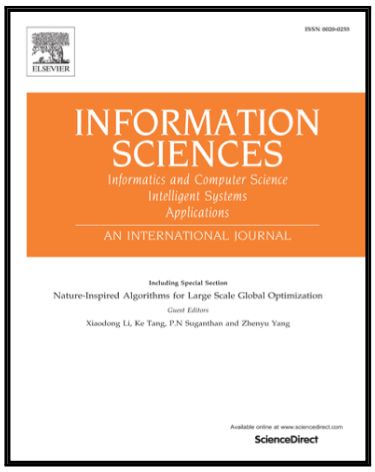

CSP-E^2: An abuse-free Contract Signing Protocol with low-storage TTP for energy-efficient electronic transaction ecosystems

Guangquan Xu, Yao Zhang, Arun Kumar Sangaiah, Xiaohong Li, Aniello Castiglione, Xi Zheng

PII: S0020-0255(18)30378-5
DOI: 10.1016/j.ins.2018.05.022
Reference: INS 13652

To appear in: *Information Sciences*

Received date: 1 December 2017
Revised date: 21 April 2018
Accepted date: 6 May 2018

Please cite this article as: Guangquan Xu, Yao Zhang, Arun Kumar Sangaiah, Xiaohong Li, Aniello Castiglione, Xi Zheng, CSP-E^2: An abuse-free Contract Signing Protocol with low-storage TTP for energy-efficient electronic transaction ecosystems, *Information Sciences* (2018), doi: 10.1016/j.ins.2018.05.022

CSP-E^2: An abuse-free Contract Signing Protocol with low-storage TTP for energy-efficient electronic transaction ecosystems

Guangquan Xu[a], Yao Zhang[a,*], Arun Kumar Sangaiah[b], Xiaohong Li[a], Aniello Castiglione[c], Xi Zheng[d]

[a] *Tianjin Key Laboratory of Advanced Networking (TANK),School of Computer Science and Technology, Tianjin University,Tianjin,300350, China*
[b] *School of Computing Science and Engineering, Vellore Institute of Technology (VIT), Vellore-632014, Tamil Nadu, India.*
[c] *Department of Computer Science, University of Salerno, Italy*
[d] *Department of Computing, Macquarie University, Australia*

Abstract

In an increasingly connected world, energy efficiency has become a significant problem. Online electronic transactions are booming based on shopping convenience, fast payments, and lower prices. Electronic trading systems are one of the main sources of energy consumption because the so-called contract signing protocol (CSP) is a crucial part of e-commerce. Therefore, the energy consumption is largely determined by the efficiency of the CSP. To this end, an abuse-free CSP with low-storage TTP is proposed to eliminate the enormous energy-consumption problem originating from the unfit protocol design. This protocol exploits a proxy key scheme and introduces a lightweight TTP storage policy that greatly reduces the TTP burden by integrating symmetric encryption aiming at constructing an energy-efficient electronic transaction ecosystem. Theoretical analysis and comparative analysis show that the storage requirement of TTP is reduced by more than 50% while ensuring fairness, optimism and abuse-freeness in our protocol. Moreover, the exchange steps of our protocol are reduced so that the protocol efficiency is improved.

Keywords: energy-efficiency, contract signing protocol, low-storage TTP, abuse-freeness, proxy key

1. Introduction

With the rapid development of the Internet, online shopping and payment have become a part of our daily lives. As a result, energy efficiency in online electronic transactions has become a crucial problem. When a buyer uses an online payment method (e.g., Alipay or WeChat), he/she makes an E-Contract with the seller via a contract signing protocol(CSP) [15, 2, 5]Z. Therefore, the efficiency of electronic transaction system [20, 19, 21] largely depends on the CSP performance.

A CSP allows two parties (e.g., Alice and Bob) to each declare their commitment to an agreed contract over the Internet. A practical CSP should ensure that the transaction is fair and optimistic. In most scenarios, contract signing involves an exchange of signatures on the given contract. Compared to contract signing in real life, Ryo and Fujisaki [15] noted that there are some difficulties in designing a practical CSP for online electronic transactions. One reason is that information exchange over the Internet is not simultaneous.

*Corresponding author.
Email addresses: losin@tju.edu.cn (Guangquan Xu), zzyy@tju.edu.cn (Yao Zhang), arunkumarsangaiah@gmail.com (Arun Kumar Sangaiah), xiaohongli@tju.edu.cn (Xiaohong Li), castiglione@acm.org (Aniello Castiglione), james.zheng@mq.edu.au (Xi Zheng)

To ensure the security of electronic transactions, most CSPs involve a trusted third party (TTP), such as an e-commerce platform. We classify CSPs based on the TTP's participation level. There are three types of CSPs: 1) a protocol without a TTP; 2) a protocol with an on-line TTP; and 3) a protocol with an off-line TTP. The protocol without a TTP exchanges signature information bit by bit, which ensures that the protocol is fair but is not efficient [25, 6] because the TTP needs to be involved in every exchange, which results in a high communication cost and low efficiency [28, 13]. The most commonly used protocol uses an off-line TTP to achieve fairness, optimism and efficiency [1, 16, 24, 26]. Using an offline approach, the TTP is included only in the registration and dispute-resolving protocols, which reduces the amount of communication. The off-line TTP mechanism can guarantee fairness: at the end of the protocol, either both parties have valid contract signatures or neither does.

However, due to the special nature of electronic transaction systems, the traditional CSPs are not appropriate. Fortunately, the new protocol proposed in this paper can simultaneously satisfy the e-commerce requirements for fairness, optimism, efficiency and abuse-freeness.

Fairness. Fairness is the fundamental property of a CSP and indicates that both sides of a protocol obtain a signed contractat the end of the protocol or that neither does. Many existing protocols are unfair. For example, Park et al.'s scheme [16] cannot resist inference attacks in which an honest-but-curious arbitrator can easily acquire the signer's secret key. Qin et al.'s scheme [24] cannot resist replay attacks. To enhance fairness, our protocol solves these two problems.

Optimism. Our scheme uses an off-line TTP that becomes involved only when a dispute occurs. Note that disputes are rare because a dishonest party cannot obtain additional benefits from the dispute. Hence, our scheme achieves the property of optimism, i.e., the TTP is rarely included in the protocol.

Efficiency. Efficiency involves a faster signing process, reduced storage, less communication and lower computation cost. In this protocol, the TTP stores only its private key, which dramatically the TTP 's burden.

Abuse-freeness. Abuse-freeness is an important CSP property. When one party can prove the validity of the intermediate result, that party can utilize the result to seek additional benefits.

In this paper, we contribute to the CSP for electronic transactions in the following aspects:

- In our CSP-E^2 scheme, we introduce a Schnorr signature as the signature method. To ensure fairness, we split the Schnorr private key into two partial keys. In this way, the TTP can obtain only a partial private key; it cannot obtain the complete signature of the sender, Alice.
- We present a low-storage TTP model as an arbitrator. In the traditional CSP protocol, a TTP is required to store the messages of the involved parties, i.e., Alice and Bob. However,because of our symmetrical encryption scheme, the TTP does not need to conserve any messages except for its own private key.
- We also propose a proxy key scheme to achieve abuse-freeness and prevent inference attacks. Before the protocol terminates, either TTP or Bob can obtain only the Alice's proxy signature. Therefore, the TTP cannot infer Alice's private key, and Bob is unable to abuse Alice's partial signature.

The rest of this paper is organized as follows. Section 2 briefly reviews the off-line TTP scheme, Schnorr signature scheme, key decomposition scheme and proxy key scheme. Section 3 presents our new low-storage TTP and abuse-free CSP. Section 4 analyzes the performance of the new protocol. Finally, section 5 provides a real-world scenario, and section 6 concludes this work.

2. Related work

Traditional off-line TTP CSPs have some flaws, i.e., we cannot guarantee the absolute trust of the TTP. Therefore, we cannot allow the TTP to have the ability to generate complete signature information. To solve this problem, Park et al. [16] proposed a protocol based on the RSA signature scheme. In this scheme, the sender only sends part of the private and public keys to the TTP for verification, and the TTP can only generate a partial signature. However, Dodis and Reyzin [11] noted that the protocol is not secure because a curious TTP can deduce the whole private key from the information it receives. To ensure the security of the protocol, Huang et al. [11] and Wang [26] each improved the protocol based on Park et al.'s work.

The improvement of Dodis et al.'s scheme is based on the "Gap Diffie-Hellman [10], which can guarantee the security of the protocol, but the computational complexity is unacceptable. Although previous works, including those of Park's and Dodis', are fair and optimistic, the greatest problem is the vulnerability to inference attack. That is, an honest-but-curious arbitrator can easily figure out the signer's secret key. To solve this problem, Wang [26] concealed the partial public key sent to the TTP by applying a zero-knowledge proof and trapdoor commitment scheme instead of sending both the partial public and private keys. However, this method increases computation cost and communication cost tremendously. Zhang et al. [27] first proposed a new property, abuse-freeness, and the property is implemented in his work based on a discrete logarithm. Based on the RSA signature scheme, Wang [26] subsequently improved upon Park et al.'s work [16] to achieve abuse-freeness. Since Zhang et al. [27] and Wang [26] have introduced protocols to achieve abuse-freeness, increasing attention has been paid to the property of abuse-freeness.

Abuse-freeness is a very important property in CSP that guarantees that both protocol sides cannot seek benefits from the intermediate results. As an real example, Alice wants to apply to a university to pursue her graduate degree. Two universities, AU and BU, extend offers to Alice. AU provides a larger scholarship, but Alice does not like its geographic position; by contrast, BU gives a smaller scholarship, but its location is more desirable to Alice. Alice can then show the scholarship that AU offered her to BU to seek greater benefits. In this situation, Alice takes advantage of the protocol security flaws, which is defined as abuse-freeness.

Early research work mainly focused on the fairness and optimism of the protocol. Recent work [24, 8, 7, 12, 25] has begun to pay attention to the efficiency and abuse-freeness of CSP. An obstacle for CSP is its efficiency issues. One of the obstacles for CSP is its efficiency issues. However, it is difficult for existing relevant protocols to guarantee those performances simultaneously. Our research ensures that the protocol is fair and optimistic while ensuring abuse-freeness and efficiency based on the Schnorr signature. Moreover, previous work did not pay attention to the TTP's storage problem. The TTP in these works often require excessive storage overhead to, e.g., conserve the message or public/private keys of other parties. To solve this problem, our work further achieves a low-storage

TTP that does not need to store any information except for its own private key. To improve the efficiency of the CPS in electronic trading systems, the low-storage TTP ensures a high utilization rate of storage.

3. Preliminaries

In this section, we highlight the most relevant methods for CSP.

3.1. Protocol with off-line TTP

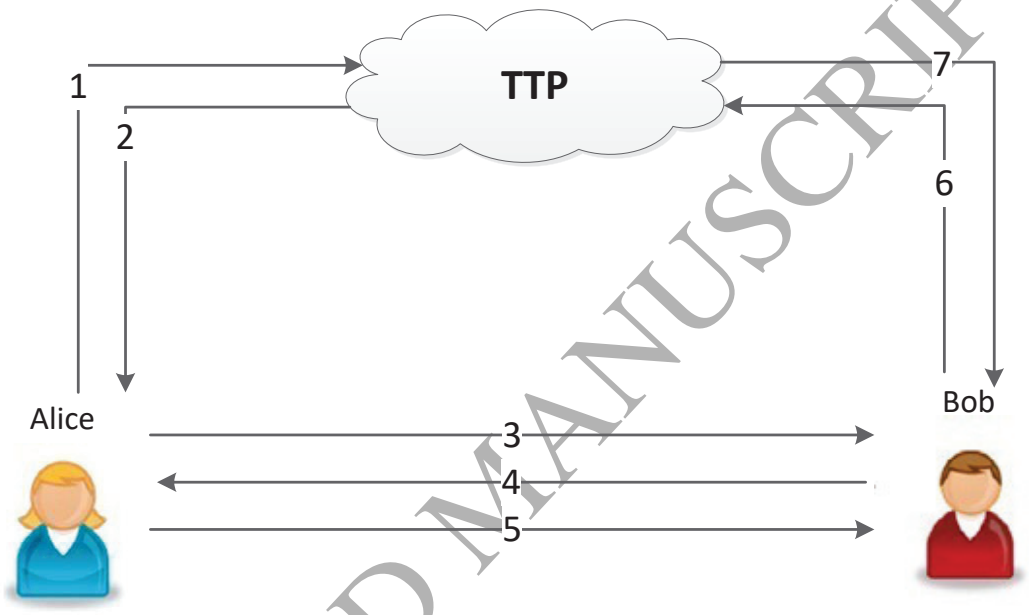

Figure 1: The process of CSP with off-line TTP

The most common protocols use off-line TTP to improve performance [3, 16, 24, 26]. The general flow of offline TTP is shown in Figure 1, where Alice and Bob are the two parties to the signature exchange. First, Alice sends her partial private key to TTP, and the TTP sends a voucher to Alice. Then, in the third through fifth steps, as shown in Figure 1, Alice and Bob follow the signature exchange protocol, in which they exchange signatures normally without invoking the TTP. If a dispute arises, Bob can send his signature and the information received from Alice to alert the TTP to arbitrate the problem. After the TTP verifies the message from Bob, it sends Alices valid signature to Bob.

3.2. Schnorr Signature

The Schnorr Signature[23] is based on the intractability of the discrete logarithm problem. Stern and Pointcheval [18] subsequently proved the security of this signature mechanism. We briefly introduce this algorithm in this section. The Schnorr signature involves the following basic notations:

- p, q: p and q are large primes that meet the conditions $q|p-1$, $q \geq 2^{140}$and $p \geq 2^{512}$.
- g: $g \in \mathbb{Z}_p^*$ and $g^q = 1 \bmod p$.
- H: $\{0,1\}^* \to \mathbb{Z}_p$, hash function.
- r: $r \in \mathbb{Z}_q^*$, $0 < r < q$, private key.
- u: $u = g^r \bmod p$, (g, u, p, q) is the public key.

The main process of the Schnorr signature is as follows:

- Selection: Select a random number $a \in {}_q$.
- Signature: Calculate $\sigma = g^a \bmod p$, $e = \mathrm{H}(\mathrm{H}(\mathrm{M}) \parallel \sigma)$, $s = \mathrm{a} + \mathrm{r}e \bmod q$, where $\parallel$ denotes concatenation.
- Sending: Send (s, e) to the verifier to ensure that (s, e) is the signature of message M.

After receiving the Schnorr signature, we need to verify its availability. We first define a function $v = Verify(s, e, u)$, then calculate $r\prime = g^s u^{-e} \bmod p$,$e\prime = \mathrm{H}(\mathrm{H}(M) \parallel r\prime)$. When $e\prime = e \bmod p$, then $v = yes$, i.e., the signature is valid. When $e\prime \neq e \bmod p$, then $v = no$, i.e., the signature is invalid.

3.3. Key decomposition

Assuming that Alice and Bob want to sign a contract, Alice first splits her private key d into d_1 and d_2, where $d = d_1 + d_2 \bmod \phi(n)$. According to the RSA signature rule, ${e_1} = {d_1}^{-1} \bmod \phi(n)$ and ${e_2} = {d_2}^{-1} \bmod \phi(\mathrm{n})$. Alice subsequently sends only e_1 and d_2 to the TTP allowing the TTP to validate σ_1 by e_1 and generate the σ_2 by d_2, where $\sigma = \sigma_1 + \sigma_2$.

This key decomposition ensures that the TTP cannot generate the complete signature while ensuring fairness. However, there is a security flaw: a curious TTP can independently deduce the private key from the information it receives.

Wang [26] improved Park et al.'s protocol by hiding the partial public key e_1. In Wang's protocol, Alice sends only d_2 and some other information to the TTP; thus, the TTP cannot deduce Alice's private key. In addition, Wang [26] first achieved abuse-freeness based on the RSA signature. However, Wang integrated an interactive zero-knowledge protocol and trapdoor commitment scheme to ensure the protocol's abuse-freeness, Which cause the computation and communication cost to become more expensive. For more details, please refer to [16, 26].

Our scheme also requires key decomposition based on the Schnorr signature. The private key d is split into r_1 and r_2 , where $r = r_1 + r_2 \bmod q$. The random number a can be split into a_1 and a_2, where $a = a_1 + a_2 \bmod p$. Alice can then calculate the corresponding signatures $s_1 = \mathrm{a}_1 + \mathrm{r}_1 e \bmod q$ and $s_2 = \mathrm{a}_2 + \mathrm{r}_2 e \bmod q$. According to the Schnorr signature rule, the public key is $u = g^r \bmod q$. If Bob receives u_1 and u_2, he can obtain the complete signature by calculating $u = u_1 + u_2 \bmod p$. To ensure efficiency, we introduce the proxy key method, a lightweight scheme that ensures our protocol remains abuse-free and improves safety. More information about the proxy key scheme is provided in the following section.

3.4. Proxy key and signature

The concept of a proxy signature was first introduced by Mambo et al.[22]. A proxy signature is useful in scenarios in which a signer cannot sign a contract personally for some reasons. Extensive research work has been conducted in the area of proxy signatures [24, 9]. Based on the Schnorr signature, Shi et al.[24] proposed a new protocol to exchange signatures via a proxy. However, this scheme has some problems: it requires Alice to interact with the TTP frequently, and it cannot resist replay attacks. Our proxy key scheme is based on the Schnorr signature but is both efficient and safe. We next briefly present this scheme:

- r_1 and Alice's r_2 are partial private keys, where $r = r_1 + r_2 \bmod q$.
- r_1',r_2': $r_1' = r_1 + d_1 \bmod q$ and $r_2' = r_2 + d_2 \bmod q$,$d_1, d_2 \in \mathbb{Z}_q^*$, proxy partial private keys.
- u_1',u_2' : $u_1' = g^{r_1'}$, $u_2' = g^{r_2'}$, proxy partial public keys. The signing process of the proxy key Schnorr signature is as follows:
- Selection: Select a random number $a \in {}_q$.
- Proxy signature: $S_{\omega_1}' = (M_\omega \times \mathrm{r}_1' + a_A) \bmod q$, $S_{\omega_2}' = (M_\omega \times \mathrm{r}_2') \bmod q$.

If a receiver obtains S_{ω_1}' and S_{ω_2}', it can restore the valid signature S_ω by calculating $\sigma_\omega = g^{a_A} \bmod p$.

4. CSP-E^2: Contract Signing Protocol With Low-Storage TTP for Energy-efficient Electronic Transaction Ecosystems

We now describe the details of our CSP. We begin with an outline and then describe the establishment of three subprotocols.

4.1. The general framework of CSP-E^2

Our protocol consists of three subprotocols as shown in Figure 2: the registration protocol, CSP and dispute resolution protocol. This scheme is based on the Schnorr signature, and we use the proxy key to enhance security.

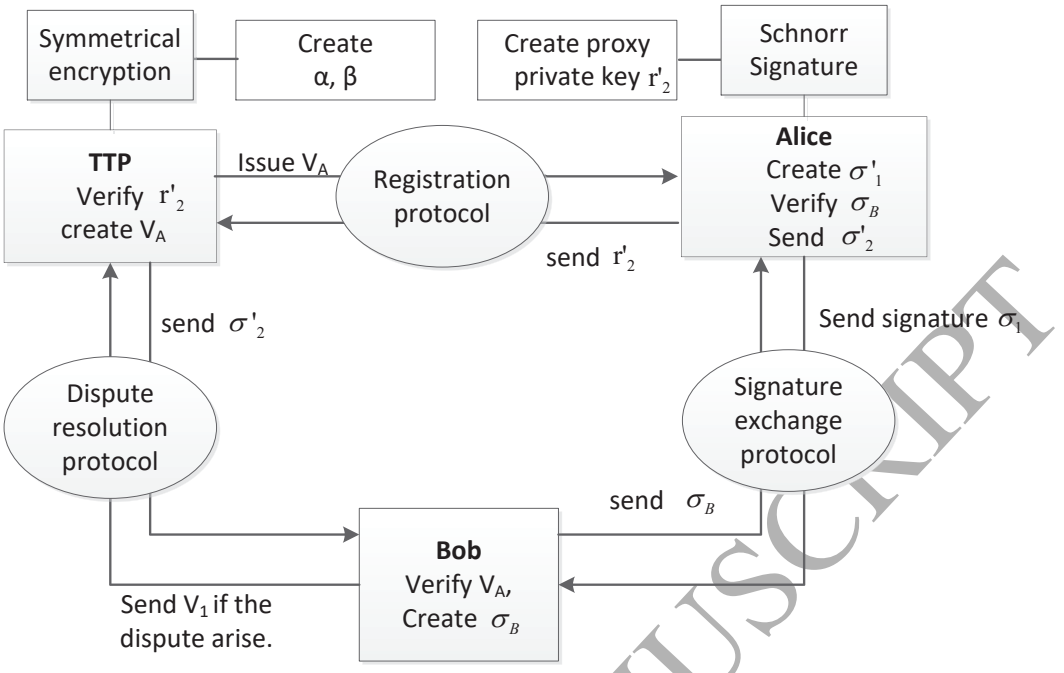

Figure 2: The general framework of CSP-E^2

To achieve abuse-freeness, based on the Schnorr signature, Alice first splits her private key r into two partial private keys r_1 and r_2, as described in [24], and then generates r_1's proxy key $r_1{}'$ and r_2 's private key $r_2{}'$. More details about $r_1{}'$ and $r_2{}'$ will be introduced in the following section. Alice then sends $r_2{}'$ to the TTP for registration. The TTP needs to verify $r_2{}'$ and C_A (C_A will be introduced in the next section). If the verification succeeds, the TTP generates a voucher V_A for Alice. Note that to achieve a low-storage TTP, the TTP must create two secret messages α and β, where $\alpha = \mathrm{r}_2{}' + k_A \bmod q$, $\beta = d + k_A \bmod q$ and α, β are included in V_A. The TTP then sends the voucher V_A to Alice. In the exchange protocol, Alice needs to send C_A, V_A and her partial proxy signature $S_1{}'$ to Bob. After Bob receives $S_1{}'$, he needs to verify $S_1{}'$ by the partial public key $u_1{}'$. If the verification suceeds, Bob generates his signature S_B and sends it to Alice. Upon receiving S_B and confirming the correctness of S_B, she sends her signature S_A to Bob. If S_A is invalid or Alice does not send S_A, Bob executes the dispute resolution protocol to obtain the valid signature S_A from the TTP. In the following section, we describe the details of the three subprotocols of our CSP.

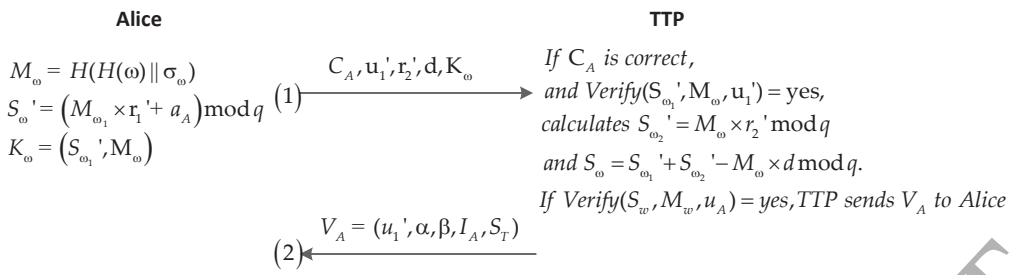

Figure 3: The registration protocol process in electronic transaction ecosystems

4.2. Registration protocol

To obtain authentication from the TTP, the initiator should first execute the registration protocol to obtain the voucher from the TTP prior to the process of signature exchange. The registration protocol process is depicted in Figure 3; the details are elaborated below.

In section 2, we described the Schnorr signature process; here, the meanings of the symbols remain unchanged. Initially, Alice sets p, q, and g and then randomly selects two numbers r_A and a_A, where r_A is Alice's private key and a_A is the parameter to generate a signature. Alice's public key u_A can be calculated by her private key r_A. Alice subsequently registers with the certificate authority (CA) to receive her certification C_A which includes u_A , g, p and q.

Then, Alice divides her key into two partial private keys r_1 and r_2, where $r_A = r_1 + r_2 \bmod p$. Alice then generates r_1's proxy key and r_2's proxy key, where $r_1' = r_1 + d_1 \bmod q$ and $r_2' = r_2 + d_2 \bmod q$. Next, Alice calculates $d = d_1 + d_2 \bmod q$, and r_1''s proxy public key $u_1' = g^{r_1'} \bmod p$ where $d_1, d_2 \in \mathbb{Z}_q^*$. Alice also randomly selects a sample message ω and calculates $\sigma_\omega = g^{a_A} \bmod p$ and $M_\omega = H(H(\omega) \parallel \sigma_\omega)$ as well as $S_{\omega_1}' = (M_\omega \times r_1' + a_A) \bmod q$, such that Alice's partial signature on ω is $K_\omega = (S_{\omega_1}', M_\omega)$. Finally, Alice sends $(C_A, u_1', r_2', d, K_\omega)$ to the TTP.

Upon receiving $(C_A, u_1', r_2', d, K_\omega)$ from Alice, the TTP first checks whether C_A is a valid certification for Alice. Then, the TTP needs to verify the partial proxy signature via the function $v_{w_1} = Verify(S_{w_1}', M_w, u_1')$. When $v_{w_1} = yes$, the TTP generates Alice's partial proxy signature $S_{w_2}' = M_w \times r_2' \bmod q$ and calculates $S_w = S_{w_1}' + S_{w_2}' - M_w \times d \bmod q$. Then the TTP executes the function $v_w = Verify(S_w, M_w, u_A)$. When $v_w = yes$, the TTP is required to generate a secret symmetric key by calculating $k_A = H(I_A \parallel r_T)$, where I_A Alice's identity and r_T is the private key of the TTP. Then, the TTP calculates $\alpha = (r_2' + k_A) \bmod q$ and $\beta = (d + k_A) \bmod q$. Finally, the TTP generates the voucher for Alice, i.e.$V_A = (u_1', \alpha, \beta, I_A, S_T)$, where $S_T = sign(u_1' \parallel \alpha \parallel \beta \parallel I_A, r_T)$, and TTP sends V_A to Alice.

We provide some notes on the registration protocol introduced above. In the preparation step, Alice needs to register with the CA to obtain the certification CA so that she can verify that here corresponding public and private keys are valid. However, to increase the reusability of the Schnorr signature key, if Alice has received a certificate C_A from CA, she does not need to repeat the registration for a long period of time. In the generation step, Alice needs to set a sample message ω and generate the corresponding signature, which is signed by r_2'. In

this way, Alice can demonstrate the validation of $u_1{}'$ to the TTP. In the registration step, the TTP calculates the partial signature $S_{\omega_2}{}'$. The TTP can subsequently use $S_{\omega_1}{}'$, $S_{\omega_1}{}'$ and d to calculate $S_{\omega_1}{}'$. After verifying $S_{\omega_1}{}'$, the TTP can check whether $r_2{}'$ is Alice's valid partial proxy private key. Finally, to achieve the low-storage, the TTP generates its secret symmetric key K_A. Furthermore, the TTP encrypts $r_2{}'$, d, and K_A into α and β. If the dispute resolution protocol is executed, TTP can obtain α and β from Bob. Therefore, the TTP does not need to keep any information except its private key. Finally, the TTP generates V_A and sends it to Alice, to verify that she has registered with the TTP.

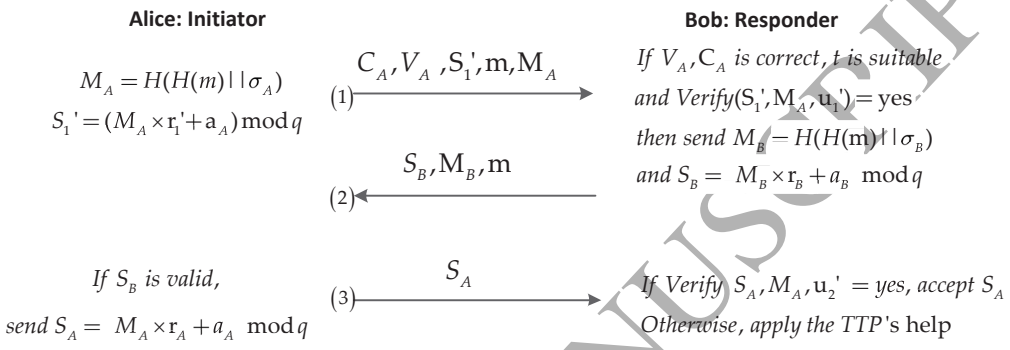

Figure 4: The signature exchange protocol of CSP

4.3. Signature exchange protocol

Before the two involved parties exchange signatures, Alice and Bob need to negotiate the contract m, which includes the identities of the involved parties and the TTP as well as a deadline T. The signature exchange process is shown in Figure 4, and a detailed description follows.

Generate partial proxy signature: Alice first computes $\delta_A = g^{a_A} \bmod p$, and $M_A = H(H(m) \parallel \delta_A)$. Then, she generates her partial proxy signature $S_1{}'$ by computing $S_1{}' = M_A \times r_1{}' + a_A \bmod q$. Finally, she sends $(C_A, V_A, S_1{}', m, M_A)$ to Bob.

After receiving $(C_A, V_A, S_1{}', m, M_A)$ from Alice, Bob needs to verify whether C_A is valid. Then, Bob checks whether the S_T included in V_A is signed by the TTP. If the conclusion is yes, Bob can confirm that V_A was generated by the TTP. Furthermore, Bob is required to check the identities of Alice, Bob and the TTP in contract m so that Bob can verify that m is written correctly. Additionally, Bob needs to check whether the deadline T is suitable.

If all of this goes well, Bob is required to verify the partial proxy signature $S_1{}'$ by $v_1 = Verify(S_1{}', M_A, u_1{}')$. When $v_1 =$ yes, Bob generates his signature by computing $\delta_B = g^{a_B} \bmod p$ and $M_B = H(H(m) \parallel \delta_B)$, as well as $S_B = M_B \times r_B + a_B \bmod q$ and Bob sends (S_B,M_B, m) to Alice.

Verify Bob's signature: Upon receiving (S_B,M_B, m) from Bob, Alice checks whether S_B is Bob's valid signature using $v_B = Verify(S_B, M_B, u_B)$. When $v_B = yes$, she computes her signature $S_A = M_A \times r_A + a_A \bmod q$ and sends S_A to Bob.

After receiving S_A from Alice, Bob verifies the signature using $v = Verify(S_A, M_A, u_A)$. When the verification

$v = yes$, the signature exchange has been executed successfully. However, if Bob did not receive S_A for some reason or the verification $v = no$, Bob can execute the dispute resolution protocol, which is introduced in the next section.

There are some notes that apply to this section. In step 3.3.1, Alice only sends her partial proxy signature $S_1{}'$ and some other verification information to Bob because the TTP can generate the complete signature based on this information. IIn step 3.3.2, a deadline T is attached to commitment m to ensure that the signing process is completed by the deadline. In addition, including the deadline reduces replay attacks.

4.4. Dispute resolution protocol

After executing the exchange protocol, Alice has acquired Bob's signature. However, Bob may not receive Alice's signature S_A, or S_A may not be valid. In that case, Bob is required to turn to the TTP to obtain Alice's real signature before the deadline T. The detailed procedures are as follows.

- First, Bob sends $(C_A, V_A, S_1{}', m, M_A, S_B, M_B)$ to the TTP. Upon receiving the message from Bob, the TTP first validates C_A and V_A. Then, the TTP verifies whether the S_B is available using $v_B = Verify(S_B, M_B, u_B)$.
- When the verification $v_B = yes$, the TTP decrypts α and β to obtain $r_2{}'$ and d via the equation $r_2{}' = \alpha - k_A \bmod q$ and $d = \beta - k_A \bmod q$. Then, the TTP calculates Alice's proxy partial signature by computing $S_2{}' = M_A \times r_2{}' \bmod q$. Next, the TTP obtains Alice's signature S_A by computing $S_A = S_1{}' + S_2{}' - M_A \times d \bmod q$. Finally, the TTP sends S_A to Bob and S_B to Alice.

Note that, in the dispute resolution protocol, the two variables α and β act as the secret keepers. These two variables are generated by the TTP in the registration protocol described in section 3.2. Then, the TTP calculates the symmetric key k_A, which is used to encrypt $r_2{}'$ and d, according to $k_A = H(I_A \parallel r_T)$. The TTP subsequently decrypts α and β to obtain $r_2{}'$ and d via the symmetric key k_A. Using this approach, the TTP does not need to store any information except for its private key r_T; thus, in our scheme, the TTP in our scheme can achieves the low-storage property.

5. Discussion

We now examine the security and efficiency properties of the new protocol.

5.1. Fairness

Fairness is a fundamental issue to be considered in a CSP and means that neither of the two signers can obtain more benefits than the other, even if he/she is dishonest in the process. We can classify situations involving unfairness into two scenarios. One is that Alice is honest while Bob is cheating. Another is that Bob is honest while Alice is cheating. Next, we discuss these two cases.

In the first case, where Alice is honest while Bob is dishonest, based on the research of Pointcheval et al. [18, 17] and Morita et al. [14], we can conclude that the partial signatures $S_1{}'$ and $S_2{}'$ cannot be forged by others, i.e., only Alice can generate $S_1{}'$, and only Alice and the TTP can generate $S_2{}'$. In the CSP, Bob receives $(C_A, V_A, S_1{}', m, M_A)$ from Alice. Bob then decides whether to send his signature S_B to Alice or not. If Bob sends his signature S_B to Alice, Alice will send her signature S_A to Bob because Alice is honest. Therefore, both Alice and Bob will receive

the signature of the other party. If Bob does not send his signature or sends an invalid signature to Alice, he cannot obtain Alice's signature. Moreover, Bob cannot generate Alice's complete signature because he has only the partial proxy signature $S_1{}'$. To receive Alice's complete signature, Bob can ask the TTP for help. However, in the dispute protocol, Bob must send his valid signature S_B to TTP. If Bob does not send his valid signature, he will not obtain Alice's signature from the TTP.

In the second case, where Alice is dishonest, while Bob is honest, to receive Bob's signature during the CSP, Alice must send her partial proxy signature $S_1{}'$ to Bob. If $S_1{}'$ is invalid, Bob will not return his signature S_B to Alice. After Alice sends $S_1{}'$ to Bob, Bob will send his signature S_B to Alice because Bob is honest. Upon receiving S_B from Bob, Alice can either send her signature S_A to Bob or not. If Alice sends her signature to Bob, both parties will receive the signature from the other party. If Alice does not send her signature to Bob, Bob will ask the TTP for help.

Based on the above discussion, we can conclude that our protocol does not favor any dishonest parties; that is, our protocol is fair.

5.2. Abuse-freeness

Abuse-freeness is an important property of CSP. During the protocol, neither of the involved parties can prove to an outside party that the intermediate results received are available. Many existing protocols can achieve such a property. For example, Wang [26] achieves abuse-freeness via a zero-knowledge protocol and trapdoor commitment scheme. However, we employ a proxy signature to make our scheme not only abuse-free but also efficient. That is, if Bob receives the five tuple $(C_A, V_A, S_1{}', m, M_A)$ from Alice, he might be able to prove the validation of the intermediate result to a third party. However, Bob can only prove that the partial proxy signature $S_1{}'$ is valid. In other words, even if the partial proxy signature $S_1{}'$ can be proved to an outside party Charlie, Charlie cannot obtain either Alice's real partial signature or Alice's entire signature because $S_1{}'$ is only an encryption result of the partial signature S_1. Therefore, Bob cannot convince Charlie that he has the a valid intermediate result, e.g., Alice's partial signature S_1. Thus, our protocol achieves abuse-freeness.

5.3. Low-storage TTP

In our protocol, the TTP has low storage requirements because it does not need to keep any information except for its private key. In contrast, the TTPs in previous schemes [16, 24, 26] are all required to maintain far more information, (e.g., partial public or private keys, partial signatures and so forth). During the registration process, Alice sends $(C_A, u_1{}', r_2{}', d, K_\omega)$ to the TTP; however, after verifying the information, the TTP does not store any information except for its private key. The TTP then returns $V_A = (u_1{}', \alpha, \beta, I_A, S_T)$ to Alice to complete the registration. When a dispute occurs, the TTP can generate Alice's partial signature to help Bob obtain Alices signature only as discussed in the previous section.

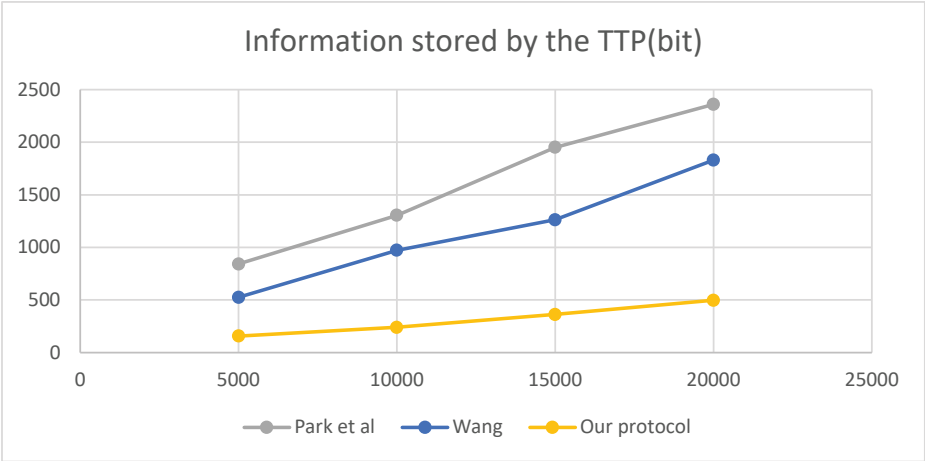

Figure 5: Information stored by the TTP (bits)

We compare the information stored by TTP with the schemes of Park et al. and Wang. The private key of the TTP in our protocol is 512 bits and the length of the RSA modulus n is 1,200 bits. We also calculate the average size of the stored information in the protocol based on the RSA signature. As shown in Figure 5, the information stored by the TTP in our protocol is far less than in the protocols of Park et al. and Wang. More precisely, our protocol reduces the storage requirements by 68% compared with Wangs work and by 79% compared with Park et al.s work. Overall, in our protocol, the TTP can fulfill its role without storing too much information.

5.4. Efficiency

In this section, we analyze the efficiency of our protocol. We analyze the efficiency only in the normal case (i.e., the dispute resolution protocol is not included in this discussion). We consider the number of modular exponentiations, which is a type of exponentiation performed over a modulus. The modular exponentiation operation computes the remainder of an integer b (base) raised to an i-th power (called the exponent) b^i and divided by a positive integer n (known as the modulus). As an example in symbol form: given the base b, the exponent i, and the modulus n, the modular exponentiation M is $M \equiv b^i (\mathrm{mod} n)$. Because modular exponentiation is by far the most computationally expensive cryptographic operation, we consider modular exponentiation as the main factor when evaluating the efficiency of the new protocol. For comparison, we select the methods of Shi et al. [24], Wang [26] and Ateniese [4] as baseline methods. We calculate the number of modular exponentiations during the process of CSP.

	Wang	Park et al.	Q.Shi et al.	Ateniese	Our protocol
Fairness	Yes	Yes	Yes	Yes	Yes
Abuse-freeness	Yes	No	Yes	No	Yes
Timely termination	Yes	No	No	Yes	Yes
Replay attack	Yes	No	No	Yes	Yes
Efficiency	low	Middle	Middle	Low	High

Table 1: Comprehensive comparison

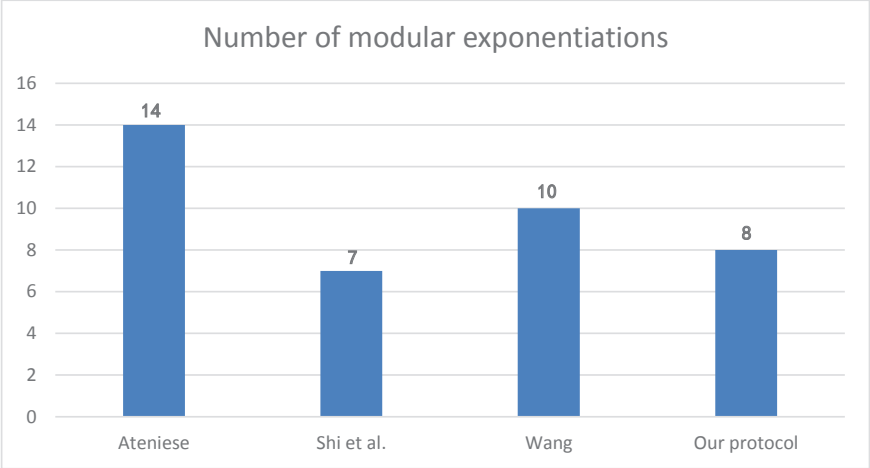

Figure 6: Number of modular exponentiations

As shown in Figure 6, our protocol's computational efficiency is moderate. Compared with the work of Ateniese [4] and Wang [26], the computational cost of our scheme is 43% and 20% lower, respectively. Compared with Shi et al.'s work, our scheme's computational cost is 14% higher, but this increase is not meaningful because 1) our scheme achieves low-storage TTP (i.e., the TTP in our protocol does not need to store much information); 2) after registering with the TTP, Alice does not need to register again for a long period, which obviously reduces the communication costs; and 3) our scheme ensures that the protocol can be completed or ended in a reasonable time. Moreover, our mechanism strengthens the defense against replay attacks. Therefore, even though the number of modular exponentiations required by our protocol is slightly higher than that in Shi et al.'s method, our protocol is superior with respect to TTP efficiency and safety. That is, the comprehensive performance of our protocol is superior to that reported in previous works.

5.5. Comprehensive comparison

In this section, we comprehensively compare our protocol with related protocols. The analysis of different protocols follows CSP between two parties.

As shown in Table 1, the protocols of Park and Shi do not satisfy the requirements of timely termination and defense against replay attack. In particular, neither Parks protocol nor Atenieses protocol are abuse-free, and the

latter's efficiency is lower than those of the others. Wangs scheme is also good, but it requires seven steps; therefore, it is not as efficient. We can conclude that our protocol is excellent not only in terms of security but also in terms of efficiency compared with the baseline approaches. Thus, our protocol is more suitable for electronic transactions, which require greater safety and efficiency.

6. Case study

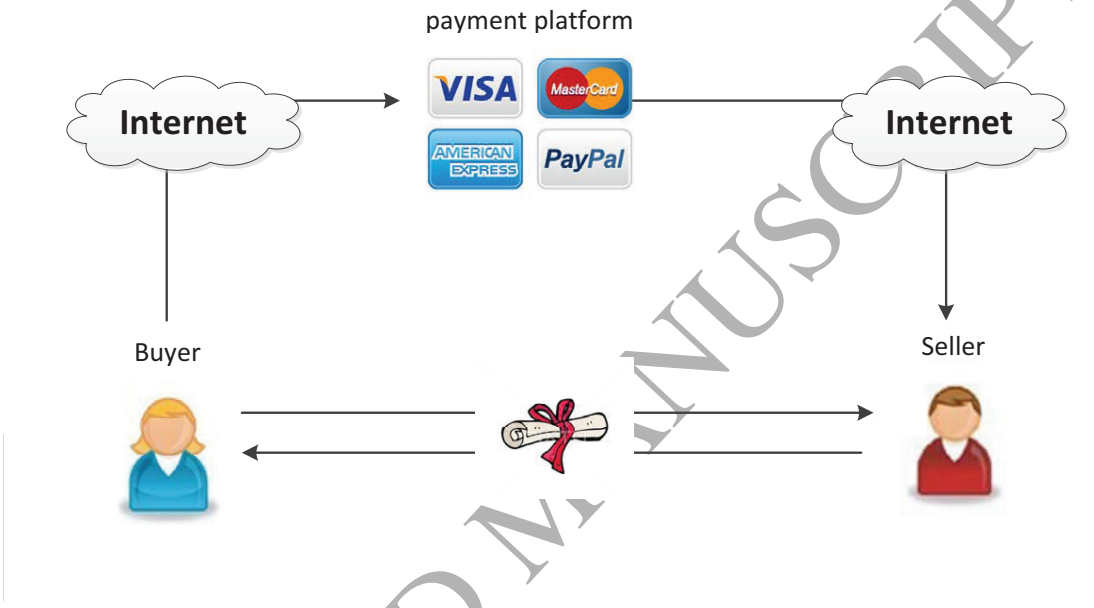

Figure 7: The process of signature exchange between taxis

Next, this paper presents a practical application of a CSP in electronic transaction ecosystems. A CSP in an electronic transaction ecosystem is considerably more complicated than is a CSP in a traditional network environment. The protocol's performance is affected not only by potential security problems but also by efficiency concerns. Reducing the communication delay requires more powerful computing power and a larger storage capacity under the electronic transaction ecosystem conditions.

With the rapid development of online payments and their intrinsic characteristics, a CSP for electronic transaction ecosystems is urgently needed. Based on its inherent features, the relationship between buyer and seller is primarily a trading relationship; thus, the online payment area may be the first field to apply the protocol. Without loss of generality, we assume that buyer A communicates with seller B, to exchange location information, goods information, customer information and other sensitive information requiring signatures. As shown in Figure 7, a CSP between the involved parties must meet the following requirements:

- Fairness: Buyer A and seller B need to communicate and they require the property of fairness: that is, either A and B receive each others signatures or A and B do not receive each others signatures. The payment platform acts as the TTP. Our protocol can guarantee that the communication process between A and B is

fair (i.e., there is no the case in which one party receives the signature of the opponent but the other does not).

- Abuse-freeness:During the communication process, the proposed protocol ensures that A and B cannot take advantage of the intermediate signatures to seek interests from third parties. Neither side of the communication can prove the validity of the intermediate results; thus, the intermediate signatures cannot be abused.
- Security: The A and B communication process must ensure the validity and security of the signature on the message: that is, the signer cannot deny the signature results, and third parties cannot forge the signatures or crack the signature encryption. The protocol proposed in this paper ensures signature validity while enhancing security to defend against various attacks.
- Efficiency: We improved the traditional CSP and reduced the exchange steps, computing cost and storage cost, thus increasing our protocol's suitability for use in online payments and electronic transaction ecosystems.

Electronic transaction ecosystems can be applied to a variety of trading environments such as a buyer dealing with a seller, a buyer dealing with a platform, and so on. In summary, our new protocol improves the traditional protocol with respect to security and more importantly, improves efficiency to make it more suitable for electronic transaction ecosystems.

7. Conclusion

In this paper, we proposed a new signature-exchange protocol based on the Schnorr signature in electronic transaction ecosystems. In contrast to most existing protocols, the new protocol is not only fair and optimistic but also abuse-free and more efficient, and it can be implemented in future systems. In particular, in our work, the TTP achieves the low-storage property. The theoretical and comparative analyses showed that our approach reduces the TTP's storage burden by 68% compared with Wangs work and by 79% compared with Park et al.s work while retaining the properties of security and optimism. All these improvements are critical for the development of electronic transaction ecosystems.

The new protocol could be implemented in electronic transaction ecosystems in the future. Therefore, our future work will focus on experiments with electronic transaction ecosystems. The theoretical analyses showed that our protocol has great advantages over traditional protocols for deployment in electronic transaction network environments. We will build an experimental environment to comprehensively evaluate the protocol.

Acknowledgments

This work is partially sponsored by the National Science Foundation of China (No. 61572355, U1736115), the Tianjin Research Program of Application Foundation and Advanced Technology (No. 15JCYBJC15700), the Fundamental Research of Xinjiang Corps (No. 2016AC015), and the State key development program of China(No. 2017YFE0111900).

References

References

[1] A. Al-Saggaf, L. Ghouti, Efficient abuse-free fair contract-signing protocol based on an ordinary crisp commitment scheme, Information Security Iet 9 (1) (2014) 50–58.

[2] A. Al-Saggaf, L. Ghouti, Efficient abuse-free fair contract-signing protocol based on an ordinary crisp commitment scheme, IET INFORMATION SECURITY 9 (1) (2015) 50–58.

[3] N. Asokan, V. Shoup, M. Waidner, Optimistic fair exchange of digital signatures, IEEE Journal on Selected Areas in Communications 18 (4) (1998) 593–610.

[4] G. Ateniese, Verifiable encryption of digital signatures and applications, ACM Trans.Inf.Syst.Secur 7 (2004) 1–20.

[5] B. B, J. MP, An efficient free fair contract signing protocol using otpk, 10th IEEE and IFIP International Conference on Wireless and Optical Communications Networks (WOCN) 1–5.

[6] H. Ferradi, R. Graud, D. Maimu, D. Naccache, D. Pointcheval, Legally fair contract signing without keystones, in: International Conference on Applied Cryptography and Network Security, 2016.

[7] T. Gaber, N. Zhang, Fair and abuse-free contract signing protocol supproting fari license reselling, in: New Technologies, Mobility and Security (NTMS), 2011.

[8] W. Gao, F. Li, B. Xu, An abuse-free optimistic fair exchange protocol based on bls signature, in: International Conference on Computational-Intelligence and Security, vol. 2, 2008.

[9] K. Ghorbani, M. Assar, M. Salmasizadeh, An optimistic fair exchange protocol for proxy signatures, in: 2014 7th International Symposium on Telecommunications, 2014.

[10] H. Huang, Authenticated key exchange protocol under computational diffiehellman assumption from trapdoor test technique, International Journal of Communication Systems 28 (2) (2015) 325–343.

[11] Q. Huang, D. W. S., W. Susilo, P2ofe: Privacy-preserving optimistic fair exchange of digital signatures, in: Rsa Conference, Cryptographers' Track, Ct-Rsa, 2014.

[12] B. Kordy, S. Radomirovic, Constructing optimistic multi-party contract signing protocols, in: Computer Security Foundations Symposium (CSF), vol. 8, 2012.

[13] Z. Liu, J. Pang, C. Zhang, Design and formal verification of a cem protocol with transparent ttp, Frontiers of Computer Science 7 (2) (2013) 279–297.

[14] H. Morita, J. Schuldt, T. Matsuda, G. Hanaoka, T. Iwata, On the security of the schnorr signature scheme and dsa against related-key attacks, ICISC 2015 (2015) 20–35.

[15] R. Nishimaki, E. Fujisaki, K. Tanaka, A multi-trapdoor commitment scheme from the rsa assumption, in: Australasian Conference on Information Security and Privacy, 2010.

[16] J. Park, E. Chong, H. Jay, I. Ray, Constructing fair exchange protocols for e-commerce via distributed computation of rsa signatures, in: Symposium on Principles of Distributed Computing, 2003.

[17] D. Pointcheval, J. Stern, Security proofs for signature schemes, Lecture Notes in Computer Science 1070 (1996) 387–398.

[18] D. Pointcheval, J. Stern, Security arguments for digital signatures and blind signatures, Journal of Cryptology 13 (3) (2000) 361–396.

[19] T. Qiu, R. Qiao, D. Wu, Eabs: An event-aware backpressure scheduling scheme for emergency internet of things, IEEE Transactions on Mobile Computing (99) (2017) 1–1.

[20] T. Qiu, A. Zhao, F. Xia, W. Si, D. Wu, Rose: Robustness strategy for scale-free wireless sensor networks, IEEE/ACM Transactions on Networking (99) (2017) 1–16.

[21] T. Qiu, K. Zheng, H. Song, M. Han, B. Kantarci, A local-optimization emergency scheduling scheme with self-recovery for smart grid, IEEE Transactions on Industrial Informatics PP (99) (2017) 1–1.

[22] Y. Qu, L. Deng, X. Bao, H. Huang, An efficient certificateless multi-proxy signature scheme without pairing, International Journal of Electronic Security & Digital Forensics 8 (2) (2016) 148.

[23] C. Schnorr, Efficient identification and signatures for smart cards, CRYPTO '89 Proceedings on Advances in cryptolog (1989) 239–252.

[24] Q. Shi, N. Zhang, M. Merabti, Fair signature exchange via delegation on ubiquitous networks, Journal of Computer and System Sciences 81 (2014) 615–631.

[25] Z. Wan, R. Deng, D. Lee, Electronic contract signing without using trusted third party. network and system security, in: Network and System Security, 2015.

[26] G. Wang, An abuse-free fair contract-signing protocol based on the rsa signature, IEEE Transactions on Information Forensics and Security 5 (2010) 158–168.

[27] L. Zhang, A. Wu, B. Qin, Identity-based optimistic fair exchange in the standard model, Security & Communication Networks 6 (8) (2013) 1010–1020.

[28] Y. Zhang, C. Zhang, J. Pang, S. Mauw, Game-based verification of contract signing protocols with minimal messages, Innovations in Systems & Software Engineering 8 (2) (2012) 111–124.