Group Time-based One-time Passwords and its Application to Efficient Privacy-Preserving Proof of Location *

Zheng Yang[1], Chenglu Jin[2], Jianting Ning[3]**, Zengpeng Li[4], Tien Tuan Anh Dinh[5], and Jianying Zhou[5]

[1] Southwest University, Chongqing, China
youngzheng@swu.edu.cn
[2] Centrum Wiskunde & Informatica, Amsterdam, Netherlands
chenglu.jin@cwi.nl
[3] Fujian Normal University, Fuzhou, China
jtning88@gmail.com
[4] Shandong University, Qingdao, China
zengpeng@email.sdu.edu.cn
[5] Singapore University of Technology and Design, Singapore
{dinhtta,jianying_zhou}@sutd.edu.sg

Abstract. Time-based One-Time Password (TOTP) provides a strong second factor for user authentication. In TOTP, a prover authenticates to a verifier by using the current time and a secret key to generate an authentication token (or password) which is valid for a short time period. Our goal is to extend TOTP to the group setting, and to provide both authentication and privacy. To this end, we introduce a new authentication scheme, called Group TOTP (GTOTP), that allows the prover to prove that it is a member of an authenticated group without revealing its identity. We propose a novel construction that transforms any asymmetric TOTP scheme into a GTOTP scheme. Our approach combines Merkle tree and Bloom filter to reduce the verifier's states to constant sizes.
As a promising application of GTOTP, we show that GTOTP can be used to construct an efficient privacy-preserving Proof of Location (PoL) scheme. We utilize a commitment protocol, a privacy-preserving location proximity scheme, and our GTOTP scheme to build the PoL scheme, in which GTOTP is used not only for user authentication but also as a tool to glue up other building blocks. In the PoL scheme, with the help of some witnesses, a user can prove its location to a verifier, while ensuring the identity and location privacy of both the prover and witnesses. Our PoL scheme outperforms the alternatives based on group digital signatures. We evaluate our schemes on Raspberry Pi hardware, and demonstrate that they achieve practical performance. In particular, the password generation and verification time are in the order of microseconds and milliseconds, respectively, while the computation time of proof generation is less than 1 second.

Keywords: Group Time-based One-Time Passwords· Proof of Location· Anonymity· Authentication· Security Model.

1 Introduction

Time-based One-Time Password (TOTP) is widely used in many two-factor authentication systems, for example, Google Authenticator [16] and Duo [9]. TOTP allows a prover to generate a time-dependent password that remains valid for a pre-defined time duration. A verifier checks the authenticity of the password by using the current time and some other information. There are two types of TOTP: *symmetric* one based on a shared secret key between the prover and verifier [32], and *asymmetric* one that can be verified publicly [30]. The first type uses the shared secret key and message authentication code to prove the authenticity of a generated password. The second type relies on hash chains, and it is one important building block of this work. An asymmetric TOTP prover randomly generates a hash chain head (pw_0) and keeps hashing the value to construct a hash chain. Namely, given a hash function H, the i-th node pw_i is computed as $pw_i := \mathsf{H}(pw_{i-1})$. The last node (tail) of the chain will be shared with verifiers for verification. To authenticate itself, the prover sends the verifier the chain's nodes in reverse order from tail to head. The verifier can quickly verify every received node but cannot forge an unseen node due to the one-wayness of the underlying hash function. Furthermore, any attackers who compromise the public key of an asymmetric TOTP cannot infer any unused passwords (unlike the first type).

TOTP is designed for the prover to prove its identity to the verifier. As such, it does not provide privacy, i.e., the verifier has to know the identity of the prover beforehand, in order to check the validity of every TOTP. In this paper, our goal is to extend TOTP to the group setting in order to achieve a meaningful notion of privacy. To this end, we propose Group Time-based One-Time Password (GTOTP) scheme, in which a

* This is the full version of the paper presented at ACSAC'21 [50].

** Jianting Ning is the corresponding author, and he is also with the State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093).

prover belonging to a group can generate one-time passwords to convince a verifier of its membership to the group without revealing its identity. GTOTP is useful in applications that need group-based, privacy-preserving authentication. One example is a service provider offering service (e.g., printing, consulting) only to the people within one community while the service user wants to preserve its privacy [4]. Another example is to combine GTOTP with any unilateral (server-only) authenticated handshake protocol [31,8] (supported by transport layer security protocol (TLS) [6,39]) following the password-based mutual authentication approach (over TLS) [12] to build privacy-preserving secure channels.

The main challenge in realizing GTOTP is efficiency. One way to construct GTOTP is to use group digital signatures (GDS) [4,28]. In particular, the group members share a group public key *gpk* for signature verification, while each member has its own secret signing key corresponding to *gpk*. During authentication, the member signs the current time-stamp, and the resulting signature is the one-time password for a fixed duration. However, this approach is expensive because it involves many public-key operations, which renders it impractical for resource-constrained devices.

We address this challenge with an efficient construction based on hash functions. Our key insight is to let each group member initialize multiple hash chain-based TOTP instances, with each instance used for only a short duration (like in a pseudonym scheme [25]). Because each TOTP instance yields a verify point (e.g., a tail node of the hash chain as in TOTP schemes [30] with public verifiability) for password verification, the result is a large number of states that the verifier must manage. In particular, the verifier needs to keep track of all verify points for all members, which does not scale to large group sizes or long validity duration for each TOTP instance. Our solution is to compress these states to a constant-sized group verification state (GVST). More specifically, we use a combination of Merkle and Bloom filter to store the states.

We then describe an application of GTOTP in building an efficient, privacy-preserving Proof of Location (PoL) scheme. A prover generates tamper-proof location proofs, with the help of some nearby witnesses, that convince a verifier that it was at the specific location at a given time. PoL has many real-world applications, such as tracking goods moving in supply chains, pandemic contact tracing, and monitoring home arrest. Our PoL construction combines GTOTP, a commitment scheme, and a privacy-preserving location proximity (PPLP) scheme. A prover first broadcasts the GTOTP passwords and a PPLP request generated based on its location to nearby parties. If a close-by party wants to be a witness, it replies with its own GTOTP passwords and a commitment of the PPLP response showing the close proximity. Since the location proof is not required to be verified immediately, we use the secret seeds of the corresponding GTOTP passwords as the keys for generating commitments. Namely, we leverage GTOTP as a tool to glue up other building blocks in our PoL scheme. During verification, the prover and the corresponding witnesses, who contributed to the proof, can open their secret seeds for commitment verification after the passwords become expired.

Contributions. We make the following contributions:

- We define group time-based one-time passwords (GTOTP), an extension of the traditional TOTP to the group setting. GTOTP provides membership authentication and privacy. Our security model adopts the game-based approach [2] to formulate anonymity and traceability (covering password unforgeability) based on a group with fixed members.
- We design an efficient and generic GTOTP construction that transforms an asymmetric TOTP scheme into a GTOTP. Our construction achieves constant memory cost at the verifier.
- To show the application of GTOTP, we construct an efficient, privacy-preserving PoL scheme using GTOTP. We formally define a security model for PoL that can be used to analyze the security properties of a PoL construction.
- We evaluate the performance of GTOTP and the PoL scheme on a Raspberry Pi. The results show that the cost (latency) of password generation is 4.12 microseconds on average, and of password verification is on the order of milliseconds. The proof generation time for the PoL scheme is less than 1 second when the number of witnesses is smaller than 10. In summary, both GTOTP and PoL schemes are practical.

Organization. Section 2 reviews the related works. Section 3 describes necessary preliminaries. Section 4 introduces the security model and construction of GTOTP. Section 5 presents the PoL construction. The evaluation results are discussed in Section 6. Section 7 concludes the paper.

2 Related Work

Time-based One-Time Password. A TOTP scheme can be constructed from a shared secret between the prover and verifier. The first asymmetric TOTP scheme, one that does not require a shared secret, is proposed by Lamport [32] based on one-way functions. In the Lamport scheme, the one-time passwords are organized in a chain in which the i-th password is generated by applying a one-way function to the (i-1)-th password. Jin et al. [26] provide the formal proof for the Lamport scheme in the standard model, and use TOTP to build a new primitive called proof of aliveness. Kogen et al. [30] proposed T/Key scheme that extends the Lamport

and S/Key scheme [20] by integrating time constraints to the passwords. These TOTP schemes are designed for authenticating one single prover to the verifier, and therefore do not have the privacy property of our group based scheme.

Proof of Location. Waters and Felten [48] proposed the first proof of location scheme in a centralized setting, in which a location manager is trusted to sign the proof based on measurement of the round-trip communication with the prover. The provider identity is encrypted with the verifier's public key, therefore is hidden from the location manager. Graham and Gray [17] proposed a scheme called SLVPGP that removes the need for the location manager. The proof in SLVPGP is based on distance related to a number of devices specified by the verifier. The scheme uses public keys of the prover, therefore it does not provide privacy.

In the decentralized settings, Zhu and Cao [51] proposed APPLAUS, in which either provers or users periodically change their pseudonyms and exchange signed messages and location proofs with each other. However, location privacy in this scheme depends on user behaviors. In other words, it assumes a user-centric location privacy model and may suffer non-negligible location privacy loss. King [29] proposed *FOAM* that relies on decentralized and trusted *zone anchor beacons* with synchronized clocks. The beacons determine the location of a user via triangulation, and store the signed location on a blockchain as the proof. The scheme, however, does not address privacy of the users. Wu et al. [49] proposed a blockchain-based zero-knowledge proof of location scheme. The user obtains a location certificate from a number of location beacons, as in FOAM, and generates a zero-knowledge proof of its with the help of the beacons. Dupin et al. [10] proposed another privacy-preserving scheme using group signatures for identity privacy, combining with secure multi-party computation for location privacy. It achieves strong privacy guarantees, but suffers high performance overhead due to the expensive cryptographic primitives.

The existing decentralized proof of location schemes use digital signatures, which may render them impractical for running frequently on low-power devices. Furthermore, they lack formal security analysis. Our PoL based on GTOTP is efficient because it does not use digital signatures, and it is provably secure. We compare our scheme against other state-of-the-arts in Section 6.3.

3 Preliminaries

We denote the security parameter by κ, an empty string by $\emptyset$, and the set of integers between 1 and n by $[n] = \{1, \ldots, n\} \subset \mathbb{N}$. We denote with $x \xleftarrow{\$} X$ the operation of sampling x uniformly at random from X. If X is a probabilistic algorithm, $x \xleftarrow{\$} X$ means that x is the output of running X with fresh random coins. Let $\|$ be the string concatenation operation. We represent a location L_A of a user A as the two-dimensional coordinates (x_A, y_B). Other notations can be found in Appendix A. In the following, we describe the syntax of the main cryptographic primitives used in our constructions. The security notions of these primitives are review in Appendix B.

Time-based One-time Passwords. An asymmetric TOTP scheme consists of 4 algorithms $(\mathsf{Setup}, \mathsf{PInit}, \mathsf{PGen}, \mathsf{Verify})$. $\mathsf{Setup}(1^\kappa, T_s, T_e, \Delta_s)$ takes as input the security parameter 1^κ, the start and end times T_s and T_e, and the password generation interval Δ_s, and outputs the password number $\mathsf{pms} = N = (T_e - T_s)/\Delta_s$. $\mathsf{PInit}(sd)$ takes as input a secret seed $sd \in \mathcal{K}_{\mathsf{TOTP}}$, and outputs the initial verify point vp, where $\mathcal{K}_{\mathsf{TOTP}}$ is the key space for the input secret seed. $\mathsf{PGen}(sd, T)$ takes as input the secret seed sd and a time slot T, and outputs a one-time password $pw \in \mathcal{PW}_{\mathsf{TOTP}}$ for T, where $\mathcal{PW}_{\mathsf{TOTP}}$ is a password space. $\mathsf{Verify}(vp, pw, T)$ takes as input the verify-point vp, a password pw, and time slot T, and outputs 1 if the password is accepted and 0 otherwise.[6] For a secure TOTP scheme, the adversary cannot forge a valid password for a future time.

Pseudo-random Function Family. A pseudo-random function (PRF) family consists of two algorithm algorithms $(\mathsf{Setup}, \mathsf{Eval})$. $\mathsf{Setup}(1^\kappa)$ takes as input the security parameter 1^κ, and outputs random secret key $k \xleftarrow{\$} \mathcal{K}_{\mathsf{PRF}}$ and system parameters $\mathsf{pms}_{\mathsf{PRF}}$, where $\mathcal{K}_{\mathsf{PRF}}$ is the key space of PRF. $\mathsf{Eval}(k, x)$ takes a input the secret key k and a message $x \in \mathcal{M}_{\mathsf{PRF}}$, and output the evaluation result $r \in \mathcal{R}_{\mathsf{PRF}}$, where $\mathcal{M}_{\mathsf{PRF}}$ and $\mathcal{R}_{\mathsf{PRF}}$ are the message space and the range space of PRF, respectively. For a specific PRF function family F, we may write $\mathsf{F}(k, x)$ to represent $\mathsf{F.Eval}(k, x)$ for short.

Bloom Filter. A Bloom filter BF is a probabilistic data structure that allows efficient membership testing without false negatives. BF consists of three algorithms $(\mathsf{Init}, \mathsf{Insert}, \mathsf{Check})$. $\mathsf{Init}(\epsilon, N)$ takes as input ϵ representing the rate of false positive, and the number of elements N, and initializes a BF which is a bit array of length $1.44\epsilon \cdot N$. $\mathsf{Insert}(m)$ inserts the element m into BF. $\mathsf{Check}(m)$ returns 1 if an element m is in the BF, and returns 0 otherwise.

[6] Note that we define a stateless verification other than the stateful one in prior work [30], but the security of stateful TOTP (with a stateful verification algorithm) implies the security of stateless TOTP in our definition. All passwords in a stateful TOTP can be verified by the initial verification state (IVS). This fact still holds even if the verification state is updated later.

Privacy-preserving Location Proximity Schemes. A PPLP protocol runs between a checker P and a responder W to determine if the two entities are in close proximity. It consists of five algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{LPInit}, \mathsf{LPResp}, \mathsf{LPCheck})$. $\mathsf{Setup}(1^\kappa)$ takes as input the security parameter 1^κ, and outputs the system parameters pms which contains a threshold θ. $\mathsf{KGen}(rk)$ takes as input a secret seed $rk \in \mathcal{K}_{\mathsf{PPLP}}$, and outputs a key pair for the checker P, where $\mathcal{K}_{\mathsf{PPLP}}$ is the range of the secret seed. $\mathsf{LPInit}(\mathsf{pk}, \mathrm{L}_\mathsf{P}, \theta)$ takes as input the public key pk, a location $\mathrm{L}_\mathsf{P} = (x_\mathsf{P}, y_\mathsf{P})$ of a checker P and a distance threshold θ for proximity test, and outputs a proximity challenge C_P. $\mathsf{LPResp}(\mathsf{pk}, C_\mathsf{P}, \mathrm{L}_\mathsf{W})$ takes as input the public key pk and a challenge C_P from a checker P, and the location of the responder W. It outputs a proximity response C_W. $\mathsf{LPCheck}(\mathsf{sk}, C_\mathsf{W})$ takes as input the secret key of a checker P and the response C_W, and outputs 1 if the checker and the responder are in close proximity, and 0 otherwise. Here we require a PPLP scheme to be secure against semi-honest adversaries who are interested in the locations of uncorrupted participants.

Authenticated Symmetric Encryption. Let $\mathcal{K}_{\mathsf{ASE}}$ be the key space, $\mathcal{M}_{\mathsf{ASE}}$ be the message space, and $\mathcal{C}_{\mathsf{ASE}}$ be the ciphertext space. An authenticated symmetric encryption scheme ASE consists of three algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$. $\mathsf{Setup}(1^\kappa)$ takes as input 1^κ, and outputs the parameter pms and a random key $k \xleftarrow{\$} \mathcal{K}_{\mathsf{ASE}}$. $\mathsf{Enc}(k, m)$ takes as input an encryption key k, and a message $m \in \mathcal{M}_{\mathsf{ASE}}$, and outputs a ciphertext $C \in \mathcal{C}_{\mathsf{ASE}}$. We say an ASE is *randomized* if Enc is a randomized algorithm. $\mathsf{Dec}(k, C)$ takes as input an encryption key k, and a ciphertext $C \in \mathcal{C}_{\mathsf{ASE}}$, and outputs a message $m \in \mathcal{M}_{\mathsf{ASE}}$. We require that the ASE scheme can resist adaptive chosen ciphertext attacks.

Collision-resistant Hash Functions. Let $\mathcal{K}_{\mathsf{CRHF}}$ be the key space, $\mathcal{M}_{\mathsf{CRHF}}$ be the message space, and $\mathcal{Y}_{\mathsf{CRHF}}$ be the hash value space. A collision-resistant hash function CRHF consists of two algorithms $(\mathsf{Setup}, \mathsf{Eval})$. $\mathsf{Setup}(1^\kappa)$ takes as input 1^κ, initializes the CRHF, and outputs the parameter pms and a random key $hk \xleftarrow{\$} \mathcal{K}_{\mathsf{CRHF}}$. $\mathsf{Eval}(hk, m)$ takes as input a random key hk and a message $m \in \mathcal{M}_{\mathsf{CRHF}}$, and outputs a hash value $y \in \mathcal{Y}_{\mathsf{CRHF}}$. For a specific collision-resistant hash function H, we write $\mathsf{H}(m)$ to represent $\mathsf{H}.\mathsf{Eval}(hk_{\mathsf{CRHF}}, m)$ when hk_{CRHF} is clear from the context.

Merkle Tree. A Merkle tree [34] allows efficient and secure verification of a large data set. Let H_1 be a collision resistant hash function. The Merkle tree consists of three algorithms $(\mathsf{MT.Build}, \mathsf{MT.GetProof}, \mathsf{MT.Verify})$. $\mathsf{MT.Build}(\{\mathrm{Lf}_i\}_{i\in[\ell]})$ takes as input ℓ data items and build a Merkle tree MTr on top of them. The leaf nodes are the data items, and the value of a non-leaf node is $\mathsf{H}_1(\text{node.LeftChild}||\text{node.RightChild})$. $\mathsf{MT.GetProof}(\mathrm{MTr}, \mathrm{Lf}_i)$ takes as input a Merkle tree MTr and a leaf Lf_i, and outputs a proof showing that Lf_i is included in the tree. Specifically, the proof includes the siblings of every node on the path from Lf_i to the tree's root MTr.Rt. $\mathsf{MT.Verify}(\mathrm{MTr.Rt}, \mathrm{Lf}_i, \mathrm{Pf}_{\mathrm{Lf}_i})$ takes as input the root MTr.Rt, a leaf node Lf_i, and the corresponding proof $\mathrm{Pf}_{\mathrm{Lf}_i}$. It computes a root Rt' based on Lf_i and proof $\mathrm{Pf}_{\mathrm{Lf}_i}$, and returns 1 if $\mathrm{Rt}' = \mathrm{MTr.Rt}$ and 0 otherwise. For a secure Merkle tree scheme, the adversary must not forge the Merkle proof for a leaf node which does not belong to the Merkle tree.

4 Group Time-based One-time Passwords

In this section, we present a new primitive, called Group Time-based One-time Password (GTOTP), which extends asymmetric TOTP to the group setting. With GTOTP, the prover belonging to a group can convince the verifier of its group membership without revealing its identity. A GTOTP scheme has three main properties. First, only members of the group can generate valid time-based one-time passwords for authentication. Second, any party that has the password can verify whether it is a valid password generated by a member of the corresponding group, without learning the identity of the password owner. Finally, the group manager can recover the identity of a password owner if necessary.

Applications that benefit from GTOP are ones that need group-based, privacy-preserving authentication. In particular, GTOTP can be used in some applications that otherwise depend on group signatures. As an example, GTOTP can enable privacy-preserving secure channels, by combining with any unilateral (server-only) authenticated handshake protocol [31,8] (supported by transport layer security protocol (TLS) [6,39]) following the password-based mutual authentication approach (over TLS) [12]. Based on such a channel, a service provider can offer services such as printing or legal advice to members of a certain group while protecting the group members' privacy [4]. In the next section, we will particularly show another application example of GTOTP in constructing a privacy-preserving proof of location scheme.

In this section, we define the security model and present an efficient construction for GTOTP. Here, we assume a group with fixed members.

4.1 Security Model

We consider a group with U members. There is a registration authority (RA) that handles parameter initialization and member enrollment. We assume that the RA is trusted by all participants of the system and works as a certificate authority to digitally sign information (e.g., group verification state) of members.

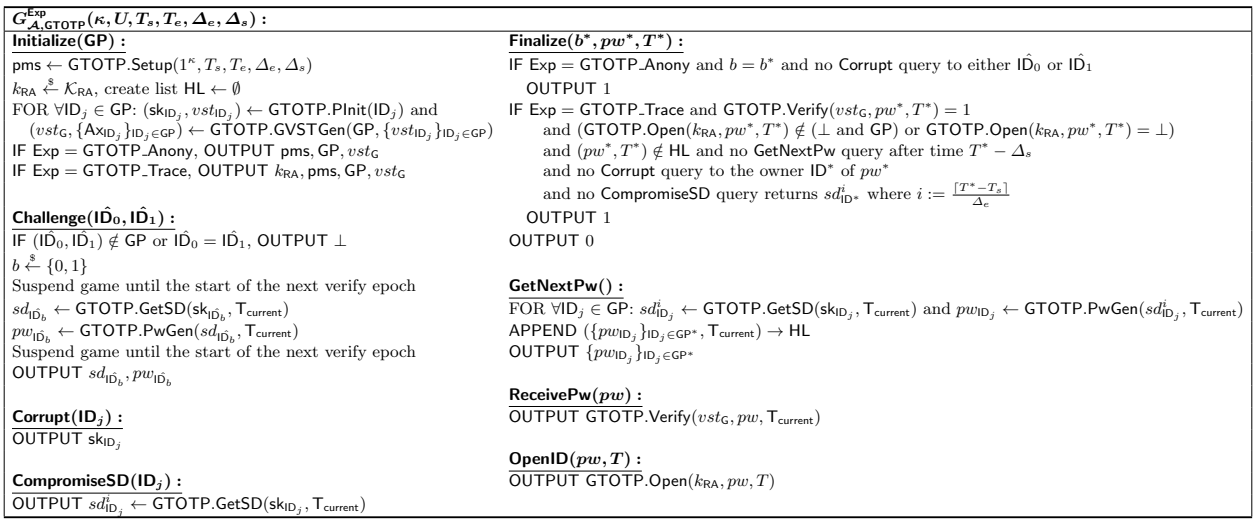

Fig. 1: Procedures Used to Define the Security of a GTOTP Scheme.

Each GTOTP instance has a life-span of Δ_t. Each group member generates one password per Δ_s. Since many passwords may correspond to the same verify point, so we assume that each verify point (if any) has a validity period of Δ_e, called the *verify epoch*.

Syntax. A GTOTP scheme involving $U \in \mathbb{N}$ parties (or provers) $\mathsf{GP} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_U)$, one verifier V, and one registration authority RA, consists of seven algorithms described below.

- $(\mathsf{pms}, k_{\mathsf{RA}}) \leftarrow \mathsf{Setup}(1^\kappa, T_s, T_e, \Delta_e, \Delta_s)$: This setup algorithm is run by the RA. It takes as input the security parameter 1^κ, the start and the end time of the protocol instance T_s and T_e, the verify epoch Δ_e, and the password generation interval Δ_s. It outputs the system parameters pms and a secret key $k_{\mathsf{RA}} \xleftarrow{\$} \mathcal{K}_{\mathsf{RA}}$ for RA, where $\mathcal{K}_{\mathsf{RA}}$ is a secret key space.
- $(\mathsf{sk}_{\mathsf{ID}_j}, vst_{\mathsf{ID}_j}) \leftarrow \mathsf{PInit}(\mathsf{ID}_j)$: This is initialization algorithm run by the group members. It takes as input the member identity ID_j, and outputs the secret key $\mathsf{sk}_{\mathsf{ID}_j} \xleftarrow{\$} \mathcal{K}_{\mathsf{GTOTP}}$ and the initial verification state vst_{ID_j} for ID_j.
- $(vst_{\mathsf{G}}, \{\mathsf{Ax}_{\mathsf{ID}_j}\}_{j\in[U]}) \leftarrow \mathsf{GVSTGen}(\mathsf{GP}, \{vst_{\mathsf{ID}_j}\}_{j\in[U]})$: This initialization algorithm is run by the RA. It takes as input the identity and verification state of all the group members, and outputs the initial group verification state vst_{G} and auxiliary outputs for the group members $\{\mathsf{Ax}_{\mathsf{ID}_j}\}_{j\in[U]}$.
- $sd^i_{\mathsf{ID}_j} \leftarrow \mathsf{GetSD}(\mathsf{sk}_{\mathsf{ID}_j}, T)$: This seed generation algorithm is run by the group member. It takes as input the secret key $\mathsf{sk}_{\mathsf{ID}_j} \in \mathcal{S}_{\mathsf{GTOTP}}$ and a time slot T, and outputs the secret seed $sd^i_{\mathsf{ID}_j}$ for generating the password at T, where $\mathcal{S}_{\mathsf{GTOTP}}$ is a secret seed space.
- $pw^{i,z}_{\mathsf{ID}_j} \leftarrow \mathsf{PwGen}(sd^i_{\mathsf{ID}_j}, T)$: The password generation algorithm is run by a group member. It takes as input the secret seed $sd^i_{\mathsf{ID}_j}$ for the time slot T, and outputs one-time password $pw^{i,z}_{\mathsf{ID}_j}$, where z is an index of the password in the i-th verify epoch.
- $\{0,1\} \leftarrow \mathsf{Verify}(vst_{\mathsf{G}}, pw^{i,z}_{\mathsf{ID}_j}, T)$: This password verification algorithm is run by the verifier. It takes as input the state vst_{G}, the password $pw^{i,z}_{\mathsf{ID}_j}$, and time slot T, and outputs 1 if $pw^{i,z}_{\mathsf{ID}_j}$ is accepted, and 0 otherwise.
- $\mathsf{ID}_j \leftarrow \mathsf{Open}(k_{\mathsf{RA}}, pw^{i,z}_{\mathsf{ID}_j}, T)$: This identity extraction algorithm is run by the *RA*. It takes as input the key k_{RA}, the password $pw^{i,z}_{\mathsf{ID}_j}$, and the time slot T, and outputs ID_j if successful, and ⊥ otherwise.

Given $(\mathsf{pms}, k_{\mathsf{RA}}) \leftarrow \mathsf{Setup}(1^\kappa, T_s, T_e, \Delta_e, \Delta_s)$, $\{(\mathsf{sk}_{\mathsf{ID}_j}, vst_{\mathsf{ID}_j}) \leftarrow \mathsf{PInit}(\mathsf{ID}_j)\}_{\mathsf{ID}_j \in \mathsf{GP}}$ and GVST $(vst_{\mathsf{G}}, \{\mathsf{Ax}_{\mathsf{ID}_j}\}_{j\in[U]}) \leftarrow \mathsf{GVSTGen}(\mathsf{GP}, \{vst_{\mathsf{ID}_j}\}_{j\in[U]})$, the GTOTP scheme is correct if $\mathsf{Verify}(vst_{\mathsf{G}}, \mathsf{PwGen}(sd^i_{\mathsf{ID}_j}, T), T)$ outputs 1 for all $\mathsf{ID}_j \in \mathsf{GP}$ and time slot $T \in [T_s, T_e]$.

Threat Model. We consider the common threats that widely exist in password-based authentication schemes (e.g., [30,22,47]) and group signatures (e.g., [4,11]). The provers are honest, but an attacker might corrupt the secret key of a prover. The RA is a trustful (and non-colluding) third party that will not be corrupted or controlled by any attackers. The communication between a prover and the RA is secure. The verifier can be malicious, who may be curious about the identity of a prover. There also exist attackers who can control the communication among parties, and therefore can intercept, inject, and tamper with the communication. The attackers may also want to impersonate an honest prover without learning its secret key. Moreover, the attackers may try to avoid being identified when she behaves as a malicious prover. We assume that each party in the system also has an internal clock that is synchronized with other parties in the system.

Security Definition. Following the game-based approach in [2], we define two security games for GTOTP to formulate security properties regarding *anonymity* (GTOTP_Anony) and *traceability* (GTOTP_Trace), respec-

tively. That is, we let $\mathsf{Exp} \in \{\mathsf{GTOTP\_Anony}, \mathsf{GTOTP\_Trace}\}$ be a variable to indicate one of the games, where these games will share procedures. We also discuss the achievable privacy of GTOTP in Appendix D.

We present the relevant procedures of the security games in Figure 1. The games can be started by calling the Initialize procedure and ended by calling the Finalize procedure. The adversary can specify a set of unique identities of her own choice for running the game when calling Initialize. During the game, the adversary $\mathcal{A}$ can sequentially call other procedures. The games are defined in a multiparty setting. We model the corruption of parties via a Corrupt procedure, the compromise of secret seed via CompromiseSD procedure, and the revelation of identity via OpenID procedure. Besides, the adversary can get passwords of group members via the GetNextPw procedure, and test the validity of a password via the ReceivePw procedure.

The traceability property of GTOTP is adapted from the unforgeability of TOTP [30] and the traceability of GDS [1]. We note that this property covers the password unforgeability property that is similar to the formalization of traceability of GDS [1]. More specifically, the adversary cannot create a password associated with an uncompromised secret seed of an uncorrupted member, such that the password is valid but cannot be opened to the corresponding member.

We model anonymity using the indistinguishability based formalization widely used in pseudonym schemes [14,44]. Specifically, we customize a $\mathsf{Challenge}(\hat{\mathsf{ID}}_0, \hat{\mathsf{ID}}_1)$ procedure for the GTOTP_Anony game. That is, the adversary can ask the challenge query at any time with two honest identities based on which the challenger would flip a random bit b, and return an unused secret seed and a password of one of the challenged parties. The returned secret seed will be called as *challenge secret seed*, and the verify epoch for the verify-point generated by the challenge secret seed will be called as *challenge verify epoch*. To get an unused challenge secret seed, the challenger would suspend the game until the start of the next verify epoch, so that this would invalidate the current verify-point. Since the adversary can ask GetNextPw() to get the passwords of challenged parties, the challenger will suspend the game again until the end of the challenge verify epoch to prevent the adversary from learning the passwords of the $\hat{\mathsf{ID}}_0$ and $\hat{\mathsf{ID}}_1$ in the challenge verify epoch (that would allow the adversary to trivially win the game). Note that the adversary can ask a $\mathsf{Corrupt}(\mathsf{ID}_j)$ query to compromise the secret key of a party ID_j and learn all passwords of this party, so we require all challenge parties to be uncorrupted (namely, there is no $\mathsf{Corrupt}(\cdot)$ query to them). Moreover, we also allow the adversary to compromise the secret seeds of uncorrupted parties via the $\mathsf{CompromiseSD}(\cdot)$ procedure. Note that our suspensions in the Challenge procedure ensure that the challenge secret seeds will not be compromised in the GTOTP_Anony game. Also, $\mathcal{A}$ cannot ask the OpenID query to obtain the identities of challenge parties. Furthermore, in the GTOTP_Trace game $\mathcal{A}$ should not compromise the secret seed that is supposed to generate the forged password pw^* of the adversary.

The goal of the adversary in the GTOTP_Anony game is to distinguish the owner of the challenge secret seed from $(\hat{\mathsf{ID}}_0, \hat{\mathsf{ID}}_1)$. Note that with the given challenge secret seed and password, the adversary can compute all other passwords of the corresponding party by herself in the challenge verify epoch. In the GTOTP_Trace game, the adversary attempts to produce a forgery (pw^*, T^*), and we say it wins the game (i.e., the experiment returns 1), if (pw^*, T^*) is a valid password-time pair and the opening algorithm returns $\perp$, or some valid identity ID_j such that $\mathsf{ID}_j \notin \mathsf{GP}$.

Definition 1. *We say that a GTOTP scheme* GTOTP *is secure if for any PPT adversary $\mathcal{A}$, the advantages* $\mathsf{Adv}^{\mathsf{GTOTP\_Anony}}_{\mathcal{A},\mathsf{GTOTP}}(\mathsf{pms}) := \left|\Pr\left[G^{\mathsf{GTOTP\_Anony}}_{\mathcal{A},\mathsf{GTOTP}}(\mathsf{pms}) = 1\right] - 1/2\right|$ *and* $\mathsf{Adv}^{\mathsf{GTOTP\_Trace}}_{\mathcal{A},\mathsf{GTOTP}}(\mathsf{pms}) := \Pr\left[G^{\mathsf{GTOTP\_Trace}}_{\mathcal{A},\mathsf{GTOTP}}(\mathsf{pms}) = 1\right]$ *of $\mathcal{A}$ in the corresponding games are negligible under the parameters* $\mathsf{pms} = (\kappa, U, T_s, T_e, \Delta_e, \Delta_s)$.

4.2 An Efficient GTOTP Scheme

Here we introduce an efficient GTOTP protocol GTOTP-MT.

Building Blocks. We mainly use an asymmetric TOTP scheme TOTP, a Merkle tree scheme MT, an unpredictable permutation scheme [37] $\pi(k_p, \cdot)$ with a random key $k_p \xleftarrow{\$} \{0,1\}^\kappa$, a Bloom filter BF, a PRF family F, a randomized authenticated symmetric encryption scheme ASE (i.e., it consists of a randomized encryption algorithm), and a collision resistant-hash function H_1. The permutation scheme $\pi(k_p, \cdot)$ takes as input a set of elements and outputs a permuted set. For privacy, we require the TOTP scheme to not use any identity relevant information in the whole protocol execution. For example, the one-way function based TOTP in [26] meets such a requirement. To use T/Key [30], one can assign each TOTP instance of a party with a random salt. We also need a TOTP that a verify point $vp^i_{\mathsf{ID}_j}$ can be obtained from any passwords being verified by it. All chain-based TOTP (e.g., [30,26]) can achieve this requirement.

A Naive Solution. One solution for realizing GTOTP is to use multiple instances of a TOTP protocol, for example [26], to generate passwords. Each instance is used only for a short period Δ_e to avoid linkability between verify points. The number E of the TOTP instances is determined by the life-span $\Delta_t = T_e - T_s$ and Δ_e, that is $E = \Delta_t/\Delta_e$. The seed $sd^i_{\mathsf{ID}_j}$ for the key generation can then be created by applying a PRF to k_{ID_j}. To achieve anonymity, we store verify points of multiple members into a single Bloom filter BF, which serves as

the group verification state (GVST). The verifier checks the verify point's membership against BF, and whether the password is based on the verify point.

The limitation of this solution is space overhead of the verification state, which is $O(U \cdot E)$. For example, assume a reasonable password generation frequency of $\Delta_s = 5s$, five-day usage $\Delta_t = 432000s$, a Bloom filter's false positive rate of $\epsilon = 40$, and the group size of $U = 100$, the size of vst_{G} becomes $1.44 \cdot 40 \cdot 100 \cdot \frac{\Delta_t}{\Delta_s} \cdot \frac{1}{1024 \cdot 1024 \cdot 8} \approx$ 59MB. This cost grows larger with more members (larger U) or longer usage period (larger Δ_t). Thus, the main challenge in constructing a GTOTP scheme is to reduce this cost for a long-time period and larger group size. We address this by using Merkle tree to achieve constant-sized verification states.

Overview. We observe that a Merkle tree supports the same functionality of a Bloom filter, that is, to prove the authenticity of the verify point of a group member. In other words, we can build a Merkle tree on the verify points generated by the naive scheme above. This introduces additional cost at the member to store and send Merkle proofs for the passwords. We choose Merkle tree for our construction because it has short proofs.

Our first design is to build one single tree for all verify points, such that the group verification state is the tree root. To achieve anonymity, we first shuffle the verify-points from group members using π, in order to remove the relationship among them before building the Merkle tree. By permuting the leaf nodes, a non-leaf nodes may be computed based on the verify points from different members. However, this permutation destroys the time order implied by the indices of the verify points. To overcome this, we use the collision-resistant hash function H_1 to explicitly bind each verify point with the index of the verify epoch. We stress that the above operations will be done by a trustful RA which is not controlled by the attackers, so an attacker cannot distinguish the owner of the verify points not generated by her.

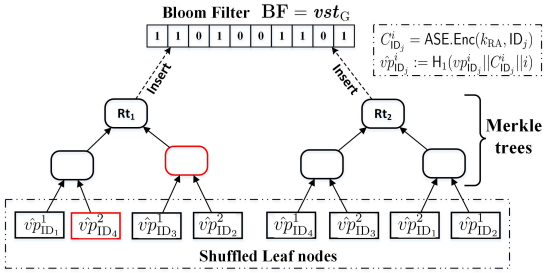

Fig. 2: Overview of GTOTP-MT.GVSTGen for $U = 4$, $E = 2$, and $\phi = 2$. Red nodes (shown as an example) in the first Merkle tree is the Merkle proof of leaf node $\hat{vp}^1_{\mathsf{ID}_1}$.

We note that both the storage cost at each member, and the verification cost at the verifier, are linear in the height of the tree. Our final design reduces these costs by partitioning the leaf nodes into disjoint sets and build a Merkle tree on each set. These trees have smaller heights than the original one, and the size of the Merkle proofs of a group member becomes $O(E \cdot \log^{\frac{U \cdot E}{\phi}})$ which is practical even when the Δ_t is large (e.g., half a year). Finally, we store the Merkle roots in a Bloom filter. Figure 2 illustrates the GVST generation process.

To achieve traceability, the RA encrypts each identity E times with ASE, binding each ciphertext with the corresponding verify point using H_1. The binding combined with the security of the Merkle tree prevents the encrypted identity from being forged. Finally, to verify a password for a time slot, the verifier first recovers the corresponding verify point, checks the Merkle tree proof, and then verifies that the tree root is in the Bloom filter.

Detailed Construction. Our construction GTOTP-MT, realizes the seven GTOTP algorithms as follows.

- $\mathsf{Setup}(1^\kappa, T_s, T_e, \Delta_e, \Delta_s)$: RA runs this algorithm to sample a secret key k_{RA}, calculates the number of passwords in a chain $N := (T_e - T_s)/\Delta_s$ and the number of TOTP protocol instances $E := (T_e - T_s)/\Delta_e$, and sets the start time $T_s := \mathsf{T}_{\mathsf{current}}$ to be current system time. Moreover, It also runs the setup algorithm $hk \leftarrow \mathsf{H}_1.\mathsf{Setup}(1^\kappa)$ to initialize a hash key hk, and generates a parameter ϕ denoting the number of sub-sets of verify points. A random key $k_p \xleftarrow{\$} \{0,1\}^\kappa$ for permutation is sampled. The parameters $\mathsf{pms} = (hk, k_p, N, E, T_s, T_e, \phi)$ are returned.
- $\mathsf{PInit}(\mathsf{ID}_j)$: The group member ID_j first runs $(\mathsf{pms}, k_{\mathsf{ID}_j}) \leftarrow \mathsf{F.Setup}(1^\kappa)$ to sample a random key $k_{\mathsf{ID}_j} \xleftarrow{\$} \mathcal{K}_{\mathsf{PRF}}$ as its secret key, where $\mathcal{K}_{\mathsf{PRF}}$ is a secret key space of PRF. For $i \in [E]$, ID_j first initializes the i-th protocol instance $\mathsf{pms}^i := \mathsf{TOTP.Setup}(1^\kappa, T_i, T_i + \Delta_e, \Delta_s)$, where $T_i = T_{i-1} + \Delta_e$ and $T_0 := T_s$. Then it computes the i-th secret seed $sd^i_{\mathsf{ID}_j} := \mathsf{F}(k_{\mathsf{ID}_j}, \mathsf{ID}_j||i)$, and the verify point $vp^i_{\mathsf{ID}_j} := \mathsf{TOTP.PInit}(sd^i_{\mathsf{ID}})$. This algorithm returns secret key $\mathsf{sk}_{\mathsf{ID}_j} = k_{\mathsf{ID}_j}$ and verification state $vst_{\mathsf{ID}_j} = \{vp^i_{\mathsf{ID}_j}\}_{i \in [E]}$.
- $\mathsf{GVSTGen}(\mathsf{GP}, \{vst_{\mathsf{ID}_j}\}_{j \in [U]})$: Upon receiving all verification states from the group members via a MASC, for $j \in [U]$ and $i \in [E]$, RA computes $C^i_{\mathsf{ID}_j} = \mathsf{ASE.Enc}(k_{\mathsf{RA}}, \mathsf{ID}_j)$, and updates the verify point $\hat{vp}^i_{\mathsf{ID}_j} := \mathsf{H}_1(vp^i_{\mathsf{ID}_j}||C^i_{\mathsf{ID}_j}||i)$ to bind the index and identity-ciphertext $C^i_{\mathsf{ID}_j}$ to the verify point. As $\mathsf{ASE.Enc}$ is randomized, we have that $C^i_{\mathsf{ID}_j} \neq C^{i+1}_{\mathsf{ID}_j}$. Let $V = \{vst_{\mathsf{ID}_j}\}_{j \in [U]} = \{\hat{vp}^i_{\mathsf{ID}_j}\}_{i \in [E], j \in [U]}$. RA shuffles V to generate a random

set $V' := \pi(k_p, V)$. Next, RA divides leaf nodes in V' into ϕ sub-sets $\{V'_1, V'_2, \ldots, V'_\phi\}$. Then, RA generates ϕ Merkle trees $\{\mathrm{MTr}_\iota := \mathsf{MT.Build}(V'_\iota)\}_{\iota\in[\phi]}$, and computes the Merkle proof of each leaf node as $\mathrm{Pf}_{\hat{vp}^i_{\mathsf{ID}_j}} := \mathsf{MT.GetProof}(\mathrm{MTr}_\iota, \hat{vp}^i_{\mathsf{ID}_j})$ for all $i \in [E]$ and $j \in [U]$, where MTr_ι is the Merkle tree containing the leaf node $\hat{vp}^i_{\mathsf{ID}_j}$. To this end, RA initiates an empty Bloom filter instance $\mathsf{BF} := \mathsf{BF.Init}(\epsilon, \phi)$, and inserts all Merkle tree roots $\{\mathrm{MTr}_\iota.\mathrm{Rt}\}_{\iota\in[\phi]}$ into a Bloom filter, i.e., $\mathsf{BF.Insert}(\mathrm{MTr}_\iota.\mathrm{Rt})$ for all $\iota \in [\phi]$. Eventually, RA sends back the corresponding Merkle proofs of the corresponding leaf nodes as auxiliary output of this algorithm $\mathsf{Ax}_{\mathsf{ID}_j} = \{\mathrm{Pf}_{\hat{vp}^i_{\mathsf{ID}_j}}\}_{i\in[E]}$ to each party ID_j via a MASC. Each party privately stores secret key $\mathsf{sk}_{\mathsf{ID}_j} := k_{\mathsf{ID}_j}$, random identity-ciphertexts $\{C^i_{\mathsf{ID}_j}\}_{i\in[E]}$, and Merkle proofs $\{\mathrm{Pf}_{\hat{vp}^i_{\mathsf{ID}_j}}\}_{i\in[E]}$. The initial group verification state is $vst_{\mathsf{GP}} := \mathsf{BF}$ which has a constant-size in $O(\phi)$.

- $\mathsf{PwGen}(\mathsf{sk}_{\mathsf{ID}_j}, \mathsf{T}_{\mathsf{current}})$: The party ID_j first computes the index i associated with $\mathsf{T}_{\mathsf{current}}$ as $i := \frac{\lceil T - T_s \rceil}{\Delta_e}$. Next, ID_j computes the secret seed $sd^i_{\mathsf{ID}_j} := \mathsf{F}(k_{\mathsf{ID}_j}, \mathsf{ID}_j||i)$ and gets the password $p\bar{w}^{i,z}_{\mathsf{ID}_j} := \mathsf{TOTP.PGen}(sd^i_{\mathsf{ID}_j}, \mathsf{T}_{\mathsf{current}})$, where z is password index in the i-th verify epoch, i.e., $z := \frac{\lceil \mathsf{T}_{\mathsf{current}} - T_s - i\cdot\Delta_e \rceil}{\Delta_s}$. The password returned by this algorithm is $pw^{i,z}_{\mathsf{ID}_j} := (p\bar{w}^{i,z}_{\mathsf{ID}_j}, C^i_{\mathsf{ID}_j}, \mathrm{Pf}_{\hat{vp}^i_{\mathsf{ID}_j}})$.
- $\mathsf{GetSD}(\mathsf{sk}_{\mathsf{ID}_j}, T)$: It calculates the index of sub-chain $i := \frac{\lceil T - T_s \rceil}{\Delta_e}$ and returns i-th seed $sd^i_{\mathsf{ID}_j} := \mathsf{F}(k_{\mathsf{ID}_j}, \mathsf{ID}_j||i)$.
- $\mathsf{Verify}(vst_{\mathsf{G}}, pw^{i,z}_{\mathsf{ID}_j}, T)$: The verifier first initializes the verification result $vr := 0$, and computes the verify point $vp^i_{\mathsf{ID}_j}$ based on $pw^{i,z}_{\mathsf{ID}_j}$ and T and updates it to be $\hat{vp}^i_{\mathsf{ID}_j} := \mathsf{H}_1(vp^i_{\mathsf{ID}_j}||C^i_{\mathsf{ID}_j}||i)$. Next, the verifier computes the corresponding root $\mathrm{MTr}_\iota.\mathrm{Rt}$ based on $\hat{vp}^i_{\mathsf{ID}_j}$ and the Merkle proof $\mathrm{Pf}_{\hat{vp}^i_{\mathsf{ID}_j}}$. Eventually, the verifier sets $vr := 1$ if and only if $\mathsf{BF.Check}(\mathrm{MTr}_\iota.\mathrm{Rt}) = 1$ and $\mathsf{TOTP.Verify}(vp^i_{\mathsf{ID}_j}, p\bar{w}^{i,z}_{\mathsf{ID}_j}, T) = 1$, and outputs vr.
- $\mathsf{Open}(k_{\mathsf{RA}}, pw^{i,z}_{\mathsf{ID}_j}, T)$: If $\mathsf{Verify}(vst_{\mathsf{G}}, pw^{i,z}_{\mathsf{ID}_j}, T) = 0$, then RA aborts. Otherwise, RA returns $\mathsf{ID}_j := \mathsf{ASE.Dec}(k_{\mathsf{RA}}, C^i_{\mathsf{ID}_j})$.

Security Analysis. Our construction achieves both anonymity and traceability, as stated by the two theorems below.

Theorem 1. *Suppose that the time-based one-time passwords scheme* TOTP*, the collision-resistant hash function* H_1*, the pseudo-random function family* F*, the Merkle tree scheme* MT *are secure,* π *is an unpredictable permutation function, the authenticated symmetric encryption* ASE *is randomized and secure, and the Bloom filter has a negligible false positive error* $2^{-\epsilon}$*. Then* GTOTP-MT *provides anonymity.*

The proof of Theorem 1 is included in Appendix E. Here, we sketch the high-level idea behind the proof. Since the secret seeds created by F are indistinguishable and the outputs of ASE.Enc and H_1 are random, the adversary cannot distinguish two verify points computed from these challenge secret seed and password. As the permutation is unpredictable, the adversary cannot learn the leaf nodes' owners (when they are not corrupted) from the Merkle proofs. Furthermore, since each Merkle tree root is mapped to m bit positions of the Bloom filter, and many roots may share the same bit positions, the adversary cannot recover the relationship of two inserted roots from the Bloom filter. Finally, because ASE is IND-CCA secure, the adversary cannot extract the member identity from the ciphertext included in the challenge password.

Theorem 2. *With the same assumptions in Theorem 1,* GTOTP-MT *provides traceability.*

The proof for Theorem 2 is included in Appendix F. We first exclude the collision among secret keys of parties based on the security of PRF. Next, we change the game to exclude the collision among the verify points due to the security of H_1. Then, we reduce the security to PRF again, so that we can replace the secret seeds of each TOTP instance with random values. Now, if the adversary can forge an unused password, then it must be able to break the security of either the TOTP scheme or the Merkle tree. Since the adversary cannot forge the Merkle proofs, it cannot bind a password to a maliciously chosen ciphertext.

5 Proof of Location

This section presents a novel application of GTOTP that enables an efficient, privacy-preserving proof of location (PoL) scheme. A PoL is a verifiable, tamper-proof statement attesting that a user is at a specific location at a specific time. A PoL scheme has many real-world applications, such as address verification, tracking and tracing of goods in supply chain systems, monitoring of health status, or humanitarian aid. For example, a bank customer can use PoL to securely prove the physical presence at a certain location to the bank as part of the verification of transactions. Due to the global COVID-19 pandemic, there is an increasing demand for contact tracing which is based on the close proximity of users. A PoL can help solve this problem. In this work, we focus on decentralized settings, that is, the participants of PoL can choose for themselves the source of locations, e.g., GPS or location beacons [29].

5.1 System Model

The PoL system consists of five entities: *prover*, *witness*, *verifier*, *Registration Authority (RA)*, and *Public Bulletin Board (PBB)*. The *prover* is a user device, e.g. a mobile phone or a smart vehicle, that generates a location proof for a specific location and time. The *witness* is a device near the prover that generates a proof ceritifying the proximity between itself and the prover. A witness could be a device like the prover, or a location beacon like a road side unit (RSU). The *verifier* is the entity that verifies the prover's location base on the proof. The RA handles enrollment of the prover, witness, and verifier. In particular, it ceritfies the other entities' keys, and reveals their identities if necessary.

The PBB is a secure storage to which entities can publish messages and read each other's messages. It provides *append-only* and *integrity* properties such that messages stored in PBB cannot be deleted or modified. We use PBB to realize anonymous data exchange and storage. In particular, the data owner and PBB establish a unilateral (PBB-only) authenticated secure channel [7,6,39] to protect *integrity* of the data while hiding the data owner's identity. One popular instantiation of PBB is Certificate Transparency [15], another is blockchains. We refer the reader to [5] for the formal model of PBB.

Syntax. We consider a PoL scheme with the following interactive sub-protocols.

- $(\mathsf{vk_G}, \{sk_{\mathsf{ID}_i}\}_{i\in[U]}, k_{\mathsf{RA}}) \leftarrow \mathit{Registration}(\mathsf{GP}, T_s, T_e, \Delta_e, \Delta_s)$. RA and a group of parties with identities $\mathsf{GP} = \{\mathsf{ID}_i\}_{i\in[U]}$ (which could be either prover or witness) can run this protocol together to generate a group verification key $\mathsf{vk_G}$ based on the public protocol parameters (including the start and end times T_s and T_e, length of each verify epoch Δ_e, and proof generation generation interval Δ_s). Meanwhile, each group member ID_i (for $i \in [U]$) will keep the generated secret key sk_{ID_i} privately. RA generates a secret key k_{RA} for realizing traceability.
- $\mathsf{LP_P} \leftarrow \mathit{Location\text{-}Proof\text{-}Gen}(sk_{\mathsf{P}}, \{sk_{\mathsf{W}_j}\}_{j\in[M]}, \mathsf{vk_G}, \mathrm{L_P}, \{\mathrm{L}_{\mathsf{W}_j}\}_{j\in[M]})$. A prover P and M witnesses $\{\mathsf{W}_j\}_{j\in[M]}$ run this protocol to generate the location proof $\mathsf{LP_P}$ for the prover's location $\mathrm{L_P}$, based on their secret keys $(sk_{\mathsf{P}}, \{sk_{\mathsf{W}_j}\}_{j\in[M]})$, witnesses' locations $\{\mathrm{L}_{\mathsf{W}_j}\}_{j\in[M]}$, current time slot $\mathsf{T_{current}}$, and verification key $\mathsf{vk_G}$, where M is the number of the witnesses and $(\mathsf{P}, \{\mathsf{W}_j\}_{j\in[M]}) \in \mathsf{GP}$.
- $vr \leftarrow \mathit{Verification}(\mathsf{vk_G}, \mathsf{LP_P}, \mathsf{sk_P}, \{\mathsf{sk}_{\mathsf{W}_j}\}_{j\in[M]}, \mathrm{L_P})$. The verifier V can check the validity of the location proof $\mathsf{LP_P}$ with the help of the proof *contributors* including the prover P and the corresponding witnesses $\{\mathsf{W}_j\}_{j\in[M]}$, in which the proof contributors may provide V necessary confidential information used to compute $\mathsf{LP_P}$. The verification result $vr \in \{0, 1\}$ is returned.

A PoL scheme has an additional non-interactive algorithm:

- $\mathsf{cID}_i \leftarrow \mathsf{Open}(k_{\mathsf{RA}}, \mathsf{LP_P}, i)$: This is an identity extraction algorithm (run by RA) that takes as input a location proof $\mathsf{LP_P}$, and outputs the identity cID_i of the i-th contributor of the proof or a failure symbol, where $\mathsf{cID}_i \in \mathsf{GP}$.

The correctness of PoL means that if honest entities with adjacent locations follows the algorithms, they will generate location proofs that pass the verification.

5.2 An Efficient Privacy-Preserving Proof of Location Scheme

In this section, we introduce an efficient PoL construction based on our proposed GTOTP.

Threat Model. Here we consider three most desirable security properties including *anonymity*, *traceability* and *location privacy* in our PoL scheme. The threats against *anonymity* and *traceability* are similar to these against GTOTP. Informally speaking, anonymity requires that the prover can attest its location to the verifier while preserving the anonymity of all other participants. A PoL scheme with traceability should prevent the adversaries from forging a valid location proof that involves either an invalid or a dishonest identity. Although we allow the attackers to corrupt participants, we assume the majority of the witnesses and the prover are uncorrupted while jointly generating a location proof. Moreover, location privacy is achieved if no adversaries can infer any information of the locations of the uncorrupted witnesses from the location proofs. In Appendix C, we define a security model to formulate those security properties.

Building Blocks. Our construction uses a GTOTP scheme GTOTP, a PRF family F, a cryptographic hash function $\mathsf{H}_2 : \{0, 1\}^* \to \mathcal{R}_h$, and a PPLP scheme PPLP. H_2 is used to realize a random oracle based commitment scheme (as in [45]). We assume that the entities establish either mutual authenticated or unilateral (server-only) authenticated secure channel depending on their roles.

Overview. One common approach for a prover to attest its presence at a certain location is to collect proofs from nearby witnesses who can testify the prover's location. However, there are three problems: (i) guaranteeing the anonymity of entities; (ii) protecting the witnesses' location privacy; (iii) binding the location proofs with the entities.

To solve these problems, we first leverage our new GTOTP scheme to achieve anonymity with efficiency. We build a GTOTP group with members being either the witnesses or prover. We assume that the location proof requests and responses are transmitted via a short-range communication (e.g., Bluetooth), such that only the nearby witnesses can receive proof requests from the prover. However, the prover still needs to show a location

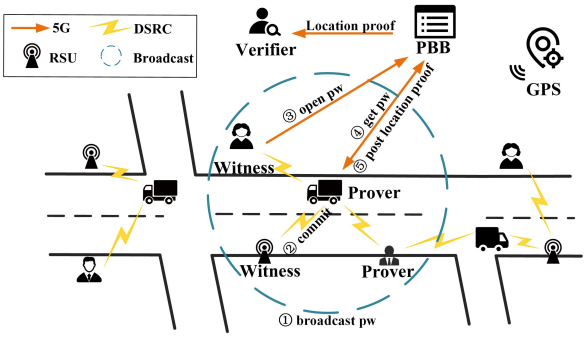

Fig. 3: Overview of Proof of Location. ① A prover broadcasts its GTOTP passwords (pw) and PPLP request to nearby witnesses via a short-range communication channel. ② Witnesses who can testify for the prover will respond with both message and location commitments regarding the PPLP responses. In steps ③ and ④, witnesses and prover exchange the password for verifying the message commitment. The prover finally assembles the location proof based on the gathered proof pieces in step ④ and publishes it to PBB. ⑤ The verifier can obtain the location proof from either PBB or prover.

proximity proof with the witness to the verifier. To protect the witnesses' location privacy, we leverage a PPLP scheme to attest the proximity between the witness and the prover. We use ephemeral public keys, each of which is associated with a location proof, for the PPLP scheme to ensure privacy.

To address the third problem, we combine a hash-based commitment scheme with the above GTOTP scheme and PPLP scheme. In particular, the prover and witnesses commit the location proximity proof generated by PPLP to the verifier as the location proof. These commitments are called *location commitments*. Here, the key of the location commitment is the secret seed of the GTOTP scheme. Note that the secret seed is verifiable since it can be used to generate the passwords in GTOTP. To enable the prover to verify the messages (including the location proofs) from witnesses, each witness generates a commitment keyed by the next password being used to commit the sent message. Such a commitment is called *message commitment*. We leverage a password to be the commitment key rather than a key derived from the secret seed since the message commitment key should be verifiable in a shorter time than the life-span of a secret seed. During verification, the prover and the witnesses can open their commitments to the verifier to prove the location. The verifier accepts the location of prover if all involved passwords and commitments of witnesses and prover are valid, and most of the witnesses' locations are close to that of the prover. Figure 3 illustrates the steps for generating location proofs.

Input: Group members with identities $\mathsf{GP} = \{\mathsf{ID}_i\}_{i\in[U]}$ which are either witness or prover, the start and end times T_s and T_e, length of each verify-epoch Δ_e, and proof generation generation interval Δ_s
Output: Verification key $\mathsf{vk_G}$

1. RA generates system parameters by running $\mathsf{pms}_1 := \mathsf{GTOTP.Setup}(1^\kappa, T_s, T_e, \Delta_e, \Delta_s)$ and $\mathsf{pms}_2 := \mathsf{PPLP.Setup}(1^\kappa)$ publishes the parameter $\mathsf{pms} = (\mathsf{pms}_1, \mathsf{pms}_2)$.
2. Each group member ID_i initializes its secret key and verification state by running $(sk^1_{\mathsf{ID}_i}, vst_{\mathsf{ID}_i}) := \mathsf{GTOTP\text{-}MT.PInit}(\mathsf{ID}_i)$ and $sk^2_{\mathsf{ID}_i} := \mathsf{F.Setup}(1^\kappa)$, and sends vst_{ID_i} to RA via a mutual authenticated secure channel as registration request.
3. Upon receiving registration request, RA generates the group verification state $(vst_{\mathsf{G}}, \{\mathsf{Ax}_{\mathsf{ID}_j}\}_{j\in U}) := \mathsf{GVSTGen}(\mathsf{GP}, \{vst_{\mathsf{ID}_j}\}_{j\in[U]})$, and sends $\mathsf{Ax}_{\mathsf{ID}_j}$ (if any) back to the corresponding party via the secure channel. Moreover, RA publishes the verification key $\mathsf{vk_G} = vst_{\mathsf{G}}$.
4. Each party ID_i keeps its secret key $sk_{\mathsf{ID}_i} = (sk^1_{\mathsf{ID}_i}, sk^2_{\mathsf{ID}_i})$ privately, and the prover would additionally store a counter cnt (which is initialized to be zero) to record the times of proof request.

Fig. 4: Registration of PoL

Detailed Construction. The details of sub-protocols *Registration*, *Location-Proof-Gen*, and *Verification* are shown in Figure 4, 5, and 6. The open algorithm is realized as follows.
$\mathsf{Open}(k_{\mathsf{RA}}, L_{\mathsf{P}}, i)$. This algorithm invokes the GTOTP-MT.Open algorithm. Specifically, RA retrieves the password pw_i of the i-th contributor and the time-stamp T of the proof, and returns $\mathsf{GTOTP\text{-}MT.Open}(k_{\mathsf{RA}}, pw_i, T)$.
Application to Contact Tracing for COVID-19. Our PoL scheme can support contact tracing, because the one-time passwords are exchanged via short-range communication, and the passwords of the contributors are recorded in the location proof. Such grouped passwords indicate contact. Meanwhile, a prover can regularly run the PoL protocol, e.g., every minute. More specifically, when the prover is confirmed to have been infected, she can let the verifier (who can be a hospital) publish the corresponding location proof in a high risk list. The RA extracts identities of the contributors of the location proof, and informs the corresponding witnesses about potential transmission. The witnesses of a location proof may not cover all close-contacts of the prover. Bluetooth-based contact tracing schemes [42,40] have certain limitations, because some people may refuse to run the contact-tracing application, or they do not have any Bluetooth-enabled devices. However, in our scheme,

Input: Prover P's secret key sk_{P} and location L_{P}, M witnesses' secret keys $\{sk_{\mathsf{W}_j}\}_{j\in[M]}$ and their locations $\{\mathrm{L}_{\mathsf{W}_j}\}_{j\in[M]}$, current time slot $\mathsf{T}_{\mathsf{current}}$, and verification key vk_{G}
Output: Location proof LP_{P}

1. The prover P generates a location proof request via the following steps:
   - Compute $cnt := cnt + 1$ and the secret seed $rk_{cnt} = \mathsf{F}(sk_{\mathsf{P}}^2, \text{'PPLP'}||cnt)$, and run key generation algorithm of the PPLP scheme $(\mathsf{epk}_{\mathsf{P}}^{cnt}, \mathsf{esk}_{\mathsf{P}}^{cnt}) := \mathsf{PPLP.KGen}(rk_{cnt})$ to get an ephemeral public and secret key pair;
   - Compute the i-th secret seed $sd_{\mathsf{P}}^i := \mathsf{GTOTP\text{-}MT.GetSD}(sk_{\mathsf{P}}^1, \mathsf{T}_{\mathsf{current}})$, and password $pw_{\mathsf{P}}^{i,z} := \mathsf{GTOTP\text{-}MT.PwGen}(sd_{\mathsf{P}}^i, \mathsf{T}_{\mathsf{current}})$;
   - Encrypt its location $C_{\mathsf{P}} := \mathsf{PPLP.LPInit}(\mathsf{pk}_{\mathsf{P}}^{cnt}, \mathrm{L}_{\mathsf{P}})$.
2. P broadcasts $(pw_{\mathsf{P}}^{i,z}, C_{\mathsf{P}})$ to nearby witnesses as location proof challenge via a short range communication channel (such as Bluetooth);
3. Upon receiving $(pw_{\mathsf{P}}^{i,z}, C_{\mathsf{P}})$, each witness rejects if $\mathsf{GTOTP\text{-}MT.Verify}(vst_{\mathsf{G}}, pw_{\mathsf{P}}^{i,z}, \mathsf{T}_{\mathsf{current}}) = 0$, otherwise it does the following:
   - Generate the PPLP response $C_{\mathsf{W}_j} := \mathsf{PPLP.LPResp}(\mathsf{epk}_{\mathsf{P}}^{cnt}, C_{\mathsf{P}}, \mathrm{L}_{\mathsf{W}_j})$;
   - Run $sd_{\mathsf{W}_j}^i := \mathsf{GTOTP\text{-}MT.GetSD}(sk_{\mathsf{W}_j}^1, \mathsf{T}_{\mathsf{current}})$, and compute $pw_{\mathsf{W}_j}^{i,z} := \mathsf{GTOTP\text{-}MT.PwGen}(sd_{\mathsf{W}_j}^i, \mathsf{T}_{\mathsf{current}})$ for the current time slot $\mathsf{T}_{\mathsf{current}}$.
   - Abort if $\mathsf{T}_{\mathsf{current}} + \Delta_s$ belongs to the next verify-epoch (which means there is no enough password to use in the current verify-epoch), otherwise initialize a commitment key using the next password $pw_{\mathsf{W}_j}^{i,z+1} := \mathsf{GTOTP\text{-}MT.PwGen}(sd_{\mathsf{W}_j}^i, \mathsf{T}_{\mathsf{current}} + \Delta_s)$;
   - Set $\mathrm{Tr}_{\mathsf{W}_j} := pw_{\mathsf{P}}^{i,z}||C_{\mathsf{P}}||C_{\mathsf{W}_j}||\mathrm{L}_{\mathsf{W}_j}||\mathsf{T}_{\mathsf{current}}$, and compute the location relevant commitment $\mathsf{LCT}_{\mathsf{W}_j} := \mathsf{H}_2(sd_{\mathsf{W}_j}^i||\mathrm{Tr}_{\mathsf{W}_j})$ and message relevant commitment $\mathsf{MCT}_{\mathsf{W}_j} := \mathsf{H}_2(pw_{\mathsf{W}_j}^{i,z+1}||\mathsf{LCT}_{\mathsf{W}_j})$;
   - Send its proof piece $(pw_{\mathsf{W}_j}^{i,z}, C_{\mathsf{W}_j}, \mathsf{LCT}_{\mathsf{W}_j}, \mathsf{MCT}_{\mathsf{W}_j})$ to prover P via a short range communication channel;
   - Record the tuple $(\mathsf{T}_{\mathsf{current}}, C_{\mathsf{W}_j})$ locally, and put $(pw_{\mathsf{W}_j}^{i,z+1}, \mathsf{MCT}_{\mathsf{W}_j})$ on a public bulletin board after time $\mathsf{T}_{\mathsf{current}} + 2\Delta_s$.
4. Upon receiving all proof pieces $\mathsf{RL}_{\mathsf{P}} = \{pw_{\mathsf{W}_j}^{i,z}, C_{\mathsf{W}_j}, \mathsf{LCT}_{\mathsf{W}_j}, \mathsf{MCT}_{\mathsf{W}_j}\}_{j\in[M]}$ from witnesses, P removes invalid pair from RL_{P} if one of the following conditions holds:
   - $\mathsf{GTOTP\text{-}MT.Verify}(vst_{\mathsf{G}}, pw_{\mathsf{W}_j}^{i,z}, \mathsf{T}_{\mathsf{current}}) = 0$;
   - $\mathsf{MCT}_{\mathsf{W}_j} \neq \mathsf{H}_2(pw_{\mathsf{W}_j}^{i,z+1}||\mathsf{LCT}_{\mathsf{W}_j})$, where $pw_{\mathsf{W}_j}^{i,z+1}$ can be obtained from the public bulletin board after time $\mathsf{T}_{\mathsf{current}} + \Delta_s$;
   - $0 = \mathsf{PPLP.LPCheck}(\mathsf{esk}_{\mathsf{P}}^{cnt}, C_{\mathsf{W}_j})$.
5. If the number of valid location proof pieces in RL_{P} is equal or greater than $\rho \leq M$, P prepares a transcript $\mathrm{Tr}_{\mathsf{P}} := \mathsf{epk}_{\mathsf{P}}^{cnt}||\mathsf{RL}_{\mathsf{P}}||\mathrm{L}_{\mathsf{P}}||\mathsf{T}_{\mathsf{current}}||cnt$ and computes the location relevant commitment $\mathsf{LCT}_{\mathsf{P}} := \mathsf{H}_2(sd_{\mathsf{P}}^i||\mathrm{Tr}_{\mathsf{P}})$; otherwise P aborts with failure.
6. P sends the location proof $\mathsf{LP}_{\mathsf{P}} = (pw_{\mathsf{P}}^{i,z}, \mathsf{epk}_{\mathsf{P}}^{cnt}, \mathsf{T}_{\mathsf{current}}, \mathsf{LCT}_{\mathsf{P}}, \mathsf{RL}_{\mathsf{P}})$ to verifier V, and put L_{P} on a public bulletin board.
7. V records LP_{P} if: for all $\mathsf{ID} \in \{\mathsf{P}, \{\mathsf{W}_j\}_{j\in[M]}\}$, $\mathsf{GTOTP\text{-}MT.Verify}(vst_{\mathsf{G}}, pw_{\mathsf{ID}}^{i,z}, \mathsf{T}_{\mathsf{current}}) = 1$

Fig. 5: Location Proof Generation of PoL

Input: Verification key vk_{G}, location proof LP_{P}, prover P's secret key sk_{P}, M witnesses' secret key $\{\mathsf{sk}_{\mathsf{W}_j}\}_{j\in[M]}$, and location L_{P} relevant to LP_{P}
Output: Verification result in $\{0, 1\}$

1. P does the following:
   - Get $T_d \in \mathsf{LP}_{\mathsf{P}}$ and return $\perp$ if $\mathsf{T}_{\mathsf{current}}$ and T_d belong to the same verify-epoch;
   - Compute the i-th secret seed $sd_{\mathsf{P}}^i := \mathsf{GTOTP\text{-}MT.GetSD}(sk_{\mathsf{P}}^1, T_d)$, and t-th the ephemeral random seed $rk_t = \mathsf{PRF}(sk_{\mathsf{P}}^1, \text{'PPLP'}||cnt')$;
   - If sd_{ID}^i has expired, open $(sd_{\mathsf{P}}^i, rk_t, \mathrm{L}_{\mathsf{P}})$ to verifier V over a unilateral authenticated secure channel, otherwise abort;
   - Put all passwords of witnesses in LP_{P} as testifying request on the public bulletin board.
2. Upon receiving notification, each witness W_j returns $\perp$ if $\mathsf{T}_{\mathsf{current}}$ and T_d belong to the same verify-epoch, otherwise it opens $sd_{\mathsf{W}_j}^i := \mathsf{GTOTP\text{-}MT.GetSD}(sk_{\mathsf{W}_j}^1, T_d)$ and $\mathrm{L}_{\mathsf{W}_j}$ to verifier V over a unilateral authenticated secure channel analogously;
3. Upon receiving secret seed and location pairs from all parties, for all $\mathsf{ID} \in \{\mathsf{P}, \{\mathsf{W}_j\}_{j\in[M]}\}$, V verifies each proof piece in LP_{P}, i.e., V marks it as *invalid* if one of the following conditions holds:
   - The $\mathsf{epk}_{\mathsf{P}}^{cnt'}$ obtained by running $(\mathsf{epk}_{\mathsf{P}}^{cnt'}, \mathsf{esk}_{\mathsf{P}}^{cnt'}) := \mathsf{PPLP.KGen}(rk_{cnt'})$ is not in LP_{P}, , where $cnt' \in \mathsf{LP}_{\mathsf{P}}$;
   - A password computed based on sd_{ID}^i, i.e., $pw_{\mathsf{ID}}^{i,z} := \mathsf{GTOTP\text{-}MT.PwGen}(sd_{\mathsf{ID}}^i, T_d)$, is invalid or $pw_{\mathsf{ID}}^{i,z} \notin \mathsf{LP}_{\mathsf{P}}$;
   - $\mathsf{LCT}_{\mathsf{ID}} \neq \mathsf{H}_2(sd_{\mathsf{ID}}^i||\mathrm{Tr}_{\mathsf{ID}})$ where $\mathsf{LCT}_{\mathsf{ID}} \in \mathsf{LP}_{\mathsf{P}}$ and $\mathrm{Tr}_{\mathsf{ID}} = \mathsf{RL}_{\mathsf{P}}||C_{\mathsf{P}}||T_i$ if $\mathsf{ID} = \mathsf{P}$ and $\mathrm{Tr}_{\mathsf{ID}} = pw_{\mathsf{P}}^i||C_{\mathsf{P}}||C_{\mathsf{W}_j}||T_i$ otherwise;
   - $\mathsf{ID} = \mathsf{P}$ and C_{P} is an invalid ciphertext of L_{P} that is verified based on the secret key sk_{P}.
   - $\mathsf{ID} = \mathsf{W}_j$ and $0 = \mathsf{PPLP.LPCheck}(\mathsf{esk}_{\mathsf{P}}^{cnt'}, C_{\mathsf{W}_j})$.
4. V returns 0 (meaning invalid location proof) if the number of invalid location proof pieces (as checked above) is not greater than $\rho \leq M$, otherwise 1 is returned;

Fig. 6: Verification of PoL

after the verifier (anonymously) releases the prover's past locations after verifying the corresponding location proofs (generated by PoL), the other entities can check whether they are in close contact with the prover. In other words, our scheme can help other people outside of the system achieve contact tracing based on the possibly contaminated location released by our scheme. This is not achievable by other proximity-tracing only schemes [42,40]. The contact locations may also help the centre for disease control to make strategic decisions to stop the propagation of the Covid-19.

Security Analysis. We show that our PoL scheme achieves security via the following theorems. Due to limit of space, their proofs will be given in the full version of this paper.

Theorem 3. *Suppose that the GTOTP scheme and the PPLP scheme are secure, and the hash function* H_2 *is modeled as a random oracle. Then* PoL *achieves anonymity.*

Since the public keys used for running the PPLP scheme are ephemeral for generating a location proof, they do not leak any identity related information. In addition, the anonymity of the GTOTP scheme implies the privacy of the entities in PoL. We present the proof in Appendix G.

Theorem 4. *With the same assumptions in Theorem 3,* PoL *also achieves traceability.*

The full proof of this theorem is presented in Appendix H. This property is derived directly from the traceability of GTOTP. We note that the traceability of GTOTP covers the unforgeability of unused passwords and secret, the commitments computed using them are also unforgeable.

Theorem 5. *With the same assumptions in Theorem 3,* PoL *achieves location privacy.*

Since the adversary cannot forge the commitments due to the traceability property, the location privacy of the entities in PoL is derived from that of the PPLP scheme. The proof of this theorem is presented in Appendix I.

Discussion on Insider Threats. In recent years, insider attacks [46,33,38] have become a major threats against cryptographic schemes. Our PoL construction can resist against insider witnesses, who intend to falsify or tamper with the location proof, as long as the dishonest witnesses are the minority in generating the target location proof. To establish the majority of the honest witnesses, the prover can invite more witnesses for executing the PoL protocol at a time. However, our construction cannot prevent an insider prover from installing malicious witnesses to provide dishonest proof of a location. The detailed solutions regarding preventing insider attackers are out of the scope of this paper. Alternatively, we allow the RA to reveal the identity of any misbehaved participants due to the traceability of PoL. To prevent insider prover, it might be possible to restrict the prover to generate the location proof with witnesses which encompass physically faithful location beacons (such as FOAM [29]). However, it is an open question to design a PoL scheme that can resist insider attackers without trustworthy location beacons.

6 Evaluation

We implement our GTOTP-MT and PoL schemes on Raspberry Pi 3. For the GTOTP-MT implementation, we use the TOTP scheme with a single hash-chain introduced in [26]. The hash functions, including H_1 and H_2, are implemented using SHA256. We implement the PPLP scheme as proposed by Järvinen et al. [23,24] which uses ElGamal based additively homomorphic encryption and Elliptic Curve Cryptography (ECC). We use NIST Curve P-256 for the implementation. Since all entities can share ECC parameters, the expensive operations in PPLP.LPResp, which is to initialize blind distance sets by, are performed offline, as in [23,24]. We do not report this cost in our experiments. The authenticated symmetric encryption ASE is implemented as 128-bit AES-GCM-SIV [19,18].

In the experiments, we set a fixed life-span $\Delta_e = 5$ minutes (m) for each verify point, but vary the number of verify points for each entity. We set the password generation interval $\Delta_s = 5$ seconds (s), which gives $N = 60$. We set $\epsilon = 40$ for the Bloom filter, which is sufficient to guarantee a negligible false-positive rate for a duration of less than 1 year. The number of Merkle trees in the GTOTP-MT scheme is $\phi = 2^{13} = 8192$, and the distance threshold in PoL is $\theta = 50$ (meters).

We run prover, verifier, and witness on Raspberry Pi 3, and the RA on a PC with Intel i7 CPU and 2GB RAM. The results reported below are averaged over 1000 runs. We note that our current implementation is not yet optimized, for example, we do not exploit all available CPU cores. In other words, the results below can be further improved with multi-threadings.

6.1 Performance of GTOTP

Initialization Time. Figure 7 (a) shows the computational cost of GTOTP-MT.PInit. Although the cost increases linearly with E, we note that each initialization takes approximately 10ms, which is practical.

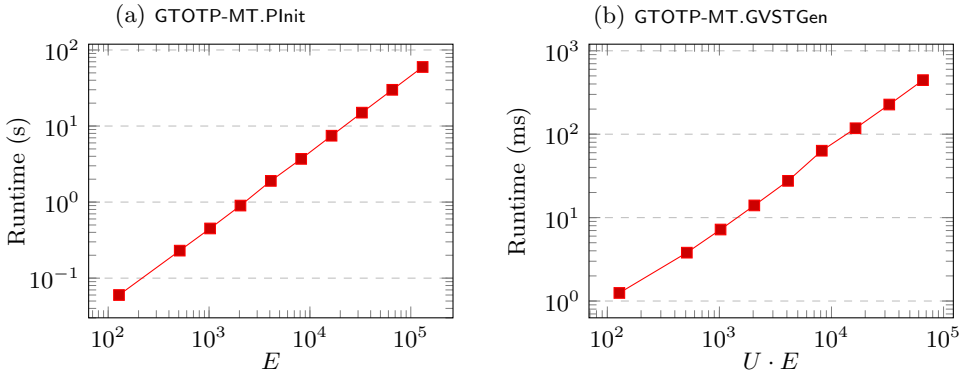

Fig. 7: Runtimes of GTOTP-MT.PInit and GTOTP-MT.GVSTGen.

Password Generation Time. To evaluate the cost of password generation, we assume that there is enough storage for caching passwords [26]. This is reasonable since each TOTP instance is used only for a short verify epoch with a small number of passwords. In particular, for $N = 60$ the storage cost is less than 2KB. Once the GTOTP instance starts, the prover can use one password in one segment in the reverse order and generates one password in the next verify epoch. This way, the prover always has one full segment of passwords in memory, and only needs to compute one hash function in the average case, and one more PRF in the worst case to switch TOTP instances. In our experiments, we observe that the cost is low: $4.12\mu s$ and $19.1\mu s$ for the average and worst case, respectively.

Group Verification State Generation Time. Figure 7 (b) shows the cost of GTOTP-MT.GVSTGen with increasing $U \cdot E$. As expected, this cost grows linearly with the total number of verify points. Each verify point contributes less than 1ms to the total time. We emphasize that this cost can be significantly improved with better hardware and implementation.

Password Verification Time. We measure the worst-case performance, in which the password being verified is the last one in a TOTP instance. The verifier needs to first go through the whole chain to verify it, which requires N hash functions, then verify the Merkle proof associated with the password, which requires $\log^{(U \cdot E)/\phi}$ times function. On average, the verifier may only compute $N/2$ hash functions per verify point. Figure 8 (a) shows verification costs in terms of the height of the Merkle trees. It can be seen that the verification time is in the order of milliseconds, which is practical. Since the chain length N is fixed, the costs vary in terms of the height of Merkle trees. As N is larger than the height, one can reduce the cost by decreasing the life-span of each verify point.

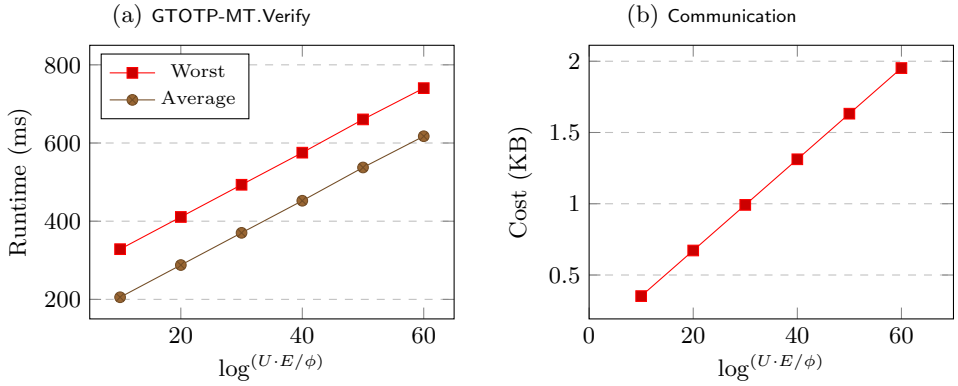

Fig. 8: Runtime of GTOTP-MT.Verify and Communication Cost.

Identity Open Time. The overhead of GTOTP-MT.Open consists of the costs of the verification algorithm and a decryption of ASE. The performance of GTOTP-MT.Verify is 10 times faster on RA (simulated by PC) than on the Raspberry Pi (as shown in Figure 8). The runtime of ASE.Dec is about $0.5\mu s$ on PC.

Communication Cost. The communication cost is determined by the size of the password, which consists of a TOTP password and the Merkle proof. Figure 8 (b) shows the communication cost, which increases with the height of the Merkle tree, as expected. Even for big trees with height of 60, the verifier receives less than 2KB for each verify point.

6.2 Performance of PoL

During registration, PoL mainly invoke the GTOTP algorithms, thus its cost is the sum of the cost of GTOTP-MT.PInit and GTOTP-MT.GVSTGen. In the PoL experiments, we fix the height of the Merkle tree to be $\log \frac{U \cdot E}{\phi} = 30$ to reduce the dimension of the parameters.

Proof Generation Time. We measure the time of proof generation (PfGen) as the sum of the cost of the prover and the witnesses. Table 1 summarizes the results for different number of witnesses M. The prover performs $4 + M(\tilde{N} + \log \frac{U \cdot E}{\phi} + 2)$ hash evaluations and $M + 10$ EC point multiplications (EPM), where $\tilde{N} = N$ for worst case and $\tilde{N} = N/2$ for average case. The main cost of the witness consists of $N + \log \frac{U \cdot E}{\phi} + 8$ hash evaluations and 12 EC point multiplications.

Table 1: Runtimes of Location Proof Generation and Verification, and the Size of Location Proof.

M	Computation time (s)				PfSize (KB)
	PfGen			Verify	
	Prover	Witness	Total	Verifier	
5	0.116/0.133	0.089/0.098	0.205/0.231	0.00065	1.16
10	0.237/0.276	0.089/0.098	0.326/0.347	0.0011	2.17
15	0.331/0.382	0. 0.089/0.098	0.42/0.48	0.0018	3.19

Verification Time. During verification, the prover and the witnesses open their secret seeds. This opening requires one PRF evaluation, which takes $10\mu s$. Table 1 reports the cost of the verifier which includes $5M$ hash evaluations and $2M$ EC point multiplications.

Proof Size. Table 1 shows the sizes of the location proofs (PfSize) when the Merkle's tree height is 30. Each time stamp costs 32 bits, H_2 costs 256 bits, and an EC point (P) 448 bit. Each ciphertext has the size of 128 bits. Thus, the total size is $32 + 2\mathsf{H}_2 + P + (3H + 2P + C) * M = (992 + 1792M)$ bits. Even with $M = 15$, each proof is only 3KB, which is practical.

6.3 Comparison

(G)TOTP Comparison. Here we generically compare GTOTP-MT with a state-of-the-arts TOTP scheme [30] and a group signature scheme [11] from the perspectives of security properties and performance. In the comparison, we use the same setting and parameters as in the above benchmarks. We let 'PwSize' denote the sie of password (which implies the communication cost), 'Unforg' denote the unforeability, 'BFc' denote the cost of BF.Check, and 'BFi' denote the cost of BF.Insert. Meanwhile, we compare the worst-case performance between GTOTP and TOTP for simplicity. Moreover, we let 'Ex' and 'Par' denote an exponentiation and a pairing operation relative to bilinear groups, respectively.

The comparison results are shown in Table 2. The initialization and password generation algorithms of GTOTP-MT scheme does not introduce significant overheads comparing with the TOTP scheme in [26]. The verification cost of GTOTP-MT mainly involves additional operations regarding verifying the Merkle proof and checking the membership of the Bloom filter. The GVSTGen algorithm of GTOTP-MT is done by a powerful RA, so it can run very fast. Note that GTOTP-MT can provide more security properties than the TOTP scheme without many additional computational overheads. And the password generation and verification algorithms of GTOTP-MT are much more efficient than the GDS[11], since it does not require any expensive pairing and exponentiation operations, and $N \leq 60$ is small in GTOTP-MT. For example, from the result in [11], we can roughly estimate the costs of EPM and pairing as 300 and 2000 times the cost of H_1, respectively.

Table 2: (G)TOTP Comparison

	Security Properties	Performance				
		Init	PwGen	Verify	GVSTGen	APwSize (Bytes)
TOTP [30]	Unforg	$E \cdot N \cdot \mathsf{H}_1$	H_1	$N \cdot \mathsf{H}_1$	-	32
GDS [11]	Anony Trace	3EPM	21Mul + 2Ex +4Par + $1\mathsf{H}_1$	21EPM + 3Ex +6Par + $1\mathsf{H}_1$	40EPM + $1\mathsf{H}_1$	892
GTOTP-MT	Anony Trace	$E \cdot N \cdot \mathsf{H}_1$ +$E \cdot$ PRF	$N \cdot \mathsf{H}_1$ + PRF	$(N + 31)\mathsf{H}_1$ + 1BFc	$2U \cdot E \cdot \mathsf{H}_1$ +$\phi \cdot$ BFi	992

PoL Comparison. In Table 3, we compare our proposed PoL scheme with some existing related schemes in the perspectives of witness setting, the security properties regarding entity privacy (Enti-Priv) and location privacy

(Loc-Priv), and the performance. The number of witness is set to $M = 5$. Since the current communication techniques (such 5G and Bluetooth) are very fast, we do not take the propagation delay of communication into account. Here we consider the location privacy of all parties (including verifier). Since most of prior works do not have any implementation, we only count the major operations and proof size for performance comparison. For simplicity and fairness, we instantiate the regular digital signature and group digital signature used by prior works with ECDSA [27] and the one [11], respectively. The public encryption scheme used by previous schemes are instantiate with the ElGamal [13]. Futhermore, we let 'ExN' denote an exponentiation in the Pailiar encryption scheme.

The witness setting may implicitly reflect the system model (incl. functionalities). We let 'CLS' denote the *centralized location server* (which might be used to either provide the positioning and location-certifying service for the prover), and 'Ad Hoc' denote the setting that the witnesses could be any entity (as long as they can be used for testifying the location for the prover, e.g., a mobile device or location beacon), and 'DLB' denote the *distributed location beacon* (which achieves the similar functionality of CLS by distributed beacons). Note that the witness setting and the security properties of a PoL scheme imply the functionalities that can be achieved by the corresponding PoL scheme.

From Table 3, we can see that only few schemes consider entity privacy. Meanwhile, only [10] and our scheme provide traceability while preserving the entity privacy and location privacy. The PoL scheme by Zhu et al. [51] utilizes some kinds of mathematical combination operations to achieve entity and location privacy (without achieving traceability), but the adversary may have non-negligible advantages in breaking these two security properties. Compared with [51], our scheme leverages provably secure cryptographic building blocks. Moreover, our scheme adopts a more flexible witness setting. Although our scheme only provides the location privacy of prover to witnesses (but not to verifier), it might be weaker than that of [49] (which uses zero-knowledge proof scheme to achieve the location privacy to verifier). However, Wu et al. [49] scheme needs distributed access points to be the location beacons (whose locations are known to the verifier) to make the zero-knowledge proofs verifiable to the verifier. Moreover, in many applications, the location privacy of the prover to the verifier is not mandatory. For example, in contact-tracing, the verifier needs to know the location of the prover for disinfection. In contrast, the entity and location privacy of both witnesses and prover to each other is the main concern in our scheme. While comparing with [10] (that provides similar security properties as our proposal), it uses much more expensive building blocks than ours.

Table 3: PoL Comparison

	Witness Setting	Security Properties					Performance			
		Trace	Enti-Priv		Loc-Priv		PfGen		Verify	PfSize (KB)
			Prover	Witness	Prover	Witness	Prover	Witness		
[48]	CLS	√	×	×	×	×	$7\text{EPM} + 1\mathsf{H}_1$	$3\text{EPM} + 1\mathsf{H}_1$	$6\text{EPM} + 2\mathsf{H}_1$	0.36
[17]	Ad Hoc	√	×	×	×	×	$27\text{EPM} + 3\mathsf{H}_1$	$4\text{EPM} + 2\mathsf{H}_1$	$28\text{EPM} + 6\mathsf{H}_1$	1.4
[51]	CLS	×	√	√	√	×	$10\text{EPM} + 5\mathsf{H}_1$	$3\text{EPM} + 1\mathsf{H}_1$	$15\text{EPM} + 5\mathsf{H}_1$	1.4
[36]	Ad Hoc	√	×	×	×	×	$8\text{EPM} + 2\mathsf{H}_1$	$5\text{EPM} + 3\mathsf{H}_1$	$17\text{EPM} + 5\mathsf{H}_1$	1.3
[29]	DLB	√	×	×	×	×	$10\text{EPM} + 5\mathsf{H}_1$	$3\text{EPM} + 1\mathsf{H}_1$	$15\text{EPM} + 5\mathsf{H}_1$	1.3
[49]	DLB	×	√	×	√	×	$28\text{EPM} + 8\mathsf{H}_1$	$56\text{EPM} + 4\text{Mul} + 2\mathsf{H}_1$	$11\text{Par} + 5\text{EPM} + 5\text{Mul} + 1\mathsf{H}_1$	0.56
[10]	Ad Hoc	√	√	√	×	√	$87\text{ExN} + 252\text{EPM} + 34\text{Ex} + 68\text{Par} + 12\mathsf{H}_1$	$597\text{ExN} + 42\text{EPM} + 2\text{Ex} + 8\text{Par} + 2\mathsf{H}_1$	$5\text{ExN} + 126\text{EPM} + 12\text{Ex} + 24\text{Par} + 6\mathsf{H}_1$	7.94
Ours	Ad Hoc	√	√	√	×	√	$464\mathsf{H}_1 + 15\text{EPM}$	$98\mathsf{H}_1 + 12\text{EPM}$	$25\mathsf{H}_1 + 10\text{EPM}$	1.16

7 Conclusions

In this paper, we proposed Group time-based one-time passwords scheme (GTOTP) that extends TOTP to the group setting. GTOTP achieves membership authentication and privacy. We presented an efficient GTOTP construction, which is based on an asymmetric TOTP scheme and other standard cryptographic building blocks including Merkle tree, pseudo-random function family, and collision-resistant hash function. We showed how to apply GTOTP to construct an efficient proof of location (PoL) scheme, which can be used for contact tracing. We believe that the GTOTP is useful in many other applications beyond PoL.

For future work, we plan to formulate our security models with Universally Composable (UC) Security [3] which admits stronger adversaries. We will also extend our GTOTP scheme to support dynamic groups.

Acknowledgement

We would like to thank our shepherd and anonymous reviewers for their invaluable comments and suggestions. This work is supported by the Natural Science Foundation of China (Grant Nos.61872051, 62032005, 61972094, and 61802214), and A*STAR under its RIE2020 Advanced Manufacturing and Engineering (AME) Industry

Alignment Fund - Pre Positioning (IAF-PP) Award A19D6a0053. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of A*STAR.

References

1. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: EUROCRYPT. LNCS, vol. 2656, pp. 614–629. Springer (2003)
2. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: EUROCRYPT. LNCS, vol. 4004, pp. 409–426. Springer (2006)
3. Canetti, R.: Universally composable security. J. ACM **67**(5), 28:1–28:94 (2020)
4. Chaum, D., Van Heyst, E.: Group signatures. In: Workshop on the Theory and Application of of Cryptographic Techniques. pp. 257–265. Springer (1991)
5. Choudhuri, A.R., Green, M., Jain, A., Kaptchuk, G., Miers, I.: Fairness in an unfair world: Fair multiparty computation from public bulletin boards. In: CCS. pp. 719–728. ACM (2017)
6. Dierks, T., Rescorla, E.: The transport layer security (tls) protocol version 1.2. Tech. rep. (2008)
7. Dodis, Y., Fiore, D.: Unilaterally-authenticated key exchange. In: FC. LNCS, vol. 10322, pp. 542–560. Springer (2017)
8. Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In: CCS. pp. 1197–1210. ACM (2015)
9. Duo: Duo security. https://duo.com/ (2018), Accessed: Dec. 2018
10. Dupin, A., Robert, J., Bidan, C.: Location-proof system based on secure multi-party computations. In: ProvSec. LNCS, vol. 11192, pp. 22–39. Springer (2018)
11. Emura, K., Hayashi, T., Ishida, A.: Group signatures with time-bound keys revisited: A new model, an efficient construction, and its implementation. IEEE Trans. Dependable Secur. Comput. **17**(2), 292–305 (2020)
12. Gajek, S., Manulis, M., Sadeghi, A., Schwenk, J.: Provably secure browser-based user-aware mutual authentication over TLS. In: AsiaCCS. pp. 300–311. ACM (2008)
13. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory (1985)
14. Garcia, F.D., van Rossum, P.: Modeling privacy for off-line RFID systems. In: CARDIS. LNCS, vol. 6035, pp. 194–208. Springer (2010)
15. Google: Certificate transparency. https://www.certificate-transparency.org/ (2017)
16. Google: Google 2-step verification. https://www.google.com/landing/2step/ (2018), Accessed: Dec. 2018
17. Graham, M., Gray, D.: Protecting privacy and securing the gathering of location proofs - the secure location verification proof gathering protocol. In: Schmidt, A.U., Lian, S. (eds.) Security and Privacy in Mobile Information and Communication Systems, First International ICST Conference, MobiSec 2009, Turin, Italy, June 3-5, 2009, Revised Selected Papers. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 17, pp. 160–171. Springer (2009)
18. Gueron, S.: Aes-gcm-siv implementations (128 and 256 bit) (2018), https://github.com/Shay-Gueron/AES-GCM-SIV
19. Gueron, S., Langley, A., Lindell, Y.: Aes-gcm-siv: Specification and analysis. IACR Cryptology ePrint Archive **2017**, 168 (2017)
20. Haller, N.: The s/key one-time password system (1995)
21. Institute, E.T.S.: Etsi ts 102 940. intelligent transportation systems (its); security; its communications security architecture and security management. Tech. Rep. (2018)
22. Jarecki, S., Krawczyk, H., Xu, J.: OPAQUE: an asymmetric PAKE protocol secure against pre-computation attacks. In: EUROCRYPT (3). LNCS, vol. 10822, pp. 456–486. Springer (2018)
23. Järvinen, K., Kiss, Á., Schneider, T., Tkachenko, O., Yang, Z.: Faster privacy-preserving location proximity schemes. In: CANS. LNCS, vol. 11124, pp. 3–22. Springer (2018)
24. Järvinen, K., Kiss, Á., Schneider, T., Tkachenko, O., Yang, Z.: Faster privacy-preserving location proximity schemes for circles and polygons. IET Inf. Secur. **14**(3), 254–265 (2020)
25. Järvinen, K., Leppäkoski, H., Lohan, E.S., Richter, P., Schneider, T., Tkachenko, O., Yang, Z.: PILOT: practical privacy-preserving indoor localization using outsourcing. In: EuroS&P. pp. 448–463. IEEE (2019)
26. Jin, C., Yang, Z., van Dijk, M., Zhou, J.: Proof of aliveness. In: ACSAC. pp. 1–16. ACM (2019)
27. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ecdsa). International Journal of Information Security **1**(1) (2001)
28. Katsumata, S., Yamada, S.: Group signatures without NIZK: from lattices in the standard model. In: EUROCRYPT (3). LNCS, vol. 11478, pp. 312–344. Springer (2019)
29. King, R.J.: Foam:introduction to proof of location. https://blog.foam.space/introduction-to-proof-of-location-6b4c77928022/ (2018), accessed: 2020-11-1
30. Kogan, D., Manohar, N., Boneh, D.: T/key: Second-factor authentication from secure hash chains. In: CCS. pp. 983–999. ACM (2017)
31. Krawczyk, H., Paterson, K.G., Wee, H.: On the security of the TLS protocol: A systematic analysis. In: CRYPTO (1). LNCS, vol. 8042, pp. 429–448. Springer (2013)
32. Lamport, L.: Constructing digital signatures from a one-way function. Tech. rep., Technical Report CSL-98, SRI International Palo Alto (1979)

33. Li, W., Wang, D., Wang, P.: Insider attacks against multi-factor authentication protocols for wireless sensor networks. J Softw **30**, 2375–2391 (2019)
34. Merkle, R.C.: A digital signature based on a conventional encryption function. In: CRYPTO. LNCS, vol. 293, pp. 369–378. Springer (1987)
35. Naor, M., Yogev, E.: Bloom filters in adversarial environments. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 9216, pp. 565–584. Springer (2015)
36. Nosouhi, M.R., Sood, K., Yu, S., Grobler, M., Zhang, J.: PASPORT: A secure and private location proof generation and verification framework. IEEE Trans. Comput. Soc. Syst. **7**(2), 293–307 (2020)
37. Puniya, P.: New Design Criteria for Hash Functions and Block Ciphers. Ph.D. thesis, USA (2007), aAI3310560
38. Rajamanickam, S., Vollala, S., Amin, R., Ramasubramanian, N.: Insider attack protection: Lightweight password-based authentication techniques using ecc. IEEE Systems Journal **14**(2), 1972–1983 (2019)
39. Rescorla, E.: The transport layer security (tls) protocol version 1.3. Tech. rep. (2018)
40. Rivest, R.L., Weitzner, D., Ivers, L., Soibelman, I., Zissman, M.: Pact: Private automated contact tracing (2020)
41. Shoup, V.: Sequences of games: A tool for taming complexity in security proofs. IACR Cryptol. ePrint Arch. **2004** (2004)
42. Troncoso, C., Payer, M., Hubaux, J.P., Salathé, M., Larus, J., Bugnion, E., Lueks, W., Stadler, T., Pyrgelis, A., Antonioli, D., et al.: Decentralized privacy-preserving proximity tracing. arXiv preprint arXiv:2005.12273 (2020)
43. Vanhoef, M., Matte, C., Cunche, M., Cardoso, L.S., Piessens, F.: Why MAC address randomization is not enough: An analysis of wi-fi network discovery mechanisms. In: AsiaCCS. pp. 413–424. ACM (2016)
44. Verheul, E., Hicks, C., Garcia, F.D.: Ifal: Issue first activate later certificates for v2x. In: EuroS&P. pp. 279–293. IEEE, IEEE (2019)
45. Wails, R., Johnson, A., Starin, D., Yerukhimovich, A., Gordon, S.D.: Stormy: Statistics in tor by measuring securely. In: CCS. pp. 615–632. ACM (2019)
46. Wang, D., He, D., Wang, P., Chu, C.: Anonymous two-factor authentication in distributed systems: Certain goals are beyond attainment. IEEE Trans. Dependable Secur. Comput. **12**(4), 428–442 (2015)
47. Wang, D., Wang, P.: Two birds with one stone: Two-factor authentication with security beyond conventional bound. IEEE Trans. Dependable Secur. Comput. **15**(4), 708–722 (2018)
48. Waters, B.R., Felten, E.W.: Secure, private proofs of location. Tech. rep. (2002), www.cs.princeton.edu/research/techreps/TR-667-03
49. Wu, W., Liu, E., Gong, X., Wang, R.: Blockchain based zero-knowledge proof of location in iot. In: 2020 IEEE International Conference on Communications, ICC 2020, Dublin, Ireland, June 7-11, 2020. pp. 1–7. IEEE (2020)
50. Yang, Z., Jin, C., Ning, J., Li, Z., Dinh, A., Zhou, J.: Group time-based one-time passwords and its application to efficient privacy-preserving proof of location. In: ACSAC. pp. 497–512. ACM (2021)
51. Zhu, Z., Cao, G.: Toward privacy preserving and collusion resistance in a location proof updating system. IEEE Trans. Mob. Comput. **12**(1), 51–64 (2013)

A Lists of Notions

Some important notations used in this paper are listed in Table 4.

Table 4: Some Important Notations for GTOTP

U	Number of group members
GP	Identifies of group members $\mathsf{GP} = \{\mathsf{ID}_1, \ldots, \mathsf{ID}_U\}$
T_s, T_e	Start and end times of a protocol instance, respectively
Δ_e, Δ_s	life-spans of verify-point and password, respectively.
E, N	Numbers of verify-points and passwords (that can be verified by each verify-point), respectively.
vp_{ID}	Verify-point of a party ID
vst_{ID}	Verification state of a party, s.t., $vst_{\mathsf{ID}} = \{vp^i_{\mathsf{ID}}\}$ for $i \in [\Delta_t/\Delta_e]$.
vst_{G}	Group verification state $vst_{\mathsf{G}} = \{vst_{\mathsf{ID}_j}\}$ for $j \in [U]$.
sd^i_{ID}	The i-th secret seed for generating the i-the verify-point of ID.
sk_{ID}	Secret key for generating the secret seeds of ID.
$pw^{i,z}_{\mathsf{ID}}$	The z-th password of ID in the i-th verify-epoch.
T_i, $\mathsf{T}_{\mathsf{current}}$	The i-th time slot and the current system time slot, respectively.
M	Number of witness
$\mathsf{P}_j, \mathsf{W}_j$	Identities of prover and witness, respectively
GP	Identifies of group members $\mathsf{GP} = \{\mathsf{P}_1, \ldots, \mathsf{P}_U\}$
$\mathsf{MCT}, \mathsf{LCT}$	Commitments of location and message, respectively
RL_{P}	Location proof pieces received by prover P
LP_{P}	Location proof generated by prover P

B Security Definitions of Building Blocks

Pseudo-Random Functions. The security of PRF requires that, on given $r = \mathsf{F}(k, x)$ for a random key $k \xleftarrow{\$} \mathcal{K}_{\mathsf{PRF}}$ and a message $x \in \mathcal{M}_{\mathsf{PRF}}$, no efficient algorithm can distinguish r from a truly random value. We define a security game $G^{\mathsf{PRF}}_{\mathcal{A},\mathsf{F}}(\kappa, q_f)$ (shown in Figure 9) that is played between a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ and a challenger based on a pseudo-random function family F and the security parameter κ.

$G^{\mathsf{PRF}}_{\mathcal{A},\mathsf{F}}(\kappa, q_f)$:	
Initialize() :	**Finalize(b^*) :**
$(\mathsf{pms}, k) \leftarrow \mathsf{F.Setup}(1^\kappa)$	IF $b^* = b$ and $x^* \notin \mathsf{FL}$, OUTPUT 1
OUTPUT pms:	ELSE OUTPUT 0
Challenge(x^*) :	**FuncQ(x) :**
$b \xleftarrow{\$} \{0,1\}$, $r_0 \xleftarrow{\$} \mathcal{R}_{\mathsf{PRF}}$, $r_1 \leftarrow \mathsf{F.Eval}(k, x^*)$	APPEND $x \rightarrow \mathsf{FL}$
OUTPUT r_b	OUTPUT $\mathsf{F.Eval}(k, x)$

Fig. 9: Procedures Used to Define Security for PRF.

Definition 2. *We say* F *is secure if no PPT adversary has a non-negligible advantage* $\mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A},\mathsf{F}}(\kappa, q_f) := \left|\Pr[G^{\mathsf{PRF}}_{\mathcal{A},\mathsf{F}}(\kappa, q_f) = 1] - \frac{1}{2}\right|$ *in breaking the security of a pseudo-random function family* F *under* κ.

Authenticated Symmetric Encryption. We define a security game $G^{\mathsf{IND\text{-}CCA}}_{\mathcal{A},\mathsf{ASE}}(\kappa, q_e)$ in Figure 10 to formulate the standard notion of indistinguishability under chosen-ciphertext attack (IND-CCA), that is played between a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ and a challenger based on ASE and the security parameter κ.

$G^{\mathsf{IND\text{-}CCA}}_{\mathcal{A},\mathsf{ASE}}(\kappa, q_e)$:		
Initialize() :	**Finalize(b^*) :**	**Challenge(m_0, m_1) :**
$(\mathsf{pms}, k) \leftarrow \mathsf{ASE.Setup}(1^\kappa)$	IF $b^* = b$ and $C^* \notin \mathsf{CL}$	$b \xleftarrow{\$} \{0,1\}$
OUTPUT pms:	OUTPUT 1	$C^* \leftarrow \mathsf{ASE.Enc}(k, m_b)$
	OUTPUT 0	OUTPUT C^*
DecP(C) :		
APPEND $C \rightarrow \mathsf{CL}$	**EncP(m) :**	
OUTPUT $\mathsf{ASE.Dec}(k, C)$	OUTPUT $\mathsf{ASE.Enc}(k, m)$	

Fig. 10: Procedures Used to Define Security for ASE.

Definition 3. *We say* ASE *is secure if no PPT adversary has a non-negligible advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{A},\mathsf{ASE}}(\kappa, q_e) := \left|\Pr[G^{\mathsf{IND\text{-}CCA}}_{\mathcal{A},\mathsf{ASE}}(\kappa,\ q_e) = 1] - \frac{1}{2}\right|$ *in breaking the security of a authenticated symmetric encryption* ASE *under* κ.

Collision-resistant Hash Functions. The CRHF security game $G^{\mathsf{CR}}_{\mathcal{A},\mathsf{H}}(\kappa)$ based on an adversary $\mathcal{A}$ and a CRHF family H is defined in Figure 11.

$G^{\mathsf{CR}}_{\mathcal{A},\mathsf{H}}(\kappa)$:	
Initialize() :	**Finalize(m, m') :**
$hk \leftarrow \mathsf{H.Setup}(1^\kappa)$	IF $m \neq m'$ and $\mathsf{H.Eval}(hk, m) = \mathsf{H.Eval}(hk, m')$
OUTPUT hk	OUTPUT 1
	OUTPUT 0

Fig. 11: Procedures used to define security for CRH.

Definition 4. *We denote with* $\mathsf{Adv}^{\mathsf{CR}}_{\mathcal{A},\mathsf{H}}(\kappa) := \Pr[G^{\mathsf{CR}}_{\mathcal{A},\mathsf{H}}(\kappa) = 1]$ *the advantage of a PPT adversary* $\mathcal{A}$ *in breaking the security of* H *under the security parameter* κ. *We say* H *is secure if no PPT adversary has non-negligible advantage* $\mathsf{Adv}^{\mathsf{CRHF}}_{\mathcal{A},\mathsf{H}}(\kappa)$ *under* κ.

Time-based One-time Passwords. We define a security game $G^{\mathsf{TOTP\_Forge}}_{\mathcal{A},\mathsf{TOTP}}(\kappa, T_s, T_e, \Delta_s)$ for a time-based one-time passwords scheme TOTP in Figure 12. The goal adversary in the game is to forge a valid password of TOTP for a future time.

$G^{\mathsf{TOTP\_Forge}}_{\mathcal{A},\mathsf{TOTP}}(\kappa, T_s, T_e, \Delta_s)$:	
Initialize(T_s, T_e, Δ_s) :	**Finalize**() :
$\mathsf{pms} \leftarrow \mathsf{TOTP.Setup}(1^\kappa, T_s, T_e, \Delta_s)$	IF $\exists (pw^*, T^*) \in \mathsf{HD}$
$sd \xleftarrow{\$} \mathcal{K}_{\mathsf{TOTP}}$	s.t. $(\mathsf{TOTP.Verify}(vp, pw^*, T^*) = 1$
$vp \leftarrow \mathsf{TOTP.PInit}(sd)$	and no GetNextPw() at $\tilde{T}$ s.t. $\tilde{T} > T^* - \Delta_s$
OUTPUT pms, vp	OUTPUT 1
	OUTPUT 0
GetNextPw() :	**ReceivePw**(pw) :
OUTPUT $\mathsf{TOTP.PGen}(sd, \mathsf{T_{current}})$	APPEND $(pw, \mathsf{T_{current}}) \rightarrow \mathsf{HD}$
	OUTPUT $\mathsf{TOTP.Verify}(vp, pw, \mathsf{T_{current}})$

Fig. 12: Procedures used to define security for TOTP

Definition 5. *We say a TOTP protocol is secure if no PPT adversary has a non-negligible advantage* $\mathsf{Adv}^{\mathsf{TOTP\_Forge}}_{\mathcal{A},\mathsf{TOTP}}(\kappa, T_s, T_e, \Delta_s) := \Pr\left[G^{\mathsf{TOTP\_Forge}}_{\mathcal{A},\mathsf{TOTP}}(\kappa, T_s, T_e, \Delta_s) = 1\right]$ *in breaking the security of a TOTP protocol* TOTP *with the given parameters.*

Bloom filter. We define a security game (derived from [35]) $G^{\mathsf{AR}}_{\mathcal{A},\mathsf{BF}}(\kappa, T, N, \epsilon)$ that is played between an adversary $\mathcal{A}$ and a challenger based on a Bloom filter BF scheme and the parameters (κ, T, N, ϵ), where T is the time of the Bloom filter being used. Also, the parameter T defines the running time of adversaries in the game. We assume that the Bloom filter instance BF itself does not record any randomness used during the execution of the initiation algorithm BF.Init. If a Bloom filter has a randomized initialization algorithm but a deterministic query algorithm that does not change the representation of the a set $IS = \{m_1, \ldots, m_N\}$ (inserted into the Bloom filter), then we say it has a *steady representation*. The procedure of $G^{\mathsf{AR}}_{\mathcal{A},\mathsf{BF}}(\kappa, T, N, \epsilon)$ is defined in Figure 13. Meanwhile, we let Get_CurrentTime() be a public function to get the current system time. Note that we model a polynomial time adversary with explicit time parameter T that is similar to the approach [35] on restricting query number of adversaries.

Initialize(T, N, ϵ) :	**Finalize**(m^*) :
$\mathsf{BF.Init}(N, \epsilon)$	If $m^* \notin IS$ and $\mathsf{BF.Check}(m^*) = 1$
$IS \leftarrow \mathcal{A}(T, N, \epsilon)$	and $\mathsf{Get\_CurrentTime}() - T_s \leq T$
$\mathsf{BF.Insert}(m_i)$ for $\forall\, m_i \in IS$	OUTPUT 1
$T_s := \mathsf{Get\_CurrentTime}()$	ELSE OUTPUT 0
OUTPUT BF, T_s	

Fig. 13: Procedures used to define security for BF.

Definition 6. *Let* $\mathsf{Adv}^{\mathsf{AR}}_{\mathcal{A},\mathsf{BF}}(\kappa, T, N, \epsilon) := \Pr[G^{\mathsf{AR}}_{\mathcal{A},\mathsf{BF}}(\kappa, T, N, \epsilon) = 1]$ *be the advantage of a PPT adversary* $\mathcal{A}$ *in breaking the security of a Bloom filter* BF *under the security parameter* κ*. We say* BF *is secure if no PPT adversary has non-negligible advantage* $\mathsf{Adv}^{\mathsf{AR}}_{\mathcal{A},\mathsf{BF}}(\kappa, T, N, \epsilon)$*.*

Merkle Tree. We define a security game $G^{\mathsf{MT\_Forge}}_{\mathcal{A},\mathsf{MT}}(\kappa)$ in Figure 14 for a Merkle tree scheme MT, which encompasses the following procedures. For a secure Merkle tree scheme, the adversary must not forge the Merkle proof for a leaf node which does not belong to the Merkle tree.

$G^{\mathsf{MT\_Forge}}_{\mathcal{A},\mathsf{MT}}(\kappa)$:	
Initialize$(\{\mathrm{Lf}_i\}_{i\in[\ell]})$:	**Finalize**$(\mathrm{Lf}^*, \mathrm{Pf}^*)$:
$\mathrm{MTr} \leftarrow \mathsf{MT.Build}(\{\mathrm{Lf}_i\}_{i\in[\ell]})$	IF $1 \leftarrow \mathsf{MT.Verify}(\mathrm{Rt}, \mathrm{Lf}^*, \mathrm{Pf}^*)$
OUTPUT MTr:	and $\mathrm{Lf}^* \notin \{\mathrm{Lf}_i\}_{i\in[\ell]}$, OUTPUT 1
	OUTPUT 0

Fig. 14: Procedures Used to Define Security for MT.

Definition 7. *We say* MT *is secure if no PPT adversary has a non-negligible advantage* $\mathsf{Adv}^{\mathsf{MT\_Forge}}_{\mathcal{A},\mathsf{MT}}(\kappa) := \Pr[G^{\mathsf{MT\_Forge}}_{\mathcal{A},\mathsf{MT}}(\kappa) = 1]$ *in breaking the security of a Merkle tree scheme* MT *under* κ*.*

Privacy-preserving Location Proximity Schemes. In Figure 15, we define a game $G^{\mathsf{LP}}_{\mathcal{A},\mathsf{PPLP}}(\kappa, q_l)$ between a semi-honest adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The adversary could be the responder who is curious about the location of the checker. In the game, the challenger would provide the adversary with the following procedures to query, where the protocol execution procedure $\mathsf{Execute}$ can be asked at most q_l times.

$G^{\mathsf{LP}}_{\mathcal{A},\mathsf{PPLP}}(\kappa, q_l)$:

Initialize() :	**Finalize(b^*) :**
$\mathsf{pms} \leftarrow \mathsf{PPLP.Setup}(1^\kappa)$, $rk \xleftarrow{\$} \mathcal{K}_{\mathsf{PPLP}}$	IF $b^* = b$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PPLP.KGen}(rk, \mathsf{pms})$	OUTPUT 1
OUTPUT $\mathsf{pms}, \mathsf{pk}$	OUTPUT 0
Challenge($\mathrm{L}^*_{\mathsf{W}}$) :	**Execute($\mathrm{L}_{\mathsf{P}}, \mathrm{L}_{\mathsf{W}}$) :**
$b \xleftarrow{\$} \{0,1\}$	$C_{\mathsf{P}} \leftarrow \mathsf{PPLP.LPInit}(\mathsf{pk}, \mathrm{L}_{\mathsf{P}}, \theta)$
Sample $\mathrm{L}^*_{\mathsf{P}}$ s.t. $\mathsf{Dist}(\mathrm{L}^*_{\mathsf{P}}, \mathrm{L}^*_{\mathsf{W}}) < \theta$ if $b = 1$	$C_{\mathsf{W}} \leftarrow \mathsf{PPLP.LPResp}(\mathsf{pk}, C_{\mathsf{P}}, \mathrm{L}_{\mathsf{W}})$
and $\mathsf{Dist}(\mathrm{L}^*_{\mathsf{P}}, \mathrm{L}^*_{\mathsf{W}}) > \theta$ if $b = 0$	OUTPUT C_{W}
$C^*_{\mathsf{W}} \leftarrow \mathsf{Execute}(\mathrm{L}^*_{\mathsf{P}}, \mathrm{L}^*_{\mathsf{W}})$	
OUTPUT C^*_{W}	

Fig. 15: Procedures Used to Define Security for PPLP.

Definition 8. *We say* PPLP *is secure if no PPT adversary has a non-negligible advantage* $\mathsf{Adv}^{\mathsf{LP}}_{\mathcal{A},\mathsf{PPLP}}(\kappa, q_l) := \left|\Pr[G^{\mathsf{LP}}_{\mathcal{A},\mathsf{PPLP}}(\kappa, q_l) = 1] - \frac{1}{2}\right|$ *in breaking the security of a PPLP scheme* PPLP *under* κ.

C Security Model of PoL

Under semi-honest threat model, we aim to achieve the following two security properties for PoL. The first is *unforgeability* and *traceability* of location proofs. The second is *anonymity* for the prover and witnesses. The third is *location privacy* of witnesses. To formulate these security properties, we define three security games $\mathsf{PoL\_Unforge}$, $\mathsf{PoL\_Anony}$, and $\mathsf{PoL\_LocPriv}$, respectively. We let $\mathsf{Exp} \in \{\mathsf{PoL\_Unforge}, \mathsf{PoL\_Anony}, \mathsf{PoL\_LocPriv}\}$ be a variable to indicate one of the games.

We present the relevant procedures of the security games in Figure **??**. The procedures are adapted from PPLP and GTOTP models to fit in the PoL setting. The $\mathsf{Initialize}(\mathsf{GP})$ procedure is defined to simulate the *Registration* protocol with the given parameters, which returns the protocol execution transcript Tr_I and the group verification key $\mathsf{vk_G}$ to the adversary. If the game is not $\mathsf{PoL\_Anony}$, the secret key k_{RA} is also returned. We use a $\mathsf{LocProofGen}(\mathrm{L}_{\mathsf{P}}, \{\mathrm{L}_{\mathsf{W}_j}\}_{j\in[M]}, \mathsf{P}, \{\mathsf{W}_i\}_{i\in[M]})$ procedure to simulate the protocol execution of *Location-Proof-Gen* protocol, with which the adversary can specify the locations and participants for running the protocol, and get the protocol execution transcript Tr_L and the resultant location proof $\mathsf{LP_P}$, where $(\mathsf{P}, \{\mathsf{W}_i\}_{i\in[M]}) \in \mathsf{GP}$. The procedure $\mathsf{LocProofVerify}(\mathsf{LP_P}, \mathrm{L}_{\mathsf{P}})$ is defined to simulate the *Verification* protocol, which enables the adversary to test the validity of a (forged) location proof and get the protocol execution transcript Tr_V.

During the game, the adversary can ask once the $\mathsf{Challenge}$ query to challenge either anonymity or location-privacy of the PoL scheme. The challenger samples two random bits $(b_a, b_l) \xleftarrow{\$} \{0,1\}$ to test the capability of the adversary in the $\mathsf{PoL\_Anony}$ game and the $\mathsf{PoL\_LocPriv}$ game, respectively. The adversary can specify the initial identities of the prover P and the M witnesses $\{\mathsf{W}_i\}$ as well as their initial locations. But in the $\mathsf{PoL\_Anony}$ game, if $b_a = 1$, the challenger would replace the identity of either the prover or the first witness with another identity $\hat{\mathsf{ID}}_1$ depending on the role $role \in \{Prover, Witness\}$ of the challenged party specified by the adversary. Similarly, W_1's location is changed to be $\mathrm{L}^*_{\mathsf{W}_1}$ (provided by the adversary either) in the $\mathsf{PoL\_LocPriv}$ game if $b_l := 1$. The goal of the adversary is to distinguish whether or not the corresponding initial information of either P or W_1 is modified by the challenger in the run of the *Location-Proof-Gen* protocol. Meantime, we formulate the fault tolerance via two parameters ρ (s.t. $\rho \leq M$) and M: for a correct location proof, we only need ρ location valid proofs from the witness, thereby tolerating $M - \rho$ failures or inaccuracies.

Definition 9. *We say that a PoL scheme* PoL *is secure if the advantages* $\mathsf{Adv}^{\mathsf{PoL\_Anony}}_{\mathcal{A},\mathsf{PoL}}(\mathsf{pms}) := \left|\Pr\left[G^{\mathsf{PoL\_Anony}}_{\mathcal{A},\mathsf{PoL}}(\mathsf{pms}) = 1\right] - 1/2\right|$, $\mathsf{Adv}^{\mathsf{PoL\_Trace}}_{\mathcal{A},\mathsf{PoL}}(\mathsf{pms}) := \Pr\left[G^{\mathsf{PoL\_Trace}}_{\mathcal{A},\mathsf{PoL}}(\mathsf{pms}) = 1\right]$, *and* $\mathsf{Adv}^{\mathsf{PoL\_LocPriv}}_{\mathcal{A},\mathsf{PoL}}(\mathsf{pms}) := \left|\Pr\left[G^{\mathsf{PoL\_LocPriv}}_{\mathcal{A},\mathsf{PoL}}(\mathsf{pms}) = 1\right] - 1/2\right|$ *of any PPT adversary* $\mathcal{A}$ *in the corresponding games are negligible under given parameters* $\mathsf{pms} = (\kappa, U, T_s, T_e, \Delta_e, \Delta_s)$.

D Discussion on GTOTP Privacy

In this section, we discuss the achievable privacy for our GTOTP schemes. We stress that each verify point of a party (i.e., the verify point of a TOTP instance) in the GTOTP scheme can be seen a pseudonym used for a verify-epoch. As other pseudonym schemes (such as IFAL [44]), we assume that each verify point is used for a short period of time, e.g., 5 mins (such a time period for a pseudonym is recommended by European Telecommunications Standards Institute (ETSI) [21]). We do not consider the privacy leakage because of the user's behaviour (or password usage pattern) within the verify-epoch. Instead, we build GTOTP schemes to guarantee that the verify points cannot be linked to leak a party's real identity.

Since a Dedicated Short Range Communications (DSRC) or Bluetooth enabled device would periodically execute service discovery protocol by sending probe requests which includes the device's Media Access Control (MAC) address, so the attackers may exploit the MACs to trace the owner of the device. To prevent this kind of threat, each mobile device can use a local a randomized and continuously-refreshed MAC instead of its unique physical global MAC, which is possible in the recent version of either IOS or Android [43,25].

E Proof of Theorem 1

Let $\mathsf{BK}_i^{\text{anony}}$ denote an event that there exists an adversary wins in the i-th (modified) $\mathsf{GTOTP\_Anony}$ game.

Game 0. This game equals the real security game. Thus, we have that $\Pr[\mathsf{BK}_0^{\text{anony}}] = \mathsf{Adv}^{\mathsf{GTOTP\_Anony}}_{\mathcal{A},\mathsf{GTOTP}}(\kappa, U, T_s, T_e, \Delta_e, \Delta_s)$.

Game 1. We change this game from the previous game by letting the challenger abort if secret keys of two parties are identical (i.e., two $\mathsf{F.Setup}$ executions output the same key). If this abort event occurs with non-negligible probability, then we could break the security PRF. We assume that there is an algorithm $\mathcal{F}$ who tries to break the security of F in the PRF game. In this case, the PRF challenger may run $\mathsf{F.Setup}$ to get a challenge key k^*. Meanwhile, $\mathcal{F}$ may run the public algorithm $\mathsf{F.Setup}$ algorithms herself $U-1$ times to get keys $k_1, \ldots, k_{U-1}$. By our assumption, there are two keys (k'_1, k'_2) in the set $(k^*, k_1, \ldots, k_{U-1})$ are identical. Since it holds with a probability $2/U$ that either k'_1 or k'_2 equals to k^*, in which case, the adversary knows the k^* and then can break the security of PRF with the knowledge of k^*. Since the adversary only has a negligible probability $\mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A},\mathsf{F}}(\kappa, 1)$ to break the PRF by assumption, we have $\Pr[\mathsf{BK}_0^{\text{anony}}] \leq \Pr[\mathsf{BK}_1^{\text{anony}}] + \frac{U}{2} \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A},\mathsf{F}}(\kappa, 1)$.

Game 2. This game proceeds exactly like the previous game, but the challenger aborts if she fails to guess which two TOTP instances that the adversary chooses for the challenge. Hence, we have that $\Pr[\mathsf{BK}_1^{\text{anony}}] = 2U \cdot E \cdot \Pr[\mathsf{BK}_2^{\text{anony}}]$. As a result, the challenge knows the challenge TOTP instances in advance.

Game 3. In this game, we change the secret seed $sd_{\mathsf{ID}_i^*}$ in the challenge verify-epoch to be a truly random value. If there exists an adversary $\mathcal{A}$ that can distinguish this game from the previous game, then it can be used to build an efficient algorithm $\mathcal{B}$ to break the PRF security. $\mathcal{B}$ can run $\mathcal{A}$ as a subroutine and simulate the game for her. As for the secret seed $sd_{\mathsf{ID}_i^*}$, $\mathcal{B}$ can obtain it from the challenger which simulates the security game of PRF. Note that in the PRF game the challenger will provide an PRF oracle $\mathcal{O}_{\mathsf{PRF}}(m)$ to evaluate the function $\mathsf{F}(sd_{\mathsf{ID}_i^*}, m)$ except for the challenge message. So $\mathcal{B}$ can get all other secret seeds of ID_i^* by querying $\mathcal{O}_{\mathsf{PRF}}(\mathsf{ID}_i^* || j)$ for $j \in [E]$. Note that we compute identity-ciphertext $C^*_{\hat{\mathsf{ID}}_b}$ in the challenge password as $C^*_{\hat{\mathsf{ID}}_b} = \mathsf{ASE.Enc}(k_{\mathsf{RA}}, pw^*_{\hat{\mathsf{ID}}_b}, T^*)$, where $\hat{\mathsf{ID}}_b \in \{\mathsf{ID}_i^*, \mathsf{ID}_j^*\}$, $pw^*_{\hat{\mathsf{ID}}_b} := \mathsf{PwGen}(sd_{\hat{\mathsf{ID}}_b}, T^*)$, and T^* is the start time of the challenge verify-epoch. Note that the change is this game takes effective if and only if ID_i^* is selected as challenge party (which happens with probability 1/2). Thus we have $\Pr[\mathsf{BK}_2^{\text{anony}}] \leq \Pr[\mathsf{BK}_3^{\text{anony}}] + 2\mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A},\mathsf{F}}(\kappa, E)$.

Game 4. The challenger proceeds as before, but sets $sd_{\mathsf{ID}_j^*} = sd_{\mathsf{ID}_i^*}$. That is, the challenger always returns a random secret seed regardless of the bit b. Obliviously, distinguishing this game from the previous game implies an algorithm breaking the security of PRF, we thus have that $\Pr[\mathsf{BK}_3^{\mathsf{GTOTP\_Anony}}] \leq \Pr[\mathsf{BK}_4^{\mathsf{GTOTP\_Anony}}] + \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A},\mathsf{F}}(\kappa, E)$. The modification in this game implies that the adversary cannot gain any non-negligible advantage from the secret seeds.

Game 5. As the permutation is unpredictable, so adversaries cannot infer the order of verify-points in input on given the output of π with non-negligible advantage. After the adversary cannot infer the leaf nodes' owners (when they are not corrupted) from the order of verify-points (which are, therefore, just random values to the adversary. Moreover, each Merkle tree root is mapped to m bit positions (which are set to be '1') of the Bloom filter. And many roots may share the same bit positions. So it is impossible for an adversary to recover the relationship from two inserted roots from the resultant Bloom filter. Now, if the adversary can distinguish the bit b, she can only try to break the security of the authenticate key encryption scheme ASE. So we can build an algorithm $\mathcal{C}$ running $\mathcal{A}$ as a subroutine. As for the challenge identity-ciphertext $C^*_{\hat{\mathsf{ID}}_b}$ (in the challenge password), $\mathcal{C}$ can query the messages $(\hat{\mathsf{ID}}_0, \hat{\mathsf{ID}}_1)$ to the ASE challenger which should encrypts one to generate the challenge ciphertext. All other identity-ciphertexts can be obtained by calling an encryption oracle simulated by the ASE challenger. Note that $\mathcal{C}$ can simulate the rest of the values using the secret keys of her own choice. $\mathcal{B}$ forwards the bit b^* returned by $\mathcal{A}$ to the ASE challenger. Thus, we have that

$\Pr[\mathsf{BK}_4^{\mathsf{GTOTP\_Anony}}] \leq \Pr[\mathsf{BK}_5^{\mathsf{GTOTP\_Anony}}] + \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{A},\mathsf{ASE}}(\kappa, E)$. The modification in this game implies that the adversary cannot gain any non-negligible advantage from the secret seeds. The advantages in the above games only give the adversary a negligible overall advantage that proves Theorem 1.

F Proof of Theorem 2

Let $\mathsf{BK}_i^{\mathrm{trace}}$ denote an event that there exists an adversary wins in the i-th (modified) $\mathsf{GTOTP\_Trace}$ game.

Game 0. This game equals the real security game, and all queries are answered honestly according to our protocol specification. Thus, we have that $\Pr[\mathsf{BK}_0^{\mathrm{trace}}] = \mathsf{Adv}^{\mathsf{GTOTP\_Trace}}_{\mathcal{A},\mathsf{GTOTP}}(\kappa, U, T_s, T_e, \Delta_e, \Delta_s)$.

Game 1. The challenger proceeds as before, but aborts if the secret keys of two parties are identical. With the same argument in the Game 1 of the proof of Theorem 1, we have that $\Pr[\mathsf{BK}_0^{\mathrm{trace}}] \leq \Pr[\mathsf{BK}_1^{\mathrm{trace}}] + \frac{U}{2} \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A},\mathsf{F}}(\kappa, 1)$. So the challenge parties should have distinct secret keys.

Game 2. In this game, we add an abort rule that the challenger aborts if the adversary outputs a password which results in a verify-point vp^* such that $\mathsf{H}_1(vp^*||C_{\mathsf{ID}_j^*}||\beta) = \mathsf{H}_1(vp^{\alpha}_{\mathsf{ID}_j}||C^{\alpha}_{\mathsf{ID}_j}||\alpha)$ and ID_j is not corrupted, where $vp^{\alpha}_{\mathsf{ID}_j}$ is generated by the challenger, and β and α are arbitrary indices. I.e., the adversary finds a hash collision to an honest verify-point. If such an abort event occurs non-negligible, we could make use of the adversary to break the security of the collision-resistant hash function H_1. Thus, we have $\Pr[\mathsf{BK}_1^{\mathrm{trace}}] \leq \Pr[\mathsf{BK}_2^{\mathrm{trace}}] + \mathsf{Adv}^{\mathsf{CRHF}}_{\mathcal{A},\mathsf{H}_1}(\kappa)$.

Game 3. The challenger proceeds as before, but rejects any password resulting in a Merkle tree root which is not generated by herself. This abort rule excludes the case that the adversary successfully exploits the false positive error $2^{-\epsilon}$ of BF within the lifespan (i.e., $T_e - T_s$) of a $\mathsf{GTOTP\text{-}MT}$ instance. Note that by appropriately choosing the parameters of the Bloom filter we can make the $\mathsf{Adv}^{\mathsf{AR}}_{\mathcal{A},\mathsf{BF}}(\kappa, T, \phi, \epsilon)$ to be negligible. Thus, we have that $\Pr[\mathsf{BK}_2^{\mathrm{trace}}] \leq \Pr[\mathsf{BK}_3^{\mathrm{trace}}] + \mathsf{Adv}^{\mathsf{AR}}_{\mathcal{A},\mathsf{BF}}(\kappa, T, \phi, \epsilon)$.

Game 4. In this game we want to reduce the security to that the Merkle tree scheme. To realize this, We consider a Merkle-forge event that an adversary $\mathcal{A}$ outputs a password pw^* which can pass the verification process with a forged Merkle proof $\mathrm{Pf}_{\hat{vp}^*} \in pw^*$. I.e., $\mathrm{Pf}_{\hat{vp}^*}$ is not computed by the challenger during the execution of $\mathsf{GVSTGen}$, where $\hat{vp}^*$ is the verify-point generated based on pw^*. We change this game from the previous game by letting the challenger abort when the Merkle-forge event occurs. If this abort event occurs with non-negligible probability, then we can make use of $\mathcal{A}$ to break the security of the Merkle tree scheme. Thus, we have that $\Pr[\mathsf{BK}_3^{\mathrm{trace}}] \leq \Pr[\mathsf{BK}_4^{\mathrm{trace}}] + \mathsf{Adv}^{\mathsf{MT\_Forge}}_{\mathcal{A},\mathsf{MT}}(\kappa)$.

As the adversary cannot forge the Merkle proof in this game, then she cannot bind a password with either an identity-ciphertext which is not the owner of the password or an invalid identity-ciphertext. In the sequel, we will show that the adversary cannot forge the other values of a password.

Game 5. This game proceeds exactly like the previous game, but the challenger aborts if she fails to guess which TOTP instance that the adversary can break. Since there are $U \cdot E$ TOTP instances at all, the probability of a correct guess is at least $\frac{1}{U \cdot E}$. Thus, we have that $\Pr[\mathsf{BK}_4^{\mathrm{trace}}] = U \cdot E \cdot \Pr[\mathsf{BK}_5^{\mathrm{trace}}]$. We assume that the guessed instance is the i^*-th TOTP instance of the party ID_j without loss of generality.

Game 6. In this game, we replace the secret seed $sd^{i^*}_{\mathsf{ID}_j} = \mathsf{F}(k_{\mathsf{ID}_j}, \mathsf{ID}_j||i^*)$ with a random value for the guessed i^*-th TOTP instance of the party ID_j. If there exists an adversary $\mathcal{A}$ who can distinguish this game from the previous game with non-negligible advantage, then we can build an algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to break the security of PRF. Specifically, $\mathcal{B}$ can submit a challenge query with a message $(\mathsf{ID}_j||i^*)$ to the PRF challenger to get back the corresponding challenge value r_b which is a value of $\mathsf{PRF}(k^*, \mathsf{ID}_j||i^*)$ or a truly random value, where k^* is a secret key chosen by the PRF challenger. Then $\mathcal{B}$ and sets $sd^{i^*}_{\mathsf{ID}_j} := r_b$.

The corresponding passwords of that instance are computed based on $sd^{i^*}_{\mathsf{ID}_j}$. For other secret seeds of ID_j, can query the PRF oracle $\mathcal{O}_{\mathsf{PRF}}(\mathsf{ID}_j||\iota)$ for $\iota \in [E] \backslash i^*$. $\mathcal{B}$ can choose all other secrets to simulate the game for $\mathcal{A}$ (including answering the $\mathsf{CompromiseSD}(\mathsf{ID}_j)$ queries of $\mathcal{A}$). Note that if r_b is the real value, then the simulated game is identical to the previous game; otherwise, it is identical to this game. Therefore, the capability of $\mathcal{A}$ on distinguishing between the games implies the breach of PRF. Due to the security of PRF, we have that $\Pr[\mathsf{BK}_5^{\mathrm{trace}}] \leq \Pr[\mathsf{BK}_6^{\mathrm{trace}}] + \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A},\mathsf{F}}(\kappa, E)$.

Game 7. The challenger proceeds as before, but aborts if an adversary $\mathcal{A}$ outputs a valid password $\bar{pw}^*$ of a TOTP instance at some time T^* such that T^* is greater than all previously opened password values. Obviously, such impersonation attempt $(\bar{pw}^*, T^*)$ can be used to break the guessed TOTP instance. We can build an algorithm $\mathcal{F}$ (as in the previous game) by running $\mathcal{A}$ as a sub-routine to break the TOTP scheme. Note that $\mathcal{F}$ can resort to the TOTP challenger to simulate the passwords of the guessed TOTP instance before time T^*. All other TOTP instances can be simulated based on the $\mathcal{F}$'s own secrets. Since the TOTP scheme is secure by assumption, we have that $\Pr[\mathsf{BK}_6^{\mathrm{trace}}] \leq \Pr[\mathsf{BK}_7^{\mathrm{trace}}] + \mathsf{Adv}^{\mathsf{TOTP\_Forge}}_{\mathcal{A},\mathsf{TOTP}}(\kappa, T_s, T_e, \Delta_s)$.

In this game, adversaries cannot use a password resulting in a verify-point that is not generated by any honest party to win the security game. So the advantage of this game is 0 as well, which concludes this proof.

G Proof of Theorem 3

In the following, we show the proof in a sequence of games following [41]. Let $\mathsf{BK}_i^{\text{anony}}$ denote an event that there exists an forgery-adversary wins in Game i.

Game 0. This game equals the real security experiment, and we have that $\Pr[\mathsf{BK}_0^{\text{anony}}] = \mathsf{Adv}_{\mathcal{A},\mathsf{PoL}}^{\mathsf{PoL\_Anony}}$.

Game 1. This game proceeds exactly like the previous game, but the challenger aborts if it fails to guess the following cases: (i) the identities $\hat{\mathsf{ID}}_0$ and $\hat{\mathsf{ID}}_1$ in the call of $\mathsf{Challenge}$ procedure; (ii) the verify-epoch being challenged. If $\mathcal{B}$ finds out that she fails to guess one of the above cases during the game, then she would abort the game with failure. The probability on a correct guess is bounded by $\frac{1}{U^2 \cdot E}$. Thus we have that $\Pr[\mathsf{BK}_0^{\text{anony}}] = U \cdot E \cdot \Pr[\mathsf{BK}_1^{\text{anony}}]$.

Game 2. In this game, we will show that if there exists an adversary $\mathcal{A}$ which can break the anonymity of PoL then we can use it to construct an efficient algorithm $\mathcal{B}$ to break the anonymity of the underlying GTOTP scheme GTOTP. Specifically, $\mathcal{B}$ simulates the game for $\mathcal{A}$. In the simulation, $\mathcal{B}$ can corrupt the parties which are not $\hat{\mathsf{ID}}_0$ and $\hat{\mathsf{ID}}_1$, and compromise all secret seeds which are not used for the challenge verify-epoch in the GTOTP game, so that $\mathcal{B}$ can faithfully simulate the $\mathsf{PoL\_Anony}$ game with all necessary secrets. Meanwhile, $\mathcal{B}$ would ask the $\mathsf{GTOTP\_Anony}$ challenge procedure $\mathsf{Challenge}(\hat{\mathsf{ID}}_0, \hat{\mathsf{ID}}_1)$ to get the secret seed $sd^*_{\hat{\mathsf{ID}}_b}$, and use $sd^*_{\hat{\mathsf{ID}}_b}$ to simulate the challenge procedure $\mathsf{Challenge}$ asked by $\mathcal{A}$. Eventually, $\mathcal{B}$ will return the bit b' obtained from $\mathcal{A}$ as its output in the $\mathsf{GTOTP\_Anony}$ game. Hence, if $\mathcal{A}$ can win the $\mathsf{GTOTP\_Anony}$ game with non-negligible advantage, so can $\mathcal{B}$ in the $\mathsf{PoL\_Anony}$ game. Therefore, $\Pr[\mathsf{BK}_1^{\text{anony}}] = \Pr[\mathsf{BK}_2^{\text{anony}}] = \mathsf{Adv}_{\mathcal{A},\mathsf{GTOTP}}^{\mathsf{GTOTP\_Anony}}$. Putting together the advantages of $\mathcal{A}$ in the above games concludes the proof.

H Proof of Theorem 4

Let $\mathsf{BK}_i^{\text{trace}}$ denote an event that there exists an adversary wins in Game i.

Game 0. This game equals the real security experiment, and we have that $\Pr[\mathsf{BK}_0^{\text{trace}}] = \mathsf{Adv}_{\mathcal{A},\mathsf{PoL}}^{\mathsf{PoL\_Trace}}$.

Game 1. This game proceeds exactly like the previous game, but the challenger aborts if it fails to guess the following cases: (i) the victim party ID^* that the adversary can break; (ii) the index i^* of the secret seed for generating the key of the corresponding commitment that the adversary can forge. Since there are $U \cdot E$ TOTP instances at all, the probability of a correct guess is at least $\frac{1}{U \cdot E}$. Thus, we have that $\Pr[\mathsf{BK}_0^{\text{trace}}] = U \cdot E \cdot \Pr[\mathsf{BK}_1^{\text{trace}}]$.

Game 2. The challenger proceeds exactly like the previous game, but aborts if the adversary $\mathcal{A}$ queries either $ck_{\mathsf{ID}^*}^{i^*,z}$ or $sd_{\mathsf{ID}^*}^{i^*}$ in a random oracle query before it is opened by the corresponding party, where (i^*, ID^*) are the information guessed in the previous game. Note that the outputs of random oracle are unique and random. Therefore, this abort rule means that the adversary successfully forges a commitment of an uncorrupted party. If this abort event happens with non-negligible probability, then we can build an algorithm $\mathcal{B}$ by using the adversary $\mathcal{A}$ to break the security of GTOTP. Specifically, $\mathcal{B}$ could simulate the game for $\mathcal{A}$ by answering her oracle queries as in the previous game. $\mathcal{B}$ can ask the $\mathsf{Corrupt}$ or $\mathsf{CompromiseSD}$ procedure to obtain the corresponding secrets for computing all other commitments that the adversary does not forge. When $\mathcal{B}$ needs to simulate the values that are generated by $\mathsf{H}_2(k^*||\mathrm{Tr}^*)$ involving $k^* \in \{ck_{\mathsf{ID}^*}^{i^*,z}, sd_{\mathsf{ID}^*}^{i^*}\}$ and $\mathrm{Tr}^* \in \{\mathrm{Tr}_{\mathsf{W}_j}, \mathrm{Tr}_{\mathsf{P}}\}$ in $\mathsf{LocProofGen}$ queries, $\mathcal{B}$ can just use a random function $\mathsf{RF}(\mathrm{Tr}^*)$ instead. This does not change the distribution of the generated location proof. Since $\mathcal{A}$ will query $\mathsf{H}_2(k^*||\cdot)$ with non-negligible probability by assumption, $\mathcal{B}$ can get the target k^* which could enable her to break the GTOTP. Note that $\mathcal{B}$ can check whether k^* queried by $\mathcal{A}$ is a true secret by using the public verify-point of the i^*-th verify-epoch. By applying the traceability of GTOTP, we have that $\Pr[\mathsf{BK}_1^{\text{trace}}] \leq \Pr[\mathsf{BK}_2^{\text{trace}}] + \mathsf{Adv}_{\mathcal{A},\mathsf{GTOTP}}^{\mathsf{GTOTP\_Trace}}(\kappa, U, T_s, T_e, \Delta_e, \Delta_s)$.

Game 3. The challenger proceeds exactly like the previous game, but rejects any location proof pieces which are not generated by herself. Let $l_h = |\mathcal{R}_h|$ be the bit-length of the range of the hash value of H_2. Since the adversary does not know the secrets for generating the commitment due to the previous game, the adversary can only randomly guess the valid value of the commitments with a successful probability $q_h/2^{l_h}$. Hence we have $\Pr[\mathsf{BK}_2^{\text{trace}}] \leq \Pr[\mathsf{BK}_3^{\text{trace}}] + q_h/2^{l_h}$. Because the adversary cannot win this game, the advantage of the adversary in this game is zero. This concludes the proof.

I Proof of Theorem 5

Let $\mathsf{BK}_i^{\text{locpriv}}$ denote an event that there exists an adversary wins in Game i.

Game 0. This game equals the real security experiment, and we have that $\Pr[\mathsf{BK}_0^{\text{locpriv}}] = \mathsf{Adv}_{\mathcal{A},\mathsf{PoL}}^{\mathsf{PoL\_LocPriv}}$.

Game 1. This game proceeds exactly like the previous game, but the challenger aborts if it fails to guess the following cases: (i) the victim party ID^* that the adversary can break; (ii) the counter cnt^* of the ephemeral key $\mathsf{epk}_{\mathsf{ID}^*}^{cnt^*}$ used in the challenge query. Since there are U parties, and each party can run at most $E \cdot N$ times

location proof generation protocol, the probability of a correct guess is at least $\frac{1}{U \cdot E \cdot N}$. Thus, we have that $\Pr[\mathsf{BK}_0^{\text{locpriv}}] = U \cdot E \cdot N \cdot \Pr[\mathsf{BK}_1^{\text{locpriv}}]$.

Game 2. This game is proceeded with exactly as the previous game, but the challenger sets the random seed $rk_{cnt^*} = \mathsf{F.Eval}(sk^2_{\mathsf{ID}^*}, \text{`PPLP'}||cnt^*)$ with a truly random value. We can reduct the security to that of PRF following the similar arguments of the Game 6 in the proof of Theorem 2. We thus have that $\Pr[\mathsf{BK}_1^{\text{locpriv}}] \leq \Pr[\mathsf{BK}_2^{\text{locpriv}}] + \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A},\mathsf{F}}(\kappa, 1)$.

Game 3. Now we reduce the security of PoL to that of PPLP. If there exists an adversary which can distinguish the bit b_l with non-negligible advantage in this game, then we can build an efficient algorithm $\mathcal{B}$ to break the security of PPLP scheme. In the simulation, $\mathcal{B}$ should first guess one uncorrupted witness W_1 queried in the challenge procedure in this game. The probability for a correct guess is bound by $\frac{1}{U}$. Based on such a correct guess, $\mathcal{B}$ ask a challenge query to the PPLP challenger using the location L_1 queried in the challenge procedure in this game. Then $\mathcal{B}$ will use the transcript obtained from the PPLP challenger to simulate the protocol transcript between P and W_1 in this PoL_LocPriv game. All other protocol steps are simulated based on the secrets chosen by $\mathcal{B}$. Eventually, $\mathcal{B}$ will return the bit b' obtained from $\mathcal{A}$ as its output in the PPLP game. Hence, if $\mathcal{A}$ can win the PPLP game with non-negligible advantage, so can $\mathcal{B}$ in the PoL_LocPriv game. Thus we have $\Pr[\mathsf{BK}_2^{\text{locpriv}}] = \Pr[\mathsf{BK}_3^{\text{locpriv}}] = U \cdot \mathsf{Adv}^{\mathsf{LP}}_{\mathcal{A},\mathsf{PPLP}}$. This concludes the proof.