FAIR PAYMENT PROTOCOLS FOR E-COMMERCE

Hao Wang and Heqing Guo
School of Computer Science & Engineering, South China University of Technology, Guangzhou, China 510641.

Abstract: It has been widely accepted that fairness is a critical property for electronic commerce. Fair payment protocol is designed to guarantee fairness in a payment process over asynchronous network. Fairness means that when the protocol terminates, either both parties get their expected items, or neither does. In this paper we first present a new generic offline fair payment protocol with fairness, timeliness and invisibility of TTP. Then we introduce the property of abuse-freeness into electronic payment and implement a fair abuse-free payment protocol.

Key words: Electronic commerce, Offline payment, Fairness, Abuse-freeness

1. INTRODUCTION

Electronic payment system is the most important building block for electronic commerce. As classified by Asokan et al. [1], there are two types of electronic payment system: *cash-like* payment and *check-like* payment. In cash-like payment system, payer first withdraws a certain amount of money (e.g. electronic coins) for the payment process, when payee received the money, s/he can deposit those coins into the bank. But in check-like payment system, payer sends some certified document (e.g. electronic check) so that the payee can have the check paid through direct bank transfer. When these two types of payment systems are to be migrated into asynchronous network, the issue of fairness has to be well studied. Fairness means that when the electronic transfer terminates, either both parties get their expected items

(e.g. electronic check and its receipt), or neither does. Fair payment protocol is designed to guarantee fairness in electronic payment system on asynchronous network.

As suggested by Louridas in [16], fair protocol and requirements of its application domains should match, which means assumptions of the protocol must be rooted in the protocol's application scenario. For this reason, we first set up the application scenario for our fair payment protocols: company B (the client, denoted as Bob) is going to buy some electronic goods from company A (the merchant, denoted as Alice) and they have settled on the goods and the price. Now they need to finish the exchange of Bob's check with Alice's goods on a relative insecure and asynchronous network. Bob's check is composed of his bank-certified account information, payment information and can be validated only after signed by his signature. With that signed check, Alice can get her money paid from Bob's bank. Note that *anonymity* is not considered in this scenario and it will be discussed as a possible extension in Section 5. With this scenario set, we can make our protocols' assumptions explicitly stated (see Section 2).

To achieve fairness, Alice must send to Bob a non-repudiation evidence of origin (**NRO**) proving she has sent the goods. And Bob's check can be used as a non-repudiation evidence of receipt (**NRR**) proving he has received the goods. In addition, a trusted third party (TTP) must be involved when an error occurs. Because it is widely accepted that no deterministic fairness can be achieved without any third party exists. To achieve timeliness, a party (say Alice) can initiate the resolve or abort sub-protocol to terminate the exchange (success or failure). Resolve means to let the TTP decide whether the exchange can be succeeded. Alice run the abort protocol to prevent Bob from resolving at a later time she will not wait.

1.1 Related Work

In 1996, Asokan et al. [2] introduce the idea of optimistic approach and presents fair protocols with offline TTP, in which TTP intervenes only when an error occurs (network error or malicious party's cheating). Ever since then, subsequent efforts in this approach resulted in efficient and fair protocols (Asokan et al. [3], S. Kremer and O. Markowitch [14], we call them as *AK protocol*) that can guarantee that both parties can terminate the protocol timely while assuring fairness (called property of *timeliness*). Although they were attacked for some designing details (see [12]), their messages & rounds optimality (see [23] for detailed discussions) and basic building blocks (main protocol, resolve and abort sub-protocols) are well analyzed and widely accepted.

Offline TTP generates evidences different from those produced by the sender or the recipient, which make the protocol suffer from *bad publicity* [17]: "intervention of the TTP can be due to a network failure rather than a cheating party", and it may cause doubt on either party's honesty. *Invisible TTP* is first introduced by Micali [20] to solve this problem. The TTP can generate exactly the same evidences as the sender or the recipient. In this way, judging the outcome evidences and received items cannot decide whether the TTP has been involved. There are two way of thinking:

The first one is to use *verifiable signature encryption* (VSE). It means to send the signature's cipher encrypted with TTP's public key before sending the signature itself. And try to convince the recipient that it is the right signature and it can be recovered (decrypted) by TTP in case of errors. Asokan et al. [3], Bao et al. [6] and Ateniese [5] make use of this approach to realize invisibility of the TTP. But as Boyd and Foo [7] has pointed out, verifiable encryption is computationally expensive.

The other approach is to use *convertible signatures* (CS) and it is recently focused approach. It means to firstly send a partial committed signature (verifiable by the recipient) that can be converted into a full signature (that is a normal signature) by both the TTP and the signer. Protocols proposed by Boyd and Foo [7] and Markowitch and Kremer [17] are early efforts to use this approach to construct fair protocols. But the former protocol is not efficient computationally and suffers from relatively heavy communication burden (for its interactive verifying process); the latter one cannot generate standard signatures as final evidences. In particular, the CS scheme proposed by Boyd and Foo is to split multiplicatively the secret key of a standard RSA signature. Recently, Park et al. [22] propose a CS scheme which splits the key additively, and based on that, present a very efficient protocol in which the partial signature is non-interactively verifiable. But unfortunately, Dodis and Reyzin [10] break the scheme by proving the TTP can obtain Alice's entire secret key with only her registration information. Dodis and Reyzin also propose an efficient CS scheme based on GDH signature, but this scheme cannot directly be applied efficient enough to construct an abuse-free protocol (further discussed in Section 5).

Abuse-freeness, as a new requirement of fair protocols, is first mentioned by Boyd and Foo [7], and formally presented by Garay et al. [11]. And Garay et al. have also realized an abuse-free contract signing protocol. Based on the Jakobsson-Sako-Impagliazzo designated verifier signature [13], they introduce a new signature scheme called *Private Contract Signature* to realize this property. The protocol has been formally analyzed by Kremer and Raskin [15], Chadha et al. [9][8]. And based on their intensely formalized study, Chadha et al. present improved definition of abuse-freeness. Briefly, abuse-freeness means that before the malicious party (say Alice) gets her full

evidence, she cannot convince any outside party that Bob has participated in the protocol. This property is quite important, especially for critical scenarios like contract signing and fair payment (further discussed in Section 4).

Previous efforts studying the fairness issue in payment systems include Asokan et al. [2] and Boyd and Foo [7]. As discussed earlier, these two protocols are not efficient and practical enough as to recent advances in area of fair exchange.

1.2 Our Work

In this paper we first present a generic fair payment protocol based on AK generic protocol and an adaptation of the convertible signature scheme proposed by Mao et al. [19] *(MP signature).* The original CS scheme uses an interactive verification protocol that is not practical for fair protocols. So we propose the use of secure non-interactive zero-knowledge proof method. And we prove that the general payment protocol satisfies the three main desired properties: fairness, timeliness and invisible TTP.

But as the normal zero-knowledge proof is universally verifiable, which may introduce defects in abuse-freeness. To solve this problem, we use a non-interactive *designated verifier proof* method to implement a fair abuse-free payment protocol. Briefly, designated verifier proof means that the proofs can convince nobody except the designated verifier (say Bob) and its underlying statement is "Either θ is true or I can sign as Bob". In this way, outside parties will not believe θ is true as Bob can simulate this proof himself.

When implementing the protocols, we have incorporated the label and message construction design principles proposed by Gurgens et al. [12].

Finally, we discuss several possible extensions to our protocols, including: possibility of using other cryptographic tools, protecting privacy in the fair payment protocol, using our results to construct a new fair abuse-free contract signing protocol and other implementation options.

The remainder of the paper is structured as follows. In Section 2, we state our protocols' assumptions and their requirements. Section 3 presents the general fair payment protocol framework. Section 4 discusses the abuse-freeness and presents the fair abuse-free protocol. In Section 5, we give some remarks and outline the possible extensions. Some concluding remarks presented in Section 6.

2. PROTOCOL REQUIREMENTS AND ASSUMPTIONS

2.1 Requirement on Fair Payment Protocols

Five requirements for fair exchange has formulated by Asokan et al. in [4] and further discussed in [25]. But their requirement definitions haven't presumed new advances in recent years. And in [18] Markowitch et al. study many former fairness definitions and present a well-knitted definition. Based on these former works, we present a complete set of requirement definitions for fair payment protocols.

Definition 1 Effectiveness

A fair payment protocol is *effective* if (the communication channels quality being fixed) there exists a successful payment exchange for the payer and the payee.

Definition 2 Fairness

A fair payment protocol is *fair* if (the communication channels quality being fixed) when the protocol run ends, either the payer gets his/her expected goods and the payee gets the payment or neither of them gets anything useful.

Definition 3 Timeliness

A fair payment protocol is *timely* if (the communication channels quality being fixed) the protocol can be completed in a finite amount of time while preserving fairness for both payer and payee.

Definition 4 Non-repudiability

A fair payment protocol is *non-repudiable* if when the exchange succeeds, either payer or payee cannot deny (partially or totally) his/her participation.

Definition 5 Invisibility of TTP

A fair payment protocol is *TTP-invisible* if after a successful exchange, the result evidences of origin/receipt and exchanged items are indistinguishable in respect to whether TTP has been involved.

2.2 Protocol Assumptions

With the application scenario set, we state our protocol's assumptions as following:

- **No Self-mutilation** Either Alice or Bob will not take any action that would hurt his/her own benefit. This assumption is quite plain and is omitted in our later analysis.
- **Communication Channel** As many fair protocols do, we assume the resilient channels between exchangers (Alice/Bob) and TTP, and unreliable channel between Alice and Bob. Messages in a resilient channel can be delayed but will eventually arrive. On the contrary, messages in unreliable channel may be lost. We also assume that both kinds of channels cannot be eavesdropped by any third party.
- **Cryptographic Tools** Encryption tools, including symmetric encryption, asymmetric encryption and normal signature scheme, are secure. In addition, the adopted signature scheme is message recovery.
- **Honest TTP** The TTP should send a valid and honest reply to every request, which means that when the TTP is involved, if a resolve decision is made, Alice gets the payment and Bob gets the goods; if a abort decision is made, Alice and Bob get the abort confirmation and they cannot resolve the exchange in any future time.

3. A GENERIC FAIR PAYMENT PROTOCOL

In this section, we present a generic fair payment protocol which is used to implement the fair abuse-free payment protocol. This generic protocol includes 4 parts: the main protocol, the resolve sub-protocol, the abort sub-protocol and the register sub-protocol. The register protocol is new as to the origin AK protocol with offline TTP. It is presented because both parties must negotiate with TTP on some common parameters like shared secret keys. The registration protocol between the Alice/Bob and TTP needs to be run only once. And the resulting common parameters can be used for any number of transactions.

Notation To describe the protocol, we need to use several notations:

- $E_k()$: a symmetric-key encryption function under key k
- $D_k()$: a symmetric-key decryption function under key k
- $E_X()$: a public-key encryption function under pk_X
- $D_X()$: a public-key decryption function under sk_X
- $S_X()$: ordinary signature function of X
- k: the key used to cipher goods
- pk_X: public key of X
- sk_X: secret key of X
- $cipher = E_k(\mathbf{goods})$: the cipher of goods under k
- l: a label that uniquely identifies a protocol run

- *f:* a flag indicating the purpose of a message
- *h:* a secure one way hash fuction

Our protocol uses the adapted MP signature as a basic building block. So we first briefly describe this signature scheme. Then the four parts of the protocol is presented.

3.1 Adapted Mao-Paterson Convertible Signature Scheme

Let n be the Alice's RSA modulus, n is a composite integer relatively prime to $\phi(n)$. Alice chooses three integers denoted by c, d and e satisfying:

$$cde \equiv 1 (\mathrm{mod}\, \phi(n)n)$$

and

$$de \neq 1 (\mathrm{mod}(\phi(n)))$$

Her public key is the pair (e,n) and private key is d. c is the secret key shared between Alice and TTP, and will be used to convert the partial signature to a final one. c,d,e also satisfy:

$$\forall m < n^2 : m^{cde} \equiv m (\mathrm{mod}\, n^2)$$

and

$$\forall m < n : m^{cde} \equiv m (\mathrm{mod}\, n)$$

The signature scheme contains one register procedure and several signing/verifying algorithms.

- **Register Procedure** Signer (say Alice) requests for key registration by sending her public key pair (e, n) and c to the TTP (for security, c is encrypted by the TTP's public key, $E_{TTP}(c)$). TTP checks the validity of n (using the function denoted by *checkn*()), if passes, he sends a random number $\omega < n$ as the reference message. ω satisfies $gcd(\omega, n) = 1$ and $gcd(\omega \pm 1, n) = 1$. Alice then computes $PS(\omega) = \omega^d$ and send it to the TTP. After TTP checks (using the function denoted by $check\,\omega()$) whether

  $$\omega \equiv PS(\omega)^{ce} (\mathrm{mod}\, n^2)$$

  If it holds, the TTP will send a certificate $cert_A = S_{TTP}(A, E_{TTP}(c), n, \omega, PS(\omega))$ to Alice.

- **Signing/Verifying Algorithms of Full Signature** They are just normal signing/verifying algorithms of RSA signature: in the MP signature scheme, the complete secret key is dc. So the signing algorithm is $FS(m) = m^{dc}$, and the verifying algorithm *Ver(FS(m), m)* is to check whether $FS(m)^e = m (\mathrm{mod}\, n)$ (outputting **true** means yes).

- **Signing/Verifying Algorithms of Partial Signature** The signing algorithm is $PS(m) = m^d$. The verifying algorithm *PVer(PS(m), m)* needs to check whether *PS(m)* and *m* have a common exponent *d* with respect to $PS(\omega)$ and ω (outputting **true** means being yes). And that is what zero-knowledge proof can do.
- **Converting Algorithm** The TTP run this algorithm *Convert(PS(m), c)* to convert *PS(m)* to *FS(m)*: $FS(m) = PS(m)^c \pmod n$. If the result *FS(m)* is a valid RSA signature on *m,* it implies that *PS(m)* is a valid partial signature. So the TTP needs not running the *PVer(PS(m), m)* to check validity of *PS(m).*

3.2 The Protocol

3.2.1 Registration Sub-protocol

To participate in a fair payment protocol, both Alice and Bob need to run the register procedure with the TTP as required by MP signature. Note that it will not affect the security if they share a same reference message ω.

3.2.2 Main Protocol

After Alice and Bob settle the price and the goods, they can follow the main protocol:

- *Step 1,* Alice sends encrypted *goods (cipher)* with the key *k* encrypted by the TTP's public key $(E_{TTP}(k))$, her partial signature on them *(a=(cipher,* $E_{TTP}(k)$, $PS_A(a))$ to initiate the payment process.
- *Step 2,* if Bob decides to give up or he doesn't receive Alice's message in time, he can simply quit and retain fairness. When he receives the message, he will first run $PVer(PS_A(a), a)$, if it equals **true,** he will send his *check* and his partial signature on it $(PS_B(check))$ to Alice. Otherwise, he quits the protocol.
- *Step 3,* if Alice decides to give up or she doesn't receive Bob's message in time, she can invoke the *abort* sub-protocol to prevent a later resolution by the TTP. When she receive the message, she will first run $PVer(PS_B(check), check)$, if it equals **true,** she will send *k* and her full signature on *a* $(FS_A(a)$ as the **NRO**) to Bob. Otherwise, she also invokes the *abort* sub-protocol.
- *Step 4,* if Bob doesn't receive the message in time, he can invoke the *resolve* sub-protocol. When he receive the message, he will check whether *k* can decrypt the *cipher* and the *goods* is satisfactory, also he will

run $Ver(FS_A(a), a)$, if all these checking pass, he will send his *check* and his full signature on it $(FS_A(check))$ to Alice. Otherwise, he will invokes the *resolve* sub-protocol.

- *Step 5,* if Alice doesn't receive the message in time, she can invoke the *resolve* sub-protocol. When she receives the message, she will run $Ver(FS_B(check), check)$, if it equals **true,** she will accept the check. Otherwise, she will invoke the *resolve* sub-protocol.

3.2.3 Resolve Sub-protocol

Whenever necessary, Alice/Bob (noted by *X*) will invoke the *resolve* protocol to let the TTP decide whether finish or abort the payment process.

- *Step 1, X* sends to the TTP $E_{TTP}(k)$, $PS_A(a)$, *check*, $PS_B(check)$ to initiate a resolve process. Because of the resilient channel between *X* and the TTP, this message will eventually arrives the TTP.
- *Step 2,* when the TTP receive the message, it will first check whether the protocol has already been resolved or aborted, if so, it will stop because it is sure that both parties have got the resolved items or the abort confirmation. Then it will decrypt $E_{TTP}(k)$ with its secret key sk_{TTP}, if succeeds, it will run $PVer(PS_A(a), a)$ and $PVer(PS_B(check), check)$. If both equals **true,** the TTP will run $Convert(PS_A(a), c_A)$ and $Convert(PS_B(check), c_B)$, send the $FS_A(check)$ to Alice and $FS_A(a)$ & *k* to Bob. If any checking fails, it will abort the protocol and send confirmations to Alice and Bob.

3.2.4 Abort Sub-protocol

In step 2 of the main protocol, Alice can invoke this sub-protocol to make the TTP abort this payment protocol run.

- *Step 1,* Alice sends an abort request to the TTP. Because of the resilient channel between *X* and the TTP, this message will eventually arrives the TTP.
- *Step 2*, if the protocol has not been resolved or aborted, the TTP will abort the protocol and send confirmations to both parties.

3.3 Analysis of the Protocol

Following is the analysis with respect to requirement definitions in Section 2.1.

CLAIM 1 *Assuming the channel between Alice and Bob is unreliable and adopted cryptographic tools are secure, the protocol satisfies the effectiveness requirement.*

PROOF: When both Alice and Bob are honest, thus they will follow the protocol to send messages. If the probability of successful transmission in the unreliable channel is δ, then the probability of successful execution of one main protocol run will roughly be δ^4. Even it's small, but it means that successful execution without TTP's involvement is still possible. Thus the protocol satisfies the effectiveness requirement.

CLAIM 2 *Assuming the channels between the TTP and exchangers (Alice and Bob) are resilient, adopted cryptographic tools are secure and the TTP is honest, the protocol satisfies the fairness requirement.*

PROOF: The first part of fairness requirement implies two aspects: fairness for Alice and fairness for Bob.

- *Fairness for Alice* Assuming Alice is honest, then risks she may face include:

  1) She did not receive any message or the message is invalid in step 3. She can request abort to prevent that Bob may call a recovery later. If Bob's recovery request arrives to the TTP before her abort request, the TTP still will send the recovered item and evidence to her. Thus will not affect her benefit.

  2) She did not receive any message or the message is invalid in step 5. She can submit a recovery request, because the TTP is honest, the exchange will be forced to complete. If Bob sent a recovery request during this period, the result will be the same; if Bob sent an abort request which arrived before Alice's recovery request, the exchange will be aborted by the TTP, and no party can gain advantage.
- *Fairness for Bob* Assuming Bob is honest, then risks he may face include:

  1) He did not receive any message or the message is invalid in step 2. He can simply stop without any risk. And at this time, Alice cannot call recovery.

  2) He did not receive any message or the message is invalid in step 4. He can request recovery and the exchange will be forced to complete. If Alice request recovery at the same time, the result will be the same.

CLAIM 3 *Assuming the channels between the TTP and exchangers (Alice and Bob) are resilient, adopted cryptographic tools are secure and the TTP is honest, the protocol satisfies timeliness requirement.*

PROOF: Alice can conclude the protocol in one of the two ways:

- requesting abort before sending the message of step 3.
- requesting recovery in any other time.

Bob can conclude the protocol in one of the three ways:

- stopping at any time before sending the message of step 2.
- requesting recovery in any other time.

With the channel assumption, the abort confirmation or the recovered information will arrive to both parties in a finite amount of time. And all these conclusions, as discussed in the proof of claim 2, will not hurt either party's interests. So the timeliness is guaranteed.

CLAIM 4 *Assuming the channels between the TTP and exchangers (Alice and Bob) are resilient, adopted cryptographic tools (including the adopted zero-knowledge proof method) are secure, the TTP is honest, the protocol satisfies non-repudiation requirement.*

PROOF: When the exchange succeeds, either by following the main protocol or resolved by the TTP, Alice will get $FS_B(check)$, and Bob will get $FS_A(a)$ & k. If a payment protocol succeeds, by showing $FS_B(check)$, Alice can convince outside parties that Bob has received *goods* and claim her money from Bob's bank. Similarly, Bob can prove that Alice has sent *goods.* In this way, the non-repudiation requirement is satisfied.

CLAIM 5 *Assuming the channels between the TTP and exchangers (Alice and Bob) are resilient, adopted cryptographic tools (including the adopted zero-knowledge proof method) are secure, the TTP is honest, the protocol guarantees invisibility of the TTP.*

PROOF: Either the TTP is involved or not, the resulting signatures $(FS_B(check), FS_A(a))$ are just the same, so the TTP is invisible.

4. IMPLEMENTING A FAIR PAYMENT PROTOCOL TO PROVIDE ABUSE-FREENESS

As discussed in Section 3, applying a secure non-interactive zero-knowledge proof method to the MP signature scheme can achieve an efficient fair payment protocol. But this kind of protocol may result in undesirable circumstances: because the partial signature's proof is universally verifiable, a not-so-honest Alice can present Bob's partial signature to an outside company proving that Bob has purchased something, and in this way to affect the company's purchasing decision.

Abuse-freeness, defined by Garay et al. [11], means that before the protocol ends, no party can prove to an outside party that he can choose whether to complete or to abort the transaction. Recently, Chadha et al. [8] propose a more precise definition of this property. They say that one party cannot prove to an outside patty that the other party has participated in the protocol (for more discussion, see [8] Section 5)

To achieve the feature of abuse-freeness, we need a non-interactive designated verifier proof to replace the normal zero-knowledge proof. The non-interactive designated verifier proof presented by Jakobsson et al. and

strengthened by [27][24] just satisfies all those requirements and they can be easily adapted to fit in. As described in Section 1, this kind of proof can convince nobody except the signature's designated verifier. In this way, the partial signature can only be verified by two parties: the signature recipient (the designated verifier, who can be convinced by the proof) and the TTP (who can convert the partial signature to check whether the result is a valid full signature).

4.1 Non-interactive Designated Verifier Proof

The original designated verifier proof by Jakobsson et al. works for an ElGamal-like public-key encryption scheme. So we replace the public generator g with ω in our protocol. And we assume that Alice knows $PS_B(\omega)$ and Bob knows $PS_A(\omega)$. This proof can convince only the designated verifier because proof sender (say Alice) generates the proof using $PS_B(\omega)$ and Bob can simulate a second proof that can pass the same verification process.

Generating Proofs Alice selects randomly $\alpha, \beta, u \in Z_q$, where q is large enough and it is publicly accessible) and calculates

$$\begin{cases} s = \omega^{\alpha} PS_B(\omega)^{\beta} \bmod n^2 \\ \Omega = \omega^{u} \bmod n^2 \\ M = m^{u} \bmod n^2 \\ v = h(s, \Omega, M, PS_A(m)) \\ r = u + d_A(v+\alpha) \bmod q \end{cases}$$

The proof of the $PS_A(m)$, denoted by $pf(PS_A(m))$, is $(\alpha, \beta, \Omega, M, r)$.

Verifying Proofs When Bob gets the $PS_A(m)$ and $pf(PS_A(m))$, he will calculate

$$\begin{cases} s = \omega^{\alpha} PS_B(\omega)^{\beta} \bmod n^2 \\ v = h(s, \Omega, M, PS_A(m)) \end{cases}$$

and verifies

$$\begin{cases} \Omega PS_A(\omega)^{v+\alpha} = \omega^r \bmod n^2 \\ MPS_A(m)^{v+\alpha} = m^r \bmod n^2 \end{cases}$$

The verifying operations $PVer(PS_A(m),m)$ is instantiated as $verify\_des(pf(PS_A(m)),m,PS_A(m),\omega,PS_A(\omega))$. If the verification fails, the function returns **false.**

Simulating transcripts Bob can simulate correct transcripts by selecting $t,\gamma,\eta \in Z_q$ and calculate

$$\begin{cases} s = \omega^{\gamma} \bmod n^2 \\ \Omega = \omega^t PS_A(\omega)^{-\eta} \bmod n^2 \\ M = m^t PS_A(m)^{-\eta} \bmod n^2 \\ v = h(s,\Omega,M,PS_A(m)) \\ \mu = \eta - v \bmod q \\ r = (\gamma - \mu) d_B^{-1} \bmod q \end{cases}$$

4.2 Implementation of the Protocol

When implementing the protocol, we follow the principles proposed by Gurgens et al. [12] as briefly described here:

- Label Design Principles
  - **Verifiability** The creation of a label should be verifiable by anybody;
  - **Uniqueness** The label should be able to uniquely identify a protocol run;
  - **Secrecy** The values that are used to compute the label must not reveal any useful information about the exchange items (i.e. the goods).
- Message Construction Principles
  - **Authenticity** All message parts should be included in the respective signature (in plaintext or as hash);
  - **Verifiability** Every recipient should be able to verify this message;
  - **Context of Message** It should be possible for the recipient of a message to identify the protocol run to which its parts belong.

The protocol is described in form of program modules (similar with Vogt et al. [25]) and the notation <*event*>:<*description*> to describe the steps of every module. The <*event*> can be sending a message from *X* to *Y* (denoted

by $X \rightarrow Y$) or some local operations of a participant (denoted by his/her name, i.e. *A, B,* or *TTP*). The <*description*> is a brief explanation of contents of the message being sent or operations performed locally.

4.2.1 Register Module

This module is a direct instantiation of the register procedure described in Section 3.1.

Registration Module
$X \rightarrow TTP$: f_{Reg}, pk_X, $E_{TTP}(c)$
TTP: **if not** *checkn*(n) **then** stop
$TTP \rightarrow X$: f_{Ref}, X, ω
$X \rightarrow TTP$: f_{Ref}, $PS_X(\omega)$
TTP: **if not** *check*ω() **then** stop
$TTP \rightarrow X$: f_{cert}, X, $cert_X$

4.2.2 Main Module

The label for a protocol run is computed by Alice: $l=h(A, B, TTP, h(cipher), h(k))$. And $E_{TTP}(k)$ is replaced by $E_{TTP}(l, k)$ to ensure this encrypted key can not be decrypted in a different protocol run. In our protocol, we denote the content to be signed by Alice for **NRO** as $a=(f_{NRO}, B, l, h(k), cipher, E_{TTP}(k))$. Bob's check is signed with A and l ($b=(A, l, check)$), so the signed check will be $PS_B(b)$ and $FS_B(b)$. With this construction, the signed check can only be used by Alice to claim the money.

Main Module
$A \rightarrow B$: f_{EOO}, B, l, $h(k)$, $cipher$, $E_{TTP}(l, k)$, $PS_A(a)$, $pf(PS_A(a))$
B: **if not** $verify\_des(pf(PS_A(a)), a, PS_A(a), \omega, PS_A(\omega))$ **then** stop
$B \rightarrow A \square f_{EOR}$, A, l, $PS_B(b)$, $pf(PS_A(b))$
A: **if** times out **then abort**
elseif not $verify\_des(pf(PS_B(b)), b, PS_B(b), \omega, PS_B(\omega))$ **then abort**
$A \rightarrow B \square f_{NRO}$, B, l, k, $FS_A(a)$
B: **if** times out **then** call **resolve**[X:=B,Y:=A]
$B \rightarrow A \square f_{NRR}$, A, l, $FS_B(b)$
A: **if** *A* times out **then** call **resolve**[X:=A,Y:=B]

4.2.3 Resolve Module

The resolve protocol is similar with the one in AK protocol. It is executed when an error happens, one party needs TTP's help to decrypt the key k and generate the final evidences for him/her. Assume the TTP keeps a record on whether the protocol has been resolved or aborted (denoted by two variables: *aborted* and *resolved*)

Resolve Module

$X \rightarrow TTP$: $f_{RecX}, f_{Sub}, Y, l, h(cipher), h(k), E_{TTP}(k), \mathbf{Rec_X}, PS_A(a), PS_B(b)$
  TTP: **if** $h(k) \neq h(D_{TTP}(E_{TTP}(k)))$ **or** *aborted* **or** *resolved* **then** stop
    else *resolved*=true
    $Convert(PS_A(a), c_A)$ and $Convert(PS_B(check), c_B)$
$TTP \rightarrow A$: $f_{NRR}, A, l, FS_A(a)$
$TTP \rightarrow B$: $f_{NRO}, B, l, k, FS_B(b)$

4.2.4 Abort Module

Alice submits an abort request using abort module, i.e. set the aborted=**true**, preventing Bob may recover in a future time which she will not wait. $\mathbf{Con}_a$ denotes the abort confirmation.

Abort Module

$X \rightarrow TTP$: f_{Abort}, l, B, **abort**
  TTP: **if** *aborted* **or** *recovered* **then** stop
    else *recovered*=true
$TTP \rightarrow A$: $f_{Cona}, A, B, l, \mathbf{Con}_a$
$TTP \rightarrow B$: $f_{Cona}, A, B, l, \mathbf{Con}_a$

5. DISCUSSIONS

In this section, we give some remarks on the cryptographic tools and implementation, resulting an outline of possible extensions.

5.1 The Dodis-Reyzin CS signature

This CS signature is quite efficient for it additively split the secret key and preserve security. But this signature is based on the GDH signature (see [10] section 4), whose verification needs zero-knowledge proofs and the partial signatures are to be verified both by the recipient (in main protocol) and

the TTP (in resolve sub-protocol). When realizing abuse-freeness, if directly applied with the designated verifier proofs, the TTP will NOT be convinced of the validity of the partial signature. As a result, a different proof method should be adopted to make both the recipient and the TTP convinced. Recently, Wang [26] has proposed a new proof method call Restrictive Confirmation Signature Scheme (RCSS) that fulfills this requirement. But the outcome is quite complicated and adds computation burdens. So we choose the RSA-based CS signature scheme, which splits the secret key multiplicatively similarly with the one in Boyd and Foo protocol. But our protocols are more efficient in that they use non-interactive partial signature verification and more importantly, our protocol assures timeliness.

5.2 The Saeednia-Kremer-Markowitch Designated Verifier Scheme

Recently, Saeednia et al. [24] has proposed a stronger and more efficient designated verifier scheme, which can be easily adapted to our abuse-free protocol if stronger requirements are assumed.

5.3 Protecting the Payer's Privacy

Anonymity is an important issue in payment system because normally customers are not willingly to have his purchase exposed (especially in a open network, that is why we assume two companies in our scenario for they can afford more secure channels and honest TTP). We propose two variations considering our protocol: 1) Not including any goods information in Bob's check. It will weaken the property of non-repudiation, as Alice won't get a signed list of purchased goods by Bob. 2) Applying anonymous but traceable e-cash into the protocol (see Wang [26]).

5.4 Extending Our Results to Contract Signing

Abuse-freeness is first introduced in context of contract signing, and our protocol can be easily transformed into fair exchange of two signatures. So constructing a new abuse-free contract signing protocol is a useful extension of our protocol.

6. IMPLEMENTING THE PROTOCOL USING AGENT MECHANISMS

The agent mechanism has been widely used in electronic commerce applications. Pagnia et al. [21] have presented implementation of fair protocol using mobile agents.

7. CONCLUSIONS

In this paper, we produce an efficient generic fair payment protocol with RSA-based convertible signature. Based on that, we implement a fair abuse-free payment protocol using non-interactive designated verifier proof as a new proposal on the issue of abuse-freeness.

We have shown that the protocol are practical because their standardized evidences and high efficiency. Our future work will be focused on the application of the fair payment protocols to real electronic commerce systems like SCM and CRM.

ACKNOWLEDGEMENTS

The authors would like to thank Steve Kremer for helpful discussions on the issues of abuse-freeness and designated verifier proof. Furthermore, we would like to thank the anonymous reviewers of I3E for valuable comments.

REFERENCES

[1] N. Asokan, Philippe A. Janson, Michael Steiner, and Michael Waidner, "The State of the Art in Electronic Payment Systems", IEEE Computer, September 1997, pp. 28-35.

[2] N. Asokan, M. Schunter, and M. Waidner, "Optimistic protocols for fair exchange", in Proceedings of the fourh ACM Conference on Computer and Communications Security, Zurich, Switzerland, April 1997, pp. 6, 8-17.

[3] N. Asokan and V. Shoup, "Optimistic fair exchange of digital signatures", in Advances in Cryptology – EUROCRYPT '98, volume 1403 of Lecture Notes in Computer Science, Springer-Verlag, 1998, pp. 591-606.

[4] N. Asokan, V. Shoup, and M. Waidner, "Asynchronous protocols for optimistic fair exchange", in Proceedings of IEEE Symposium on Research in Security and Privacy, 1998, pp. 86-99. (Printed version contains some errors. Errata sheet is distributed together with the electronic version).

[5] G. Ateniese, "Efficient verifiable encryption (and fair exchange) of digital signatures", in Proceedings of the 6th ACM conference on Computer and communications security, Nov. 1999.

[6] F. Bao, R. Deng and W. Mao, "Efficient and practical fair exchange protocols with off-line TTP", in Proceedings of IEEE Symposium on Security and Privacy, Oakland, May 1998, pp. 77-85.

[7] C. Boyd, E. Foo. "Off-line Fair Payment Protocols using Convertible Signatures", in Advances in Cryptology---ASIA CRYPT'98,1998.

[8] R. Chadha, J. Mitchell, A. Scedrov and V. Shmatikov, "Contract signing, optimism and advantage", in CONCUR 2003 - Concurrency Theory, 14-th International Conference, Marseille, France, September 2003. Volume 2761 of Lecture Notes in Computer Science, Springer-Verlag, 2003, pp. 366-382

[9] R. Chadha, M. Kanovich and A. Scedrov, "Inductive methods and contract-signing protocols", in Proceedings of 8th ACM confererence on Computer and Communications Security(CCS-8). Philadelphia, Pennsylvania, ACM Press, November 7, 2001, pp. 176-185.

[10] Y. Dodis and L. Reyzin, "Breaking and repairing optimistic fair exchange from PODC 2003", in Proceedings of the 2003 ACM workshop on Digital rights management, Oct. 2003.

[11] J. Garay, M. Jakobsson and P. MacKenzie. "Abuse-free optimistic contract signing", in Advances in Cryptology - CRYPTO '99, volume 1666 of Lecture Notes in Computer Science, SpringerVerlag, 1999, pp. 449-466.

[12] S. Gurgens, C. Rudolph and H. Vogt, "On the Security of Fair Non-repudiation Protocols", in Proceedings of 2003 Information Security Conference, volume 2851 of Lecture Notes in Computer Science, Bristol, UK, Oct. 2003, pp. 193--207.

[13] M. Jakobsson, K. Sako, R. Impagliazzo, "Designated verifier proofs and their applications", in Eurocrypt'96, volume 1070 of Lecture Notes in Computer Science, Springer Verlag, 1996, pp. 143-154.

[14] S. Kremer and O. Markowitch, "Optimistic non-repudiable information exchange", in 21th Symposium on Information Theory in the Benelux, Werkgemeenschap Informatie- en Communicatietheorie, Enschede, may 2000, pp. 139-146.

[15] S. Kremer and J.F. Raskin, "Game Analysis of Abuse-free Contract Signing", in 15th Computer Security Foundations Workshop (CSFW 2002), IEEE Computer Society. Cape Breton, Nova Scotia, Canada, 2002.

[16] P. Louridas, "Some guidelines for non-repudiation protocols", ACM SIGCOMM Computer Communication Review, Oct. 2000.

[17] O. Markowitch and S. Kremer, "An optimistic non-repudiation protocol with transparent trusted third party", in Information Security: ISC 2001, volume 2200 of Lecture Notes in Computer Science, Malaga, Spain. Springer-Verlag, Oct. 2001, pp. 363-378.

[18] O. Markowitch, S. Kremer and D. Gollmann, "On Fairness in Exchange Protocols", in Information Security and Cryptology - ICISC 2002, volume 2587 of Lecture Notes in Computer Science. Springer-Verlag, 2002.

[19] W. Mao and K. Paterson, "Convertible Undeniable Standard RSA Signatures", 2000; http://citeseer.ist.psu.edu/mao00convertible.html.

[20] S. Micali, "Certified e-mail with invisible post offices", Available from author: an invited presentation at the RSA'97 conference, 1997.

[21] H. Pagnia, H. Vogt, F.C. Gatner, and U.G. Wilhelm, "Solving Fair Exchange with Mobile Agents", Volume 1882 of Lecture Notes in Computer science, Springer, September 2000, pp. 57-72.

[22] J.M. Park, Edwin K. P. Chong and H. J. Siegel, "Constructing fair-exchange protocols for E-commerce via distributed computation of RSA signatures", in Proceedings of the twenty-second annual symposium on Principles of distributed computing, July 2003.

[23] B. Pfitzmann, M. Schunter and M. Waidner, "Optimal efficiency of optimistic contract signing", in Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing, June 1998.

[24] S. Saeednia, S. Kremer and O. Markowitch, "An efficient strong designated verifier scheme", in 6th International Conference on Information Security and Cryptology (ICISC 2003), Lecture Notes in Computer Sciences, Springer-Verlag. Seoul, Korea, Nov. 2003.

[25] H. Vogt, H. Pagnia, and F.C. Gartner, "Modular Fair Exchange Protocols for Electronic Commerce", In Proceedings of the 15th Annual Computer Security Applications Conference, Phoenix, Arizona. IEEE, Dec. 1999.

[26] C.H. Wang, "Untraceable Fair Network Payment Protocols with Off-Line TTP", in Advances in Cryptology - ASIACRYPT 2003: 9th International Conference on the Theory and Application of Cryptology and Information Security, volume 2894 of Lecture Notes in Computer Science, Spring-Verlag, 2003, pp. 173 - 187

[27] G. Wang, "An Attack on Not-interactive Designated Verifier Proofs for Undeniable Signatures", Cryptology ePrint Archive, Report 2003/243, 2003; http://eprint.iacr.org/2003/243/.