A Robust Secure Self-Certified Concurrent Signature Scheme from Bilinear Pairings

Chien-Hua Tsai[1] and Pin-Chang Su[2]
[1]Department of Accounting Information, Chihlee University of Technology, 220 Taiwan
[2]Department of Information Management, National Defense University, 112 Taiwan

Abstract: *The idea of concurrent signature schemes is that two parties produce two respective ambiguous signatures that are concurrently bound to their corresponding signatories only while either of the party releases a keystone. The main construct is that both parties need to reach a consensus on the true fairness in mutually exchanging the signatures, and, moreover, the protocols assume that there is no collusion between a trusted third party and any of the parties. However, by collaborating over business interests with the participants as strategic partners, the trusted third party may obtain access to sensitive key data held in escrow, leading them to the collusion attack associated with malicious intentions. To circumvent the misbehavior among the participating individuals, an identity authentication process can be used prior to exchanging or having access to any confidential information. In this paper, we propose a self-certified concurrent signature from bilinear pairings as an alternative solution to strengthen the security level for solving the fair exchange problem. Apart from resisting to the collusion attack, the proposed scheme provides the advanced security properties to prevent from the message substitution, the identity forgery and impersonation, and other generic attacks in an increasingly insecure network environment.*

Keywords: *Bilinear pairings, concurrent signature, fair exchange, self-certified, trusted third party.*

Received December 26, 2019; accepted February 21, 2021
https://doi.org/10.34028/18/4/6

1. Introduction

The Internet provides an important business platform for trade marketing, distribution sales and financial transactions between marketers, consumers and businesses. As this technology continues to evolve and has fueled the growth of e-business applications, it therefore has raised some security concerns, such as identity management, mutual authentication or data privacy. In order to facilitate the commercial transaction processes, security protocols are required to manage the business activities. A fair exchange protocol which was proposed by Asokan *et al*. [1], for example, is designed to secure two mutually distrusting parties to achieve the exchange of electronic messages, digital signatures or electronic payments fairly over the Internet. When considering the fairness exchange problem for both sides involved, a Trusted Third Party (TTP) approach is used to attain the fairness solution. The TTP model is arguably the optimistic fair exchange protocol where the expected item received or exchanged fits the criteria as long as the TTP is available and this protocol also should give the minimum possible workload to reduce both the communication and computation costs. With the aim to meet fair exchange protocols in an efficient and practical manner, many variations of mechanisms have since been studied, but most of them lead to performance problems [4, 13, 21] or do not provide adequate security safeguards [2, 11, 12] in the exchange phase to be seriously considered for real applications. Another concern regarding fair exchange protocols that rely on the TTP to handle the services during the exchange is the escrow problem either with online or offline TTPs. Within the framework of fair exchange protocols, participants' signatures or cryptographic keys are retained in escrow so that the TTP may gain access to those vital secrets.

Owing to these reasons, one of alternatives called the concept of concurrent signatures, has been suggested to the properties of fair exchange protocols. Concurrent signatures proposed by Chen *et al*. [8] provide an alternative solution without any TTP mediations to the problem of the fair exchange. The idea is that two parties make bilateral ambiguous signatures to bind to their relevant signers concurrently while a secret (i.e., the keystone) is released by one of the two parties. To enhance full anonymity of concurrent signatures, Susilo *et al*. [31] extend this use of concurrent signatures to perfect concurrent signatures from Schnorr's [29] signature algorithm and bilinear pairings. However, in Susilo *et al*.'s [31] scheme, the initial signer can create the individual two keystones which cannot properly bind the ambiguous signature to the matching signer, and this may result in not perfect ambiguity. To surmount the perfect ambiguity problem, various concurrent-signature solutions for the investigation have been suggested, such as anonymously lattice-based group signatures [25], identity-based perfect concurrent signatures [9],

asymmetrical concurrent signatures [24], tripartite concurrent signatures [32], the fairness of perfect concurrent signatures [38], multi-party concurrent signatures [34], and so on. Unfortunately, previous studies in [8, 13, 24, 32, 38] have indicated that the existing concurrent signatures in terms of security suffer from the message substitution attack [22]. In addition, some proposed schemes [15, 19, 42] find that diverse ambiguous signatures for distinct messages might be linked to the same keystone simultaneously and this may jeopardize the accountability property of concurrent signatures. In other words, every ambiguous signature of a signed message should be identified uniquely to the signatory who signs and is bound by the individual keystone, rather than couldn't be traced to this signer. If any third parties perform an identity authentication process to validate the identity of the originator or signer prior to the binding of ambiguous signatures concurrently, the ambiguous signatures of respective messages for accountability can be guaranteed. Besides, we are concerned about the keystone information without the signers' identities bound to their authentic signatories when a relation is established, from the majority of the aforementioned concurrent-signature works [8, 20, 37]. Furthermore, Wang *et al*. [39] and Li *et al*. [18] employ a concurrent signature algorithm of quantum cryptography to perform the fair exchange problem. Although quantum mechanical properties offer faster and more secure interactions to the keystone information, quantum computing is still very much in its experimental stage. Quantum technology is typically used in conjunction with other approaches when to be practical [16], such as privacy enhancing, error-correcting and authentication mechanisms. Based on these observations, we propose that the identity authentication should be performed properly before exchanging ambiguous signatures or secret information.

In this work, our aim is to highlight the security concerns of the existing concurrent-signature solutions, such as confidentiality, non-repudiation and escrowed private keys, and to design the alternative cryptographic protocol with high levels of security, robustness and desired efficiency. In order to provide a robust secure solution with respect to the problem of fair exchange of signatures, we apply a self-certified mechanism [14] that is constructed on bilinear pairings on elliptic curves [17, 23], for concurrent signatures. The self-certified approach is one of the most popular public-key authentication schemes, in which the public key is created from both the participant and the TTP rather than this one TTP, and it makes the TTP more reliable and resistant to attack. Additionally, the elliptic curve pairings (or bilinear maps) have been used to design the protocol as the key cryptographic primitives, and the security of pairing-based cryptosystems relies on the hardness of the computational Diffie-Hellman (CDH) or bilinear Diffie-Hellman (BDH) problems [7, 26] that make them extremely difficult to break. With the self-certified pairing-based concurrent signature technique described above, this proposed scheme has the ability for the fair exchange to secure against message substitution, the identity forgery and impersonation, and other generic attacks in an open network environment.

The main contributions of this study are three-fold. First, this solution proposes a self-certified alternative from bilinear pairings on elliptic curves to concurrent signature schemes, and the mechanism through the underlying mathematical structure enabling the reliable transmission of critical information (e.g., keystones) delivers better security than the existing concurrent-signature approaches in authentication, signature construction and verification phases in terms of security properties such as accountability, confidentiality, and non-repudiation. Second, unlike other most concurrent-signature models generating the keystone information without identity binding, the proposed scheme is able to link multiple ambiguous signatures for different messages to the corresponding keystones exactly by binding pairing-based operations to their identities. It is also beneficial, especially in the case of offline identity authentication services without access to any TTPs. Third, as for the fair exchange of information between participants, the studied protocols assume that the TTP does not conspire with others to reveal confidential information. Yet misbehaving or colluding with participating entities happens due to their own business interests to the collusion attack. Consequently, every specific communication behaviour cannot be trusted according to the declared purpose. The study provides a rigorous identity authentication procedure to avoid either the collusion of participating parties involved in a communication or the illicit eavesdropping of the TTP during the key escrow process. The rest of the paper is organized as follows. Section 2 provides some background that links theoretical prerequisites to the study, such as a basic understanding of bilinear pairings, an introduction of concurrent signatures, and a brief description of the self-certified mechanism. Section 3 presents the methodology of a robust secure concurrent signature work based on bilinear pairings. Section 4 analyses the security of the proposed scheme, and Section 5 covers the results of performance evaluation along with the corresponding computational measurements. Finally, section 6 concludes the paper and addresses the important concerns for further research.

2. Preliminary Backgrounds

We first introduce some related backgrounds on bilinear pairings and the underlying field properties. Next we present the notion of concurrent signatures

and the concurrent signature algorithm. Then a brief description of the self-certified mechanism will be included in this section. These methods are appropriately used in this study, and are thoroughly described in section 3.

2.1. Bilinear Pairings and Basic Complexity Assumptions

The bilinear pairings have been used to design delicate cryptographic protocols as well-known Weil and Tate pairings over elliptic or hyperelliptic curves [17]. These pairing families were initially applied to their Menezes-Okamoto-Vanstone (MOV) attack by Menezes *et al*. [23]. This MOV algorithm for attacking elliptic curve cryptosystems uses the Weil pairing to convert the Discrete Logarithm Problem (DLP) to a certain finite field. These suitable pairings can be realized for specially chosen elliptic curves so that the reduction to DLP in a finite field is possible. For example, Joux [17] first proposes the one-off tripartite consensus in exchanging cryptographic keys based on the Weil and Tate pairings to reduce DLP on some elliptic or hyperelliptic curves to a finite field. Some of the basic concepts in bilinear pairings that relate to cryptographic settings will be briefly summarized in the following way.

Recall that the DLP gives two distinct elements P, Q from a finite cyclic group and determine a unique integer $n \in Z_q^*$, such that $Q = nP$. Let G_1 be a cyclic additive group which is generated by P, and its order is any prime number q. Let G_2 be a cyclic multiplicative group that all of its elements have the same order q. Also, let a and b be two elements of a finite group. In this context, we say that there exists an admissible bilinear pairing $\hat{e}: G_1 \times G_1 \to G_2$ for groups G_1 and G_2 if this mapping has the following properties:

1. *Bilinearity*: For all P, Q, R, a and b such that P, Q, $R \in G_1$ and a, $b \in Z_q^*$, then $\hat{e}(P+Q,R) = \hat{e}(P,R) \cdot \hat{e}(Q,R)$, $\hat{e}(P,Q+R) = \hat{e}(P,Q) \cdot \hat{e}(P,R)$, and $\hat{e}(aP,bQ) = \hat{e}(P,Q)^{ab} = \hat{e}(abP,Q) = \hat{e}(P,abQ)$.
2. *Non-Degeneracy*: For all $P \in G_1$ and $P \neq Q$, there is some $Q \in G_1$ such that $\hat{e}(P,Q) \neq 1$ and vice versa.
3. *Computability*: For all P, $Q \in G_1$, there exists an efficient algorithm for computing $\hat{e}(P,Q)$.

Note that G_1 is an additive group over elliptic or hyperelliptic curves and G_2 is a multiplicative group over a finite field. Generally, these mappings will be derived on the basis of either the Weil or Tate pairing over suitable elliptic curve groups G_1 and G_2. The security of many pairing-based protocols is dependent on the intractability of the hardness of solving the CDH or BDH problems [5, 7, 26, 41].

2.2. Concurrent Signatures

Chen *et al*. [8] were the first to adopt the concurrent signature scheme which allows two parties to create and exchange the partial ambiguous signatures from the third party's perspective until an additional secret called keystone is released by one of the two parties. When the keystone becomes publicly available, both signatures are tied to their respective signatories simultaneously. Any third parties are subsequently able to use this acknowledgement to confirm the identity of signers from the ambiguous signatures. A concurrent signature is a type of digital signature scheme, which by the definition [8, 9, 42] typically consists of the following algorithms:

1. *SETUP*: An algorithm that inputs a secure parameter l, outputs the message space M, the signature space S, the keystone space K, the keystone fix space F, a function *KEYGEN*: $K \to F$ and any other system argument π. For example, the algorithm selects a random number $s \in Z_q^*$, chooses a cryptographic hash function H: $(0, 1)^* \to G_1$, and sets $M = S = K = F = Z_q^*$.
2. *KEYGEN*: An algorithm that inputs a secure parameter l, outputs the public key P_i and the corresponding private key s_i; for instance, the signer U_i submits his/her identity ID_i to the key generation center (*KGC*), and the *KGC* sets U_i's public key $P_i = H(ID_i)$ and calculates the related private key $s_i = s \cdot P_i$ afterwards.
3. *ASIGN*: An algorithm that inputs $\{P_i, P_j, s_i, h_2, h_3, m\}$, where $h_2, h_3 \in F$, $P_i, P_i \neq P_j$ are public keys, s_i is the private key that corresponds to P_i, and $m \in M$, outputs an ambiguous signature $\sigma = (s, h_2, h_3)$ where $s \in S$. For example, the initial signer A randomly picks a key stone $k \in K$ and an arbitrary integer $\alpha_A \in Z_q^*$. The ambiguous signature $\sigma_A = ASIGN(P_A, P_B, s_A, f, m_A)$ is computed following the five steps:
   a. $\beta_1 = \hat{e}(P, P_{Pub})^{\alpha_A}$.
   b. $\beta_2 = h_2(\hat{e}(P_{Pub}, P_B)^{\alpha_A})$.
   c. $c = m_A \oplus \beta_2$.
   d. $f = h_3(k \| \beta_1 \| c) \in F$.
   e. $S = \alpha_A \cdot P_{Pub} - f \cdot s_A$.
4. *AVERIFY*: An algorithm that takes $\{\sigma, P_i, P_j, m\}$, where $\sigma = (s, h_2, h_3)$ with $s \in S$, $h_2, h_3 \in F$, P_i and P_j are public keys, and $m \in M$, accepts or rejects the message. Then it verifies that the two sides of the equation are true, i.e., $AVERIFY(\sigma', P_j, P_i, m) = AVERIFY(\sigma, P_i, P_j, m)$ for $\sigma' = (s, h_3, h_2)$. Case in point: The algorithm receives the parameter settings $\{\sigma_A, P_A, P_B, P_{Pub}, m_A\}$, and performs equality checking on these parameters:
   a. $\beta_1 = \hat{e}(P, S) \cdot \hat{e}(P_{Pub}, P_A)^f$,

b. $\beta_2 = h_2(\hat{e}(S, P_B) \cdot \hat{e}(P_A, S_B)^l)$,
c. $m_A = c \oplus \beta_2$.

5. *VERIFY*: An algorithm that takes $\{k, S\}$, where $k \in K$ denotes a keystone and S is the expression given by $S = \{\sigma, P_i, P_j, m\}$, where $\sigma = (s, h_2, h_3)$ with $s \in S$, $h_2, h_3 \in F$, P_i and P_j are public keys, and $m \in M$, accepts or rejects the message. The algorithm verifies whether the equality $KEYGEN(k) = h_3$ holds. If not, it outputs reject and terminates. Otherwise, it runs *AVERIFY*(S). To illustrate, the *VERIFY* algorithm accepts the set of parameters $\{k, \sigma_A, P_A, P_B, P_{Pub}, m_A\}$, and validates the keystone passed to the cryptographic hash function h_3 meets the checking. If the validity of k outputs an accepting run, it outputs $k \in K$, otherwise it produces $k \notin K$ and terminates. If the keystone k is set to true, this situation corresponds to running in *AVERIFY*(S) mode.

2.3. Self-Certified Mechanism

The self-certified public key scheme was introduced by Girault [14] in 1991 to implicitly establish the authenticity of itself without any TTPs, e.g., Certificate Authorities (CAs). Moreover, the TTP is unable to incidentally forge a public key without knowing a user's private key. Or, to put it another way, self-certified public keys as an independent technique, make a recommendation to the key escrow issue. In such a system, users select their own private keys, and the corresponding public key is directly computed using both the user's and a *CA*'s private keys. Then, the user's public/private key pair satisfies the relationship of unforgeability in computational difficulty which is verified as an implicit consensus through the appropriate use of the private key. The *CA* has no need to store more and more private keys as the number of users increases. After Girault's protocol [14], many other self-certified public key schemes also appear to have been different models suggested on the literature [27, 30, 36, 40]. The self-certified signature scheme is largely comprised of *KEYGEN*, *EXTRACT*, *SIGN* and *VERIFY* of four algorithms.

1. *KEYGEN*: An algorithm that inputs a secure parameter l, outputs the message space M, the signature space S, the identity space ID, two cryptographic hash functions H and F. Given a cryptographic hash function H: $(0, 1)^* \rightarrow G_1$, the system sets $M = S = ID = Z_q^*$. The *CA* selects a master private key s at random and computes the relevant public key P_{CA}. Every user picks his/her own partial private key x_{ID} and calculates the corresponding partial public key Y_{ID}. The actual public key of the user is composed of the public key of *CA*, the partial public key and the identity of the user containing the set of system parameters Φ.
2. *EXTRACT*: *An algorithm* that inputs $\{s, Y_{ID}, ID, \Phi\}$, where $s \in S$, Y_{ID} is the partial public key corresponding to x_{ID}, ID is an arbitrary number, and Φ is the set of system parameters, outputs the partial private key d_{ID} issued to the user. The *CA* sends d_{ID} to the user with $\{P_{CA}, ID, Y_{ID}\}$ over a secure channel. Therefore, (P_{CA}, ID, Y_{ID}) and (x_{ID}, d_{ID}) represent the actual public and private keys for the user respectively.
3. *SIGN*: *An algorithm* that takes $\{(P_{CA}, ID, Y_{ID}), (x_{ID}, d_{ID}), \Phi, m\}$, where (P_{CA}, ID, Y_{ID}) is the actual public key, (x_{ID}, d_{ID}) is the actual private key, Φ is the set of system parameters, and $m \in M$, outputs a signature $\sigma = ((P_{CA}, ID, Y_{ID}), (x_{ID}, d_{ID}), \Phi)$ for the message m.
4. *VERIFY*: *An algorithm* that takes $\{m, s\}$, where $m \in M$, and s is the form $s = \{\sigma, m\}$ of a signed message, where $\sigma = ((P_{CA}, ID, Y_{ID}), (x_{ID}, d_{ID}), \Phi)$ with $s \in S$, outputs a valid or invalid signature. In a nutshell, the algorithm reports true if the signature on the message is correct, or false otherwise.

3. The Proposed Concurrent Signature Scheme

In this section, we propose a robust secure concurrent signature scheme for fairly matching signers' signatures based on bilinear pairings. Since bilinear pairings possess an efficiently computable group homomorphism for elliptic curve cryptographic operations [3], the underlying structure of optimization modules can be readily used to achieve fast pairing goals (e.g., less message exchange) in improving execution. The proposed scheme also exploits self-certified public keys to tackle the problem of the uneven keystone between parties (e.g., concurrent binding controversies). In addition, the identity-based mechanism is integrated with the protocol to increase the levels of security for eliminating the overhead required to manage certificate problems (e.g., key escrow issues). Combined with the interleaving structural designs, this protocol has certainly yielded promising results on the improvements of security and capability.

The proposed scheme comprises the following five phases: initial phase, key generation phase, authentication phase, signature construction and verification phase, and concurrent signature binding phase. There are three of the main characters, *Alice*, Bob and KGC, representing respectively the initial signer, the matching signer, and the key generation center. The operational context of an interaction sequence diagram in this study is shown in Figure 1, and Table 1 gives an overview of the core elements, their notation, and their symbolic definitions.

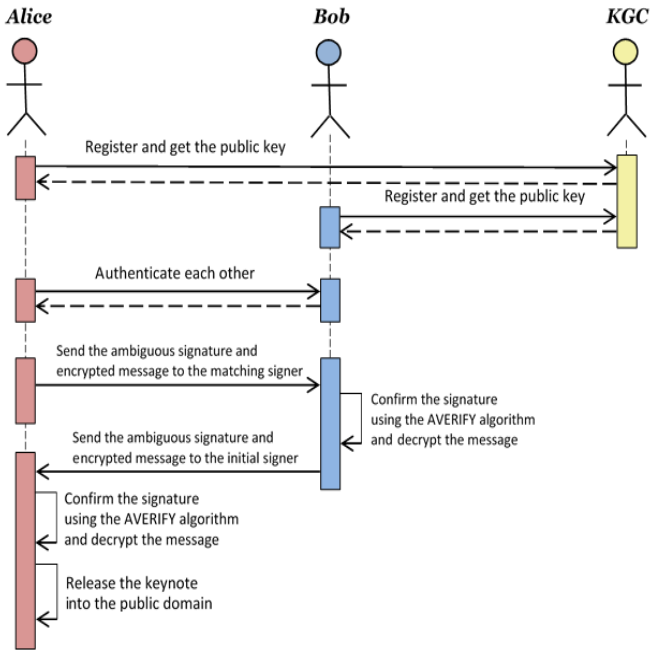

Figure 1. The sequence diagram of the proposed model's operational context.

Table 1. The notation and definitions used in the proposed scheme.

Element	Notation	Definition
1	G_1	an additive cyclic group of prime order q
2	G_2	a multiplicative cyclic group of prime order q
3	$\hat{e}$	an admissible bilinear map: $G_1 \times G_1 \rightarrow G_2$
4	P	a base point of an elliptic curve E
5	q	the order of G_1 and G_2
6	s	the master secret key (i.e., *KGC*'s private key)
7	*KGC*	key generation center
8	P_{Pub}	*KGC*'s public key
9	$H()$	one-way hash function: $(0, 1)^* \rightarrow G_1$
10	$h_1(), h_2(), h_3()$	one-way hash functions: $(0, 1)^* \rightarrow Z_q^*$
11	ID_A, ID_B	the identities of *Alice* and *Bob*
12	x_{A1}, x_{A2}	random values selected by *Alice*
13	Q_A, Q_B	public keys for *Alice* and *Bob*
14	D_A	the partial private key for *Alice*
15	R_A	the private key for *Alice*
16	α_A, α_B	random values selected by *Alice* and *Bob*
17	k	an extra secret (i.e., the keystone)
18	σ_A, σ_B	ambiguous signatures from *Alice* and *Bob*
19	m_A, m_B	the signed messages sent by *Alice* and *Bob*
20	$\|$	the concatenation operation

3.1. Initial Phase

KGC first generates two finite cyclic groups G_1, G_2 with a prime order q where G_1 is an additive group and G_2 is a multiplicative group, a random generator $P \in G_1$, and an admissible bilinear map: $G_1 \times G_1 \rightarrow G_2$.

Next, KGC selects a system security parameter $s \in Z_q^*$, as the master secret key of the system and calculates the corresponding public key from the following Equation (1).

$$P_{Pub} = s \cdot P \tag{1}$$

Then, four one-way hash functions, H: $(0, 1)^* \rightarrow G_1$, h_1: $(0, 1)^* \rightarrow Z_q^*$, h_2: $(0, 1)^* \rightarrow Z_q^*$, and h_3: $(0, 1)^* \rightarrow Z_q^*$ are also defined by *KGC*.

Last, *KGC* publishes a uniformly tuple system parameter $\{G_1, G_2, P, q, \hat{e}, H, h_1, h_2, h_3, P_{Pub}\}$ and keeps the master secret key s to itself.

3.2. Key Generation Phase

Suppose that *Alice* is the initial signer and *Bob* is the matching signer. The process of key generation proceeds as follows:

Alice chooses a casual integer $x_{A1} \in Z_q^*$, and computes the registration message Y_A as in Equation (2).

$$Y_A = x_{A1} \cdot P \tag{2}$$

After which, *Alice* keeps the value of x_{A1} secret and sends her identity ID_A to *KGC*, along with the quantity Y_A.

Then, *KGC* calculates *Alice*'s public key Q_A and her partial private key D_A via the following two mathematical expressions Equations (3) and (4).

$$Q_A = H(P_{Pub} \| ID_A \| Y_A) \tag{3}$$

$$D_A = s \cdot Q_A \tag{4}$$

Next, *KGC* randomly picks a secure element of $r \in Z_q^*$, computes the two specified U, V certificates according to Equations (5) and (6), and transmits them to *Alice*.

$$U = r \cdot P \tag{5}$$

$$V = \mathrm{D_A} + r \cdot Y_A \tag{6}$$

Upon receiving the two metrics, *Alice* can trivially recover the partial private key D_A from Equation (7) using both certificate parameters.

$$D_A = V - x_{A1} \cdot U \tag{7}$$

To verify D_A that is originally created by *KGC*, *Alice* may also use Equations (3) and (8) to determine the validity.

$$\hat{e}(D_A, P) \stackrel{?}{=} \hat{e}(Q_A, P_{Pub}) \tag{8}$$

If Equation (8) holds true at these specific values, *Alice* confirms its authenticity and takes the pair (x_{A1}, D_A) as her actual private key.

After that, *Alice* picks another single arbitrary random number $x_{A2} \in Z_q^*$, and computes the associated parameters, like a transition value v and a *KGC*-approved credential pair (w, Z), by the following measurements Equations (9), (10), and (11). Note that the credential of (w, Z) can be used for the verification of identity as a self-certified process in offline applications without access to KGC.

$$v = \hat{e}(x_{A2} \cdot Q_A, P) \tag{9}$$

$$w = h_1(v \| Q_A) \tag{10}$$

$$Z = x_{A2} \cdot Q_A + w \cdot x_{A1} \cdot Q_A + w^2 \cdot D_A \tag{11}$$

Finally, *Alice* sends the credential (w, Z), her identity ID_A, and the registration message Y_A, along with *KGC*'s public key P_{Pub}, to the matching signer *Bob*.

3.3. Authentication Phase

After Alice and Bob have completed the registration protocol and obtained their respective valid identities from *KGC*, they can communicate with each other over the public network. Before the communication can begin, the mutual authentication process is carried out to ensure that both the parties can be trusted. To do so, for example, *Bob* needs to ensure that the credential (w, Z) is valid by verifying the relevant values P_{Pub}, ID_A and Y_A of *Alice*.

For Alice to send the message to Bob at the proposed system, Bob takes the input parameters to calculate the two values Q_A and v' via Equations (3) and (12).

$$v' = \hat{e}(Z, P) \cdot \hat{e}(-Q_A, w \cdot Y_A + w^2 \cdot P_{Pub}) \tag{12}$$

Once the parameters have been determined, *Bob* can easily pass them through the hash function Equation (13), that creates the digest w.

$$w = h_1(v' \| Q_A) \tag{13}$$

Bob is able to prove the digest w correctly invoking its values by Equation (14) to that of the hash function if $v' = v$ is true.

$$\begin{aligned} \hat{e}(Z, P) &= \hat{e}(x_{A2} \cdot Q_A + w \cdot x_{A1} \cdot Q_A + w^2 \cdot D_A, P) \\ &= \hat{e}(x_{A2} \cdot Q_A, P) \cdot \hat{e}((w \cdot x_{A1} + w^2 \cdot s) \cdot Q_A, P) \\ &= v \cdot \hat{e}(Q_A, (w \cdot x_{A1} + w^2 \cdot s) \cdot P) \\ &= v \cdot \hat{e}(Q_A, w \cdot Y_A + w^2 \cdot P_{Pub}) \end{aligned} \tag{14}$$

Then, using Equation (12) again, *Bob* correctly decides the sentence as in Equation (15).

$$v' = \hat{e}(Z, P) \cdot \hat{e}(-Q_A, w \cdot Y_A + w^2 \cdot P_{Pub}) = v \tag{15}$$

3.4. Signature Construction and Verification Phase

Since the initial signer *Alice* needs to generate a keystone (i.e., an extra piece of secret information) and send an ambiguous signature to the matching signer *Bob* in order to construct a concurrent two-party signature for fair exchange information, the matching signer Bob is required to respond to this signature by producing a subsequent ambiguous signature on a message with regard to the keystone fix. The signature construction process works as follows.

Alice picks an arbitrary number of $\alpha_A \in Z_q^*$, a random keysotne $k \in K$ where K is the keystone space, and a message $m_A \in M$ where M is the message space. Also, she computes corresponding parameters, such as a disguised message c, a keystone fix f, and a signature value S, of the ambiguous signature and they are estimated by Equations (16), (17), (18), (19), and (20).

$$\beta_1 = \hat{e}(P, P_{Pub})^{\alpha_A} \tag{16}$$

$$\beta_2 = h_2(\hat{e}(P_{Pub}, Q_B)^{\alpha_A}) \tag{17}$$

$$c = m_A \oplus \beta_2 \tag{18}$$

$$f = h_3(k \| \beta_1 \| c) \in F \tag{19}$$

$$S = \alpha_A \cdot P_{Pub} - f \cdot D_A \tag{20}$$

When the corresponding parameters are obtained from these computations, *Alice* creates her ambiguous signature σ_A as in Equation (21) and sends σ_A, c, and S to the matching signer *Bob*.

$$\sigma_A = ASIGN(Q_A, Q_B, D_A, f, m_A) \tag{21}$$

Upon receiving *Alice*'s ambiguous signature σ_A with other two parameters, *Bob* takes these specific values to apply to Equations (22), (23), and (24) for the validity of the ambiguous signature by checking whether each of the results is acceptable or not as calculated from Equation (25).

$$\beta_1 = \hat{e}(P, S) \cdot \hat{e}(P_{Pub}, Q_A)^f \tag{22}$$

$$\beta_2 = h_2(\hat{e}(S, Q_B) \cdot \hat{e}(Q_A, S_B)^f) \tag{23}$$

$$m_A = c \oplus \beta_2 \tag{24}$$

$$AVERIFY(m_A, \sigma_A, Q_A, Q_B, P_{Pub}) \tag{25}$$

If the verification algorithm of *Alice*'s ambiguous signature σ_A fails, the communication process is immediately terminated, or *Bob* chooses a random number of $\alpha_B \in Z_q^*$ and a signed message of $m_B \in M$, and computes the relevant parameters through the use of Equations (26), (27), (28), (29), and (30) in similar calculations for *Alice* to produce his ambiguous signature σ_B by Equation (31). Then, *Bob*'s corresponding parameters σ_B, c', and S', are transmitted back to the initial signer *Alice*.

$$\beta_3 = \hat{e}(P, P_{Pub})^{\alpha_B} \tag{26}$$

$$\beta_4 = h_2(\hat{e}(P_{Pub}, Q_A)^{\alpha_B}) \tag{27}$$

$$c' = m_B \oplus \beta_4 \tag{28}$$

$$f' = h_3(f \| \beta_3 \| c') \in F \tag{29}$$

$$S' = \alpha_B \cdot P_{Pub} - f' \cdot D_B \tag{30}$$

$$\sigma_B = ASIGN(Q_A, Q_B, D_B, f', m_B) \tag{31}$$

After receiving the specific information of *Bob*'s ambiguous signature, *Alice* applies them to Equations (32), (33), and (34) for testing whether every element of the parameters satisfies the verification algorithm as in Equation (35) for the specific conditions.

$$\beta_3 = \hat{e}(P, S') \cdot \hat{e}(P_{Pub}, Q_B)^{f'} \tag{32}$$

$$\beta_4 = h_2(\hat{e}(S', Q_A) \cdot \hat{e}(Q_B, S_A)^{f'}) \tag{33}$$

$$m_B = c' \oplus \beta_4 \tag{34}$$

$$AVERIFY(m_B, \sigma_B, Q_A, Q_B, P_{Pub}) \tag{35}$$

If Equation (35) is true, the verification algorithm of *Bob*'s ambiguous signature σ_B succeeds, otherwise *Alice* immediately aborts the protocol service.

3.5. Concurrent Signature Binding Phase

Once the ambiguous signatures are properly generated by both parties via the ASIGN algorithm and successfully verified by the AVERIFY algorithm, the keynote k is released to the public domain by *Alice*. Next, the two-party ambiguous signatures are incorporated into the binding process to build the concurrent signature (m_A, σ_A, c, S, m_B, σ_B, c', S'). Then any new participants are able to check the validity of *Alice*'s and *Bob*'s signatures and to bind their identities if the secret parameters indeed satisfy Equation (36).

$$VERIFY(k, m_A, \sigma_A, c, S, m_B, \sigma_B, c', S') \quad (36)$$

4. Security Analysis of the Proposed Scheme

The security of the proposed pairing-based concurrent signature model is based primarily on the difficulty of solving the CDH or BDH problems [7, 26]. Also, the self-certified system [14] is incorporated into the proposed algorithm for the construction of concurrent signatures without any trusted third parties. The proposed model as such satisfies the security requirements on the basis of these techniques including accountability, ambiguity, confidentiality, correctness, fairness, non-repudiation, unforgeability, and the self-certified approach [8, 20, 42]. In this section, we first prove the correctness of the proposed scheme and then show that our approach has all the security goals.

4.1. Correctness of the Proposed Model

The correctness of the proposed scheme can be verified by examining if the initial signer *Alice* and the matching signer *Bob* mutually authenticate each other. That is, any participant would like to communicate with each other and the authentication procedure is correctly established. We use the widely-accepted Burrows-Abadi-Needham logic (a.k.a., the BAN logic) [6] to assumptions and assertions, to prove the correctness and analyze its information exchange measures. In the proposed scheme, the generic forms of the messages exchanged between *Alice* and *Bob* are expressed as follows:

Message 1. *Alice* $\rightarrow$ *Bob*: $\langle w, x_{A2} \cdot Q_A + w \cdot x_{A1} \cdot Q_A + w^2 \cdot D_A, P_{Pub}, ID_A, x_{A1} \cdot P \rangle$.

Message 2. *Bob* $\rightarrow$ *Alice*: $\langle w, x_{B2} \cdot Q_B + w \cdot x_{B1} \cdot Q_B + w^2 \cdot D_B, P_{Pub}, ID_B, x_{B1} \cdot P \rangle$.

The following assumptions about the initial state of the proposed scheme are made:

A_1: $Alice \mid\equiv \#(x_{A1}, x_{A2}, D_A)$;

A_2: $Bob \mid\equiv \#(x_{B1}, x_{B2}, D_B)$;

A_3: $Alice \mid\equiv (Alice \overset{v}{\longleftrightarrow} Bob)$;

A_4: $Bob \mid\equiv (Alice \overset{v}{\longleftrightarrow} Bob)$;

A_5: $Alice \mid\equiv Bob \mid\equiv (Alice \overset{v}{\longleftrightarrow} Bob)$;

A_6: $Bob \mid\equiv Alice \mid\equiv (Alice \overset{v}{\longleftrightarrow} Bob)$.

- *Lemma 1. The matching signer Bob correctly verifies the authenticity of the initial signer Alice's message.*
- *Proof. Alice* sends a private message to *Bob*. With the request message, *Bob* receives the validation ticket v with the related parameter values to prove the correctness of the verification message. The following states describe the message propagation.

  S_1: According to *Message* 1, we have $Bob: \triangleleft (w, Z_{Pub})_v, P_{Pub}, ID_A, Y_A$.

  S_2: Using the assumption A_4, the further rule, $Bob \mid\equiv Alice \mid\sim (x_{A1}, x_{A2}, D_A)$ is derived.

  S_3: The freshness propagation rule, $Bob \mid\equiv \#(w, Z_{Pub})_v$, is obtained using the assumption A_1.

  S_4: The nonce verification rule, $Bob \mid\equiv Alice \mid\equiv \#(w, Z_{Pub})_v$, is derived from S_2 and S_3.

  S_5: The jurisdiction rule, $Bob \mid\equiv (x_{A1}, x_{A2}, D_A)$, can be inferred from the assumption A_4 and S_4.

Accordingly, *Bob* can use the related parameters that uniquely identify the message from *Alice*.

- *Lemma* 2. The initial signer Alice properly verifies the authenticity of the matching signer Bob's response message.
- *Proof.* When the correctness of the request message is verified, *Alice* performs the following message passing steps based on regard to required parameters to determine the accuracy of *Bob*'s reply message.

  S_6: According to *Message* 2, we get $(w, Z_{Pub})_v$, P_{Pub}, ID_B, Y_B.

  S_7: Using the assumption A_3, the further rule, $Alice \mid\equiv Bob \mid\sim (x_{B1}, x_{B2}, D_B)$, is derived.

  S_8: From the assumption A_1, we apply the freshness conjuncatenation rule to yield $Alice \mid\equiv \#(w, Z_{Pub})_v$.

  S_9: The nonce verification rule, $Alice \mid\equiv Bob \mid\equiv (w, Z_{Pub})_v$, is obtained from S_8.

  S_{10}: The jurisdiction rule, $Alice \mid\equiv (x_{B1}, x_{B2}, D_B)$, is able to be inferred from the assumption A_1, A_3 and S_9.

Thus, *Alice* correctly examines the response message from *Bob*.

By *Lemma* 1 and *Lemma* 2, these show that both the initial signer *Alice* and the matching signer *Bob* achieve the goal of mutual authentication each other.

4.2. Security Discussion of the Proposed Design

This subsection presents the security countermeasures of the proposed scheme in terms of various security properties, including the accountability, ambiguity,

confidentiality, correctness, fairness, non-repudiation, unforgeability, and self-certified approach.

1. *Accountability*: Accountability of digital signature protocols is the property that ensures that the actions of an individual can be identified uniquely to this originator. The context of accountability for concurrent signature schemes is the way that any third parties could firmly believe that the ambiguous signature of a signed message is the only one produced by the signer in accordance with the released keystone. Aside from the one he/she sent to other signers using his/her ambiguous signature that can satisfy the verification of the *VERIFY* and *AVERIFY* algorithms, no signer can generate an ambiguous signature on the respective messages. In the present study, the keystone information *k* binds (m_A, σ_A) to *Alice* as well as (m_B, σ_B) to *Bob* as given in Equations (25), (35), and (36). If any of the signers sends a fake ambiguous signature to others, the forged signature won't pass the verification of both the *AVERIFY* and *VERIFY* algorithms.
2. *Ambiguity*: Ambiguity property emphasizes that any third parties cannot identify who is the exactly signatory of an ambiguous signature before the keystone is released by one of the two participating parties. As for the ambiguous signatures of Equations (23) and (33) in this study, the messages exchanged, m_A and m_B, between the involved parties are applied to the secure one-way hash function as given in the protocol of Equations (19) and (29). An adversary is not able to determine that *Alice* or *Bob* is the actual signer of an ambiguous signature until the keystone *k* is released, except that the adversary knows that the actual signer is among them. Thereupon, the proposed scheme fulfills the ambiguity property.
3. *Confidentiality*: Confidentiality means the peculiarity that information is not accessed or disclosed to unauthorized individuals, parties, or processes. In the study, all messages, including encryption keys, are transmitted in encrypted forms (e.g., in a bilinear form or a cryptographic hash algorithm). If a malicious third party intercepts the ciphertext (c, c') and attempts to decode the messages as Equations (24) and (34), $m_A = c \oplus \beta_2$ and $m_B = c' \oplus \beta_4$, it is practically impossible to decrypt the ciphertext with the unrelated keys or parameters like β_2 and β_4, since the malicious third party has to face the difficulties of solving the CDH and One-Way Hash Function (OWHF) problems. Thus, the proposed methodology for concurrent signature schemes meets the characteristic for confidentiality.
4. *Correctness*: According to the correctness definition given by Chen *et al*. [8], if an ambiguous signature σ has been genuinely created by calling on the *ASIGN* algorithm on a message m and the *AVERIFY* algorithm correctly outputs "accept" with a set of the input values, both the message *m* and the ambiguous signature σ of m are correct. Because the *AVERIFY* algorithm takes as *Alice*'s input (m_A, σ_A, Q_A, Q_B, P_{Pub}) from Equation (25) and *Bob*'s input (m_B, σ_B, Q_A, Q_B, P_{Pub}) by Equation (35), and returns the results of "accept", the proposed study satisfies the requirement for correctness.
5. *Fairness*: The fairness of concurrent signature is guaranteed only once the initial signer uses the concurrent signature protocol he/she has built, and only if this ambiguous signature is seen by the matching signer since he/she therefore gets the keystone information. According to the study, if a dishonest party pretends to be the role of the matching signer as *Bob*, who performs an improper disclosure of the ambiguous signature, the dishonest party cannot successfully find the ambiguous signature. Since the messages are bound into the ambiguous signature via the hash function of Equations (19) and (29) at the same time, that's established between *Alice* and *Bob* when they exchange messages, and is released through the keystone information that participants can confirm the ambiguous signature, the dishonest signature is subject to the verification of the *VERIFY* algorithm and it is very easy to verify if the keystone information is assured of *Bob*'s authenticity.
6. *Non-Repudiation*: Non-repudiation refers to the assurance to ensure that a party to a communication cannot refuse the validity of their signatures that the message has been actually sent. On the basis of the study, Alice can't deny she signed the message m_A with σ_A, as the concurrent signature's signer is one and only. After the communication Alice cannot deny she get the right information of Bob's ambiguous signature, because the ciphertext c' is signed by Bob, it can only be deciphered by the right keystone *k*. Also Bob can't refuse he has received the signed message by Alice, and he is the only accepter of the specific information. While the concurrent signature protocol is achieved and the keystone *k* is released, any third parties can use the VERIFY algorithm to examine the authenticity of their signatures. Consequently, the property of non-repudiation can be achieved through the proposed model.
7. *Unforgeability*: In the context of the forgery attack defense the unforgeability property is quite important, and it stresses only the signer, who has the private key, can yield the valid ambiguous signature linked with the keystone for the associated message. In the work, if an opponent pretends to be a matching signer as *Bob*, the opponent signs his/her message m'_B to get the forged ambiguous signature σ'_B (of m'_B) and sends σ'_B to *Alice*. When *Alice* receives σ'_B, the fraudulent ambiguous signature

won't pass the signature verification of the *VERIFY* algorithm since the value of the fraud signature (m'_B, σ'_B) is reflected in its resistance to the keystone k. Hence the proposed protocol fulfills the unforgeability feature.

8. *Self-Certified Approach*: The self-certified public key approach emphasizes that the relevant public key needs not be accompanied with an extra certificate to be authenticated by a trusted system authority, while the private key is still chosen by each participant himself/herself and remains unknown to the authority. In the present study, *Alice* or *Bob* obtains a valid certificate (e.g., *Alice*'s credential (w, Z)) while registering the corresponding identification information on *KGC*, and keeps the private key (e.g., *Alice*'s secret key x_{A1}) secret. If *KGC* doesn't know *Alice*'s private key, *KGC* cannot impersonate *Alice* to derive *Alice*'s public key Q_A. Due to such an advantage, apart from verifying the identity, e.g. *Alice*'s security parameter v, of the communicating parties in an offline environment without KGC, this procedure ensures that services are protected against specific attacks, including collusion attacks at the server side. Using the self-certified system can strengthen application security services and prevent security vulnerabilities.

We have used essential BAN logic [6] properties together with a formal proof to assert the correctness of the proposed scheme and discussed its security characteristics in the context of demands for information security and the circumstances in which those requirements meet these goals to implement the concurrent signature using the technique of bilinear pairings. If we reinvestigate the similarity models of the existing concurrent signature approaches by comparing the security features, it is obvious that the proposed method provides security requirements for the effectiveness of countermeasures. Conversely, Chen *et al*. [8] introduce the concept of concurrent signatures and the original protocol doesn't involve offering the security-related requirements in data transmission, such as accountability, confidentiality or non-repudiation, but rather providing the secure features on ambiguity, correctness, fairness and unforgeability. Wang *et al*.'s [37] improved concurrent signature algorithm strengthens the component of accountability that enables distinctiveness of the message to be bound with the keystone and this signatory, and Zang and Xu's [42] identity-based method also improves the unforgeability characteristic to prevent forgery attacks. Unfortunately, we found the schemes given in both of them [20, 37] might cause minor confidentiality breaches since the keystone information doesn't consist of validating the signer's identities when establishing relationships and it is vulnerable to the identity forgery and impersonation attack or even the message substitution attack, e.g. the keystone hashed by the function as the digest $H(k, m_A, m_B)$. After all, an identity-authenticated key agreement protocol assures that the parties can share the keystone and adversaries cannot leverage this information to glean any additional information about the signatories, e.g. Equation (19) turning out to be the identity-bound keystone in our case.

Table 2 presents a comparison between the proposed mechanism and the existing four concurrent-signature techniques from the claimed security objectives. Symbol "√" indicates that the algorithm satisfies the security feature, and symbol "×" refers that the model partially or not supports the security requirement. As shown in Table 2, the current solution offers the relevant security attributes in the application of concurrent-signature cryptosystems, while the existing works suffer from the potential security issues such as accountability, confidentiality, non-repudiation and self-certified approach.

Table 2. Comparison of security requirements of the proposed scheme and the existing concurrent-signature schemes.

Algorithms Security objectives	Chen et al.'s scheme [8]	Wang *et al.*'s scheme [37]	Zang & Xu's scheme [42]	Liaw *et al.*'s scheme [20]	The proposed scheme
Accountability	×	√	√	√	√
Ambiguity	√	√	√	√	√
Confidentiality	×	×	√	×	√
Correctness	√	√	√	√	√
Fairness	√	√	√	√	√
Non-repudiation	×	×	√	×	√
Unforgeability	√	√	√	√	√
Self-certified approach	×	×	×	√	√

5. Performance Evaluation of the Proposed Scheme

Having analyzed the effectiveness of security countermeasures for the proposed scheme, we evaluate the performance of the proposed algorithm in terms of the corresponding phase's execution time, and show that it also brings a promising efficiency compared with the existing works with respect to the application of concurrent-signature approaches. We will examine the theoretical framework of these various solutions for solving the techniques of cryptology related to both computation and communication costs incurred by each phase in accordance with the concept of modular arithmetic operations [10, 33, 35]. The notation of modular multiplications has been widely used in many public-key cryptosystems for evaluating the complexity in terms of time and resources required, and the main operations shown in Table 3 include modular multiplication, modular addition, modular exponentiation, modular inversion, SHA-1 (Secure Hash Algorithm 1) [28] hash, exclusive disjunction, and bilinear maps (pairings).

Though the above-mentioned four techniques have not the exactly same steps as the proposed concurrent-signature way, we still try to establish the baseline whenever possible to measure the outcomes of different stages of the process of action. Table 4 summarizes the computational costs of each step involved in these concurrent signature protocols. Compared to other related algorithms for performing cryptographic operations, we observed that the proposed scheme takes a little more T_{MUL} time than others do without using the bilinear pairing operations for the key generation, authentication, and signature construction and verification phases. For example, our method consumes $613.2T_{MUL}$ time in handling the key generation process including the use of the participant's identity of encryption from pairings, whereas other approaches for generating crypto keys, e.g., Zang and Xu's [42] and Liaw *et al.*'s [20] algorithms, spend less time for this purpose as $29.4T_{MUL}$ and $189.2T_{MUL}$ time respectively. In addition, Zang and Xu's [42] model uses less cost (taking $538.4T_{MUL}$ time) by adopting bilinear pairings methodology during the period of signature construction and verification, and it may leave the model exposed to the impersonation or collusion attacks attack since an adversary can successfully assume the identity of the legitimate participant with no prior encryption of Alice's or Bob's identity. Certainly, their solution does not use a self-certification mechanism and depends on the continuous availability of the TTP; that is to say, it requires that the dedicated server authenticates the identity of the requesting participants all the time.

Table 3. The modular multiplication notation.

Symbol	Description	Operation cost
T_{ECMUL}	the time for the point multiplication operation on elliptic curves	$\approx 29T_{MUL}$
T_{ECADD}	the time for the point addition operation on elliptic curves	$\approx 5T_{MUL}$
T_{INVS}	the time for the operation of modular multiplicative inverse	$\approx 240T_{MUL}$
T_{EXP}	the time for the operation of modular exponentiation	$\approx 240T_{MUL}$
T_{ADD}	the time for the operation of modular addition	The time complexity for T_{ADD} is negligible.
t_h	the time for the SHA-1 (Secure Hash Algorithm 1) operation	$\approx 0.4T_{MUL}$
$T_{\oplus}$	the time for the Exclusive-OR operation	The time complexity for $T_{\oplus}$ is negligible.
T_{BP}	the time for the bilinear pairing operation	$\approx 120T_{MUL}$

Note: Modular multiplication is a fundamental operation in many popular public-key cryptosystems.
It converts various operations units to the time complexity in terms of T_{MUL}.

The computational cost of the proposed protocol is increased apparently (running in $1800.4T_{MUL}$ time) for raising the level of security and less ineffective than that of Zang and Xu's [42] pairing-based algorithm on the phase of constructing and verifying signatures due to the consumption in online or offline identity authentication services. Even though the length of our work is longer than Zang and Xu's [42] approach, the proposed scheme is better than the previous solution that not only does ensure the study is significantly more robust against the identity forgery, impersonation or collusion attacks, but it provides an offline identity authentication as the anonymous credential without access to any TTPs.

Table 4. Performance comparison between the proposed scheme and the existing concurrent-signature algorithms.

Method / Cost Stage	Chen *et al.*'s scheme [8]		Wang *et al.*'s scheme [37]		Zang and Xu's scheme [42]		Liaw *et al.*'s scheme [20]		The proposed scheme	
	Computational cost	Rough estimation	Computational cost	Rough estimation	Computational cost	Rough estimation	Computational cost	Rough estimation	Computational cost	Rough estimation
Initial	$2T_{EXP}$	$480T_{MUL}$	$2T_{EXP}$	$480T_{MUL}$	$2t_h + 1T_{ECMUL}$	$29.8T_{MUL}$	$2T_{ECMUL}$	$58T_{MUL}$	$4t_h + 1T_{ECMUL}$	$30.6T_{MUL}$
Key generation	None	None	None	None	$1t_h + 1T_{ECMUL}$	$29.4T_{MUL}$	$8t_h + 2T_{MUL} + 6T_{ECMUL} + 8T_{ADD} + 2T_{ECADD}$	$189.2T_{MUL}$	$3t_h + 8T_{ECMUL} + 3T_{BP} + 4T_{ECADD}$	$613.2T_{MUL}$
Authentication	None	None	None	None	$2t_h + 4T_{ECMUL} + 2T_{BP} + 1T_{ECADD} + 1T_{\oplus}$	$361.8T_{MUL}$	$4t_h + 4T_{ECMUL} + 4T_{ADD}$	$117.6T_{MUL}$	$2t_h + 2T_{ECMUL} + 2T_{BP} + 1T_{ECADD}$	$303.8T_{MUL}$
Signature construction and verification	$9t_h + 2T_{MUL} + 2T_{INVS} + 4T_{EXP} + 3T_{ADD}$	$485.6T_{MUL}$	$9t_h + 2T_{MUL} + 2T_{INVS} + 4T_{EXP} + 3T_{ADD}$	$485.6T_{MUL}$	$1t_h + 2T_{ECMUL} + 4T_{BP} + 1T_{\oplus}$	$538.4T_{MUL}$	$13t_h + 2T_{MUL} + 3T_{ECMUL} + 5T_{INVS} + 12T_{EXP} + 10T_{ADD}$	1406.6 T_{MUL}	$6t_h + 12T_{ECMUL} + 12T_{BP} + 2T_{ECADD} + 5T_{\oplus}$	1800.4 T_{MUL}
Concurrent signature binding	$2t_h + 1T_{ADD}$	$0.8T_{MUL}$	$2t_h + 1T_{ADD}$	$0.8T_{MUL}$	Not Stated	Not Stated	$2t_h + 1T_{ADD}$	$0.8T_{MUL}$	$2t_h$	$0.8T_{MUL}$

Annex 1: Since Chen *et al.*'s [8] and Wang *et al.*'s [37] algorithms don't use the concept of bilinear pairings, there is no the consumption of the computational costs in both key generation and authentication phases.
Annex 2: There are five phases as mentioned previously (see Section 3 above) in the proposed scheme, inclusive of the approach of self-certified public keys and information encryption techniques. This is the design of an enhanced security and it leads to a little bit time consuming due to the complicated computations.

6. Conclusions

This paper presents an alternative pairing-based concurrent signature scheme based on the difficulty of solving the CDH or BDH problems. To improve the security of fair exchange information signed concurrently between the initial signer and the matching signer, the self-certified technique is properly incorporated into the concurrent signature protocol. Apart from the primary benefits of security improvements in the underlying field operations, the rigorous authentication process, including offline identity verification, embedded in the proposed scheme is robust enough to prevent the server-aided computations from the collusion attacks of the adversaries.

We give the correctness proof of the proposed concurrent signature protocol, that the matching signer *Bob* correctly verifies the authenticity of the initial signer *Alice*'s message, as well as the initial signer *Alice* properly verifies the authenticity of the matching signer *Bob*'s response message. By analyzing the security features in Section 4, the current study satisfies the practical use of eight different security requirement parts for a pairing-based concurrent signature cryptosystem described in the preceding section while using the cryptographic primitives that secure better. In addition, we have evaluated the computational cost to illustrate the importance of accounting for the efficiency defense in signature construction and verification, and the results show that the pairing-based model is able to achieve a gradually more positive attitude towards efficiencies when compared to other existing concurrent-signature algorithms.

To the best of our knowledge, the mechanism described in this paper is the first attempt at using a pairing-based concurrent signature with the self-certified public key system. Providing an effective and secure solution in malicious cyber activities requires the features of established use accordingly. We are convinced that the current scheme provides significant ameliorations with the security characteristics mentioned previously for the application of pairing-based concurrent signature cryptosystems. Though more scalar multiplications are required to calculate the concurrent signature between parties in our scheme, which raises an important question about how to deliver computationally less expensive to perform the cryptography-related operations in bilinear pairings or bilinear maps, we truly develop a provably secure pairing-based concurrent signature scheme in terms of security requirements, which adapts robustly to the applications of e-cash systems, e-payment systems or e-contract signing protocols.

References

[1] Asokan N., Schunter M., and Waidner M., "Optimistic Protocols for Fair Exchange," *in Proceedings of the 4th ACM Conference on Computer and Communications Security*, Zurich, pp. 7-17, 1997.

[2] Asokan N., Shoup V., and Waidner M., "Optimistic Fair Exchange of Digital Signatures," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 593-610, 2000.

[3] Azarderakhsh R., Fishbein D., Grewal G., Hu S., Jao D., Longa P., and Verma R., "Fast Software Implementations of Bilinear Pairings," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 6, pp. 605-619, 2017.

[4] Bao F., Deng R., and Mao W., "Efficient and Practical Fair Exchange Protocols with Off-Line TTP," *in Proceedings of IEEE Symposium on Security and Privacy*, Oakland, pp. 77-85, 1998.

[5] Boneh D. and Boyen X., "Secure Identity Based Encryption without Random Oracles," *in Proceedings of Annual International Cryptology Conference*, Santa Barbara, pp. 443-459, 2004.

[6] Burrows M., Abadi M., and Needham R., "A Logic of Authentication," *ACM Transactions on Computer Systems*, vol. 8, no. 1, pp. 18-36, 1990.

[7] Chatterjee S. and Menezes A., "on Cryptographic Protocols Employing Asymmetric Pairings-The Role of Ψ Revisited," *Discrete Applied Mathematics*, vol. 159, no. 13, pp. 1311-1322, 2011.

[8] Chen L., Kudla C., and Paterson K., "Concurrent Signatures," *in Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, Interlaken, pp. 287-305, 2004.

[9] Chow S. and Susilo W., "Generic Construction of (Identity-Based) Perfect Concurrent Signatures," *in Proceedings of the International Conference on Information and Communications Security*, Beijing, pp. 194-206, 2005.

[10] David J., Kalach K., and Tittley N., "Hardware Complexity of Modular Multiplication and Exponentiation," *IEEE Transactions on Computers*, vol. 56, no. 10, pp. 1308-1319, 2007.

[11] Dodis Y., Lee P., and Yum D., "Optimistic Fair Exchange in a Multi-User Setting," *in Proceedings of the International Workshop on Public Key Cryptography*, Beijing, pp. 118-133, 2007.

[12] Galbraith S., Malone-Lee J., and Smart N., "Public Key Signatures in the Multi-User Setting," *Information Processing Letters*, vol. 83, no. 5, pp. 263-266, 2002.

[13] Garay J., Jakobsson M., and MacKenzie P., "Abuse-Free Optimistic Contract Signing," *in Proceedings of the Annual International*

Cryptology Conference, California, pp. 449-466, 1999.

[14] Girault M., "Self-Certified Public Keys," *in Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques*, Brighton, pp. 490-497, 1991.

[15] Huang X. and Wang L., "A Fair Concurrent Signature Scheme Based on Identity," *in Proceedings of the High Performance Computing and Applications*, Shanghai, pp. 198-205, 2010.

[16] Jain N., Stiller B., Khan I., Elser D., Marquardt C., and Leuchs G., "Attacks on Practical Quantum Key Distribution Systems (and How to Prevent Them)," *Contemporary Physics*, vol. 57, no. 3, pp. 366-387, 2016.

[17] Joux A., "A One Round Protocol for Tripartite Diffie-Hellman," *Journal of Cryptology*, vol. 17, no. 4, pp. 263-276, 2004.

[18] Li B., Zhao H., and Li J., "Lattice-Based Concurrent Signatures in the Standard Model," *in Proceedings of the International Conference on Computer Science and Application Engineering*, Sanya, pp. 1-5, 2020.

[19] Li Y., He D., and Lu X., "Accountability of Perfect Concurrent Signature," *in Proceedings of the International Conference on Computer and Electrical Engineering*, Phuket, pp. 773-777, 2008.

[20] Liaw S., Lu E., Chang H., and Su P., "New Security Concurrent Signature Design," *Journal of Internet Technology*, vol. 19, no. 3, pp. 741-751, 2018.

[21] Markowitch O. and Saeednia S., "Optimistic Fair Exchange with Transparent Signature Recovery," *in Proceedings of the International Conference on Financial Cryptography*, Grand Cayman, pp. 339-350, 2002.

[22] Menezes A. and Smart N., "Security of Signature Schemes in a Multiuser Setting," *Designs, Codes and Cryptography*, vol. 33, no. 3, pp. 261-274, 2004.

[23] Menezes A., Okamoto T., and Vanstone S., "Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field," *IEEE Transactions on Information Theory*, vol. 39, no. 5, pp. 1639-1646, 1993.

[24] Nguyen K., "Asymmetric Concurrent Signatures," *in Proceedings of the 7th international conference on Information and Communications Security*, Beijing, pp. 181-193, 2005.

[25] Preethi T. and Amberker B., "Traceable Signatures using Lattices," *The International Arab Journal of Information Technology*, vol. 17, no. 6, pp. 965-975, 2020.

[26] Qin B., Liu S., Sun S., Deng R., and Gu D., "Related-Key Secure Key Encapsulation from Extended Computational Bilinear Diffie-Hellman," *Information Sciences*, vol. 406-407, pp. 1-11, 2017.

[27] Saeednia S., "A Note on Girault's Self-Certified Model," *Information Processing Letters*, vol. 86, no. 6, pp. 323-327, 2003.

[28] Schneier B., "Schneier on Security: Cryptanalysis of SHA-1," 2005.

[29] Seurin Y., "On the Exact Security of Schnorr-Type Signatures in the Random Oracle Model," *in Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Cambridge, pp. 554-571, 2012.

[30] Shao Z., "Self-Certified Signature Scheme from Pairings," *Journal of Systems and Software*, vol. 80, no. 3, pp. 388-395, 2007.

[31] Susilo W., Mu Y., and Zhang F., "Perfect Concurrent Signature Schemes," *in Proceedings of International Conference on Information and Communications Security*, Málaga, pp. 14-26, 2004.

[32] Susilo W. and Mu Y., "Tripartite Concurrent Signatures," *in Proceedings of IFIP TC11 the 20th International Information Security Conference*, Chiba, pp. 425-441, 2005.

[33] Tahat N. and Abdallah E., "A New Signing Algorithm Based on Elliptic Curve Discrete Logarithms and Quadratic Residue Problems," *Italian Journal of Pure and Applied Mathematics*, vol. 32, pp. 125-132, 2014.

[34] Tonien D., Susilo W., and Naini R., "Multi-Party Concurrent Signatures," *in Proceedings of the International Conference on Information Security*, Samos, pp. 131-145, 2006.

[35] Tsai C. and Su P., "An ECC-Based Blind Signcryption Scheme for Multiple Digital Documents," *Security and Communication Networks*, vol. 2017, pp. 1-14, 2017.

[36] Tsaur W., "Several Security Schemes Constructed Using ECC-Based Self-Certified Public Key Cryptosystems," *Applied Mathematics and Computation*, vol. 168, no. 1, pp. 447-464, 2005.

[37] Wang C., Chen C., and Wu C., "Using and Improvement of Concurrent Signature for Fair Exchange," *Communications of Chinese Cryptology and Information Security Association*, vol. 16, no. 3, pp. 60-71, 2010.

[38] Wang G., Bao F., and Zhou J., "The Fairness of Perfect Concurrent Signatures," *in Proceedings of the 8th international conference on Information and Communications Security*, Raleigh, pp. 435-451, 2006.

[39] Wang H., Yao G., and Wang B., "A Quantum Concurrent Signature Scheme Based on the Quantum Finite Automata Signature Scheme," *in Proceedings of International Conference on*

Anti-counterfeiting, Security, and Identification, Xiamen, pp. 125-129, 2020.
[40] Wu T., Chang Y., and Lin T., "Improvement of Saeednia's Self-Certified Key Protocol," *Electronics Letters*, vol. 34, no. 11, pp. 1094-1095, 1998.
[41] Zhang J. and Mao J., "A Novel ID-Based Designated Verifier Signature Scheme," *Information Sciences*, vol. 178, no. 3, pp. 766-773, 2008.
[42] Zhang Z. and Xu S., "Cryptanalysis and Improvement of a Concurrent Signature Scheme Based on Identity," *in Proceedings of the 2nd International Conference on Software Engineering and Service Science*, Beijing, pp. 453-456, 2011.

Chien-Hua Tsai is currently an Associate Professor in the Department of Accounting Information at Chihlee University of Technology, Taiwan. He received his Ph.D. degree in Electrical Engineering and Computer Science from Case Western Reserve University, Ohio, USA in 2000. His research interests include Information Systems Security, Secure Communication Protocols, Public Key Cryptosystems and Electronic Transaction Security in Computer and Network Security. He has published several articles in most academic journals from Information Systems and e-Business Management, Security and Communication Networks, Computers and Electrical Engineering, Journal of Internet Technology, Journal of e-Business, Management Review and so on.

Pin-Chang Su is presently working as a Professor in the Department of Information Management at National Defense University, Taiwan. He received his Ph.D. degree in Electrical Engineering from Chang Gung University, Taiwan in 2007. His research mainly focuses on Algorithms Design in Error-Control Coding, Information Security, Cryptographic Systems and E-Commerce Technologies. His published articles can be found in most academic journals like KSII Transactions on Internet and Information Systems, Computers and Electrical Engineering, Security and Communication Networks, Journal of Internet Technology, Journal of Chung Cheng Institute of Technology, Journal of e-Business and so forth.