

Decentralized Backup and Recovery of TOTP Secrets

Conor Gilsenan
conorgilsenan@berkeley.edu
UC Berkeley

Noura Alomar
nnalomar@berkeley.edu
UC Berkeley

Andrew Huang
87andrewh@berkeley.edu
UC Berkeley

Serge Egelman
egelman@berkeley.edu
UC Berkeley

ABSTRACT

This work proposes a set of security, privacy, and usability design requirements for the backup and recovery systems of apps implementing the Time-based One-Time Password (TOTP) algorithm, a widely deployed method of two-factor authentication (2FA). We explain how several prevalent apps fail to satisfy these requirements and outline how our scheme leverages decentralized security techniques to satisfy the majority of these requirements and provide stronger security and privacy guarantees.

CCS CONCEPTS

• **Security and privacy** → **Multi-factor authentication**; *Usability in security and privacy*.

KEYWORDS

two factor authentication, 2FA, time-based one-time passwords, TOTP, backup, recovery, usability, security, privacy

ACM Reference Format:

Conor Gilsenan, Noura Alomar, Andrew Huang, and Serge Egelman. 2020. Decentralized Backup and Recovery of TOTP Secrets. In *Hot Topics in the Science of Security Symposium (HotSoS '20)*, April 7–8, 2020, Lawrence, KS, USA. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/3384217.3386396>

1 INTRODUCTION AND RELATED WORK

To authenticate using TOTP [2], client devices use a shared secret to generate a unique OTP that can be verified by the server. If the client loses the shared secret, then it cannot generate the OTPs required for authentication and risks account lockout. In practice, the shared secret is easily lost when people lose a device, buy a new device, or uninstall the authenticator app. Dozens of consumer authenticator apps support the TOTP protocol [6, 8], but there is mixed support for backup/recovery of the shared secret and most architectures rely, in one way or another, on passwords.

In this work, we present a backup/recovery scheme that intentionally avoids passwords ("something you know"). Instead, we rely on "something you have" and build on previous work that explored "somebody you know" as an authentication factor. Brainard et. al. [10] built a system that allowed users to obtain assistance from pre-registered *helpers* when they lost their primary authentication factor (password, token). Schechter et. al. [11] built a system that allowed pre-appointed *trustees* to verify a user's identity during

account recovery. While both of these approaches relied on trusted central servers, our work explores a decentralized setting that assumes the central server is untrusted. Our main research focus is to study the usability of backup, recovery, and authentication systems that rely primarily on social networks. The prototype of our protocol discussed in this abstract is a step towards that goal.

2 DESIGN GOALS

We believe the following design goals prioritize security, privacy, and usability for a backup/recovery system for TOTP secrets. Several prevalent schemes rely on a centralized server to facilitate communication and provide short- or long-term storage.

R1: Backup & Recovery. Users must be able to recover TOTP secrets after losing all personal devices at once.

R2: No accounts on central server. Requiring the user to register a new account on the central server adds unnecessary friction and reveals personal information (email addresses, phone numbers). This new account must also be protected with a password and 2FA. Reintroducing the knowledge factor to protect the backups of the possession factor is somewhat circular logic and presents real usability challenges. The backup/recovery system should not require creating an account on the central server.

R3: High-entropy keys. One popular backup architecture is to encrypt TOTP secrets with keys derived from human provided passwords and store the ciphertext on the central server. Since people commonly choose weak and reused passwords, these keys will likely be low-entropy and vulnerable to offline attacks. TOTP secrets and associated metadata should be encrypted using randomly generated high-entropy keys.

R4: Infeasible access to plaintext. It must be computationally infeasible for the central server to access plaintext secrets.

R5: No centralized long-term storage. The central server should not be relied upon for long-term storage because it represents a single point of failure and tempting target for attackers.

3 ASSESSMENT OF PREVALENT 2FA APPS

Here, we assess several prominent TOTP 2FA authenticator apps against the proposed requirements (summary in Table 1).

Google Authenticator [7] does not offer any backup capability by design, so fails all requirements.

Authy 2FA Authenticator [4] encrypts TOTP secrets using a key derived from a password (violates **R3** and **R4**) and sends the ciphertext to Authy servers for long-term storage (violates **R5**). The Authy app requires a valid phone number, which is essentially a user account (violates **R2**).

LastPass Authenticator [3] stores TOTP secrets in the user's existing LastPass vault (violates **R2**), which is encrypted locally with a key derived from a password (violates **R3** and **R4**) and the ciphertext is stored on LastPass servers (violates **R5**).

Table 1: 2FA authenticator apps & design requirements

	R1	R2	R3	R4	R5
Google Authenticator	–	–	–	–	–
Authy 2FA Authenticator	●	–	–	–	–
LastPass Authenticator	●	–	–	–	–
Duo Mobile	●	◐	–	–	–
Microsoft Authenticator	●	–	●	◐	–
BLUES 2FA Authenticator	●	●	●	●	◐

● = satisfies; ◐ = partially satisfies; – = does not satisfy;

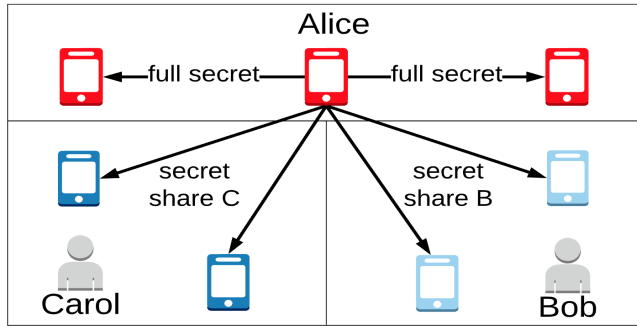


Figure 1: Adding a new TOTP secret; Alice has three paired personal devices (red) and her Trusted Friends, Carol and Bob, each have two paired personal devices (blues).

Duo Mobile [1] locally encrypts TOTP secrets using a key derived from a password (violates **R3** and **R4**) and stores the ciphertext in either Google Drive or iCloud (violates **R5**). We say that **R2** is partially satisfied because, even though a valid Google or iCloud account is required, Duo does leverage users' existing accounts and does not require a new account on a Duo service.

Microsoft Authenticator [9] encrypts TOTP secrets using randomly generated keys (satisfies **R3**) obtained from a Microsoft key service. Ciphertext is stored in iCloud for iOS devices and a Microsoft storage service for Android devices (violates **R5**). Since Microsoft can access the plaintext of Android backups¹ (they have both the key and ciphertext), but cannot access the plaintext of iOS backups (they cannot access the ciphertext), we say that **R4** is partially satisfied. An account is required to access the key- and storage- services² (violates **R2**).

4 BLUES 2FA AUTHENTICATOR APP

In this work, we present Blues 2FA, a protocol for the decentralized backup and recovery of TOTP secrets that provides stronger security and privacy guarantees and, we argue, a more compelling user experience than existing alternatives.

Adapting the Krypton secure pairing protocol [5], BLUES 2FA allows users to pair personal devices by scanning a QR code. Full TOTP secrets are synced across all personal devices in real-time, which allows users with multiple devices to *self-recover* if only a single device is lost.

¹https://twitter.com/Alex_T_Weinert/status/1195822117922562054

²https://twitter.com/Alex_T_Weinert/status/1195856795773751296

Users can also securely add Trusted Friends³ by scanning a QR code on the friend's device. Each full TOTP secret is encrypted with a random key and shares of that key are created using Shamir's secret sharing [12]. All devices of each Trusted Friend are sent the ciphertext and a share of the corresponding key. This prevents an individual Trusted Friend from generating valid OTPs, but allows the user to contact a configured subset of Trusted Friends to *socially recover* if they lose all personal devices at once. Figure 1 details the process of adding a new TOTP secret and shows how full secrets and secret shares are distributed among devices.

In our backup and recovery scheme, TOTP secrets are encrypted on client devices using randomly generated keys (satisfies **R3**). A central server allows anonymous devices (satisfies **R2**) to exchange end-to-end encrypted and authenticated messages (satisfies **R4**). Online devices can promptly retrieve and delete their messages from the central server. However, messages queued for offline devices could be lost before they are successfully retrieved, so we mark **R5** as only partially satisfied.

5 PILOT STUDY DESIGN AND FUTURE WORK

We implemented the user experience dictated by the Blues 2FA protocol and conducted a pilot of an in-lab survey driven study. Users enabled TOTP 2FA on a Dropbox account and logged in/out several times while completing a series of tasks which exercised *self-recovery* and *social recovery* functionality. We formed a concrete list of improvements that we will implement in our prototype before replicating this in-lab study at full scale. We also plan to execute a longitudinal study in a more realistic environment.

REFERENCES

- [1] [n.d.]. *Duo Restore - Guide to Two-Factor Authentication*. Retrieved December 13, 2019 from <https://guide.duo.com/duo-restore>
- [2] 2011. *TOTP: Time-Based One-Time Password Algorithm*. Retrieved October 02, 2019 from <https://tools.ietf.org/html/rfc6238>
- [3] 2017. *Announcing Cloud Backup for LastPass Authenticator: Easier multifactor security for everyone*. Retrieved December 13, 2019 from <https://blog.lastpass.com/2017/05/announcing-cloud-backup-for-lastpass-authenticator-easier-multifactor-security-for-everyone.html>
- [4] 2018. *How Authy 2FA Backups Work*. Retrieved December 13, 2019 from <https://authy.com/blog/how-the-authy-two-factor-backups-work/>
- [5] 2018. *Our Zero-Trust Infrastructure*. Retrieved February 28, 2020 from <https://krypt.co/blog/posts/krypton-our-zero-trust-infrastructure.html>
- [6] 2019. *Apple App Store Search Results - "authenticator"*. Retrieved October 02, 2019 from <https://www.apple.com/us/search/authenticator>
- [7] 2019. *Google Authenticator for Android (Open Source Version)*. Retrieved December 13, 2019 from <https://github.com/google/google-authenticator-android/blob/efac95c88ef8d9f8be3c887fbc2c2fd4f45dbde/README.md>
- [8] 2019. *Google Play Store Search Results - "2FA authenticator"*. Retrieved October 02, 2019 from <https://play.google.com/store/search?q=2fa%20authenticator&c=apps&hl=en>
- [9] 2019. *How it works: Backup and restore for Microsoft Authenticator*. Retrieved February 28, 2020 from <https://techcommunity.microsoft.com/t5/Azure-Active-Directory-Identity/How-it-works-Backup-and-restore-for-Microsoft-Authenticator/ba-p/1006678>
- [10] John Brainard, Ari Juels, Ronald L Rivest, Michael Szydlo, and Moti Yung. 2006. Fourth-factor authentication: somebody you know. In *Proceedings of the 13th ACM conference on Computer and communications security*. 168–178.
- [11] Stuart Schechter, Serge Egelman, and Robert W Reeder. 2009. It's not what you know, but who you know: a social approach to last-resort authentication. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 1983–1992.
- [12] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

³People known to the user in real life, such as a family member, friend, or co-worker.