTP, cryptography, computer security, cybersecurity.

INTRODUCTION

2FA has been considered the cornerstone of modern security practices [1], wherein an additional layer of protection against simple password-based systems is introduced [2, 3]. Probably the most popular and widely adopted form of 2FA is the so-called TOTP-a one-time password based on the current time [4]. Instead, it grants users with a unique temporary code they are required to input alongside their regular password while logging into accounts. With TOTP, it is inclusions of security and simplicity that make it very popular to protect sensitive information on a wide array of platforms. Most of the TOTP implementations are designed for mobile devices and rely on applications that generate these one-time codes. But that again faces limits of dependence on the availability and security of the mobile device. While mobile-based solutions are convenient and popular, there is an increasing requirement for desktop-based solutions-especially cases where mobile devices may be impractical or undesirable [5]. TOTP desktop implementations will bring in much-needed integration with desktop security, better accessibility for users whose mobile phones are not compatible, and strong disaster recovery. The paper comes up with a provision

**Correspondence to: Ivaylo Atanasov, University of Chemical Technology and Metallurgy*
8 Kliment Ohridski Blvd., Sofia 1797, Bulgaria, e-mail: ivaylo@uctm.edu

for an advanced algorithmic handling approach to TOTP on desktop environments for such needs while ensuring very high standards of security and usability.

ADVANCED ALGORITHMIC APPROACH TO SECURE 2FA USING TOTP

Each step of the algorithm is designed to maintain high security standards while providing a seamless experience for the user. Fig. 1 shows the high-level architecture of the suggested algorithm and gives a visual depiction of the workflow that connects the various processing phases.

The user's key that is utilized for both authentication and further cryptographic operations is retrieved via a password input at the start of the architecture. Secure hashing of user credentials using random salt and comparison against safely stored hashed values are two steps in the authentication process. The technique smoothly switches to the creation and display of Time-Based One-Time Passwords (TOTP) upon successful authentication. The user key is used to decrypt encrypted secret strings retrieved from a secure repository, allowing the production of TOTP codes for the user's accounts in real-time. This ongoing process ensures that users can effectively access and manage their one-time authentication codes:

1. User authentication

- Initialization: At startup, the user enters a password to retrieve the user key, used for both authentication and file encryption/ decryption
- Hashing: The entered credentials are hashed with random salt
- Verification: The hashed value is compared with a securely stored version
- Rate limiting: Maximum 3 attempts and 5 min lockout after exceeding limit.
- Outcome:
- Success: If there is a match, the user is authenticated and can proceed to the main menu of the program
- Failure: If there is a mismatch, an error message is displayed, and the user is prompted to try again or exit the program.

1. Logging
- Complete audit logging of critical operations
- Error and authentication attempt logging

2. Displaying TOTP CodesRetrieval – Encrypted secret strings are retrieved from a secure repository.
- Decryption: Secrets are decrypted using the user key.
- Generation: The decrypted data is used to generate and display the current TOTP codes for the user's accounts.
- Operation: The program displays TOTP codes that refresh every 30 s and auto-closes after 5 min of inactivity.

3. Add New Account (Optional)
- Initiation: The user provides an image file containing a QR code.

4. QR code processing and secure storage
- Decoding: The QR code image is processed using Open-Source Computer Vision Library (cv2) for image preprocessing, then the QR code data is decoded using the pyzbar library to extract the encoded secret string [6].
- Validation: The extracted secret string is validated based on the RFC 6238 standard [7].
- Key Derivation: The user key can be derived using a Key Derivation Function (KDF) such as PBKDF2 [8], bcrypt [9] or Argon2 [10, 11] for enhanced security.
- Encryption: The validated secret string is encrypted with a strong symmetric encryption algorithm (AES-256) using the user-derived key [12].
- Storage: The encrypted strings are securely stored on the user's computer using robust storage security mechanisms.
- Cleanup: The file containing the QR image is securely deleted.

5. Backup and Restore

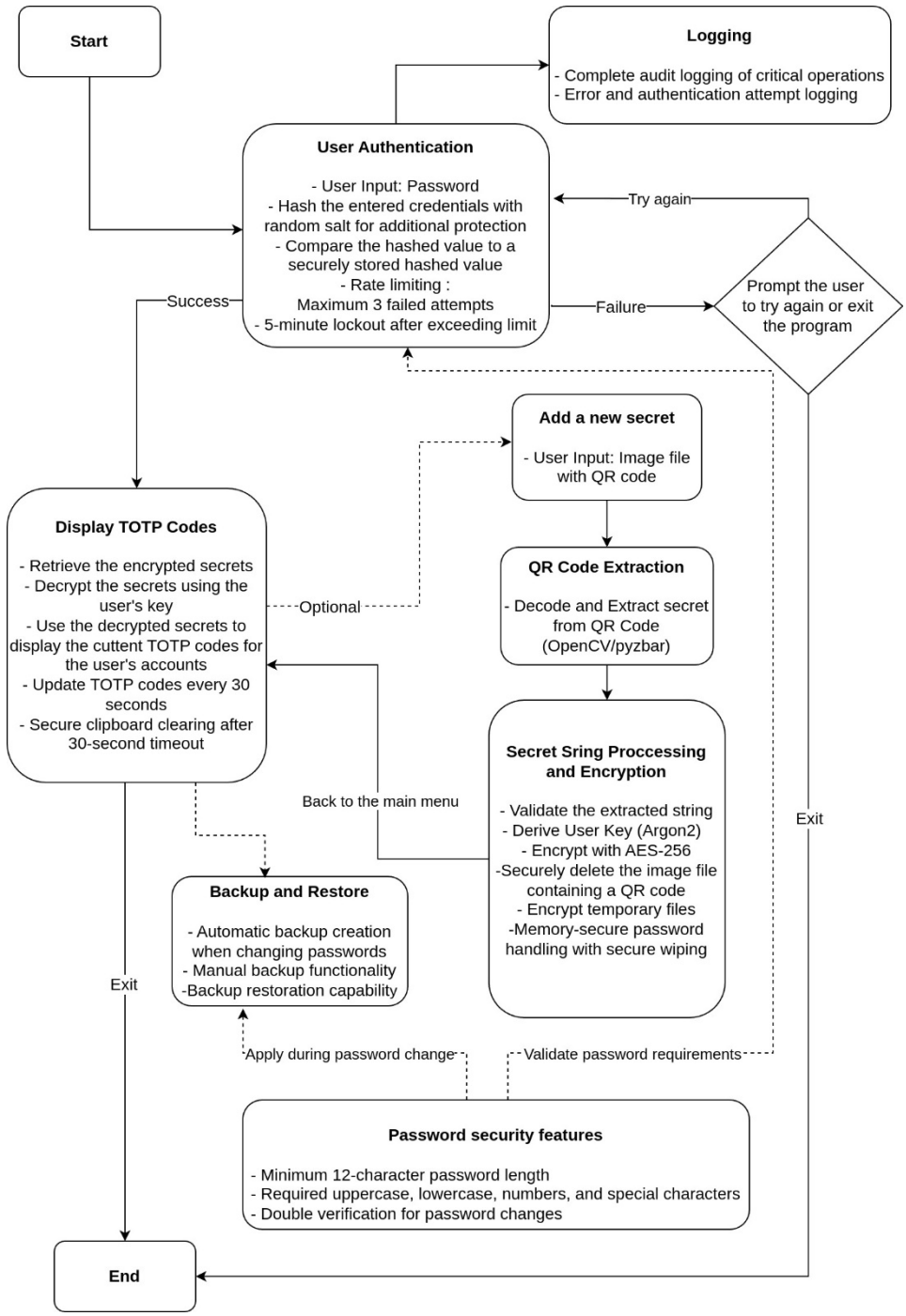

Fig. 1. High-level architecture of the proposed algorithm.

- Automatic backup creation when changing passwords
- Manual backup functionality
- Backup restoration capability

A client program was developed based on this algorithm. Parts of the code implemented in Python are shown in Fig. 2.

Upon authenticating via username and password to the server's web interface, a one-time generated TOTP verification code obtained from the TOTP Authenticator v1.0 program should be entered, as illustrated in Fig. 3.

```python
import shutil
from PIL import Image
from pyzbar.pyzbar import decode
import time
from functools import wraps
import secrets
import ctypes
from threading import Timer, Lock
from io import BytesIO
import weakref
from Cryptodome.Cipher import AES
from Cryptodome.Random import get_random_bytes
import argon2

# Basic settings
CONFIG = {
    'MAX_LOGIN_ATTEMPTS': 3,
    'LOGIN_TIMEOUT': 300,
    'CLIPBOARD_TIMEOUT': 30,
    'PASSWORD_MIN_LENGTH': 12,
    'LOG_FILE': os.path.expanduser('~/.2fa-secrets-new/argon/totp.log'),
    'BACKUP_DIR': os.path.expanduser('~/.2fa-secrets-new/argon/backups'),
    'TEMP_DIR': os.path.expanduser('~/.2fa-secrets-new/argon/temp'),
    'TOTP_PERIOD': 30,
    'ARGON2_TIME_COST': 10,
    'ARGON2_MEMORY_COST': 409600,
    'ARGON2_PARALLELISM': 8,
    'SALT_LENGTH': 32
}
# **Security Note:** Sensitive secrets extracted from QR codes are stored
# in an encrypted format,
# Path to file containing encrypted secrets (DO NOT COMMIT)
ENCRYPTED_FILE_PATH = os.path.expanduser('~/.2fa-secrets-new/argon/encrypted_totp_secrets.bin')
LABELS_FILE_PATH = os.path.expanduser('~/.2fa-secrets-new/argon/totp_labels.bin')
FAILED_ATTEMPTS_FILE = os.path.expanduser('~/.2fa-secrets-new/argon/failed_attempts.json')
```

```python
class CryptoOperations:
    @staticmethod
    def derive_key(password: str, salt: bytes) -> bytes:
        hasher = argon2.low_level.hash_secret_raw(
            password.encode('utf-8'),
            salt,
            time_cost=CONFIG['ARGON2_TIME_COST'],
            memory_cost=CONFIG['ARGON2_MEMORY_COST'],
            parallelism=CONFIG['ARGON2_PARALLELISM'],
            hash_len=32,
            type=argon2.low_level.Type.ID
        )
        return hasher

    @staticmethod
    def encrypt_data(data: str, password: str, salt: bytes) -> tuple:
        key = CryptoOperations.derive_key(password, salt)
        cipher = AES.new(key, AES.MODE_GCM)
        ciphertext, tag = cipher.encrypt_and_digest(data.encode())
        return salt, cipher.nonce, tag, ciphertext

    @staticmethod
    def decrypt_data(encrypted_data: tuple, password: str) -> str:
        salt, nonce, tag, ciphertext = encrypted data
```

Fig. 2. Key python code snippet for TOTP desktop implementation.

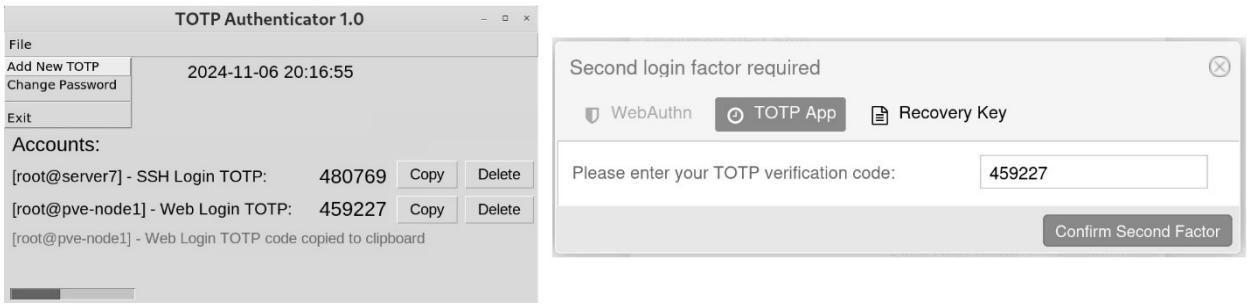

Fig. 3. Two-factor authentication process.

CONCLUSIONS

The advanced 2FA algorithm and client provides a user-friendly solution to handle TOTP authentication in desktop systems. This utility application uses strong encryption for the files that contain the sensitive secret keys using advanced cryptographic libraries which guarantee data integrity and confidentiality of the user authentication information against unauthorized access. This TOTP desktop-based solution is a rich alternative to mobile authentication applications for users who would prefer the use of desktops or whose phones simply cannot support some authentications. The client easily integrates into desktop-centric workflows for ease of access, hence enhancing efficiency in an authentication exercise. Furthermore, the program provides robust options for disaster recovery, which increases security and user confidence. Should their mobile device be lost or compromised, users can revert to their secrets backed up and encrypted on their computer. This ensures the possibility of users regaining quick access to their accounts without the usual downtime or security risks normally associated with losing a mobile device. The developed TOTP client turns convenience into a secure experience through strong security protocols and an accessible,

user-friendly interface. It offers a comprehensive solution to satisfy a variety of different needs by users in various situations and to keep their sensitive information continuously protected in desktops.

Proposed future system enhancements encompass cloud backup integration, system keyring integration for credential management, and automated security update deployment mechanisms.

REFERENCES

1. I. Atanasov, From Firewall to AI: Strengthening Linux Server Security, Science, Engineering & Education, 9, 1, 2024, 3-12.
2. J. Williamson, K. Curran, Best Practice in Multi-factor Authentication, Semiconductor Science and Information Devices, 3, 1, 2021.
3. B.S. Archana, A. Chandrashekar, A.G. Bangi, B.M. Sanjana, S. Akram, Survey on usable and secure two-factor authentication, Proc. 2nd IEEE Int. Conf. Recent Trends Electron. Inf. Commun. Technol. (RTEICT), Bangalore, India, 2017. DOI: 10.1109/RTEICT.2017.8256716
4. J.X. Zhao, Research and Design on an Improved TOTP Authentication, Appl. Mech. Mater., 411-414, 2013, 595-599.
5. J. Berrios, E. Mosher, S. Benzo, C. Grajeda, I. Baggili, Factorizing 2FA: Forensic analysis of two-factor authentication applications, Forensic Sci. Int. Digit. Investig., 45, 2023, 301569. https://doi.org/10.1016/j.fsidi.2023.301569
6. G.R. Bradski, et al., Learning OpenCV - Computer Vision with the OpenCV Library, IEEE Robot. Autom. Mag., 16, 3, 2009, 100-100.
7. D. M'Raihi S. Machani, M. Pei, J. Rydell, RFC 6238 TOTP: Time-Based One-Time Password Algorithm, Internet Eng. Task Force, 2011.
8. A.F. Iuorio, A. Visconti, Understanding Optimizations and Measuring Performances of PBKDF2, Proc. 2nd Int. Conf. Wireless Intell. Distrib. Environ. Commun., 2019.
9. T.P. Batubara, S. Efendi, E.B. Nababan, Analysis Performance BCRYPT Algorithm to Improve Password Security from Brute Force, J. Phys.: Conf. Ser., 2019.
10. A. Biryukov, D. Dinu, D. Khovratovich, S. Josefsson, Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications, RFC 9106, 2021.
11. G. Song, S. Eum, H. Kwon, M. Sim, M. Lee, H. Seo, Optimized Quantum Circuit for quantum security strength analysis of Argon2, Electronics, 12, 21, 2023, 4485. https://doi.org/10.3390/electronics12214485
12. K.H. Brown, Advanced Encryption Standard (AES), NIST, 2023.