Decentralized Threshold Signatures With Dynamically Private Accountability

Meng Li, *Senior Member, IEEE*, Hanni Ding, Qing Wang, *Student Member, IEEE*, Mingwei Zhang, Weizhi Meng, *Senior Member, IEEE*, Liehuang Zhu, *Senior Member, IEEE*, Zijian Zhang, *Senior Member, IEEE*, and Xiaodong Lin, *Fellow, IEEE*

***Abstract*— Threshold signature is a fundamental cryptographic primitive used in many practical applications. As proposed by Boneh and Komlo (CRYPTO'22), TAPS is a threshold signature that is a hybrid of privacy and accountability. It enables a combiner to combine t signature shares while revealing nothing about the threshold t or signing quorum to the public and asks a tracer to track a signature to the quorum that generates it. However, TAPS has three disadvantages: it 1) structures upon a centralized model, 2) assumes that both combiner and tracer are honest, and 3) leaves the tracing unnotarized and static. In this work, we introduce Decentralized, Threshold, dynamically Accountable and Private Signature (DeTAPS) that provides decentralized combining and tracing, enhanced privacy against untrusted combiners (tracers), and notarized and dynamic tracing. Specifically, we adopt Dynamic Threshold Public-Key Encryption (DTPKE) to dynamically notarize the tracing process, design non-interactive zero knowledge proofs to achieve public verifiability of notaries, and utilize the Key-Aggregate Searchable Encryption to bridge TAPS and DTPKE so as to awaken the notaries securely and efficiently. In addition, we formalize the definitions and security requirements for DeTAPS. Then we present a concrete construction and formally prove its security and privacy. To evaluate the performance, we build a prototype based on SGX2 and Ethereum.**

Manuscript received 20 June 2023; revised 5 October 2023; accepted 6 November 2023. Date of publication 28 December 2023; date of current version 5 January 2024. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62372149, Grant U23A20303, and Grant 62002094; and in part by the National Key Research and Development Program of China under Grant 2021YFB2701202. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. George Theodorakopoulos. *(Corresponding author: Zijian Zhang.)*

Meng Li, Hanni Ding, Qing Wang, and Mingwei Zhang are with the Key Laboratory of Knowledge Engineering With Big Data, Ministry of Education, the School of Computer Science and Information Engineering, the Anhui Province Key Laboratory of Industry Safety and Emergency Technology, and the Intelligent Interconnected Systems Laboratory of Anhui Province, Hefei University of Technology, Hefei 230009, China (e-mail: mengli@hfut.edu.cn; hanniding@mail.hfut.edu.cn; qingwang@mail.hfut.edu.cn; mwzhang@mail.hfut.edu.cn).

Weizhi Meng is with the Department of Applied Mathematics and Computer Science, Cyber Security Section, Technical University of Denmark (DTU), 2800 Kongens Lyngby, Denmark (e-mail: weme@dtu.dk).

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

Xiaodong Lin is with the School of Computer Science, University of Guelph, Guelph, ON N1G 2W1, Canada (e-mail: xlin08@uoguelph.ca).

Digital Object Identifier 10.1109/TIFS.2023.3347968

***Index Terms*— Threshold signature, security, privacy, accountability.**

I. Introduction

A. Background

THRESHOLD signatures [1], [2] allow a group of n parties to sign a message if no less than t parties participate in the signing process. They are a crucial tool for many practical applications [3], [4], [5]. For instance, the initiation of a new financial project calls for at least t enterprises to collaborate. Among the threshold signatures, there are two types of threshold signatures standing out: Accountable Threshold Signature (ATS) and Private Threshold Signature (PTS). ATS is a kind of threshold signature scheme where the signature can identify the original signing group that generated the signature. Specifically, a tracing algorithm takes as input a message, a valid signature on the message, and the public key to output a group of signers that generated the signature [6], [7]. A PTS is a kind of threshold signature scheme where the signature on a message m reveals nothing about t or the quorum of t original signers [8], [9]. Besides unforgeability, these two signatures offer complete accountability and complete privacy for the signing quorum, respectively.

B. Existing Work

A recent work Threshold, Accountable, and Private Signature (TAPS) [10] proposed by Boneh and Komlo (CRYPTO'22) has achieved both accountability and privacy. In TAPS, a key generation function takes n and t as input, and generates a public key pk and n private keys $sk_1, sk_2, \cdots, sk_n$ for the n signers; during the signing process, each signer from a quorum of t signers $\mathcal{S}$, holding a private key sk, generates a signature share σ_i; a combiner holding a combining key sk_c uses $\{\sigma_i\}_{i=1}^{n}$ to generate a complete signature σ; a signature verification function takes as input pk, m, and σ to output accept or reject; a tracer (or anyone) with a tracing key sk_t can trace a signature to the quorum that generates it. The benefits of TAPS are remarkable: the signing group keeps the sk_t secret so that t and $\mathcal{S}$ remain private from the public, but the t signers are accountable in case of misbehaviors [11].

C. Motivations and New Goals

Our motivations come from a real-world scenario in financial areas. For example, a group of companies are in a

long-term collaboration and at least t companies are required to initiate a new and confidential project by co-signing a new contract. A "third" party, which is not fully trustworthy, is responsible for combining their signatures and generate a threshold signature as a collaboration proof. Meanwhile, in case of any criminal activities or misbehaviors, $t!'$ entities, such as police department, finance department, and insurance company, are required to participate in the signing process as a witness. Furthermore, if one of the t companies engages in some criminal activities, its identity will be recovered by the t' witnesses and a not-so-trustworthy tracer, and it will be sanctioned according to law or regulation. Combined with the observations on TAPS, we acquire **four motivations**. **M1. Centralized combining and tracing**. The role of combiner and tracer is important to generating and tracing a complete signature. However, the centralized setting is prone to a single point of failure. **M2. Untrusted combiner**. The combining key sk_c is kept by the only combiner that could be untrusted, e.g., lose or leak the key. The threshold t is also exposed to the combiner. As designed in TAPS, t is part of privacy and is hidden from the public. Therefore, we take the privacy one step further by assuming an untrusted combiner. **M3. Untrusted tracer**. Similarly, the tracing key sk_t is kept by an untrusted tracer and t is exposed. **M4. Unnotarized and static tracing**. The tracing key sk_t is kept by the sole tracer that can use sk_t to recover any quorum of t signers. We argue that the tracing process is a sensitive process that should be notarized by a dynamic and relevant group of notaries, i.e., t' notaries or witnesses [12], [13]. Meanwhile, the value of t' varies according to the matter and relevant authorities. The idea resembles the one in threshold encryption where a ciphertext can only be decrypted when at least t' users cooperate [14].

These four motivations have driven us to provide enhanced security and privacy in threshold signatures using a decentralized approach, i.e., decentralized threshold signatures with dynamically private accountability. Namely, we have four new goals as follows. **G1. Enhanced security against a single point of failure**. The threshold signature system should be secure in a decentralized manner such that one (a small number of) combiner/tracer's breakdown does not affect the whole system. **G2. Enhanced privacy against untrusted combiners**. The threshold signature system is privacy-preserving during the signing process. Specifically, not only the quorum of t original singers, but t, sk_t, and t' are hidden from combiners. **G3. Enhanced privacy against untrusted tracers**. The threshold signature system is privacy-preserving during the tracing process. To be specific, t, sk_c, and t' are hidden from tracers. **G4. Notarized and dynamic tracing**. The tracing process should be notarized by t' parties among a group of authorities. The value of t' is a variable parameter, which is related to the specific tracing requirement.

Remark 1 (Privacy of t signers after tracing): We notice that once a tracer has traced a complete signature to its t signers, the signers' identities as well as t are revealed to the tracer. This looks contradictory to **G3** where we protect t and make **G3** only applicable to the realm before tracing. However, we can choose to protect t signers from tracers (will be explained Section IV).

D. Our Approach

To achieve the four abovementioned goals, we propose an approach as follows. (1) We transit the centralized model of TPAS into a decentralized one by using a Consortium Blockchain (CB) [15], [16] to distribute the combining and tracing capabilities. Each blockchain node can be either a combiner or a tracer such that the combiner (tracer) actually performing the combining (tracing) is determined by the underlying consensus mechanism. In this way, an adversary cannot predict such a performer to attack. (2) We protect t and t' from the untrusted combiners and untrusted tracers during the combining and tracing by deploying a Trusted Execution Environment (TEE) [17], [18], [19] on combiners and tracers. The combining and tracing will be conducted within an enclave, over which the combiners and tracers have no control over the data inside. (3) We propose "dynamically private accountability", i.e., limit the tracing capability of untrusted tracers by asking another quorum of t' parties to notarize the tracing process. We denote this quorum as $\mathcal{N} = \{N_1, N_2, \cdots, N_{t'}\}$. Specifically, the tracer can only trace from a complete signature to its t signers only if there are t' notaries allow it. This is realized by adopting Dynamic Threshold Public-Key Encryption (DTPKE) [14] to designate t' notaries for the tracing process.

In summary, we introduce a new type of threshold signature scheme, called DeTAPS, that provides dynamic accountability while maintaining full privacy for the signing quorum and notarizing quorum. A **Decentralized, Threshold, dynamically Accountable and Private Signature** scheme, or simply **DeTAPS**, works as follows: (i) a key generation procedure generates a public key pk and n private keys $\{sk_1, sk_2, \cdots, sk_n\}$, a combining key sk_c, and a tracing key sk_t, (ii) a signing protocol among a quorum of t signers and a combiner generate a signature σ on a message m, (iii) a signature verification algorithm takes as input pk, m, and σ and outputs true or false, and (iv) a tracing algorithm takes as input sk_t, m, and σ, and outputs the original quorum of t signers. For security model, we assume that the combiners and tracers are malicious, which are not allowed to know t or t'. We define the precise syntax for the DeTAPS scheme and the security requirements in Section III.

E. Technical Challenges

Given the general approach, we are still faced with three technical challenges when constructing DeTAPS. *C1. How to securely select the t' notaries while guaranteeing public verifiability?* In this work, we ask the t signers to choose t' notaries whose identities are kept secret. In the meantime, we have to guarantee public verifiability of the t' notaries, i.e., there are enough authenticated notaries selected by the t signers during combining. *C2. How to securely awaken the t' notaries to the call for partially decryption of encrypted threshold signatures when necessary?* There are several technical candidates for solving this problem. (1) Encrypt-and-Decrypt: It is workable, but time consuming and clumsy. (2) Private Set Intersection (PSI) [20], [21]: It provides strong security but requires more than one interaction, which results in high costs. (3) Attribute-Based Encryption (ABE) [22], [23]: It achieves fine-grained

access control but incurs high computational costs. *C3. How to allow a notary to efficiently locate the encrypted signatures related to himself from all the ciphertexts on the CB?* Some technical candidates are as follows. (1) Indistinguishable Bloom Filter (IBF) [24], [25]: It is efficient but needs to share a set of keys between signers and notaries. (2) Designated Verifier Signature [26], [27]: It requires additional signing by the signers and it cannot provide confidentiality. To tackle C1, we design Non-Interactive Zero Knowledge Proofs (NIZKPs) to enable the public to verify the t' notaries in a secure manner. To overcome C2 and C3, we utilize the Key-Aggregate Searchable Encryption (KASE) [28] as a bridge between TAPS and DTPKE to reconcile security and efficiency [29], [30], [31].

We provide some details on how we construct DeTAPS. **Setup**. We assume that any quorum of t signers have communicated with each other via face to face or a secure channel to determine t' notaries $\mathcal{N} = \{N_1, N_2, \cdots, N_{t'}\}$. Each quorum of t signers has a unique and random signer group identifier $gid \in \mathcal{G}$ in each signing period. This can be done by asking a representative of each quorum to anonymously write a random number on the blockchain. $\mathcal{G}$ will be updated in future periods. Each notary has a pseudo-identity pid. A KASE aggregation key k_a is generated in the beginning for each notary. **Sign**. Each signer of a quorum of t signers generates a signature share σ_i on the same message m and sends its ciphertext to the CB. **Combine**. During combining, the enclave E within the combiner C encrypts σ to be an encrypted threshold signature $\overline{\sigma}$ by using the combining key sk_c. After combining, E computes an index ind of $\mathcal{N}$. **Trace**. Upon a tracing call, each related notary computes a trapdoor td by using k_a and pid. The index and trapdoor are sent to a smart contract that searches on ind with td to retrieve a matched $\overline{\sigma}$ to the requesting notary. The notary sends a partial decryption of $\overline{\sigma}$ to the CB. Only if the designated t' notaries are awaken to perform partially decryption, can a tracer T trace within its enclave to the original quorum of t tracers by using the tracing key sk_t. In addition, the encrypted threshold signature can be verified by the public.

F. Our Contributions

Our contributions are summarized as follows.

- We design a decentralized framework for threshold signatures to distribute the combining (tracing) capabilities to multiple combiners (tracers).
- We design a TEE-based execution engine to secure the combining (tracing) process against untrusted combiners (tracers).
- We adopt DTPKE to dynamically notarize the tracing process and integrate TAPS with DTPKE by using KASE to awaken the notaries.
- We formally prove the security and privacy of DeTAPS. We build a prototype and evaluate its performance.

Paper Organization. The paper is organized as follows. Section II briefly reviews some preliminaries. Section III formalizes the system model, security, and privacy of DeTAPS. Section IV describes DeTAPS. Section V analyzes its security and privacy. Section VI evaluates the performance of DeTAPS. Section VII concludes this paper.

II. Preliminaries

In this section, we briefly review some preliminaries that work as building blocks.

A. ATS

An accountable threshold signature is a tuple of five polynomial time algorithms (KeyGen, Sign, Combine, Verify, Trace). $\mathsf{KeyGen}(1^\lambda, n, t)$ is a probabilistic algorithm that takes security parameter λ, the number of signers n, and the threshold t as input, output the private key set sk_i as well as the combined public key pk. $\mathsf{Sign}(sk_i, m)$ is a probabilistic algorithm that uses private key set sk_i to sign message m to output a signatures set σ_i. $\mathsf{Combine}(pk, m, \mathcal{S}, \{\sigma_i\}_{i\in\mathcal{S}})$ is a deterministic algorithm takes as input the public key pk, the message m, the signer set $\mathcal{S}$, and the output σ_i of previous algorithm as inputs to output ATS signature σ_m. $\mathsf{Verify}(pk, m, \sigma_m)$ is a deterministic algorithm that uses public key pk and message m to verify whether σ_m is a valid ATS signature, with valid outputs 1 and invalid outputs 0.

We use the Schnorr scheme in [6] as the ATS for DeTAPS. An ATS is secure if it is unforgeable and accountable, i.e., if for every Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$, the function $\mathbf{Adv}^{\text{forg}}_{A,\mathsf{ATS}}$ of winning an unforgeability and accountability attack game is a negligible function of λ [10].

B. DTPKE

DTPKE is a kind of threshold public-key encryption where a ciphertext can be decrypted when at least t users collaborate. More importantly, the size of the decryptor set and the threshold are not fixed during the setup, but at the encryption time. DTPKE is a tuple of seven algorithms (Setup, Join, Encrypt, ValidateCT, ShareDecrypt, ShareVerify, Combine), $\mathsf{Setup}(1^\lambda) \rightarrow (mk, ek, vk, ck)$ is a probabilistic algorithm that takes security parameter λ as input to output master secret key mk, encryption key ek, combing key ck, and verification key vk. $\mathsf{Join}(mk, id) \rightarrow (usk, upk, uvk)$ is a probabilistic algorithm that takes the master secret key mk, and the identity id of new user as input to output the user's private key usk, the user's public key upk, and the user's verification key upk. $\mathsf{Enc}(ek, \mathcal{U}, t', m) \rightarrow c$ is a probabilistic algorithm that takes encryption key ek, a set $\mathcal{U}$ of users, threshold t', and message m to output a c. $\mathsf{ValidateCT}(ek, \mathcal{U}, t', c) \rightarrow \{0, 1\}$ is a deterministic algorithm that takes encryption key ek, set $\mathcal{U}$, a threshold t', and a ciphertext c as input to check whether c is valid, if valid output 1, else output 0. $\mathsf{ShareDecrypt}(pid, usk, c) \rightarrow \sigma_a^j$ is a deterministic algorithm that takes user id, user's private key usk, and ciphertext c to output a decryption share σ_a^j or $\perp$. $\mathsf{ShareVerify}(vk, pid, uvk, c, \sigma_a) \rightarrow \{0, 1\}$ is a deterministic algorithm that takes verification key vk, user id, user's verification key uvk, ciphertext c, decryption share σ_a to check whether σ_a is a valid decryption share, if valid output 1, else output 0. $\mathsf{Combine}(ck, \mathcal{U}, t', c, \mathcal{N}, \{\sigma_a^j\}_{j\in[t']}) \rightarrow$ is a deterministic algorithm that takes combining key ck, set $\mathcal{U}$, threshold t', ciphertext c, subset $\mathcal{N} \subseteq \mathcal{U}$, and a decryption share set $\{\sigma_a^j\}_{j\in[t']}$ to output a signature σ_m.

We use the scheme in [14] as the DTPKE for DeTAPS. Its non-adaptive adversary, non-adaptive corruption, chosen-plaintext attacks (IND-NAA-NAC-CPA) security is based on the Multi-sequence of Exponents Diffie-Hellman (MSE-DDH) assumption, where $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{A},\mathsf{DTPKE}}(l,m,t') \leq \mathbf{Adv}^{\text{mse-ddh}}(l,m,t')$ [14], [32], [33]. For succinctness, we write $\mathsf{Enc}(ek,\mathcal{N},m)$, $\mathsf{ValidateCT}(ek,\mathcal{N},c)$, and $\mathsf{Combine}(ck,\mathcal{N},c,\{\sigma^j_m\}_{j\in[t']})$ as a shorthand for the three functions.

C. KASE

KASE allows a data owner to share a set of files with a group of selected data users, which can perform keyword search over the set of files. Specifically, the data owner distributes an aggregate key to the data users. Then, the data user sends an aggregate trapdoor to conduct keyword search over the set of files. KASE is a tuple of seven algorithms (Setup, Keygen, $\mathsf{Encrypt}$, $\mathsf{Extract}$, $\mathsf{Trapdoor}$, Adjust, Test). $\mathsf{Setup}(\lambda,n)\rightarrow(\mathcal{B},\mathcal{PK},H)$ is a probabilistic algorithm that takes a security parameter λ and maximum possible number of documents n as input to output the system parameters $(\mathcal{B},\mathcal{PK},H)$. $\mathsf{KeyGen}(\lambda)\rightarrow(mpk,msk)$ is a probabilistic algorithm that takes a security parameter λ as input to output a pair of keys (mpk,msk). $\mathsf{Extract}(msk,\mathcal{S})\rightarrow k_a$ is a deterministic algorithm that takes owner's master-secret key msk, and subset $\mathcal{S}$ which contains the indices of documents as input to output a aggregate key k_a. $\mathsf{Enc}(mpk,i)\rightarrow(c_1,c_2,c_\omega)$ is a probabilistic algorithm that takes owner's master-public key mpk, file index i, and keyword subset $\mathcal{S}$ as input to output ciphertext (c_1,c_2,c_ω). $\mathsf{Trapdoor}(k_a,\omega)$ is a deterministic algorithm that takes aggregate key k_a, and keyword ω as input to output trapdoor td. $\mathsf{Adjust}(\mathcal{B},\mathcal{PK},H,i,\mathcal{S},td)\rightarrow td_i$ is a deterministic algorithm that takes system parameters $(\mathcal{B},\mathcal{PK},H)$, file index i, subset $\mathcal{S}$, trapdoor td as input to output the right trapdoor td_i. $\mathsf{Test}(td_i,(c_1,c_2,c_\omega),i)\rightarrow\{0,1\}$ is a deterministic algorithm that takes right trapdoor td_i, ciphertext (c_1,c_2,c_ω), and file index i to check whether c_ω is valid, if valid output 1, else output 0. We use the scheme in [28] the KASE for DeTAPS. Specifically, the functions of the KASE in the DeTAPS operate as follows:

We use the scheme in [28] the KASE for DeTAPS. KASE achieves controlled searching and query privacy based on the Discrete Logarithm (DL) assumption and the Bilinear Diffie-Hellman Exponent (BDHE) assumption [32].

D. PKE

A public key encryption scheme PKE is a triple of algorithms (KeyGen, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$). $\mathsf{KeyGen}(1^\lambda)\rightarrow(pk,sk)$ is a probabilistic algorithm that takes as input a security parameter λ to output a public key pk and a secret key sk. $\mathsf{Encrypt}(pk,m)\rightarrow c$ is a probabilistic algorithm that encrypts a message m using pk and finally outputs a ciphertext c. $\mathsf{Decrypt}(sk,c)\rightarrow m$ is a deterministic algorithm that takes c and sk as input and outputs a plaintext m.

We use EIGamal as the PKE for DeTAPS. A PKE scheme is semantically secure if for every PPT adversary $\mathcal{A}$, $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{A},\text{PKE}}(\lambda)$ is negligible [34].

E. COM

A commitment scheme is a pair of algorithms (Commit, Verify). $\mathsf{Commit}(x,r)\rightarrow com$ is a deterministic algorithm that takes x and random number r as input and outputs a commitment com. $\mathsf{Verify}(x,r,com)\rightarrow\{0,1\}$ is a deterministic algorithm that determines whether the commitment is valid, if $com'=com$, output 1, else output 0.

We use Pedersen commitment as the COM algorithm for DeTAPS. A COM scheme is secure if it is unconditionally hiding and computationally binding, i.e., for every PPT adversary $\mathcal{A}$, $\mathbf{Adv}^{\text{bind}}_{\mathcal{A},\text{COM}}(\lambda)$ is negligible.

F. SIG

A signature scheme SIG is a triple of algorithms (KeyGen, Sign, Verify). $\mathsf{KeyGen}(1^\lambda)\rightarrow(pk,sk)$ is a probabilistic algorithm that takes as input a security parameter λ to output a public key pk and a secret key sk. $\mathsf{Sign}(sk,m)\rightarrow\sigma$ is a probabilistic algorithm run by a signer with a signer key sk to output a signature σ on m. $\mathsf{Verify}(pk,m,\sigma)\rightarrow\{0,1\}$ is a deterministic algorithm that verifies the signature σ on a message m to decide whether to accept or reject σ.

We use the ECDSA (Elliptic Curve Digital Signature Algorithm) signature scheme as the SIG for DeTAPS. A SIG scheme is strongly unforgeable if for every PPT adversary $\mathcal{A}$, $\mathbf{Adv}^{\text{euf-cma}}_{\mathcal{A},\text{SIG}}(\lambda)$ is negligible.

G. NIZKP

A non-interactive zero-knowledge proof protocol enables a prover to convince a verifier that a certain statement is true, without revealing any information about the underlying information for its truth. It involves two algorithms (P,V) invoked as $\pi\leftarrow\mathsf{Prove}(1^\lambda,m)$, $b\leftarrow\mathsf{Verify}(\pi)$.

H. Intel SGX2

Software Guard eXtensions (SGX) is a hardware extension of Intel Architecture that enables an application to establish a protected execution space, i.e., an enclave [35], [36], [37], [38]. SGX stores enclave pages and SGX structures in the protected memory called Enclave Page Cache (EPC). SGX guarantees confidentiality of code/data and detection of an integrity violation of an enclave instance from software attacks. SGX allows one to verify that a piece of software has been correctly instantiated on the platform via attestation. Since SGX imposes limitations regarding memory commitment and reuse of enclave memory, Intel introduces SGX2 to extend the SGX instruction set to include dynamic memory management support for enclaves [18], [19]. SGX2 instructions offer software with more capability to manage memory and page protections from inside an enclave while preserving the security of the SGX architecture and system software.

For formal foundation for Secure Remote Execution (SRE) of enclaves, Subramanyan et al. [39] addressed the formal modeling and verification of enclave platforms via three steps. First, they defined the properties required for SRE of enclaves. Second, they presented Trusted Abstract Platform (TAP), an idealization of enclave platforms together with a parameterized adversary model. They gave machine-checked proofs exhibiting that the TAP provided SRE against the adversaries. Third, they gave machine-checked proofs showing that formal models of two proposals for trusted hardware platforms offered SRE.

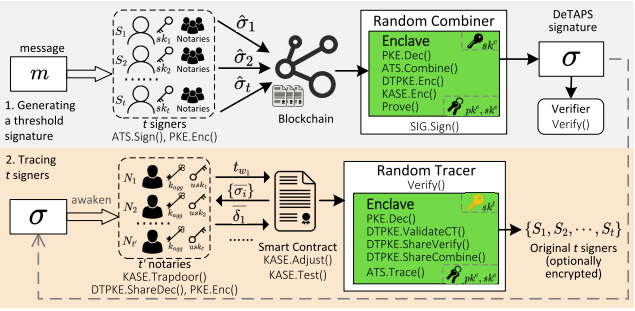


Fig. 1. System Architecture of DeTAPS.

I. Consortium Blockchain

As an underlying technique in Bitcoin, blockchain is a ledger recording transactions among users who do not fully trust each other in a decentralized network. The transactions are packed into separate blocks by a set of nodes using a predefined consensus algorithm, and the blocks are sequentially linked into a chain by their cryptographic hashes. Nodes participate in creating new blocks to compete for some rewards such as financial incentives. Consortium blockchain is a specific blockchain maintained by a group of authorized entities [40], [41]. For participation, only qualified parties are allowed to access and maintain the blockchain. It aims to secure transactions between users who do not fully trust each other but work collaboratively toward a common goal. Its consensus process is controlled by the authorized entities. For our setting of DeTAPS, a consortium is perfect regarding system model and security model. This is why we choose a consortium blockchain to lay the communication basis.

III. Decentralized, Threshold, Dynamically Accountable and Private Signature

In this section, we formalize the notion of DeTAPS, including system model, unforgeability, accountability, and privacy.

A. System Model

The system architecture of DeTAPS is depicted in Fig. 1. It consists of signer, combiner, notary, tracer, and consortium blockchain. We list the key notations in Table I.

1) Signer: When a group of t signers $\mathcal{S} = \{S_1, S_2, \cdots, S_t\}$ prepare to generate a signature on a message m, they request the pseudo-identity from t' parties $\mathcal{N} = \{N_1, N_2, \cdots, N_{t'}\}$ as notaries. Then, each group manager generates a signer group identifier $gid \in \mathcal{G}$ in current signing period and reports it to CB. Next, each signer sends a signature share on m to CB.

2) Combiner: Each combiner C_i is equipped with an enclave E_i. C_i has a pair of signing keys and E_i has a pair of encryption keys. The combining key is secured in the E_i. After being elected as a winning node, C_i retrieves all signature shares from the CB and the E_i decrypts them to collect related signature shares and combine them into a complete signature. Next, it generates an encrypted signature via DTPKE, computes an index via KASE, and constructs a non-interactive zero knowledge proof. Finally, C_i signs the message, encrypted signature, index, and the proof.

Remark 2 (For overlooked signature shares): During combining in an enclave, there will be overlooked signature shares that exist for the protection of t. We do not cast them out of the enclave and retrieve them for the next combining. Instead, we store these shares in the enclave, which has an enough storage space.

TABLE I
Experimental Parameters

Notation	Meaning
λ	security parameter
n	number of signers
n_1	number of combiners
n_2	number of tracers
n_3	number of notaries
t	threshold
sk_i	key for i-th signer
$\{sk_i^s\}_{i=1}^{n_1}$	n_1 signing keys
$\{sk_i^c\}_{i=1}^{n_1}$	n_1 combining keys
$\{sk_i^t\}_{i=1}^{n_2}$	n_2 tracing keys
gid	signer group identifier
$\mathcal{G}$	set of signer groups
k_a	aggregate key
$\mathcal{N}$	set of notaries
$\mathcal{S}$	signing quorum
$\mathcal{M}$	message space
m	message for signing
σ_i, $\widehat{\sigma}_i$	ATS signature share, encryption of σ
σ^m, $\overline{\sigma}$	Combined signature, σ^m under DTPKE
σ	DeTAPS signature

Remark 3 (Why multiple combiners?): There is only one combiner in TAPS, which is prone to the general problems of centralized model [42]. In DeTAPS, we have distributed such an ability to all blockchain nodes that hold a combining key in an enclave. The combining process will be assigned to a randomly node based on the blockchain consensus result. In this way, an adversary will have more difficulty in compromising the actual combiner in current period. This idea also applies to why we have multiple tracers.

3) Notary: There is a set of parties working as notaries. In real life, they can be a notary office or a local authority. Each notary N_i has a pseudo-identity, shares an aggregate key, and acts as a user (not necessarily a blockchain node) in the CB network. Upon a tracing call, each notary N_i computes a trapdoor. N_i sends the trapdoor to the CB and waits for matching results. If there is a decryption task, N_i verifies the results and then generates a decryption share of the encrypted signature. Next, N_i sends an encrypted response to the CB.

4) Tracer: Each tracer T_j is also equipped with an enclave E_j. The tracing key is secured in the E_j. After being elected as a winning node, T_j retrieves all encrypted decryption shares from the CB and the E_j decrypts them to verify decryption shares. Finally, E_j collects t' related valid shares to combine a complete signature and trace the original quorum of t tracers.

5) Consortium Blockchain: DeTAPS is built upon a decentralized framework where a CB records all the transactions sent by signers, combiners, notaries, and tracers. There are two pools on the CB: a signature share pool $\mathcal{SSL}$ for combiners to track and a decryption share pool $\mathcal{DSL}$ for tracers to monitor. Each of them is deployed on a Smart Contract (SC).

$\mathbf{Exp}^{\mathbf{forg}}$

1. $(n, n_1, n_2, t, \mathcal{S}, \mathsf{state}) \xleftarrow{\$} \mathcal{A}(1^\lambda)$ where $t \in [n]$, $\mathcal{S} \subseteq [n]$
2. $(PK, (sk_1, \cdots, sk_n), \{sk_i^s, sk_i^c\}_{i=1}^{n_1}, \{sk_i^t\}_{i=1}^{n_2}, \mathcal{G}, k_a) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, n, n_1, n_2, t)$
3. $(m', \sigma') \xleftarrow{\$} \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(PK, (sk_1, \cdots, sk_n), \{sk_i^s, sk_i^c\}_{i=1}^{n_1}, \{sk_j^t\}_{j=1}^{n_2}, \mathcal{G}, k_a, \mathsf{state})$

where $\mathcal{O}_1(\mathcal{S}_i, m_i)$ returns the signature shares $\{\mathsf{Sign}(sk_j, m_i, \mathcal{S}_i, \mathcal{N})\}_{j\in\mathcal{S}_i}$

Winning condition:

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

Let $\mathcal{S} \leftarrow \cup\mathcal{S}_i$, union over all queries to $\mathcal{O}_1(\mathcal{S}_i, m')$, let $\mathcal{S}_t \leftarrow \mathsf{Trace}(sk_i^t, m', \sigma')$

Output 1 if $\mathsf{Verify}(PK, m', \sigma') = 1$ and either $\mathcal{S}_t \nsubseteq \mathcal{S} \cup \mathcal{S}'$ or if $\mathcal{S}_t = \mathsf{fail}$

Fig. 2. Experiment of unforgeability and accountability.

$\mathbf{Exp}^{\mathbf{privP}}$

1. $b_1 \xleftarrow{\$} \{0,1\}$, $b_2 \xleftarrow{\$} \{0,1\}$
2. $(n, n_1, n_2, t_0, t_1, t_0', t_1', \mathcal{S}_0, \mathcal{S}_1, \mathcal{N}_0, \mathcal{N}_1, \mathsf{state}) \xleftarrow{\$} \mathcal{A}(1^\lambda)$, $t_0, t_1 \in [n], t_0', t_1' \in [n_3]$
3. $(PK, (sk_1, \cdots, sk_n), \{sk_i^s, sk_i^c\}_{i=1}^{n_1}, \{sk_j^t\}_{j=1}^{n_2}, \mathcal{G}, k_a) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, n, n_1, n_2, t_{b_1})$
4. $(b_1', b_2') \leftarrow \mathcal{A}^{\mathcal{O}_2(\cdot,\cdot,\cdot),\mathcal{O}_3(\cdot,\cdot,\cdot,\cdot,\cdot),\mathcal{O}_4(\cdot,\cdot)}(PK, \mathsf{state})$
5. Output $(b_1' = b_1) \wedge (b_2' = b_2)$.

where $\mathcal{O}_2(\mathcal{N}_0, \mathcal{N}_1, m||\sigma||gid)$: $\widehat{\sigma} \leftarrow \mathsf{PKE.Enc}(pk^e, m||\sigma||\mathcal{N}_{b_2}||gid)$

$\mathcal{O}_3(\mathcal{S}_0, \mathcal{S}_1, \mathcal{N}_0, \mathcal{N}_1, m)$: $\sigma \xleftarrow{\$} \mathsf{Combine}(sk_i^c, m, \mathcal{S}_{b_1}, \{\mathsf{Sign}(sk_j, m, \mathcal{S}_{b_1}, \mathcal{N}_{b_2}\}_j)$
for $\mathcal{S}_0, \mathcal{S}_1 \subseteq [n]$, $|\mathcal{S}_0| = t_0$ and $|\mathcal{S}_1| = t_1$, $\mathcal{N}_0, \mathcal{N}_1 \subseteq [n_3]$, $|\mathcal{N}_0| = t_0'$ and $|\mathcal{N}_1| = t_1'$

$\mathcal{O}_4(m, \sigma)$ returns $\mathsf{Trace}(sk_i^t, m, \sigma)$.

Restriction: if σ is computed from $\mathcal{O}_3(\cdot,\cdot,\cdot,\cdot,m)$, $\mathcal{A}$ never queries $\mathcal{O}_4$ at (m, σ).

$\mathbf{Exp}^{\mathbf{privS}}$

1. $b_1 \xleftarrow{\$} \{0,1\}$, $b_2 \xleftarrow{\$} \{0,1\}$
2. $(n, n_1, n_2, t, t_0', t_1', \mathcal{S}_0, \mathcal{S}_1, \mathcal{N}_0, \mathcal{N}_1, \mathsf{state}) \xleftarrow{\$} \mathcal{A}(1^\lambda)$ where $t \in [n], t_0', t_1' \in [n_3]$
3. $(PK, (sk_1, \cdots, sk_n), \{sk_i^s, sk_i^c\}_{i=1}^{n_1}, \{sk_j^t\}_{j=1}^{n_2}, \mathcal{G}, k_a) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, n, n_1, n_2, t)$
4. $(b_1', b_2') \leftarrow \mathcal{A}^{\mathcal{O}_2(\cdot,\cdot,\cdot),\mathcal{O}_4(\cdot,\cdot)}(PK, (sk_1, sk_2, \cdots, sk_n), \mathsf{state})$
5. Output $(b_1' = b_1) \wedge (b_2' = b_2)$.

Restriction: $|\mathcal{S}_0| = |\mathcal{S}_1| = t$, $\mathcal{N}_0, \mathcal{N}_1 \subseteq [n_3]$, $|\mathcal{N}_0| = t_0'$, $|\mathcal{N}_1| = t_1'$

Fig. 3. Two experiments of privacy against the public and the signers.

Definition 1: A decentralized, threshold, dynamically accountable and private signature, or DeTAPS, is a tuple of five polynomial time algorithms $\Pi = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{Verify}, \mathsf{Trace})$ as shown in Fig. 2 where

– $\mathsf{Setup}(1^\lambda, n, n_1, n_2, t) \rightarrow (PK, (sk_1, sk_2, \cdots, sk_n), \{sk_i^c\}_{i=1}^{n_1}, \{sk_i^c\}_{i=1}^{n_1}, \{sk_j^t\}_{j=1}^{n_2}, \mathcal{G}, k_a)$ is a probabilistic algorithm that takes as input a security parameter λ, the number of signers n, the number of combiners n_1, the number of tracers n_2, and a threshold t to output a public key PK, n signer keys $\{sk_1, sk_2, \cdots, sk_n\}$, n_1 signing keys $\{sk_i^s\}$, n_1 combining keys $\{sk_i^c\}$, n_2 tracing keys $\{sk_j^t\}$, a set of signer groups $\mathcal{G}$, and an aggregate key k_a.

- $\mathsf{Sign}(sk_i, m, \mathcal{S}, \mathcal{N}) \rightarrow \widehat{\sigma_i}$ is a probabilistic algorithm run by a signer with a signer key sk_i and a set of notaries $\mathcal{N}$ to generate an encrypted signature share $\widehat{\sigma_i}$ on message m in message space $\mathcal{M}$.
- $\mathsf{Combine}(sk_i^c, m, \mathcal{S}, \{\widehat{\sigma_j}\}_{j\in\mathcal{S}}) \rightarrow \sigma$ is a probabilistic algorithm run by a combiner with a combining key sk_i^c, a message m, a signing quorum $\mathcal{S} = \{S_1, S_2, \cdots, S_t\}$, and t encrypted signature shares $\{\widehat{\sigma_j}\}_{j\in\mathcal{S}}$. If the shares are valid, $\mathsf{Combine}$ outputs a DeTAPS signature σ.
- $\mathsf{Verify}(PK, m, \sigma) \rightarrow \{0, 1\}$ is a deterministic algorithm that verifies the signature σ on a message m with respect to the public key PK.
- $\mathsf{Trace}(sk_i^t, m, \sigma) \rightarrow \mathcal{S}$ is a deterministic algorithm run by a tracer with a tracing key sk_i^t, a message m, and a signature σ. If σ is valid, Trace outputs a set $\mathcal{S}$ who have generated σ. Otherwise, it outputs a symbol $\perp$.
- For *correctness*, we require that for all $t \in [n]$, all t-size sets $\mathcal{S}$, all $m \in \mathcal{M}$, and $(PK, (sk_1, sk_2, \cdots, sk_n), \{sk_i^c\}_{i=1}^{n_1}, \{sk_j^t\}_{j=1}^{n_2}, \mathcal{G}, k_a) \leftarrow \mathsf{Setup}(1^\lambda, n, n_1\ n_2, t)$ the following two conditions hold:

$$\Pr[\mathsf{Verify}(PK, m, \mathsf{Combine}(sk^c, sk^s, m, \mathcal{S}, \{\mathsf{Sign}(sk_i, m, \mathcal{S}, \mathcal{N})\}_{i\in\mathcal{S}})) = 1] = 1,$$

$$\Pr[\mathsf{Trace}(sk^t, m, \mathsf{Combine}(sk^c, sk^s, m, \mathcal{S}, \{\mathsf{Sign}(sk_i, m, \mathcal{S}, \mathcal{N})\}_{i\in\mathcal{S}})) = \mathcal{S}] = 1.$$

B. Unforgeability and Accountability

DeTAPS has to satisfy unforgeability and accountability, i.e., existential unforgeability under a chosen message attack with traceability [10]. Informally, unforgeability refers to an adversary compromising less than t signer cannot generate a valid signature on a message [43], [44], and accountability refers to an adversary compromising t or more signers cannot

Exp$^{\text{privC}}$

1. $b_1 \xleftarrow{\$} \{0,1\}$, $b_2 \xleftarrow{\$} \{0,1\}$
2. $(n, n_1, n_2, t_0, t_1, t_0', t_1', \mathcal{S}_0, \mathcal{S}_1, \mathcal{N}_0, \mathcal{N}_1, \mathsf{state}) \xleftarrow{\$} \mathcal{A}(1^\lambda)$ where $t_0, t_1 \in [n], t_0', t_1' \in [n_3]$
3. $(PK, (sk_1, \cdots, sk_n), \{sk_i^s, sk_i^c\}_{i=1}^{n_1}, \{sk_j^t\}_{j=1}^{n_2}, \mathcal{G}, k_a) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, n, n_1, n_2, t_{b_1})$
4. $(b_1', b_2') \leftarrow \mathcal{A}^{\mathcal{O}_2(\cdot,\cdot,\cdot),\mathcal{O}_3(\cdot,\cdot,\cdot,\cdot,\cdot),\mathcal{O}_4(\cdot,\cdot)}(PK, sk_i^s, \mathsf{state}), i \in [n_1]$
5. Output $(b_1' = b_1) \wedge (b_2' = b_2)$.

Restriction: sk_i^s can be the one used in $\mathcal{O}_3(\mathcal{S}_0, \mathcal{S}_1, \mathcal{N}_0, \mathcal{N}_1, m)$.

Exp$^{\text{privT}}$

1. $b_1 \xleftarrow{\$} \{0,1\}$, $b_2 \xleftarrow{\$} \{0,1\}$
2. $(n, n_1, n_2, t_0, t_1, t_0', t_1', \mathcal{S}_0, \mathcal{S}_1, \mathcal{N}_0, \mathcal{N}_1, \mathsf{state}) \xleftarrow{\$} \mathcal{A}(1^\lambda)$ where $t_0, t_1 \in [n], t_0', t_1' \in [n_3]$
3. $(PK, (sk_1, \cdots, sk_n), \{sk_i^s, sk_i^c\}_{i=1}^{n_1}, \{sk_j^t\}_{j=1}^{n_2}, \mathcal{G}, k_a) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, n, n_1, n_2, t_{b_1})$
4. $(b_1', b_2') \leftarrow \mathcal{A}^{\mathcal{O}_2(\cdot,\cdot,\cdot),\mathcal{O}_3(\cdot,\cdot,\cdot,\cdot,\cdot),\mathcal{O}_4(\cdot,\cdot)}(PK, \mathsf{state})$
5. Output $(b_1' = b_1) \wedge (b_2' = b_2)$.

Fig. 4. Two experiments of privacy against the combiners and the tracers.

generate a valid message-signature pair that traces to at least one signer. We formalize these two properties in the adversarial experiment in Fig. 2. Let $\mathbf{Adv}^{\text{forg}}_{\mathcal{A},\Pi}(\lambda)$ be the probability that $\mathcal{A}$ wins the experiment against the DeTAPS scheme Π.

Definition 2 (Unforgeability and Accountability): A DeTAPS scheme Π is unforgeable and accountable if for all PPT adversaries A, there is a negligible function negl such that $\mathbf{Adv}^{\text{forg}}_{\mathcal{A},\Pi}(\lambda) \leq \mathsf{negl}(\lambda)$.

C. Privacy

(1) Privacy against public. A party who observes a series of (m, σ) pairs, acquires nothing about t, t' or the signers. (2) Privacy against signers. Collaborating signers who observe a series of (m, σ) pairs, acquires nothing about t' or signers. (3) Privacy against combiners. A combiner cannot learn t, t', or signers. (4) Privacy against tracers. A tracer cannot learn t, t', or signers. We formalize the four properties in the adversarial experiment in Fig. 3 and Fig. 4.

Definition 3 (Privacy): A DeTAPS scheme is private if for all PPT adversaries $\mathcal{A}$, $\mathbf{Adv}^{\text{privP}}_{\mathcal{A},\Pi}(\lambda)$, $\mathbf{Adv}^{\text{privS}}_{\mathcal{A},\Pi}(\lambda)$, $\mathbf{Adv}^{\text{privC}}_{\mathcal{A},\Pi}(\lambda)$, and $\mathbf{Adv}^{\text{privT}}_{\mathcal{A},\Pi}(\lambda)$, are negligible functions of λ.

In Exp$^{\text{privP}}$, $\mathcal{A}$ generates four thresholds t_0, t_1, t_0' and t_1' in $[n]$ and is given PK. $\mathcal{A}$ submits a string of signature queries to a signing oracle $\mathcal{O}_1$, where each query contains a message m and four sets $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{W}_0$, and $\mathcal{W}_1$. Then, $\mathcal{A}$ receives a signature generated using either $\mathcal{S}_0$ or $\mathcal{S}_1$ (same for $\mathcal{N}_0$ or $\mathcal{N}_1$). $\mathcal{A}$ can access a tracing oracle $\mathcal{O}_2$ while not being able to determine whether the string of signatures it observed related to the left or the right sequence of sets.

In Exp$^{\text{privS}}$, $\mathcal{A}$ generates (t, t'), and is given all the signing keys. Same as Exp$^{\text{privP}}$, $\mathcal{A}$ cannot determine whether $\mathcal{O}_1$ that takes four sets $\mathcal{S}_1$, $\mathcal{S}_1$, $\mathcal{N}_0$, and $\mathcal{N}_1$ responds using wither $\mathcal{S}_0$ or $\mathcal{S}_1$ (same for $\mathcal{W}_0$ or $\mathcal{W}_1$).

IV. Our Construction

In this section, we present a concrete construction from a secure ATS scheme. The DeTAPS construction consists of eight building blocks:

- An ATS = (KeyGen, Sign, Combine, Verify, Trace);
- A DTPKE = (Setup, Join, Enc, Validate, ShareDecrypt, ShareVerify, Combine);
- A KASE = (Setup, KeyGen, Extract, Enc, Trapdoor, Adjust, Test);
- A PKE = (KeyGen, Encrypt, Decrypt);
- A COM = (Commit, Verify);
- A SIG = (KeyGen, Sign, Verify);
- A non-interactive zero knowledge argument of knowledge (P, V).
- An enclave $E = (init.E, config.E)$.

The DeTAPS scheme is shown in Fig. 5 and we put the generation of NIZKPs in the Appendix. In our construction, a DeTAPS signature on a message m is a tuple $\sigma = (\overline{\sigma}, \pi, \eta)$ where (1) $\overline{\sigma}$ is a dynamic threshold public-key encryption of an ATS signature σ^m on m, encrypted by using the ATS public key pk, (2) π is a zero-knowledge proof that $\mathcal{N}$ used as notaries is a valid subset of $[n_3]$, the decryption of $\overline{\sigma}$ is a valid ATS signature on m, the encryption of $(c_1, c_2, \{ind_i\}_{i\in\mathcal{N}})$ is $(gid, \mathcal{N})$, and (3) η is the combiner's signature on $\overline{\sigma}, \pi)$. We note that the shadows (gray rectangles) in the Combine and Trace in Fig. 5 indicate that the operations covered by the shadow are conducted within the TEE.

Remark 4 (Encryption of ATS signature): This step initiates dynamically private accountability by involving a quorum of t' notaries $\mathcal{N}$ to encrypt the underlying ATS signature σ^m. It is triggered by a quorum of t signers who designate $\mathcal{N}$ in generating a signature share $\mathsf{ATS.Sign}(sk_i, m, \mathcal{S})$. When combining t signature shares, an enclave E_i computes a threshold signature σ^m and then encrypts σ^m by invoking $\overline{\sigma} \leftarrow \mathsf{DTPKE.Enc}(ek, \mathcal{N}, \sigma^m)$. To facilitate successful tracing, each relevant notary N_j has to generate a decryption share of $\overline{\sigma}$ by using $\sigma_j^m \leftarrow \mathsf{DTPKE.ShareDecrypt}(pid_j, usk_j, \overline{\sigma})$ for a tracer to combine t' decryption shares and run $\mathcal{S} \leftarrow \mathsf{ATS.Trace}(pk, m, \sigma^m)$.

Remark 5 (Encryption of gid and $\mathcal{N}$): After an encrypted threshold signature is published and its signers are held accountable, we need to awaken its notaries to decrypt the encrypted threshold signature. To this end, we resort to KASE. The enclave creates a index by computing $(c_1^{gid}, c_2^{gid}, \{ind_i\}_{i\in\mathcal{N}}) \leftarrow \mathsf{KASE.Enc}(mpk, gid, \mathcal{N})$ where gid resembles file index and items in $\mathcal{N} = \{pid_i\}$ are keywords. Since the $|\{ind_i\}| = t'$, we use some dummy pids to hide t'. In tracing, a notary uses an aggregate-key k_a to

$\underline{\mathsf{Setup}(1^\lambda, n, n_1, n_2, t)}$:

1. $(pk, \{sk_i\}_{i=1}^n) \leftarrow \mathsf{ATS.KeyGen}(1^\lambda, n, t)$ //n: number of signers
2. $r_{pk} \leftarrow \mathcal{R}_\lambda$, $\mathsf{com}_{pk} \leftarrow \mathsf{COM.Comm}(pk, r_{pk})$
3. $(mk, ek, vk, ck) \leftarrow \mathsf{DTPKE.Setup}(1^\lambda)$
4. $(pk_j^s, sk_j^s) \leftarrow \mathsf{SIG.KeyGen}(1^\lambda, j)$, $j \in [n_1]$ //Combiner C_j's signing keys
5. $(pk_j^e, sk_j^e) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda, j)$, $j \in [n_1]$ //Enclave E_j's encryption keys
6. $(usk_o, upk_o, uvk_o) \leftarrow \mathsf{DTPKE.Join}(mk, o)$, $o \in [n_3]$ //n_3: number of notaries
7. $(\mathcal{B}, \mathcal{PK}, H) \leftarrow \mathsf{KASE.Setup}(\lambda, |G|)$
8. $(mpk, msk) \leftarrow \mathsf{KASE.KeyGen}(\lambda)$
9. $k_a \leftarrow \mathsf{KASE.Extract}(msk, \mathcal{G})$ — **stored in Enclave** E_j
10. $sk_j^c \leftarrow (\ pk, sk_j^e, t, ek, r_{pk}\ )$, $j \in [n_1]$ //Combining key
11. $sk_j^t \leftarrow (\ sk_j^e, ck, pk\ )$, $j \in [n_2]$ //Tracing key
12. $gid \leftarrow \mathsf{HASH}(GID, time)$, $GID \in \mathcal{G}$ //gid: signer group identifier
13. $PK \leftarrow (\mathsf{com}_{pk}, ek, vk, \{pk_i^s\}_{j=1}^{n_j}, \{pk_j^e\}_{j=1}^{n_1}, \mathcal{B}, \mathcal{PK}, H, mpk, \{gid\})$
14. Output $(PK, \{sk_i\}_{i=1}^n, \{sk_j^s\}_{j=1}^{n_1}, \{sk_j^c\}_{j=1}^{n_1}, \{sk_j^t\}_{j=1}^{n_2}, \mathcal{G}, k_a)$

$\underline{\mathsf{Sign}(sk_i, m, \mathcal{S}, \mathcal{N}) \rightarrow \widehat{\sigma_i}}$:

1. $\sigma_i \leftarrow \mathsf{ATS.Sign}(sk_i, m, \mathcal{S})$
2. $\widehat{\sigma_i} \leftarrow \mathsf{PKE.Enc}(pk_j^e, m||\sigma_i||\mathcal{N}||gid)$ //Sent to CB to be processed by a combiner

$\underline{\mathsf{Combine}(\ sk_j^c = (pk, sk_j^e, t, ek, r_{pk})\ , sk_j^s, m, \mathcal{S}, \{\widehat{\sigma_i}\}_{i\in\mathcal{S}}) \rightarrow \sigma}$

1. $(m||\sigma_i||\mathcal{N}||gid) \leftarrow \mathsf{PKE.Dec}(\widehat{\sigma}_i, sk_j^e)$ //**Enter** E_i here; $\sigma_i \in$ Tx pool
2. $\sigma^m \leftarrow \mathsf{ATS.Combine}(pk, m, \mathcal{S}, \{\sigma_i\}_{i\in\mathcal{S}})$
3. $\overline{\sigma} \leftarrow \mathsf{DTPKE.Enc}(ek, \mathcal{N}, \sigma^m)$
4. $(c_1^{gid}, c_2^{gid}, \{ind_o\}_{o\in\mathcal{N}}) \leftarrow \mathsf{KASE.Enc}(mpk, gid, \mathcal{N})$
5. Generate a proof for the relation:

$$\mathcal{R}((t', \mathsf{com}_{pk}, ek, mpk, m, \overline{\sigma}, gid, c_1^{gid}, c_2^{gid}, \{ind_o\}_{o\in\mathcal{N}}); (\mathcal{N}, \sigma^m, r_{pk}, pk)) = 1 \text{ iff}$$
$$\begin{Bmatrix} \mathcal{N} \subseteq [n_3],\ \mathsf{ATS.Verify}(pk, m, \sigma^m) = 1,\ \mathsf{COM.Verify}(pk, r_{pk}, \mathsf{com}_{pk}) = 1, \\ \overline{\sigma} = \mathsf{DTPKE.Enc}(ek, \mathcal{N}, \sigma^m),\ (c_1, c_2, \{ind_o\}_{o\in\mathcal{N}}) \leftarrow \mathsf{KASE.Enc}(mpk, gid, \mathcal{N}). \end{Bmatrix}$$

6. $\eta \leftarrow \mathsf{SIG.Sign}(sk_j^s, (m, \overline{\sigma}, c_1^{gid}, c_2^{gid}, \{ind_o\}_{o\in\mathcal{N}}, \pi))$
7. Output a DeTAPS signature $\sigma \leftarrow (\overline{\sigma}, c_1^{gid}, c_2^{gid}, \{ind_o\}_{o\in\mathcal{N}}, \pi, \eta)$

$\underline{\mathsf{Verify}(PK, m, \sigma) \rightarrow \ \{0,1\}}$

1. Accept σ if $\mathsf{SIG.Verify}(pk_j^s, m, \sigma) = 1$ and π is valid; reject otherwise.

$\underline{\mathsf{Trace}(sk_j^t = (\ sk_j^e, ck, pk\ ), m, \sigma) \rightarrow \mathcal{S}}$

1. $td_o \leftarrow \mathsf{KASE.Trapdoor}(k_a, pid_o)$ //For each notary with a pseudo identity pid_o
2. $td_o^{gid} \leftarrow \mathsf{KASE.Adjust}(\mathcal{B}, \mathcal{PK}, H, gid, \mathcal{G}, td_o)$ //For each gid and td_o
3. $\{0,1\} \leftarrow \mathsf{KASE.Test}(td_o^{gid}, (c_1^{gid}, c_2^{gid}, \{ind_o\}))$ //Search all indexes to locate $\overline{\sigma}$
4. If $\mathsf{SIG.Verify}(pk_j^s, (m, \overline{\sigma}, \pi), \eta) \neq 1$, output fail and return.
5. $\delta_o \leftarrow \mathsf{DTPKE.ShareDecrypt}(pid_o, usk_o, \overline{\sigma})$ //For each pid_o and each m
6. $\overline{\delta}_o \leftarrow \mathsf{PKE.Enc}(pid_o, uvk_o, \delta_o, pk_j^e)$
7. $(pid_o, uvk_o, \sigma_o^m) \leftarrow \mathsf{PKE.Dec}(\overline{\delta}_o, sk_j^e)$
8. $\{0,1\} \leftarrow \mathsf{DTPKE.ValidateCT}(ek, \mathcal{N}, \overline{\sigma})$ //Assume that $\overline{\sigma}$ is preloaded
9. $\{0,1\} \leftarrow \mathsf{DTPKE.ShareVerify}(vk, pid_o, uvk_o, \overline{\sigma}, \delta_o)$
10. $\sigma^m \leftarrow \mathsf{DTPKE.Combine}(ck, \mathcal{N}, \overline{\sigma}, \{\delta_o\}_{o\in\mathcal{N}})$
11. $\mathcal{S} \leftarrow \mathsf{ATS.Trace}(pk, m, \sigma^m)$.

Fig. 5. The DeTAPS scheme.

compute a trapdoor $td_i \leftarrow \mathsf{KASE.Trapdoor}(k_a, pid_i)$ for the SC to look for matching indexes.

Remark 6 (The generation of π): There are five parts in π. The first one and second one are done by committing to a vector, proving that every commitment is well formed [10], [46], and generating NIZKPs using the Fiat-Shamir transform. The last three are done by generating NIZKPs as well.

Remark 7 (Protect t from Tracer): After the original quorum of signers is revealed in an enclave, we can encrypt their identities by a target party's public key for directional tracing. If not, we can wait for some time to reveal a batch of quorums including t and then re-setup the system with a new t.

Remark 8 (Random selection of combiners and tracers): The random selection of combiners and tracers depends on the underlying consensus mechanism. For example, in Ethereum, the consensus mechanism is clique and ethash.

Correctness. DeTAPS is correct if the ATS scheme, DTPKE scheme, KASE scheme, PKE scheme, COM scheme, SIG scheme, and (P, V) are correct.

V. Security and Privacy of DeTAPS

Now we prove that the scheme is unforgeable, accountable, and private.

Theorem 1: The DeTAPS scheme Π in Fig. 5 is unforgeable, accountable, and private, assuming that the ATS is secure, the COM is hiding and binding, the PKE is semantically secure, the TEE is confidentiality-preserving, the DTPKE is IND-NAA-NAC-CPA secure, the KASE is privacy-preserving, the (P, V) is an argument of knowledge and honest verifier zero knowledge (HVZK), and the SIG is strongly unforgeable. The proof of Theorem 1 is captured in the following five lemmas.

Lemma 1: The DeTAPS scheme Π is unforgeable and accountable if the ATS is secure, the (P, V) is an argument of knowledge, and COM is blinding, i.e., for all PPT adversaries $\mathcal{A}$, there exists adversaries $\mathcal{A}_1$, and $\mathcal{A}_2$, such that

$$\mathbf{Adv}^{\text{forg}}_{\mathcal{A},\Pi}(\lambda) \leq \left(\mathbf{Adv}^{\text{forg}}_{\mathcal{A}_1,\mathsf{ATS}}(\lambda) + \mathbf{Adv}^{\text{bind}}_{\mathcal{A}_2,\mathsf{COM}}(\lambda)\right) \cdot \alpha(\lambda) + \beta(\lambda), \tag{1}$$

where α and β are the knowledge error and tightness of the proof system.

Proof. We prove Lemma 1 by defining experiments Exp 0, Exp 1, and Exp 2.

Exp 0. It is the experiment of unforgeability and accountability $\mathbf{Exp}^{\mathbf{forg}}$ defined in Fig. 2 applied to Π. If E_0 stands for $\mathcal{A}$ wins Exp_0, then

$$\mathbf{Adv}^{\text{forg}}_{\mathcal{A},\Pi}(\lambda) = \Pr[E_0]. \tag{2}$$

Exp 1. It is identical to Exp 0 with a strengthened winning condition: $\mathcal{A}$ has to output a valid forgery (m', σ') where $\sigma' = (\overline{\sigma}', c_1^{gid'}, c_2^{gid'}, \{ind'_o\}, \pi', \eta')$ with a witness satisfying $\mathcal{R}((t', \mathsf{com}_{pk}, ek, mpk, m', \overline{\sigma}', gid', c_1^{gid'}, c_2^{gid'}, \{ind'_o\}); (\mathcal{N}'', \sigma''^m, r''_{pk}, pk'')) = 1$.

Assume $\mathcal{A}'$ is an adversary in Exp 1. It invokes $\mathcal{A}$ and answers to $\mathcal{A}$'s queries until receives from $\mathcal{A}$ the $(m', \overline{\sigma}', c_1^{gid'}, c_2^{gid'}, \{ind'_o\})$ to provide a statement $(t', \mathsf{com}_{pk}, ek, mpk, m', \overline{\sigma}', gid', c_1^{gid'}, c_2^{gid'}, \{ind'_o\})$. $\mathcal{A}'$ executes the extractor Ext for (P, V) on $\mathcal{A}$'s remaining execution. Ext produces a witness $w = (\mathcal{N}'', \sigma''^m, r''_{pk}, pk'')$. $\mathcal{A}'$ uses w and sk^s to generate π' and η' such that $\sigma' = (\overline{\sigma}', c_1^{gid'}, c_2^{gid'}, \{ind'_o\}, \pi', \eta')$ is a valid signature on m'. $\mathcal{A}'$ outputs (m', σ') and w. By definition of Ext, if E_1 stands for $\mathcal{A}'$ wins Exp 1, then

$$\Pr[E_1] \geq (\Pr[E_0] - \alpha(\lambda))/\beta(\lambda). \tag{3}$$

Exp 2. The adversary now has pk and r_{pk}. We strengthen the winning condition by requiring $pk = pk''$. Let E_2 stand for $\mathcal{A}$ wins Exp 2 and E stand for $pk \neq pk''$. Therefore, $\Pr[E_2] = \Pr[E_1 \wedge \neg E] \geq \Pr[E_1] - \Pr[E]$. Assume that there is an adversary $\mathcal{A}_2$ such that $\Pr[E] = \mathbf{Adv}^{\text{bind}}_{\mathcal{A}_2,\mathsf{COM}}(\lambda)$. We have

$$\Pr[E_2] \geq \Pr[E_1] - \mathbf{Adv}^{\text{bind}}_{\mathcal{A}_2,\mathsf{COM}}(\lambda). \tag{4}$$

Next, we construct an adversary $\mathcal{A}_1$ that invokes $\mathcal{A}$ and answers to $\mathcal{A}$'s queries. When $\mathcal{A}$ outputs a forgery (m', σ') and a witness $(\mathcal{N}'', \sigma''^m, r''_{pk}, pk'')$ that meet the winning condition of Exp 1 and Exp 2, $\mathcal{A}_1$ outputs (m', σ''^m). By $\mathcal{R}$, we have σ''^m is a valid signature on m' with respect to pk''. By Exp 2, we have $pk = pk''$. Therefore, if $\mathcal{A}$ wins Exp 2, then (m', σ''^m) is a valid forgery for the ATS scheme. Since the ATS is secure, we have that $\Pr[E_2]$ is at most negligible, i.e.,

$$\mathbf{Adv}^{\text{forg}}_{\mathcal{A}_1,\mathsf{ATS}}(\lambda) \geq \Pr[E_2]. \tag{5}$$

Lastly, combining (2), (3), (4), and (5) proves (1). This completes the proof of the lemma. □

Lemma 2: The DeTAPS scheme Π is private against the public if the COM is hiding, the PKE is semantically secure, the TEE is confidentiality-preserving, the DTPKE is IND-NAA-NAC-CPA secure, the KASE is privacy-preserving, the (Prove, Veriy) is HVZK, and the SIG is strongly unforgeable, i.e., for all PPT adversaries $\mathcal{A}$, there exists adversaries $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$, $\mathcal{A}_4$, $\mathcal{A}_5$, $\mathcal{A}_6$, and $\mathcal{A}_7$, such that

$$\begin{aligned}\mathbf{Adv}^{\text{priP}}_{\mathcal{A},\Pi}(\lambda) \leq 2\Big(&\epsilon_{\mathcal{A}_1}(\lambda) + \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{A}_2,\mathsf{PKE}}(\lambda) + \mathbf{Adv}^{\text{ind-obs}}_{\mathcal{A}_3,\mathsf{TEE}}(\lambda) \\ &+ \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{A}_4,\mathsf{DTPKE}}(\lambda) + \mathbf{Adv}^{\text{ind-cka}}_{\mathcal{A}_5,\mathsf{KASE}}(\lambda) \\ &+ Q \cdot \mathbf{Adv}^{\text{hvzk}}_{\mathcal{A}_6,(\mathsf{P},\mathsf{V})}(\lambda) + \mathbf{Adv}^{\text{euf-cma}}_{\mathcal{A}_7,\mathsf{SIG}}(\lambda)\Big)\end{aligned} \tag{6}$$

where $\epsilon_{\mathcal{A}_1}(\lambda)$ is hiding statistical distance of COM, obs is an observation function of $\mathcal{A}_3$, and Q is query number.

Proof. We prove Lemma 2 by defining seven experiments.

Exp 0. It is the experiment of privacy against the public $\mathbf{Exp}^{\mathbf{priP}}$ defined in Fig. 3 applied to Π. If E_0 stands for $\mathcal{A}$ wins Exp 0, then

$$\mathbf{Adv}^{\text{priP}}_{\mathcal{A},\Pi}(\lambda) = |2\Pr[E_0] - 1|. \tag{7}$$

Exp 1. It is identical to Exp 0 except that step 2 of Setup in Fig. 5 is modified such that $r_{pk} \leftarrow \mathcal{R}_\lambda$, $\mathsf{com}_{pk} \leftarrow \mathsf{COM.Comm}(0, r_{pk})$, where 0 is committed instead of pk. Since COM is hiding, the adversary's **Adv** in Exp 1 is indistinguishable from its **Adv** in Exp 0, i.e., say E_3 stands for $\mathcal{A}_1$ wins Exp 1, then

$$|\Pr[E_1] - \Pr[E_0]| \leq \epsilon_{\mathcal{A}_1}(\lambda). \tag{8}$$

Exp 2. It is identical to Exp 1 except that the signing oracle $\mathcal{O}_1(\mathcal{S}_0, \mathcal{S}_1, \mathcal{N}_0, \mathcal{N}_1, m)$ is modified such that step 2 of Sign in Fig. 5 now returns $\widehat{\sigma}_i \leftarrow \mathsf{PKE.Enc}(pk_j^e, 0)$, where 0 is encrypted instead of $(m||\sigma_i||\mathcal{N}||gid)$. Since PKE is semantically secure, $\mathcal{A}_2$'s **Adv** in Exp 2 is indistinguishable from its **Adv** in Exp 1, i.e., say E_2 stands for $\mathcal{A}_2$ wins Exp 2, then

$$|\Pr[E_2] - \Pr[E_1]| \leq \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{A}_2,\mathsf{PKE}}(\lambda). \tag{9}$$

Exp 3. It is identical to Exp 2 except that the is modified such that step 1 of Combine in Fig. 5 now returns the same $(m||\sigma_i||\mathcal{N}||gid)$ within a different enclave.

We assume that $\mathcal{A}_3$ only observes outputs of an observation function obs. The confidentiality-preserving property of the TEE is proved by the fact that for any two traces that have equivalent attacker operations and equivalent observations of the enclave execution, but possibly different enclave private states and executions, $\mathcal{A}_3$'s execution, i.e., its sequence of states, is identical [39]. In specific,

$$\begin{aligned}&\forall_{\pi_1,\pi_2}.\ \big(\mathcal{A}_{e_1}(\pi_1[0]) = \mathcal{A}_{e_2}(\pi_2[0]) \wedge \\ &\forall_i.\ curr(\pi_1[i]) = curr(\pi_2[i]) \wedge I^P(\pi_1[i]) = I^P(\pi_2[i]) \wedge \\ &\forall_i.\ curr(\pi_1[i]) = e \Rightarrow obs_{e_1}(\pi_1[i+1]) = obs_{e_2}(\pi_2[i+1])\big) \\ &\Rightarrow (\forall_i.\ \big(\mathcal{A}_{e_1}(\pi_1[i]) = \mathcal{A}_{e_2}(\pi_2[i])\big) \end{aligned} \tag{10}$$

where $\mathcal{A}$ is $\mathcal{A}_3$, e_1 and e_2 are two different enclaves, π_1 and π_2 are two traces with the same initial state for enclaves e_1 and e_2, $curr$ is the current mode of the platform, and I^P is these bits of non-determinism in a particular state. Therefore, given that the TEE is confidentiality-preserving, $\mathcal{A}_3$'s **Adv** in Exp 3 is indistinguishable from its **Adv** in Exp 2, i.e., say E_3 stands for $\mathcal{A}_3$ wins Exp 3, then

$$|\Pr[E_3] - \Pr[E_2]| \leq \mathbf{Adv}^{\text{ind-obs}}_{\mathcal{A}_3,\text{TEE}}(\lambda). \tag{11}$$

We refer the interested reader to [39] for the more detailed information.

Exp 4. It is identical to Exp 4 except that the signing oracle $\mathcal{O}_1(\mathcal{S}_0, \mathcal{S}_1, \mathcal{N}_0, \mathcal{N}_1, m)$ is modified such that step 3 of Combine now returns $\overline{\sigma} \leftarrow \mathsf{DTPKE.Enc}(ek, \mathcal{N}, 0)$, where 0 is encrypted instead of σ^m. Since DTPKE is secure, $\mathcal{A}_4$'s **Adv** in Exp 4 is indistinguishable from its **Adv** in Exp 3, i.e., say E_5 stands for $\mathcal{A}_4$ wins Exp 4, then

$$|\Pr[E_4] - \Pr[E_3]| \leq \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{A}_4,\mathsf{DTPKE}}(\lambda). \tag{12}$$

Exp 5. It is identical to Exp 4 except that the signing oracle $\mathcal{O}_1(\mathcal{S}_0, \mathcal{S}_1, \mathcal{N}_0, \mathcal{N}_1, m)$ is modified such that step 4 of Combine now returns $(c_1^{gid}, c_2^{gid}, \{ind_o\}_{o\in\mathcal{N}}) \leftarrow \mathsf{KASE.Enc}(mpk, gid, \{r_j\})$, where a random set is encrypted instead of $\mathcal{N}$. Since KASE is privacy-preserving, $\mathcal{A}_5$'s **Adv** in Exp 5 is indistinguishable from its **Adv** in Exp 4, i.e., say E_5 stands for $\mathcal{A}_5$ wins Exp 5, then

$$|\Pr[E_5] - \Pr[E_4]| \leq \mathbf{Adv}^{\text{indcka}}_{\mathcal{A}_5,\mathsf{KASE}}(\lambda). \tag{13}$$

Exp 6. It is identical to Exp 6 except that the signing oracle $\mathcal{O}_1(\mathcal{S}_0, \mathcal{S}_1, \mathcal{N}_0, \mathcal{N}_1, m)$ is modified such that step 5 of Combine now generates a proof π by using the simulator, which is given $(t', \mathsf{com}_{pk}, ek, mpk, m, \overline{\sigma}, gid, c_1^{gid}, c_2^{gid}, \{ind_o\}_{o\in\mathcal{N}})$ as input. Since the simulated proofs are computationally indistinguishable from real proofs, $\mathcal{A}_6$'s **Adv** in Exp 6 is indistinguishable from its **Adv** in Exp 5, i.e., say E_3 stands for $\mathcal{A}_6$ wins Exp 6, then

$$|\Pr[E_6] - \Pr[E_5]| \leq Q \cdot \mathbf{Adv}^{\text{hvzk}}_{\mathcal{A}_6,(\mathsf{P},\mathsf{V})}(\lambda). \tag{14}$$

Exp 7. It is identical to Exp 7 except that responses to $\mathcal{O}_2(m, \sigma)$ are fail. If SIG is strongly unforgeable, $\mathcal{A}_7$'s **Adv** in Exp 7 is indistinguishable from its **Adv** in Exp 6, i.e., say E_2 stands for $\mathcal{A}_7$ wins Exp 7, then

$$|\Pr[E_7] - \Pr[E_6]| \leq \mathbf{Adv}^{\text{euf-cma}}_{\mathcal{A}_7,\mathsf{SIG}}(\lambda). \tag{15}$$

In Exp 7, $\mathcal{A}_7$'s view is independent of b. Consequently, $\mathcal{A}_7$ has no advantage in Exp 7, i.e.,

$$\Pr[E_7] = 1/2. \tag{16}$$

Lastly, combining (7)-(14) proves (6). This completes the proof of lemma 2. □

Lemma 3: The DeTAPS scheme Π is private against the signers.

Proof. The proof of Lemma 3 is identical to the proof of Lemma 2. □

Lemma 4: The DeTAPS scheme Π is private against the combiners.

Proof. The proof of Lemma 4 is almost identical to the proof of Lemma 2 except that the Exp 1 is removed because the combiner has the signing key, i.e.,

$$\begin{aligned}\mathbf{Adv}^{\text{priP}}_{\mathcal{A},\Pi}(\lambda) \leq 2\Big(&Q \cdot \mathbf{Adv}^{\text{hvzk}}_{\mathcal{A}_1,(\mathsf{Prove},\mathsf{Verify})}(\lambda) + \epsilon(\lambda) \\ &+ \mathbf{Adv}^{\text{indcka}}_{\mathcal{A}_3,\mathsf{KASE}}(\lambda) + \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{A}_4,\mathsf{DTPKE}}(\lambda) \\ &+ \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{A}_5,\mathsf{PKE}}(\lambda)\Big)\end{aligned} \tag{17}$$

Lastly, combining (7), (9)-(14) proves (15). This completes the proof of Lemma 4. □

Lemma 5: The DeTAPS scheme Π is private against the tracers.

Proof. Although the tracer carries out the tracing process within its enclave, its view is the same as one from the public. Therefore, the proof of Lemma 5 is identical to the proof of Lemma 2. □

We show how to generate the proofs as follows.

1. Prove $\mathcal{N} \subseteq [n_3]$:

1.1 Prove $V = \prod_{i=1}^{n_3} pk_i^{b_i}$

Prover:

- choose randomly $\alpha_i \xleftarrow{\$} \mathbb{Z}_q, 1 \leq i \leq n_3$, compute $B = \prod_{i=1}^{n_3} pk_i^{\alpha_i}$
- compute $H = \mathsf{hash}(pk_1, \cdots, pk_{n_3}, V, B)$
- send $(B, \alpha_1' = b_1 H + \alpha_1, \cdots, \alpha_n' = b_n H + \alpha_n)$ to verifier

Verifier:

- compute $H = \mathsf{hash}(pk_1, \cdots, pk_{n_3}, V, B)$
- check $V^H \cdot B \stackrel{?}{=} \prod_{i=1}^{n_3} pk_i^{\alpha_i'}$

1.2 Prove $V_0 = g^{\psi}$ and $V_1 = g^{\sum_{i=1}^{n_3} b_i} \cdot h^{\psi}$:

1.2.1 Prove $V_0 = g^{\psi}$

Prover:

- choose randomly $\alpha \xleftarrow{\$} \mathbb{Z}_q$, compute $B = g^{\alpha}$
- compute $H = \mathsf{hash}(g, V_0, B)$
- send $(g, V_0, B, \alpha' = \psi H + \alpha)$ to the verifier

Verifier:

- compute $H = \mathsf{hash}(g, V_0, B)$
- check $V_0^H B \stackrel{?}{=} g^{\alpha'}$

1.2.2 Prove $V_1 = g^{\sum_{i=1}^{n_3} b_i} \cdot h^{\psi}$:

Prover:

- choose randomly $\alpha_i \xleftarrow{\$} \mathbb{Z}_q, 1 \leq i \leq n_3 + 1$, compute $B = \prod_{i=1}^{n_3} g^{\alpha_i} \cdot h^{\alpha_{n_3+1}}$
- compute $H = \mathsf{hash}(g, h, V_1, B)$
- send $(B, \alpha_1' = b_1 H + \alpha_1, \cdots, \alpha_n' = b_{n_3} H + \alpha_{n_3}, \alpha_{n_3+1}' = \psi H + \alpha_{n_3+1})$ to the verifier

Verifier:

- compute $H = \mathsf{hash}(g, h, V_1, B)$
- check $V_1^H \cdot B \stackrel{?}{=} \prod_{i=1}^{n_3} g^{\alpha_i'} \cdot h^{\alpha_{n_3+1}'}$

1.3 Prove $b_i(1-b_i)=0$ for $i=1,2,\cdots,n_3$: page 67 in Guaranteed Correct Sharing of Integer Factorization with Off-line Share-holders, PKC'98:

Common input: $Com, g, h \in \mathbb{G}$, Prover's input: $r \in \mathbb{Z}_q$

To prove either $Com = h^r$ or $Com = gh^r$

Prover:

if $Com = h^r$

- choose randomly $w, r_1, c_1 \in \mathbb{Z}_q$
- compute $A = h^w$, $B = h^{r_1}(Com/g)^{-c_1}$, and $H = \mathsf{hash}(Com, A, B)$
- send $(Com, A, B, c_1, c_2 = H - c_1, r_1, r_2 = w + zc_2)$ to verifier

else if $Com = gh^r$

- choose randomly $w, r_2, c_2 \in \mathbb{Z}_q$
- compute $A = h^{r_2}Com^{-c_2}$, $B = h^w$, and $H = \mathsf{hash}(Com, A, B)$
- send $(Com, A, B, c_1 = H - c_2, c_2, r_1 = w + rc_1, r_2)$ to verifier

Verifier:

- check $H \stackrel{?}{=} c_1 + c_2 \bmod q$
- check $h^{r_1} \stackrel{?}{=} B(Com/g)^{c_1} \bmod q$
- check $h^{r_2} \stackrel{?}{=} ACom^{c_2} \bmod q$

2. Prove $\mathsf{ATS.Verify}(pk, m, \sigma^m) = 1$: Sec5.1, Sec5.4, Fig5, Fig6 in TAPS

2.1 Prove $g^z = [\prod_{i=1}^n pk_i^{b_i}]^c \cdot R$ where $R = \sigma^m$ is protected.

Prover:

- choose randomly $k_z, k_{b1}, k_{b2}, \cdots, k_{bn} \leftarrow \mathbb{Z}_q$, compute $A = g^{k_z} \prod_{i=1}^n pk_i^{-c \cdot k_{bi}}$
- choose randomly $r \leftarrow \mathbb{Z}_q$, compute $B = g^z g^r$, $z' = z + r$, and $R' = Rg^r$
- compute $H = \mathsf{hash}(pk_1, \cdots, pk_n, c, A, B, z', R')$
- send $(\hat{z} = z'H + k_z, \hat{b}_1 = b_1 H + k_{b1}, \cdots, \hat{b}_n = b_n H + k_{bn})$ to verifier

Verifier:

- compute $H = \mathsf{hash}(pk_1, \cdots, pk_n, c, A, B, z', R')$
- check $A \cdot R'^H [\prod_{i=1}^n pk_i^{\hat{b}_i}]^c \stackrel{?}{=} g^{\hat{z}}$

2.2 Prove $T_0 = g^{\psi}$ and $T_1 = g^{\sum_{i=1}^n b_i} \cdot h^{\psi}$:

2.2.1 Prove $T_0 = g^{\psi}$

Prover:

- choose randomly $\alpha \stackrel{\$}{\leftarrow} \mathbb{Z}_q$, compute $B = g^{\alpha}$
- compute $H = \mathsf{hash}(g, T_0, B)$
- send $(g, T_0, B, \alpha' = \psi H + \alpha)$ to verifier

Verifier:

- compute $H = \mathsf{hash}(g, T_0, B)$
- check $T_0^H B \stackrel{?}{=} g^{\alpha'}$

2.2.2 Prove $T_1 = g^{\sum_{i=1}^n b_i} \cdot h^{\psi}$:

Prover:

- choose randomly $\alpha_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q, 1 \le i \le n+1$, compute $B = \prod_{i=1}^n g^{\alpha_i} \cdot h^{\alpha_{n+1}}$
- compute $H = \mathsf{hash}(g, h, T_1, B)$
- send $(B, \alpha'_1 = b_1 H + \alpha_1, \cdots, \alpha'_n = b_n H + \alpha_n, \alpha'_{n+1} = \psi H + \alpha_{n+1})$ to verifier

Verifier:

- compute $H = \mathsf{hash}(g, h, T_1, B)$
- check $T_1^H \cdot B \stackrel{?}{=} \prod_{i=1}^n g^{\alpha'_i} \cdot h^{\alpha'_{n+1}}$

2.3 Prove $b_i(1-b_i)=0$ for $i=1,2,\cdots,n$: same to 1.3

3. Prove $\mathsf{COM.Verify}(pk, r_{pk}, \mathsf{com}_{pk}) = 1$:

Prover:

- set $A = \mathsf{com}_{pk}$
- choose randmly $\alpha_1, \alpha_2 \in \mathbb{Z}_q$, compute $B = g^{\alpha_1} h^{\alpha_2}$, and $H = \mathsf{hash}(g, h, A, B)$
- send $(A, B, \alpha'_1 = Hpk + \alpha_1, \alpha'_2 = Hr_{pk} + \alpha_2)$ to verifier

Verifier:

- computes $H = \mathsf{hash}(g, h, A, B)$
- check $A^H B \stackrel{?}{=} g^{\alpha'_1} h^{\alpha'_2}$

4. Prove $\overline{\sigma} = \mathsf{DTPKE.Enc}(ek, \mathcal{N}, \sigma^m)$:

since $C_1 = u^{-k}$ and $C_2 = h^{ks}$, prove similar to 1.2.1

5. Prove $(c_1, c_2, \{ind_i\}) \leftarrow \mathsf{KASE.Enc}(mpk, gid, \mathcal{N})$:

since $ind = e(g, H(pk_i))^t / e(g_1, g_{|\{gid\}|})^t$, $pk_i \in \{\mathcal{N}\}$

Prover:

- set $A = e(g, H(pk_i))/e(g_1, g_{|\{gid\}|})$
- choose randmly $\alpha \in \mathbb{Z}_q$, compute $B = A^{\alpha}$, and $H = \mathsf{hash}(ind, A, B)$
- send $(ind, A, B, \alpha' = Ht + \alpha)$ to verifier

Verifier:

- computes $H = \mathsf{hash}(ind, A, B)$
- check $ind^H B \stackrel{?}{=} A^{\alpha'}$

TABLE II
EXPERIMENTAL PARAMETERS

Parameter	Value
n, n_3	[10, 50], [10, 50]
n_1, n_2, n_4	5, 5, [100, 1000]
$\lvert m \rvert$, λ	[1, 10], 512
t, t'	{5, 10, 15}

VI. PERFORMANCE EVALUATION

In this section, we build a prototype of DeTAPS based on Intel SGX2 and Ethereum blockchain. We evaluate its performance regarding computational costs and communication overhead of five phases.

A. Experimental Settings

1) Dataset and Parameters: Since there are no specialized datasets, we synthesize the input data. Table II lists key experimental paraments. We vary the number of signers n from 10 to 50, the number of notaries n_3 from 10 to 50, the number of signatures n_4 from 100 to 1000, the length of message m from 1 KBytes to 10 KBytes, the threshold t and the number of notaries t' from 5 to 15. The security parameter λ is 512, the number of combiner n_1 and the number of tracer n_2 is set to 5. Our codes are uploaded to github.com/UbiPLab/DeTAPS.

2) Setup: We implement DeTAPS on a Linux server running Ubuntu 20.04 with an Intel(R) Xeon(R) Platinum 8369B CPU @ 2.70GHz We use HMAC-SHA256 as the pseudo-random function to implement the hash functions. We use AES as the symmetric encryption. We use Geth as the primary tool for Ethereum network environment establishing. We use remix to write the SC and deploy it by a light-weighted browser plugin metamask. We use puppeth to create the genesis block. We use Python to implement all cryptographic primitives. The implementation details are shown in Fig. 6.

B. Computational Cost

In Setup, DeTAPS generates all keys. In Signing, a signer computes a signature share. In Combining, a combiner combines a signature from t signature shares. In Verifying,

TABLE III
COMMUNICATION OVERHEAD

Phase	Signing	Combining		Verifying	Tracing			
Party	Signer	Enclave	Combiner	Verifier	Notary	SC	Enclave	Tracer
Theory	$\mathrm{Tx}^{\mathrm{Sign}}$	$m, \overline{\sigma}, \mathsf{KASE.Enc}(mpk, gid, \mathcal{N}), \pi$	$\mathrm{Tx}^{\mathrm{Comb}}$	b	$(td, \overline{\delta})$	$\lvert\overline{\sigma}\rvert t'$	$\mathsf{PKE.Enc}(\mathcal{S})$	$\mathsf{PKE.Enc}(\mathcal{S})$
Practice	1.34 MB	17.63 KB	18.06 KB	1 bit	0.06 KB	$1.51t'$ KB	2.57 KB	2.57 KB

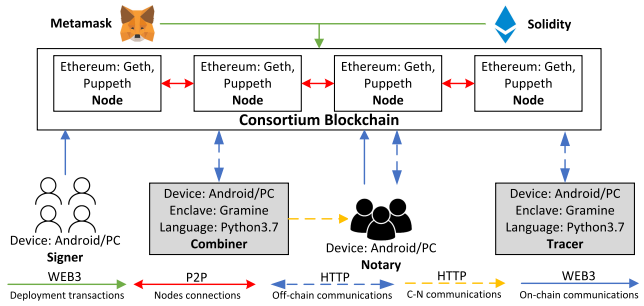
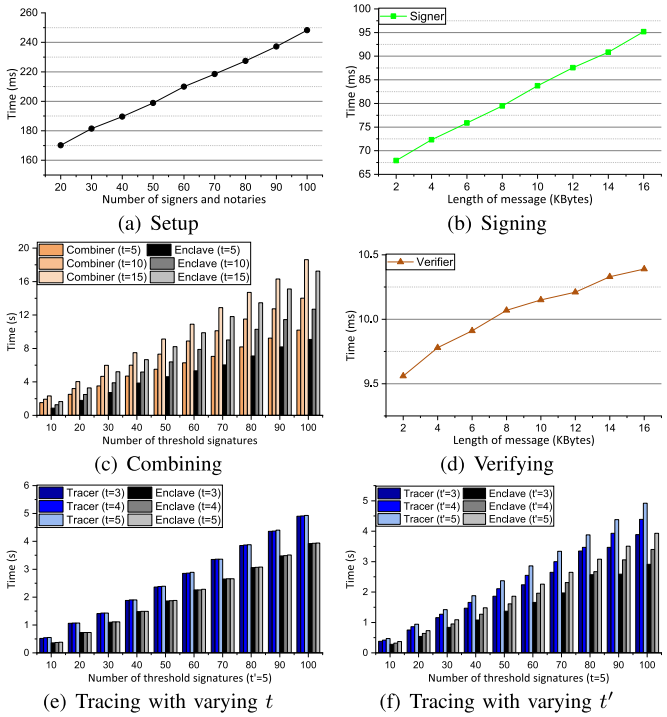


Fig. 6. Implementation details of DeTAPS.



(a) Setup (b) Signing (c) Combining (d) Verifying (e) Tracing with varying t (f) Tracing with varying t'

Fig. 7. Computational costs.

a verifier verifies a threshold signature. In Tracing, a notary computes a trapdoor, the SC searches on indexes, a tracer traces a threshold signature. We compute the average consumed time of ten experiments for each figure below. In Fig. 7(a), Setup with $n = 50$ and $n_3 = 50$ is about 177 ms. In Fig. 7(b), Signing a 10-KByte message is about 52 ms. In Fig. 7(c), Combining is around 10 s for a 10-KByte message, 100 threshold signatures, and $t = 5$, i.e., 500 signature shares. In Fig. 7(d), Verifying is around 10 ms for a 10-KByte message. In Fig. 7(e), Tracing with varying t is about 4.9 s for the enclave given 100 threshold signatures, $t = 3$, and $t' = 5$. In Fig. 7(f), Tracing with varying t' is about 3.89 s for the enclave given 100 threshold signatures, $t' = 3$, and $t = 5$.

C. Communication Overhead

We analyze the communication overhead by counting the length of transmitted messages of all parties for one signing group. In Signing, a signer sends a signing transaction $\mathrm{Tx}^{\mathrm{Sign}}$ including a signature share. In Combining, an enclave outputs a message m, an encrypted threshold signature $\overline{\sigma}$, an encrypted group number $\mathsf{KASE.Enc}(mpk, gid, \mathcal{N})$, and a proof π. A combiner sends a combining transaction $\mathrm{Tx}^{\mathrm{Comb}}$ including (m, σ). In Verifying, the verifier outputs 1 bit. In Trace, a notary outputs a trapdoor td, the SC outputs t' encrypted threshold signatures, the enclave outputs a ciphertext $\mathsf{PKE.Enc}(\mathcal{S})$, and the tracer relays it to a target party. We record the communication overhead in Table III.

VII. CONCLUSION

In this work, we have presented DeTAPS, a new threshold signature scheme that achieves unforgeability, accountability, and privacy. DeTAPS takes a step further towards providing strong privacy as well as notarized and dynamic tracing in a distributed network. In DeTAPS, the signature threshold t and the witness threshold t' is hidden from distributed combiners and tracers by using an enclave to secure the combining and tracing. We formally prove the security and privacy of DeTAPS. Experimental results show that DeTAPS is efficient, e.g., combining (tracing) a threshold signature for 5 singers (notaries) in the enclave is only 86 (38) ms.

REFERENCES

[1] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," in *Proc. 6th Annu. Int. Cryptol. Conf. (CRYPTO)*, Santa Barbara, CA, USA, Aug. 1989, pp. 307–315.

[2] V. Shoup, "Practical threshold signatures," in *Proc. 17th Int. Conf. Theory Appl. Cryptograph. Techn.*, Bruges, Belgium, May 2000, pp. 207–220.

[3] I. Damgård and M. Koprowski, "Practical threshold RSA signatures without a trusted dealer," in *Proc. 18th Int. Conf. Theory Appl. Cryptograph. Techn.*, May 2001, pp. 153–165.

[4] 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.

[5] R. Bacho and J. Loss, "On the adaptive security of the threshold BLS signature scheme," in *Proc. 29th ACM Conf. Comput. Commun. Secur. (CCS)*, Los Angeles, CA, USA, Nov. 2022, pp. 193–207.

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

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

[8] 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 (PKC)*, Miami, FL, USA, Jan. 2003, pp. 31–46.

[9] P.-A. Fouque and J. Stern, "Fully distributed threshold RSA under standard assumptions," in *Proc. 7th Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Dec. 2001, pp. 310–330.

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

[11] A. Scafuro and B. Zhang, "One-time traceable ring signatures," in *Proc. 26rd Eur. Symp. Res. Comput. Secur. (ESORICS)*, Oct. 2021, pp. 481–500.

[12] E. Syta et al., "Keeping authorities 'honest or bust' with decentralized witness cosigning," in *Proc. 37th IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, May 2016, pp. 526–545.

[13] 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.

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

[15] T. Okamoto and K. Takashima, "Decentralized attribute-based signatures," in *Proc. 16th Int. Workshop Theory Pract. Public Key Cryptography (PKC)*, Nara, Japan, Feb. 2013, pp. 125–142.

[16] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, May 2016, pp. 839–858.

[17] Y. Xiao, N. Zhang, J. Li, W. Lou, and Y. T. Hou, "PrivacyGuard: Enforcing private data usage control with blockchain and attested off-chain contract execution," in *Proc. 25rd Eur. Symp. Res. Comput. Secur. (ESORICS)*, Guildford, U.K., Sep. 2020, pp. 610–629.

[18] B. C. Xing, M. Shanahan, and R. Leslie-Hurd, "Intel® software guard extensions (Intel® SGX) software support for dynamic memory allocation inside an enclave," in *Proc. Hardw. Architectural Support Secur. Privacy*, Seoul, (South) Korea, Jun. 2016, pp. 1–9.

[19] Intel. *Which Platforms Support Intel® Software Guard Extensions (Intel® SGX) SGX2?* Accessed: Dec. 30, 2023. [Online]. Available: https://www.intel.com/content/www/us/en/support/articles/000058764/software/intel-security-products.html

[20] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu, "Efficient batched oblivious PRF with applications to PSI," in *Proc. 23rd ACM Conf. Comput. Commun. Secur. (CCS)*, Oct. 2016, pp. 818–829.

[21] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "PSI from PaXoS: Fast, malicious private set intersection," in *Proc. 39th Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, May 2020, pp. 739–767.

[22] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, Alexandria, EG, USA, Oct. 2006, pp. 89–98.

[23] L. Cheng and F. Meng, "Server-aided revocable attribute-based encryption revised: Multi-user setting and fully secure," in *Proc. 26rd Eur. Symp. Res. Comput. Secur.*, Oct. 2021, pp. 192–212.

[24] R. Li and A. X. Liu, "Adaptively secure conjunctive query processing over encrypted data for cloud computing," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, San Diego, CA, USA, Apr. 2017, pp. 697–708.

[25] X. Lei, A. X. Liu, R. Li, and G.-H. Tu, "SecEQP: A secure and efficient scheme for SkNN query problem over encrypted geodata on cloud," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Macao, China, Apr. 2019, pp. 662–673.

[26] R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk, "Universal designated-verifier signatures," in *Proc. 9th Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2003, pp. 523–542. Accessed: Dec. 30, 2023.

[27] Y. Li, W. Susilo, Y. Mu, and D. Pei, "Designated verifier signature: Definition, framework and new constructions," in *Proc. 4th Int. Conf. Ubiquitous Intell. Comput.*, Hong Kong, Jul. 2007, pp. 1191–1200.

[28] 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./Jul. 2016.

[29] G. Yang, D. S. Wong, X. Deng, and H. Wang, "Anonymous signature schemes," in *Proc. 9th Int. Conf. Theory Pract. Public-Key Cryptography (PKC)*, New York, NY, USA, Apr. 2006, pp. 347–363.

[30] F. Guo, R. Chen, W. Susilo, J. Lai, G. Yang, and Y. Mu, "Optimal security reductions for unique signatures: Bypassing impossibilities with a counterexample," in *Proc. 37th Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, USA, Aug. 2017, pp. 517–547.

[31] Z. Liu, K. Nguyen, G. Yang, H. Wang, and D. S. Wong, "A lattice-based linkable ring signature supporting stealth addresses," in *Proc. 24th Eur. Symp. Res. Comput. Secur.*, Sep. 2019, pp. 726–746.

[32] D. Boneh and X. Boyen, "Hierarchical identity based encryption with constant size ciphertext," in *Proc. 24th Int. Conf. Theory Appl. Cryptograph. Technique*, May 2005, pp. 440–456.

[33] C. Delerablé, P. Paillier, and D. Pointcheval, "Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys," in *Proc. 1st Int. Conf. Pairing-Based Cryptography*, Jul. 2007, pp. 39–59.

[34] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 3rd ed. Boca Raton, FL, USA: CRC Press, 2021, pp. 1–598.

[35] F. McKeen et al., "Innovative instructions and software model for isolated execution," in *Proc. 2nd Int. Workshop Hardw. Architectural Support Secur. Privacy*, Tel-Aviv, Israel, Jun. 2013, pp. 1–8.

[36] Intel. *Intel® Software Guard Extensions*. [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/get-started.html

[37] M. Li, Y. Chen, 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, Aug./Jul. 2023, doi: 10.1109/TDSC.2022.3192367.

[38] M. Li et al., "Astraea: Anonymous and secure auditing based on private smart contracts for donation systems," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 4, pp. 3002–3018, Aug. 2023, doi: 10.1109/TDSC.2022.3204287.

[39] P. Subramanyan, R. Sinha, I. Lebedev, S. Devadas, and S. A. Seshia, "A formal foundation for secure remote execution of enclaves," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Dallas, TX, USA, Oct. 2017, pp. 2435–2450.

[40] M. Li, L. Zhu, Z. Zhang, C. Lal, M. Conti, and M. Alazab, "Anonymous and verifiable reputation system for e-commerce platforms based on blockchain," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4434–4449, Dec. 2021, doi: 10.1109/TNSM.2021.3098439.

[41] M. Li et al., "Anonymous, secure, traceable, and efficient decentralized digital forensics," *IEEE Trans. Knowl. Data Eng.*, early access, Dec. 30, 2023, doi: 10.1109/TKDE.2023.3321712.

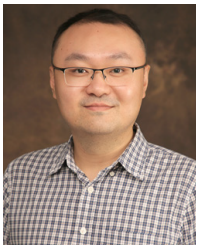
[42] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1508–1532, 2nd Quart., 2019.

[43] B. Tahir, A. Jolfaei, and M. Tariq, "A novel experience-driven and federated intelligent threat-defense framework in IoMT," *IEEE J. Biomed. Health Informat.*, early access, Dec. 30, 2023.

[44] M. Ali, G. Kaddoum, W.-T. Li, C. Yuen, M. Tariq, and H. V. Poor, "A smart digital twin enabled security framework for vehicle-to-grid cyber-physical systems," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 5258–5271, 2023.

[45] F. Boudot, "Efficient proofs that a committed number lies in an interval," in *Proc. 17th Int. Conf. Theory Appl. Cryptograph. Technique*, Bruges, Belgium, May 2000, pp. 431–444.

[46] W. Mao, "Guaranteed correct sharing of integer factorization with off-line shareholders," in *Proc. 1st Int. Workshop Public Key Cryptography (PKC)*, Pacifico Yokohama, Japan, May 1998, pp. 27–42.

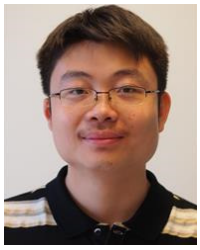


Meng Li (Senior Member, IEEE) received the Ph.D. degree in computer science and technology from the School of Computer Science and Technology, Beijing Institute of Technology (BIT), China, in 2019. He is currently an Associate Professor and a Dean Assistant with the School of Computer Science and Information Engineering, Hefei University of Technology (HFUT), China. He is also a Post-Doctoral Researcher with the Department of Mathematics and HIT Center, University of Padova, Italy, where he is with the Security and PRIvacy Through Zeal (SPRITZ) Research Group led by Prof. Mauro Conti (IEEE Fellow). He was sponsored by ERCIM 'Alain Bensoussan' Fellowship Program (from 2020 to 2021) to conduct post-doctoral research supervised by Prof. Fabio Martinelli at CNR, Italy. He was sponsored by the China Scholarship Council (CSC) (from 2017 to 2018) for the joint Ph.D. study supervised by Prof. Xiaodong Lin (IEEE Fellow) at the Broadband Communications Research (BBCR) Laboratory, University of Waterloo, and Wilfrid Laurier University, Canada. His research interests include security, privacy, applied cryptography, blockchain, and vehicular networks. In this area, he has published more than 60 papers in international peer-reviewed transactions, 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 KNOWLEDGE AND DATA ENGINEERING, *ACM Transactions on Database Systems*, IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON SMART GRID, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, MobiCom, ICICS, SecureComm, TrustCom, and IPCCC. He is an Associate Editor of IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, and IEEE INTERNET OF THINGS JOURNAL.

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.

Qing Wang (Student Member, IEEE) received the B.S. degree in information and computational science from Harbin Normal University, China, in 2019, and the M.S. degree in foundation of mathematics from Shandong University, China, in 2022. She is currently pursuing the Ph.D. degree in computer science and technology with the Hefei University of Technology. Her research interests include applied cryptography and searchable encryption.

Mingwei Zhang received the B.E. degree from the Jiangsu University of Science and Technology in 2021. He is currently pursuing the M.S. degree with the School of Computer Science and Information Engineering, Hefei University of Technology. His research interests include security, privacy, applied cryptography, blockchain, and vehicular networks.



Weizhi Meng (Senior Member, IEEE) received the Ph.D. degree in computer science from the City University of Hong Kong (CityU), Hong Kong. He was a Research Scientist with the Department of Infocomm Security (ICS), Institute for Infocomm Research, A*STAR, Singapore, and a Senior Research Associate with the Department of Computer Science, CityU. He is currently an Associate Professor with the Department of Applied Mathematics and Computer Science, Cybersecurity Section, Technical University of Denmark (DTU), Denmark. His research interests include cybersecurity and intelligent technology in security, including intrusion detection, smartphone security, biometric authentication, HCI security, trust computing, blockchain in security, and malware analysis.

Liehuang Zhu (Senior Member, IEEE) received the M.S. degree in computer science from Wuhan University, Wuhan, China, in 2001, and the Ph.D. degree in computer science from the Beijing Institute of Technology, Beijing, China in 2004. He is currently 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 150 journals 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 for the Special Issue of IEEE WIRELESS COMMUNICATIONS and IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS.

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

Xiaodong Lin (Fellow, IEEE) received the Ph.D. degree in information engineering from the Beijing University of Posts and Telecommunications, China, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Canada. He is currently a Professor with the School of Computer Science, University of Guelph, Canada. His research interests include computer and network security, applied cryptography, computer forensics, and software security. He received the Outstanding Achievement in Graduate Studies Award for his Ph.D. degree.