Efficient Construction of Verifiable Timed Signatures and Its Application in Scalable Payments

Xiaotong Zhou, Debiao He, *Member, IEEE*, Jianting Ning, *Member, IEEE*, Min Luo, and Xinyi Huang, *Member, IEEE*

***Abstract*— Despite the myriad benefits offered by blockchain technology, most of them still face several interrelated issues, such as limited transaction throughput, exorbitant transaction fees, and protracted confirmation times. Payment channel networks have emerged as a promising scalability solution, allowing two mutually distrustful users to engage in multiple off-chain transactions. However, existing schemes based on Hash Time Lock Contract or Anonymous Multi-hop Lock generally cannot ensure strong unlinkability of payments, due to the fact that the time-lock information still remains on the blockchain. To enhance on-chain privacy, a versatile tool was recently proposed by Thyagarajan et al. (*CCS'20*), named *Verifiable Timed Signatures*, but it suffers from the dual insufficiencies of linear-increasing performance and time unverifiability (i.e., performance is linear to the number of signature shares, and signatures cannot be ensured recoverable after the specified time). In this paper, we first propose an approach to reduce computational overhead of VTS, which can be applied to enhance other established schemes, such as VTD (*S&P'22*) and VTLRS (*ESORICS'22*). To further reduce the computational complexity from $\mathcal{O}(n)$ to $\mathcal{O}(1)$, we introduce a new cryptographic primitive called *Verifiable Timed Adaptor Signatures*. Moreover, we extend the VTAS to VTAS$^+$ which provides the security property of verifiable recovery. We demonstrate the practicality of our proposal via presenting a concrete instantiation and constructing a privacy-enhanced payment channel network. Finally, the comprehensive evaluation reveals that our solutions exhibit superior performance than the state-of-the-art schemes.**

Manuscript received 27 April 2023; revised 9 August 2023; accepted 13 August 2023. Date of publication 17 August 2023; date of current version 4 September 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFA1000600; in part by the National Natural Science Foundation of China under Grant U21A20466, Grant 62272350, and Grant 62032005; in part by the New 20 Project of Higher Education of Jinan under Grant 202228017; and in part by the Fundamental Research Funds for the Central Universities under Grant 2042023KF0203. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Lalitha Sankar. *(Corresponding author: Debiao He.)*

Xiaotong Zhou is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China (e-mail: xtzhou163@163.com).

Debiao He is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China (e-mail: hedebiao@163.com).

Jianting Ning is with the Fujian Provincial Key Laboratory of Network Security and Cryptology/Center for Applied Mathematics of Fujian Province, College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China (e-mail: jtning88@gmail.com).

Min Luo is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the Shanghai Technology Innovation Centre of Distributed Privacy-Preserving Artificial Intelligence, Matrix Elements Technologies, Shanghai 201204, China (e-mail: mluo@whu.edu.cn).

Xinyi Huang is with the Artificial Intelligence Thrust, Information Hub, Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511455, China (e-mail: xinyi@ust.hk).

Digital Object Identifier 10.1109/TIFS.2023.3306107

***Index Terms*— Blockchain, cryptocurrency, timed signatures, time lock puzzles, payment channel networks.**

I. Introduction

BLOCKCHAIN, as the foundational technology of cryptocurrencies, makes decentralized transactions more flexible and efficient than traditional financial systems. According to Deloitte reports, the vast majority of retailers plan to accept cryptocurrency payments by 2025,[1] underscoring the growing appeal of this payment method. However, the issue of scalability continues to hinder the widespread adoption of blockchain technology. As an illustration, Bitcoin is limited to only tens of transactions per second, while payment networks like Visa have the capacity to process up to 24,000 transactions per second during periods of high demand [1].

Payment channels (PCs) [2] as well as their extension, *payment channel networks* (PCNs) [3], have emerged as promising solutions to address the above scalability challenges. While these technologies have gained wide adoption for scaling payments in major cryptocurrencies, they still encounter two primary challenges: poor adaptability due to the script mechanism and privacy leakage caused by time-lock transparency. Specifically, Hash Time Lock Contract (HTLC) [4] is a prevalent off-chain payment method that ensures the transfer of digital assets only when a specific condition is met. However, the implementation of HTLC requires the Turing complete script language (e.g., smart contract), which can pose difficulties for scriptless blockchains like Monero and Mimblewimble. An alternative cryptographic component called anonymous multi-hop locks (AMHL) [5] was later proposed to overcome the dependency on scripts. AMHL generalizes the locking mechanism and can be deployed in a scriptless setting. In spite of that, it still depends on time-lock approaches employed by the currencies to facilitate payment expiration. Consequently, AMHL cannot ensure strong hop unlinkability as the time-lock details are even recorded on the chain. Namely, the time-lock of the i-th hop is $\triangle$ (time interval) longer than that of hop $i+1$ (as shown in Fig. 1). Due to the transparency of time-lock on the chain, an attacker can easily correlate this information and detect the multi-hop payment path.

To address the issue of time-lock transparency, a versatile tool, called *verifiable timed signature* (VTS) [6], has been recently proposed to enrich atomic lock. In the context of PCNs, VTS serves as a powerful tool to hide time information

[1] https://www2.deloitte.com/content/dam/Deloitte/us/Documents/technology/us-cons-merchant-getting-ready-for-crypto.pdf

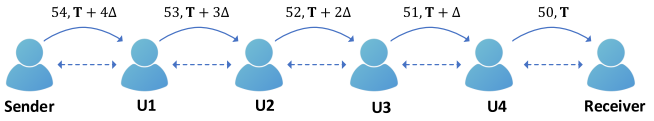

Fig. 1. Payment channel networks. Payment channels are established between successive parties as indicated by dotted lines with two arrowheads. This example illustrates that the Sender transfers 50 coins to the Receiver though four intermediary users, where each of the intermediaries receives a fee of one coin. As the payment travels through each payment, the associated expiration time decreases ($\mathbf{T}+c\Delta$, for $c \in \{0, 1, 2, 3, 4\}$).

and can be employed to redeem the locked funds when channels expire. The verifiability property of VTS ensures that the payer can publicly verify whether a timed signature comprises a valid signature of the redeeming transaction, without having to open it. By employing VTS, the time-lock information is completely removed from on-chain transactions, thereby solving the privacy issues related to payment paths.

Motivation: Based on VTS's construction proposed by Thyagarajan et al. [6], several sequential works such as VTD [7] and VTLRS [8] for efficiency or functionality of scalable payments were introduced. Despite these advancements, two significant bottlenecks still remain: *linear-increasing performance* and *time unverifiability*. In particular, the former is primarily due to the complex t-out-of-n secret sharing [9] and cut-and-choose approach [10] used in VTS, which results in a linearly increasing cost (i.e., $\mathcal{O}(n)$, where n is the total number of signature shares) for a committer to compute verifiable signatures. Moreover, The latter arises when an incorrect delay time is embedded in the commitment, making the signature unextractable even the verifier forces opening after the specified time. This enables a malicious committer to deceive the verifier into approving signatures that cannot be correctly opened after specified time, potentially resulting in financial losses for the verifier. For instance, if the actually delay time is longer than the declared time for the verifier to recover the signature, the verifier may lose money due to the inability to recover the signature within the declared time. In light of these challenges, the following problem naturally arises:

"Can we improve the efficiency of the VTS and enable verifiable recovery for a predefined time?"

Contributions: In this article, we provide an affirmative answer to the aforementioned problem by presenting a proficient method for secret sharing and a cryptographic notion, referred to as *adaptor signature*. The major contributions of this work are fourfold, namely:

- *Performance-improved verifiable timed signature.* We propose an improved VTS (called IVTS), which adopts the structure of VTS proposed in [6] and primarily force on improving the secret sharing of the signature and the public key. Furthermore, the proposed modifications are also applicable to other schemes, such as verifiable timed linkable ring signatures (VTLRS) [8] and verifiable timed Dlog (VTD) [7].
- *Verifiable timed adaptor signature with practical instantiations.* To further optimize the computation cost from $\mathcal{O}(n)$ to $\mathcal{O}(1)$, we introduce a new cryptographic concept termed as *verifiable timed adaptor signature* (VTAS). This primitive enables anyone to authenticate the pre-signature and actually extract the original signature through a sequential computation. In addition, our extension VTAS$^+$ offers *verifiable recovery*[2] to address the time unverifiability issue faced by the existing VTS as well as the proposed IVTS and VTAS. We also present the instantiations with Schnorr and ECDSA, demonstrating the practicality and versatility of our construction.
- *Generic construction of a privacy-preserving scriptless PCN.* Based on VTAS, we propose ASTChannel between two parties to achieve offline-payments without any time-lock contract or script. Moreover, we construct privacy-preserving multi-hop payments ASTNet to maintain the strong unlinkability property.
- *Security analysis and performance evaluation*. We conduct a comprehensive analysis of the privacy and security properties of our schemes and evaluate their performance. The results demonstrate that our proposal is highly suitable for fair contract signing, payment channels, and multisig transactions in the blockchain domain.

II. Preliminaries

We present the concept of PCNs and several building blocks which are utilized in the subsequent construction, including adaptor signatures, time-lock puzzles, verifiable timed signature, and non-interactive zero knowledge.

A. Payment Channels

Payment channel (PC) is a scalability-optimizing technique to enable off-chain transactions between parties without the need to record every transaction on the blockchain. The process involves three steps: (a) *Channel Open:* Two users Alice and Bob first add an on-chain transaction to deposit coins (called channel capacity or balance) into a joint address controlled by both parties. This transaction represents Alice paying up to a specified amount of coins to Bob, which she can redeem them after a given period $\mathbf{T}$; (b) *Channel Update:* Within this time window, Alice and Bob can execute off-chain transactions by reallocating the balance from the joint address; (c) *Channel Closure:* Once the latest payment transaction from the joint address is recorded on the blockchain, the PC is regarded as being closed.

Payment channel networks (PCN) is an extension of PC that allow for multiple connected PCs to form a more scalable and efficient payment system. For a more comprehensive understanding of the topic, we recommend readers refer to [5], [11], and [12] for more detail.

B. Anonymous Multi-Hop Locks (AMHL)

The primitive of AMHL [5] is a promising cryptographic component for script-free off-chain payment, which enables $n-1$ participants to initiate n payment locks (referred to as $l_1, l_2, \ldots, l_n$) in the path. Malavolta et al. [5] introduced the methodologies centered around ECDSA signatures and one-way homomorphic functions. This innovation facilitated the amalgamation of locks featuring diverse signatures within a single payment path. Although AMHL guarantees atomicity of multi-hop payments without relying on script languages, the

[2]Assuming the commit step is successful, the recipient is assured that the forced open algorithm returns a valid value after time $\mathbf{T}$.

temporal correlation of the time-locks hinders its unlinkability property.

C. BLS Signature

We provide a review of the BLS [13]. Let $(\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T, g_0, g_1, q)$ be a bilinear group, which is equipped with an efficiently computable bilinear pairing $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$. $\mathcal{H}$ is a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{G}_1$. The details of BLS are shown as follows.

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$: Select $sk \in \mathbb{Z}_q^*$, calculate $pk = g_0^{sk} \in \mathbb{G}_0$, and return (sk, pk).
- $\sigma \leftarrow \mathsf{Sign}(sk, m)$: Return $\sigma = \mathcal{H}(m)^{sk} \in \mathbb{G}_1$.
- $\perp / 1 \leftarrow \mathsf{Verify}(\sigma, m, pk)$: If $e(g_0, \sigma) = e(pk, \mathcal{H}(m))$ holds, then return 1 and otherwise return $\perp$.

D. Adaptor Signature

An adaptor signature (AS) [14] scheme is generally constructed from a strongly unforgeable digital signature $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrf})$ and a hard relation $(y, Y) \in R$. It is comprised of four main algorithms (pSign, pVrf, Adapt, Ext).

1) $\widehat{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$: It receives a private key sk, a massage m, and a statement Y as inputs, and generates a pre-signature $\widehat{\sigma}$ as output.
2) $\perp / 1 \leftarrow \mathsf{pVrf}(pk, m, Y, \widehat{\sigma})$: It receives a public key pk, a message m, a statement Y, and a pre-signature $\widehat{\sigma}$ as inputs, and outputs 1 if the pre-signature is legitimate and $\perp$ otherwise.
3) $\sigma \leftarrow \mathsf{Adapt}(\widehat{\sigma}, y)$: It receives $\widehat{\sigma}$ and y as inputs, and generates a signature σ as output.
4) $y \leftarrow \mathsf{Ext}(\sigma, \widehat{\sigma}, Y)$: It takes σ, $\widehat{\sigma}$, and Y as inputs, and generates a witness y, or outputs $\perp$.

Definition 1: (Pre-signature Correctness). The *pre-signature correctness* of an AS scheme satisfies the following equation for any message m and a hard relation $(Y, y) \in R$:

$$\Pr\left[\begin{array}{c|c} \mathsf{pVrf}(m, Y, \widehat{\sigma}, pk) = 1 \wedge & (sk, pk) \leftarrow \mathsf{KGen}(1^\lambda) \\ \mathsf{Vrf}(m, \sigma, pk) = 1 \wedge & \widehat{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y) \\ (Y, y') \in R & \sigma := \mathsf{Adapt}(\widehat{\sigma}) \\ & y' := \mathsf{Ext}(\sigma, \widehat{\sigma}, Y) \end{array}\right] = 1.$$

Definition 2: (Unforgeability). The *aEUF-CMA* of an AS scheme is based on the experiment $\mathbf{Exp}_{\mathsf{AS},\mathcal{A}}^{\mathsf{aEUF-CMA}}(1^\lambda)$ (see Fig. 2).

If the advantage of any $\mathcal{PPT}$ adversary $\mathcal{A}$ is negligible, then the AS scheme is considered *aEUF-CMA*:

$$\Pr[\mathbf{Exp}_{\mathsf{AS},\mathcal{A}}^{\mathsf{aEUF-CMA}}(1^\lambda) = 1] = \mathrm{negl}(\lambda).$$

Definition 3: (Witness Extractability) The *witness extractability* of an AS scheme is based on the experiment $\mathbf{Exp}_{\mathsf{AS},\mathcal{A}}^{\mathsf{aWitExt}}(1^\lambda)$ (see Fig. 2). If the advantage of any $\mathcal{PPT}$ adversary $\mathcal{A}$ is negligible, then the AS scheme is considered to have *witness extractability*:

$$\Pr[\mathbf{Exp}_{\mathsf{AS},\mathcal{A}}^{\mathsf{aWitExt}}(1^\lambda) = 1] = \mathrm{negl}(\lambda).$$

The two-party signature scheme $\mathsf{2P\text{-}SIG} = (\mathsf{Setup}, \mathsf{KGen}, \prod_{\mathsf{Sign}}, \mathsf{KAg}, \mathsf{Vrf})$ can be extended to the 2-party adaptor signature (2P-AS) scheme [15], which enables two signers to collaboratively create an adaptor signature by utilizing their individual secret keys. The 2P-AS scheme comprises of the subsequent algorithms.

$\mathbf{Exp}_{\mathsf{AS},\mathcal{A}}^{\mathsf{aEUF-CMA}}(1^\lambda)$

$\mathcal{L} \leftarrow \emptyset$
$(sk, pk) \leftarrow \mathsf{KGen}(1^\lambda)$
$m \leftarrow \mathcal{A}^{\mathsf{Sig}\mathcal{O}(\cdot),\mathsf{pSig}\mathcal{O}(\cdot)}(pk)$
$(Y, y) \leftarrow \mathsf{GenR}(1^\lambda)$
$\widehat{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$
$\sigma \leftarrow \mathcal{A}^{\mathsf{Sig}\mathcal{O}(\cdot),\mathsf{pSig}\mathcal{O}(\cdot)}(\widehat{\sigma}, Y)$
output $(m \notin \mathcal{L} \wedge \mathsf{Vrf}(pk, m, \sigma))$

$\mathbf{Exp}_{\mathsf{AS},\mathcal{A}}^{\mathsf{aWitExt}}(1^\lambda)$

$\mathcal{L} \leftarrow \emptyset$
$(sk, pk) \leftarrow \mathsf{KGen}(1^\lambda)$
$(m, Y) \leftarrow \mathcal{A}^{\mathsf{Sig}\mathcal{O}(\cdot),\mathsf{pSig}\mathcal{O}(\cdot)}(pk)$
$\widehat{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$
$\sigma \leftarrow \mathcal{A}^{\mathsf{Sign}\mathcal{O}(\cdot),\mathsf{pSign}\mathcal{O}(\cdot)}(\widehat{\sigma})$
$y' = \mathsf{Ext}(\sigma, \widehat{\sigma}, Y)$
output $(m \notin \mathcal{L} \wedge (Y, y') \notin R \wedge \mathsf{Vrf}(pk, m, \sigma))$

$\mathsf{Sig}\mathcal{O}(m)$

$\sigma \leftarrow \mathsf{Sign}(sk, m)$
$\mathcal{L} \leftarrow \mathcal{L} \cup m$
output σ

$\mathsf{pSig}\mathcal{O}(m, Y)$

$\widehat{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$
$\mathcal{L} \leftarrow \mathcal{L} \cup m$
output $\widehat{\sigma}$

Fig. 2. Experiments for the AS scheme.

1) $\widehat{\sigma} \leftarrow \prod_{\mathsf{pSign}}(pk_1, pk_2, sk_1, sk_2, m, Y)$: This interaction algorithm receives the key pairs (sk_i, pk_i) from party $\mathcal{P}_i$ $(i \in \{1,2\})$, a massage m, and a statement $Y \in L_R$ as inputs, and outputs a pre-signature $\widehat{\sigma}$.
2) $\perp / 1 \leftarrow \mathsf{pVrf}(apk, m, Y, \widehat{\sigma})$: The pre-verification algorithm receives a joint public key apk, a message m, a statement Y and a pre-signature $\widehat{\sigma}$ as inputs, and outputs 1 if $\widehat{\sigma}$ is legitimate or $\perp$ otherwise.
3) $\sigma \leftarrow \mathsf{Adapt}(apk, \widehat{\sigma}, y)$: The adaptor signature receives apk, $\widehat{\sigma}$ and y as inputs, and returns a valid signature σ.
4) $y \leftarrow \mathsf{Ext}(apk, \sigma, \widehat{\sigma}, Y)$: The extract algorithm receives apk, σ, $\widehat{\sigma}$ and Y as inputs, and then extracts a witness y as output or returns $\perp$.

The more details of 2P-AS are shown in [15].

E. Time-Lock Puzzles

The time-lock puzzles (TLP) allow the creation of a puzzle that can only be solved until a predetermined time period has passed. A TLP scheme is composed of three algorithms as follows:

1) $pp \leftarrow \mathsf{PSetup}(1^\lambda, \mathbf{T})$: It receives a security parameter 1^λ, and a time parameter $\mathbf{T}$ as inputs, and generates the system parameter pp as output.
2) $Z \leftarrow \mathsf{Gen}(pp, s)$: The puzzle generation algorithm receives pp and a secret s as inputs, and returns a time puzzle Z as output.
3) $s \leftarrow \mathsf{Solve}(pp, Z)$: The time puzzle solving algorithm receives pp and a puzzle Z as inputs, and recovers the secret s as output.

Definition 4: (Correctness)A time-lock puzzles scheme $\mathsf{TLP} = (\mathsf{PSetup}, \mathsf{Gen}, \mathsf{Solve})$ is considered correct if for any polynomial $\mathbf{T}$ and any security parameter λ the condition holds:

$$s = \mathsf{Solve}(\mathsf{Gen}(\mathbf{T}, s)), \forall s \in \{0,1\}^*.$$

Definition 5: (Security) For any polynomial-size adversary $\mathcal{A}$ with a polynomial $\hat{\mathbf{T}}(\cdot)$ (where all polynomials $\mathbf{T}(\cdot) \geq \hat{\mathbf{T}}(\cdot)$),

$\mathsf{TLP} = (\mathsf{PSetup}, \mathsf{Gen}, \mathsf{Solve})$ is secure if the following equation is satisfied:

$$\Pr\left[b \leftarrow \mathcal{A}(pp, Z) \,\middle|\, \begin{array}{c} b \leftarrow_{\$} \{0,1\}, \\ Z \leftarrow \mathsf{PGen}(\mathbf{T}(\lambda)) \end{array}\right] \leq \frac{1}{2} + \mathrm{negl}(\lambda).$$

F. Verifiable Timed Signature

VTS [6] is executed between a committer and a verifier. The committer generates a timed signature of time $\mathbf{T}$. The verifier is capable of verifying the validity of the signature σ that was committed in the timed signature, and also has the ability to force the opening of σ after a specific time period $\mathbf{T}$.

The VTS scheme designed for a digital signature scheme $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrf})$, comprises four algorithms.

1) $(C, \pi) \leftarrow \mathsf{Commit}(\sigma, \mathbf{T})$: This algorithm is executed by a committer. It receives a signature σ created via $\mathsf{DS.Sign}$ and a hardness time $\mathbf{T}$ as inputs. It generates a commitment C and a proof π as outputs.
2) $\perp / 1 \leftarrow \mathsf{Vrf}(pk, m, C, \pi)$: It receives a public key pk, a message m, a commitment C and a proof π as inputs. It returns 1 if the signature σ embedded in the commitment is valid, that is $\mathsf{DS.Vrf}(pk, m, \sigma) = 1$. Otherwise, it returns $\perp$.
3) $\sigma \leftarrow \mathsf{Open}(C)$: This opening algorithm is executed by the committer. It receives C as input and recovers the signature σ as output.
4) $\sigma \leftarrow \mathsf{ForceOp}(C)$: It receives the commitment C as input, and recovers σ after time $\mathbf{T}$.

Definition 6: (Soundness) A VTAS scheme $\mathsf{VTAS} = (\mathsf{Commit}, \mathsf{Vrf}, \mathsf{Open}, \mathsf{ForceOp})$ satisfies soundness for the digital signature scheme $\mathsf{DS} = \{\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrf}\}$, if for any $\mathcal{PPT}$ adversary $\mathcal{A}$ and all messages m, the following equation holds:

$$\Pr\left[\begin{array}{c} b_0 = 1 \wedge \\ b_1 = 0 \end{array} \middle| \begin{array}{c} (pk, m, C, \pi, \mathbf{T}) \leftarrow \mathcal{A}(1^{\lambda}) \\ (\sigma, r) \leftarrow \mathsf{ForceOp}(C) \\ b_0 = \mathsf{Vrf}(pk, m, C, \pi) \\ b_1 = \mathsf{DS.Vrf}(pk, m, \sigma) \end{array}\right] \leq \mathrm{negl}(\lambda).$$

Definition 7: (Privacy) A VTAS scheme $\mathsf{VTAS} = (\mathsf{Commit}, \mathsf{Vrf}, \mathsf{Open}, \mathsf{ForceOp})$ satisfies privacy for the digital signature scheme $\mathsf{DS} = \{\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrf}\}$, if there exists a $\mathcal{PPT}$ simulator $\mathcal{S}$ and a polynomial $\hat{\mathbf{T}}(\cdot)$, all PRAM[3] adversaries $\mathcal{A}$ that can run at most $t < \mathbf{T}(\lambda)$. The following equation holds for all messages m:

$$\left|\begin{array}{l} \Pr\left[\mathcal{A}(pk, m, C, \pi) = 1 \,\middle|\, \begin{array}{c} (pk, sk) \leftarrow \mathsf{DS.KGen}(1^{\lambda}) \\ \sigma \leftarrow \mathsf{DS.Sign}(sk, m) \\ (C, \pi) \leftarrow \mathsf{Commit}(\sigma, \mathbf{T}) \end{array}\right] \\ -\Pr\left[\mathcal{A}(pk, m, C, \pi) = 1 \,\middle|\, \begin{array}{c} (pk, sk) \leftarrow \mathsf{DS.KGen}(1^{\lambda}) \\ (C, m, \pi) \leftarrow \mathcal{S}(pk, \mathbf{T}) \end{array}\right] \end{array}\right| \leq \mathrm{negl}(\lambda).$$

G. Non-Interactive Zero Knowledge

A non-interactive zero knowledge (NIZK) protocol [16] involves a prover and a verifier, who interact through a single message. The prover aims to convince the verifier of the knowledge of a certain value without revealing any sensitive information. A NIZK argument scheme with respect to a relation $\mathcal{R}$ comprises the following algorithms.

1) $crs \leftarrow \mathsf{setup}(1^{\lambda})$: It receives a security parameter 1^{λ} as input, and generates a common reference string crs as output.
2) $\pi \leftarrow \mathsf{proof}(crs, x, w)$: The zero-knowledge proof generation algorithm receives crs, a statement x with the witness w as inputs, and generates a proof π as output.
3) $\perp / 1 \leftarrow \mathsf{verify}(crs, x, \pi)$: It receives crs, x and π as inputs, and returns the value 1 when the proof is valid, or $\perp$ otherwise.

NIZK protocol must satisfy three properties, namely *completeness*, *soundness*, and *zero-knowledge* [16].

III. Improved Verifiable Timed Signature

Based on the existing VTS scheme [6], we improve their secret sharing approach and present a more efficient construction (called IVTS) compatible with BLS, Schnorr and ECDSA schemes (the details of Schnorr and ECDSA are provided in Appendix A and B, respectively). We demonstrate that IVTS provides significant performance improvements and the same security guarantees as the original VTS. Moreover, this enhanced approach can also be applied to other timed cryptography-related signatures, such as [7] and [8].

We will present IVTS scheme for BLS, which relies on the use of time-lock puzzles, particularly the Linearly-Homomorphic-TLP [17] (LTLP), and NIZK range proof (ZK) for the corresponding language $\mathcal{L}$.

$$\mathcal{L}_{\mathsf{range}} := \left\{\begin{array}{l} \mathsf{stmt} = (Z, a, b, \mathbf{T}) : \exists \mathsf{wit} = (\sigma, r) \text{ s.t.} \\ (Z \leftarrow \mathsf{LTLP.PGen}(\mathbf{T}, \sigma, r)) \wedge (\sigma \in [a, b]) \end{array}\right\}$$

We set the security parameter of secret shares as n and t is defined as $n/2 + 1$. The size of the signature σ is to be $|\sigma| = \lambda$. Let $\mathcal{H}^*$ be a secure hash function $\mathcal{H}^* : \{0,1\}^* \rightarrow I \subset \{1, 2, \ldots, n\}$, where $|I| = t - 1$. We simplify the assumption that the **Force Open** algorithm is capable of solving approximately $n - t + 1$ puzzles simultaneously.

Setup: With the input 1^{λ}, it generates the parameter $sp \leftarrow \mathsf{LTLP.PSetup}(1^{\lambda})$ and $crs_{\mathsf{range}} \leftarrow \mathsf{ZKsetup}(1^{\lambda})$, it outputs $pp = \{sp, crs_{\mathsf{range}}\}$.

Commit: With the input $(pp, \sigma, pk, m, \mathbf{T})$, this algorithm executes as follows.

- Select a random $t-1$ degree-polynomial $p(x) = \sum_{i=1}^{t-1} a_i x^i$, where $\{a_i\}_{i \in \{1,..t-1\}}$ are random numbers.
- For all $i \in [1, n]$, calculate

$$\sigma_i = \sigma \cdot \mathcal{H}(m)^{p(i)},$$
$$pk_i = pk \cdot g_0^{p(i)},$$

  where σ_i, pk_i are t-out-of-n Shamir secret shares, respectively. The signature σ can be recovered from any subset of t shares out of the total of n shares.
- For all $i \in [1, n]$, select $r_i = \{0,1\}^{\lambda}$ and generate time-lock puzzles of σ_i, that is,

$$Z_i = \mathsf{LTLP.PGen}(pp, \sigma_i, \mathbf{T}),$$
$$\pi_{\mathsf{range},i} = \mathsf{ZK.proof}(crs_{\mathsf{range}}, (Z_i, 0, 2^{\lambda}, \mathbf{T}), (\sigma_i, r_i)).$$

[3]PRAM algorithms are computational procedures that can be completed in polynomial time with a polynomial amount of parallel computational resources.

- Invoke the hash function $I \leftarrow \mathcal{H}^*(pk, m, (pk_1, Z_1, \pi_1), \ldots, (pk_n, Z_n, \pi_n))$.
- Return the commitment $C = (Z_1, \ldots, Z_n, \mathbf{T})$ and the proof $\pi = (\{pk_i\}_{i\in\{1,\ldots,n\}}, I, \{\sigma_i\}_{i\in I})$.

Verification: With the input (pp, m, pk, C, π), the verification algorithm executes as follows.

- If none of the conditions below are satisfied output 1, otherwise output $\perp$:
  1) $\exists j \notin I$ s.t. $\prod_{i\in I} pk_i^{l_i(0)} \cdot pk_j^{l_j(0)} \neq pk$, where $l_i(\cdot)$ represents the Lagrange polynomial basis for the i-th share.
  2) $\exists i \in [1, n]$ s.t. $\mathsf{ZK.verify}(crs_{\mathsf{range}}, (Z_i, 0, 2^\lambda, \mathbf{T}), \pi_{\mathsf{range},i}) \neq 1$.
  3) $\exists i \in I$ s.t. $Z_i \neq \mathsf{LTLP.PGen}(pp, \sigma_i, \mathbf{T})$ or $e(g_0, \sigma_i) \neq e(pk_i, \mathcal{H}(m))$.
  4) $I \neq \mathcal{H}^*(pk, m, (pk_1, Z_1, \pi_1), \ldots, (pk_n, Z_n, \pi_n))$.

Open: It outputs the complete signature σ.

FOpen: With the input (pp, m, pk, C, π), the force open algorithm executes as follows.

- Select a random number $v \in [1, n]/I$ and compute $\sigma_v \leftarrow \mathsf{LTLP.PSolve}(pp, Z_v)$. As $t-1$ puzzles have been opened in the commit phase, the force open algorithm is only required to solve only one time-lock puzzle.
- Recover the secret $\sigma = \prod_{i\in\{I,v\}}(\sigma_i)^{l_i(0)}$, and output σ.

Our proposed VTS scheme for BLS satisfies privacy and soundness. The formal proofs for these theorems are shown in Section VI-A.

IV. Verifiable Timed Adaptor Signature

Although IVTS has improved the performance of Thyagarajan et. al.'s scheme [6], the problem of linear correlation between system parameters (e.g., the number of signature shares n) and performance still exists. To achieve the efficiency of $O(1)$, we introduce a new cryptographic tool called VTAS, together with a generic construction of VTAS and two efficient instantiations compatible with Schnorr and ECDSA. It is important to emphasize that while VTAS boasts impressive performance attributes, it cannot apply to unique signatures, such as BLS, due to constraints associated with adapter signatures [15].

A. Definition

A VTAS for a digital signature scheme $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrf})$, which can construct an adaptor signature scheme $\mathsf{AS} = (\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrf}, \mathsf{Ext})$, includes the following four algorithms.

1) $(C, \pi) \leftarrow \mathsf{Commit}(pk, \hat{\sigma}, \sigma, Y, \mathbf{T}, m)$: This algorithm receives a public key pk, pre-signature $\hat{\sigma}$ (generated via $\mathsf{AS.pSign}(sk, m, Y)$), a signature σ, a statement Y, a hardness time $\mathbf{T}$ and message m as inputs, and generates a commitment C and a proof π as outputs.
2) $\perp / 1 \leftarrow \mathsf{Vrf}(pk, C, \pi, m)$: It receives a public key pk, a commitment C, a proof π and a message m as inputs, and outputs 1 if the signature σ embedded in (C, π) are valid and the signature can be recovered after time $\mathbf{T}$ (i.e., $\mathsf{AS.pVrf}(pk, m, Y, \hat{\sigma}) = 1$, $\mathsf{DS.Vrf}(pk, m, \sigma) = 1$, $(Y, y) \in R$). Otherwise, it outputs $\perp$.
3) $\sigma \leftarrow \mathsf{Open}(C, y)$: This algorithm is executed by the committer. It receives C and y as inputs, and outputs the signature σ.
4) $\sigma \leftarrow \mathsf{ForceOp}(C)$: This algorithm receives C as input, and outputs σ after time $\mathbf{T}$.

A VTAS scheme has two security requirements: *privacy* and *soundness*, which are formalized as below.

Definition 8: (Soundness) A VTAS scheme $\mathsf{VTAS} = (\mathsf{Commit}, \mathsf{Vrf}, \mathsf{Open}, \mathsf{ForceOp})$ satisfies soundness for the digital signature scheme $\mathsf{DS} = \{\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrf}\}$, if for all $\mathcal{PPT}$ adversary $\mathcal{A}$ the equation holds for all messages $m \in \{0, 1\}^*$:

$$\Pr\left[\begin{array}{c|c} & (pk, m, \hat{\sigma}, Y, C, \pi, \mathbf{T}) \leftarrow \mathcal{A}(1^\lambda) \\ b_0 = 1 \wedge & (\sigma, r) \leftarrow \mathsf{ForceOp}(C) \\ b_1 = 0 & b_0 = \mathsf{VTAS.Vrf}(pk, m, \hat{\sigma}, Y, C, \pi) \\ & b_1 = \mathsf{DS.Vrf}(pk, m, \sigma) \end{array}\right] \leq \mathrm{negl}(\lambda).$$

Definition 9: (Privacy) A VTAS scheme $\mathsf{VTAS} = (\mathsf{Commit}, \mathsf{Vrf}, \mathsf{Open}, \mathsf{ForceOp})$ satisfies privacy for the signature scheme $\mathsf{DS} = \{\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrf}\}$ and corresponding adaptor signature scheme $\mathsf{AS} = \{\mathsf{pSign}, \mathsf{pVrf}, \mathsf{Adapt}, \mathsf{Ext}\}$, if there exists a $\mathcal{PPT}$ simulator $\mathcal{S}$ and a polynomial $\hat{\mathbf{T}}(\cdot)$, all PRAM adversaries $\mathcal{A}$ who can run at most $t < \mathbf{T}(\lambda)$, the following equation holds for all messages $m \in \{0, 1\}^*$:

$$\left|\begin{array}{l} \Pr\left[\begin{array}{c|c} & (pk, sk) \leftarrow \mathsf{DS.KGen}(1^\lambda) \\ \mathcal{A}(pk, m, \hat{\sigma}, & \sigma \leftarrow \mathsf{AS.pSign}(sk, m, Y) \\ Y, C, \pi) = 1 & \hat{\sigma} \leftarrow \mathsf{AS.Adapt}(\sigma, y) \\ & (C, \pi) \leftarrow \mathsf{VTAS.Commit}(\hat{\sigma}, y, \mathbf{T}) \end{array}\right] \\ -\Pr\left[\begin{array}{c|c} \mathcal{A}(pk, m, \hat{\sigma}, & (pk, sk) \leftarrow \mathsf{DS.KGen}(1^\lambda) \\ Y, C, \pi) = 1 & (m, \hat{\sigma}, Y, C, \pi) \leftarrow \mathcal{S}(pk, \mathbf{T}) \end{array}\right] \end{array}\right| \leq \mathrm{negl}(\lambda).$$

B. Generic Construction

We next focus on constructing a generic VTAS scheme from a secure adaptor signature scheme. Our approach involves four essential building blocks: digital signature scheme (DS), corresponding adaptor signature (AS), time-lock puzzle (TLP) and proof system (ZK). The specific steps of the VTAS construction are presented as follows.

Setup: With the input 1^λ, it executes as follows.

- Generate the parameters of time-lock puzzles $pp \leftarrow \mathsf{TLP.PSetup}(1^\lambda)$.
- Generate the parameter of NIZK $crs \leftarrow \mathsf{ZK.setup}(1^\lambda)$.
- Output system parameters (pp, crs).

Commit: With the input $(pp, pk, \hat{\sigma}, \sigma, Y, \mathbf{T})$, the commit algorithm executes as follows.

- Verify the input data. If $\mathsf{AS.pVrf}(pk, m, Y, \hat{\sigma}) \neq 1$ or $\mathsf{DS.Vrf}(pk, m, \sigma) \neq 1$, output $\perp$. Otherwise, compute $y \leftarrow \mathsf{Ext}(\sigma, \hat{\sigma}, Y)$.
- Compute $Z = \mathsf{TLP.PGen}(pp, y)$ and $\pi \leftarrow \mathsf{ZK.proof}(crs, (Z_i, Y), y)$.
- Output $C = (Z, \mathbf{T})$ and π.

$\underline{\mathsf{pSign}(sk, m, Y)}$
$x = sk$
$k \leftarrow_{\$} \mathbb{Z}_q$
$r = \mathcal{H}((g^k \cdot Y)||m)$
$\hat{s} = k + r \cdot x$
return $\widehat{\sigma} = (r, \hat{s})$

$\underline{\mathsf{pVrf}(pk, m, Y, \widehat{\sigma})}$
$Q = g^{\hat{s}} \cdot pk^{-r} \cdot Y$
$r' = \mathcal{H}(Q||m)$
return$(r := r')$

$\underline{\mathsf{Adapt}(\widehat{\sigma}, y)}$
$s = \hat{s} + y$
return(r, s)

$\underline{\mathsf{Ext}(\sigma, \hat{\sigma}, Y)}$
$y^* = s - \hat{s}$
if $(Y, y^*) \in R$
then return y^*
else return $\perp$

Fig. 3. Schnorr-based adaptor signature scheme directly cited from [15].

$\underline{\mathsf{pSign}(sk, m, I_Y)}$
$x = sk, (Y, \pi_Y) = I_Y$
$K \leftarrow_{\$} \mathbb{Z}_q, \hat{K} = g^k$
$K = Y^k, r = f(K)$
$\hat{s} = k^{-1}(\mathcal{H}(m) + r \cdot x)$
$\pi = \mathsf{P}_Y((\hat{K}, K), k)$
return $\widehat{\sigma} = (r, \hat{s}, K, \pi)$

$\underline{\mathsf{pVrf}(pk, m, I_Y, \widehat{\sigma})}$
$u = \mathcal{H}(m) \cdot \hat{s}^{-1}$
$v = r \cdot \hat{s}^{-1}$
$K' = g^u \cdot pk^v$
return $((I_Y \in L_R) \wedge (r = f(K)) \wedge \mathsf{V}_Y(K', K), \pi))$

$\underline{\mathsf{Adapt}(\widehat{\sigma}, y)}$
$s = \hat{s} \cdot y^{-1}$
return(r, s)

$\underline{\mathsf{Ext}(\sigma, \widehat{\sigma}, I_Y)}$
$y^* = s^{-1} \cdot \hat{s}$
if $(I_Y, y^*) \in R$
then return y^*
else return $\perp$

Fig. 4. ECDSA-based adaptor signature directly cited from [15].

Verification: With the input $(pp, m, pk, \hat{\sigma}, Y, C, \pi)$, the verification algorithm executes as follows.

- If $\mathsf{AS.pVrf}(pk, m, Y, \hat{\sigma}) \neq 1$ or $\mathsf{ZK.verify}(crs, (Z, Y), \pi) \neq 1$, return $\perp$. Otherwise, return 1.

Open: With the input $(pp, pk, \hat{\sigma}, \sigma, Y, \mathbf{T})$, the Open algorithm outputs σ.

Force Open: With the input $(pp, m, pk, \hat{\sigma}, Y, C, \pi)$, the force open algorithm executes as follows.

- Compute $y \leftarrow \mathsf{TLP.Solve}(pp, Z)$. In order to recover σ, the force open algorithm has to solve a time-lock puzzle.
- Compute $\sigma \leftarrow \mathsf{AS.Adapt}(\hat{\sigma}, y)$, and output σ.

C. Efficient Instantiation

Our instantiation consists of three components: a standard digital signature $\mathsf{DS}^{\mathsf{SIG}}$, an adaptor signature $\mathsf{AS}^{\mathsf{SIG}}$ and a time-lock puzzle TLP. Specially, VTAS can be implemented using Schnorr (i.e., $\mathsf{SIG} = \mathsf{Schnorr}$), ECDSA (i.e., $\mathsf{SIG} = \mathsf{ECDSA}$) or other schemes. The corresponding adaptor signatures $\mathsf{AS}^{\mathsf{Schnorr}}$, $\mathsf{AS}^{\mathsf{ECDSA}}$ are shown in Fig. 3 and Fig. 4, respectively. Additionally, we employ a secure TLP from [17]. The instaniation is described as follows.

Setup: With the input 1^λ, this algorithm generates the system parameters of TLP, AS and NIZK.

- Select two prime numbers (p', q') and let $N = p' \cdot q'$. Send p' and q' to committer securely.
- Pick $\hat{g} \in \mathbb{Z}_N^*$ and compute $g_1 = -\hat{g}^2 \pmod N$.
- Initialize the adaptor signature ($\mathsf{AS}^{\mathsf{Schnorr}}$ or $\mathsf{AS}^{\mathsf{ECDSA}}$) and obtain $(\mathbb{G}, p, g_2)$.
- Publish $pp = (N, g_1, \mathbb{G}, p, g_2)$.

Commit: Let $\mathbf{T} = 2^k$ be an integer. With the input $(pp, pk, \widehat{\sigma}, \sigma, Y, 2^k)$, the commit algorithm executes as follows.

- Verify the input data. If $\mathsf{AS}^{\mathsf{SIG}}.\mathsf{pVrf}(pk, m, Y, \widehat{\sigma}) \neq 1$ or $\mathsf{DS}^{\mathsf{SIG}}.\mathsf{Vrf}(pk, m, \sigma) \neq 1$, output $\perp$. Otherwise, compute $y \leftarrow \mathsf{AS}^{\mathsf{SIG}}.\mathsf{Ext}(\sigma, \widehat{\sigma}, Y)$.
- Generate a time puzzle.
  1) Compute $h = g_1^{2^{2^k}} \pmod N$ using modular reduction of 2^{2^k} by $\phi(N)/2$ for optimization.
  2) Select a random number $r \in \mathbb{Z}_{N^2}^*$ to compute $u = g_1^r \pmod N$ and $v = h^{r \cdot N} \cdot (1 + N)^y \pmod{N^2}$.
  3) The time puzzle is set as $Z = (u, v)$.
- Generate a NIZK proof, where the statement is $x = (Y, Z)$ and the witness is $w = (y, r)$.
  1) Select two random numbers $\alpha \in \mathbb{Z}_{N^2}^*$, $\beta \in \mathbb{Z}_q^*$ to compute $R_1 = g_1^\alpha \pmod N$, $R_2 = h^{\alpha \cdot N}(1 + N)^\beta \pmod{N^2}$ and $R_3 = g_2^\beta$.
  2) Compute $e = \mathcal{H}(N, g_1, g_2, pk, \widehat{\sigma}, Y, Z, R_1, R_2, R_3)$, and then set $z_1 = \alpha - e \cdot r$, $z_2 = \beta - e \cdot y$.
  3) The NIZK proof is set as $\pi = (R_1, R_2, R_3, z_1, z_2)$.
- Output $C = (Z, 2^k)$ and π.

Verification: With the input $(pp, m, pk, \widehat{\sigma}, Y, C, \pi)$, the verification algorithm executes as follows.

- Parse $C = (Z, 2^k)$ and $\pi = (R_1, R_2, R_3, e, z_1, z_2)$.
- If $\mathsf{AS}^{\mathsf{SIG}}.\mathsf{pVrf}(pk, m, Y, \widehat{\sigma}) \neq 1$, output $\perp$ and 1 otherwise.
- Compute $e = \mathcal{H}(N, g_1, g_2, pk, \widehat{\sigma}, Y, Z, R_1, R_2, R_3)$.
- Check whether the equations $R_1 = g_1^{z_1} \cdot u^e$, $R_2 = v^e \cdot h^{z_1 \cdot N}(1 + N)^{z_2}$, $R_3 = g_2^{z_2} Y^e$ hold. If the above equations all hold, output 1. Otherwise, output $\perp$.

Open: With the input $(pp, pk, \widehat{\sigma}, \sigma, Y, 2^k)$, the Open algorithm outputs σ.

FOpen: With the input $(pp, m, pk, \widehat{\sigma}, Y, C, \pi)$, the force open algorithm executes as follows.

- Parse $C = (Z, 2^k)$, $Z = (u, v)$.
- Compute $w = u^{2^T} \pmod N$ via repeated squaring.
- Recover the time-lock solution

$$y = \frac{v/(w)^N \pmod{N^2} - 1}{N}.$$

- Compute $\sigma \leftarrow \mathsf{AS}^{\mathsf{SIG}}.\mathsf{Adapt}(\widehat{\sigma}, y)$, and output σ.

Our proposed VTAS satisfies privacy and soundness. The security theorems and proofs are shown in section VI-B.

D. Extension

We also propose VTAS$^+$, a security-enhanced VTAS that utilizes the strong fair property of timed commitments protocol [18]. In practice, a malicious committer may try to deceive the verifier into accepting signatures that can not be opened after specified time, which poses a significant threat

to fairness. VTAS$^+$ addresses this problem by allowing the verifier to verify that the commitment can be forcibly opened in 2^k sequential steps. The most operations of VTAS$^+$ are similar to that of VTAS, except that the following:

During the commitment phase, the committer is required to demonstrate that h has been correctly generated, specifically, that h satisfies the equation $h = g_1^{2^{2^k}} \pmod N$.

1) Construct the vector L of length k,

$$L = \left\langle g_1^2, g_1^4, g_1^{16}, g_1^{256}, \ldots, g_1^{2^{2^i}}, \ldots, g_1^{2^{2^k}} \right\rangle \pmod N$$
$$= \langle b_0, b_1, b_3, b_4, \ldots, b_i, \ldots, b_k \rangle .$$

To demonstrate that the tuple (g_1, b_{i-1}, b_i) satisfies the form $(g_1, g_1^{\mu}, g_1^{\mu^2})$ for some μ, the committer must provide proof for each $i \in [1, k]$ as follows.

2) Select random numbers $\varphi_1, \varphi_2, \ldots, \varphi_k \in \mathbb{Z}_q^*$ to compute $a_i = g_1^{\varphi_i}$, $w_i = b_{i-1}^{\varphi_i}$ for $i = 1, \ldots, k$.
3) For $i = 1, \ldots, k$, compute $c_i = \mathcal{H}(g_1, b_{i-1}, b_i, a_i, w_i)$ and $h_i = c_i \cdot 2^{2^{i-1}} + \varphi_i$.
4) Set $\pi_1 = (R_1, R_2, R_3, e, z_1, z_2)$, $\pi_{2,i} = (a_i, w_i, h_i)$. The NIZK proof is set as $\pi_2 = (\pi_{2,1}, \pi_{2,2}, \ldots \pi_{2,k})$.
5) Output $C = (Z, 2^k)$ and $\pi = (\pi_1, \pi_2)$.

In the verification phase, the verifier will check the construction of h via the following.

- Parse $C = (Z, 2^k)$ and $\pi_1 = (R_1, R_2, R_3, e, z_1, z_2)$, $\pi_{2,i} = (a_i, w_i, h_i)$.
- Compute $c_i = \mathcal{H}(g_1, b_{i-1}, b_i, a_i, w_i)$ where $i = 1, \ldots, k$.
- Check whether the equations $g_1^{h_i} \cdot b_{i-1}^{-c_i} = a_i$, $b_{i-1}^{h_i} \cdot b_i^{-c_i} = w_i$ hold. If all the mentioned equations satisfied, output 1 or $\perp$ otherwise.

As the aforementioned security modifications, VTAS$^+$ possesses the property of verifiable recovery. The practical design of this function involves the integration of zero-knowledge proofs, ensuring the accurate embedding of the witness k, thereby safeguarding the completeness, soundness, and zero-knowledge properties, all of which do not compromise the security of VTAS$^+$. In terms of implementation, these k proofs and verifications can all be carried out in multi-threading framework.

V. Scalable Payment Applications

In this section, we utilize VTAS to construct a payment channel ASTChannel for scriptless blockchains, and further build a payment channel network ASTNet with strong privacy protection. If the payment channels demand higher levels of assurance, such as verifiable recovery, we can also employ VTAS$^+$ to design the channels without further elaboration.

A. Payment Channel

The main construction of our proposed unidirectional payment channel ASTChannel is illustrated in Fig. 5.

1) Channel Initialization: Firstly, Alice and Bob pick their partial private keys sk_{A} and sk_{B} as well as create their joint public key vk_{AB} (line 1). Then, they create three types of transactions: a funding transaction $Tx_{\mathsf{F}} = (vk_{\mathsf{A}} : bal_{\mathsf{A}} || vk_{\mathsf{AB}} : bal_{\mathsf{C}})$, a redeeming transaction $Tx_{\mathsf{R}} = (vk_{\mathsf{AB}} : bal_{\mathsf{C}} || vk_{\mathsf{A}} : bal_{\mathsf{A}})$ and a commitment transaction $Tx_{\mathsf{C}}^0 = (vk_{\mathsf{AB}} : bal_{\mathsf{C}} || vk_{\mathsf{A}} : bal_{\mathsf{A}}^0, vk_{\mathsf{B}} : bal_{\mathsf{B}}^0)$. The funding transaction Tx_{F} denotes that Alice transfers channel capacity bal_{A} from vk_{A} to the joint address vk_{AB}. The redeeming transaction is utilized to retrieve the locked balance bal_{A} from vk_{AB} to vk_{A}. The commitment transaction Tx_{C}^0 is used to adjust the distribution of the balance from vk_{AB} to vk_{A} and vk_{B}. They generate two-party pre-signature $\widehat{\sigma}_{\mathsf{R}}$ of the redeeming transaction Tx_{R}, and Bob retrieves the signature σ_{R} through the witness y (Line 4-5). Subsequently, Bob creates the timed commitment and proof (C, π) of signature σ_{R}, and transmits them to Alice for verification (Line 6). If the verification process is successful, Alice signs Tx_{F} by employing her private key sk_{A} (Line 7). Following that, Alice and Bob jointly generate the pre-signature $\widehat{\sigma}_{vk_{\mathsf{AB}}^0}$ (Line 8-9). Eventually, Alice broadcasts the signed transaction Tx_{F}.

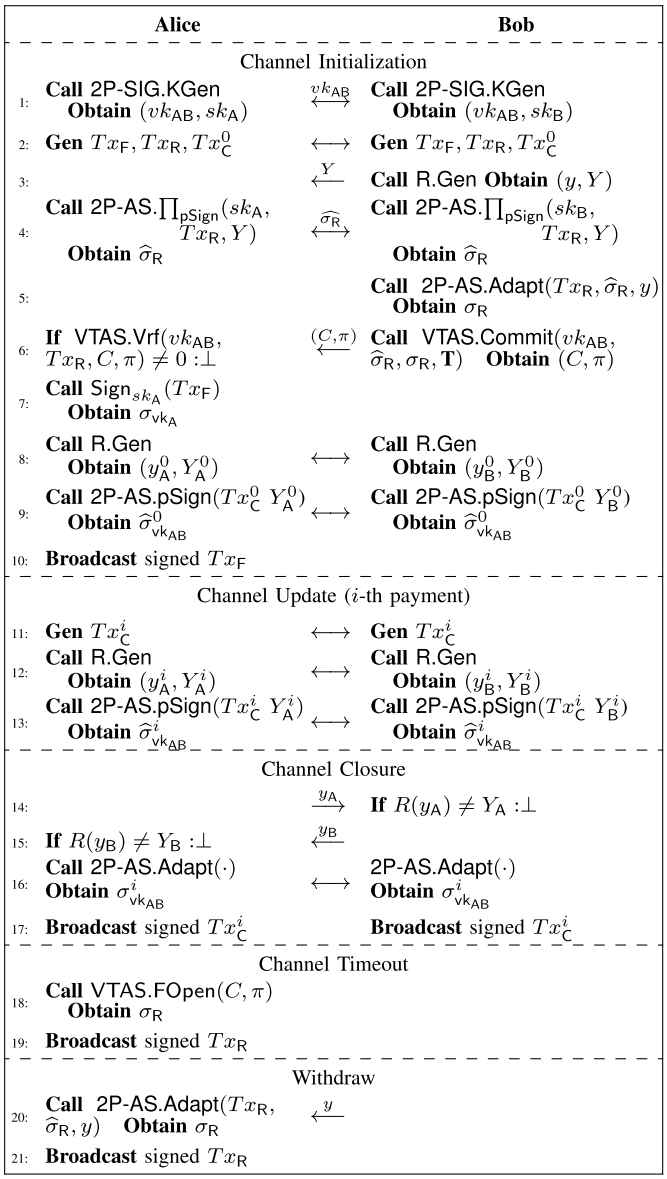

Fig. 5. The construction of ASTChannel.

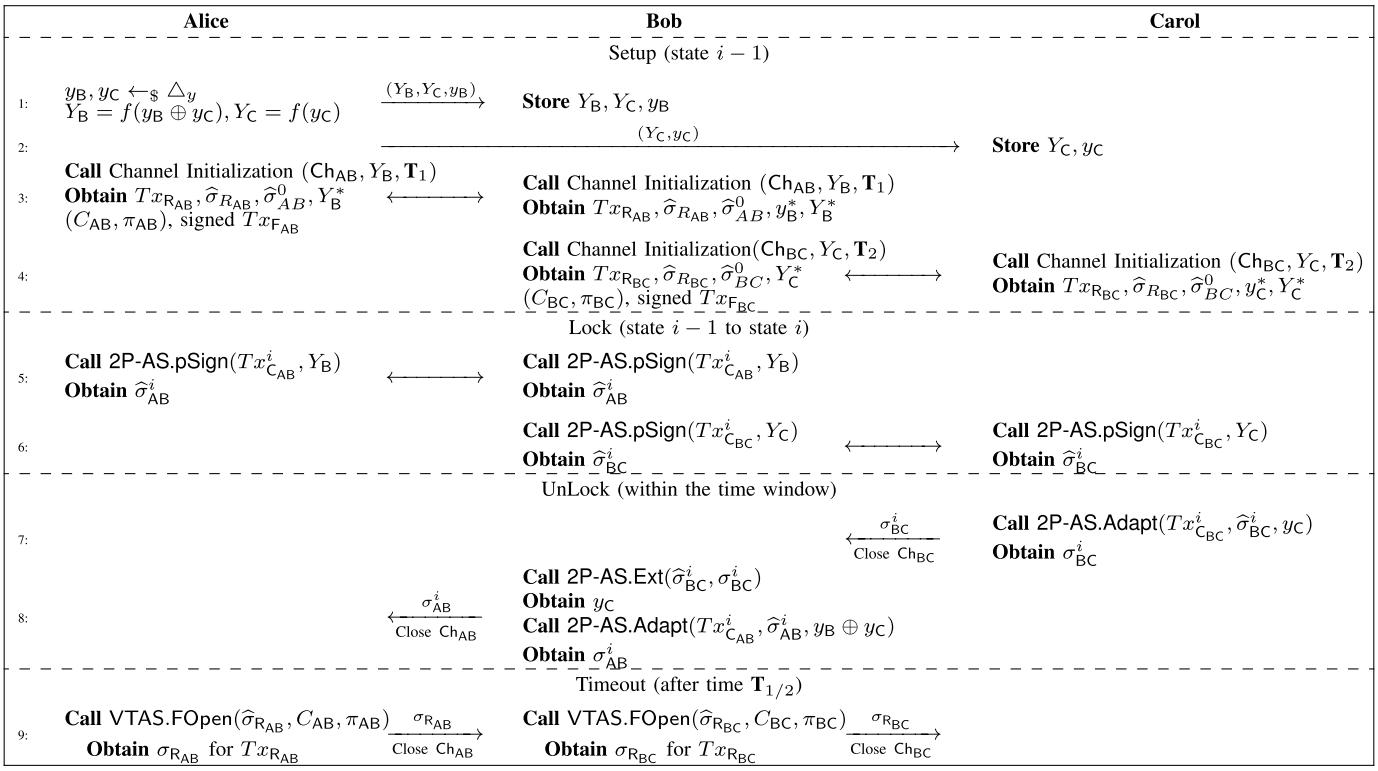

Fig. 6. The construction of ASTNet.

2) Channel Update: Alice and Bob update the channel commitment $Tx^i_{\mathsf{C}} = (vk_{\mathsf{AB}} : bal_{\mathsf{C}} || vk_{\mathsf{A}} : bal^i_{\mathsf{A}}, vk_{\mathsf{B}} : bal^i_{\mathsf{B}})$ (Line 11). Then, they regenerate and verify the relations $(y^i_{\mathsf{A}}, Y^i_{\mathsf{A}})$, $(y^i_{\mathsf{A}}, Y^i_{\mathsf{A}})$ respectively (Line 12). Finally, Alice and Bob create a new two-party pre-signature $\widehat{\sigma}_{vk^i_{\mathsf{AB}}}$ of Tx^i_{C} by utilizing their partial private key sk_{A} and sk_{B} (Line 13).

3) Channel Closure: Upon deciding to close the channel, Alice and Bob share and verify the latest witness y_{A} and y_{B} (Line 14-15). Then, Alice and Bob adapt and obtain the full signature of the Tx^i_{C} (Line 16). Each of them can update the signed transaction into the blockchain (Line 17).

4) Channel Timeout: If the channel expires, Alice will have sufficient time to solve the timed-lock puzzle. She can withdraw her funds by invoking VTAS.FOpen, which returns the signature σ_{R} for Tx_{R} (Line 18). After that, Alice can broadcast the redeeming transaction to the blockchain for claiming her funds (Line 19).

5) Withdraw: If any of the parties intend to withdraw the funds and close the channel, Bob will reveal the witness y to Alice (Line 20). Alice can then adapt the signature of Tx_{R} and update the signed Tx_{R} to the blockchain (Line 21).

In terms of security, the privacy offered by VTAS guarantees that Alice will not have knowledge of the signature σ_{R} for Tx_{R} prior to time $\mathbf{T}$. Therefore, Alice is unable to prematurely close the channel. Furthermore, the soundness of VTAS guarantees that Alice can obtain the valid signature σ_{R} on the refund transaction after time $\mathbf{T}$.

B. Payment Channel Network

We further discuss the extensions of our proposed PC that makes it applicable in the payment channel networks, called ASTNet. To ensure clarity, let us examine a scenario where Alice plans to make a coin transfer to Carol through the payment channels set up with Bob, who establishes payment channels with both Alice and Carol (see Fig. 6).

Setup: To initiate the payment, Alice first generates statements and witnesses for each PC along the payment path. Specifically, she randomly selects two witness $y_{\mathsf{B}}, y_{\mathsf{C}}$ and computes $Y_{\mathsf{B}} = f(y_{\mathsf{B}} \oplus y_{\mathsf{C}})$ and $Y_{\mathsf{C}} = f(y_{\mathsf{C}})$, where $f : w \rightarrow x$ generates a statement x from the witness w. Alice sends $(Y_{\mathsf{C}}, y_{\mathsf{C}})$ to Carol and $(Y_{\mathsf{B}}, Y_{\mathsf{C}}, y_{\mathsf{B}})$ to Bob (Line 1-2). Then, Bob initializes payment channel $\mathsf{Ch}_{\mathsf{AB}}$ with Alice and $\mathsf{Ch}_{\mathsf{BC}}$ with Carol, respectively. (Line 3-4). For clarity, the time intervals $\triangle_{\mathbf{T}}$ between two channels are omitted.

Lock: All the parties obtain the channel statements (as state i-1). Alice and Bob pre-sign $Tx^i_{\mathsf{C}_{\mathsf{AB}}}$ with Y_{B} in order to lock the channel $\mathsf{Ch}_{\mathsf{AB}}$ (Line 5). Similarly, Bob and Carol use Y_{C} to lock the channel $\mathsf{Ch}_{\mathsf{BC}}$ (Line 6). The state of each channel is updated to i.

Unlock: Within the time window, the receiver Carol obtains σ^i_{BC} by adapting $\widehat{\sigma}^i_{\mathsf{BC}}$ with the witness y_{C}. Then, Carol broadcasts the signed transaction $Tx_{\mathsf{C}_{\mathsf{BC}}}$ to close the channel $\mathsf{Ch}_{\mathsf{BC}}$ (Line 7). Subsequently, Bob extracts y_{C} from $\widehat{\sigma}^i_{\mathsf{BC}}$ and σ^i_{BC}, computes the witness $y_{\mathsf{B}} \oplus y_{\mathsf{C}}$, and uses it to recovers the signature σ^i_{AB} to open the left lock (Line 8). Finally, the payment is complete.

Timeout: After a time interval of $\mathbf{T}$, Alice and Bob solve the time-lock puzzle $(y^*_{\mathsf{B}}, y^*_{\mathsf{C}})$ and obtain $\sigma_{\mathsf{R}_{\mathsf{AB}}}$ and $\sigma_{\mathsf{R}_{\mathsf{BC}}}$, respectively (Line 9). It enables them to redeem the locked coins from the channel by publishing $(Tx_{\mathsf{R}_{\mathsf{AB}}}, \sigma_{\mathsf{R}_{\mathsf{AB}}})$ and $(Tx_{\mathsf{R}_{\mathsf{BC}}}, \sigma_{\mathsf{R}_{\mathsf{BC}}})$ on the blockchain.

Our proposed ASTNet can mitigate time-lock correlation attacks and enhance the on-chain privacy of AMHL. The VTAS privacy feature ensures that neither Alice nor Bob can broadcast the refund transaction before their respective time constraints. In addition, the soundness of VTAS further guarantees that Alice and Bob can open $\sigma_{\mathsf{R_{AB}}}$ and $\sigma_{\mathsf{R_{BC}}}$ upon reaching the timeout.

VI. Security Analysis

We present a comprehensive security analysis of our schemes, including IVTS and VTAS.

A. Security Analysis of the Improved VTS

The security analysis of IVTS (proposed in Section III) closely follows that of [6], which is also applicable to Schnorr-based and ECDSA-based schemes.

Theorem 1: (Soundness) Suppose LTLP is a secure time-lock puzzle scheme, then our proposed IVTS for BLS satisfies soundness property as described in Definition 6.

Proof: The proof of the soundness theorem can be examined by analyzing its interactive version using the Fiat-Shamir approach to protocols with a fixed number of rounds.

Assuming an efficient adversary $\mathcal{A}$ can break the soundness of IVTS by producing a commitment $(Z_1, Z_2 \ldots, Z_n)$ that violates the verification algorithm, a valid signature on m can be recovered by performing interpolation on σ_i where $\sigma_i \in I$ and satisfies the verification algorithm. By the soundness of the range NIZK, the probability of a puzzle being will-formed is negligible. Given $(Z_1, Z_2 \ldots, Z_n)$, a polynomial-time algorithm can recover a set I^* by solving the puzzles and verifying which signatures satisfy the verification algorithm. For the verifier to accept, I^* must equal I, meaning that the likelihood of the prover accurately guessing a randomly selected n-bit string with exactly $n/2$-0s is $(n/2!)^2/n!$. The above statement remains valid even if there exist a polynomial count of simulated proofs. Hence, using a simulation-sound scheme for the range NIZK results in an IVTS with the property of soundness.

Theorem 2: (Privacy) Suppose LTLP is a perfect correctness scheme, the proposed IVTS satisfies privacy property as described in Definition 7.

Proof: We demonstrate that the protocol outlined in Section III maintains privacy in the presence of an adversary with a limited depth of $\mathbf{T}^{\varepsilon}$, where $\varepsilon \leq 1$.

Hybrid $\mathcal{H}_0$: This is the original execution of the protocol in Section III.

Hybrid $\mathcal{H}_1$: Hybrid $\mathcal{H}_1$ is the same as Hybrid $\mathcal{H}_0$, with the exception that we adopt a lazy sampling function to simulate a random oracle. Using lazy sampling, it can generate a set I^* ($|I^*| = t-1$) in advance. When the cut-and-choose instance requires the random oracle, it outputs the set I^*. The above changes are only in the syntax, and they do not affect the overall distribution.

Hybrid $\mathcal{H}_2$: Hybrid $\mathcal{H}_2$ is similar to Hybrid $\mathcal{H}_1$, with the exception that a simulated *crs* is employed in the execution. Due to the zero-knowledge property of NIZK, we can consider this modification computationally indistinguishable from Hybrid $\mathcal{H}_1$.

Hybrid $\mathcal{H}_3 \ldots \mathcal{H}_{3+n}$: In the Hybrid $\mathcal{H}_{3+i}$ ($i \in [0, n]$), the proof $\pi_{\mathsf{range},i}$ is generated by the simulator. The zero-knowledge property of NIZK ensures that the distance between adjacent hybrids is related to the security parameter.

Hybrid $\mathcal{H}_{3+n+1} \ldots \mathcal{H}_{3+2n-t+1}$: In the hybrid Hybrid $\mathcal{H}_{3+i}$ ($i \in [0, n-t+1]$), the puzzle corresponding to the i-th element of the complement set of I^*, denoted by $\hat{I}^*$, is generated via $\mathsf{LTLP.PGen}(pp, 0^{\lambda}, r_i)$, where a depth-bounded distinguisher is employed. The indistinguishably relies on the security of time-lock puzzle.

Hybrid $\mathcal{H}_{3+2n-t+2}$: The committer operates as below. Firstly, it select a random $t-1$ degree-polynomial $p(x) = \sum_{i=1}^{t-1} a_i x^i$, where $\{a_i\}_{i \in \{1,..t-1\}}$ are random numbers. For all $i \in [1, n]$, it computes $pk_i = pk \cdot g_0^{p(i)}$. We have that $\prod_{i \in I^*} pk_i^{l_i(0)} \cdot pk_j^{l_j(0)} \neq pk$ as expected. Consequently, the modifications made to this hybrid are purely limited to syntax, and thus the resulting distribution remains identical to that of the prior hybrid.

Simulator $\mathcal{S}$: $\mathcal{S}$ is designed to be equivalent to the previous hybrid, and it should be emphasized that the security analysis does not rely on any information regarding the witness. This brings to the end of our proof.

B. Security Analysis of VATS

Theorem 3: (Soundness) The proposed VTAS satisfies soundness if the underlying adaptor signature has pre-signature adaptability, NIZK has soundness property and TLP has correctness property.

Proof: We assume there exists a $\mathcal{PPT}$ adversary $\mathcal{A}$ that can break the soundness of our proposed VTAS with non-negligible advantage. Specifically, this implies that for a given public key pk, an adversary $\mathcal{A}$ can create a commitment $(m, \widehat{\sigma}, Y, C, \pi)$ such that $\mathsf{AS}^{\mathsf{SIG}}.\mathsf{pVrf}(pk, m, Y, \widehat{\sigma}) = 1$, and the proof (C, π) generated by an incorrect witness can still pass the verification successfully. Due to the correctness property of TLP, the witness y just can be revealed after time $\mathbf{T}$.

Assuming $\mathcal{A}$ can generate a valid NIZK instance (x, π), where $x = (N, g_1, g_2, pk, \widehat{\sigma}, Y)$ and $\pi = (R_1, R_2, R_3, e, z_1, z_2)$. A knowledge extractor M is present that can rewind the prover to the oracle query $e = \mathcal{H}(N, g, pk, \widehat{\sigma}, Y, Z, R_1, R_2, R_3)$. M recreates the random oracle $\mathcal{H}$ where $e^* = \mathcal{H}(N, g_1, g_2, pk, \widehat{\sigma}, Y, Z, R_1, R_2, R_3)$ and continues the committing steps. After that, $\mathcal{A}$ can obtain another valid commitment (x, π_1^*), where $\pi_1 = (R_1, R_2, R_3, e^*, z_1^*, z_2^*)$. We can combine a set of independent equations $z_1 = \alpha - e \cdot r, z_2 = \beta - e \cdot y, z_1^* = \alpha - e^* \cdot r, z_2^* = \beta - e^* \cdot y$, and obtain:

$$r = (z_1 - z_1^*)(e^* - e),\ y = (z_2 - z_2^*)(e^* - e).$$

Based on the analysis above, we can solve the discrete logarithm with a non-negligible probability. Thus, our proposed VTAS scheme meets the soundness requirement.

TABLE I
COMPARISON OF COMPUTATION COST

Scheme	BLS			Schnorr			ECDSA		
	Commit	*Verify*	*FOpen*	*Commit*	*Verify*	*FOpen*	*Commit*	*Verify*	*FOpen*
VTS [6]	$\mathcal{O}(tn)$	$\mathcal{O}(n)$	**T**+$\mathcal{O}(t)$	$\mathcal{O}(tn)$	$\mathcal{O}(n+t)$	**T**+$\mathcal{O}(t)$	$\mathcal{O}(tn)$	$\mathcal{O}(n+t)$	**T**+$\mathcal{O}(t)$
IVTS	$\mathcal{O}(n)$	$\mathcal{O}(n)$	**T**+$\mathcal{O}(t)$	$\mathcal{O}(n)$	$\mathcal{O}(n+t)$	**T**+$\mathcal{O}(t)$	$\mathcal{O}(n)$	$\mathcal{O}(n+t)$	**T**+$\mathcal{O}(t)$
VTAS	N/A	N/A	N/A	$\mathcal{O}(1)$	$\mathcal{O}(1)$	**T**+$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	**T**+$\mathcal{O}(1)$
VTAS$^+$	N/A	N/A	N/A	$\mathcal{O}(k)$	$\mathcal{O}(k)$	**T**+$\mathcal{O}(1)$	$\mathcal{O}(k)$	$\mathcal{O}(k)$	**T**+$\mathcal{O}(1)$

n: The total number of shared signatures. t: The threshold number of required signatures.
T: The signature can only be opened after a specified time. N/A: Not applicable.
k: It is an integer satisfied $\mathbf{T} = 2^k$. $+$: With the property of verifiable recovery.

Theorem 4: (Privacy) The proposed VTAS satisfies privacy if the underlying adaptor signature has pre-signature adaptability, TLP has correctness property and the NIZK has zero-knowledge property.

Proof: We demonstrate that the protocol outlined in Section IV-C maintains privacy in the presence of an adversary with a limited depth of $\mathbf{T}^{\varepsilon}$, where $\varepsilon \leq 1$.

Hybrid $\mathcal{H}_0$: This hybrid is the original execution of the protocol in Section IV-C.

Hybrid $\mathcal{H}_1$: Hybrid $\mathcal{H}_1$ is similar to Hybrid $\mathcal{H}_0$ except that a simulated random oracle is used instead of a real hash function. The simulator defines the output of the random oracle for any given input.

Hybrid $\mathcal{H}_2$: Hybrid $\mathcal{H}_2$ is the similar to Hybrid $\mathcal{H}_1$ except that a simulated *crs* is employed in the execution. This change is computationally indistinguishable due to the zero-knowledge property of NIZK. The time-lock puzzle is generated via $u = g^r \pmod N$ and $v = h^{r \cdot N} \cdot (1+N)^{0^\lambda} \pmod{N^2}$, where a depth-bounded distinguisher is employed. The indistinguishably relies on the security of TLP.

Hybrid $\mathcal{H}_3$: In the Hybrid $\mathcal{H}_3$, given an instance $(N, g_1, g_2, pk, Y, u, v)$, the simulator randomly chooses $e^*, z_1^*, z_2^* \in \mathbb{Z}_q^*$ and calculates $R_1^* = g^{z_1^*} \cdot u^{e^*}$, $R_2^* = v^{e^*} \cdot h^{z_1^* \cdot N}(1+N)^{z_2^*}$, $R_3^* = g^{z_2^*} Y^{e^*}$. Finally, the simulator sets $\mathcal{O}_{\mathcal{H}}(N, g_1, g_2, pk, \widehat{\sigma}, Y, Z, R_1^*, R_2^*, R_3^*) = e^*$. Consequently, the distribution of the hybrid remains unchanged because the outputs of the random oracle are random and uniformly distributed.

Simulator $\mathcal{S}$: $\mathcal{S}$ is designed to be equivalent to the previous hybrid, and does not rely on any information regarding the witness. This brings to the end of our proof.

VII. PERFORMANCE EVALUATION

This section provides a performance analysis considering two perspectives: theoretical complexity and experimental performance. The computational complexity and space complexity are analyzed to determine the theoretical complexity. The experimental performance is evaluated by comparing the time cost and communication overhead of our proposed schemes with VTS [6].

A. Theoretical Complexity

To assess the performance of our proposed solutions, namely IVTS, VTAS, and VTAS$^+$ (an extension of VTAS), we conducted a comparative analysis of their computational and communication complexities with respect to the relevant solution proposed in [6].

In terms of computational complexity (as shown in Table I), VTS outperforms the other three schemes, mainly due to the efficiency of pre-signature verification in the adaptor signature. From a functional perspective, VTS, IVTS, and VTAS are not able to achieve the verifiable recovery property. IVTS outperforms VTS$^+$ in terms of BLS, Schnorr, and ECDSA, by utilizing lightweight secret sharing technology in the commitment and proof phases. Furthermore, although the computational overhead of VTAS$^+$ is slightly inferior to that of VTAS, the former can provide the verifiable recovery property, which ensures that force opening will return a valid signature after targeted time. However, both schemes are currently incompatible with BLS due to the deterministic nature of BLS construction, which precludes transformation into adaptor signature. We plan to explore this direction in our future research endeavors.

Regarding the computational complexity, as presented in Table II, both VTS and IVTS schemes exhibit a linear communication overhead of $\mathcal{O}(n)$, where n represents the number of signature shares. In contrast, our VTAS scheme significantly reduces the communication overhead to a constant $\mathcal{O}(1)$ regardless of the shares, leading to consistent and minimal communication costs. Furthermore, VTAS$^+$ introduces an additional layer of security by incorporating a variable communication overhead of $\mathcal{O}(k)$, which is directly proportional to the lock duration chosen for the payment channel.

Therefore, our scheme has comparable computational complexity and communication complexity to the existing solutions. The details of empirical time and communication consumption will be presented in the following section.

B. Experimental Performance

We further evaluate the performance of VTS, IVTS, VTAS and VTAS$^+$ in the practical environment. Our simulation was executed on a personal laptop equipped with 64-bit

TABLE II
COMPARISON OF COMMUNICATION OVERHEAD

Scheme	BLS	Schnorr	ECDSA
VTS [6]	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
IVTS	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
VTAS	N/A	$\mathcal{O}(1)$	$\mathcal{O}(1)$
VTAS$^+$	N/A	$\mathcal{O}(k)$	$\mathcal{O}(k)$

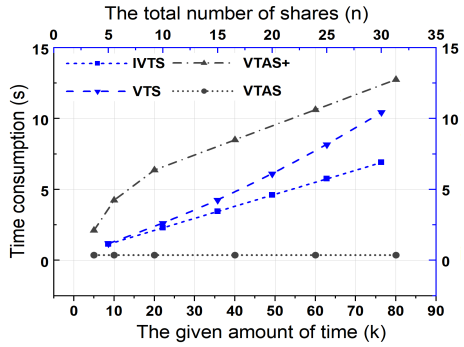

Fig. 7. Computational costs of *FOpen* (omitted **T**).

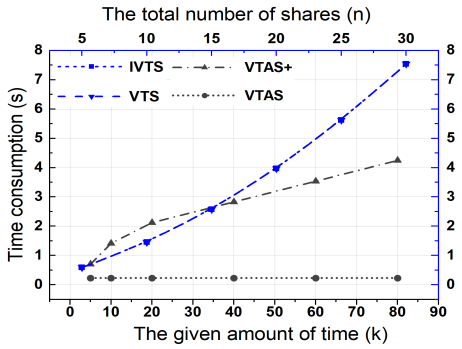

Fig. 8. Computational costs of *Verification*.

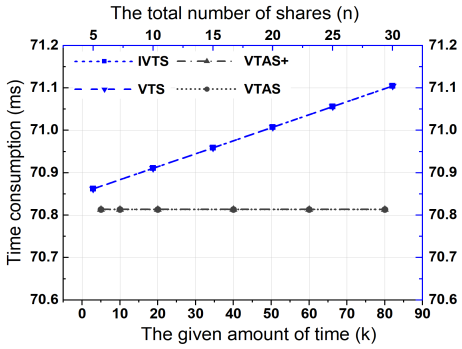

Fig. 9. Computational costs of *FOpen* (omitted **T**).

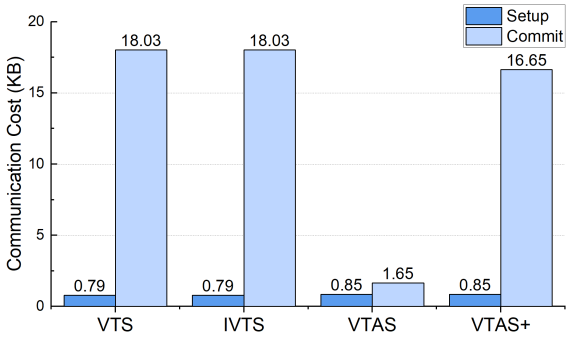

Fig. 10. Communication overhead ($n = 20, k = 20$).

Windows 10 and 3.40 GHz Intel(R) Core(TM) i7 CPU. The implementation was based on the cyclic group and quadratic reside group via Miracle library. The length of q and N was set as 256 bit and 1024 bit, respectively. We simulated the instances of Schnorr-based signature and evaluated the time cost of *Commit*, *Verify* and *FOpen* algorithms. We performed 1000 iterations of each operation and computed the average time taken.

Based on the above implementation, we present the empirical outcomes of computational expenses. We use a line graph with dual axes to illustrate the time cost over variables n and k. The x-axis had two parallel axes representing the given amount of time $\mathbf{T}$ ($\mathbf{T} = 2^k$) and the total number of shares n, while the y-axis had two perpendicular axes representing the time cost in second or millisecond, respectively. The experimental performance of three algorithms is shown in Fig. 7, Fig. 8 and Fig. 9. These figures demonstrate that VTAS maintains a constant-size computational time in each algorithm, rather than linearly increasing time with the maximum number of shares or time. Furthermore, VTAS exhibits the best performance in both commitment and verification. On the other hand, while VTS, IVTS, and VTAS$^+$ demonstrate a linear growth trend, IVTS outperforms VTS in the commitment step due to the efficient secret sharing construction in VTS. Although VTAS$^+$ exhibits the highest performance for the committer, its total time overhead can be reduced through parallel computing.

In addition, we have evaluated the communication overhead of our schemes as well as VTS, which is shown in Fig. 10. The costs associated with initialization transmission and storage for all schemes are within 1 KB. The communication costs for the *Commit* operation are 18.03 KB, 18.03 KB, 1.65 KB, and 16.65 KB for VTS, IVTS, VTAS, and VTAS$^+$, respectively.

Therefore, our proposed schemes exhibit superior performance and security property when compared to the existing scheme. Based on actual demands of applications, the appropriate scheme can be selected for practical application scenarios.

VIII. RELATED WORK

In this section, based on the technologies behind blockchain scalable payment, we review the literature related to our proposed work, in terms of payment channel networks and verifiable timed signatures.

A. Payment Channel Networks

PC and PCN have been proposed as solutions to cryptocurrencies scalabiltiy problems. Their approach is based on the Hash Time-Lock Contract (HTLC) [4], which enables a participant to receive payment if they provide a pre-image of a specific hash value prior to a specified time. Extensive literature exists proposing constructions for payment channels [11], [19], [20]. While TumbleBit [21] and Bolt [12] support off-chain payments and provide payment anonymity,

the privacy protection they offer is limited to single-hop payments, and extending them to support multi-hop payments remains an open challenge. Other existing PCN schemes are typically hindered to settings with TEE [22] or an underling script language [23]. Consequently, AMHL [5] and lockable signatures [24] were proposed to deal with the dependency on the scripts, also referred to as scriptless script. While both of them can provide a certain level of on-chain privacy, in terms of atomic lock, they cannot achieve strong unlinkability of payment path limited by the transparent time-related information.

B. Verifiable Timed Signatures

Timed cryptography encompasses a range of cryptographic primitives that enable senders to transmit messages to the future. After a predefined time interval, the information can be disclosed or validated, potentially at the conclusion of a sequential computation. This class includes prominent tools such as Time-Lock puzzles [17], [25], [26], Timed commitment [18], and Timed signatures [27]. The VTS scheme can be considered as a version of verifiably encrypted signatures that incorporates timing information [28], [29]. One major benefit of VTS is that the signature can be obtained without requiring a trusted third party. The fundamental concept behind VTS [6] involves the committer computing a t-out-of-n secret sharing for a special signature. Through the cut-and-choose technique, a subset of $t-1$ shares are revealed. The verifier checks the correctness of the opened shares as well as the puzzles (using the prover's random coins). Upon successful verification, the verifier can ensure that a legitimate share for reconstructing a valid signature exists in at least one of the unopened puzzles. This structure has spawned many different time cryptography constructions, such as VTLRS [8] and VTD [7]. However, they all face two main challenges, linear-increasing performance and time unverifiability as aforementioned. In this paper, we improve the efficiency of VTS scheme proposed by [6], and construct a high-performance and security-enhanced scheme.

IX. Conclusion

In this work, we explored challenges of scalable payment for scriptless blockchains and presented three efficient verifiable timed (adaptor) signature schemes. Our first key contribution was a generic transformation approach to enhance the performance of the existing VTS and other timed cryptography schemes. To balance security and efficiency, we further focused on adaptor signatures and introduced the new cryptographic primitives called VTAS and VTAS$^+$, which supports additional property of verifiable recovery. In addition, we utilized it to design a unidirectional ASTNet enabling multi-hop payments on scriptless blockchains. The security analysis presented that our proposals satisfy the necessary security requirements. Furthermore, we conducted experimental comparisons with the existing state-of-art schemes and demonstrated a trade-off between efficiency and functionality. In our future research, we intend to explore the bidirectional payment channel and conduct an in-depth evaluation of the constructed payment channel network, considering blockchain networks like Ethereum and Hyperledger Fabric.

Appendix

A. Improved Verifiable Timed Schnorr Signature

The Schnorr signature [30] executes on a cyclic group $\mathbb{G}$ of prime order q, where g serves as the generator, and utilizes a hash function $\mathcal{H}: \{0,1\}^* \to \mathbb{Z}_q^*$.

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$: Pick $sk \in \mathbb{Z}_q^*$ randomly, calculate $pk = g^{sk}$, and return (sk, pk).
- $\sigma \leftarrow \mathsf{Sign}(sk, m)$: Pick $r \in \mathbb{Z}_q^*$ randomly to calculate $R = g^r, h = \mathcal{H}(R, m), s = r + h \cdot sk \pmod q$ and return $\sigma = (R, s)$.
- $\perp / 1 \leftarrow \mathsf{Verify}(\sigma, m, pk)$: Compute $h = \mathcal{H}(m, R)$ and check that if $g^s = R \cdot pk^h$ output 1, otherwise output $\perp$.

The improved verifiable timed Schnorr signature is executed as follows.

Setup: Same as Section III.

Commit: With the input $(pp, \sigma = (R, s), pk, \mathbf{T})$, the commit algorithm executes as follows.

- Pick two random $t-1$ degree-polynomials $p_1(x) = \sum_{i=1}^{t-1} a_i x^i + s$ and $p_2(x) = \sum_{i=1}^{t-1} b_i x^i$, where $\{a_i, b_i\}_{i \in \{1,..t-1\}}$ are random numbers.
- Compute $h = \mathcal{H}(R, m)$. For all $i \in [1, n]$, select $r_i \leftarrow \{0,1\}^\lambda$ and compute

$$s_i = p_1(i), pk_i = pk \cdot g^{p_2(i)}, R_i = g^{s_i} pk_i^{-h},$$
$$\pi_{\mathsf{range},i} = \mathsf{ZK.proof}(crs_{\mathsf{range}}, (Z_i, 0, 2^\lambda, \mathbf{T}), (s_i, r_i)).$$

  where s_i, pk_i are t-out-of-n Shamir secret shares, respectively.
- For all $i \in [1, n]$, generate time-lock puzzles of s_i, $Z_i = \mathsf{LTLP.PGen}(pp, s_i, \mathbf{T})$.
- Invoke hash function $I = \mathcal{H}^*(pk, m, R, (pk_1, R_1, Z_1, \pi_{\mathsf{range},1}), \ldots, (pk_n, R_n, Z_n, \pi_{\mathsf{range},n}))$
- Return the commitment $C = (Z_1, \ldots, Z_n, \mathbf{T})$ and the proof $\pi = (R, I, \{s_i\}_{i \in I}, \{pk_i, R_i, \pi_{\mathsf{range},i}\}_{i \in \{1,..,n\}})$.

Verification: With the input (pp, m, pk, C, π), it executes as follows.

- If any of the following condition is satisfied output $\perp$, else output 1:
  1) $\exists j \notin I$ s.t. $\prod_{i \in I} pk_i^{l_i(0)} \cdot pk_j^{l_j(0)} \neq pk$ or $\prod_{i \in I} R_i^{l_i(0)} \cdot R_j^{l_j(0)} \neq R$, where $l_i(\cdot)$ represents the Lagrange polynomial basis for the i-th share.
  2) $\exists i \in [1, n]$ s.t. $\mathsf{ZK.verify}(crs_{\mathsf{range}}, (Z_i, 0, 2^\lambda, \mathbf{T}), \pi_{\mathsf{range},i}) \neq 1$.
  3) $\exists i \in I$ s.t. $Z_i \neq \mathsf{LTLP.PGen}(pp, s_i)$ or $g^{s_i} \neq R_i \cdot pk_i^h$.
  4) $I \neq \mathcal{H}^*(pk, m, R, (pk_1, R_1, Z_1, \pi_{\mathsf{range},1}), \ldots, (pk_n, R_n, Z_n, \pi_{\mathsf{range},n}))$.

Open: It outputs the complete signature $\sigma = (R, s)$.

Force Open: With the input (pp, m, pk, C, π), the force open algorithm executes as follows.

- Pick $v \in \{1, \ldots, n\}/I$ randomly and compute $s_v \leftarrow \mathsf{LTLP.PSolve}(pp, Z_v)$.
- Recover the secret $s = \sum_{i \in \{I, v\}} l_i(0) s_i \pmod q$, and output $\sigma = (R, s)$.

B. Improved Verifiable Timed ECDSA Signature

The ECDSA signature [31] is based on a group $\mathbb{G}$ of prime order q, where g serving as the base point or generator of the group. Let a hash function as $\mathcal{H}: \{0,1\}^* \to \mathbb{Z}_q^*$.

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$: Pick $sk \in \mathbb{Z}_q^*$ randomly, calculate $pk = g^{sk}$, and return (sk, pk).
- $\sigma \leftarrow \mathsf{Sign}(sk, m)$: Compute $h = \mathcal{H}(m)$ and pick $k \in \mathbb{Z}_q^*$ randomly. Let (K_x, K_y) represent the point $K = g^k$, then set $r = K_x \pmod q$ and compute $s = (h + r \cdot sk)k^{-1} \pmod q$. Return $\sigma = (r, s)$.
- $0/1 \leftarrow \mathsf{Verify}(\sigma, m, pk)$: Compute $h = \mathcal{H}(m)$, $(x, y) = (g^h \cdot pk^r)^{s^{-1}}$, and check that if $r = x \pmod q$ return 1, else output $\perp$.

The improved verifiable timed ECDSA signature is executed as follows.

Setup: Same as Section III.

Commit: With the input $(pp, \sigma = (r, s), pk, \mathbf{T})$, the commit algorithm executes as follows.

- Select a random $t-1$ degree-polynomial $p(x) = \sum_{i=1}^{t-1} a_i x^i + s$, where $\{a_i\}_{i\in\{1,..t-1\}}$ are random numbers.
- Compute $h = \mathcal{H}(m)$ and $R = (g^h \cdot pk^r)^{s^{-1}}$.
- For all $i \in [1, n]$, select $r_i \leftarrow \{0,1\}^\lambda$ and compute

$$s_i = p_1(i),\ R_i = (g^h \cdot pk^r)^{s_i^{-1}},$$
$$\pi_{\mathsf{range},i} = \mathsf{ZK.proof}(crs_{\mathsf{range}}, (Z_i, 0, 2^\lambda, \mathbf{T}), (s_i, r_i)).$$

  where s_i, R_i are t-out-of-n Shamir secret shares, respectively.
- For all $i \in [1, n]$, generate time-lock puzzles with s_i. Compute $Z_i = \mathsf{LTLP.PGen}(pp, s_i, \mathbf{T})$.
- Invoke hash function $I = \mathcal{H}^*(pk, m, r, R, (R_1, Z_1, \pi_{\mathsf{range},1}), \ldots, (R_n, Z_n, \pi_{\mathsf{range},n}))$.
- Return the commitment $C = (Z_1, \ldots, Z_n, \mathbf{T})$ and the proof $\pi = (r, R, I, \{s_i\}_{i\in I}, \{R_i, Z_i, \pi_{\mathsf{range},i}\}_{i\in\{1,..,n\}})$.

Verification: With the input (pp, m, pk, C, π), it executes as follows.

- Compute $h = \mathcal{H}(m)$.
- If any of the following criteria are met, output $\perp$. Otherwise, output 1.
  1) $\exists j \notin I$ s.t. $\prod_{i\in I} R_i^{l_i(0)} \cdot R_j^{l_j(0)} \neq R$, where $l_i(\cdot)$ represents the Lagrange polynomial basis for the i-th share.
  2) $\exists i \in [1, n]$ s.t. $\mathsf{ZK.verify}(crs_{\mathsf{range}}, (Z_i, 0, 2^\lambda, \mathbf{T}), \pi_{\mathsf{range},i}) \neq 1$.
  3) $\exists i \in I$ s.t. $Z_i \neq \mathsf{LTLP.PGen}(pp, s_i)$, or $R_i \neq (g^h \cdot pk^r)^{s_i^{-1}}$.
  4) $I = \mathcal{H}^*(pk, m, r, R, (R_1, Z_1, \pi_{\mathsf{range},1}), \ldots, (R_n, Z_n, \pi_{\mathsf{range},n}))$.

Open: It outputs the complete signature $\sigma = (r, s)$.

Force Open: With the input (pp, m, pk, C, π), the force open algorithm executes as follows.

- Pick $v \in \{1, \ldots, n\}/I$ randomly and compute $s_v \leftarrow \mathsf{LTLP.PSolve}(pp, Z_v)$.
- Recover the secret $s = \sum_{i\in\{I,v\}} l_i(0)s_i \pmod q$, and output $\sigma = (r, s)$.

REFERENCES

[1] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *Proc. 41st Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2018, pp. 1545–1550.

[2] R. Krishna et al., "Visual genome: Connecting language and vision using crowdsourced dense image annotations," 2016, *arXiv:1602.07332*.

[3] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 455–471.

[4] M. Herlihy, "Atomic cross-chain swaps," in *Proc. ACM Symp. Princ. Distrib. Comput.*, Jul. 2018, pp. 245–254.

[5] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–12.

[6] 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.

[7] S. A. Thyagarajan, G. Malavolta, and P. Moreno-Sanchez, "Universal atomic swaps: Secure exchange of coins across all blockchains," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 1299–1316.

[8] S. A. K. Thyagarajan, G. Malavolta, F. Schmid, and D. Schröder, "Verifiable timed linkable ring signatures for scalable payments for Monero," in *Proc. 27th Eur. Symp. Res. Comput. Secur.*, in Lecture Notes in Computer Science, vol. 13555. Cham, Switzerland: Springer, 2022, pp. 467–486.

[9] J. C. Benaloh and J. Leichter, "Generalized secret sharing and monotone functions," in *Proc. 8th Annu. Int. Cryptol. Conf.*, in Lecture Notes in Computer Science, vol. 403. Cham, Switzerland: Springer, 1988, pp. 27–35.

[10] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology—CRYPTO 1986* (Lecture Notes in Computer Science), vol. 263. Heidelberg, Germany: Springer, 1986, pp. 186–194.

[11] C. Decker and R. Wattenhofer, "A fast and scalable payment network with Bitcoin duplex micropayment channels," in *Proc. 17th Int. Symp. Stabilization, Saf., Secur. Distrib. Syst. (SSS)*, in Lecture Notes in Computer Science, vol. 9212. Cham, Switzerland: Springer, 2015, pp. 3–18.

[12] M. Green and I. Miers, "BOLT: Anonymous payment channels for decentralized currencies," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 473–489.

[13] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *J. Cryptol.*, vol. 17, no. 4, pp. 297–319, Sep. 2004.

[14] L. Aumayr et al., "Generalized channels from limited blockchain scripts and adaptor signatures," *Cryptol. ePrint Arch.*, vol. 2020, p. 476, Jan. 2020. [Online]. Available: https://eprint.iacr.org/2020/476

[15] A. Erwig, S. Faust, K. Hostáková, M. Maitra, and S. Riahi, "Two-party adaptor signatures from identification schemes," in *Proc. 24th IACR Int. Conf. Pract. Theory Public Key Cryptogr.*, in Lecture Notes in Computer Science, vol. 12710. Cham, Switzerland: Springer, 2021, pp. 451–480.

[16] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *Proc. 20th Annu. ACM Symp. Theory Comput.*, 1988, pp. 103–112.

[17] G. Malavolta and S. A. K. Thyagarajan, "Homomorphic time-lock puzzles and applications," in *Proc. 39th Annu. Int. Cryptol. Conf.*, in Lecture Notes in Computer Science, vol. 11692. Cham, Switzerland: Springer, 2019, pp. 620–649.

[18] D. Boneh and M. Naor, "Timed commitments," in *Proc. 20th Annu. Int. Cryptol. Conf.*, in Lecture Notes in Computer Science, vol. 1880. Cham, Switzerland: Springer, 2000, pp. 236–254.

[19] S. Lee and H. Kim, "On the robustness of lightning network in bitcoin," *Pervas. Mobile Comput.*, vol. 61, Jan. 2020, Art. no. 101108.

[20] A. Kurt, E. Erdin, M. Cebe, K. Akkaya, and A. S. Uluagac, "LNBot: A covert hybrid botnet on Bitcoin lightning network for fun and profit," in *Proc. 25th Eur. Symp. Res. Comput. Secur.*, vol. 12309. Cham, Switzerland: Springer, 2020, pp. 734–755.

[21] E. Heilman, L. AlShenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "TumbleBit: An untrusted Bitcoin-compatible anonymous payment hub," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2017, pp. 1–10.

[22] J. Lind, O. Naor, I. Eyal, F. Kelbert, P. Pietzuch, and E. G. Sirer, "TeeChain: Reducing storage costs on the blockchain with offline payment channels," in *Proc. 11th ACM Int. Syst. Storage Conf.*, Jun. 2018, p. 125.

[23] S. Dziembowski, S. Faust, and K. Hostáková, "General state channel networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 949–966.
[24] S. A. Krishnan Thyagarajan and G. Malavolta, "Lockable signatures for blockchains: Scriptless scripts for all signatures," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2021, pp. 937–954.
[25] C. Baum, B. David, R. Dowsley, J. B. Nielsen, and S. Oechsner, "TARDIS: A foundation of time-lock puzzles in UC," in *Proc. 40th Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, in Lecture Notes in Computer Science, vol. 12698. Cham, Switzerland: Springer, 2021, pp. 429–459.
[26] 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)*, Oct. 2017, pp. 576–587.
[27] J. A. Garay and M. Jakobsson, "Timed release of standard digital signatures," in *Proc. 6th Int. Conf. FC*, in Lecture Notes in Computer Science, vol. 2357. Cham, Switzerland: Springer, 2002, pp. 168–182.
[28] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Advances in Cryptology—EUROCRYPT 2003* (Lecture Notes in Computer Science), vol. 2656, E. Biham, Ed. Heidelberg, Germany: Springer, 2003, pp. 416–432.
[29] C. Hanser, M. Rabkin, and D. Schröder, "Verifiably encrypted signatures: Security revisited and a new construction," in *Proc. 20th Eur. Symp. Res. Comput. Secur.*, in Lecture Notes in Computer Science, vol. 9326. Cham, Switzerland: Springer, 2015, pp. 146–164.
[30] C. P. Schnorr, "Efficient identification and signatures for smart cards," in *Advances in Cryptology—CRYPTO 1990* (Lecture Notes in Computer Science), vol. 435. Berlin, Germany: Springer-Verlag, 1990, pp. 239–252.
[31] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, Aug. 2001.

Xiaotong Zhou received the bachelor's and master's degrees in information security from Wuhan University in 2016 and 2019, respectively, where she is currently pursuing the Ph.D. degree with the School of Cyber Science and Engineering. Her research interests include security and privacy, including privacy protection and blockchain security.

Debiao He (Member, IEEE) received the Ph.D. degree in applied mathematics from the School of Mathematics and Statistics, Wuhan University, in 2009. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. He has authored or coauthored more than 100 research papers in refereed international journals and conferences, such as IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, AND USENIX SECURITY SYMPOSIUM. His main research interests include cryptography and information security, in particular, cryptographic protocols. He is an Editorial Board of several international journals, such as ACM Distributed Ledger Technologies: Research and Practice, Frontiers of Computer Science, and IEEE TRANSACTIONS ON COMPUTERS.

Jianting Ning (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, in 2016. He was a Research Scientist with the School of Computing and Information Systems, Singapore Management University, and a Research Fellow with the Department of Computer Science, National University of Singapore. He is currently a Professor with the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, China. He has published papers in major conferences/journals, such as ACM CCS, NDSS, ASIACRYPT, ESORICS, ACSAC, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, and IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING. His research interests include applied cryptography and information security.

Min Luo received the Ph.D. degree in computer science from Wuhan University in 2003. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. He has published papers in international conferences/journals, such as S&P, *ACM TRETS*, IEEE SYSTEMS JOURNAL, and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. His research interests include applied cryptography and blockchain technology.

Xinyi Huang (Member, IEEE) received the Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently an Associate Professor with the Thrust of Artificial Intelligence, Information Hub, The Hong Kong University of Science and Technology (Guangzhou), China. His work has been cited more than 10000 times at Google Scholar. He has published over 160 research papers in refereed international conferences and journals, such as ACM CCS, Crypto, Asiacrypt, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and IEEE TRANSACTIONS ON INFORMATION SECURITY AND FORENSICS. His research interests include cryptography and information security. He is on the editorial board of *International Journal of Information Security* and *Science China Information Sciences*. He has served as the program/general chair or a program committee member for over 120 international conferences.