information

Article

A New Approach to Keep the Privacy Information of the Signer in a Digital Signature Scheme

Dung Hoang Duong [1], Willy Susilo [1] and Viet Cuong Trinh [2,*]

[1] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Northfields Avenue, Wollongong, NSW 2500, Australia; hduong@uow.edu.au (D.H.D.); wsusilo@uow.edu.au (W.S.)

[2] Faculty of Information and Communication Technology, Hong Duc University, 565 Quang Trung, Thanh Hoa, Vietnam

* Correspondence: trinhvietcuong@hdu.edu.vn

Received: 8 April 2020; Accepted: 8 May 2020; Published: 11 May 2020

Abstract: In modern applications, such as Electronic Voting, e-Health, e-Cash, there is a need that the validity of a signature should be verified by only one responsible person. This is opposite to the traditional digital signature scheme where anybody can verify a signature. There have been several solutions for this problem, the first one is we combine a signature scheme with an encryption scheme; the second one is to use the group signature; and the last one is to use the strong designated verifier signature scheme with the undeniable property. In this paper, we extend the traditional digital signature scheme to propose a new solution for the aforementioned problem. Our extension is in the sense that only a designated verifier (responsible person) can verify a signer's signature, and if necessary (in case the signer refuses to admit his/her signature) the designated verifier without revealing his/her secret key is able to prove to anybody that the signer has actually generated the signature. The comparison between our proposed solution and the three existing solutions shows that our proposed solution is the best one in terms of both security and efficiency.

Keywords: digital signature; signer privacy preserving; group signature; strong designated verifier signature

1. Introduction

The digital signature scheme has been widely used in practical applications today. As a handwritten signature, the validity of a digital signature should be publicly verified by anyone. However, in some specific modern applications, such as Electronic Voting, e-Health, e-Cash, etc, we would like that only the responsible person can verify the validity of a signature. For example, in an Electronic Voting system (similarly for e-Health application, e-Payment application or e-Bidding application) where a responsible person (the host such as the government for example) usually would like to ask users (voters) to agree or disagree on a plan. To this aim, a voter signs on an agreeing or disagreeing message, then uploads his/her signature on a public server. However, in some specific plans, the voter is not willing to let other voters in the system know whether or not he/she agrees or disagrees on these plans. In fact, the voter would like that there is only one responsible person who can know whether or not he/she agrees or disagrees on a given plan. On the other hand, if necessary, in case the voter refuses to admit his/her signature, the responsible person without revealing his/her secret key should be able to prove to anybody that the voter has actually generated the signature. To deal with this problem, there exist several following methods:

- Method 1: using additional a public key encryption scheme. A voter encrypts his/her signature under the public key of the responsible person, so only the responsible person with his/her corresponding secret key at hands can decrypt and then check the validity of the voter's signature. Other voters in the system cannot decrypt, hence they cannot verify the voter's signature, or they cannot know whether or not the voter agrees or disagrees on a given plan.

- Method 2: using the group signature scheme [1]. In a group signature scheme, a user in a group can generate a signature on behalf of the group, and only the group manager can know exactly who has actually generated the signature. So, if the group includes all users in the system and the responsible person plays the role of the group manager, only the responsible person can know whether or not the voter agrees or disagrees on a given plan.
- Method 3: using the recently introduced strong designated verifier signature scheme [2] (SDVS for short). In a SDVS scheme, a signature only can be checked by a chosen designated verifier, but a signature can be generated by both the signer and the designated verifier. That means no one can tell a signature has been actually generated by the signer or the designated verifier. This special property is useful in some specific practical scenarios as mentioned in [2–7], however it is undesirable in some other scenarios such as Electronic Voting, e-Health, e-Payment applications. In fact, in the e-Voting system, if we use the SDVS scheme, then the responsible person (the host such as the government for example) will play the role of the designated verifier. Moreover, the responsible person also can generate the voter's signature. This leads to the fact that if the responsible person would like that a voter agrees on a given plan, he/she simply forges the signature of this voter, and the voter cannot prove to anybody that he/she actually hasn't generated this signature. In contrast, in some cases, the voter also can refuse that he/she has already generated this signature, and of course the responsible person also cannot prove that the voter has actually generated this signature. This obviously is not a desirable property for the e-Voting system. Recent works [8–10] extended the SDVS scheme to propose a new method to deal with the above problem, the proposed schemes are named SDVS schemes with the undeniable property. In this type of scheme, we have a judge who can decide that a given signature is generated by a signer or a designated verifier. This type of scheme can deal with well the disputing between the signer and designated verifier, it, however, has the shortcoming that the judgment should be honest (this could face problems when the scheme is used in practice), and obviously to maintain the judgment, the system has to pay the cost on the storage, computation complexity and communication overhead.

We also note that in method 2 and method 3, the user cannot generate the usual signature (anyone can verify the signature). So, when we use method 2 or method 3 in the e-Voting system for example, to support the functionality of generating the usual signature we have to use an additional traditional signature scheme.

We motivate from the aforementioned problem to ask a question that whether or not there exists an efficient signature scheme where a signer can freely choose to generate either a usual signature (anyone can verify this signature) or a designated verifier signature (only the designated verifier can verify this signature, and the designated verifier cannot generate this signature as in SDVS scheme). Moreover, if necessary, in case the signer refuses to admit his/her signature, the designated verifier without revealing his/her secret key is able to prove to anybody that the signer has actually produced the signature. Regarding the application, we believe that besides Electronic Voting, e-Health, e-Payment, this type of scheme could also find many other applications in practice.

1.1. Our Contribution

In this paper, we affirmatively answer the above question by proposing an efficient signature scheme where a user can choose to produce either a usual signature or a designated verifier signature. Concretely, our scheme has the following properties:

- only the designated verifier can verify the signer's designated verifier signature, it, therefore, can keep the privacy information of the signer;

- the designated verifier cannot forge the signer's designated verifier signature. This means that the responsible person cannot forge the voter's signature as in the case of Electronic Voting application;
- in case the signer refuses to admit his/her designated verifier signature, the designated verifier without revealing his/her secret key is able to prove to anybody that the signer has actually generated this signature. By this way, there is no need to have a judgment in the system, and more importantly, this forces the signer to be honest;
- the user can choose to generate either a usual signature or a designated verifier signature. That means our proposed scheme can be seen as an improvement of the traditional signature scheme in terms of functionality. Note that in the existing group signature scheme and SDVS scheme, the signer cannot generate the usual signature.

We name our scheme signer privacy-preserving digital signature scheme with undeniable property (SPPS for short) scheme. In our scheme, to achieve the privacy information of the signer, technically we prove that everyone (except the designated verifier) is not able to distinguish between the signer's designated verifier signature and random elements. That means everyone (except the designated verifier) cannot verify the signer's designated verifier signature. The comparison Table 1 shows that our proposed method is one of the most efficient ones: using additional encryption scheme (method 1—M1); using group signature (method 2—M2); and finally using SDVS schemes with undeniable property (method 3—M3). As shown in the comparison table, our scheme just needs two exponentiations for signing and three pairings for verifying. Note that the time to compute one pairing is very fast and it is practical even for lightweight devices. More precisely, we refer to Section 5.3 in the paper [11], where the authors have tested that the time to compute one paring using a weak device is about 5.5 ms and the time to compute one exponentiation in two groups $\widetilde{\mathbb{G}}$ and $\mathbb{G}$ are about 6.4 ms and 0.6 ms, respectively. One also can argue that in method 1, if we use RSA or Elliptic-curve cryptography (ECC) for the signature and encryption schemes instead of Pairing-based cryptography, this method will be efficient. We however note that Pairing-based signature schemes provide more advanced properties than RSA or Elliptic-curve-based signature schemes (that is why Pairing-based signature schemes currently still have deeply studied) such that short signature size, randomizable signature, aggregate signature, full security in the standard model, etc. Fortunately, our scheme is an extension of the Pointcheval-Sanders signature scheme [12] which supports all the above-advanced properties, therefore our scheme naturally also supports all the above-advanced properties. In addition, when using method 1, the security of the combined scheme (signature scheme and encryption scheme) should be carefully considered.

We also note that our SPPS scheme can be applied in any context that the privacy of the signer is needed.

Table 1. Performance comparison. Sig is a signature algorithm, Ver is a verification algorithm, Enc and Dec are encryption algorithm and decryption algorithm, respectively, while E denotes exponentiation operation, P denotes pairing operation, H denotes hash operation. $|p|$, $|p'|$, $|\mathbb{G}|$ and $|\widetilde{\mathbb{G}}|$ are the size of the prime p, p' and the groups $\mathbb{G}, \widetilde{\mathbb{G}}$. Note that if we set the security parameter $\lambda = 80$, then we can implement a curve of type 3 Pairing where $|p'|$, $|\mathbb{G}|$ and $|\widetilde{\mathbb{G}}|$ are about 170 bits, 170 bits and 1024 bits, while method 3 uses finite field where $|p|$ is much more bigger than 170 bits (about 1024 bits) when the security parameter $\lambda = 80$, see [13] for more details. Regarding method 2, we use BBS scheme [14], which is one of the most efficient group signature scheme to date.

	Signature Size	**Public-K Size**	**Secret-K Size**	**Sign Time**	**Verify Time**
M1	$\lvert Enc(Sig)\rvert$	$\lvert PK_{Enc} + PK_{Sig}\rvert$	$\lvert SK_{Enc} + SK_{Sig}\rvert$	$Sig + Enc$	$Dec + Ver$
M2—[14]	$9\lvert\mathbb{G}\rvert$	$4\lvert\mathbb{G}\rvert + 2\lvert\widetilde{\mathbb{G}}\rvert$	$1\lvert\mathbb{G}\rvert + 1\lvert p'\rvert$	$12E + 5P$	$10E + 10P$
M3—[10]	$5\lvert p\rvert$	$2\lvert p\rvert$	$2\lvert p\rvert$	$6E + 3H$	$9E + 1H$
Ours	$2\lvert\mathbb{G}\rvert$	$2\lvert\mathbb{G}\rvert + 3\lvert\widetilde{\mathbb{G}}\rvert$	$3\lvert p'\rvert$	$2E$	$2E + 3P$

1.2. Related Work and Organization of the Paper

The anonymity of the signer is also supported in attribute-based signature scheme, which was first introduced in [15]. In this system, each user possesses a set of attributes and then receives a corresponding secret key from the authority. To generate a signature on a given message m, the user relies on his/her secret key and a predicate which is satisfied by his/her attribute set. That means any user whose attribute set satisfies this predicate is able to generate this signature. The anonymity of the signer here is in the sense that from a given signature, anybody only knows that there are a set of users who are able to generate this signature, anybody (even the authority) cannot know exactly who actually generated this signature. This is obviously not suitable for the contexts of e-Voting, e-Health, e-Bidding or e-Payment applications, where the responsible person needs to know exactly who has generated this signature.

Another scheme, which also supports the anonymity of signer, is the *ring signature* scheme [16]. In this scheme, a user can generate a signature on behalf of a public set of users, and no one can identify who has actually generated this signature. Different from group signature, in the ring signature scheme we do not have the group manager to identify the signature.

The last one which supports the anonymity of signer is the anonymous signature scheme, this scheme was introduced by Yang et al. [17]. The anonymity of the signer is achieved in the sense that to run the verification procedure the verifier needs both the signer's public key and the signed message. If we hide the signed message, no one can identify who has actually generated the signature. This is also not suitable for the contexts of Electronic Voting, e-Health, e-Bidding or e-Payment applications.

PAPER ORGANIZATION. The next section introduces the definition and security model of our SPPS scheme and some used tools to construct our scheme. The construction of our SPPS scheme and its security analysis are introduced in Section 3. We give the conclusion of the paper in the last Section 4.

2. Preliminaries

In this section, we present the security model of our SPPS scheme, the assumptions which are used to prove the security of our scheme, and some useful tools used to construct our scheme.

2.1. Definition and Security Model

2.1.1. Definition

As in the usual signature scheme, a user in our scheme possesses a pair of the public key and secret key, as well as plays the roles of both the signer and the (designated) verifier. To generate the usual signature, the signer uses only his/her secret key as usual, however, to generate the designated verifier signature he/she uses both his/her secret key and the public key of the chosen designated verifier. To verify the usual signature, the verifier uses only the public key of the signer, however, to verify the designated verifier signature the designated verifier uses both the public key of the signer and his/her secret key. Moreover, in case the signer refuses to admit his/her designated verifier signature, the designated verifier uses his/her secret key to prove that the signer has actually generated this designated verifier signature.

Formally, there are seven probabilistic algorithms in our SPPS scheme.

- $\mathsf{Setup}(1^\lambda)$: the input of this algorithm is the security parameter λ, the output is the system parameters param.
- Key Generation: the input of this algorithm is the param, the output is a pair consisting of a public key and secret key (pk_i, sk_i).
- Usual Signing: the input of this algorithm are a message m, the param and a secret key sk_i, the output is a usual signature σ.

- Designated Signing: the input of this algorithm are a message m, the param, a secret key sk_i and the public key pk_j of chosen designated verifier. The output is a designated verifier signature σ.
- Usual Verification: the input of this algorithm are the param, a usual signature σ and the public key pk_i of the signer. The algorithm returns 1 in case σ is a valid signature of the message m under pk_i, and 0 otherwise.
- Designated Verification: the input of this algorithm are the param, a designated verifier signature σ, the public key pk_i of the signer and the secret key sk_j of the designated verifier. The algorithm returns 1 in case σ is a valid signature of the message m under pk_i, and 0 otherwise.
- Proving: this algorithm includes two sub-algorithms:
  - first: takes as input param, a designated verifier signature σ on a message m and a designated verifier's secret key sk_j, outputs a public proof proof;
  - second: takes as input param, a public proof proof and the corresponding signer's public key pk_i. The algorithm returns 1 in the case where σ is a valid signature of the message m under pk_i, and 0 otherwise. Note that the second sub-algorithm is run by anyone.

Correctness: If all $m, pk_i, sk_i, pk_j, sk_j$ are valid, and $\sigma = \text{Usual Signing}(\text{param}, m, sk_i)$ or $\sigma = \text{Designated Signing}(\text{param}, m, sk_i, pk_j)$, then the following equations must hold:

$$\text{Usual Verification}(\text{param}, \sigma, m, pk_i) = 1$$

$$\text{Designated Verification}(\text{param}, \sigma, m, pk_i, sk_j) = 1$$

$$\text{Proving}(\text{param}, sk_j, \sigma, m, pk_i) = 1$$

2.1.2. Adversary's Oracles

We allow the forger $\mathcal{F}$ to query the challenger $\mathcal{C}$ the following oracles.

- UsualSigningRequest(m, pk_i): when $\mathcal{F}$ requests a usual signature of user i on a message m, the challenger $\mathcal{C}$ returns a valid corresponding usual signature σ.
- DesignatedSigningRequest(m, pk_i, pk_j): when $\mathcal{F}$ requests a designated verifier signature of user i on a message m with designated verifier public key pk_j, the challenger $\mathcal{C}$ returns a valid corresponding designated verifier signature σ.
- RequestDesignatedVerifying(σ, m, pk_i, pk_j): when $\mathcal{F}$ requests to know the validity of a designated verifier signature σ, the challenger $\mathcal{C}$ returns the validity of σ.

2.1.3. Security Model

The security requirements of our SPPS scheme are that an attacker is unable to forge the usual signature as well as the designated verifier signature. Moreover, he/she also cannot verify any designated verifier signature. To this aim, based on security models in [7], we consider the unforgeability property which guarantees that the attacker cannot forge any signature, and the privacy of signer's identity (PSI) property which guarantees that the attacker cannot verify any designated verifier signature.

Unforgeability: Regarding this property, we allow the colluding of any user. The security game is described as follows.

Initialization Phase: At this step $\mathcal{C}$ first uses the Setup algorithm to produce the public parameters param. Next, he/she uses the Key Generation algorithm to produce the target user's key pair (pk_i, sk_i) and the target designated verifier's key pair (pk_j, sk_j). $\mathcal{C}$ sends param and (pk_i, pk_j, sk_j) to $\mathcal{F}$.

Query Phase: At this step, $\mathcal{F}$ adaptively asks the following oracles: UsualSigningRequest(m, pk_i), DesignatedSigningRequest(m, pk_i, pk_j) and RequestDesignatedVerifying(σ, m, pk_i, pk_j).

Output Phase: At this step, $\mathcal{F}$ outputs (m^*, σ^*). $\mathcal{F}$ is said to win the game if the followings are correct:

1. m^* has never been queried to UsualSigningRequest(m, pk_i) and DesignatedSigningRequest(m, pk_i, pk_j);
2. Usual Verification(param, σ^*, m^*, pk_i) outputs 1, or Designated Verification(param, $\sigma^*, m^*, pk_i, pk_j$) outputs 1.

Let $SuccUn_{\mathcal{F}}^{\Pi}$ be the success probability that the forger $\mathcal{F}$ wins the above security game.

Definition 1. *A* SPPS *scheme* Π *is said to achieve the existentially unforgeable property against any polynomial-time forger* $\mathcal{F}$ *if the success probabilities* $SuccUn_{\mathcal{F}}^{\Pi}$ *above is negligible.*

Privacy of the signer's identity—PSI: This property is defined in the sense that an attacker cannot distinguish between a valid designated verifier signature and random elements, that means he/she cannot check the validity of a designated verifier signature.

The security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{F}$ is described as follows.

Initialization Phase: At this step $\mathcal{C}$ first uses the Setup algorithm to produce the public parameters param. Next, he/she uses the Key Generation algorithm to produce three target users' key pairs $(pk_{i_0}, sk_{i_0}), (pk_{i_1}, sk_{i_1}), (pk_j, sk_j)$, where i_0 and i_1 are target signers, j is the target designated verifier. $\mathcal{C}$ sends param and $(pk_{i_0}, pk_{i_1}, pk_j)$ to $\mathcal{F}$.

Query Phase: At this step, $\mathcal{F}$ adaptively asks the following oracles: UsualSigningRequest(m, pk_{i_k}), DesignatedSigningRequest(m, pk_{i_k}, pk_j) and RequestDesignatedVerifying$(\sigma, m, pk_{i_k}, pk_j)$, $k \in \{0, 1\}$.

Challenge Phase: At this step, $\mathcal{F}$ first outputs m^*, $\mathcal{C}$ then randomly picks $b \overset{\$}{\leftarrow} \{0, 1\}$ and produces $\sigma^* =$ Designated Signing(param, m^*, sk_{i_b}, pk_j), then sends σ^* to $\mathcal{F}$.

$\mathcal{F}$ still can request queries as above except that he/she cannot request query RequestDesignatedVerifying$(\sigma^*, m^*, pk_{i_k}, pk_j)$ for any $k \in \{0, 1\}$.

Guess Phase: At this step, $\mathcal{F}$ outputs a guess bit b' for b, $\mathcal{F}$ is said to win the game if $b' = b$.

Let $AdvPSI_{\mathcal{F}}^{\Pi}$ be the advantage that $\mathcal{F}$ wins the above security game, where $AdvPSI_{\mathcal{F}}^{\Pi}$ is defined by the success probability that $\mathcal{F}$ wins the above security game deviates from one-half.

Definition 2. *A* SPPS *scheme* Π *is said to achieve the* PSI *property against any polynomial-time adversary* $\mathcal{F}$ *if the advantage* $AdvPSI_{\mathcal{F}}^{\Pi}$ *above is negligible.*

2.2. Bilinear Groups

Let $\mathbb{G}$, $\widetilde{\mathbb{G}}$ and $\mathbb{G}_T$ be three finite multiplicative abelian groups of large prime order $p > 2^{\lambda}$ where λ is the security parameter. Let $g, \tilde{g}$ be generators of $\mathbb{G}, \widetilde{\mathbb{G}}$, respectively. Let e be an admissible asymmetric bilinear map $e : \mathbb{G} \times \widetilde{\mathbb{G}} \to \mathbb{G}_T$ such that for all $a, b \in \mathbb{Z}_p$

1. $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{ab}$;
2. for $g \neq 1_{\mathbb{G}}$ and $\tilde{g} \neq 1_{\widetilde{\mathbb{G}}}$, $e(g, \tilde{g}) \neq 1_T$;
3. $e(g, \tilde{g})$ is efficiently computable.

We call the tuple $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, g, \tilde{g}, e)$ a bilinear map group system, and according to [13] it is in:

1. Type 1 pairings if $\mathbb{G} = \widetilde{\mathbb{G}}$
2. Type 2 pairings if $\mathbb{G} \neq \widetilde{\mathbb{G}}$ but there is an efficiently computable homomorphism $\phi : \widetilde{\mathbb{G}} \to \mathbb{G}$
3. Type 3 pairings if $\mathbb{G} \neq \widetilde{\mathbb{G}}$ but there doesn't exist efficiently computable homomorphism between $\widetilde{\mathbb{G}}$ and $\mathbb{G}$.

2.3. Pointcheval-Sanders Signature Scheme

At CT-RSA'16, Pointcheval et al. proposed a new signature scheme, this scheme achieves full security under a new assumption which they named PS assumption 1. In the recent paper at CT-RSA'18 [12], the author showed that the hardness of this new assumption and the hardness of the well-known q-SDH assumption [18] are equivalent.

More precisely, the Pointcheval–Sanders signature scheme includes four following probabilistic algorithms.

- Setup: the input of this algorithm is the security parameter λ, the output is the public parameter param $= (p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, g, \tilde{g}, e)$.
- KeyGen: the input of this algorithm is the security parameter λ, the output is the user's key pairs. Concretely, user's secret key includes two elements $(x, y) \in \mathbb{Z}_p^*$, and user's public key includes three elements $(\tilde{h} \in \widetilde{\mathbb{G}}, \tilde{X} = \tilde{h}^x, \tilde{Y} = \tilde{h}^y)$.
- Sign: the inputs of this algorithm are the user's secret key (x, y) and a message $m \in \mathbb{Z}_p$. The algorithm randomly picks $h \overset{\$}{\leftarrow} \mathbb{G}, h \neq 1_{\mathbb{G}}$, and generates the signature $\sigma = (\sigma_1 = h, \sigma_2 = h^{(x+ym)})$.
- Verify: the input of this algorithm are a message m, signature $\sigma = (\sigma_1, \sigma_2)$ and the corresponding public key $(\tilde{h}, \tilde{X}, \tilde{Y})$. The algorithm returns 1 which indicates that the signature is valid if the following conditions are verified:

$$\begin{aligned} \sigma_1 &\neq 1_{\mathbb{G}}; \\ e(\sigma_1, \tilde{X}\tilde{Y}^m) &= e(\sigma_2, \tilde{h}). \end{aligned}$$

  Otherwise, the algorithm returns 0 which indicates that the signature is invalid.

The PS assumption 1, given in [12], permits to prove that such signature scheme is secure.

Definition 3. ***PS assumption 1:*** *Let $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, g, \tilde{g}, e)$ be a bilinear group setting of type 3. Choose $u, v \overset{\$}{\leftarrow} \mathbb{Z}_p$, we define the oracle $\mathcal{O}(m)$ on input $m \in \mathbb{Z}_p$ that chooses a random $h \in \mathbb{G}$ and outputs the pair (h, h^{u+mv}). Given $(g, \tilde{g}, \tilde{g}^u, \tilde{g}^v, g^v)$ and unlimited access to this oracle, no adversary can have non-negligible success probability to generate a pair (h, h^{u+m^*v}), with $h \neq 1_{\mathbb{G}}$, for a new m^* not asked to $\mathcal{O}$.*

In this paper, we prove that our SPPS scheme is secure under the PS assumption 1 and a new assumption which is a simple modification of the well known XDH assumption. The XDH assumption is described as follows.

Definition 4. **XDH** ***assumption****: Assume that $(\boldsymbol{p}, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_{\boldsymbol{T}}, \boldsymbol{g}, \tilde{\boldsymbol{g}}, \boldsymbol{e})$ is a bilinear group system of type 3. Pick $\boldsymbol{a}, \boldsymbol{b} \overset{\$}{\leftarrow} \mathbb{Z}_p$. Given $(\boldsymbol{g}, \tilde{\boldsymbol{g}}, \boldsymbol{g}^{\boldsymbol{a}}, \boldsymbol{g}^{\boldsymbol{b}}, \boldsymbol{T})$, it is hard to distinguish $\boldsymbol{T}$ is either $\boldsymbol{g}^{\boldsymbol{ab}}$ or a random element in $\mathbb{G}$.*

We define a simple modification of XDH assumption, we name MXDH assumption, as follows.

Definition 5. **MXDH** ***assumption****: Assume that $(\boldsymbol{p}, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_{\boldsymbol{T}}, \boldsymbol{g}, \tilde{\boldsymbol{g}}, \boldsymbol{e})$ is a bilinear group system of type 3. Pick $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \overset{\$}{\leftarrow} \mathbb{Z}_p$ and $\tilde{\boldsymbol{h}} \overset{\$}{\leftarrow} \widetilde{\mathbb{G}}$. Given $\overrightarrow{\boldsymbol{Y}} = (\boldsymbol{g}, \tilde{\boldsymbol{g}}, \boldsymbol{g}^{\boldsymbol{a}}, \boldsymbol{g}^{\boldsymbol{b}}, \boldsymbol{g}^{\boldsymbol{c}}, \boldsymbol{g}^{\boldsymbol{ac}}, \tilde{\boldsymbol{h}}, \tilde{\boldsymbol{h}}^{\boldsymbol{a}}, \boldsymbol{T})$, it is hard to distinguish $\boldsymbol{T}$ is either $\boldsymbol{g}^{\boldsymbol{abc}}$ or a random element in $\mathbb{G}$.*

It is easy to see that one needs to have the value $\tilde{g}^b$ or g^{bc} to distinguish T. However, these values are not provided in $\overrightarrow{Y}$.

3. SPPS Scheme

We rely on the Pointcheval–Sanders signature scheme [12] to construct our SPPS scheme. In our SPPS scheme a user can choose to generate either a usual signature (using the same signing algorithm as in [12]) or a designated verifier signature.

3.1. Detailed Description

The scheme is described as follows.

Setup(1^λ): the input of this algorithm is the security parameter λ, the output is the public parameters param $= (p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, g, \tilde{g}, e)$.

Key Generation: the input of this algorithm is param. To produce the output, the algorithm first randomly chooses $(x_i, y_i, z_i) \xleftarrow{\$} \mathbb{Z}_p^*, \tilde{h}_i \xleftarrow{\$} \widetilde{\mathbb{G}}$, computes $\tilde{X}_i = \tilde{h}_i^{x_i}, \tilde{Y}_i = \tilde{h}_i^{y_i}, Y_i = g^{y_i}, Z_i = g^{z_i}$. The algorithm finally outputs the secret key $sk_i = (x_i, y_i, z_i)$ and public key $pk_i = (\tilde{h}_i, \tilde{X}_i, \tilde{Y}_i, Y_i, Z_i)$

Usual Signing: the input of this algorithm are param, sk_i and the message $m \in \mathbb{Z}_p$ (note that in practice we use a hash function to hash a long message $m \in \{0,1\}^*$ to a short message $m \in \mathbb{Z}_p$). The algorithm first randomly chooses $h \xleftarrow{\$} \mathbb{G}$ such that $h \neq 1_{\mathbb{G}}$, then outputs the usual signature $\sigma = (\sigma_1, \sigma_2)$ where $\sigma_1 = h$ and $\sigma_2 = h^{(x_i + y_i m)}$.

Designated Signing: the input of this algorithm are param, sk_i, pk_j and the message $m \in \mathbb{Z}_p$. The algorithm first randomly chooses $h \xleftarrow{\$} \mathbb{G}$ such that $h \neq 1_{\mathbb{G}}$, then outputs the designated verifier signature $\sigma = (\sigma_1, \sigma_2)$ where $\sigma_1 = h$ and $\sigma_2 = h^{(x_i + y_i m)} \cdot (g^{z_j})^{y_i z_i}$.

Usual Verification: the input of this algorithm are param, pk_i, m and the usual signature $\sigma = (\sigma_1, \sigma_2)$. The algorithm checks that whether $\sigma_1 \neq 1_{\mathbb{G}}$ and

$$e(\sigma_2, \tilde{h}_i) = e(\sigma_1, \tilde{X}_i \tilde{Y}_i^m).$$

If this is the case the algorithm outputs 1, else it outputs 0.

Designated Verification: the input of this algorithm are param, pk_i, sk_j, m and the designated verifier signature $\sigma = (\sigma_1, \sigma_2)$. The algorithm checks that whether $\sigma_1 \neq 1_{\mathbb{G}}$ and

$$e(\sigma_2, \tilde{h}_i) = e(\sigma_1, \tilde{X}_i \tilde{Y}_i^m) \cdot e(Z_i, \tilde{Y}_i^{z_j}).$$

If this is the case the algorithm outputs 1, else it outputs 0.

Proving:

- first: the designated verifier j with his/her secret key $sk_j = (x_j, y_j, z_j)$ and the signature $\sigma = (\sigma_1, \sigma_2)$ of user i at hands generates the proof as follows:

$$\text{proof} = (\sigma_1', \sigma_2') = (\sigma_1^{\frac{1}{z_j}}, \sigma_2^{\frac{1}{z_j}}) = (h^{\frac{1}{z_j}}, h^{\frac{x_i + y_i m}{z_j}} \cdot g^{y_i z_i}).$$

- second: on input a message m, anybody with the proof at hands can verify whether or not that the user i with his/her public key pk_i has signed on the message m by checking that:

$$\begin{aligned} \sigma_1' &\neq 1_{\mathbb{G}}; \\ e(\sigma_2', \tilde{h}_i) &= e(\sigma_1', \tilde{X}_i \tilde{Y}_i^m) \cdot e(Z_i, \tilde{Y}_i). \end{aligned}$$

Correctness. The correctness of Usual Verification algorithm is straightforward followed from Pointcheval-Sanders signature scheme [12].

Regarding the correctness of Designated Verification, let $\sigma = (\sigma_1 = h, \sigma_2 = h^{(x_i+y_i m)} \cdot (g^{z_j})^{y_i z_i})$, we have:

$$\begin{aligned} e(\sigma_2, \tilde{h}_i) &= e(h^{(x_i+y_im)} \cdot (g^{z_j})^{y_iz_i}, \tilde{h}_i) \\ &= e(h^{(x_i+y_im)}, \tilde{h}_i) \cdot e(g^{z_jy_iz_i}, \tilde{h}_i) \\ &= e(h, \tilde{h}_i^{(x_i+y_im)}) \cdot e(g^{z_i}, \tilde{h}_i^{y_iz_j}) \\ &= e(\sigma_1, \tilde{X}_i\tilde{Y}_i^m) \cdot e(Z_i, \tilde{Y}_i^{z_j}), \end{aligned}$$

and similar to the correctness of the Proving algorithm.

Remark 1. *It is clear that from the* proof*, one cannot extract z_j, which means the designated verifier can runs the* Proving *algorithm without revealing his/her secret key. However, the crucial point here is that the signer realizes that he/she cannot refuse his/her designated verifier signature, therefore in practice the designated verifier may never need to runs the* Proving *algorithm.*

3.2. Security Analysis of SPPS *Scheme*

In this section, we prove that our SPPS scheme is existentially unforgeable under the PS assumption 1, and our SPPS scheme achieves PSI property under the MXDH assumption.

Theorem 1. *Our* SPPS *scheme is existentially unforgeable under the* PS *assumption 1.*

Proof. Let $\mathcal{F}$ be a forger against the security of our SPPS scheme, and $\mathcal{C}$ be an adversary against PS assumption 1. We will prove that $\mathcal{C}$ can simulate $\mathcal{F}$ and based on the $\mathcal{F}$'s output to solve the PS assumption 1.

To this aim, at the beginning of the security game (see Section 2.1.3) $\mathcal{C}$ first gets an instance of the PS assumption 1: $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, g, \tilde{g}, e, g^v, \tilde{g}^u, \tilde{g}^v)$, and gets the right to access oracle $\mathcal{O}$. Note that $\mathcal{O}(m)$, on input $m \in \mathbb{Z}_p$, outputs a pair (h, h^{u+vm}) where h is a random element in $\mathbb{G}$.

Initialization Phase: $\mathcal{C}$ chooses $t, z_i, x_j, y_j, z_j \xleftarrow{\$} \mathbb{Z}_p^*$, implicitly sets $x_i = u, y_i = v$, then computes $\tilde{h}_i = \tilde{g}^t, \tilde{X}_i = (\tilde{g}^u)^t, \tilde{Y}_i = (\tilde{g}^v)^t, Y_i = g^v, Z_i = g^{z_i}$, sets $pk_i = (\tilde{h}_i, \tilde{X}_i, \tilde{Y}_i, Y_i, Z_i)$. Next, $\mathcal{C}$ chooses $\tilde{h}_j \xleftarrow{\$} \widetilde{\mathbb{G}}$, computes $\tilde{X}_j = \tilde{h}_j^{x_j}, \tilde{Y}_i = \tilde{h}_j^{y_j}, Y_j = g^{y_j}, Z_j = g^{z_j}$, sets $pk_j = (\tilde{h}_j, \tilde{X}_j, \tilde{Y}_j, Y_j, Z_j)$ and $sk_j = (x_j, y_j, z_j)$.

Finally, $\mathcal{C}$ gives param $= (p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, g, \tilde{g}, e)$ and (pk_i, pk_j, sk_j) to $\mathcal{F}$.

Query Phase: $\mathcal{C}$ now answers the requested oracles from $\mathcal{F}$ as follows.

- UsualSigningRequest(m, pk_i): $\mathcal{C}$ needs to answer to $\mathcal{F}$ the usual signature of user i on message m. To this aim, $\mathcal{C}$ simply asks $\mathcal{O}$ on input m to obtain the pair $(h, h^{u+vm}) = (h, h^{x_i+y_im})$, then returns it to $\mathcal{F}$.
- DesignatedSigningRequest(m, pk_i, pk_j): $\mathcal{C}$ needs to answer to $\mathcal{F}$ the designated verifier signature of user i on message m with designated verifier j. To this aim, $\mathcal{C}$ first requests the oracle $\mathcal{O}$ on input m to get the pair $(h, h^{u+vm}) = (h, h^{x_i+y_im})$. Next, $\mathcal{C}$ produces designated verifier signature $\sigma = (\sigma_1, \sigma_2)$ as follows.

$$\sigma_1 = h, \sigma_2 = h^{x_i+y_im} \cdot (g^{y_i})^{z_jz_i}$$

  Note that $\mathcal{C}$ knows z_i, z_j. Finally, $\mathcal{C}$ sends σ to $\mathcal{F}$.
- RequestDesignatedVerifying(σ, m, pk_i, pk_j): $\mathcal{C}$ needs to answer to $\mathcal{F}$ that whether or not σ is a valid designated verifier signature on m with designated verifier j. To this aim, $\mathcal{C}$ simply runs the Designated Verification(param, σ, m, pk_i, sk_j) and returns the output to $\mathcal{F}$.

Output Phase: At this phase, $\mathcal{F}$ outputs (m^*, σ^*) with the requirement that m^* has never been queried to $\mathsf{UsualSigningRequest}(m^*, pk_i)$ and $\mathsf{DesignatedSigningRequest}(m^*, pk_i, pk_j)$, that means m^* has never been queried to oracle $\mathcal{O}$. There are two cases: First, σ^* is a usual signature that means $\sigma^* = (\sigma_1^*, \sigma_2^*)$, where:

$$\sigma_1^* = h', \sigma_2^* = h'^{x_i + y_i m^*} = h'^{u+vm^*}$$

$\mathcal{C}$ simply uses the pair (σ_1^*, σ_2^*) as a valid pair to solve the PS assumption 1.

Second, σ^* is a designated verifier signature that means $\sigma^* = (\sigma_1^*, \sigma_2^*)$, where:

$$\sigma_1^* = h', \sigma_2^* = h'^{x_i + y_i m^*} \cdot g^{y_i z_j z_i}$$

$\mathcal{C}$ computes

$$K = \sigma_2^* / (g^{y_i})^{z_j z_i} = h'^{x_i + y_i m^*} = h'^{u+vm^*}.$$

Note that $\mathcal{C}$ knows z_i, z_j. So, $\mathcal{C}$ also can use the valid pair (σ_1^*, K) to solve the PS assumption 1.

It is easy to see that the simulation is perfect and $\mathcal{C}$ has never aborted the game, so if $\mathcal{F}$ can break the security of our SPPS scheme with non-negligible success probability, then $\mathcal{C}$ also can solve the PS assumption 1 with non-negligible success probability ($SuccUn_{\mathcal{F}}^{\Pi}$ is non-negligible), which concludes our proof.

□

Theorem 2. *Our* SPPS *scheme achieves* PSI *property under the* MXDH *assumption. More precisely,*

$$AdvPSI_{\mathcal{F}}^{\Pi} \leq 2\epsilon_{\mathsf{MXDH}}$$

Proof. Let $\mathcal{F}$ as above and $\mathcal{C}$ be an adversary against MXDH assumption. The proof includes a series of games, let E_i be the event that the adversary $\mathcal{F}$ returns the correct guessed bit in Game G_i.

- G_0: Game G_0 is the original game. In this game, $\mathcal{C}$ first chooses $x_{i_k}, y_{i_k}, z_{i_k}, x_j, y_j, z_j \xleftarrow{\$} \mathbb{Z}_p^*$ where $k = 0, 1$, so $\mathcal{C}$ can easily produce $\mathsf{param}, pk_{i_k}, pk_j$, and $\mathcal{C}$ sends $\mathsf{param}, pk_{i_k}, pk_j$ to $\mathcal{F}$.

  On the other hand, with $x_{i_k}, y_{i_k}, z_{i_k}, x_j, y_j, z_j$ at hands where $k = 0, 1$, $\mathcal{C}$ can easily answer any query ($\mathsf{UsualSigningRequest}(m, pk_{i_k})$, $\mathsf{DesignatedSigningRequest}(m, pk_{i_k}, pk_j)$ and $\mathsf{RequestDesignatedVerifying}(\sigma, m, pk_{i_k}, pk_j)$) from $\mathcal{F}$.

  Next, at the challenge phase, $\mathcal{C}$ first picks randomly a bit $\mathsf{b} \xleftarrow{\$} \{0, 1\}$, $\mathcal{C}$ then uses $x_{i_\mathsf{b}}, y_{i_\mathsf{b}}, z_{i_\mathsf{b}}, z_j$ to produce the challenge designated verifier signature σ^*. From the definition, we have: $Pr|E_0| = AdvPSI_{\mathcal{F}}^{\Pi} + \frac{1}{2}$
- G_1: This game is the same as game G_0 except that the value $g^{y_{i_0} z_{i_0} z_j}$ now is generated randomly.

  Let consider the following game: $\mathcal{C}$ first gets an instance of the MXDH assumption: $(g, \tilde{g}, g^a, g^b, g^c, g^{ac}, \tilde{h}, \tilde{h}^a, T)$, note that T is either g^{abc} or a random element in $\mathbb{G}$. $\mathcal{C}$ next randomly picks $x_{i_0}, x_{i_1}, y_{i_1}, z_{i_1} \xleftarrow{\$} \mathbb{Z}_p^*$, then uses them to produce param, pk_{i_1}.

  For pk_{i_0} and pk_j, $\mathcal{C}$ first implicitly sets $y_{i_0} = a, z_{i_0} = c, z_j = b$, randomly picks $x_j, y_j \xleftarrow{\$} \mathbb{Z}_p^*$, then uses them to produce: pk_j (note that $\mathcal{C}$ knows $Z_j = g^b$ from assumption). $\mathcal{C}$ also produces pk_{i_0} as follows:

  $$pk_{i_0} = (\tilde{h}, \tilde{X}_{i_0} = \tilde{h}^{x_{i_0}}, \tilde{Y}_{i_0} = \tilde{h}^a, Y_{i_0} = g^a, Z_{i_0} = g^c)$$

  Note that $\tilde{h}_{i_0} = \tilde{h}$, $\mathcal{C}$ finally sends $\mathsf{param}, pk_{i_k}, pk_j$ where $k = 0, 1$ to $\mathcal{F}$.

  At the first query phase, $\mathcal{C}$ answers queries from $\mathcal{F}$ as follows.

– $\mathsf{UsualSigningRequest}(m,\ pk_{i_0})$: $\mathcal{C}$ simply picks $t \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, then produces $\sigma = (\sigma_1, \sigma_2)$ where

$$\sigma_1 = g^t, \sigma_2 = g^{tx_{i_0}} \cdot (g^a)^{tm}$$

$\mathcal{C}$ also can easily answer $\mathsf{UsualSigningRequest}(m,\ pk_{i_1})$ since he/she knows $x_{i_1}, y_{i_1}, z_{i_1}$.

– $\mathsf{DesignatedSigningRequest}(m,\ pk_{i_0}, pk_j)$: $\mathcal{C}$ first picks $t \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, then produces $\sigma = (\sigma_1, \sigma_2)$ where

$$\sigma_1 = g^t, \sigma_2 = g^{tx_{i_0}} \cdot (g^a)^{tm} \cdot T$$

Note that if $T = g^{abc}$ then σ is in the correct form. As above, since $\mathcal{C}$ knows $x_{i_1}, y_{i_1}, z_{i_1}$, so he/she can easily answer $\mathsf{DesignatedSigningRequest}(m,\ pk_{i_1}, pk_j)$, note that $\mathcal{C}$ also knows $Z_j = g^b$.

$$\sigma_1 = g^t, \sigma_2 = g^{tx_{i_1}} \cdot (g^{y_{i_1}})^{tm} \cdot (g^b)^{y_{i_1} z_{i_1}}$$

– $\mathsf{RequestDesignatedVerifying}(\sigma = (\sigma_1, \sigma_2), m, pk_{i_0}, pk_j)$: $\mathcal{C}$ simply checks whether:

$$e(\sigma_1, \tilde{X}_{i_0} \cdot \tilde{Y}_{i_0}^m) \cdot e(T, \tilde{h}) == e(\sigma_2, \tilde{h}).$$

If it is right, $\mathcal{C}$ outputs 1, else $\mathcal{C}$ outputs 0. Note that $\tilde{h}_{i_0} = \tilde{h}$. It is also easy for $\mathcal{C}$ to answer $\mathsf{RequestDesignatedVerifying}(\sigma = (\sigma_1, \sigma_2), m, pk_{i_1}, pk_j)$.

At the challenge phase, $\mathcal{F}$ first sends m^* to $\mathcal{C}$, $\mathcal{C}$ then picks $t \overset{\$}{\leftarrow} \mathbb{Z}_p^*, \mathsf{b} \overset{\$}{\leftarrow} \{0,1\}$, then produces $\sigma^* = (\sigma_1^*, \sigma_2^*)$, where:

$$\sigma_1^* = g^t, \sigma_2^* = g^{tx_{i_\mathsf{b}}} \cdot Y_{i_\mathsf{b}}^{tm^*} \cdot T_\mathsf{b}.$$

Note that if $\mathsf{b} = 0$, $T_\mathsf{b} = T$, else $T_\mathsf{b} = g^{by_{i_1} z_{i_1}}$. We have that if $T = g^{abc}$, σ^* is in the correct form.

The second phase query is similar to the first one.

At the guess phase, $\mathcal{F}$ outputs b' as the guess for bit b, if $b' = \mathsf{b}$ then $\mathcal{C}$ returns 1 means that $T = g^{abc}$ and 0 otherwise, means that T is a random element in $\mathbb{G}$.

It is straightforward to realize that if $T = g^{abc}$ then $\mathcal{C}$ has simulated the game which is identical to Game G_0, and the probability that $\mathcal{C}$ outputs the correct answer is $Pr|E_0|$. In case T is a random element, $\mathcal{C}$ has simulated the game which is identical to Game G_1, and the probability that $\mathcal{C}$ outputs the correct answer is $Pr|E_1|$. Let β be the bit outputted by $\mathcal{C}$, we have: $\left| Pr[\beta = 1 | T = g^{abc}] - Pr[\beta = 1 | T \overset{\$}{\leftarrow} \mathbb{G}] \right| = |Pr|E_0| - Pr|E_1||$, therefore:

$|Pr|E_0| - Pr|E_1|| \le \epsilon_{\mathsf{MXDH}}$

- G_2: This game is the same as game G_1 except that $g^{y_{i_1} z_{i_1} z_j}$ now is generated randomly.

  As in the game G_1, we have: $|Pr|E_1| - Pr|E_2|| \le \epsilon_{\mathsf{MXDH}}$.

  Note that both $g^{y_{i_0} z_{i_0} z_j}$ and $g^{y_{i_1} z_{i_1} z_j}$ are generated randomly in game G_2, therefore the queries as well as the challenge designated verifier signature provide no help for adversary $\mathcal{F}$, therefore $Pr|E_2| = \frac{1}{2}$. That means, we have: $|Pr|E_0| - Pr|E_1|| + |Pr|E_1| - Pr|E_2|| \le 2\epsilon_{\mathsf{MXDH}}$, $Pr|E_0| \le 2\epsilon_{\mathsf{MXDH}} + Pr|E_2| = 2\epsilon_{\mathsf{MXDH}} + \frac{1}{2}$

  Thus $AdvPSI_{\mathcal{F}}^{\Pi} \le 2\epsilon_{\mathsf{MXDH}}$, which concludes our proof.

□

4. Conclusions

In several modern applications such as Electronic Voting, e-Health, e-Payment, etc, the privacy information of the signer is needed. There exist several methods to keep the privacy information of the signer such as combining a signature scheme with an encryption scheme, using a strong designated verifier signature scheme, or using a group signature scheme. In this paper, we extend the traditional signature scheme to propose an alternative method to solve this problem. Our proposed method has two advantages: it is an improvement of the traditional signature scheme in term of functionality (adding the function of keeping privacy information of the signer); it is the most efficient one when compared with three existing methods which deal with the same problem of keeping the privacy information of the signer.

Author Contributions: Methodology: V.C.T.; writing—original draft: V.C.T. and D.H.D.; formal analysis, writing—review and editing: W.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2018.301.

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

References

1. Chaum, D.; van Heyst, E. Group Signatures. In *Advances in Cryptology EUROCRYPT'91, vol. 547 of Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 1991; pp. 257–265.
2. Jakobsson, M.; Sako, K.; Impagliazzo, R. Designated verifier proofs and their applications. In *Advances in Cryptology EUROCRYPT'96, vol. 1070 of Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 143–154.
3. Ming, Y.; Jin, Q.; Zhao, X. Designated verifier proxy signature scheme with multi-warrant in the standard model. *J. Inf. Comput. Sci.* **2013**, *10*, 2097–2107. [CrossRef]
4. Lin, H.Y.; Wu, T.S.; Huang, S.K. An eicient strong designated verifier proxy signature scheme for electronic commerce. *J. Inf. Sci. Eng.* **2012**, *28*, 771–785.
5. Huang, Q.; Yang, G.; Wong, D.S.; Susilo, W. Identity-based strong designated verifier signature revisited. *J. Syst. Softw.* **2011**, *84*, 120–129. [CrossRef]
6. Lin, H.Y.; Wu, C.H.; Jiang, Y.R. On Delegatability of a Certificateless Strong Designated Verifier Signature Scheme. In *New Trends in Computer Technologies and Applications. ICS 2018. Communications in Computer and Information Science*; Springer: Singapore, 2019; Volume 1013, ISBN 978-981-13-9190-3.
7. Huang, Q.; Yang, G.; Wong, D.S.; Susilo, W. Efficient strong designated verifier signature schemes without random oracle or with non-delegatability. *Int. J. Inf. Secur.* **2011**. [CrossRef]
8. Yang, B.; Sun, Y.; Yu, Y.; Xia, Q. A strong designated verifier signature scheme with secure disavowability. In Proceedings of the 4th International Conference on Intelligent Networking and Collaborative Systems (INCoS'12), Bucharest, Romania, 19–21 September 2012; pp. 286–291.
9. Yang, B.; Yu, Y.; Sun, Y. A novel construction of SDVS with secure disavowability. *Cluster Comput.* **2013**, *16*, 807–815. [CrossRef]
10. Hu, X.; Tan, W.; Xu, H.; Wang, J.; Ma, C. Strong Designated Verifier Signature Schemes with Undeniable Property and Their Applications. *Secur. Commun. Netw.* **2017**, *2017*, 7921782. [CrossRef]
11. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-Policy Attribute-Based Encryption. In Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 20–23 May 2007; pp. 321–334.
12. Pointcheval, D.; Sanders, O. Reassessing Security of Randomizable Signatures. In Proceedings of the Topics in Cryptology—CT-RSA 2018—The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, 16–20 April 2018.
13. Galbraith, S.D.; Paterson, K.G.; Smart, N.P. Pairings for cryptographers. *Discrete Appl. Math.* **2008**, *156*, 3113–3121. [CrossRef]
14. Boneh, D.; Boyen, X.; Shacham, H. Short Group Signatures. In *Advances in CRYPTO'04, vol. 3152 of Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 41–55.

15. Maji, H.K.; Prabhakaran, M.; Rosulek, M. Attribute-based signatures. In *CT-RSA'11*; Springer: Berlin/Heidelberg, Germany, 2011; LNCS 6558, pp. 376–392.
16. Rivest, R.L.; Shamir, A.; Tauman, Y. How to Leak a Secret. In *ASIACRYPT 2001*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 552–565.
17. Yang, G.; Wong, D.S.; Deng, X.; Wang, H. Anonymous Signature Schemes. *Public Key Cryptogr.* **2006**, *3958*, 347–363.
18. Boneh, D.; Boyen, X. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptol.* **2008**, *21*, 149–177. [CrossRef]