Threshold Signatures With Verifiably Timed Combining and Message-Dependent Tracing

Meng Li, *Senior Member, IEEE*, Hanni Ding, Yifei Chen, *Graduate Student Member, IEEE*, Yan Qiao, *Member, IEEE*, Zijian Zhang, *Senior Member, IEEE*, Liehuang Zhu, *Senior Member, IEEE*, and Mauro Conti, *Fellow, IEEE*

***Abstract*—Threshold Signature (TS) is one of the fundamental cryptographic primitives adopted in many practical applications. Current Threshold, Accountable, and Private Signature (TAPS) schemes suffer from delayed combining, unverifiable combining, and message-independent tracing. More precisely, a malicious combiner may delay the combination of signature shares and replace some signature shares from honest signers with ones from colluding signers, and an unrestricted tracer can reveal signers' identities arbitrarily. In this work, we introduce a new scheme called TiMTAPS under a stronger security model. First, we sew homomorphic time-lock puzzles into the Schnorr signature, allowing puzzles to be combined and opened as needed. Second, we knit the Schnorr signature with homomorphic commitment for verifiable combining. Third, we infuse the combining phase with an identity-based key encapsulation mechanism for message-dependent tracing. Next, formalize the definitions and requirements for TiMTAPS. Then, we present a concrete construction and formally prove its privacy and security. We build a prototype of TiMTAPS based on Ethereum. Results from extensive experiments exhibit its practicability and efficiency, e.g., combining (tracking) 10 signature sets with a threshold value of 5 requires only 3.72 s (12.44 s) for the threshold signature.**

***Index Terms*—Threshold signature, security, privacy, timed cryptography, commitment, key encapsulation.**

Received 26 March 2025; revised 26 July 2025; accepted 19 August 2025. Date of publication 8 September 2025; date of current version 16 September 2025. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62372149, Grant U23A20303, and Grant 62572168; in part by the Anhui Provincial Natural Science Foundation under Grant 2508085MF151; in part by the Key Laboratory of Knowledge Engineering with Big Data (the Ministry of Education of China), under Grant BigKEOpen2025-04. The associate editor coordinating the review of this article and approving it for publication was Dr. Tianwei Zhang. *(Corresponding authors: Yan Qiao; Zijian Zhang.)*

Meng Li, Hanni Ding, Yifei Chen, and Yan Qiao are with the Key Laboratory of Knowledge Engineering with Big Data, Ministry of Education, the School of Computer Science and Information Engineering, and the Intelligent Interconnected Systems Laboratory of Anhui Province, Hefei University of Technology, Hefei 230002, China (e-mail: mengli@hfut.edu.cn; hanniding@mail.hfut.edu.cn; yifeichen@mail.hfut.edu.cn; qiaoyan@hfut.edu.cn).

Zijian Zhang is with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China, and also with the Southeast Institute of Information Technology, Beijing Institute of Technology, Putian, Fujian 351100, China (e-mail: zhangzijian@bit.edu.cn).

Liehuang Zhu is with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: liehuangz@bit.edu.cn).

Mauro Conti is with the Department of Mathematics and the HIT Center, University of Padua, 35131 Padua, Italy, and also with Örebro University, 701 82 Örebro, Sweden (e-mail: mauro.conti@unipd.it).

Digital Object Identifier 10.1109/TIFS.2025.3607250

I. Introduction

A. Background

THRESHOLD Signature (TS) [1] is one of the most fundamental cryptographic building blocks. It is an important branch of digital signatures that emerged under the catalysis of group decision-making demands and security requirements. TS distributes the key to n independent signers, and any t signers can cooperate to provide a valid signature. In a nutshell, a TS scheme is a n-to-1 protocol that enables each signer S_i in a group of signers $\mathcal{S} = \{S_1, S_2, \cdots, S_n\}$ to sign a message m with a private key sk_i and send a signature share δ_i to a combiner $\mathcal{C}$. Later on, $\mathcal{C}$ can use t signature shares to produce a complete signature σ on m with regards to a public key pk. TS has been playing an important role for extensive applications [2], [3], [4].

With the increasing diversification of practical application requirements, TSs have evolved into various specialized variants to meet specific needs [5], [6], [7], [8], [9]. Among these, the most representative ones are Private Threshold Signatures (PTS) and Accountable Threshold Signatures (ATS). In PTS, the scheme is designed to further enhance the privacy of participants. Specifically, the signature on a message reveals nothing about t or the quorum of t original signers [10], [11]. In contrast, ATS focus on ensuring the accountability of the signature. A tracing algorithm can identify the original signing group that generated a signature [12], [13]. Both schemes have their own advantages. It is also worth the effort to achieve smooth integration of PTS and ATS.

B. Existing Work

The seminal paper in this line of research is the Threshold, Accountable, and Private Signature (TAPS) [14] proposed by Boneh and Komlo in 2022. It is the first to combine the two advantages of ATS and PTS, featuring both privacy and accountability. A TAPS signature remains private from the public while a tracer with a secret tracing key is able to trace a signature back to the original signers. There are two schemes following a similar blueprint. Threshold, Accountable, and Private Signature with Hidden Witnesses (HiTAPS) [15] noticed the need to limit the capabilities of tracers and added witnesses to address this issue. HiTAPS also includes eyewitness tracing. It designates a set of witnesses during the signing phase and initiates the tracing activity secretly. Decentralized, Threshold,

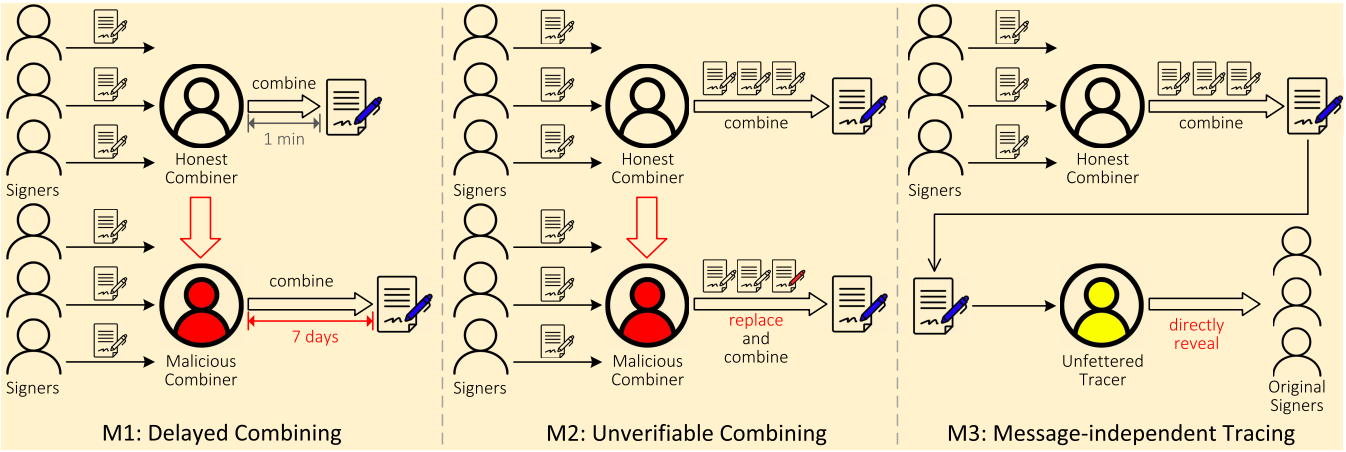


Fig. 1. Three Motivations behind This Work.

dynamically Accountable, and Private Signature (DeTAPS) [16] has simultaneously noted issues such as single point of failure and untrusted combiners/trackers, providing decentralized combining/tracing, enhanced privacy against untrusted combiners/tracers, and notarized and dynamic tracing.

C. Motivations and Goals

In this work, we explore a stronger security model where the combiner can misbehave during signature share combination. Specifically, there are three motivations, namely timed combining, verifiable combining, and message-independent tracing, which are depicted in Fig. 1. **M1. Delayed Combining.** A malicious combiner may choose to delay combining signature shares, aborting the combination process. This causes serious consequences in multiple applications. For instance, during a collaboration among t companies, each company signs an agreement that will come into effect in a period of time T, i.e., the t signature shares generated by t company representatives will be combined after time T. However, if the combiner does not stick to the plan, the complete signature will not be produced, thus forcing the agreement to stay void. To solve this problem, the Time-Lock Puzzle (TLP) [17], [18] came into our sight. TLP is a cryptographic puzzle that can be forcefully solved in time 2^{t_0}, but cannot be solved "significantly faster" by using parallel computers [19]. TLP enables a message to be encrypted and then decrypted in the near future. It has been extensively adopted in time-release cryptography [20], [21], [22], [23]. Such a feature enlightens us to channel TLP into TS to defend against the combiner's reluctance to perform their duties, ensuring **Timed Combining**. **M2. Unverifiable Combining.** Existing TSs are not required to be publicly verifiable and in all known constructions, which results in a collusion attack against honest signers. In TAPS [14], say each signer in a signer set $\mathcal{S}_1 = \{S_1, S_2, S_3\}$ sends a signature share to the combiner $\mathcal{C}$ who colludes with $\mathcal{S}_2 = \{S_1, S_2, S_4\}$. The malicious C can ask $\mathcal{S}_2$ to prepare three new signature shares to replace the ones of $\mathcal{S}_1$ and generate a new complete signature. Following TLP, it naturally brings to mind a recent work [24] that introduces the notion of Verifiable Timed Signature (VTS). Verifiability in VTS means that anyone can check if a time-lock contains a valid signature σ on the message without solving it first and that the σ can be obtained by solving the same for time T [24]. However, we notice that verifiability in timed combining embodies something unique. Each TS combination should be verifiable, i.e., anyone can publicly verify that the complete signature is generated by combining the t signature shares from the original t signers after the signature shares are "encrypted" into puzzles, ensuring **Verifiable Combining**. **M3. Message-Independent Tracing.** Traceability is a powerful feature for any ATS that is performed by a privileged tracer. Such a role can reveal the original signers from a complete signature without any restriction. In a sense, the tracer's power is unfairly strong [25], [26], which calls for some kind of restriction. This is also evidenced by previous works [15], [16], [25], [26], [27], [28]. Further, existing TAPS schemes fall short of mentioning the potential of signed messages during the tracing phase. Namely, the tracing can be triggered by specifying messages of which signatures should be traced, which is more functionally convenient and ensures **message-dependent tracing**.

D. Our Approach and Technical Challenges

Driven by the two motivations above, we propose a novel approach for TAPS to achieve timed combining and verifiable combining as follows. First, each signer conceals their signature share in a TLP towards timed combining. Second, each signer generates a commitment-based proof towards verifiable combining. Next, we deploy a blockchain [29], [30], [31] to combine signature share and facilitate public verification of complete signatures. Although the approach, we are faced with two new technical challenges.

1) TC1. Time-Locking TS Shares While Batching Puzzle Solving of Multiple TLPs: We first time-lock each TS share into a puzzle. This requires delicate integration of two techniques when treating a TS share as the solution s in the TLP since their values are drawn from different groups. After that, opening t puzzles one by one will not only incur high computation time but also delay the opening and combining (if only one opening entity is employed). Meanwhile,

each TS share consist of two parts where one is combined through addition and the other one is multiplication, thus necessitating corresponding puzzle homomorphism for batch solving. Therefore, Time-locking TS shares while batching the puzzle solving of t timed puzzled generated by t signers is challenging.

2) TC2. Time-Locking TS Shares While Proving the Authenticity of the Generated TS: After the TS shares are time-locked and sent to the untrusted combiner, the honest signers lose control of their shares. Later on, the combiner releases the complete TS signature σ upon share combination. Now we seek a public verifier to check that σ is indeed combined from the time-locked TS shares of the original t signers. The verification requires the signers to generate a proof (e.g., a commitment) that converges to echo σ. Meanwhile, such a commitment should also support additive homomorphism and multiplicative homomorphism due to the underlying operations. Therefore, time-locking TS shares while proving the authenticity of the generated TS is challenging.

3) TC3. Tracing TS Signatures From Messages Efficiently: In previous works, a tracer is constrained by either an admitter or a group of witnesses. At a first glance, it may appear that we can directly borrow their techniques as an enhancement for TAPS. However, the admitter was originally designed in message-dependent opening for Group Signatures (GSs) [25], [26], which structurally differs from TSs. Besides, asking signers to calculate a token and an encapsulation as in [25] is costly. In the witness-based approaches [15], [16], the utilization of Dynamic Threshold Public-Key Encryption (DTPKE) [33] and Key-Aggregate Searchable Encryption (KASE) [34] brought too many heavy cryptographic operations to the system [16], which we will provide details in Section III. Therefore, tracing TS signatures from messages efficiently is challenging.

E. Our Contributions

To tackle the three technical challenges, we propose a novel scheme TiMTAPS that facilitates timed combining, verifiable combining, and message-dependent tracing.

For TC1, we design a TAPS protocol with timed combining by leveraging the Schnorr signature the Homomorphic Time-Lock Puzzle (HTLP) [35] and a smart contract [36], [37]. Specifically, we use linearly homomorphic TLP to time-lock the z_i in a Schnorr signature, which is computed into a sum z by the combiner. Instead of using one combiner as a centralized manipulator, we deploy distributed combiners to retrieve TS shares from a Combining Smart Contract (CSC). It aggregates each time-locked z_i, i.e., tlp_i, to obtain a TLP of z and opens it after time T. Similarly, we use multiplicatively homomorphic TLP to process the R_i in a Schnorr signature and obtain R. If an operating combiner procrastinates, the time-locked shares can be solved only after time T.

For TC2, we enhance the protocol above to support public verifiability by utilizing Homomorphic Commitment (HC) [32] and a smart contract. All signers generate a commitment cmt_i of TS share and send it to V. V receives a commitment cmt of σ from the combiner and retrieves all HCs from the CSC. To verify that all the signature shares are indeed included in the combination, V aggregates the HCs to obtain a commitment of σ and compares it with cmt. If they are equal, the verification passes. Note that σ is encrypted throughout the process.

For TC3, we further improve the obtained protocol to support message-dependent tracing by employing Identity-Based Encryption (IBE) [39]. Instead of asking the combiner to encrypt σ as in [14], we encrypt σ into c_z^m by using m and an identity encryption algorithm IEnc. By doing so, we manage to maintain the computational cost while making the tracing associated with m. Given that the combination is shifted to an on-chain procedure, we move the encryption off-chain and require the combiner to generate a Non-Interactive Zero Knowledge (NIZK) proof [40] π for proving execution integrity. Next, we ask an admitter to extract a decapsulation key dk corresponding to m by using an extraction algorithm IExt. Finally, the tracer decrypts c_z^m to obtain σ by using IDec and reveals the signers' identities.

Our contributions are summarized as follows.

- We design the first TAPS scheme that facilitates timed combining of signature shares, verifiable combining of original signature shares, and message-dependent tracing of TS signatures.
- We formally define and prove the privacy and security of TiMTAPS.
- We build a prototype of TiMTAPS. We conduct extensive experiments to evaluate its performance and compare with existing work.

II. Related Work

A. TAPS

TAPS [14] builds upon an ATS scheme, a commitment scheme, a signature scheme, and a public-key encryption scheme. Its technical crux is to secure the public key pk' of the ATS scheme into a commitment $\mathsf{com}_{pk'}$ and encrypt the complete signature σ into c, after the combination of shares, while proving to the public that c is calculated from σ, σ is valid, and pk' is stored in $\mathsf{com}_{pk'}$. TAPS entrusts a tracing key to a trusted tracer that offers accountability in case of a dispute. By doing so, the threshold t is kept a secret from the public and only known to the signers, combiner, and tracer. TAPS also defines the precise syntax and the security requirements for a TAPS scheme, which paves the way for subsequent efforts.

HiTAPS [15] proposes witnessed tracing for TAPS to contain the power of tracing signatures in a box. During the signing phase, it nominates a set of witnesses to approve the tracing activity. Specifically, the combiner has to encrypt the complete signature σ using DTPKE [33] and public-key encryption. It computes a keyed-hash tag as a foreshadowing for the initial witnesses to initiate the tracing activity secretly. Next, the combiner generates a proof that proves to the validity of σ. HiTAPS also has an optimized protocol HiTAPS2 that reduces the combiner's communication overhead by leveraging the homomorphism of ElGamal encryption [41].

DeTAPS [16] put forth the notion of securing threshold t and witness threshold t' from the untrusted combiner. This is done by deploying the Trusted Execution Environment (TEE) on the combiner. In spite of the new feature, the increased costs and

TABLE I
THE COMPARISON WITH EXISTING WORK

Scheme	Combiner Model	Tracer Model	Privacy[1]	Unforgeability	Accountability	Timed Combining	Verifiable Combining	Message-Dependent Tracing
TAPS [14]	Centralized Honest	Centralized Honest	√	√	√			
HiTAPS [15]	Centralized Honest	Centralized Honest	√	√	√			
DeTAPS [16]	Decentralized Semi-honest[2]	Decentralized Honest	√	√	√			
TiMTAPS	Decentralized Malicious[3]	Decentralized Malicious[4]	√	√	√	√	√	√

1: Privacy consists of privacy against the public and privacy against the signers. 2: Curious about the privacy of signers but do not deviate from the protocol. 3: Can delay combining and replace signature shares. 4: Can trace signers from signatures arbitrarily.

potential vulnerabilities of TEE [42], [43], [44], [45] make the whole system clumsy. Besides, DeTAPS makes use of DTPKE to notarize the tracing process, designs non-interactive zero knowledge proofs to achieve public verifiability of witnesses, and leverages Key-Aggregate Searchable Encryption (KASE) to integrate TAPS and DTPKE to notify the notaries in a secure manner. Note that there is a difference between the security assumption of the combiner in DeTAPS and the one in TiMTAPS given that both of them adopt a malicious model for the combiner. The combiner in DeTAPS is assumed to be untrusted without any malicious actions while the one in TiMTAPS can misbehave.

In contrast to the previous TAPS schemes, TiMTAPS's promotion over them is twofold. First, TiMTAPS tackles the problem of a procrastinator during combining. Each combination of signature shares is guaranteed to complete in time. Second, TiMTAPS solves the problem of a manipulator during combining. Each combination of signature shares is promised to calculate from the signature shares of the original signers. Last, TiMTAPS addresses the issue of tracing while neglecting the underlying message. In Table I, we compare TiMTAPS with existing work regarding seven features including combiner model, tracer model, privacy, unforgeability, accountability, timed combining, verifiability, message-dependent tracing.

III. PRELIMINARIES

A. ATS

An ATS scheme Δ consists of five algorithms [14]:

$\mathsf{AKGen}(1^\lambda, n, t) \rightarrow (pp^{\mathrm{ATS}}, \{sk_i\}_{i=1}^n, pk)$: given a security parameter λ, a number of signers n, and a threshold t, outputs public parameters pp^{ATS}, a set of private keys $\{sk_i\}_{i=1}^n$ and a public key pk.

$\mathsf{ASign}(sk_i, m, \mathcal{S}) \rightarrow \sigma_i$: given a secret key sk_i of signer S_i belonging to a signing group $\mathcal{S} \subseteq [n]$, and a message m, outputs a signature share σ_i, which interacts with $\mathcal{S}$.

$\mathsf{ACombine}(pk, m, \mathcal{S}, \{ss_j\}_{S_j \in \mathcal{S}}) \rightarrow \sigma$: given a public key pk, message m, a signing group $\mathcal{S}$, and signature shares $\{ss_j\}_{S_j \in \mathcal{S}}$, outputs a signature $\sigma = (R, z, \mathcal{S})$.

$\mathsf{AVerify}(pk, m, \sigma = (R, z, \mathcal{S})) \rightarrow \{0, 1\}$: given a public key pk, a message m, and a combined signature σ, outputs 1 (valid) if $|\mathcal{S}| = t$ and the Schnorr verification algorithm accepts the triple $(pk_{\mathcal{S}}, m, \sigma')$ where $pk_{\mathcal{S}} = \prod_{S_j \in \mathcal{S}} pk_j$ and $\sigma' = (R, z, \mathcal{S})$; otherwise outputs 0 (invalid).

$\mathsf{ATrace}(pk, m, \sigma) \rightarrow \{\mathcal{S}, \perp\}$: given a public key pk, a message m, and a complete signature σ, invokes $\mathsf{AVerify}(pk, m, \sigma)$, and outputs $\mathcal{S}$ or $\perp$ to indicate failure.

B. HTLP

An HTLP scheme Λ consists of four algorithms [35]:

$\mathsf{PSetup}(1^\lambda, T) \rightarrow pp^{\mathrm{HTLP}}$: given a security parameter λ and a time hardness parameter T, outputs public parameters pp^{HTLP}.

$\mathsf{PGen}^{\mathrm{add/mul}}(pp, s) \rightarrow Z$: given public parameters pp and a solution s, outputs an additive/multiplicative puzzle Z.

$\mathsf{PSolve}^{\mathrm{add/mul}}(pp, Z) \rightarrow s$: given public parameters pp and an additive/multiplicative puzzle Z, outputs solution s.

$\mathsf{PEval}^{\mathrm{add/mul}}(C, pp, Z_1, \cdots, Z_n) \rightarrow Z$: given a circuit $C \in \mathcal{C}_\lambda$, public parameters pp and a set of n puzzle $(Z_1, \cdots, Z_n)$, outputs an additive/multiplicative puzzle Z.

C. HC

An HC scheme Ω consists of four algorithms [32]:

$\mathsf{CSetup}(1^\lambda) \rightarrow pp^{\mathrm{HC}}$: given a security parameter λ, outputs public parameters pp^{HC}.

$\mathsf{CCom}(s, r) \rightarrow cmt$: given a string s and a random number r, outputs a commitment cmt.

$\mathsf{CAdd}(cmt_0, cmt_1) \rightarrow cmt$: given two commitments cmt_0 and cmt_1, outputs a new commitment cmt.

$\mathsf{COpen}(cmt) \rightarrow \{m, \perp\}$: given a commitment cmt, outputs a string m or $\perp$ to indicate failure.

D. IBE

An IBE scheme Γ consists of four algorithms [39]:

$\mathsf{ISetup}(1^\lambda, 1^k) \rightarrow (pp^{\mathrm{IBE}}, mk)$: given security parameters λ and k, outputs public parameter pp^{IBE} and a master key mk.

$\mathsf{IExt}(mk, id) \rightarrow dk$: given the master key mk and an identity id, outputs a private key sk.

$\mathsf{IEnc}(id, m) \rightarrow c$: given an identity id and a message m, outputs a ciphertext c.

$\mathsf{IDec}(sk, c) \rightarrow m$: given a private key sk and a ciphertext c, outputs a message m.

In TiMTAPS, the admitter issues a tracing token for a message m and the tracer opens the signature to reveal its original signers by using the tracking token.

E. NIZK

A non-interactive zero-knowledge proof protocol enables a prover to convince a verifier that a certain statement is true,

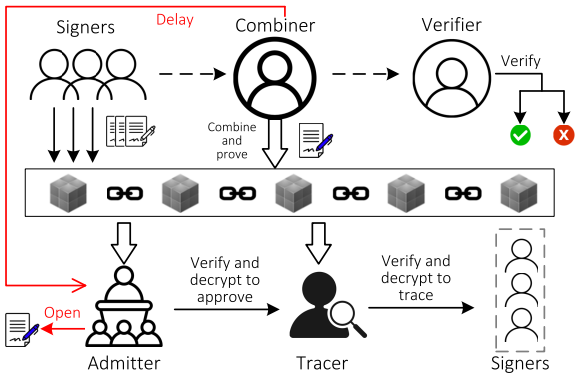


Fig. 2. System Model of TiMTAPS.

without revealing any information about the underlying secret. It consists of two algorithms (P, V):

$\mathsf{P}(1^\lambda, s, \mathsf{smt})$: given a security parameter λ, a secret s, and a statement smt, outputs the proof π.

$\mathsf{V}(\mathsf{smt}, \pi)$: given a statement smt and a proof π, outputs 1 if it is valid; otherwise, outputs 0.

An interactive proof generation process can be made non-interactive by applying the Fiat-Shamir heuristic [46] using a hash function to some public values and computed values.

F. PKE

A PKE scheme Ψ consists of three algorithms:

$\mathsf{EKGen}(1^\lambda) \rightarrow (sk, pk)$: given a security parameter λ, outputs a private key sk and a corresponding public key pk.

$\mathsf{Enc}(pk, m) \rightarrow c$: given a public key pk and a message m, outputs a ciphertext c.

$\mathsf{Dec}(sk, c) \rightarrow m$: given a ciphertext c and a private key sk, outputs a message m.

G. Sig

A SIG scheme Φ consists of three algorithms:

$\mathsf{SKGen}(1^\lambda) \rightarrow (sk, pk)$: given a security parameter λ, outputs a private key sk and a corresponding public key pk.

$\mathsf{Sign}(sk, m) \rightarrow \sigma$: given a private key sk and a message m, outputs a signature σ.

$\mathsf{Verify}(pk, m, \sigma) \rightarrow \{0, 1\}$: given a public key pk, a message m, and a signature σ, outputs 1 if σ is valid, otherwise outputs 0.

IV. TiMTAPS

A. System Model

The system model of TiMTAPS is depicted in Fig. 2, consisting of signer, combiner, admitter, tracer, and blockchain. We list the key notations in Table I.

1) Signer: A signer S_i belongs to a group of t signers $\mathcal{S} = \{S_1, S_2, \cdots, S_t\}$ that collaborate with one another to compute a signature on a message m. Each group has a signing group identifier $\mathcal{S}^{\text{id}}$, which serves only for one message. Each signer S_i computes a message tuple of six items $M_i = (ss_j, tlp_{jz}, tlp_{jR}, EmS_j, Eid_j^{AD}, Eid_j^V, cmt_j)$ where ss_i is a signature share, tlp_{iz} and tlp_{iR} are two HTLPs, EmS_i is the ciphertext of m and $\mathcal{S}^{\text{id}}$, Eid_i^{AD} and Eid_j^V are the ciphertexts of $\mathcal{S}^{\text{id}}$, and cmt_i is a commitment. Finally, S_i sends M_i to the blockchain via a transaction Tx.

2) Blockchain: The blockchain BC records all the transactions sent by signers, combiner, and tracer. It has a data pool of signature shares for the combiner to request, which is deployed on a Combining Smart Contract (CSC) with a unique address. The BC is maintained by several nodes, one of which is selected in each period to pack a new block. The smart contracts deployed on the blockchain mainly serve two core functions: managing on-chain data storage and providing role-based access interfaces for users with different identities.

3) Combiner: A combiner C with an identity id_C has a pair of encryption keys $(sk_C^{\text{enc}}, pk_C^{\text{enc}})$ and a pair of signing keys $(sk_C^{\text{sig}}, pk_C^{\text{sig}})$. It is a blockchain node and we assume that every blockchain node is a combiner. The consensus-selected node C in each period retrieves all signature shares $\{ss_i\}$ from the CSC and combines related signature shares into a complete signature σ. Note that the two puzzle sets $\{tlp_{jz}\}_j$ and $\{tlp_{jR}\}_j$ are not aggregated by C because this step is conducted by the admitter. But C has to calculate a commitment cmt for σ and a ciphertext Em^V for m to be used in the verification check later. Next, C encrypts m to a ciphertext c_z^m, and t to c_t. Finally, C generates a proof π to prove that both m is encapsulated, and computes a signature δ and finally the TiMTAPS signature $\sigma^{\text{TiM}} = (id_C, cmt, Em^V, R, c_z^m, c_t, \pi, \delta)$.

4) Admitter: The admitter AD is a trusted tracing authority that strengthens signers' anonymity against the tracer. This is achieved by holding the power of issuing a message-specific token for and the tracer to open a TS signature. Specifically, it generates a tracing token, i.e., decapsulation key dk, based on m and forwards it to the tracer, completing authorization. If C deliberately delays, AD aggregates the time lock to recover the complete signature σ.

5) Verifier: After receiving a $\sigma^{\text{TiM}} = (id_C, cmt, Em^V, R, c_z^m, c_t, \pi, \delta)$ from C, any honest verifier V verifies that whether both σ is a valid ATS signature and σ is encapsulated by using the NIZK proof π. Next, V receives a commitment cmt of σ from C and retrieves relevant HCs $\{cmt_j\}_{j=1}^t$ from the CSC. V aggregates the HCs to obtain a commitment cmt_{agg} of σ and compares it with cmt. If all the verifications pass, then V outputs 1, otherwise 0.

6) Tracer: When some signed message m is identified to be problematic and the tracer TR is granted a tracing token from the AD, TR decrypts c_z^m to obtain σ and then reveal the identities of the original signers.

Definition 1 (TiMTAPS) A threshold, accountable, and private signature with timed combining, verifiable combining, message-dependent tracing, i.e., TiMTAPS, is a tuple of six polynomial time algorithms $\Pi = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{Verify}, \mathsf{Trace}, \mathsf{Open})$:

- $\mathsf{Setup}(1^\lambda, 1^k, n, t, T) \rightarrow (pp, \{sk_i\}_{i=1}^n, \mathcal{PK}, sk_C, sk_{TR}, sk_A, sk_V^{enc})$: given security parameters λ and k, a number of signers n, a threshold t, and a time hardness parameter T, outputs public parameters pp, n private keys $\{sk_i\}_{i=1}^n$, a public key set $\mathcal{PK}$, a secret combining key sk_C, a secret tracing key sk_{TR}, a secret admiting key sk_A, and a secret key sk_V^{enc}.

Exp$^{\text{pPub}}$

1. $b \xleftarrow{\$} \{0,1\}$
2. $(n, t_0, t_1, \mathcal{S}_0, \mathcal{S}_1, \mathsf{state}) \xleftarrow{\$} \mathcal{A}_0(1^\lambda)$, $t_0, t_1 \in [n]$ //$\mathcal{A}_0$ picks two thresholds
3. $(pp, \{sk_i\}_{i=1}^n, \mathcal{PK}, sk_C, sk_{TR}, sk_A, sk_V^{enc}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, n, t, T)$
4. $b' \leftarrow \mathcal{A}_1^{\mathcal{O}_1(\cdot,\cdot,\cdot),\mathcal{O}_2(\cdot,\cdot)}(\mathcal{PK}, \mathsf{state})$ //$\mathcal{A}_1$ outputs a guess b'
5. Output $(b' = b)$.

where $\mathcal{O}_1(\mathcal{S}_0, \mathcal{S}_1, m)$: $\sigma^{\text{TiM}} \leftarrow \mathsf{Combine}(\mathcal{S}^{\text{id}}, \mathcal{S}, id_C, sk_C, m, \{\mathsf{Sign}(\mathcal{S}^{\text{id}}, sk_j, m, \mathcal{S}_b)\}_{S_j \in \mathcal{S}_b})$ //Sign with $\mathcal{S}_b$ for $\mathcal{S}_0 \subseteq [n]$, $\mathcal{S}_1 \subseteq [n]$, $|\mathcal{S}_0| = t_0$, and $|\mathcal{S}_1| = t_1$,

and $\mathcal{O}_2(m, \sigma^{\text{TiM}})$ returns $\mathsf{Trace}(\mathcal{PK}, sk_V^{\text{enc}}, sk_{TR}, sk_A, m, \sigma^{\text{TiM}}, \{cmt_j\}_{j=1}^t)$.

Restriction: if σ^{TiM} is computed from $\mathcal{O}_1(\cdot,\cdot,m)$, $\mathcal{O}_2$ will not be fed with (m, σ^{TiM}).

Fig. 3. Experiment of Privacy against the Public.

Exp$^{\text{pSig}}$

1. $b \xleftarrow{\$} \{0,1\}$
2. $(n, t, \mathcal{S}, \mathsf{state}) \xleftarrow{\$} \mathcal{A}_0(1^\lambda)$, $t \in [n]$
3. $(pp, \{sk_i\}_{i=1}^n, \mathcal{PK}, sk_C, sk_{TR}, sk_A, sk_V^{enc}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, n, t, T)$
4. $b' \leftarrow \mathcal{A}_1^{\mathcal{O}_1(\cdot,\cdot,\cdot),\mathcal{O}_2(\cdot,\cdot)}(\mathcal{PK}, \{sk_i\}_{i=1}^n, \mathsf{state})$ //$\mathcal{A}_1$ receives secret keys of all signers
5. Output $(b' = b)$.

where $\mathcal{O}_1(\mathcal{S}_0, \mathcal{S}_1, m)$: $\sigma^{\text{TiM}} \leftarrow \mathsf{Combine}(\mathcal{S}^{\text{id}}, \mathcal{S}, id_C, sk_C, m, \{\mathsf{Sign}(\mathcal{S}^{\text{id}}, sk_j, m, \mathcal{S}_b)\}_{S_j \in \mathcal{S}_b})$ for $\mathcal{S}_0 \subseteq [n]$, $\mathcal{S}_1 \subseteq [n]$, $|\mathcal{S}_0| = t_0$, and $|\mathcal{S}_1| = t_1$,

and $\mathcal{O}_2(m, \sigma^{\text{TiM}})$ returns $\mathsf{Trace}(\mathcal{PK}, sk_V^{\text{enc}}, sk_{TR}, sk_A, \sigma^{\text{TiM}}, \{cmt_j\}_{j=1}^t)$.

Restriction: if σ^{TiM} is computed from $\mathcal{O}_1(\cdot,\cdot,m)$, $\mathcal{O}_2$ will not be fed with (m, σ^{TiM}).

Fig. 4. Experiment of Privacy against the Signers. (Yellow box means secret keys).

- $\mathsf{Sign}(\mathcal{S}^{\text{id}}, sk_j, m, \mathcal{S}) \rightarrow (ss_i, tlp_{jz}, tlp_{jR}, EmS_j, Eid_j^V, Eid_j^{AD}, cmt_j)$: given a group id $\mathcal{S}^{\text{id}}$, a private key sk_j, a message m, and a signer set $\mathcal{S}$, outputs a signature share ss_j, two HTLPs tlp_{jz}, tlp_{jR}, three ciphertexts EmS_j, Eid_j^{AD}, Eid_j^V, and a commitment cmt_j.
- $\mathsf{Combine}(\mathcal{S}^{\text{id}}, \mathcal{S}, id_C, sk_C, m, \{ss_j, EmS_j\}) \rightarrow \sigma^{\text{TiM}}$: given a signer set $\mathcal{S}$, a secret combining key sk_C, a message m, signature shares ss_j and ciphertext EmS_j where $\Psi.\mathsf{Dec}(sk_C^{\text{enc}}, EmS_j) = m \| \mathcal{S}^{\text{id}}$, outputs a complete TiMTAPS signature $\sigma^{\text{TiM}} = (id_C, cmt, Em^V, R, c_z^m, c_t, \pi, \delta)$ including an identity id_C, a commitment cmt, a ciphertext Em^V, an ATS signature part R, two ciphertexts c_z^m, c_t, a proof π, and a signature δ.
- $\mathsf{Verify}(\mathcal{PK}, sk_V^{\text{enc}}, m, \sigma^{\text{TiM}}, \{cmt_j\}_{\mathcal{S} \in \mathcal{S}_j}) \rightarrow \{0,1\}$: given a public key set $\mathcal{PK}$, a secret verification key sk_V^{enc},a message m, t commitments $\{cmt_j\}_{\mathcal{S} \in \mathcal{S}_j}$, a TiMTAPS signature σ^{TiM}, outputs either 1 (valid) or 0 (invalid).
- $\mathsf{Trace}(\mathcal{PK}, sk_V^{\text{enc}}, sk_{TR}, sk_A, m, \sigma^{\text{TiM}}, \{cmt_j\}_{j=1}^t) \rightarrow \{\mathcal{S}, \perp\}$: given a public key set $\mathcal{PK}$, a secret verification key sk_V^{enc}, a secret tracing key sk_{TR}, a secet key sk_A, a message m, a TiMTAPS signature σ^{TiM}, and a set of t commitments $\{cmt_j\}_{j=1}^t$, outputs the identities of the original signing group $\mathcal{S}$ or $\perp$ to indicate failure.
- $\mathsf{Open}(sk_A, \mathcal{S}^{\text{id}}, \{tlp_{jz}, tlp_{jR}\}_{\Psi.\mathsf{Dec}(sk_A^{\text{enc}}, Eid_j) = \mathcal{S}^{\text{id}}}) \rightarrow \sigma$: given a signing group identifier $\mathcal{S}^{\text{id}}$, two sets of HTLPs $\{tlp_{jz}\}$ and $\{tlp_{jR}\}$ where $\Psi.\mathsf{Dec}(sk_A^{\text{enc}}, Eid_j) = \mathcal{S}^{\text{id}}$, outputs a TS signature σ.

For simplicity here, we omit public parameters pp in the input of the last five functions. For correctness, we require that for all $t \in [n]$, all t-size sets $\mathcal{S}$, all $m \in \mathcal{M}$, and $\mathsf{Setup}(1^\lambda, 1^k, n, t, T) \rightarrow (pp, \{sk_i\}_{i=1}^n, \mathcal{PK}, sk_C, sk_{TR}, sk_A, sk_V^{enc})$, the following two conditions hold:

$$\Pr[\mathsf{Verify}(\mathsf{Combine}(\{\mathsf{Sign}(sk_j, m, \mathcal{S})\}_{S_j \in \mathcal{S}}) = 1] = 1,$$
$$\Pr[\mathsf{Trace}(\mathsf{Combine}(\{\mathsf{Sign}(sk_j, m, \mathcal{S})\}_{S_j \in \mathcal{S}}) = \mathcal{S}] = 1.$$

B. Privacy

We have two privacy requirements.

(1) **Privacy against the public**. A party who observes a sequence of ciphertext-TiMTAPS signature pairs $\{(m_i, \sigma_i^{\text{TiM}})\}_i$, cannot acquire anything useful about t or the original signers of each σ_i^{TiM}.

(2) **Privacy against signers**. Signers who collaborate and observe a sequence of ciphertext-TiMTAPS signature pairs $\{(m_i, \sigma_i^{\text{TiM}})\}_i$, cannot acquire anything useful about the original signers of each σ_i^{TiM}.

We formalize the two properties in the two experiments in Fig. 3 and Figure 4. Let Evt^{pPub} be the event that the game in Figure 3 outputs 1 and Evt^{pSig} be the event that the game in Figure 4 outputs 1. We define two winning advantage functions for an adversary $\mathcal{A}$ against the scheme Π, as a function of the security parameter λ:

$$Adv_{\mathcal{A},\Pi}^{\text{privP}}(\lambda) := |2\Pr[Evt^{\text{pPub}}] - 1|,$$
$$Adv_{\mathcal{A},\Pi}^{\text{privS}}(\lambda) := |2\Pr[Evt^{\text{pSig}}] - 1|.$$

Definition 2 (Privacy) A TiMTAPS scheme Π is private if for all Probabilistic Polynomial Time (PPT) adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, both $Adv_{\mathcal{A},\Pi}^{\text{privP}}(\lambda)$ and $Adv_{\mathcal{A},\Pi}^{\text{privS}}(\lambda)$ are negligible functions of λ.

Exp$^{\text{ua}}$

1. $(n, t, \mathcal{S}, \text{state}) \xleftarrow{\$} \mathcal{A}_0(1^\lambda)$ where $t \in [n]$, $\mathcal{S} \subseteq [n]$
2. $(pp, \{sk_i\}_{i=1}^n, \mathcal{PK}, sk_C, sk_{TR}, sk_A, sk_V^{enc}) \leftarrow \text{Setup}(1^\lambda, 1^k, n, t, T)$
3. $b' \leftarrow \mathcal{A}_1^{\mathcal{O}_3(\cdot,\cdot)}(\mathcal{PK}, \{sk_i\}_{\mathcal{S}_i \in \mathcal{S}}, sk_C, sk_{TR}, \text{state})$ //$\mathcal{A}_1$ receives secret keys of all signers, combiners, and tracer

where $\mathcal{O}_3(\mathcal{S}_j, m_j)$ returns the signature shares $\{\text{Sign}(sk_j, m_j, \mathcal{S}_i)\}_{S_j \in \mathcal{S}_i}$

Winning condition:

Let $(\mathcal{S}_1, m_1), (\mathcal{S}_2, m_2), \cdots$ be $\mathcal{A}_1$'s a sequence of queries to $\mathcal{O}_3$

Let $\mathcal{S}' \leftarrow \cup \mathcal{S}_i$, union over all queries to $\mathcal{O}_3(\mathcal{S}_i, m')$, if there is no such $\mathcal{O}_3$ on m', set $\mathcal{S}'$ as $\emptyset$

Let $\mathcal{S}_t \leftarrow \text{Trace}(\mathcal{PK}, sk_V^{\text{enc}}, sk_{TR}, sk_A, m_{\text{forg}}, \sigma_{\text{forg}}^{\text{TiM}}, \{cmt_j\}_{j=1}^t)$ //Trace a forgey $(m_{\text{forg}}, \sigma_{\text{forg}}^{\text{TiM}})$

Output 1 if $\text{Verify}(\mathcal{PK}, m_{\text{forg}}, \sigma_{\text{forg}}^{\text{TiM}}, \{cmt_j\}_{j=1}^t) = 1$ and //$\mathcal{A}$ succeeds if a group outside of $\mathcal{S} \cup \mathcal{S}'$ is condemned either $\mathcal{S}_t \nsubseteq \mathcal{S} \cup \mathcal{S}'$ or if $\mathcal{S}_t = \perp$ //or the tracing fails

Fig. 5. Experiment of Unforgeability and Accountability.

C. Unforgeability and Accountability

TiMTAPS has to satisfy unforgeability and accountability, like any TS scheme, i.e., existential unforgeability under a chosen message attack with traceability. Informally, the two properties are defined as follows.

- **Unforgeability**: an adversary that enslaves less than t signer cannot forge a valid signature on a message.
- **Accountability**: an adversary that enslaves t or more signers cannot generate a valid message-signature pair that traces to one honest signer.

We formalize these two properties in the experiment in Fig. 5. Let $Adv_{\mathcal{A},\Pi}^{\text{ua}}(\lambda)$ be the probability that $\mathcal{A}$ wins the experiment against the TiMTAPS scheme Π.

Definition 3 (Unforgeability and Accountability) A TiMTAPS scheme Π is both unforgeable and accountable if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, function $Adv_{\mathcal{A},\Pi}^{\text{ua}}(\lambda)$ is a negligible function of λ.

In the attack model, the attacker is assumed to be a combiner who attempts to manipulate the final signature result. To illustrate the attack process more intuitively, we use an election scenario as an example. Suppose a group of n voters are selecting between two candidates, A and B, with the voting process coordinated by a combiner (i.e., a vote collector). After all votes are cast, the legitimate outcome is that candidate A wins. However, a malicious combiner may try to forge the aggregated signature to falsely announce that B was elected. Our scheme enables the public to independently verify the consistency between the final result and the voters' original inputs. This allows for the detection of any manipulation and ensures that the result remains verifiable.

V. Our Construction of TiMTAPS

In this section, we present a concrete construction Π of TiMTAPS (Fig. 5). It consists of five main functions as described in Section IV-A and nine building blocks as follows:

- ATS Δ = (AKGen, ASign, ACombine, AVerify, ATrace);
- HTLP Λ = (PSetup, PGen, PSolve, PEval);
- HC Ω = (CSetup, CCom, CAdd, COpen);
- IBE Γ = (ISetup, IExt, IEnc, IDec);
- NIZK Υ = (P, V);
- PKE Ψ = (EKGen, Enc, Dec);
- SIG Φ = (SKGen, Sign, Verify);
- Two hash functions H, Hash.

A. Setup

First of all, the system is setup by running the key generation functions and setup functions of the building blocks (ATS, HTLP, HC, IBM, PKE, SIG), and committing to the public key pk of the ATS scheme. Then, all public parameters pp and the public key set $\mathcal{PK}$ are released. Each signer S_i obtains a secret key sk_i. Each C, TR, A and V receive a pair of encryption keys, C receive a pair of signing keys. The BC is also initialized among combiners with a CSC. It awaits transactions from signers, combiners, verifiers, and the tracer.

Remark 1 (Protection of pk) The public key of the Δ scheme, i.e., $pk = (t, \{pk_i\}_{i=1}^n)$, has to be kept secret from the public since t is a secret value and pk is also an input for both Δ.Combine and Δ.Trace. Once an adversary $\mathcal{A}$ gets hold of a message and a set of its signature shares, it can combine the shares to acquire an ATS signature. If $\mathcal{A}$ acquires a pair of ATS message-signature (m, σ), it can reveal original signers.

B. Sign

Now we assume that t collaborating signers in a signing group $\mathcal{S} = \{S_j\}$ are about to generate a TiMTAPS signature σ^{TiM} on a message m. For each signer S_j, it computes a signature share $ss_j = (z_j, R_j)$ and encrypts $m \| \mathcal{S}^{\text{id}}$ to get EmS_j. Here, we require the encryption of $m \| \mathcal{S}^{\text{id}}$ for the sake of unlinkability. Two HTLPs tlp_{jz}, tlp_{jR} and two ciphertexts Eid_j^{AD}, Eid_j^V are calculated as backup against a potentially procrastinating combiner. Since we are sewing and knitting several cryptographic techniques together, special attention is required to guarantee function suitability and parameter docking. For instance, the $\sum z_j$ and the $\prod R_j$ should be smaller than $N = p_1 q_2$ from HTLP [35].

A commitment cmt_j is also calculated to commit to z_j and R_j, paving the way for checking verifiable combining later. Finally, S_j outputs $M_j = (ss_i, tlp_{jz}, tlp_{jR}, EmS_j, Eid_j^V, Eid_j^{AD}, cmt_j)$ and sends to the BC a signing transaction $\text{Tx}_j^{\text{Sign}} = (\text{“Sign''}, M_j, \sigma_j^{\text{bc}})$. Here, σ_j^{bc} is a transaction signature generated by using a blockchain account public key.

Remark 2 (Protection of m and $\mathcal{S}^{id}$) The messages signed by any set must be protected since exposing of m will reveal the link of the t signers, thereby leaking t. The $\mathcal{S}^{\mathrm{id}}$ is encrypted for V and TR to perform verification and tracing, respectively.

C. Combine

In the beginning of combination of signature shares, C has to encrypt t for verification, decrypt all EmS_i for a signing group identifier and then collect relevant signature shares. Say there is a collected set of signature shares $\{ss_j\}_{S_j \in \mathcal{S}}$ and each $ss_j = (z_j, R_j)$. C aggregated $\{z_j\}$ and $\{R_j\}$ separately to acquire an ATS signature $\sigma = (z, R)$, then computes a new commitment $cmt = g_3^z \cdot R$ and a ciphertext Em^V, both of which are used for verification. Note that m has to be encrypted in case of privacy leakage, i.e., being linked by an adversary if two messages are the same from the same signing group.

Next, C encrypts z by using the IBE to bind the potential tracing of σ^{TiM} to m and generates a proof π that σ is a valid ATS signature on m, cmt_{pk} correctly commits to pk, and c_z^m is correctly encrypted from z. Note that z has to be in G, which is set to be a subgroup of $Z_{p_4}^*$ with g_4 being the generator.

Last, C computes a signature δ, outputs a TiMTAPS signature $\sigma^{\mathrm{TiMTAPS}} = (id_C, cmt, Em^V, R, c_z^m, c_t, \pi, \delta)$, and sends a combining transaction $\mathrm{Tx}_C^{\mathrm{Comb}} = (\text{“Combine”}, m, \sigma^{\mathrm{TiM}}, \sigma_C^{\mathrm{bc}})$. Here, we give the *details of generating and verifying* π in Fig. 7, Fig. 8, and Fig. 9.

D. Verify

Upon seeing $\mathrm{Tx}_C^{\mathrm{Comb}}$, any honest verifier V decrypts Eid_j^V to collect pertinent $\{cmt_j\}$ of some signing group $\mathcal{S}$ from CBC. Then, V verifies the validity of σ^{TiM} by checking the validity of δ, checking the validity of σ via π, and checking the verifiable combining with cmt and $\{cmt_j\}$. To acknowledge the validity of σ^{TiM}, V uploads a verification transaction $\mathrm{Tx}_V^{\mathrm{veri}} = (\text{“Combine”}, m, \sigma^{\mathrm{TiM}}, \sigma_V^{\mathrm{bc}})$.

E. Trace

On the verification of σ^{TiM}, a tracer TR first invokes Verify on $(m, \sigma^{\mathrm{TiM}})$. If it is valid, TR requests A for a tracing token dk on $(m, \sigma^{\mathrm{TiM}})$ to decrypt c_z^m to obtain z. Then, TR can trace from σ to its original signing group $\mathcal{S}$. A tracing transaction $\mathrm{Tx}_C^{\mathrm{Trac}} = (\text{“Trace”}, \mathsf{H}(m, \sigma), \sigma_{TR}^{\mathrm{bc}})$. is sent to BC.

Thanks to this enhanced functionality, we are able to restrict the ability of the tracer without any complicated interactive operations, e.g. threshold encryption and decryption.

F. Open

When some functioning combiner procrastinates, A first classifies tlp_{jz} and tlp_{jR} similarly to the handling of cmt_j in Trace. Next, AD computes two aggregated tlp_z and tlp_R, both of which are unlocked into z and R. An opening transaction $\mathrm{Tx}_C^{\mathrm{Comb}} = (\text{“Open”}, \mathsf{H}(m, \sigma), \sigma_{AD}^{\mathrm{bc}})$ is sent to BC. For further steps, AD either stores $\sigma = (z, R)$ to await a release date or securely submits it to a regulatory authority, e.g., financial supervisory authority, to enforce punishment.

VI. Privacy and Security of TiMTAPS

Now we formally prove that TiMTAPS is private, unforgeable, and accountable.

Theorem 1: The TiMTAPS scheme Π in Fig. 5 is private, unforgeable, and accountable, assuming that Δ is secure, Λ is secure, Ω is hiding and binding, Γ is adaptively IND-CPA secure, Υ is an argument of knowledge and honest verifier zero knowledge (HVZK), Ψ is semantically secure, and Φ is strongly unforgeable. The proof of Theorem 1 emerges from proving the following three lemmas, for which we give concrete security bounds.

Lemma 1: TiMTAPS Π is private against the public if Ω is hiding, Ψ is semantically secure, Γ is adaptively IND-CPA secure, Υ is HVZK, Φ is strongly unforgeable, i.e., for all PPT adversaries $\mathcal{A}$, there exists adversaries $\mathcal{B}_0$, $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$, $\mathcal{B}_4$, and $\mathcal{B}_5$, such that

$$\begin{aligned}\mathrm{Adv}_{\mathcal{A},\Pi}^{\mathrm{pPub}}(\lambda) \le 2(&\epsilon_{\mathcal{B}_0}(\lambda) + \mathrm{Adv}_{\mathcal{B}_1,\Psi}^{\text{ind-cpa}}(\lambda) \\ &+ n\mathrm{Adv}_{\mathcal{B}_1,\Psi}^{\text{ind-cpa}}(\lambda) + \mathrm{Adv}_{\mathcal{B}_2,\Gamma}^{\text{a-ind-cpa}}(\lambda) \\ &+ q \cdot \mathrm{Adv}_{\mathcal{B}_3,\Upsilon}^{\text{hvzk}}(\lambda) + \mathrm{Adv}_{\mathcal{B}_4,\Phi}^{\text{euf-cma}}(\lambda)) \end{aligned} \tag{1}$$

where $\epsilon_{\mathcal{B}_0}(\lambda)$ is hiding statistical distance of Ω and q is query number.

Proof. We prove Lemma 1 by designing a series of six hybrids (experiments).

Hybrid 0. This is the experiment of privacy against the public $\mathbf{Exp}^{\mathrm{pPub}}$ defined in Fig. 3 applied to Π. We define Evt_0 as $\mathcal{A}$ wins Exp 0, thus

$$\mathrm{Adv}_{\mathcal{A},\Pi}^{\mathrm{pPub}}(\lambda) = |2\Pr[Evt_0] - 1|. \tag{2}$$

Hybrid 1. This is identical to Hybrid 0 except that the step 8 of Setup in Fig. 6 is altered to be $r_{pk} \leftarrow \mathcal{R}_\lambda$, $\mathsf{com}_{pk} \leftarrow \Omega.\mathsf{CCom}(0, r_{pk})$, i.e., $\mathcal{B}$ commits to 0 rather than pk. Since Ω is hiding, the adversary's advantage in Hybrid 1 is at most negligibly different from its advantage in Hybrid 0. We define Evt_1 as $\mathcal{B}_1$ wins Hybrid 1, thus

$$|\Pr[Evt_1] - \Pr[Evt_0]| \le \epsilon_{\mathcal{B}_0}(\lambda). \tag{3}$$

Hybrid 2. This is identical to Exp 1 except that the oracle $\mathcal{O}_1(\mathcal{S}_0, \mathcal{S}_1, m)$ is altered such that the step 1 of $\mathsf{Combine}$ in Fig. 6 outputs $\Psi.\mathsf{Enc}(pk_V^{\mathrm{enc}}, 0)$, i.e., and $\mathcal{B}$ encrypts 0 rather than t. Since Ψ is semantically secure, $\mathcal{B}$'s advantage in Hybrid 2 is at most negligibly different from its advantage in Exp 1. We define Evt_2 as $\mathcal{B}_2$ wins Exp 2, thus

$$|\Pr[Evt_2] - \Pr[Evt_1]| \le \mathrm{Adv}_{\mathcal{B}_1,\Psi}^{\text{ind-cpa}}(\lambda). \tag{4}$$

Hybrid 3. This is identical to Exp 2 except that the oracle $\mathcal{O}_1(\mathcal{S}_0, \mathcal{S}_1, m)$ is altered such that the step 3 of $\mathsf{Combine}$ in Fig. 6 encrypts 0 rather than b_i $(1 \le i \le n)$. Similarly, we have

$$|\Pr[Evt_3] - \Pr[Evt_2]| \le n\mathrm{Adv}_{\mathcal{B}_2,\Psi}^{\text{ind-cpa}}(\lambda). \tag{5}$$

Hybrid 4. This is identical to Exp 3 except that the oracle $\mathcal{O}_1(\mathcal{S}_0, \mathcal{S}_1, m)$ is altered such that the step 5 of $\mathsf{Combine}$ in Fig. 6 outputs $\Gamma.\mathsf{IEnc}(m, 0)$, i.e., $\mathcal{B}$ encrypts 0 rather than m. Since Γ is adaptively IND-CPA secure, $\mathcal{B}$'s advantage in Hybrid 4 is at most negligibly different from its advantage in Exp 3. We define Evt_3 as $\mathcal{B}_3$ wins Exp 4, thus

$$|\Pr[Evt_4] - \Pr[Evt_3]| \le \mathrm{Adv}_{\mathcal{B}_3,\Psi}^{\text{a-ind-cpa}}(\lambda). \tag{6}$$

$\mathsf{Setup}(1^\lambda, 1^k, n, t, T) \to (pp, \{sk_i\}_{i=1}^n, \mathcal{PK}, sk_C, sk_{TR}, sk_A, sk_V^{\text{enc}})$

1. $(pp^{\text{ATS}} = (g_1, h_1), t, \{sk_i\}_{i=1}^n, pk = \{pk_i\}_{i=1}^n) \leftarrow \Delta.\mathsf{AKGen}(1^\lambda, n, t)$, $sk_i \xleftarrow{\$} \mathbb{Z}_{q_1}$, $pk_i = g_1^{sk_i}$
2. $pp^{\text{HTLP}} = (T, N, g_2, h_2 = g_2^{2^T}) \leftarrow \Lambda.\mathsf{PSetup}(1^\lambda, T)$, where $\tilde{g} \xleftarrow{\$} \mathbb{Z}_N$, $g_2 = -\tilde{g}^2 \bmod N$, $N = p_1 q_2$, $p_1 = 2p_1' + 1$, $q_2 = 2q_2' + 1$, and p_1', q_2', p_1, q_2 are primes
3. $pp^{\text{HC}} = (g_3, h_3) \leftarrow \Omega.\mathsf{CSetup}(1^\lambda)$
4. $(pp^{\text{IBE}}, mk) \leftarrow \Gamma.\mathsf{ISetup}(1^\lambda, 1^k)$ where $e \xleftarrow{\$} Z_{q_4}$, $pp^{\text{IBE}} = (g_4, g_5 = g_4^e, \{D_i\}_{i=0}^k)$, $D_i = g_4^{d_i} g_5^{d_i'}$, $mk = (p_2, p_3)$, $p_2(x) = \sum_{i=0}^k d_i x^i$, $p_3(x) = \sum_{i=0}^k d_i' x^i$
5. $(sk_C^{\text{enc}}, pk_C^{\text{enc}}) \leftarrow \Psi.\mathsf{EKGen}(1^\lambda)$, $(sk_C^{\text{sig}}, pk_C^{\text{sig}}) \leftarrow \Phi.\mathsf{SKGen}(1^\lambda)$ //Assume that there are several combiners
6. $(sk_{TR}^{\text{enc}}, pk_{TR}^{\text{enc}}) \leftarrow \Psi.\mathsf{EKGen}(1^\lambda)$, $(sk_A^{\text{enc}}, pk_A^{\text{enc}}) \leftarrow \Psi.\mathsf{EKGen}(1^\lambda)$, $(sk_V^{\text{enc}}, pk_V^{\text{enc}}) \leftarrow \Psi.\mathsf{EKGen}(1^\lambda)$
7. $r_{pk} \xleftarrow{\$} \mathcal{R}_\lambda$, $cmt_{pk} \leftarrow \Omega.\mathsf{CCom}(pk, r_{pk})$ //Commit to pk that is not made public
8. $pp \leftarrow (pp^{\text{ATS}}, pp^{\text{HTLP}}, pp^{\text{HC}}, pp^{\text{IBE}})$ //Public parameters
9. $\mathcal{PK} \leftarrow (pk_C^{\text{enc}}, pk_C^{\text{sig}}, pk_{TR}^{\text{enc}}, pk_A^{\text{enc}}, pk_V^{\text{enc}}, cmt_{pk})$ //Public key set
10. $sk_C \leftarrow (pk, sk_C^{\text{enc}}, sk_C^{\text{sig}}, t, cmt_{pk}, r_{pk})$, $sk_{TR} \leftarrow (pk, sk_{TR}^{\text{enc}})$, $sk_A \leftarrow (sk_A^{\text{enc}}, mk)$ //Secret key sets
11. Output $(pp, \{sk_i\}_{i=1}^n, \mathcal{PK}, sk_C, sk_{TR}, sk_A, sk_V^{\text{enc}})$

$\mathsf{Sign}(\mathcal{S}^{\text{id}}, sk_j, m, \mathcal{S}) \to (ss_j, tlp_{jz}, tlp_{jR}, EmS_j, Eid_j^{AD}, Eid_j^V, cmt_j)$ for $1 \le j \le t$:

1. $ss_j \leftarrow \Delta.\mathsf{ASign}(sk_j, m, \mathcal{S})$, where $S_j \in \mathcal{S}$, $r_j \xleftarrow{\$} \mathbb{Z}_{q_1}$, $R_j = g_1^{r_j}$, $z_j = r_j + sk_j \cdot c$, $ss_j = (z_j, R_j)$ that satisfies $g_1^{z_j} = pk_j^c \cdot R_j$ for $c = \mathsf{H}(pk, R, m)$, $R = \prod_{S_j \in \mathcal{S}} R_j$, and $\mathsf{H} : \mathcal{PKS} \times \mathbb{G} \times \mathcal{M} \to \mathbb{Z}_{q_1}$
2. $EmS_j \leftarrow \Psi.\mathsf{Enc}(pk_C^{\text{enc}}, m || \mathcal{S}^{\text{id}}) = ((g_6)^{r_{mS_j}}, g_6^m \cdot (pk_C^{\text{enc}})^{r_{mS_j}}, g_6^{S^{id}} \cdot (pk_C^{\text{enc}})^{r_{mS_j}})$ //S_j protects m and S^{id}
3. $tlp_{jz} \leftarrow \mathsf{PGen}^{\text{add}}(pp^{\text{HTLP}}, z_j)$, $tlp_{jz} = (u_{jz}, v_{jz})$, $r_{jz} \xleftarrow{\$} [1, N^2]$, $u_{jz} = g_2^{r_{jz}} \bmod N$, $v_{jz} = h_2^{r_{jz} N} (1+N)^{z_j} \bmod N$
4. $tlp_{jR} \leftarrow \mathsf{PGen}^{\text{mul}}(pp^{\text{HTLP}}, R_j)$, $tlp_{jR} = (u_{jR}, v_{jR})$, $r_{jR} \xleftarrow{\$} [1, N^2]$, $u_{jR} = g_2^{r_{jR}} \bmod N$, $v_{jR} = h_2^{r_{jR}} \cdot R_j \bmod N$
5. $Eid_j^{AD} \leftarrow \Psi.\mathsf{Enc}(pk_A^{\text{enc}}, \mathcal{S}^{\text{id}})$, $Eid_j^V \leftarrow \Psi.\mathsf{Enc}(pk_V^{\text{enc}}, \mathcal{S}^{\text{id}})$ //S_j encrypts $\mathcal{S}^{\text{id}}$ for Verifier and Admitter
6. $cmt_j \leftarrow \Omega.\mathsf{CCom}(z_j, R_j)$, where $cmt_j = g_3^{z_j} R_j$ //S_j prepares for verification later
7. Output $(ss_j, tlp_{jz}, tlp_{jR}, EmS_j, Eid_j^{AD}, Eid_j^V, cmt_j)$

$\mathsf{Combine}(\mathcal{S}^{\text{id}}, \mathcal{S}, id_C, sk_C, m, \{ss_j, EmS_j\}) \to \sigma^{\text{TiM}}$:

1. $c_t \leftarrow \Psi.\mathsf{Enc}(pk_V^{\text{enc}}, t)$, where $c_t = (c_{t0}, c_{t1}) = (g_6^{\overline{r}}, g_6^t \cdot (pk_V^{\text{enc}})^{\overline{r}})$ //C encrypts t for verification later
2. $m_j || \mathcal{S}_j^{\text{id}} \leftarrow \Psi.\mathsf{Dec}(sk_C^{\text{enc}}, EmS_j)$ //Find ss_j, where $S_j \in \mathcal{S}$
3. Parse ss_j as (z_j, R_j), $\sigma \leftarrow \Delta.\mathsf{ACombine}(pk, m, \mathcal{S}, \{ss_j\}_{S_j \in \mathcal{S}})$, $\sigma = (z, R)$ where $z = \sum_{S_j \in \mathcal{S}} z_j$, $R = \prod_{S_j \in \mathcal{S}} R_j$, $c = \mathsf{Hash}(pk', c_t, R, m)$
4. Set $(b_1, b_2, \cdots, b_n) \in \{0,1\}^n$, $b_i = 1$ if $\mathcal{S}_i \in \mathcal{S}$, encrypt b_i with ElGamal, $\gamma \xleftarrow{\$} Z_{q_1}$, $\alpha = H(c_0, \{c_{b_i}\}_{\mathcal{S}_i \in \mathcal{S}})$, $\phi_i = \alpha^i \gamma (1 - b_i)$ //thus, $g_1^z = (\prod_{i=1}^n (pk_i)^{b_i})^c \cdot R$
5. $cmt = g_3^z \cdot R$, $Em^V \leftarrow \Psi.\mathsf{Enc}(pk_V^{\text{enc}}, m)$ //C computes commitment and encrypts m for verification later
6. $c_z^m \leftarrow \Gamma.\mathsf{IEnc}(m, z)$, $c_z^m = (w_1, w_2, c)$, $r' \xleftarrow{\$} \mathbb{Z}_{q_4}$, $w_1 = g_4^{r'}$, $w_2 = g_5^{r'}$, $\mathcal{D}_m = \prod_{i=0}^k D_i^{m^i}$, $ek = \mathcal{D}_m^{r'}$, $c_z = z \cdot ek$
7. Generate a proof π by using $\Upsilon.\mathsf{P}$ for the relation: $\mathcal{RL}((c_t, m, cmt_{pk}, c_z^m); (\sigma, pk, r_{pk})) = 1$ iff
$$\{\Delta.\mathsf{Verify}(pk, m, \sigma) = 1,\ \Omega.\mathsf{Verify}(pk, r_{pk}, cmt_{pk}) = 1,\ \Gamma.\mathsf{IEnc}(m, z) = c_z^m\}$$
8. $\delta \leftarrow \Phi.\mathsf{Sign}(sk_C^{\text{sig}}, (m, id_C, cmt, c_z^m, c_t, \pi))$ //C generates a normal digital signature
9. Output a TiMTAPS signature $\sigma^{\text{TiM}} = (id_C, cmt, Em^V, R, c_z^m, c_t, \pi, \delta)$

$\mathsf{Verify}(\mathcal{PK}, sk_V^{\text{enc}}, \sigma^{\text{TiM}}, \{cmt_j\}_{S_j \in \mathcal{S}}) \to \{0,1\}$: decrypt Em^V and output 1 if (1) $\Phi.\mathsf{Verify}(pk_C^{\text{sig}}, (m, id_C, cmt, c_z^m, c_t, \pi), \delta) = 1$, (2) $\Upsilon.\mathsf{V}(\mathsf{smt}, \pi) = 1$, and (3) $cmt = \sum_{S_j \in \mathcal{S}} cmt_j$.

$\mathsf{Trace}(\mathcal{PK}, sk_V^{\text{enc}}, sk_{TR}, sk_A, m, \sigma^{\text{TiM}}, \{cmt_j\}_{j=1}^t) \to \{\mathcal{S}, \perp\}$:

1. If $\mathsf{Verify}(\mathcal{PK}, sk_V^{\text{enc}}, \{cmt_j\}_{j=1}^t, m, \sigma^{\text{TiM}}) = 0$, stop.
2. $dk \leftarrow \Gamma.\mathsf{IExt}(mk, m)$ where $dk = (p_2(m), p_3(m))$ //TR receives dk, i.e., a tracing token, from A
3. Parse $c_z^m = (w_1, w_2, c)$, $\sigma \leftarrow \Gamma.\mathsf{IDec}(dk, c_z^m)$ where $ek = w_1^{p_2(m)} \cdot w_2^{p_3(m)}$ and $z = c_z \cdot ek^{-1}$ //TR recovers σ
4. Output $\Delta.\mathsf{ATrace}(pk, m, \sigma)$. //$TR$ reveals the identities of original signers

$\mathsf{Open}(sk_A, \mathcal{S}^{\text{id}}, \{tlp_{jz}, tlp_{jR}\}_{\Psi.\mathsf{Dec}(sk_A^{\text{enc}}, Eid_j) = \mathcal{S}^{\text{id}}}) \to \sigma$:

1. $tlp_z \leftarrow \Lambda.\mathsf{PEval}^{\text{add}}(C, pp, \{tlp_{jz}\})$, $tlp_R \leftarrow \Lambda.\mathsf{PEval}^{\text{mul}}(C, pp, \{tlp_{jR}\}_{S_j \in \mathcal{S}})$
2. $z \leftarrow \Lambda.\mathsf{PSolve}^{\text{add}}(pp, tlp_z)$, $R \leftarrow \Lambda.\mathsf{PSolve}^{\text{mul}}(pp, tlp_R)$, output $\sigma = (z, R)$

Fig. 6. The TiMTAPS scheme Π.

(1.1) Prove $g_1^z = (\prod_{i=1}^n pk_i^{b_i})^c \cdot R$

C as the prover:
- $\beta_z, \beta_{b_1}, \cdots, \beta_{b_n} \xleftarrow{\$} \mathbb{Z}_{q_1}$, $A = g_1^{\beta_z} \prod_{i=1}^n pk_i^{-c \cdot \beta_{bi}}$
- $h = \mathsf{Hash}(g_1^z, A)$
- $\hat{z} = z \cdot h + \beta_z$, $\hat{b_i} = b_i \cdot h + \beta_{b_i}$ for $1 \le i \le n$
- send $(A, \hat{z}, \hat{b_i})$ to V

V: $h = \mathsf{Hash}(g_1^z, A)$,
check $A \cdot R^h [\prod_{i=1}^n pk_i^{\hat{b_i}}]^c \stackrel{?}{=} g_1^{\hat{z}}$

(1.2) Prove $c_{t0} = g_6^{\overline{r}}$ and $c_{t1} = g_6^{\sum_{i=1}^n b_i} \cdot (pk_V^{\mathrm{enc}})^{\overline{r}}$

C:
- $r \xleftarrow{\$} \mathbb{Z}_{q_1}$, $\beta_{b_1}, \cdots, \beta_{b_n} \xleftarrow{\$} \mathbb{Z}_{q_1}$
- $A = g^r$, $B = g_1^{\sum_{i=1}^n \beta_{b_i}} (pk_V^{\mathrm{enc}})^r$
- $h = \mathsf{Hash}(c_{t0}, c_{t1}, A, B)$
- $\hat{r} = \overline{r} \cdot h + r$, $\hat{b_i} = b_i \cdot h + \beta_{b_i}$ for $1 \le i \le n$
- send $(A, B, \hat{r}, \{\hat{b_i}\}_{i=1}^n)$ to V

V: check $A \cdot c_{t0}^h \stackrel{?}{=} g_6^{\hat{r}}$, $B \cdot c_{t1}^h \stackrel{?}{=} (\prod_{i=1}^n g_1^{\hat{b_i}}) \cdot (pk_V^{\mathrm{enc}})^{\hat{r}}$

(1.3) Prove $(1 - b_i)b_i = 0$, $i \in [1, n]$

C has to prove $c_0 = g_1^{\gamma}$, $c_{b_i} = g_1^{b_i} h_i^{\gamma}$, and $\prod_{i=1}^n c_{b_i}^{\alpha^i (1 - b_i)} = \prod_{i=1}^n h_i^{\phi_i}$
- $\beta_\gamma, \beta_{b_1}, \cdots, \beta_{b_n} \leftarrow \mathbb{Z}_{q_1}, \beta_{\phi_1}, \cdots, \beta_{\phi_n} \leftarrow \mathbb{Z}_{q_1}$
- $A = g_1^{\beta_\gamma}, B_i = g_1^{\beta_{b_1}} h_i^{\beta_\gamma}$ for $1 \le i \le n$
- $C = \prod_{i=1}^n c_{b_i}^{\alpha^i \beta_{b_i}} h_i^{\beta_{\phi_i}}$
- $h = \mathsf{Hash}(c_0, \{c_{b_i}\}_{i=1}^n, \prod_{i=1}^n h_i^{\phi_i}, A, \{B_i\}_{i=1}^n, C)$
- $\hat{\gamma} = \gamma * h + \beta_\gamma$, $\hat{b_i} = b_i \cdot h + \beta_{b_i}$, $\hat{\phi_i} = \phi_i \cdot h + \beta_{\phi_i}$ for $1 \le i \le n$
- send $(A, B, C, \hat{\gamma}, \{\hat{b_i}\}_{i=1}^n, \{\hat{\phi_i}\}_{i=1}^n)$ to V

V: $h = \mathsf{Hash}(c_0, \{c_{b_i}\}_{i=1}^n, \prod_{i=1}^n h_i^{\phi_i}, A, \{B_i\}_{i=1}^n, C)$
check $A \cdot c_0^h \stackrel{?}{=} g_1^{\hat{\gamma}}$ and $B_i \cdot c_{b_i}^h \stackrel{?}{=} g_1^{\hat{b_i}} h_i^{\hat{\gamma}}$ and $C \cdot \prod_{i=1}^n c_{b_i}^{h \cdot \alpha^i} \stackrel{?}{=} \prod_{i=1}^n c_{b_i}^{\alpha^i \hat{b_i}} h_i^{\hat{\phi_i}}$

Fig. 7. Proving and Verifying $\Delta.\mathsf{Verify}(pk, m, \sigma) = 1$.

C:
- $\alpha_1, \alpha_2 \xleftarrow{\$} \mathbb{Z}_{q_3}$, $cmt' = g_3^{\alpha_1} h_2^{\alpha_2}$
- $h = \mathsf{Hash}(g_3, h_3, cmt_{pk}, cmt')$
- $\alpha_1' = h \cdot pk + \alpha_1, \alpha_2' = h \cdot r_{pk} + \alpha_2$
- send $(cmt_{pk}, cmt', \alpha_1', \alpha_2')$ to V

V: $h = \mathsf{Hash}(g, h, cmt_{pk}, cmt')$,
check $(cmt_{pk})^h cmt' \stackrel{?}{=} g_3^{\alpha_1'} h_2^{\alpha_2'}$

Fig. 8. Proving and Verifying $\Omega.\mathsf{Verify}(pk, r_{pk}, cmt_{pk}) = 1$.

C: assume $z = g_4^x$ (via a mapping function), $dk = g_4^a g_5^b$
- $\alpha_1, \alpha_2, \alpha_3, \xleftarrow{\$} \mathbb{Z}_{q_4}$, $A = g_4^{\alpha_1}$, $B = g_4^{\alpha_2}$, $C = g_5^{\alpha_3}$
- $h = \mathsf{Hash}(g_4, g_5, c, g_4^x)$
- $\alpha_1' = h \cdot x + \alpha_1, \alpha_2' = h \cdot a + \alpha_2, \alpha_3 = h \cdot b + \alpha_3$
- send $(c, g_4^x, \alpha_1', \alpha_2', \alpha_3')$ to V

V: $h = \mathsf{Hash}(g_4, g_5, c, g_4^x)$
check $A \cdot (g_4^x)^h \cdot B \cdot C \cdot c^h \stackrel{?}{=} g_4^{\alpha_1'} g_4^{\alpha_2'} h_5^{\alpha_3'}$

Fig. 9. Proving and Verifying $\Gamma.\mathsf{IEnc}(m, z) = c_z^m$.

Hybrid 5. This is identical to Exp 4 except that the oracle $\mathcal{O}_1(\mathcal{S}_0, \mathcal{S}_1, m)$ is altered such that step 6 of Combine outputs a proof π by using the simulator [48]. Given that simulated proofs are indistinguishable from real proofs computationally, $\mathcal{B}$'s advantage in Hybrid 5 is at most negligibly different from its advantage in Hybrid 4. We define Evt_5 as $\mathcal{B}_4$ wins Hybrid 5, thus

$$|\Pr[Evt_5] - \Pr[Evt_4]| \le q \cdot \mathrm{Adv}^{\mathrm{hvzk}}_{\mathcal{B}_4, \Upsilon}(\lambda). \tag{7}$$

Hybrid 6. This is identical to Exp 5 except that the oracle $\mathcal{O}_2(m, \sigma^{\mathrm{TiM}})$ outputs $\perp$. If Φ is strongly unforgeable, $\mathcal{B}$'s advantage in Exp 6 is indistinguishable from its advantage in Hybrid 5. We define Evt_6 as $\mathcal{B}_5$ wins Exp 5, thus

$$|\Pr[E_6] - \Pr[E_5]| \le \mathrm{Adv}^{\text{euf-cma}}_{\mathcal{B}_5, \Phi}(\lambda). \tag{8}$$

In Hybrid 6, $\mathcal{B}_5$'s view is independent of b. As a result, the advantage has no advantage in Hybrid, i.e.,

$$\Pr[Evt_5] = 1/2. \tag{9}$$

Integrating (2) –(8) proves (1). We complete the proof.□

Lemma 2: TiMTAPS Π is private against the signers if Ψ is semantically secure, Γ is adaptively IND-CPA secure, Υ is HVZK, Φ is strongly unforgeable, i.e., for all PPT adversaries $\mathcal{A}$, there exists adversaries $\mathcal{B}_0$, $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$, and $\mathcal{B}_4$, such that

$$\begin{aligned}\mathrm{Adv}^{\mathrm{pPub}}_{\mathcal{A}, \Pi}(\lambda) \le 2(&\mathrm{Adv}^{\text{ind-cpa}}_{\mathcal{B}_1, \Psi}(\lambda) + \mathrm{Adv}^{\text{a-ind-cpa}}_{\mathcal{B}_2, \Gamma}(\lambda) \\ &+ q \cdot \mathrm{Adv}^{\mathrm{hvzk}}_{\mathcal{B}_3, \Upsilon}(\lambda) + \mathrm{Adv}^{\text{euf-cma}}_{\mathcal{B}_4, \Phi}(\lambda)).\end{aligned} \tag{10}$$

$\epsilon_{\mathcal{B}_0}(\lambda)$ is deleted here because the signers have access to pk. The proof of Lemma 3 is omitted here since it is almost identical to the proof of Lemma 2

Lemma 3: TiMTAPS Π is unforgeable and accountable if Δ is secure, Υ is an argument of knowledge, and Ω is blinding, i.e., for all PPT adversaries $\mathcal{A}$, there exists adversaries $\mathcal{B}_0$, and $\mathcal{B}_1$, such that

$$\mathrm{Adv}^{\mathrm{forg}}_{\mathcal{A}, \Pi}(\lambda) \le (\mathrm{Adv}^{\mathrm{forg}}_{\mathcal{B}_0, \Delta}(\lambda) + \mathrm{Adv}^{\mathrm{bind}}_{\mathcal{B}_1, \Omega}(\lambda)) \cdot \rho(\lambda) + \varsigma(\lambda), \tag{11}$$

where ρ and ς are the knowledge error and tightness of the proof system.

Proof. We prove Lemma 1 by designing a series of three hybrids (experiments).

Hybrid 0. This is the experiment of unforgeability and accountability $\mathbf{Exp}^{\mathrm{ua}}$ defined in Fig. 5 applied to Π. We define Evt_0 as $\mathcal{A}$ wins Hybrid 0, thus

$$\mathrm{Adv}^{\mathrm{ua}}_{\mathcal{A}, \Pi}(\lambda) = \Pr[Evt_0]. \tag{12}$$

The constructions of the remaining two hybrids are similar to the ones in [14]. Therefore, we omit the proof here due to our limited space and kindly refer interested readers to it for more details. Generally, the idea of Hybrid 1 is to enhance the winning condition in Hybrid 0 to ask the adversary to output a valid forgery $(m_{\mathrm{forg}}, \sigma^{\mathrm{TiM}}_{\mathrm{forg}})$ along with a witness w satisfying $\mathcal{RL}$. We define Evt_1 as $\mathcal{B}_0$ wins Hybrid 1, leading to

$$\Pr[Evt_1] \ge (\Pr[Evt_0] - \varsigma(\lambda))/\rho(\lambda). \tag{13}$$

The idea of Hybrid 1 is to require that the forged ATS public key in w equals to pk. If they are not the same, we successfully launch an attack on Ω. We define Evt_2 as $\mathcal{B}_1$ wins Hybrid 2, leading to

$$\Pr[Evt_2] \ge \Pr[Evt_1] - \mathrm{Adv}^{\mathrm{bind}}_{\mathcal{B}_1, \Omega}(\lambda). \tag{14}$$

TABLE II
KEY NOTATIONS OF TIMTAPS

Notation	Definition
TS	Threshold Signature
VTS	Verifiable Timed Signature
TLP	Time-Lock Puzzle
HTLP	Homomorphic Time-Lock Puzzles
HC	Homomorphic Commitment
IBE	Identity-Based Encryption
NIZK	Non-Interactive Zero Knowledge
$\mathcal{S}$, S, BC, C	Signer group, signer, blockchain, combiner
AD, V, TR	Admitter, verifier, tracer
CSC	Combining Smart Contract
λ, k	Security parameter, number of key queires
t	Threshold of the TS scheme
T	Time for a TLP to be opened
m, id_{sig}	Message, group identity
ss, (z, R)	Signature share, Schnorr sgnature
tlp, cmt	Time-locked puzzle, commitment
Tx	Blockchain transaction
c_m	Ciphertext of m
π, δ	Non-interactive zero knowledge, signature
dk	Tracing token, decapsulation key

Next, for every Hybrid 2 adversary, there is an adversary $\mathcal{A}$ that attacks Δ and wins $\mathbf{Exp}^{\text{ua}}$ in Figure 5, with the same advantage as $\mathcal{A}$'s advantage in Hybrid 2, leading to

$$\text{Adv}^{\text{forg}}_{\mathcal{A},\Pi}(\lambda) \geq \Pr[Evt_2]. \tag{15}$$

Integrating (12) –(15) proves (11). We complete the proof.□

VII. PERFORMANCE EVALUATION

A. Experimental Settings

Dataset: We randomly generated the dataset since no dedicated dataset is available. **Parameters:** We vary the number of signers from 10 to 100, the message length from 100 KB to 1000 KB, the number of signature groups from 10 to 100 (when the threshold is 3, 4 and 5), the number of signers from 5 to 40, and the threshold from 10 to 90. To clarify the specific variables involved in our implementation and performance evaluation, we provide the values of core parameters in the Sign phase: Number of polynomial terms in the IBE component: k=5; number of signers in a group: n=20; threshold value: t=5; number of signer groups: 1; message size: 200 KB to 1000 KB. The source codes including additional parameters (e.g., prime group sizes and generator) and an instruction file is uploaded to https://github.com/UbiPLab/TiMTAPS. **Metrics:** We evaluate the computation cost and communication cost for each stage and its corresponding entity. **Baselines:** We also compare the performance of TiMTAPS with three baseline methods: TAPS [14], HiTAPS [15], and DeTAPS [16]. **Setup:** We implemented TiMTAPS on a Linux server running with Ubuntu 18.04 and Intel(R) Core(TM) i5-10210U CPU @1.60GHz 2.11GHz. We use the SHA256 hash function, EIGamal as asymmetric encryption, and web3.py as a tool to connect with Ethereum.

B. Computational Cost

We use different symbols to represent cryptographic operations: Gen, Ver, Sig denote generation, verification, and signing. Enc and Dec denote ElGamal encryption and decryption. $\mathsf{Enc}_{\mathsf{E}}$ and $\mathsf{Dec}_{\mathsf{E}}$ denote encryption and decryption when interacting with Enclave. $\mathsf{Enc}_{\mathsf{DTPKE}}$, Val, ShaDec, ShaVer, ShaCom correspond to Dynamic Threshold Public-Key Encryption (DTPKE) encryption, Validate, ShareDecrypt, ShareVerify, and ShareCombine. Com and Trac denote ATS combination and tracing. $\mathsf{Enc}_{\mathsf{KASE}}$, Trap, Adj, Tes denote KASE encryption, Trapdoor, Adjust and Test. IExt, IEnc, IDec denote IBE extract, encryption, and decryption. PGen, PEval and PSlove denote HTLP generation, evaluation, and solution. Tx denotes the communication of BlockChain. H denotes the hash function. We record the theoretical results in Table III.

In Setup, the system generates the required keys and parameters for the entire signature cycle. As shown in Fig. 10(a), the execution time increases with the number of signers. When the number of signers $n = 50$, the setup time is 75 ms.

In Sign, the signer generates shared signatures, including ATS signature sharing and time lock sharing. From Fig. 10(b), the execution time increases with the message length. When the signed message is 600 KB, the required time is 2.26 s. In the Combine phase, the combiner generates the TiMTAPS signature. From Fig. 10(c), the execution time increases as the number of signature groups grows and the threshold value rises. When the threshold is $t = 4$ and the number of signature groups is 50, the computation takes 18.61 s.

In Verify, a verifier checks the validity of a signature. From Fig. 10(d), the verification time increases with the number of signers. When $n = 25$, the required time is 353 ms.

In Trace, the tracer traces the signature. As shown in Fig. 10(e), the execution time increases exponentially with the threshold value. When the threshold $t = 4$, it takes 26 s to track 50 groups of signers.

In Open, ATS signature is given by admitter. Opening is divided into two steps, puzzle aggregation and puzzle opening. Since the opening time of time-lock puzzles is directly determined by the time parameter T, it is independent of the value of the signer and threshold, so the computational cost mainly measures the impact of increasing the threshold on the puzzle aggregation time. As depicted in Fig. 10, the puzzle aggregation time increases as the threshold increases. When the threshold value $t = 50$, the time is 251 ms.

C. Communication Overhead

In Sign, a signer sends a transaction to the blockchain, sending the signed information to the blockchain, which contains data $|ss_j|+|tlp_{jz}|+|tlp_{jR}|+|EmS_j|+|Eid_j^{AD}|+|Eid_j^{V}|+|cmt_j| = 160B+672B+944B+1520B+416B+416B+44B = 4172B$.

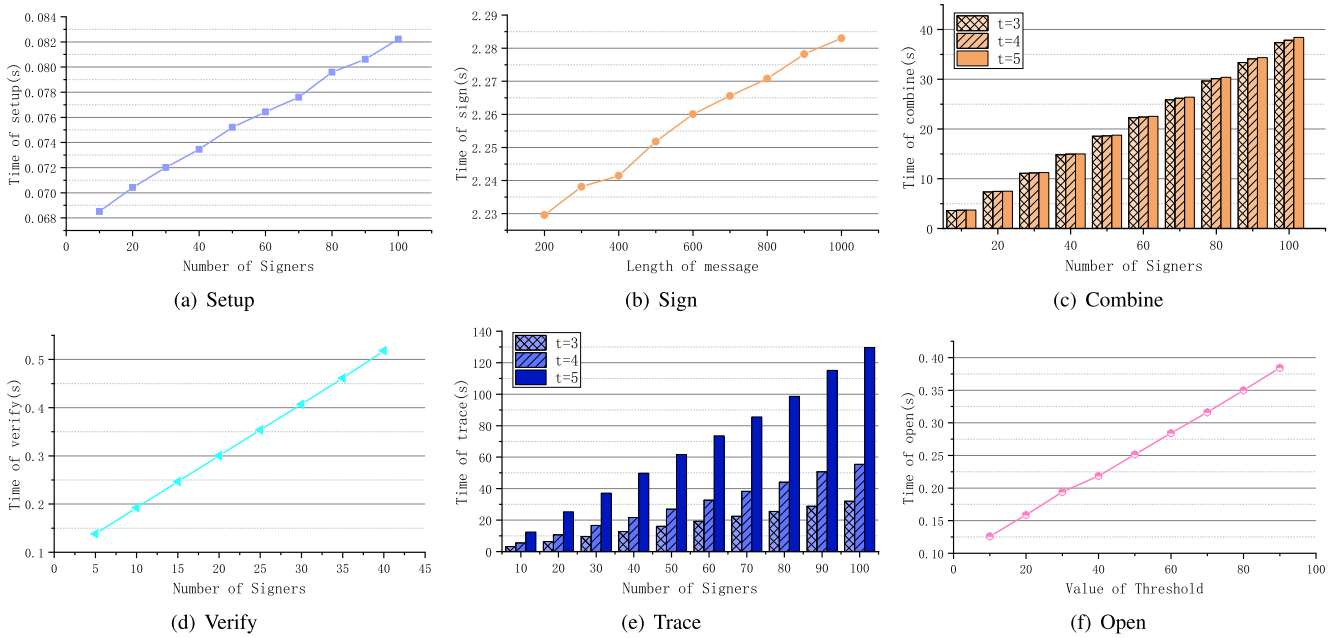
In Combine, combiner first establishes a transaction to obtain information from the blockchain, the transaction contains data $t(|ss_j| + |EmS_j|) = t(160B + 1520B) = (1680 * t)B$, and then combiner establishes a transaction to transmit the combined result to the blockchain, the transaction contains data $|id_C|+|cmt|+|Em^V|+|R|+|c_z^m|+|c_t|+|\pi|+|\delta| = 28B+44B+1104B+112B+568B+328B+14676B+1080B = 17940B$.

In Verify, verifier establishes transactions to get data from the blockchain, and transactions contain data $|id_C| + |cmt| + |Em^V| + |R| + |c_z^m| + |c_t| + |\pi| + |\delta| + t|cmt_j| = 28B + 44B +$

TABLE III
COMPARISON OF COMPUTATIONAL COSTS

Scheme	Sign	Combine		Verify	Trace					Open
	Signer	Enclave	Combiner	Verifier	N/W	SC	Enclave	Tracer	Admitter	Admitter
TAPS [14]	$\mathsf{Sig}+\mathsf{Enc}$	N/A	$t\mathsf{Dec}+\mathsf{Com}+\mathsf{Enc}+\mathsf{Sig}+\mathsf{Gen}_{\pi_1,\pi_2}$	$\mathsf{Ver}_{\pi_1,\pi_2}+\mathsf{Ver}_{\sigma}$	N/A	N/A	N/A	$\mathsf{Ver}_{\pi_1,\pi_2}+\mathsf{Ver}_{\sigma}+\mathsf{Dec}+\mathsf{Trac}$	N/A	N/A
HiTAPS [15]	$\mathsf{Sig}+3\mathsf{Enc}$	N/A	$3t\mathsf{Dec}+\mathsf{Com}+\mathsf{Enc}_{\mathsf{DTPKE}}+t'(2\mathsf{Enc}+2\mathsf{Gen}_{\pi_1}+\mathsf{H})+\mathsf{Sig}+\mathsf{Gen}_{\pi_2}+n\mathsf{Gen}_{\pi_3}$	$2t'\mathsf{Ver}_{\pi_1}+\mathsf{Ver}_{\pi_2}+n\mathsf{Ver}_{\pi_3}+\mathsf{Ver}_{\sigma}$	$2\mathsf{Dec}+\mathsf{Val}+\mathsf{H}+\mathsf{ShaDec}+\mathsf{Enc}$	N/A	N/A	$2t'\mathsf{Ver}_{\pi_1}+\mathsf{Ver}_{\pi_2}+n\mathsf{Ver}_{\pi_3}+\mathsf{Ver}_{\sigma}+t'\mathsf{Dec}+\mathsf{Val}+\mathsf{ShaCom}+\mathsf{Trac}$	N/A	N/A
DeTAPS [16]	$\mathsf{Sig}+4\mathsf{Enc}+\mathsf{Tx}$	$\mathsf{Enc}_{\mathsf{E}}+\mathsf{Dec}_{\mathsf{E}}$	$\mathsf{Tx}+\mathsf{Enc}_{\mathsf{E}}+\mathsf{Dec}_{\mathsf{E}}+4t\mathsf{Dec}+\mathsf{Com}+\mathsf{Enc}_{\mathsf{DTPKE}}+\mathsf{Enc}_{\mathsf{KASE}}+\mathsf{Gen}_{\pi_2,\pi_4}+n\mathsf{Gen}_{\pi_3}+\mathsf{Gen}_{\pi_5,\pi_6}+\mathsf{Sig}$	$\mathsf{Ver}_{\pi_2,\pi_4}+\mathsf{Ver}_{\pi_5,\pi_6}+n\mathsf{Ver}_{\pi_3}+\mathsf{Ver}_{\sigma}$	$\mathsf{Trap}+\mathsf{ShaDec}+\mathsf{Enc}$	$\mathsf{Adj}+\mathsf{Tes}$	$\mathsf{Enc}_{\mathsf{E}}+\mathsf{Dec}_{\mathsf{E}}$	$\mathsf{Tx}+\mathsf{Enc}_{\mathsf{E}}+\mathsf{Dec}_{\mathsf{E}}+\mathsf{Ver}_{\sigma}+\mathsf{Ver}_{\pi_2,\pi_4}+\mathsf{Ver}_{\pi_5,\pi_6}+n\mathsf{Ver}_{\pi_3}+t'(\mathsf{Dec}+\mathsf{ShaVer})+\mathsf{Val}+\mathsf{ShaCom}+\mathsf{Trac}$	N/A	N/A
TiMTAPS	$\mathsf{Sig}+3\mathsf{Enc}+2\mathsf{PGen}+\mathsf{Gen}_{\mathsf{Com}}+\mathsf{Tx}$	N/A	$\mathsf{Tx}+2\mathsf{Enc}+t\mathsf{Dec}+\mathsf{Com}+\mathsf{IEnc}+\mathsf{Gen}_{\mathsf{Com}}+\mathsf{Gen}_{\pi_2,\pi_7}+n\mathsf{Gen}_{\pi_3}+\mathsf{Sig}$	$\mathsf{Dec}+\mathsf{Ver}_{\pi_2,\pi_7}+n\mathsf{Ver}_{\pi_3}+\mathsf{Ver}_{\sigma}+\mathsf{Ver}_{\mathsf{Com}}$	N/A	N/A	N/A	$\mathsf{Ver}_{\pi_2,\pi_7}+n\mathsf{Ver}_{\pi_3}+\mathsf{Ver}_{\sigma}+\mathsf{Ver}_{\mathsf{Com}}+2\mathsf{Dec}+\mathsf{IDec}+\mathsf{Trac}$	$\mathsf{IExt}+\mathsf{Enc}$	$t\mathsf{Dec}+2(\mathsf{PEval}+\mathsf{PSlove})$

N/W denotes Notary or Witness, π_{1-7} are zero-knowledge proofs, correspond to EIGamal, ATS, Com, DTPKE, KASE, $\mathcal{N}\in n_1$ and IBE
n is the maximum number of signers, n_1 is the maximum number of Notaroes or Witnesses, t is the threshold of ATS and t' is the threshold of DTPKE



(a) Setup (b) Sign (c) Combine

(d) Verify (e) Trace (f) Open

Fig. 10. Computational Costs.

$1104B+112B+568B+328B+14676B+1080B+t*44B=(17940+44*t)B$.

In Trace, tracer establishes a transaction with the blockchain to obtain data, then tracked, and the transaction contains the data $|id_C|+|cmt|+|Em^V|+|R|+|c_z^m|+|c_t|+|\pi|+|\delta|+t|cmt_j|=28B+44B+1104B+112B+568B+328B+14676B+1080B+t*44B=(17940+44*t)B$.

In Open, admitter establishes a transaction with the blockchain to obtain data, and the transaction contains the data $t(|tlp_{jz}|+|tlp_{jR}|)=t(672B+944B)=(1616*t)B$. Similarly, we present the theoretical results of the communication overhead in Table IV.

D. Comparison

Now we compare the computing overhead and communication overhead of TiMTAPS with related work. The computing overhead and communication overhead of TAPS are both minimal, but comes with losing some privacy and security. Both HiTAPS and DeTAPS realize the witness of the tracking process. The former introduces DTPKE, combined with Hash, to incorporate the witness throughout the process. As a result, there is an increase in both computational and communication overhead at each step compared to TAPS. The latter, due to the further integration of blockchain, SGX technology, and KASE technology, has further increased the computing overhead

TABLE IV
COMPARISON OF COMMUNICATION OVERHEAD

Scheme	Sign	Combine		Verify	Trace					Open
	Signer	Enclave	Combiner	Verifier	Notary/Witness	SC	Enclave	Tracer	Admitter	Admitter
TAPS [14]	$\|\sigma_i\|$	N/A	$\|\sigma^{\mathsf{TAPS}}\|$	$\|b\|$	N/A	N/A	N/A	$\|\mathcal{S}\|$	N/A	N/A
HiTAPS [15]	$\|m\|, \|\hat{\sigma}_i\|, \|\hat{\mathcal{W}}_i\|$	N/A	$\|m\|, \|\sigma^{\mathsf{HiTAPS}}\|$	$\|b\|$	$\|\hat{\delta}_i\|$	N/A	N/A	$\|\mathcal{S}\|$	N/A	N/A
DeTAPS [16]	$\mathsf{TX}^{\mathsf{Sign}}$	$m, \bar{\sigma}, \mathsf{KASE.Enc}(mpk, gid, \mathcal{N})$	$\mathsf{TX}^{\mathsf{Comb}}$	$\|b\|$	$(td, \bar{\delta}_i)$	$t'\|\bar{\sigma}\|$	$\mathsf{PKE.Enc}(\mathcal{S})$	$\|\mathcal{S}\|$	N/A	N/A
TiMTAPS	$\mathsf{TX}^{\mathsf{Sign}}$	N/A	$\mathsf{TX}^{\mathsf{Comb}}$	$\|b\|$	N/A	N/A	N/A	$\|\mathcal{S}\|$	$\|td\|$	$\|\sigma\|$

and communication overhead of each link. In TiMTAPS, we are guided by IBE technology to reduce the computational overhead of Combine and Trace, and reduce the communication overhead of Trace, while maintaining decentralized and witness tracking. Overall, TiMTAPS optimizes both computational and communication overhead compared to related work while retaining core functionality.

VIII. DISCUSSIONS

A. Regarding T

The admitter addresses the issue of the combiner failing to provide the signature on schedule by utilizing time-lock puzzle technology. The admitter will initiate signature recovery only after confirming that the combiner has failed to provide the signature. In this case, the selection of time T required to open the time-lock puzzle is critically important. When *T* is too small, the existence of the time lock becomes meaningless, the admitter will shortly start solving the puzzles, making it scarcely different from directly sending a signature share to the admitter. When *T* is too big, the entities who are waiting for the complete signature will have to wait for a long time, which adversely affect the user experience.

B. Regarding Id_C

The intention of introducing id_C was to distinguish between different combiners in multi-session or multi-round execution environments, thereby increasing distinction in more complex systems. Although id_C does not directly participate in the cryptographic computations, it is embedded as part of the final TiMTAPS signature. In this way, we can differentiate between distinct combining processes and facilitate subsequent task allocation for tracers. Therefore, id_C is essential in the practical deployment.

C. Regarding Fiat-Shamir and Chain State

If the same sigma appears in two chains with different block hashes, the Fiat-Shamir hash avoids malleability by including timestamp into the proof generation process. The interactive proof generation process is possible, but at a cost of extra communication overhead. In this work, we have used the Fiat-Shamir heuristic to make the proof generation non-interactive. We believe it is possible to explore other non-interactive zero knowledge proof alternative methods while balancing security and efficiency.

It is possible that the chain state is not uniform due to network partition, sharding, 51% attack, consensus failure, and difference between light nodes and full nodes. However, we can solve it by resorting to a Finality mechanism that prevents the chain from forking, a state synchronization protocol, cross-shard coordination, and state detection and repair, etc.

D. Future Work

Our future work may focus on improving the interaction process between signers and the combiner, enabling the combiner to collect signature shares without revealing signer identities or *t*. This could be achieved by leveraging techniques such as anonymous communication and Trusted Execution Environment (TEE). These directions are expected to further mitigate the trust assumptions placed on the combiner in the current scheme.

IX. CONCLUSION

In this work, we propose TiMTAPS, a novel threshold signature scheme that enables timed tracking, verifiable combination, and message-based tracking. TiMTAPS retains the core principles of decentralization and traceability while enhancing the witnessing process through Identity-Based Encryption (IBE). By leveraging a time-lock puzzle for securely encapsulating signature information and constructing homomorphic commitment, the scheme ensures timed tracking and verifiability of the signature process. We formally prove the security and privacy of TiMTAPS, demonstrating its resilience against potential threats. Furthermore, our experimental results confirm the practicability and efficiency of the scheme. Specifically, in a setting where 10 signature sets are combined with a threshold value of 5, and each signature group consists of 20 signers, TiMTAPS achieves the threshold signature operation within 3.72 seconds for combination and 12.44 seconds for tracking. These results showcase the effectiveness of TiMTAPS in real-world scenarios, offering a efficient solution for decentralized threshold signatures with timed tracking capabilities.

REFERENCES

[1] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," in *Proc. 6th Annu. Int. Cryptol. Conf. (CRYPTO)*, Santa Barbara, CA, USA, 2007, pp. 307–315.

[2] I. Damgård and M. Koprowski, "Practical threshold RSA signatures without a trusted dealer," in *Proc. 18th Int. Conf. Theory Appl. Cryptograph. Techn.*, Innsbruck, Austria, 2001, pp. 152–165.

[3] T. Attema, R. Cramer, and M. Rambaud, "Compressed Σ-protocols for bilinear group arithmetic circuits and application to logarithmic transparent threshold signatures," in *Proc. 27th Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Singapore, Dec. 2021, pp. 526–556.

[4] R. Bacho and J. Loss, "On the adaptive security of the threshold BLS signature scheme," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Los Angeles, CA, USA, Nov. 2022, pp. 193–207.

[5] M. Li, Y. Chen, S. Zheng, D. Hu, C. Lal, and M. Conti, "Privacy-preserving navigation supporting similar queries in vehicular networks," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 1133–1148, Mar. 2022, doi: 10.1109/TDSC.2020.3017534.

[6] M. Li, Y. Chen, C. Lal, M. Conti, M. Alazab, and D. Hu, "Eunomia: Anonymous and secure vehicular digital forensics based on blockchain," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 225–241, Jan. 2023, doi: 10.1109/TDSC.2021.3130583.

[7] M. Li, Y. Chen, C. Lal, M. Conti, F. Martinelli, and M. Alazab, "Nereus: Anonymous and secure ride-hailing service based on private smart contracts," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 4, pp. 2849–2866, Jul. 2023, doi: 10.1109/TDSC.2022.3192367.

[8] M. Li et al., "Anonymous, secure, traceable, and efficient decentralized digital forensics," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 5, pp. 1874–1888, May 2024, doi: 10.1109/TKDE.2023.3321712.

[9] M. Li et al., "Accurate, secure, and efficient semi-constrained navigation over encrypted city maps," *IEEE Trans. Dependable Secure Comput.*, vol. 22, no. 3, pp. 2642–2658, May 2025, doi: 10.1109/TDSC.2024.3521396.

[10] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-Diffie–Hellman-group signature scheme," in *Proc. 6th Int. Workshop Theory Pract. Public Key Cryptography*, Miami, FL, USA, 2002, pp. 31–46.

[11] P.-A. Fouque and J. Stern, "Fully distributed threshold RSA under standard assumptions," in *Proc. 7th Int. Conf. Theory Appl. Cryptol. Inf. Secur. (ASIACRYPT)*, Gold Coast, QLD, Australia, 2001, pp. 310–330.

[12] S. Micali, K. Ohta, and L. Reyzin, "Accountable-subgroup multisignatures: Extended abstract," in *Proc. 8th ACM Conf. Comput. Commun. Secur.*, Philadelphia, PA, USA, 2001, pp. 245–254.

[13] J. Nick, T. Ruffing, and Y. Seurin, "MuSig2: Simple two-round Schnorr multisignatures," in *Proc. 41st Annu. Int. Cryptol. Conf.*, 2021, pp. 189–221.

[14] D. Boneh and C. Komlo, "Threshold signatures with private accountability," in *Proc. 42nd Annu. Int. Cryptol. Conf. (CRYPTO)*, Santa Barbara, CA, USA, 2022, pp. 551–581.

[15] M. Li, H. Ding, Q. Wang, Z. Zhang, and M. Conti, "Threshold signatures with private accountability via secretly designated witnesses," in *Proc. 29th Australas. Conf. Inf. Secur. Privacy (ACISP)*, Sydney, CA, Australia, 2024, pp. 389–407.

[16] M. Li et al., "Decentralized threshold signatures with dynamically private accountability," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 2217–2230, 2024.

[17] R. L. Rivest, A. Shamir, and D. A. Wagner, *Timelock puzzles and timed-release crypto*, Massachusetts Institute of Technology, Cambridge, MA, USA, Tech. Rep. 1, vol. 1996, pp. 1–9.

[18] M. Mahmoody, T. Moran, and S. Vadhan, "Time-lock puzzles in the random Oracle model," in *Proc. 31st Annu. Int. Cryptol. Conf. (CRYPTO)*, Santa Barbara, CA, USA, 2011, pp. 39–50.

[19] H. Lin, R. Pass, and P. Soni, "Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles," in *Proc. IEEE 58th Annu. Symp. Found. Comput. Sci. (FOCS)*, Berkeley, CA, USA, Oct. 2017, pp. 576–587.

[20] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Proc. 12th Annu. Int. Cryptol. Conf. (CRYPTO)*, Santa Barbara, CA, USA, 2007, pp. 139–147.

[21] (1993). *Timed-Release Crypto*. [Online]. Available: https://cypherpunks.venona.com/date/1993/02/msg00129.html

[22] D. Boneh and M. Naor, "Timed commitments," in *Proc. 20th Annu. Int. Cryptol. Conf. (CRYPTO)*, Santa Barbara, CA, USA, 2000, pp. 236–254.

[23] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[24] S. A. K. Thyagarajan, A. Bhat, G. Malavolta, N. Döttling, A. Kate, and D. Schröder, "Verifiable timed signatures made practical," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 1733–1750.

[25] Y. Sakai, K. Emura, G. Hanaoka, Y. Kawai, T. Matsuda, and K. Omote, "Group signatures with message-dependent opening," in *Proc. 5th Pairing-Based Cryptography*, Cologne, Germany, 2013, pp. 270–294.

[26] K. Ohara, Y. Sakai, K. Emura, and G. Hanaoka, "A group signature scheme with unbounded message-dependent opening," in *Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur.*, Hangzhou, China, May 2013, pp. 517–522.

[27] S. Xu, X. Huang, J. Yuan, Y. Li, and R. H. Deng, "Accountable and fine-grained controllable rewriting in blockchains," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 101–116, 2023.

[28] S. Xu et al., "Efficient ciphertext-policy attribute-based encryption with blackbox traceability," *Inf. Sci.*, vol. 538, pp. 19–38, Oct. 2020.

[29] S. Nakamoto. (2009). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[30] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the Internet of Things: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1676–1717, 2nd Quart., 2019.

[31] N. Yang, C. Tang, Z. Xiong, Q. Chen, J. Kang, and D. He, "SG-FCB: A Stackelberg game-driven fair committee-based blockchain consensus protocol," in *Proc. IEEE 44th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2024, pp. 403–414.

[32] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. 8th Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, USA, 2007, pp. 129–140.

[33] C. Delerablée and D. Pointcheval, "Dynamic threshold public-key encryption," in *Proc. 28th Annu. Int. Cryptol. Conf. (CRYPTO)*, Santa Barbara, CA, USA, 2008, pp. 317–334.

[34] B. Cui, Z. Liu, and L. Wang, "Key-aggregate searchable encryption (KASE) for group data sharing via cloud storage," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2374–2385, Aug. 2016.

[35] G. Malavolta and S. A. K. Thyagarajan, "Homomorphic time-lock puzzles and applications," in *Proc. 39th Annu. Int. Cryptol. Conf. (CRYPTO)*, Santa Barbara, CA, USA, 2019, pp. 620–649.

[36] N. Szabo. (1994). *Smart Contracts*. [Online]. Available: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html

[37] G. Wood. (2024). *Ethereum: A Secure Decentralised Generalised Transaction Ledger SHANGHAI VERSION 56b3dfd-2024-08-19*. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[38] I. Cascudo, I. Damgård, B. David, N. Döttling, and R. Dowsley, "Efficient UC commitment extension with homomorphism for free (and applications)," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2019, pp. 606–635.

[39] S. Heng and K. Kurosawa, "K-resilient identity-based encryption in the standard model," in *Proc. Cryptographers' Track RSA Conf.*, San Francisco, CA, USA, 2004, pp. 67–80.

[40] A. De Santis, S. Micali, and G. Persiano, "Non-interactive zero-knowledge proof systems," in *Proc. 4th Annu. Int. Cryptol. Conf. (CRYPTO)*, 1987, pp. 52–72.

[41] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Proc. 1st Annu. Int. Cryptol. Conf.*, vol. 31, Santa Barbara, CA, USA, 1985, pp. 469–472.

[42] S. Lee, M.-W. Shih, P. Gera, T. Kim, H. Kim, and M. Peinado, "Inferring fine-grained control flow inside SGX enclaves with branch shadowing," in *Proc. 26th USENIX Secur. Symp.*, 2017, pp. 557–574.

[43] S. Zhao, Q. Zhang, Y. Qin, W. Feng, and D. Feng, "Sectee: A software-based approach to secure enclave architecture using tee," in *Proc. 26th ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, London, U.K., Nov. 2019, pp. 1723–1740.

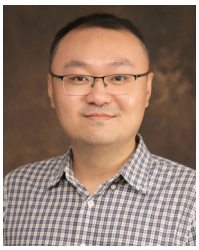
[44] G. Dessouky, T. Frassetto, and A.-R. Sadeghi, "HybCache: Hybrid side-channel-resilient caches for trusted execution environments," in *Proc. 29th USENIX Secur. Symp.*, Aug. 2020, pp. 451–468.

[45] I. Puddu, M. Schneider, M. Haller, and S. Capkun, "Frontal attack: Leaking control-flow in SGX via the CPU frontend," in *Proc. 30th USENIX Secur. Symp.*, 2020, pp. 663–680.

[46] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proc. 3rd Adv. Cryptol.*, Santa Barbara, CA, USA, 2007, pp. 186–194. Accessed: Jul. 1, 2025.

[47] *Ethereum*. [Online]. Available: https://ethereum.org/en

[48] Y. Lindell, "How to simulate it—A tutorial on the simulation proof technique," in *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, 2021. [Online]. Available: https://eprint.iacr.org/2016/046.pdf



Meng Li (Senior Member, IEEE) received the Ph.D. degree in computer science and technology from the School of Computer Science and Technology, Beijing Institute of Technology (BIT), China, in 2019. He is an Associate Professor and a Personnel Secretary with the School of Computer Science and Information Engineering, Hefei University of Technology (HFUT), China. He was a Post-Doctoral Researcher with the Department of Mathematics and the HIT Center, University of Padua, Italy, where he is with the Security and the PRIvacy Through Zeal (SPRITZ) Research Group led by Prof. Mauro Conti. From October 2020 to March 2021, he was sponsored by the ERCIM Alain Bensoussan Fellowship Program to conduct post-doctoral research at CNR, Italy, supervised by Prof. Fabio Martinelli. From September 2017 to August 2018, he was sponsored by China Scholarship Council (CSC), as a Joint Ph.D. Student at the Broadband Communications Research (BBCR) Laboratory, University of Waterloo and Wilfrid Laurier University, Canada, supervised by Prof. Xiaodong Lin (IEEE Fellow). From January 2025 to June 2025, he is supported by CSC, as a Visiting Scholar collaborating with the HIT Center, University of Padua, Italy, with Prof. Mauro Conti. His research interests include security, privacy, applied cryptography, blockchain, TEE, and the Internet of Vehicles. In this area, he has published 124 papers in topmost journals and conferences, including IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, TODS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, IEEE TRANSACTIONS ON SERVICES COMPUTING, COMST, IEEE S&P, USENIX Security, ACM MobiCom, and ACM ISSTA. He is a Senior Member of CIE, CIC, and CCF. He was a recipient of the 2024 IEEE HITC Award for Excellence (Early Career Researcher) and the 2025 IEEE TCSVC Rising Star Award. He has served as a TPC Member for conferences, including ICDCS, Inscrypt, ICICS, and TrustCom. He is an Associate Editor of IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, and IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT.

Hanni Ding received the B.E. degree in 2023. She is currently pursuing the M.S. degree with the School of Computer Science and Information Engineering, Hefei University of Technology. Her research interests include security, privacy, and digital signatures.

Yifei Chen (Graduate Student Member, IEEE) received the M.S. degree from the School of Computer Science and Information Engineering, Hefei University of Technology, in 2022, where he is currently pursuing the Ph.D. degree. His research interests include security, privacy, applied cryptography, TEE, blockchain, and the Internet of Vehicles.

Yan Qiao (Member, IEEE) received the Ph.D. degree in computer science from Beijing University of Posts and Telecommunications in 2012. She was a Post-Doctoral Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. She is currently an Associate Professor with the School of Computer Science and Information Engineering, Hefei University of Technology, China. She focuses on network monitoring, anomaly detection for time series, and artificial intelligence. She is an Associate Editor of IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT and COMNET.

Zijian Zhang (Senior Member, IEEE) received the Ph.D. degree from the School of Computer Science and Technology, Beijing Institute of Technology. He is currently an Associate Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. He was a Visiting Scholar with the Computer Science and Engineering Department, State University of New York at Buffalo, in 2015. His research interests include design of authentication and key agreement protocol and analysis of entity behavior and preference.

Liehuang Zhu (Senior Member, IEEE) received the M.S. degree from Wuhan University, Wuhan, China, in 2001, and the Ph.D. degree from Beijing Institute of Technology, Beijing, China, in 2004, both in computer science. He is a Full Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include data security and privacy protection, blockchain applications, and AI security. He has authored more than 200 journal and conference papers in these areas. He received the Best Paper Award at IEEE/ACM IWQoS 2017, IEEE TrustCom 2018, and IEEE IPCCC 2014. He has served as the Program Co-Chair for MSN 2017, IWWS 2018, and INTRUST 2014. He is an Associate Editor of IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, *IEEE Network*, and IEEE INTERNET OF THINGS JOURNAL. He was a Guest Editor of special issue of IEEE WIRELESS COMMUNICATIONS and IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS.

Mauro Conti (Fellow, IEEE), photograph and biography not available at the time of publication.