information

Article

A Traceable Universal Designated Verifier Transitive Signature Scheme

Shaonan Hou [1], Chengjun Lin [1,]* and Shaojun Yang [1,2]

[1] School of Mathematics and Statistics, Fujian Normal University, Fuzhou 350117, China; hsn1027@outlook.com (S.H.); shaojunyang@fjnu.edu.cn (S.Y.)
[2] Key Laboratory of Analytical Mathematics and Applications, Fujian Normal University, Ministry of Education, Fuzhou 350117, China
* Correspondence: linchengjun@fjnu.edu.cn

Abstract: A transitive signature scheme enables anyone to obtain the signature on edge (i, k) by combining the signatures on edges (i, j) and (j, k), but it suffers from signature theft and signature abuse. The existing work has solved these problems using a universal designated verifier transitive signature (UDVTS). However, the UDVTS scheme only enables the designated verifier to authenticate signatures, which provides a simple way for the signer to deny having signed some messages. The fact that the UDVTS is not publicly verifiable prevents the verifier from seeking help arbitrating the source of signatures. Based on this problem, this paper proposes a traceable universal designated verifier transitive signature (TUDVTS) and its security model. We introduce a tracer into the system who will trace the signature back to its true source after the verifier has submitted an application for arbitration. To show the feasibility of our primitive, we construct a concrete scheme from a bilinear group pair $(\mathbb{G}, \mathbb{G}_{\mathbb{T}})$ of prime order and prove that the scheme satisfies unforgeability, privacy, and traceability.

Keywords: transitive signature; universal designated verifier; traceability

Citation: Hou, S.; Lin, C.; Yang, S. A Traceable Universal Designated Verifier Transitive Signature Scheme. *Information* **2024**, *15*, 43. https://doi.org/10.3390/info15010043

Academic Editor: Marco Baldi

Received: 8 December 2023
Revised: 4 January 2024
Accepted: 9 January 2024
Published: 12 January 2024

1. Introduction

In today's information age, network information security is a hot topic all over the world. As a cryptographic technique to provide authentication services for electronic data, digital signatures allow a signer who has generated a public/private key pair to sign a message, and any other entity who knows the public key can verify the integrity and the source of data.

However, using traditional digital signatures to authenticate graph-based big data with chain relationships can result in significant communication costs. Given an administrative domain that involves four nodes, as depicted in Figure 1, A and B both belong to the same administrative domain if the edge (A, B) is authentic. If both the edge (A, B) and the edge (B, C) are authentic, then A and C are in the same administrative domain. We need to provide two signatures (σ_{AB} and σ_{BC}) when disclosing the relationship between A and C to others using traditional digital signatures. Such an authentication method is very costly with big data. In 2002, Micali and Rivest [1] proposed the concept of a transitive signature, where anyone can compute a valid signature of the edge (A, C), from the signatures of two edges (A, B) and (B, C).

This method greatly reduces communication costs, but it creates the problem that transitive signatures may be abused, i.e., the signer cannot control who verifies the data, as the signature is publicly verifiable, allowing any entity to verify its validity.

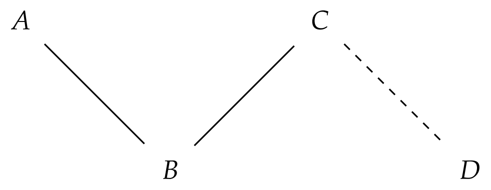

Figure 1. An administrative domain. (Solid lines represent actual edges in the administrative domain, while dotted lines represent non-existent edges).

To remove the public verifiability of transitive signatures, Hou et al. [2] introduced a universal designated verifier transitive signature (UDVTS), where only the designated verifier authenticates signatures. The verifier is designated by the combiner who combines transitive signatures. However, disputes may arise regarding the source of a signature. There are two cases: (1) the signer denies the signature generated by himself, e.g., we assume the edge (A, B) is signed by a signer in Figure 1. B can convince the verifier (not in the same administrative domain as B) that A and C are in the same administrative domain. The signer denies that he had signed the edge (A, B) when the private information was leaked. (2) The signer did not sign an edge (C, D), but the verifier framed the signer by using a simulated signature to make it appear as if the signer had signed the edge. This is because the non-transferability of the designated verifier signature requires the verifier to generate a signature that is indistinguishable from a designated verifier signature. It is also clear that, in the dispute, no arbitration service whatsoever is provided.

The above issues arise from the lack of public verifiability in the UDVTS. Arbitration service is, therefore, a closely watched issue in designated verifier signatures.

1.1. Our Contributions

While transitive signature disputes are easily resolved, the public verifiability also facilitates verifiers leaking signatures to a third party. Furthermore, UDVTS implementation sacrifices the public verifiability of transitive signatures to control the verifier (only the designated verifier can authenticate signatures). To achieve both the goal of controlling the verifier and arbitrability, this paper introduces the concept of a traceable universal designated verifier transitive signature (TUDVTS). Meanwhile, in order to avoid creating a huge arbitration institution, this article mandates a single individual with tracking and arbitration abilities, known as the tracer. The tracer can determine whether the disputed signature originated from the signer.

The specific contributions of this paper are as follows:

(1) Our goal is to substitute the public verifiability of transitive signatures with single entity verifiability, where the entity is trusted. Therefore, we adopt the UDVTS proposed by [2]. Considering that the common issue of designated verifier signatures is that arbitration service cannot be provided when there is a signature dispute, we have introduced a tracer into the system that can find the true source of the signature.

(2) We propose the concept of TUDVTS. We introduce to the UDVTS a tracer who is responsible for arbitrating any disputed data, and the chain relationships in the graphic remain hidden from other entities except the tracer. Therefore, this scheme protects the rights and interests of the signer and the verifier.

(3) We describe definitions of TUDVTS and its security model. The security requirements of TUDVTS include unforgeability, privacy, and traceability. The unforgeability of TUDVTS means that any adversary cannot forge a transitive signature or a designated verifier signature even if it is allowed to obtain signatures on many other messages and public keys of its choice. The TUDVTS scheme encompasses two distinct forms of privacy: the non-transferability of TUDVTS and the privacy of transitive signatures. The former means that there is no way for anyone to distinguish between the two designated verifier signatures produced by the combiner or the verifier. The latter means that there is no way for anyone to distinguish between the two transitive

signatures produced by the signer or the combiner. The traceability implies that the tracer can determine whether the signer signed the message.

(4) We construct a TUDVTS scheme. By incorporating traceability, our scheme not only achieves the goal of controlling the verifier but also provides an arbitration service when signature disputes arise. We have conducted proofs to establish that our scheme satisfies the unforgeability, privacy, and traceability.

1.2. Related Works

We recall transitive signatures and universally designated verifier signatures in this section.

Transitive Signatures (TS). The earliest transitive signature schemes, DLTS and RSATS-1, were proposed by Micali and Rivest [1], where their security relies on the discrete logarithm problem and the RSA assumption, respectively. Note that the former can resist adaptive chosen-message attacks, while the latter can only resist non-adaptive chosen-message attacks. In the same year, Bellare and Neven [3] proposed the "node certification paradigm" and constructed two schemes based on RSA assumption and factoring, respectively. In addition, they proposed other new schemes based on the one-more discrete logarithm problem and one-more gap Diffie-Hellman problem [4]. By employing braid groups, Wang et al. [5] designed a transitive signature scheme that was not susceptible to quantum attacks at the time. Previous schemes required special hash functions, which made them less efficient. Lin et al. [6] introduced a scheme that utilizes general hash functions to achieve improved efficiency by reducing computational time.

Hou et al. [2] proposed the UDVTS scheme to solve the problem of transitive signatures being stolen or abused; only the designated verifier has the ability to authenticate the signature. To serve more scenarios, Zhu et al. [7] proposed a transitive signature scheme with multiple verifiers designated and named it UDMVTS. Lin et al. [8] proposed a more efficient UDVTS scheme based on RSA assumption. Geontae et al. [9] designed the first lattice-based transitive signature scheme, which has the advantage of being resistant to quantum attacks. Then, Geontae et al. [10] designed the first identity-based transitive signature scheme.

All the above schemes are only applicable to undirected graphs. Currently, there are no proposed transitive signature schemes for general directed graphs. In fact, Hohenberger [11] shared that it is difficult to construct a general transitive signature scheme for directed graphs because it requires a special Abel TGII group, and there is no construction of such a group yet. Kuwakado and Tanaka [12] first proposed a TS scheme for directed trees but did not provide a concrete proof. Yi et al. [13] pointed out that Kuwakado's scheme was not secure and then constructed a directed transitive signature scheme provably secure under the standard model. Neven et al. [14] designed a simpler transitive signature scheme to reduce the signature size. Camacho and Hcvia [15] constructed a scheme using a hash function with a common-prefix proof. Xu et al. [16] constructed a scheme using the RSA accumulator that preserves the composed signature size and protects its path information.

Universal Designated Verifier Signatures (UDVS). Steinfeld et al. [17] introduced the concept of UDVS: only the verifier designated by the signature holder is allowed to authenticate signatures. Steinfeld et al. [17] proposed the first UDVS scheme by utilizing a BLS [18] short signature in the same year. They then combined the standard RSA and Schnor schemes to propose two identity-based UDVS schemes [19]. Ng et al. [20] proposed another scheme with multiple verifiers designated, allowing multiple verifiers to authenticate the signature. Zhang et al. [21] use the model proposed by Steinfeld et al. [17] to design two new identity-based UDVS schemes. The security of the above schemes all rely on the random oracle model. Zhang et al. [22] designed the first UDVS scheme that is provably secure in the standard model. Shahandashti et al. [23] provided a generic construction of UDVS from standard digital signatures. Since then, a multitude of UDVS schemes with distinct features have been put forward, such as multi-signer and multiple designated verifiers UDVS [24], UDVS without delegatability [25], universal designated verifier ring signatures [26], and universal designated multi-verifiers content extraction signatures [27]. To resist quantum attacks, Li et al. [28] constructed the first lattice-based UDVS scheme. Moreover, Tang et al. [29] pointed out that the strong privacy in traditional universal designated verifier signature schemes leads to the problem of unfairness to the verifier and, thus, designed a traceable universal designated verifier signature proof scheme.

2. Preliminaries

This section first gives a description of related symbols and introduces some basic concepts and related knowledge.

2.1. Notations

The notation $a \overset{R}{\leftarrow} A$ means that an element a is randomly sampled from the set A. A $\mathcal{PPT}$ (probabilistic polynomial-time) algorithm means that the algorithm is both probabilistic and runs in polynomial time. We define $\mathcal{O}(\cdot)$ as a random oracle that responds to every unique query with a random response chosen uniformly from its output domain. If a query is repeated, it responds the same way every time that query is submitted. The notation $\Pr[X]$ denotes the probability of event X happening. We equate the notion of "negligible probability" with probabilities smaller than any inverse polynomial in n.

2.2. Graphs

This paper considers an undirected graph $G = (V, E)$, where V is a points set and $E \subseteq V \times V$ is an edges set. $\widetilde{G} = (V, \widetilde{E})$ represents the transitive closure of G. It means that $(i, j) \in \widetilde{E}$ if and only if G contains a path from i to j. $G^* = (V^*, E^*)$ represents the transitive reduction in G. It is the graph with the minimum number of edges that possesses the equivalent transitive closure as G.

2.3. Admissible Bilinear Pairing

Let $\mathbb{G}$, $\mathbb{G}_{\mathbb{T}}$ be two groups of prime order p and let g be a generator of $\mathbb{G}$. An admissible bilinear mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbb{T}}$ has the following properties:

(1) Bilinearity: $e(g^a, g^b) = e(g, g)^{ab}$, for all $a, b \overset{R}{\leftarrow} \mathbb{Z}_p$.
(2) Non-degeneracy: $e(g, g) \neq 1$.
(3) Computability: $e(g_1, g_2)$ is efficiently computable for all $g_1, g_2 \overset{R}{\leftarrow} \mathbb{G}$.

2.4. Complexity Assumptions

Definition 1 (One-more Bilinear Diffie–Hellman (BDH) problem [30])**.** *Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbb{T}}$ be a bilinear mapping, where $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ are groups of prime order p, and g is a generator of $\mathbb{G}$. Let $A = g^a, B = g^b$, where $a, b \overset{R}{\leftarrow} \mathbb{Z}_p$. Given $(e, \mathbb{G}, \mathbb{G}_{\mathbb{T}}, p, g, A, B)$ and the following two oracles:*

(1) The H_1 oracle $\mathcal{O}^{\mathcal{H}_1}$: inputs a point $i \in N$, returns a random point $h_i \in \mathbb{G}$.
(2) The CDH oracle $\mathcal{O}^{\mathcal{CDH}}$: inputs a point $h_i \in \mathbb{G}$, returns a point $(h_i)^a \in \mathbb{G}$.

An adversary is said to have solved the one-more BDH problem if it successfully computes n values of $e(g, h_i)^{ab}$ when the number of $\mathcal{O}^{\mathcal{CDH}}$ has queried strictly less than n.

3. Traceable Universal Designated Verifier Transitive Signature Scheme

The following describes the definitions and security model of the TUDVTS scheme. Hou et al. [2] proposed a universal designated verifier transitive signature (UDVTS). As a special case of UDVTS, the idea of the TUDVTS scheme is to allow the combiner to convert a composed signature into a translated signature before designating a verifier. Translation is performed by using the tracer's public key. After the verification, the tracer can seek the true source of a designated verifier signature by using his own secret key. A TUDVTS scheme consists of ten efficient algorithms as follows:

- $pp \leftarrow \mathbf{Setup}(1^k)$. The initialization algorithm that takes as input the security parameter k, outputs the public parameters pp.
- $(pk_i, sk_i) \leftarrow \mathbf{KGen}(pp)$. The key generation algorithm that takes as input the public parameters pp, outputs all users' public/secret key pairs (pk_i, sk_i).
- $\sigma_{ij} \leftarrow \mathbf{TSign}(i, j, sk_s)$. The transitive signing algorithm that takes as input the signer's secret key sk_s and nodes $i, j \in \mathbb{N}$ and outputs an original signature of edge (i, j) relative to sk_s.
- $\{0, 1\} \leftarrow \mathbf{TVry}(i, j, pk_s, \sigma_{ij})$. The verification algorithm that takes as input the signer's public key pk_s, nodes $i, j \in \mathbb{N}$, and a candidate signature σ_{ij}, which outputs 1 if accepting the signature or 0 for rejecting it.
- $\{\perp, \sigma_{ik}\} \leftarrow \mathbf{Comp}(i, j, k, pk_s, \sigma_{ij}, \sigma_{jk})$. The composition algorithm that takes as input the signer's public key pk_s,nodes $i, j, k \in \mathbb{N}$, and two signatures σ_{ij}, σ_{jk}, which outputs the composed signature σ_{ik} or the symbol $\perp$ to indicate failure.
- $\{\widehat{\sigma}_{ij}, t\} \leftarrow \mathbf{Trans}(pk_t, \sigma_{ij})$. The translation algorithm that takes as input the tracer's public key pk_t and a transitive signature σ_{ij} of edge (i, j) and outputs a translated signature $\widehat{\sigma}_{ij}$. In addition, the combiner selects and saves a secret value t.
- $\sigma_{DV} \leftarrow \mathbf{DS}(i, j, pk_v, \widehat{\sigma}_{ij}, t)$. The signature holder's designation algorithm that takes as input the verifier's public key pk_v, nodes $i, j \in \mathbb{N}$, a secret value t, and a translated signature $\widehat{\sigma}_{ij}$, which outputs a designated verifier signature σ_{DV}.
- $\widehat{\sigma}_{DV} \leftarrow \mathbf{Sim}(i, j, pk_s, pk_t, sk_v, \widehat{\sigma}_{ij})$. The transcript simulation algorithm that takes as input the signer's public key pk_s, the tracer's public key pk_t, the verifier's secret key sk_v, nodes $i, j \in \mathbb{N}$, and the translated signature $\widehat{\sigma}_{ij}$, which outputs a simulated signature $\widehat{\sigma}_{DV}$.
- $\{0, 1\} \leftarrow \mathbf{DV}(i, j, pk_s, sk_v, \sigma_{DV})$. The designated verifying algorithm that takes as input the signer's public key pk_s, the verifier's secret key sk_v, nodes $i, j \in \mathbb{N}$, and a designated verifier signature σ_{DV}, which outputs 1 if accepting the signature or 0 for rejecting it.
- $\sigma_{ij} \leftarrow \mathbf{Trace}(sk_t, \widehat{\sigma}_{ij})$. The tracing algorithm that takes as input the tracer's secret key sk_t and the translated signature $\widehat{\sigma}_{ij}$, which outputs the transitive signature σ_{ij}.

Correctness: we require five obvious correctness properties in TUDVTS. The first four points are the correctness requirements of UDVTS. Algorithm **TVry** checks the correctness of **TSign** and **Comp**. Algorithm **DV** checks the correctness of **DS** and **Sim**. Algorithm **Trans** checks the correctness of **Trace**.

- Correctness of **TSign**: If $\sigma_{ij} \leftarrow \mathbf{TSign}(i, j, sk_s)$, then
$$\Pr[1 \leftarrow \mathbf{TVry}(i, j, pk_s, \sigma_{ij})] = 1.$$
- Correctness of **Comp**: If $\sigma_{ik} \leftarrow \mathbf{Comp}(i, j, k, pk_s, \sigma_{ij}, \sigma_{jk})$, then
$$\Pr[1 \leftarrow \mathbf{TVry}(i, k, pk_s, \sigma_{ik})] = 1,$$
where σ_{ij}, σ_{jk} are legitimate signatures (the signature is either obtained by the signer or by running **Comp** on legitimate signatures).
- Correctness of **DS**: If $\sigma_{DV} \leftarrow \mathbf{DS}(i, j, pk_v, \widehat{\sigma}_{ij})$, then
$$\Pr[1 \leftarrow \mathbf{DV}(i, j, pk_s, sk_v, \sigma_{DV})] = 1.$$
- Correctness of **Sim**: If $\sigma_{DV} \leftarrow \mathbf{Sim}(i, j, pk_s, pk_t, sk_v, \widehat{\sigma}_{ij})$, then
$$\Pr[1 \leftarrow \mathbf{DV}(i, j, pk_s, sk_v, \sigma_{DV})] = 1.$$
- Correctness of **Trace**: If $\widehat{\sigma}_{ij} \leftarrow \mathbf{Trans}(pk_t, \sigma_{ij})$, then $\sigma_{ij} \leftarrow \mathbf{Trace}(sk_t, \widehat{\sigma}_{ij})$.

Security Models

A secure TUDVTS scheme satisfies unforgeability, privacy, and traceability. The following are definitions of these security properties.

Unforgeability: the unforgeability of TUDVTS is similar to the unforgeability of UDVTS. TUDVTS encompasses two distinct forms of unforgeability. The first property refers to the fact that any adversary cannot output a forgery even if it is allowed to obtain transitive signatures on many other messages of its choice, i.e., the transitive signature unforgeability (TS-unforgeability). The second property refers to the impossibility for any adversary to forge a valid designated verifier signature even if they possess a valid translated signature from before, i.e., the designated verifier signature unforgeability (DV-unforgeability). Note that it is possible that the translated signature in DV-unforgeability is forged. As described in reference [29], we only consider this case where the translated signature is sent by the designator.

TS-unforgeability requires that any adversary cannot output a transitive signature on a disconnected edge. A disconnected edge is one that does not belong to the transitive closure of the graph composed of all pairs signed by the signer.

We let $\text{Forge}^{cma}_{\mathcal{A},\text{TS}}$ denote an execution of the experiment for a given TUDVTS and adversary $\mathcal{A}$. The experiment is defined as follows:

- **Setup**: the public parameters pp and the signer's public/secret key-pair (pk_s, sk_s) are generated by running **Setup** and **KGen**, respectively. Then, it is sent to $\mathcal{A}$.
- **TSign Query**: $\mathcal{A}$ picks an edge (i, j). Then, the transitive signature σ_{ij} is generated by running **TSign** and sent to $\mathcal{A}$.

In the end, $\mathcal{A}$ outputs an edge (i', j') and its signature $\sigma_{i'j'}$. The experiment outputs 1 if $1 \leftarrow$ **TVry**$(i', j', pk_s, \sigma_{i'j'})$ and $(i', j') \notin \widetilde{G}$, where $\widetilde{G}$ is the transitive closure of the graph G composed of all pairs (i, j) submitted to **TSign Query**.

The advantage of $\mathcal{A}$ in the $\text{Forge}^{cma}_{\mathcal{A},\text{TS}}$ is defined as

$$\text{Adv}^{cma}_{\mathcal{A},\text{TS}}(k) = \Pr[\text{Forge}^{cma}_{\mathcal{A},\text{TS}} = 1].$$

Definition 2 (TS-Unforgeability)**.** *The transitive signature is unforgeable under adaptive chosen-message attacks if* $\text{Adv}^{cma}_{\mathcal{A},\text{TS}}(k)$ *is negligible for any* $\mathcal{PPT}$ *adversary* $\mathcal{A}$*.*

DV-unforgeability requires that any adversary cannot output a designated verifier signature on an edge using the designated verifier's public key, where the edge and the public key have not been used as input to query the designated verifier signature.

We let $\text{Forge}^{cma,cpka}_{\mathcal{A},\text{TUDVTS}}$ denote an execution of the experiment for a given TUDVTS and adversary $\mathcal{A}$. The experiment is defined as follows:

- **Setup**: the public parameters pp and all users' public/secret key-pairs (pk_i, sk_i) are generated by running **Setup** and **KGen**, respectively. Then, it is sent to $\mathcal{A}$.
- **Trans Query**: $\mathcal{A}$ picks an edge (i, j). The transitive signature σ_{ij} is first generated by running **TSign**. Then, the translated signature $\widehat{\sigma}_{ij}$ is generated by running **Trans** and sent to $\mathcal{A}$. The secret value t is kept private.
- **DS Query**: $\mathcal{A}$ picks an edge (i, j), a verifier's public key pk_{v_i} and the corresponding translated signature $\widehat{\sigma}_{ij}$. He initially acquires the signature $\widehat{\sigma}_{ij}$ using the aforementioned procedure in the absence of the translated signature. Then, σ_{DV} is generated by running **DS** and sent to $\mathcal{A}$.
- **DV Query**: $\mathcal{A}$ requests the verification result of $((i, j), \sigma_{DV})$ using the chosen public key pk_{v_i}. The verification result is generated by running **DV** and sent to $\mathcal{A}$.
- **SKey Query**: $\mathcal{A}$ picks a verifier's public key pk_{v_i}. Then, the corresponding private key sk_{v_i} is sent to $\mathcal{A}$.

In the end, it returns a forgery σ'_{DV} on edge (i', j') with pk_{v_k} chosen by himself. The experiment outputs 1 if:

- $1 \leftarrow \mathbf{DV}(i', j', pk_s, sk_{v_k}, \sigma'_{DV})$.
- $((i', j'), pk_{v_k})$ has never been submitted to **DS Query**.
- pk_{v_k} has never been submitted to **SKey Query**.

The advantage of $\mathcal{A}$ in the $\text{Forge}^{cma,cpka}_{\mathcal{A},\text{TUDVTS}}$ is defined as

$$\text{Adv}^{cma,cpka}_{\mathcal{A},\text{TUDVTS}}(k) = \Pr[\text{Forge}^{cma,cpka}_{\mathcal{A},\text{TUDVTS}} = 1].$$

Definition 3 (DV-Unforgeability)**.** *A TUDVTS scheme is unforgeable under adaptive chosen-message and chosen-public-key attacks if* $\text{Adv}^{cma,cpka}_{\mathcal{A},\text{TUDVTS}}(k)$ *is negligible for any* $\mathcal{PPT}$ *adversary* $\mathcal{A}$*, where* $\mathcal{A}$ *invokes at most* q_1 **Trans Query**, q_2 **DS Query**, q_3 **DV Query** *and* q_4 **SKey Query** *in time t.*

The main difference between the above definition and [2] is that here, $\mathcal{A}$ queries the translated signature instead of the transitive signature.

Privacy: the privacy has been systematically discussed by Hou et al. in [2]. There are two types of privacy in the TUDVTS scheme: non-transferability of TUDVTS and privacy of transitive signature. As stated in [17], the designated verifier has the capability to produce a signature that cannot be distinguished from the signature generated by the signature holder. That is, the verifier is unable to provide convincing evidence to others that the signer has indeed signed the message. As stated in [1], The second condition states that a transitive signature and a composed signature on the same edge cannot be distinguished. This implies that the **Comp** algorithm can operate properly even if its input was generated using **Comp** itself.

The non-transferability of TUDVTS requires that any distinguisher cannot distinguish a designated verifier signature and the corresponding simulated signature.

We let $\text{Priv}^{cma,cpka}_{\mathcal{D},\text{TUDVTS}}$ denote an execution of the experiment for a given TUDVTS and distinguisher $\mathcal{D}$. The experiment is defined as follows:

- **Setup**: the public parameters pp and all users's public/secret key-pairs (pk_i, sk_i) are generated by running **Setup** and **KGen**, respectively. Then, it is sent to $\mathcal{D}$.
- **Stage 1**: the distinguisher $\mathcal{D}$ adaptively makes **Trans Query, DS Query, DV Query, Sim Query, SKey Query**: it responds to $\mathcal{D}$ in the same way as in game $\text{Forge}^{cma,cpka}_{\mathcal{A},\text{TUDVTS}}$.
  - **Sim Query:** assuming that $\mathcal{D}$ requests a simulated signature on edge (i, j) using the chosen public key pk_{v_i}, he initially acquires the signature $\widehat{\sigma}_{ij}$ using the aforementioned procedure in the absence of the translated signature. Then, the simulated signature $\widehat{\sigma}_{DV}$ is generated by running **Sim** and sent to $\mathcal{D}$.
- **Challenge stage**: $\mathcal{D}$ returns (i', j') and pk_{v_k} that satisfy the following conditions:
  - $(i', j') \notin G$.
  - $((i', j'), pk_{v_k})$ has never been submitted to **DS Query** and **Sim Query**.
  - pk_{v_k} has never been submitted to **SKey Query**.

  In reply, the experiment randomly samples $b \in \{0, 1\}$. If $b = 1$, then the signature σ_{DV} is generated by running **DS** and returned to $\mathcal{D}$. Otherwise, the signature $\widehat{\sigma}_{DV}$ is generated by running **Sim** and returned to $\mathcal{D}$.
- **Stage 2**. Upon the receipt of the signature, $\mathcal{D}$ can still proceed with the query in **stage 1**. However, he cannot choose $((i', j'), pk_{v_k})$ for **DS Query** or **Sim Query**.
- **Guess stage**. $\mathcal{D}$ outputs his guess $b' \in \{0, 1\}$.

If $b' = b$, then $\mathcal{D}$ wins the game. The advantage of $\mathcal{D}$ in the $\text{Priv}^{cma,cpka}_{\mathcal{D},\text{TUDVTS}}$ is defined as

$$\text{Adv}^{cma,cpka}_{\mathcal{D},\text{TUDVTS}}(k) = \left|\Pr[b' = b] - 1/2\right|.$$

Definition 4 (Non-Transferability of TUDVTS)**.** *If* $\mathrm{Adv}^{cma,cpka}_{\mathcal{D},\mathrm{TUDVTS}}(k)$ *is negligible for any* $\mathcal{PPT}$ *distinguisher* $\mathcal{D}$*, then the TUDVTS scheme is non-transferable under adaptive chosen-message and chosen-public-key attacks.*

Privacy of transitive signatures requires that any distinguisher cannot distinguish between transitive signatures and composed signatures on the same edges.

Definition 5 (Privacy of Transitive Signature)**.** *If the input of* **Comp** *is legitimate signatures, then the distributions that the composed signature and the signature generated by the signer follow are statistically indistinguishable.*

Traceability: as stated in [29], in order to determine whether the signer signed the message, we introduce a tracer in UDVTS. The tracer can restore the translated signature $\widehat{\sigma}_{ij}$ to its corresponding transitive signature σ_{ij}. According to TV-unforgeability, only the signer and the combiner have the ability to obtain valid transitive signatures. Thus, the tracer can track the identity of the translated signature generator by checking whether the transitive signature is valid.

Definition 6 (Traceability)**.** *If the translated signature is calculated by the combiner, then the transitive signature can be recovered by the tracer.*

4. Our TUDVTS Scheme

In this section, we present a concrete construction of the TUDVTS scheme and its security results.

4.1. Construction

Our scheme TUDVTS = (**Setup**, **KGen**, **TSign**, **TVry**, **Comp**, **Trans**, **DS**, **DV**, **Sim**, **Trace**) is constructed as follows: Given a group generator **GGen** that takes as input 1^k and outputs a triple $(\mathbb{G}, \mathbb{G}_{\mathbb{T}}, p)$, where p is a lagre prime and $\mathbb{G}, \mathbb{G}_{\mathbb{T}}$ are two p-order multiplicative cyclic groups. Denote $H_1 : \mathbb{N} \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p$ as two hash functions.

- **Setup**(1^k): this algorithm first obtains $(\mathbb{G}, \mathbb{G}_{\mathbb{T}}, p)$ by running **GGen**. Then, it generates a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbb{T}}$ and a generator g of $\mathbb{G}$ and outputs the public parameters $pp = (p, g, e, \mathbb{G}, \mathbb{G}_{\mathbb{T}}, H_1, H_2)$.
- **KGen**(pp): this algorithm takes as input the public parameters pp. It computes $A = g^a, B = g^b, D = g^d$, where $a, b, d \overset{R}{\leftarrow} Z_p^*$. It outputs three pairs of public/secret keys $\left\{(A = g^a, a), (B = g^b, b), (D = g^d, d)\right\}$, which denote the signer, the verifier, and the tracer, respectively.
- **TSign**(i, j, sk_s): this algorithm takes as input the signer's secret key a and nodes $i, j \in \mathbb{N}$. It computes $\sigma_{ij} = (h_i h_j^{-1})^a$ if $i < j$, where $h_i = H_1(i), h_j = H_1(j)$. If $i > j$, swap i and j.
- **TVry**$(i, j, pk_s, \sigma_{ij})$: this algorithm takes as input the signer's public key A, nodes $i, j \in \mathbb{N}$, and a signature σ_{ij}. If $e(\sigma_{ij}, g) = e(h_i h_j^{-1}, A)$, then it outputs 1 (accept). Otherwise, it outputs 0 (reject).
- **Comp**$(i, j, k, pk_s, \sigma_{ij}, \sigma_{jk})$: this algorithm takes as input the signer's public key A, nodes $i, j, k \in \mathbb{N}$, and two signatures σ_{ij} of (i, j) and σ_{jk} of (j, k). If σ_{ij} and σ_{jk} are both valid signatures, then it outputs the composed signature $\sigma_{ik} \leftarrow \sigma_{ij} \cdot \sigma_{jk}$ of (i, k). Otherwise, it outputs $\perp$.
- **Trans**(σ_{ij}, pk_t): this algorithm takes as input the tracer's public key D and the signature σ_{ij} of (i, j). The combiner computes $T_1 = g^t, T_2 = \sigma_{ij} D^t$, where $t \overset{R}{\leftarrow} Z_p^*$. He outputs the translated signature $\widehat{\sigma}_{ij} = (T_1, T_2)$.
- **DS**$(i, j, pk_v, \widehat{\sigma}_{ij}, t)$: this algorithm takes as input the verifier's public key B and the translated signature $\widehat{\sigma}_{ij}$. The combiner randomly chooses $r \in \mathbb{Z}_p$, and calculates

$R = e(g, B)^r$, $h = H_2(i, j, h_i, h_j, R)$, $R_1 = e(D^{th}, B)$, $T = T_2^h g^r \pmod{p}$ and $c = e(T, B)$. Then, he outputs the designated verifier signature $\sigma_{DV} = (R_1, h, c)$.

- **Sim**$(i, j, pk_s, pk_t, sk_v, \widehat{\sigma}_{ij})$: this algorithm takes as input the signer's public key A, the tracer's public key D and notes $i, j \in \mathbb{N}$. The verifier randomly picks $r' \in \mathbb{Z}_q$, and calculates $R' = e(g, B)^{r'}$, $h' = H_2(i, j, h_i, h_j, R')$, $R'_1 = e(D^{h'}, T_1^b)$, $T' = T_2^{h'} g^{r'} \pmod{p}$ and $c' = e(T', B)$. Then, he outputs a simulated signature $\widehat{\sigma}_{DV} = (R'_1, h', c')$.
- **DV**$(i, j, pk_s, sk_v, \sigma_{DV})$: this algorithm takes as input the signer's public key A, the tracer's public key D, notes $i, j \in \mathbb{N}$ and the signature σ_{DV}. The designated verifier calculates $P_1 = e(h_i h_j^{-1}, A^{bh})R_1$, $P = cP_1^{-1}$ and checks whether $h = H_2(i, j, h_i, h_j, P)$. If this holds, then the algorithm outputs 1 (accept). Otherwise, it outputs 0 (reject).
- **Trace**$(sk_t, \widehat{\sigma}_{ij})$: this algorithm takes as input the translated signature $\widehat{\sigma}_{ij}$. The tracer computes $\sigma_{ij} = T_2 / T_1^d$ and checks whether $e(\sigma_{ij}, g) = e(h_i h_j^{-1}, A)$. If this holds then σ_{ij} is legitimate.

4.2. Correctness

Here, we show five correctness properties in our TUDVTS.

- Correctness of **TSign**: if $\sigma_{ij} = (h_i h_j^{-1})^a \leftarrow \textbf{TSign}(i, j, sk_s)$, where $h_i = H_1(i), h_j = H_1(j)$, then

$$e(\sigma_{ij}, g) = e((h_i h_j^{-1})^a, g) = e(h_i h_j^{-1}, g^a) = e(h_i h_j^{-1}, A).$$

- Correctness of **Comp**: If $\sigma_{ik} = \sigma_{ij} \cdot \sigma_{jk} = (h_i h_j^{-1})^a \cdot (h_j h_k^{-1})^a = (h_i h_k^{-1})^a$, where $h_k = H_1(k)$, then

$$e(\sigma_{ik}, g) = e((h_i h_k^{-1})^a, g) = e(h_i h_k^{-1}, g^a) = e(h_i h_k^{-1}, A).$$

- Correctness of **DS**: if $R_1 = e(D^{th}, B)$ and

$$\begin{aligned}
P &= e(T_2^h g^r, B)[e(h_i h_j^{-1}, A^{bh})R_1]^{-1} \\
&= e(\sigma_{ij}^h, B)e(D^{th}, B)e(g^r, B)[e(h_i h_j^{-1}, A^{bh})R_1]^{-1} \\
&= e(h_i h_j^{-1}, A^{bh})e(D^{th}, B)e(g^r, B)[e(h_i h_j^{-1}, A^{bh})R_1]^{-1} \\
&= e(g^r, B),
\end{aligned}$$

then $h = H_2(i, j, h_i, h_j, P)$.

- Correctness of **Sim**: If $R'_1 = e(D^{h'}, T_1^b)$ and

$$\begin{aligned}
P &= e(T_2^{h'} g^{r'}, B)[e(h_i h_j^{-1}, A^{bh'})R'_1]^{-1} \\
&= e(\sigma_{ij}^{h'}, B)e(D^{th'}, B)e(g^{r'}, B)[e(h_i h_j^{-1}, A^{bh'})R'_1]^{-1} \\
&= e(h_i h_j^{-1}, A^{bh'})e(D^{h'}, T_1^b)e(g^{r'}, B)[e(h_i h_j^{-1}, A^{bh'})R'_1]^{-1} \\
&= e(g^{r'}, B),
\end{aligned}$$

then $h' = H_2(i, j, h_i, h_j, P)$.

- Correctness of **Trace**: if $T_1 = g^t, T_2 = \sigma_{ij} D^t$, then

$$\sigma_{ij} = T_2 / T_1^d = \sigma_{ij} D^t / g^{td} = \sigma_{ij} D^t / D^t = \sigma_{ij}.$$

4.3. Security Analysis

Here, we show the following theorems and provide rigorous formal proofs.

Theorem 1 (DV-Unforgeability). *Assuming the one-more BDH assumption holds in* $(\mathbb{G}, \mathbb{G}_{\mathbb{T}})$ *using the public parameters pp, we can conclude that the TUDVTS scheme satisfies DV-unforgeability under adaptive chosen-message and chosen-public-key attacks, with parameters* (t, q_1, q_2, q_3, q_4)*.*

Proof. Suppose there are n verifiers in the system. If there exist a $\mathcal{PPT}$ adversary $\mathcal{A}$ for breaking DV-unforgeability of TUDVTS with $\mathrm{Adv}_{\mathcal{A},\mathrm{TUDVTS}}^{cma,cpka}(k)$, we construct a challenger $\mathcal{C}$ for solving one-more BDH problem with $\mathrm{Adv}_{\mathcal{C}}^{one\text{-}moreBDH}(k)$, such that

$$\frac{1}{n}(1-\frac{1}{n})^{q_5}\mathrm{Adv}_{\mathcal{A},\mathrm{TUDVTS}}^{cma,cpka}(k) \leq \mathrm{Adv}_{\mathcal{C}}^{one\text{-}moreBDH}(k),\ \forall k \in \mathbb{N}.$$

Given an instance $pp = (e, \mathbb{G}, \mathbb{G}_{\mathbb{T}}, p, g, A, B)$ of one-more BDH problem, a public parameter D, $\mathcal{O}^{\mathcal{H}_1}(\cdot)$ and $\mathcal{O}^{\mathcal{CDH}}(\cdot)$, where $A = g^a, B = g^b, D = g^d$. $\mathcal{C}$'s aim is to output n values of $e(H_1(i), g)^{ab}$, under the requirement that has been made $\mathcal{O}^{\mathcal{CDH}}(\cdot)$ less than n queries. Denote the set that comprises all queried vertices as V. Denote the function that stores all queried edge signatures as $\Delta : V \times V \to \mathbb{G}$. $\mathcal{C}$ performs the simulation work according to this instance as follows with adversary $\mathcal{A}$:

- **Setup**:
  1. $\mathcal{C}$ sets A as the signer's public key and sets D as the tracer's public key.
  2. $\mathcal{C}$ computes $y_i = g^{x_i}$ as the public key of the verifier $i (i \neq l)$, where $x_i \xleftarrow{R} Z_p^*$ is his private key. For $i = l$, $\mathcal{C}$ sets B as his public key. Then, $\mathcal{C}$ maintains a list L and adds all the pairs (y_i, x_i) to L, where $x_l = \perp$.
  3. $\mathcal{C}$ sends $(pp, A, B, D, y_1, \cdots, y_n)$ to $\mathcal{A}$.
- H_1 **Query**:
  1. $\mathcal{C}$ maintains a list L_1 to record the hash values output by calling $\mathcal{O}^{\mathcal{H}_1}(\cdot)$.
  2. When $\mathcal{A}$ queries $H_1(i)$, $\mathcal{C}$ completes the following:
     - If $i \notin V$, then $V \leftarrow V \bigcup \{i\}$; $h_i \xleftarrow{R} \mathbb{G}$; $H_1(i) \leftarrow h_i$; $L_1 \leftarrow L_1 \cup h_i$; $\Delta(i,i) \leftarrow 1$.
     - $\mathcal{C}$ returns $H_1(i)$ to $\mathcal{A}$.
- H_2 **Query**:
  1. $\mathcal{C}$ maintains a list L_2 to record the hash values output by H_2 oracle. $\mathcal{A}$ randomly picks a verifier's public key y_i and a number $r_i \in \mathbb{Z}_p$ and computes $R = e(g, y_i)^{r_i}$.
  2. When $\mathcal{A}$ queries $H_2(i, j, h_i, h_j, R)$, $\mathcal{C}$ completes the following:
     - Firstly, obtains h_i and h_j as above if the two hash values do not exist.
     - Returns $h \xleftarrow{R} \mathbb{Z}_p$ to $\mathcal{A}$. Then, $\mathcal{C}$ adds all the message/value pairs (R, h) to L_2.
- **Trans Query**: assuming that $\mathcal{A}$ requests a translated signature on an edge (i, j) that he has chosen. In reply, $\mathcal{C}$ firstly obtains the signature σ_{ij} if $\Delta(i, j)$ is empty. $\mathcal{C}$ performs the following (assume $i < j$):
  1. If $i \notin V$ or $j \notin V$, $\mathcal{C}$ invokes $\mathcal{O}^{\mathcal{H}_1}(\cdot)$ to obtain $H_1(i)$ or $H_1(j)$.
  2. If $\Delta(i, j)$ is empty, then
     $\Delta(i,j) \leftarrow \mathcal{O}^{\mathcal{CDH}}(H_1(i)H_1(j)^{-1})$; $\Delta(j,i) \leftarrow \Delta(i,j)^{-1}$.
  3. For all $k \in V \setminus \{i, j\}$,
     If $\Delta(k,i)$ is empty, then $\Delta(k,j) \leftarrow \Delta(k,i) \cdot \Delta(i,j)$; $\Delta(j,k) \leftarrow \Delta(k,j)^{-1}$.
     If $\Delta(k,j)$ is empty, then $\Delta(k,i) \leftarrow \Delta(k,j) \cdot \Delta(j,i)$; $\Delta(i,k) \leftarrow \Delta(k,i)^{-1}$.
  4. $\sigma_{ij} \leftarrow \Delta(i,j)$.
  5. $\mathcal{C}$ randomly picks $t \in \mathbb{Z}_p$, computes $T_1 = g^t$ and $T_2 = \sigma_{ij} D^t$. $\mathcal{C}$ maintains a list L_T and stores all the random numbers t to L_T.
  6. Returns (T_1, T_2) to $\mathcal{A}$, and stores the corresponding t in L_T.
- **DS Query**: assuming that $\mathcal{A}$ requests a designated verifier signature on edge (i, j) using the chosen public key y_i, $\mathcal{C}$ firstly obtains the translated signature (T_1, T_2) as above if the signature does not exist. Then, $\mathcal{C}$ randomly selects $r \in \mathbb{Z}_p$ and calculates

$R = e(g, B)^r$, $R_1 = e(D^{th}, B)$, $T = T_2^h g^r \pmod{p}$ and $c = e(T, B)$, returns $\sigma_{DV} = (R_1, h, c)$ to adversary $\mathcal{A}$.

- **DV Query**: assuming that $\mathcal{A}$ requests a verification result of $((i, j), \sigma_{DV})$ using the chosen public key y_i, $\mathcal{C}$ calculates $P_1 = e(h_i h_j^{-1}, A^{bh})R_1$, $P = cP_1^{-1}$, returns 1 if $h = H_2(i, j, h_i, h_j, P)$, otherwise returns 0.
- **SKey Query**: assuming that $\mathcal{A}$ requests the corresponding private key using the chosen public key y_i, $\mathcal{C}$ outputs the corresponding private key x_i if $i \neq l$. Otherwise, the operation aborts. The probability of $\mathcal{C}$ not aborting is $(1 - \frac{1}{n})^{q_5}$.
- **Forgery**: eventually, $\mathcal{A}$ takes as input $r^* \xleftarrow{R} \mathbb{Z}_P$ and $R^* = e(g, y_k)^{r^*}$, obtains h^* by asking for the H$_2$ oracle, when the edge (i^*, j^*) and the verifier's public key y_k chosen by himself. Then, he obtains a translated signature $\widehat{\sigma}_{ij}^* = (T_1^*, T_2^*)$ by **Trans Query** and computes $c^* = e(T_2'^{h^*} g^{r^*}, y_k)$. In the end, $\mathcal{A}$ returns a forgery signature $\sigma_{DV}^* = (R_1^*, h^*, c^*)$. If $y_k \neq B$, then the operation aborts. The probability of $\mathcal{C}$ not aborting is $\frac{1}{n}$. We assume $i^*, j^* \in V$, otherwise $\mathcal{C}$ can query the $\mathcal{O}^{\mathcal{H}_1}$ by himself. Let the graph $G = (V, E)$ be composed of all pairs (i, j) submitted to **Trans Query** and let $\widetilde{G} = (V, \widetilde{E})$ be the transitive closure of G. σ_{DV}^* is valid if it satisfies the following:
  - $1 \leftarrow$**DV**$(i, j, pk_s, sk_{v_k}, \sigma_{DV}^*)$.
  - $((i^*, j^*), B)$ has never been submitted to **DS Query**.
  - y_k has never been submitted to **SKey Query**.

$\mathcal{C}$ can compute the BDH values of all vertices in V by using σ_{DV}^*. V is decomposed into m disjoint subsets $V_t \subset V, \forall t = 1, 2, \cdots, m$, which is intended to separate i^* and j^*. Let $i^* \in V_1$ but $j^* \notin V_1$. For all $t \neq 1$, $\mathcal{C}$ picks a reference node $s_t \in V_t$. The BDH values for all nodes in V_t can be calculated by performing the following steps:

1. $\sigma_{s_t} = (h_{s_t})^a = (H_1(s_t))^a \leftarrow \mathcal{O}^{\mathcal{CDH}}(H_1(S_t))$.
2. $c_{s_t} \leftarrow e(\sigma_{s_t}, B)$.
   For all $z \in V_t \setminus \{s_t\}$,
3. $c_{zs_t} \leftarrow e(\sigma_{zs_t}, B)$.
4. $c_z \leftarrow c_{zs_t} \cdot c_{s_t}$.

Otherwise, the BDH values for all nodes in V_1 can be calculated by performing the following steps:

1. $c_{i^*j^*} \leftarrow [c^* \cdot R^{*-1}]^{h^{-1}} \cdot R_1^{*-1}$.
2. $c_{i^*} \leftarrow c_{i^*j^*} \cdot c_{j^*}$.
   For all $k \in V_1 \setminus \{i^*\}$,
3. $c_{ki^*} \leftarrow e(\sigma_{ki^*}, B)$.
4. $c_k \leftarrow c_{ki^*} \cdot c_{i^*}$.

Now $\mathcal{C}$ outputs the BDH values $c_i = e(\sigma_i, B) = e(g, H_1(i))^{ab}$ for all $i \in V$. For each $V_t (t \neq 1)$, $\mathcal{C}$ queried $\mathcal{O}^{\mathcal{CDH}}(\cdot)$ $|V_t| - 1$ times to compute the signature σ_{zs_t} (the number of edges in a minimal spanning tree of V_t), and queried $\mathcal{O}^{\mathcal{CDH}}(\cdot)$ once to compute σ_{s_t}, the number of $\mathcal{O}^{\mathcal{CDH}}(\cdot)$ for each $V_t (t \neq 1)$ is summed to $|V_t|$. For V_1, $\mathcal{C}$ did not need the additional query to compute the σ_{i^*}. Therefore, $\mathcal{C}$ outputs $|V|$ BDH values using $\sum_{t \neq 1} |V_t| + |V_1| - 1 = |V| - 1$ CDH oracle and, hence, solves the one-more BDH problem.

Below, we consider the probability of $\mathcal{C}$ not aborting. If $\mathcal{C}$ does not abort, the following conditions must be satisfied:

- $y_l = B$ has never been submitted to **SKey Query**.
- In the output forgery, $\mathcal{A}$ chooses the public key B.

Obviously, the probability that both conditions are satisfied is greater than $\frac{1}{n}(1 - \frac{1}{n})^{q_5}$; hence,

$$\frac{1}{n}(1 - \frac{1}{n})^{q_5} \mathrm{Adv}_{\mathcal{A},\mathrm{TUDVTS}}^{cma,cpka}(k) \leq \mathrm{Adv}_{\mathcal{C}}^{one\text{-}more\ BDH}(k).$$

This completes the proof. □

Theorem 2 (TS-Unforgeability)**.** *Assuming the one-more BDH assumption holds in the bilinear group pair* ($\mathbb{G}, \mathbb{G}_{\mathbb{T}}$) *using the public parameters* pp*, we can conclude that the transitive signature is unforgeable under an adaptive chosen-message attack.*

Proof. Given an instance $pp = (e, \mathbb{G}, \mathbb{G}_{\mathbb{T}}, p, g, A, B)$ of one-more BDH problem, H_1 oracle $\mathcal{O}^{\mathcal{H}_1}(\cdot)$ and the CDH oracle $\mathcal{O}^{\mathcal{CDH}}(\cdot)$. $\mathcal{C}$'s aim is to output n values of $e(H_1(i), g)^{ab}$, but the CDH oracle has been made strictly less than n queries. Let V denote the set that comprises all queried vertices. $\mathcal{C}$ performs the simulation work according to this instance as follows with adversary $\mathcal{A}$:

- **Setup**: $\mathcal{C}$ sets A as the signer's public key and returns (A, pp) to $\mathcal{A}$.
- **TSign Query**: assuming that $\mathcal{A}$ requests a signature on an edge (i, j) that he has chosen. In reply, $\mathcal{C}$ performs the following (Assume $i < j$):
  1. If $i \notin V$ or $j \notin V$, $\mathcal{C}$ invokes $\mathcal{O}^{\mathcal{H}_1}(\cdot)$ to obtain $H_1(i)$ or $H_1(j)$.
  2. If $\Delta(i, j)$ is empty, then
     $\Delta(i, j) \leftarrow \mathcal{O}^{\mathcal{CDH}}(H_1(i)H_1(j)^{-1})$; $\Delta(j, i) \leftarrow \Delta(i, j)^{-1}$.
  3. For all $k \in V \setminus \{i, j\}$,
     If $\Delta(k, i)$ is empty, then $\Delta(k, j) \leftarrow \Delta(k, i) \cdot \Delta(i, j)$; $\Delta(j, k) \leftarrow \Delta(k, j)^{-1}$.
     If $\Delta(k, j)$ is empty, then $\Delta(k, i) \leftarrow \Delta(k, j) \cdot \Delta(j, i)$; $\Delta(i, k) \leftarrow \Delta(k, i)^{-1}$.
  4. $\sigma_{ij} \leftarrow \Delta(i, j)$.
  5. Returns σ_{ij} to $\mathcal{A}$.
- Then, $\mathcal{A}$ adaptively invokes the H_1 oracle. $\mathcal{C}$ responds to $\mathcal{A}$ in the same way as in the proof above.

  In the end, $\mathcal{A}$ outputs an edge (i', j') and the signature $\sigma_{i'j'}$. If $1 \leftarrow$ **TVry**$(i', j', A, \sigma_{i'j'})$ and $(i', j') \notin \widetilde{G}$, then $\sigma_{i'j'}$ is said to be a valid forgery.

The one-more BDH problem is solved based on this forgery, as in the proof of the Theorem 1 method. $\sigma_{i'j'}$ is used to find the BDH value of vertex i'. Thus, solving for the BDH values of all vertices in V, $\mathcal{C}$ queries at most $|V| - 1$ $\mathcal{O}^{\mathcal{CDH}}(\cdot)$. This completes the proof. □

Theorem 3 (Non-Transferability of TUDVTS)**.** *Our scheme satisfies the non-transferability of TUDVTS against the adaptive chosen-message and chosen-public-key $\mathcal{PPT}$ distinguisher $\mathcal{D}$.*

Proof. Suppose $\mathcal{C}$ is a challenger. $\mathcal{C}$'s goal is to distinguish between a designated verifier signature and its corresponding simulated signature.

- **Setup**:
  1. $\mathcal{C}$ sets $A = g^a, D = g^d$ as the public key of the signer and the tracer, respectively, where $a, d \xleftarrow{R} Z_p^*$.
  2. $\mathcal{C}$ sets $y_i = g^{x_i}$ as the ith ($i \neq l$) verifier's public/private key-pair, where $x_i \xleftarrow{R} Z_p^*$. Then, $\mathcal{C}$ maintains a list L and adds all the public/private key-pairs (y_i, x_i) to L.
  3. $\mathcal{C}$ sends $(pp, A, y_1, \cdots, y_n)$ to $\mathcal{D}$.
- **Stage 1**: the distinguisher $\mathcal{D}$ adaptively invokes H$_1$ **Query**, H$_2$ **Query**, **Trans Query, DS Query, DV Query, Sim Query, SKey Query**. It responds to $\mathcal{D}$ in the same way as in game Forge$^{cma,cpka}_{\mathcal{A},\mathrm{TUDVTS}}$.
  - **SKey Query**: if $\mathcal{D}$ requests the private key associated with a chosen public key y_i, $\mathcal{C}$ verifies the list L and provides the matching private key x_i in response.
  - **Sim Query**: assuming that $\mathcal{D}$ requests a simulated signature on edge (i, j) using the chosen public key y_i, $\mathcal{C}$ firstly obtains the translated signature (T_1, T_2) as above if the signature does not exist. Then, $\mathcal{C}$ randomly selects $r \in \mathbb{Z}_p$ and calculates $R' = e(g, y_i)^{r'}$, $R_1' = e(D^{h'}, T_1^b)$, $T' = T_2^{h'} g^{r'} \pmod{p}$ and $c' = e(T', y_i)$., returns $\widehat{\sigma}_{DV} = (R_1', h', c')$ to distinguisher $\mathcal{D}$.

- **Challenge stage**: $\mathcal{D}$ returns (i', j') and y_k that satisfy the following conditions:
  - $(i', j') \notin G$.
  - $((i', j'), y_k)$ has never been submitted to **DS Query** and **Sim Query**.
  - y_k has never been submitted to **SKey Query**.

  In reply, $\mathcal{C}$ randomly samples $b \in \{0, 1\}$. If $b = 1$, then the signature $\sigma_{DV} = (R_1, h, c)$ is generated by running **DS** and returned to $\mathcal{D}$. Otherwise, the signature $\widehat{\sigma}_{DV} = (R'_1, h', c')$ is generated by running **Sim** and returned to $\mathcal{D}$.
- **Stage 2**: upon the receipt of the signature, $\mathcal{D}$ can still proceed with the query in **Stage 1**. However, he cannot query the translated signature on edge (i', j'), and cannot choose $((i', j'), y_k)$ for **DS Query** or **Sim Query**.
- **Guess stage**: $\mathcal{D}$ outputs his guess $b' \in \{0, 1\}$.

  Suppose $\sigma^*_{DV} = (R^*_1, h^*, c^*)$ is a valid designated verifier signature, since

$$\Pr[\sigma^*_{DV} = \sigma_{DV}] = \Pr\begin{bmatrix} R^* = e(g,B)^{r^*} = e(g,B)^{r} = R \\ h^* = h, \\ R^*_1 = e(D^{t^*h^*}, B) = e(D^{th}, B) = R_1 \\ c^* = c \end{bmatrix} = \Pr\begin{bmatrix} r^* = r \\ t^* = t \end{bmatrix} = \frac{1}{p^2},$$

where $r, r^*, t, t^* \in \mathbb{Z}_p$. Similarly,

$$\Pr[\sigma^*_{DV} = \widehat{\sigma}_{DV}] = \Pr\begin{bmatrix} R^* = e(g,B)^{r^*} = e(g,B)^{r'} = R' \\ h^* = h' \\ R^*_1 = e(D^{t^*h^*}, B) = e(D^{t'h'}, B) = R'_1 \\ c^* = c' \end{bmatrix} = \Pr\begin{bmatrix} r^* = r' \\ t^* = t' \end{bmatrix} = \frac{1}{p^2},$$

where $r', r^*, t', t^* \in \mathbb{Z}_p$.

Therefore, the designated verifier signature and the simulated signature on the same edge and public key are statistically indistinguishable. This completes the proof. □

Theorem 4 (Privacy of Transitive Signature)**.** *If the input of* **Comp** *is legitimate signatures, then the distributions of the composed signature and the signature generated by the signer are statistically indistinguishable.*

Proof. Let σ_{ij} and σ_{jk} be the legitimate signatures of edge (i, j) and edge (j, k) with respect to the public key A. Then, $\sigma_{ij} = (h_i h_j^{-1})^a$ and $\sigma_{jk} = (h_j h_k^{-1})^a$, where $h_i = H_1(i), h_j = H_1(j)$, $h_k = H_1(k)$. Taking $(A, i, j, k, \sigma_{ij}, \sigma_{jk})$ as input **Comp**, it outputs the composed signature

$$\sigma_{ik} = \sigma_{ij} \cdot \sigma_{jk} = (h_i h_j^{-1})^a \cdot (h_j h_k^{-1})^a = (h_i h_k^{-1})^a.$$

Hence, the signature generated by the signer and the composed signature on the same edge are statistically indistinguishable. This completes the proof. □

Theorem 5 (Traceability)**.** *The TUDVTS scheme is traceable, which can check whether the signer signed the message.*

Proof. Suppose the verifier sends a translated signature $\widehat{\sigma}_{ij} = (T_1, T_2)$ to the tracer. The tracer can calculate:

$$\sigma_{ij} = T_2 / T_1^d.$$

Then, the tracer checks whether $e(\sigma_{ij}, g) = e(h_i h_j^{-1}, A)$ holds. If this holds then σ_{ij} is legitimate, i.e., the signer signed the edge. This completes the proof. □

5. Efficiency Analysis

Since no other TUDVTS schemes have been proposed, we mainly discuss the size of the signature generated by the scheme and the time cost of its sub-algorithms and compare its efficiency with UDVTS. Let $|\mathbb{Z}_p|, |\mathbb{G}|, |\mathbb{G}_T|$ denote the bit length of the element in $\mathbb{Z}_p, \mathbb{G}$ and $\mathbb{G}_T$, respectively. Let t_1, t_2, t_3, t_4, t_5 be the time cost of performing one *exponentiation*, *pairing*, *hash*, *inverse*, and *multiply* operation, respectively. We have the following Tables 1 and 2.

Table 1. Signature size.

Algorithm	TSign	TVry	Comp	Trans	DS	DV	Sim
UDVTS	$|\mathbb{G}|$	—	$|\mathbb{G}|$	—	$|\mathbb{G}_T|$	—	$|\mathbb{G}_T|$
TUDVTS	$|\mathbb{G}|$	—	$|\mathbb{G}|$	$2|\mathbb{G}|$	$|\mathbb{Z}_p| + 2|\mathbb{G}_T|$	—	$|\mathbb{Z}_p| + 2|\mathbb{G}_T|$

Table 2. Time cost.

Algorithm	TSign	TVry	Comp	Trans	DS	DV	Sim
UDVTS	$t_1 + 2t_3 + t_4 + t_5$	$2t_2 + 2t_3 + t_4$	t_5	—	t_2	$t_1 + t_2 + 2t_3 + t_4 + t_5$	$t_1 + t_2 + 2t_3 + t_4 + t_5$
TUDVTS	$t_1 + 2t_3 + t_4 + t_5$	$2t_2 + 2t_3 + t_4$	t_5	$2t_1 + t_5$	$4t_1 + 3t_2 + t_3 + 2t_5$	$t_1 + t_2 + t_3 + 2t_4 + 4t_5$	$5t_1 + 3t_2 + t_3 + t_5$

6. Conclusions

This paper introduces the concept of traceable universal designated verifier transitive signatures (TUDVTS) and formally depicts the framework of TUDVTS and its security model. In the new framework, the tracer can find the true source of the disputed signature. We next construct a concrete scheme in the random oracle model, whose unforgeability relies on the one-more BDH assumption. The public key, the transitive signature, the translated signature, and the designated verifier signature in our construction are $3|\mathbb{G}|$ bits, $|\mathbb{G}|$ bits, $2|\mathbb{G}|$ bits, and $2|\mathbb{G}_T| + |\mathbb{Z}_p|$ bits, respectively.

Author Contributions: Conceptualization, S.H., C.L. and S.Y.; methodology, S.H., C.L. and S.Y.; writing—original draft preparation, S.H.; writing—review and editing, C.L. and S.Y.; supervision, C.L. and S.Y.; project administration, S.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (62032005, 62172096, 62272104), Natural Science Foundation of Fujian Province, China (2020J02016).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Micali, S.; Rivest, R.L. Transitive Signature Schemes. In *Topics in Cryptology—CT-RSA 2002*; Preneel, B., Ed.; Springer: Berlin/Heidelberg, Germany, 2002; pp. 236–243.
2. Hou, S.; Huang, X.; Liu, J.K.; Li, J.; Xu, L. Universal Designated Verifier Transitive Signatures for Graph-Based Big Data. *Inf. Sci.* **2015**, *318*, 144–156. [CrossRef]
3. Bellare, M.; Neven, G. Transitive Signatures Based on Factoring and RSA. In *Advances in Cryptology—ASIACRYPT 2002*; Zheng, Y., Ed.; Springer: Berlin/Heidelberg, Germany, 2002; pp. 397–414.
4. Bellare, M.; Neven, G. Transitive Signatures: New Schemes and Proofs. *IEEE Trans. Inf. Theory* **2005**, *51*, 2133–2151. [CrossRef]
5. Wang, L.; Cao, Z.; Zheng, S.; Huang, X.; Yang, Y. Transitive Signatures from Braid Groups. In *Progress in Cryptology—INDOCRYPT 2007*; INDOCRYPT'07; Springer: Berlin/Heidelberg, Germany, 2007; pp. 183–196.

6. Lin, C.; Zhu, F.; Wu, W.; Liang, K.; Choo, K.K.R. A New Transitive Signature Scheme. In *Network and System Security*; Chen, J., Piuri, V., Su, C., Yung, M., Eds.; Springer: Cham, Switzerland, 2016; pp. 156–167.
7. Zhu, F.; Zhang, Y.; Lin, C.; Wu, W.; Meng, R. A Universal Designated Multi-Verifier Transitive Signature Scheme. In *Information Security and Cryptology*; Chen, X., Lin, D., Yung, M., Eds.; Springer: Cham, Switzerland, 2018; pp. 180–195.
8. Lin, C.; Wu, W.; Huang, X.; Xu, L. A New Universal Designated Verifier Transitive Signature Scheme for Big Graph Data. *J. Comput. Syst. Sci.* **2017**, *83*, 73–83. [CrossRef]
9. Noh, G.; Jeong, I.R. Transitive Signature Schemes for Undirected Graphs from Lattices. *KSII Trans. Internet Inf. Syst.* **2019**, *13*, 3316–3332.
10. Noh, G.; Chun, J.Y. Identity-Based Transitive Signature Scheme from Lattices. *J. Korea Inst. Inf. Secur. Cryptol.* **2021**, *31*, 509–516.
11. Rivest, R.; Hohenberger, S. The Cryptographic Impact of Groups with Infeasible Inversion. Doctoral Dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 2003.
12. Kuwakado, H.; Tanaka, H. Transitive Signature Scheme for Directed Trees. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2003**, *86-A*, 1120–1126.
13. Yi, X. Directed Transitive Signature Scheme. In *Topics in Cryptology—CT-RSA 2007, The Cryptographers' Track at the RSA Conference 2007, San Francisco, CA, USA, 5–9 February 2007, Proceedings*; Abe, M., Ed.; Springer: Berlin/Heidelberg, Germnay, 2007; Volume 4377, pp. 129–144.
14. Neven, G. A Simple Transitive Signature Scheme for Directed Trees. *Theor. Comput. Sci.* **2008**, *396*, 277–282. [CrossRef]
15. Camacho, P.; Hevia, A. Short Transitive Signatures for Directed Trees. In *Topics in Cryptology—CT-RSA 2012*; Dunkelman, O., Ed.; CT-RSA 2012. Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7178, pp. 35–50.
16. Xu, J.; Chang, E.; Zhou, J. Directed Transitive Signature on Directed Tree. In Proceedings of the Singapore Cyber-Security Conference (SG-CRC) 2016-Cyber-Security by Design, Singapore, 14–15 January 2016; Cryptology and Information Security Series; Mathur, A., Roychoudhury, A., Eds.; IOS Press: Amsterdam, The Netherlands, 2016; Volume 14, pp. 91–98.
17. Steinfeld, R.; Bull, L.; Wang, H.; Pieprzyk, J. Universal Designated-Verifier Signatures. In *Advances in Cryptology—ASIACRYPT 2003*; Laih, C.S., Ed.; ASIACRYPT 2003. Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2894, pp. 523–542.
18. Boneh, D.; Lynn, B.; Shacham, H. Short Signatures from the Weil Pairing. In *Advances in Cryptology—ASIACRYPT 2001*; Boyd, C., Ed.; Springer: Berlin/Heidelberg, Germany, 2001; pp. 514–532.
19. Steinfeld, R.; Wang, H.; Pieprzyk, J. Efficient Extension of Standard Schnorr/RSA Signatures into Universal Designated-Verifier Signatures. In *Public Key Cryptography-PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, 1–4 March 2004*; Lecture Notes in Computer Science; Bao, F., Deng, R.H., Zhou, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; Volume 2947, pp. 86–100.
20. Ng, C.Y.; Susilo, W.; Mu, Y. Universal Designated Multi Verifier Signature Schemes. In Proceedings of the 11th International Conference on Parallel and Distributed Systems, ICPADS 2005, Fuduoka, Japan, 20–22 July 2005; pp. 305–309.
21. Zhang, F.; Susilo, W.; Mu, Y.; Chen, X. Identity-Based Universal Designated Verifier Signatures. In *Embedded and Ubiquitous Computing-EUC 2005 Workshops, EUC 2005 Workshops: UISW, NCUS, SecUbiq, USN, and TAUES, Nagasaki, Japan, 6–9 December 2005, Proceedings*; Lecture Notes in Computer Science; Enokido, T., Yan, L., Xiao, B., Kim, D., Dai, Y., Yang, L.T., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3823, pp. 825–834.
22. Zhang, R.; Furukawa, J.; Imai, H. Short Signature and Universal Designated Verifier Signature Without Random Oracles. In *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, 7–10 June 2005, Proceedings*; Lecture Notes in Computer Science; Ioannidis, J., Keromytis, A.D., Yung, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3531, pp. 483–498.
23. Shahandashti, S.F.; Safavi-Naini, R. Generic Constructions for Universal Designated-Verifier Signatures and Identitybased Signatures from Standard Signatures. *IET Inf. Secur.* **2009**, *3*, 152–176. [CrossRef]
24. Chang, T.Y. An ID-Based Multi-Signer Universal Designated Multi-Verifier Signature Scheme. *Inf. Comput.* **2011**, *209*, 1007–1015. [CrossRef]
25. Huang, X.; Susilo, W.; Mu, Y.; Wu, W. Universal Designated Verifier Signature Without Delegatability. In *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, 4–7 December 2006, Proceedings*; Lecture Notes in Computer Science; Ning, P., Qing, S., Li, N., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4307, pp. 479–498.
26. Li, J.; Wang, Y. Universal Designated Verifier Ring Signature (Proof) Without Random Oracles. In *Emerging Directions in Embedded and Ubiquitous Computing, EUC 2006 Workshops: NCUS, SecUbiq, USN, TRUST, ESO, and MSA, Seoul, Korea, 1–4 August 2006, Proceedings*; Lecture Notes in Computer Science; Zhou, X., Sokolsky, O., Yan, L., Jung, E., Shao, Z., Mu, Y., Lee, D.C., Kim, D., Jeong, Y., Xu, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4097, pp. 332–341.
27. Wang, M.; Zhang, Y.; Ma, J.; Wu, W. A Universal Designated Multi Verifiers Content Extraction Signature Scheme. *Int. J. Comput. Sci. Eng.* **2020**, *21*, 49–59. [CrossRef]
28. Li, B.H.; Liu, Y.Z.; Yang, S. Lattice-Based Universal Designated Verifier Signatures. In Proceedings of the 15th International Conference on e-Business Engineering, ICEBE, Xi'an, China, 12–14 October 2018; IEEE Computer Society, pp. 329–334.

29. Tang, F.; Ma, S.; Ma, C.L. Traceable Universal Designated Verifier Signature Proof Scheme. *Ruan Jian Xue Bao/J. Softw.* **2022**, *33*, 4305.
30. Gao, W.; Wang, G.; Wang, X.; Li, F. Round-Optimal ID-Based Blind Signature Schemes without ROS Assumption. *J. Commun.* **2012**, *7*, 909–920. [CrossRef]