An Abuse-Free Fair Contract Signing Protocol Based on the RSA Signature

Guilin Wang
Infocomm Security Department
Institute for Infocomm Research (I^2R)
21 Heng Mui Keng Terrace, Singapore 119613
glwang@i2r.a-star.edu.sg

ABSTRACT

A fair contract signing protocol allows two potentially mistrusted parities to exchange their commitments (i.e., digital signatures) to an agreed contract over the Internet in a *fair* way, so that either each of them obtains the other's signature, or neither party does. Based on the RSA signature scheme, a new digital contract signing protocol is proposed in this paper. Like the existing RSA-based solutions for the same problem, our protocol is not only fair, but also *optimistic*, since the third trusted party is involved only in the situations where one party is cheating or the communication channel is interrupted. Furthermore, the proposed protocol satisfies a new property, i.e., it is *abuse-free*. That is, if the protocol is executed unsuccessfully, none of the two parties can show the validity of intermediate results to others. Technical details are provided to analyze the security and performance of the proposed protocol. In summary, we present the first abuse-free fair contract signing protocol based on the RSA signature, and show that it is both secure and efficient.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; K.4.4 [**Computer and Society**]: Electronic Commerce—*Security, Distributed Commercial Transactions*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*

General Terms

Algorithms, Design, Legal Aspects, Security, Theory.

Keywords

Contract signing, fair-exchange, digital signatures, RSA, e-commerce, cryptographic protocols, security

1. INTRODUCTION

1.1 Background

Contract signing plays a very important role in any business transaction, in particular in situations where the involved parties do not trust each other to some extent already. In the paper-based scenario, contract signing is truly simple due to the existence of "simultaneity". That is, both parties generally sign two hard copies of the same contract at the same place and at the same time. After that, each party keeps one copy as a legal document that shows both of them have committed to the contract. If one party does not abide by the contract, the other party could provide the signed contract to a judge in court.

As the electronic commerce is becoming more and more important and popular in the world, it is desirable to need a mechanism that allows two parties to sign a digital contract via the Internet. However, the problem of contract signing becomes difficult in this setting, since there is no simultaneity any more in the scenario of computer networks. In other words, the simultaneity has to be mimicked in order to design a digital contract signing protocol. This requirement is essentially captured by the concept of *fairness*: At the end of the protocol, *either* both parties have valid signatures for a contract *or* neither does, even if one of them tries to cheat or the communication channel is out of order. In fact, Even and Yacobi [20] proved that fairness is impossible to be achieved in a deterministic two-party contract signing protocol. The intuitive reason could be explained as follows. The purpose is to go from the initial fair state, in which no party has what he/she expects, to the desired fair state in which both obtain what they want. However, information is exchanged in computer networks non-simultaneously, so an unfair state must be passed through.

1.2 Related Work

From the view point of technique, the problem of digital contract signing belongs to a wide topic: fair exchange, i.e., how to enable two (or multiple) potentially mistrusted parities exchanging digital items over computer networks in a fair way, so that each party gets the other's item, or neither party does. Actually, fair exchange includes the following different but related issues: contract signing protocols [20, 12, 16, 6, 2, 4, 23, 33, 7], certified e-mail systems [39, 30, 5, 28, 1], non-repudiation protocols [38, 31, 27], and e-payment schemes in electronic commerce [15, 34]. For more references and discussions on the relationships between those conceptions, please refer to [3, 31]. In this paper, we mainly focus on the problem of digital contract signing. Since a party's commitment to a digital contract is usually defined as his/her digital signature on the contract, digital contract signing is essentially implied by fair exchange of digital signatures between two potentially mistrusted parities.

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.
WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-046-9/05/0005.

There is a rich history of contract signing (i.e., fair exchange of digital signatures) because this is a fundamental problem in electronic transactions. According to the involvement degree of a trusted third party (TTP), contract signing protocols can be divided into three types: (1) gradual exchanges without any TTP; (2) protocols with an on-line TTP; and (3) protocols with an off-line TTP. Early efforts [25, 19, 16] mainly focused on the first type protocols to meet *computational fairness*: Both parties exchange their commitments/secrets "bit-by-bit". If one party stops prematurely, both parties have about the same fraction of the peer's secret, which means that they can complete the contract off-line by investing about the same amount of computing work. The major advantage of this approach is that no TTP is involved. However, this approach is unrealistic for most real-world applications due to the following several reasons. First of all, it is assumed that the two parties have equivalent computation resources. Otherwise, such a protocol is favorable to the party with stronger computing power, who may conditionally force the other party to commit the contract by its own interest. At the same time, such protocols are inefficient because the costs of computation and communication are extensive. In addition, as pointed out in [12], this approach has the unsatisfactory property of uncertain termination. For example, suppose two parties are signing a house-sale contract. If the protocol stops prematurely on the side of the buyer, the seller will never be sure whether the buyer is continuing with the protocol, or has terminated - and perhaps even has engaged in another house-sale contract signing protocol with another seller. The buyer may be in a similar situation if the protocol terminated on the side of the seller.

In the second type of fair exchange protocols [12, 17, 38], an on-line TTP is always involved in every exchange. In this scenario, a TTP is essentially a mediator: (a) Each party first sends his/her item to the TTP; (b) Then, the TTP checks the validity of those items; (c) If all expected items are correctly received, the TTP finally forwards each item to the party who needs it. Generally speaking, contract signing protocols with an on-line TTP could be designed more easily since the TTP facilitates each step of exchanging, but may be still expensive and inefficient because the TTP needs to be paid and must be part of every execution. In practice, the TTP is prone to become a bottleneck in the whole system, especially in the situation where many users rely on a single TTP.

Compared with the schemes belonging to previous two types, contract signing protocols with off-line TTP [2, 3, 4, 6, 34] are more appealing and practical for most applications. Because those protocols are *optimistic* in the sense that the TTP is not invoked in the execution of exchange unless one of the two parties misbehaves or the communication channel is out of order. Bao et al. [6] and Ateniese [4] constructed fair exchange protocols of digital signatures from *verifiably encrypted signatures*, while Asokan et al. [2, 3] proposed such protocols by using *verifiable escrows*. The basic ideas behind those two cryptographic primitives are similar, as explained below. To get the digital signature from the other party Bob, a party Alice first encrypts her signature under the TTP's public encryption key, and proves to Bob that the ciphertext indeed corresponds to her signature, interactively or non-interactively. Then, Bob sends his digital signature (or some digital item) to Alice. After receiving the expected item from Bob, Alice reveals her signature to Bob. The point is that if Alice refuses to do so after getting Bob's item, the TTP can decrypt Alice's encrypted signature and sends the result to Bob. The difference between those two kinds of schemes is that in the verifiable escrow based schemes, Alice, the creator of the encryption, has the ability to control the conditions under which the encryption could be decrypted by the TTP. Though their techniques can be applied to a variety of signature schemes, the overheads of computation and communication are usually expensive. In particular, the schemes in [2, 3, 6] are inefficient, since expensive cut-and-choose techniques [21] are used to prove the correctness of the encrypted signature. In addition, it is noticed in [8] that the Schnorr and ElGamal signatures based fair-exchange schemes in [4] should be improved to avoid a security flaw.

In [33], Micali constructed several simple fair exchange schemes based on any secure signature and encryption algorithms. However, Bao et al. [7] pointed his contract signing protocol is actually unfair because there is an intrinsic flaw in the dispute resolution protocol, which is the policy exploited by the TTP to settle potential disputes between the two parties involved in a contract signing.

Based on an RSA multisignature scheme, Park et al. [34] proposed a novel fair exchange protocol with an off-line trusted party in PODC 2003. Their protocol was fair and optimistic but *insecure*, since Dodis and Reyzin [18] broke their protocol by pointing out that an honest-but-curious TTP can easily derive a user's private key after the end of his/her registration. Moreover, as an improvement of Park et al.'s scheme, Dodis et al. even constructed a provably secure fair exchange protocol from the non-interactive two-signature of Boldyreva [13]. Their scheme works in gap Diffie-Hellman (GDH) groups[1]. The pairing based cryptosystems [14, 13] are typical examples constructed from GDH groups. However, note that in such cryptosystems, the computation of the pairing is still time-consuming, although several papers have investigated into speeding up the pairing computation [9, 22].

Furthermore, we remark that in the essence Dodis et al.'s scheme is *not* an improvement of Park et al.'s scheme, since the security of their scheme is based on the GDH problem instead of the RSA probem or factoring problem [36]. Note that the RSA cryptosystem [36] is now the de facto industrial standard and is widely used in many applications, it is highly desirable to construct fair exchange protocols based on RSA. Actually, as we mentioned before, several such schemes have been proposed: Asokan et al.'s scheme [2, 3]

[1]A group G is called a gap Deffie-Hellman group, if it is infeasible to solve the computational Diffie-Hellman (CDH) problem in G, but the decisional Deffie-Hellman (DDH) problem in G can be solved feasibly. We give more explanations on those problems. Let $G = \langle g \rangle$ denote a multiplicative cyclic group. Then, the CDH problem is to output the value of g^{ab} when $g, g^a, g^b \in G_q$ are given, where a and b are unknown random numbers. In the DDH problem, it is required to determine whether g^{ab} equals g^c when $g, g^a, g^b, g^c \in G_q$ are given, where a, b and c are unknown random numbers. Actually, another related problem is the discrete logarithm (DL) problem. That is, given $g, g^a \in G$ where a is a random number, how to solve a. In fact, it is easy to know that the DL problem is at least as difficult as the CDH problem, and the CDH problem is at least as difficult as the DDH problem.

from verifiable escrow, Ateniese's scheme [4] from verifiably encrypted signature, and Park et al.'s scheme [34] from multisignature. However, all those schemes are *not abuse-free* [23]. That is, a party can get verifiable intermediate results when the signature exchange protocol is executed unsuccessfully. Consequently, this party may obtain some benefits by showing such universally verifiable intermediate results to a third party. For example, if Bob is looking for a job and he has received two offers from competing companies A and C. Bob prefers to join company C though the offered salary is not so much satisfactory. In contrast, company A promises a higher salary but he does not really like to join it due to some personal reason, such as weather, culture or something else. In this scenario, Bob may first pretend to sign an employment contract with company A. Then, he terminates the execution of the contract signing protocol after he obtained the intermediate results generated by company A. By showing such universally verifiable proofs to company C, Bob may get a higher salary from company C. There exists the same problem in other similar situations.

Therefore, running contract protocols without the property of abuse-freeness is a risk for a honest party, as a possible dishonest party maybe does not really want to sign the contract with her, but only use her willingness to sign to get leverage for another contract. Consequently, this is an important security requirement for contract signing protocols, especially in the situations where partial commitments to a contract may be beneficial to a dishonest party or an outsider. However, except the discrete logarithm based scheme of Garay et al. [23], all other optimistic contract signing protocols [2, 3, 4, 6, 7, 33, 34] are not abuse-free.

1.3 Our Work

Motivated by the above example that shows the importance of abuse-freeness, and the question of how to improve Park et al.'s scheme in a secure way, this paper proposes a new contract signing protocol for two mutually distrusted parties. Our protocol is based on an RSA multisignature, which is formally proved to be secure by Bellare and Sandhu [11]. Like the schemes in [2, 4, 34], our protocol is fair and optimistic. Furthermore, different from the above existing schemes, our protocol is *abuse-free*. The reason is that we integrate an interactive protocol, proposed for confirming RSA undeniable signatures by Gennaro et al. [24], into our scheme to prove the validity of the intermediate results. Technical analysis and discussion are provided in detail to show that our scheme is secure and efficient.

More specifically, the new protocol satisfies the following desirable properties.

(1) **Fairness**: Our protocol guarantees the two parities involved to obtain or not obtain the other's signature *simultaneously*. This property implies that even a dishonest party who tries to cheat cannot get an advantage over the other party.

(2) **Optimism**: The third trusted party (TTP) is involved only in the situation where one party is cheating or the communication channel is interrupted. So it could be expected that the TTP is only involved in settling disputes between users *rarely*, due to the fact that fairness is always satisfied, i.e., cheating is not beneficial to the cheater.

(3) **Abuse-Freeness**: If the protocol is not executed successfully, any of the two parties cannot show the validity of the intermediate results generated by the other to an outsider[2]. As we mentioned before, the unique known abuse-free contract signing protocol [23] is based on the discrete logarithm problem, instead of the RSA cryptosystem.

(4) **Provable Security**: Under the standard assumption that the RSA problem is intractable [36, 11], the protocol is provably secure in the random hash function model [10], where a hash function is treated as if it were a "black box" containing a random function.

(5) **Timely Termination**: The execution of a protocol instance will be terminated in a predetermined time. This property is implemented by adding a reasonable deadline t in a contract, as suggested by Micali in [33]. If one party does not send his/her signature to the other party after the deadline t, both of them are free of liability to their partial commitments to the contract and do not need to wait any more.

(6) **Compatibility**: In our protocol, each party's commitment to a contract is a standard digital signature. This means that to use the protocol in existing systems, there is no need to modify the signature scheme or message format at all. Thus, it will be very convenient to integrate the contract signing protocol into existing softwares for electronic transactions.

(7) **TTP's Statelessness**: To settle potential disputes between users, the TTP is not required to maintain a database to searching or remembering the state information for each protocol instance. So the overhead on the side of the TTP is reduced greatly, compared with the previous schemes in [2, 3, 23].

(8) **High Performance**: In a typical implementation, the protocol execution in a normal case requires only interaction of several rounds between two parties, transmission of about one thousand bytes of data, and computation of a few modular exponentiations by each party.

The rest of the paper is organized as follows. Section 2 reviews Park et al.'s scheme and its security. In Section 3, we propose a new contact signing protocol based on the RSA signature. Then, we analyze its security and efficiency in Sections 4 and 5, respectively. Finally, Section 6 concludes the paper.

2. PARK ET AL.'S SCHEME AND ITS SECURITY

In this section, we briefly overview Park et al.'s scheme and the attack on it identified by Dodis and Reyzin. For more details, please refer to the original papers [34, 18].

In Park et al.'s scheme, Alice sets an RSA modulus $n = pq$, where p and q are two k-bit safe primes, and picks her

[2]Note that if the two parties signed a contract by successfully executing the protocol, it does not matter whether the intermediate results are publicly verifiable or can be proved to others by one party. Because, in this case, both parties' digital signatures, i.e., the their complete commitments to the contract, are already publicly verifiable.

random public key $e \in_R \mathbb{Z}^*_{\phi(n)}$, and calculates her private key $d = e^{-1} \mod \phi(n)$, where $\phi(n) = (p-1)(q-1)$. Then, she registers her public key with a certification authority (CA) to get her certificate C_A. After that, Alice randomly splits d into d_1 and d_2 so that $d = d_1 + d_2 \mod \phi(n)$, where $d_1 \in_R \mathbb{Z}^*_{\phi(n)}$. To get a voucher V_A from a TTP, Alice is required to send (C_A, e_1, d_2) to the TTP, where $e_1 = d_1^{-1} \mod \phi(n)$. The voucher V_A is the TTP's signature that implicitly shows two facts: (1) e_1 can be used to verify a partial signature generated by using secret key d_1, and (2) the TTP knows a secret d_2 that matches with RSA key pairs (d_1, e_1) and (d, e).

When Alice and Bob want to exchange their signatures on a message m, Alice first computes $\sigma_1 = h(m)^{d_1} \mod n$, and sends (C_A, V_A, σ_1) to Bob, where $h(\cdot)$ is a secure hash function. Upon receiving (C_A, V_A, σ_1), Bob checks the validity of C_A and V_A, and whether $h(m) \equiv \sigma_1^{e_1} \mod n$. If all those verifications go through, Bob returns his signature σ_B to Alice, since he is convinced that the expected $\sigma_2 = h(m)^{d_2} \mod n$ can be revealed by Bob or the TTP. After receiving valid σ_B, Alice reveals $\sigma_2 = h(m)^{d_2} \mod n$ to Bob. Finally, Bob obtains Alice's signature σ_A for message m by setting $\sigma_A = \sigma_1 \sigma_2 \mod n$, since we have

$$h(m) \equiv \sigma_A^e = h(m)^{(d_1+d_2)e} = h(m)^{de} \mod n.$$

The security problem in Park et al.'s scheme is that an honest-but-curious TTP can easily derive Alice's private key d. The reason is that with the knowledge of (n, e, e_1, d_2), the TTP knows that the integer $e - (1 - ed_2)e_1$ is a non-zero multiple of $\phi(n)$. It is well known that knowing such a multiple of $\phi(n)$, Alice's RSA modulus n can be easily factored. Consequently, the TTP can get Alice's private key d by the extended Euclidean algorithm.

The point is that we do not want the TTP having the ability of making a user's signatures independently, though the TTP is a (partially) trusted party. The main reason is that as the pivotal secret of any cryptosystem, the private key should not be revealed to any party, including a partially trusted party. In addition, if there is a completely trusted TTP, the problem of fair exchange can be solved trivially as follows. Firstly, each party gives his/her private key to the TTP before exchanging items so that the TTP can generate signatures on behalf of any party if necessary. Then, the TTP issues a voucher for each registered party to show that it knows this party's private key. When Alice and Bob want to exchange their signatures on a message m, they first exchange their vouchers issued by the TTP. By doing so correctly, it is proved that both of them have registered with the TTP. After that, their signatures can be delivered directly to the other side. If one party, say Alice, does not receive Bob's signature on m, she applies the TTP's help by providing her signature and message m. After checking the correctness of this information, the TTP will generate and send Bob's signature on m to Alice by using Bob's private key.

3. THE PROPOSED PROTOCOL

In this section, we describe our new contract signing protocol based on the RSA signature [36]. The basic idea is that Alice first splits her private key d into d_1 and d_2 so that $d = d_1 + d_2 \mod \phi(n)$, as Park et al. did in [34]. Then, *only* d_2 is delivered to the TTP, while Alice keeps (d, d_1, d_2) as secrets. To exchange her signature $\sigma_A = h(m)^d \mod n$ with Bob, Alice first sends partial signature $\sigma_1 = h(m)^{d_1} \mod n$ to Bob, and proves that σ_1 is prepared correctly in an interactive zero-knowledge way by exploiting Gennaro et al.'s protocol [24]. After that, Bob sends his signature σ_B on message m to Alice, since he is convinced that even if Alice refuses to reveal the second partial signature $\sigma_2 = h(m)^{d_2} \mod n$, the TTP can do the same thing.

As usual, we assume that the communication channel between Alice and Bob is *unreliable*, i.e., messages inserted into such a channel may be lost due to the failure of computer network or attacks from adversaries. However, the TTP is linked with Alice and Bob by *reliable* communication channels, i.e., messages inserted into such a channel will be delivered to the recipient after a finite delay.

3.1 Registration Protocol

To use our protocol for exchanging digital signatures, only the initiator Alice needs to register with the TTP. That is, Alice is required to get a voucher V_A from the TTP besides obtaining a certificate C_A from a certification authority (CA). To this end, the following procedures are executed.

(1) Alice first sets an RSA modulus $n = pq$, where p and q are two k-bit safe primes, i.e., there exist two primes p' and q' such that $p = 2p' + 1$ and $q = 2q' + 1$. Then, Alice selects her random public key $e \in_R \mathbb{Z}^*_{\phi(n)}$, and calculates her private key $d = e^{-1} \mod \phi(n)$, where $\phi(n) = (p-1)(q-1)$. Finally, Alice registers her public key with a CA to get her certificate C_A, which binds her identity and the corresponding pubic key (n, e) together.

(2) Alice randomly splits d into d_1 and d_2 such that $d = d_1 + d_2 \mod \phi(n)$ by choosing $d_1 \in_R \mathbb{Z}^*_{\phi(n)}$, and computes $e_1 = d_1^{-1} \mod \phi(n)$. At the same time, she generates a sample message-signature pair (w, σ_w), where $w \in \mathbb{Z}^*_n \setminus \{1, -1\}$, $\mathrm{ord}(w) \geq p'q'$, and $\sigma_w = w^{d_1} \mod n$. Then, Alice sends (C_A, w, σ_w, d_2) to the TTP but keeps (d, d_1, d_2, e_1) secret.

(3) The TTP first checks Alice's certificate C_A is valid. After that, the TTP checks that the triple (w, σ_w, d_2) is prepared correctly. If everything is in order, the TTP stores d_2 securely, and creates a voucher V_A by computing $V_A = Sign_{TTP}(C_A, w, \sigma_w)$. That is, V_A is the TTP's signature on message (C_A, w, σ_w), which guarantees that the TTP can issue a valid partial signature on behalf of Alice by using the secret d_2.

We give some notes on the above registration protocol. To get her certificate from a CA, Alice has to prove that modulus n is the product of two *safe* primes. This technical issue is addressed in [24]. Of course, step (1) can be omitted if Alice has obtained such a certificate before she registers with the TTP. To validate the correctness of the triple (w, σ_w, d_2), the TTP needs to do the followings. Firstly, the TTP validates that w is an element of order at least of $p'q'$ by checking that $w \in \mathbb{Z}^*_n \setminus \{1, -1\}$, and that both $\gcd(w-1, n)$ and $\gcd(w+1, n)$ are not prime factors of n ([24], Lemma 1). Then, Alice is required to show that she knows the discrete logarithm of σ_w to the base w via a zero-knowledge protocol interactively or non-interactively (see Section 4.3 of [24]). Finally, the TTP checks whether $w \equiv (\sigma_w w^{d_2})^e \mod n$. If

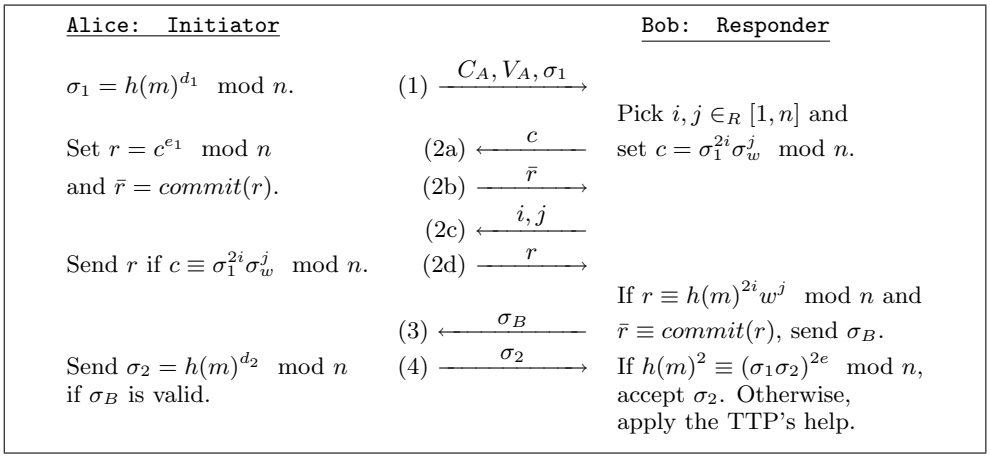

Figure 1: Signature Exchange Protocol.

all those validations pass, the TTP accepts (w, σ_w, d_2) as a valid triple and creates the voucher V_A for Alice.

Though the above registration protocol is a little complicated, we remark that this stage needs to be executed *only once* for a sufficiently long period, for example, one year. In this period, Alice can fairly sign any number of contracts with all potential parties. Furthermore, it seems reasonable in the real world to require users to first register with the TTP before they are served. The reason is that the TTP is usually unlikely to provide free service for settling disputes between users. Moreover, for enhancing efficiency, the sample message w can be fixed as a constant, e.g., $w = 2$, as pointed out by Gennaro et al. [24]. Compared with schemes based on verifiably encrypted signatures [2, 4, 6], one disadvantage of our registration protocol is that the TTP needs to keep a distinct secret d_2 for each registered user. However, this shortcoming can be eliminated by some simple techniques. For example, the TTP can encrypt each concatenation of d_2 and the corresponding user's unique identifier by exploiting a secure symmetric-key encryption algorithm, and then stores the results into its database. To extract a user's d_2 later, the TTP only needs to decrypt the corresponding record using the unique symmetric key.

3.2 Signature Exchange Protocol

We assume that a contract m has been agreed between Alice and Bob before they begin to sign it. In addition, it is supposed that the contract explicitly contains the following information: a predetermined but reasonable deadline t, the identities of Alice, Bob, and the TTP. Our signature exchange protocol is briefly illuminated in Figure 1, and further described in detail as follows.

(1) Firstly, the initiator Alice computes her partial signature $\sigma_1 = h(m)^{d_1} \mod n$, and then sends the triple (C_A, V_A, σ_1) to the responder Bob. Here, $h(\cdot)$ is a cryptographically secure hash function.

(2) Upon receiving (C_A, V_A, σ_1), Bob first verifies that C_A is Alice's certificate issued by a CA, and that V_A is Alice's voucher created by the TTP. Then, Bob checks if the identities of Alice, Bob, and the TTP are correctly specified in the contract m. If all those validations hold, Bob initiates the following interactive protocol with Alice to check whether σ_1 is Alice's valid partial signature on contact m.

(2a) Bob picks two numbers $i, j \in_R [1, n]$ at random, and sends a challenge c to Alice by computing $c = \sigma_1^{2i}\sigma_w^j \mod n$.

(2b) After getting the challenge c, Alice calculates the respondence $r = c^{e_1} \mod n$, and then returns her commitment $\bar{r} = commit(r)$ to Bob, where $commit(\cdot)$ is a secure commitment scheme (See [35], for example).

(2c) When the commitment $\bar{r}$ is received, Bob sends the pair (i, j) to Alice.

(2d) Alice checks whether the challenge c is prepared properly, i.e., $c \equiv \sigma_1^{2i}\sigma_w^j \mod n$. If the answer is positive, Alice reveals the respondence r to Bob. With the knowledge of r, Bob accepts σ_1 as *valid* if and only if $r \equiv h(m)^{2i}w^j \mod n$ and $\bar{r} \equiv commit(r)$.

(3) Only if σ_1 is Alice's valid partial signature and the deadline t specified in contract m is sufficient for applying dispute resolution from the TTP, Bob sends his signature σ_B on contract m to Alice, since he is convinced that another partial signature σ_2 can be released by the TTP, in case Alice refuses to do so.

(4) Upon receiving σ_B, Alice checks whether it is Bob's valid signature on message m. If this is correct, she sends Bob the partial signature σ_2 by computing $\sigma_2 = h(m)^{d_2} \mod n$. When Bob gets σ_2, he sets $\bar{\sigma}_A = \sigma_1\sigma_2 \mod n$, and accepts σ_2 as *valid* if and only if $h(m)^2 = \bar{\sigma}_A^{2e} \mod n$. In this case, Bob can recover Alice's standard RSA signature σ_A on message m from $\bar{\sigma}_A$ (more details are provided later). If Bob does not receive the value of σ_2 or only receives an invalid σ_2 from Alice timely, he applies help from the TTP via the dispute resolution protocol before the deadline t expires (see Section 3.3).

The following is further explanation of our signature exchange protocol. Firstly, the interactive protocol exploited

in step (2) is exactly the confirmation protocol for RSA undeniable signatures by Gennaro et al. [24], with respect to the private key (d_1, e_1) and the public key (n, w, σ_w). Note that similar approaches are used to construct e-payment protocol [15] and certified e-mail system [5]. In [24], it is proved that a successful execution of this zero-knowledge protocol guarantees that $\sigma_1 = \beta h(m)^{d_1} \mod n$, where $\beta \in \{1, -1, \alpha_1, \alpha_2\}$ and α_i's $(i = 1, 2)$ denote the two non-trivial elements of order 2. In this case, Bob accepts σ_1 as valid and sends his signature σ_B on contract m to Alice in step (3), since he is convinced that another partial signature σ_2 can be revealed by either Alice or the TTP. After that, if Alice does not reveal the value of σ_2 or only sends invalid σ_2 to Bob before the deadline t, Bob resorts to the TTP to get the correct value of σ_2. If Alice honestly reveals $\sigma_2 = h(m)^{d_2} \mod n$ to Bob in step (4), we have $h(m)^2 \equiv \bar{\sigma}_A^{2e} \mod n$, i.e., $\bar{\sigma}_A = \sigma_1\sigma_2 \mod n$ is valid. In such condition, Bob can recover the correct value of σ_A from $\bar{\sigma}_A$ by using the following *recovery algorithm*:

(a) set $\sigma_A = \bar{\sigma}_A$, if $h(m) = \bar{\sigma}_A^e \mod n$;
(b) set $\sigma_A = -\bar{\sigma}_A \mod n$, if $h(m) = -\bar{\sigma}_A^e \mod n$;
(c) get σ_A by factoring n, else, i.e., $h(m) \neq \pm\bar{\sigma}_A^e \bmod n$.

We describe how Bob can factor n and then get the value of $\bar{\sigma}_A$ in case (c), i.e., $h(m)^2 = \bar{\sigma}_A^{2e} \mod n$ but $h(m) \neq \pm\bar{\sigma}_A^e \mod n$. Note that the equality $h(m)^2 = \bar{\sigma}_A^{2e} \mod n$ implies that $\bar{\sigma}_A = \beta h(m)^d \mod n$, where $\beta \in \{1, -1, \alpha_1, \alpha_2\}$. When $\beta = \pm 1$, corresponding to cases (a) and (b), Bob can easily find the value of σ_A. So we conclude that case (c) means $\bar{\sigma}_A = \alpha_i h(m)^d \mod n$, $i = 1$ or 2. Recall that $\mathrm{ord}(\alpha_i) = 2$ and e is an odd number (due to $e \in \mathbb{Z}^*_{\phi(n)}$ and $\phi(n) = 4p'q'$), so we have $\bar{\sigma}_A^e = (\alpha_i h(m)^d)^e \mod n = \alpha_i h(m) \mod n$. Therefore, Bob can get the value of α_i by computing $\alpha_i = \bar{\sigma}_A^e h(m)^{-1} \mod n$. It is well known that with the knowledge of such a non-trivial element of order 2, Alice's RSA modulus n can be easily factored, i.e., $(\alpha_i - 1)$ and $(\alpha_i + 1)$ are the two prime factors of n. Consequently, Bob can get Alice's private key d by using extended Euclidean algorithm, and then obtain the value σ_A by computing $\sigma_A = h(m)^d \mod n$.

Based on the above discussion, we conclude that case (c) does not happen in the real world unless Alice wants to reveal her private key. That is, if Alice reveals $\sigma_1 = \alpha_i h(m)^{d_1} \mod n$ and $\sigma_2 = h(m)^{d_2} \mod n$, Bob will not only always recover her signature σ_A on contract m, but also could derive her private key d (and then forge signatures). So we ignore case (c) in the discussions hereafter under an implicit assumption that any user does not want to compromise his/her own private key.

3.3 Dispute Resolution Protocol

If Bob has sent his signature σ_B to Alice but does not receive the value of σ_2 or only receives an invalid σ_2 from Alice before the deadline t, then he sends the TTP $(C_A, V_A, m, \sigma_1, \sigma_B)$ to apply dispute resolution. Upon receiving Bob's application, the TTP performs as follows:

(1) The TTP first verifies whether C_A, V_A, and σ_B are Alice's valid certificate, voucher, and Bob's signature on contract m, respectively. After that, the TTP checks whether the deadline t embedded in m expires, and whether Alice, Bob and itself are the correct parties specified in m. If any validation fails, the TTP sends an error message to Bob. Otherwise, continue.

(2) Then, the TTP computes $\sigma_2 = h(m)^{d_2} \mod n$, and checks whether $h(m)^2 \equiv (\sigma_1\sigma_2)^{2e} \mod n$. If this equality holds, the TTP sends (m, σ_2) to Bob and forwards (m, σ_B) to Alice. Otherwise, i.e., $h(m)^2 \neq (\sigma_1\sigma_2)^{2e} \mod n$, the TTP sends an error message to Bob.

In the following, we explain why our dispute resolution protocol works. Since the TTP sets $\sigma_2 = h(m)^{d_2} \mod n$, we conclude that $h(m)^2 \equiv (\sigma_1\sigma_2)^{2e} \mod n$ if and only if $\sigma_1 \equiv \beta h(m)^{d_1} \mod n$, where $\beta \in \{1, -1, \alpha_1, \alpha_2\}$. That is, the TTP can determine whether Bob has sent a valid σ_1 to apply dispute resolution by checking $h(m)^2 =? (\sigma_1\sigma_2)^{2e} \mod n$. If this equality holds, the TTP reveals the correct value of σ_2 to Bob and forwards Bob's signature σ_B on contract m to Alice. After getting the correct σ_2, Bob can recover Alice's signature σ_A on contract m by employing the recovery algorithm given in previous section. In the case of $h(m)^2 \neq (\sigma_1\sigma_2)^{2e} \mod n$, the TTP knows that Bob is a cheater, and so only sends an error message to him.

Note that if the σ_1 sent to the TTP is prepared as $\sigma_1 = \alpha_i h(m)^{d_1} \mod n$, the TTP can also get Alice's private key d as Bob does.

Remark 1. Deadline t is a very important parameter in our protocol. If Bob receives valid σ_1 at a time which is very close to the deadline t, he should not reveal his signature σ_B to Alice. In this situation, Bob could have several choices to guarantee the fairness: (1) Ignore this protocol instance; (2) Get valid σ_2 from the TTP directly by initiating dispute resolution protocol; or (3) Require Alice use a new deadline t' and run the signature exchange protocol again.

4. SECURITY DISCUSSION

Based on the descriptions and discussions presented in last section, we know that in the normal situation, i.e., both involved parties are honest and the communication channel is in order, each of the two parties can get the other's signature correctly, and the TTP is not involved. In other words, our scheme is *complete* and *optimistic*. At the same time, if Bob shows the partial signature σ_1 with the proof $(c, \bar{r}, i, j, r)$ to others, nobody (other than Alice and the TTP) believes that σ_1 is indeed Alice's partial signature on contract m. Because, for any contract m, Bob himself can simulate such a proof for any (valid or invalid) σ_1 as follows: By choosing two random numbers i and j, then set $c = \sigma_1^{2i}\sigma_w^j \mod n$, $r = h(m)^{2i}w^j \mod n$, and $\bar{r} = commit(r)$. Furthermore, such a simulated proof is computationally indistinguishable from the real proof generated by Alice and Bob together. Therefore, the proposed protocol is also *abuse-free*.

Moreover, our protocol overcomes the security flaw in Park et al.'s scheme. Namely, if Alice is honest, the TTP cannot derive Alice's private key d from d_2 and other public information. Otherwise, the RSA signature scheme can be broken as follows. For any RSA public key (n, e), an attacker first chooses an even number d_2, and then inquiries the signing oracle for a polynomial number of adaptively chosen messages $m^{(i)}$. Then, from the corresponding answers $\sigma^{(i)}$, the attacker computes $\sigma_1^{(i)} = \sigma^{(i)}(h(m)^{d_2})^{-1} \bmod n$. Finally, the attacker calls the TTP as a subroutine to get the private key d. In fact, the above reduction is also valid to prove that except Alice herself, anybody (including the TTP) cannot forge a valid partial signature σ_1 for a new message with non-negligible probability. Formal proofs can be obtained by straightforwardly adapting the techniques

of Bellare and Sandu (see the 5th paragraph on page 5 of [11]). In other words, under the assumption that the RSA is intractable [36], the proposed protocol is provably secure in the random oracle model [10].

In addition, the TTP is *stateless* in our contract signing protocol, because it does not need to keep any state information related to each protocol instance. However, the schemes in [2, 3, 23] all require the TTP maintain a database to remember and search state information. Otherwise, a dishonest party could cheat successfully and then breach fairness. Namely, in those schemes, the TTP has to correctly record whether a specific protocol instance is solved or aborted after receiving the application from a particular party. So the TTP's workload and liability in our solution are reduced significantly. Hence, the cost of pay for the TTP can be cut accordingly, and performance of the TTP could be further improved. Obviously, this property is truly meaningful for a practical system. The compatibility is met naturally, since our basic goal is to define each party's commitment to a contract as his/her standard signature on the contract, instead of a signature satisfying some special structures [3, 33, 7]. As we have mentioned in Introduction, this is also an appealing property since the contract signing protocol can be conveniently integrated into existing softwares for electronic transactions.

Similar to the approach adopted by Micali in [33], a reasonable deadline t is added in each contract, hence the execution of a protocol instance will be terminated in a predetermined time limit, i.e., no later than the expiration of deadline. The result is that each party is free of liability to his/her partial commitment to the contract after the deadline t. The key point is that after the deadline specified in a contract, the TTP does not accept a dispute resolution application related with that contract. More discussion on this issue could be found in [33].

Now, we discuss the most important security property for a fair exchange protocol: fairness. That is, we have to show that in our scheme, any of the two involved parties cannot take advantage over the other in the process of signature exchanging even if he or she behaves dishonestly. We classified our discussion into two cases: (1) Alice is honest, but Bob is cheating; and (2) Bob is honest, but Alice is cheating. For simplicity, however, the effect of deadline on the fairness is not explained explicitly below.

Case 1: *Alice is honest, but Bob is cheating.* First of all, according to the results of Gennaro et al. [24] and Bellare et al. [11], except Alice and the TTP, any adversary including Bob cannot forge signatures σ_1 or σ_2 for a new message m' with non-negligible probability even if he has adaptively interacted with Alice and/or the TTP polynomial times (in the security parameter k). This means that nobody can generate valid σ_1 except Alice, and that nobody can generate valid σ_2 except Alice and the TTP.

Case (1) implies that in step (1) of our signature exchange protocol, Alice first properly computes $\sigma_1 = h(m)^{d_1} \mod n$, and sends the triple (C_A, V_A, σ_1) to Bob, where C_A is Alice's public key certificate issued by a trusted CA, and V_A is Alice's valid voucher created by the TTP. The purpose of step (2) in our signature exchange protocol is that Alice interactively convinces Bob to accept valid σ_1 in a zero-knowledge proof way. According to Theorem 1 in [24], we know that even if Bob cheats in any possible way, he cannot learn other information except σ_1 is valid, i.e., $\sigma_1 = \beta h(m)^{d_1} \mod n$, for some $\beta \in \{1, -1, \alpha_1, \alpha_2\}$. Actually, β must be 1 since Alice is honest in this setting. This also implies that Bob cannot factor Alice's RSA modulus n by first getting a non-trivial element of order 2.

Upon receiving the valid value of σ_1, Bob has to make a choice whether he needs to send his signature σ_B on contract m to Alice. If Bob does, honest initiator Alice returns back her second partial signature $\sigma_2 = h(m)^{d_2} \mod n$ as Bob expects. In such situation, Bob gets Alice's signature on contract m by setting $\sigma_A = \sigma_1\sigma_2 \mod n$, while Alice also obtains Bob's signature σ_B simultaneously. If Bob does not send σ_B or only sends an incorrect σ_B to Alice, he cannot get the value of σ_2 from Alice in step (4). Furthermore, in this setting, Bob also cannot get the value of σ_2 from the TTP so that Alice does not obtain his signature σ_B. The reason is that in our dispute resolution protocol, to get the value of σ_2 from the TTP Bob has to submit valid σ_1 and σ_B to the TTP. Once those values are submitted, Bob indeed gets σ_2 from the TTP but Alice receives (m, σ_B) from the TTP, too. Therefore, once again, Bob and Alice get the other's signature on contract m at the same time.

Case 2: *Bob is honest, but Alice is cheating.* In our signature exchange protocol, Alice may cheat in any or some of the following steps: step (1), step (2) and step (4). First of all, according to the specification of our signature exchange protocol, to get the signature σ_B on contract m from the honest responder Bob, the initiator Alice has to convince Bob accepting σ_1 as a valid partial signature in the step (2). Recall that step (2) is exactly Gennaro et al.'s confirmation protocol for RSA undeniable signatures, and that their protocol satisfies the property of soundness (Theorem 1, [24]). The *soundness* means that the possible cheating Alice (prover), even computationally unbounded, cannot convince Bob (verifier) to accept an invalid σ_1 as valid with non-negligible probability. Therefore, we conclude that to get σ_B from Bob, Alice has to send valid σ_1 (with valid C_A and V_A) in step (1) and perform honestly in step (2). In other words, Alice has to send $\sigma_1 = \beta h(m)^{d_1} \mod n$ to Bob unless she does not want to get Bob's signature σ_B, where $\beta \in \{1, -1, \alpha_1, \alpha_2\}$. Based on our discussion in previous section, we know that Alice is not so silly by preparing and sending $\sigma_1 = \alpha_i h(m)^{d_1} \mod n$ to Bob. Otherwise, Bob can drive her private key d (and then computes signature σ_A), though he indeed gets Bob's signature σ_B. Therefore, to get signature σ_B from Bob, Alice has to compute $\sigma_1 = \pm h(m)^{d_1} \mod n$ and send it to Bob. In this situation, Bob receives valid $\sigma_1 = \pm h(m)^{d_1} \mod n$ from Alice before Alice gets valid σ_B from Bob. After that, step (4) is the only one possible cheating chance for Alice, i.e., she may refuse to reveal σ_2 or just send an incorrect σ_2 to Bob. However, this cheating behavior does not harm Bob essentially, since he can get the value of σ_2 from the TTP via our dispute resolution protocol. The reason is that Bob has received valid σ_1 before he sends σ_B to Alice. After getting the value of σ_2 from the TTP, Bob can recover Alice's signature σ_A according to the recovery algorithm specified in section 3.2. Therefore, in case (b) where Bob is honest but Alice is dishonest, Alice cannot get Bob's signature such that Bob does not obtain her signature.

Based on the above analysis, we conclude that the proposed protocol is not advantageous to any dishonest party.

Table 1. Comparison of Efficiency

	Asokan et al. [2, 3]	Ateniese [4]	Park et al. [34]	Our Protocol
Number of Exponentiations.	75	13.3	7	10.5
Data to Be Exchanged (bytes)	8000	916	600	1216

In other words, our contract signing protocol satisfies the property of *fairness*.

5. EFFICIENCY

Table 1 shows the comparison of efficiency between our new protocol and several other RSA-based solutions, i.e., Asokan et al.'s scheme [2, 3] from verifiable escrow, Ateniese's scheme [4] from verifiably encrypted signature, and Park et al.'s scheme [34] from multisignature. In the comparison, we analyze the overheads of computation and communication in the signature exchange protocol needed by *both* Alice and Bob in *normal* case. In other words, the operations of the dispute resolution protocol are not discussed here. Moreover, we take the number of modular exponentiations as the computational cost since exponentiation is the most expensive cryptographic operation in the finite field $\mathbb{Z}_n$. In addition, note that a modular exponentiation in $\mathbb{Z}_n$ requires about $1.5 \times |n|$ modular multiplications, and that exponentiation of the form $a_1^{x_1} a_2^{x_2}$ is only equivalent to 1.167 single exponentiation by means of an exponent array (pages 618 of [32]).

For comparison, we make similar but different assumptions from [4, 34]. Namely, we assume that the length of RSA modulus n is 1200-bit, and that the hash function $h(\cdot)$ has 128-bit fixed output. For simplicity, we also assume that σ_B could be generated and verified by one modular exponentiation separately, and that the voucher V_A can be validated by one modular exponentiation, too. However, the overhead related to Alice's certificate C_A is excluded as did in [4, 34], since such validation may be as simple as to check the certificate list on CA's web site.

Some numbers listed in Table 1 are different from the results that appeared in [4, 34], since we take into consideration all exponentiations needed in the signature exchange protocols by both Alice and Bob, while Anteniese *only* concerned the amount of each signature algorithm, and Park et al. [34] *only* considered the overhead required for creating/verifying the fairness primitives (i.e., σ_1 and V_A). For example, Anteniese *did not* include the overheads of creating and checking the proof for proving the equality of two discrete logarithms, while Park et al. *did not* estimate the overheads of generating and verifying Alice's signature σ_A. Our analysis is more reasonable since it accurately reflects what happens in practice. In addition, note that the numbers for the Asokan et al.'s scheme were taken from [34] directly.

According to the results in Table 1, the computational efficiency of our scheme is in the middle between Park et al.'s scheme and Ateniese's scheme, while the communication cost of our scheme increases by 103% and 33% more than that of Park et al.'s scheme and Ateniese's scheme, respectively. The overhead of communication becomes larger naturally, since our scheme exploits interactive protocol to prove the validity of σ_1. The bonus in our new scheme is that Bob cannot show the validity of σ_1 to other parties, i.e., abuse-freeness, as we discussed before. We believe that this cost deserves the advantage of our scheme in the situations where the intermediate results should not be revealed unfairly. Actually, all those three schemes are suited for most applications where the cost of communication is not the main concern.

6. CONCLUSION

In this paper, based on the standard RSA signature scheme, we proposed a new digital contract signing protocol that allows two potentially mistrusted parties to exchange their digital signatures on a contract in an efficient and secure way. Like the existing RSA-based solutions, the new protocol is fair and optimistic, i.e., two parties get or do not get the other's digital signature simultaneously, and the trusted third party is only needed in abnormal cases that occur occasionally. However, different from all previous RSA-based contract signing protocol, the proposed protocol is further abuse-free. That is, if the contract signing protocol is executed unsuccessfully, each of the two parties cannot show the validity of intermediate results generated by the other party to outsiders. In other words, each party cannot convince an outsider to accept the partial commitments coming from the other party. This is an important security property for contract signing, especially in the situations where partial commitments to a contract may be beneficial to a dishonest party or an outsider. Technical details are provided to show that our protocol meets a number of desirable properties, not only those just mentioned.

In addition, exploiting some techniques of Park et al. [34], our protocol can be adapted to fair payments in e-commerce (though their solution has a security flaw). In this setting, one customer purchases a digital goods from a merchant via the Internet by paying a digital check or cash. The extended scheme could implement such an electronic transaction between two parties fairly. That is, it is guaranteed that the customer gets the digital goods from the merchant if and only if the merchant gets the money from the customer.

Finally, using the technique of threshold RSA signature introduced by Shoup [37], the proposed protocol could be extended for the scenarios where the trust on a single TTP needs to be distributed into multiple TPPs, or a contract is required to be signed only by a given quota of members cooperatively.

7. ACKNOWLEDGMENTS

The author would like to thank all five anonymous referees for their helpful and detailed comments on the paper, as well as Dr. Feng Bao, Prof. Robert H. Deng, and Dr. Jianying Zhou for good discussions on the topic of fair-exchange.

8. REFERENCES

[1] M. Abadi, N. Glew, B. Horne, and B. Pinkas. Certified email with a light on-line trusted third party:

Design and implementation. In: *Proc. of 2002 International World Wide Web Conference* (*WWW'02*), pp. 387-395. ACM press, 2002.

[2] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In: *EUROCRYPT'98*, LNCS 1403, pp. 591-606. Springer-Verlag, 1998.

[3] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4): 591-606, 2000.

[4] G. Ateniese. Efficient verifiable encryption (and fair exchange) of digital signature. In: *Proc. of AMC Conference on Computer and Communications Security* (*CCS'99*), pp. 138-146. ACM Press, 1999.

[5] G. Ateniese and C. Nita-Rotaru. Stateless-receipient certified E-mail system based on verifiable encryption. In: *CT-RSA'02*, LNCS 2271, pp. 182-199. Springer-Verlag, 2002.

[6] F. Bao, R.H. Deng, and W. Mao. Efficient and practical fair exchange protocols with off-line TTP. In: *Proc. of IEEE Symposium on Security and Privacy*, pp. 77-85, 1998.

[7] F. Bao, G. Wang, J. Zhou, and H. Zhu. Analysis and improvement of Micali's fair contract signing protocol. In: *Information Security and Privacy* (*ACISP'04*), LNCS 3108, pp. 176-187. Springer-Verlag, 2004.

[8] F. Bao. Colluding attacks to a payment protocol and two signature exchange schemes. In: *ASIACRYPT 2004*, LNCS 3329, pp. 417-429. Springer-Verlag, 2004.

[9] P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In: *CRYPTO 2002*, LNCS 2442, pp.354-368. Springer-Verlag, 2002.

[10] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In: *Proc. of the 1st ACM conference on Computer and communications Security* (*CCS'93*), pp. 62-73. ACM press, 1993.

[11] M. Bellare and R. Sandhu. The security of practical two-party RSA signature schemes. Manuscript, 2001. `http://www-cse.ucsd.edu/users/mihir/papers/`.

[12] M. Ben-Or, O. Goldreich, S. Micali, and R. L. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1): 40-46, 1990.

[13] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-group signature scheme. In: *Public Key Cryptography - PKC'03*, LNCS 2567, pp. 31-46. Springer-Verlag, 2003.

[14] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In: *ASIACRYPT 2001*, LNCS 2248, pp. 514-532. Springer-Verlag, 2001.

[15] C. Boyd and E. Foo. Off-line fair payment protocols using convertible signatures. In: *ASIACRYPT 1998*, LNCS 1514, pp. 271-285. Springer-Verlag, 1998.

[16] I.B. Damgård. Practical and provably secure release of a secret and exchange of signatures. *Journal of Cryptology*, 8(4): 201-222, 1995.

[17] R. Deng, L. Gong, A. Lazar, and W. Wang. Practical protocol for certified electronic mail. *Journal of Network and Systems Management*, 1996, 4(3): 279-297.

[18] Y. Dodis and L. Reyzin. Breaking and repairing optimistic fair exchange from PODC 2003. In: *Proc. of ACM Workshop on Digital Rights Management* (*DRM'03*), pp. 47-54. ACM press, 2003.

[19] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6): 637-647, 1985.

[20] S. Even and Y. Yacobi. Relations among public key signature schemes. Technical Report 175, Computer Science Dept., Technion, Israel, 1980.

[21] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In: *CRYPTO'86*, LNCS 263, pp. 186-194. Springer-Verlag, 1987.

[22] S. D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In: *Algorithmic Number Theory* (*ANTS'02*), LNCS 2369, pp.324-337. Springer-Verlag, 2002.

[23] J. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In: *CRYPTO'99*, LNCS 1666, pp. 449-466. Springer-Verlag, 1999.

[24] R. Gennaro, T. Rabin, and H. Krawczyk. RSA-based undeniable signature. *Journal of Cryptology*, 13(4): 397-416, 2000. A preliminary version of this paper appeared in the proceedings of *CRYPTO'97*.

[25] O. Goldreich. A simple protocol for signing contracts. In: *CRYPTO'83*, pp. 133-136. Plenum Press, 1984.

[26] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, April 1988, 17(2): 281-308.

[27] S. Gürgens, C. Rudolph, and H. Vogt. On the security of fair non-repudiation protocols. In: *Information Security Conference* (*ISC 2003*), LNCS 2851, pp. 193-207. Springer-Verlag, 2003.

[28] K. Imamoto and K. Sakurai. A cerified e-mail system with receiver's selective usage of delivery authortiy. In: *Indocrypt 2002*, LNCS 2551, pp. 326-338. Springer-Verlag, 2002.

[29] P. Liu, P. Ning, and S. Jajodia. Avoiding loss of fairness owing to process crashes in fair data exchange protocols. In: *International Conference on Dependable Systems and Networks* (*DSN'00*), pp. 631-640. IEEE Computer Society, 2000.

[30] S. Kremer and O. Markowitch. Selective receipt in cerified e-mail. In: *Indocrypt 2001*, LNCS 2247, pp. 136-148. Springer-Verlag, 2001.

[31] S. Kremer, O. Markowitch, and J. Zhou. An intensive survey of fair non-repudiation protocols. *Computer Communications*, 25(17): 1606-1621. Elsevier, Nov. 2002.

[32] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[33] S. Micali. Simple and fast optimistic protocols for fair electronic exchange. In: *Proc. of 22th Annual ACM Symp. on Principles of Distributed Computing* (*PODC'03*), pp. 12-19. ACM Press, 2003.

[34] J. M. Park, E. Chong, H. J. Siegel, and I. Ray. Constructing fair exchange protocols for e-commerce via distributed computation of RSA signatures. In: *Proc. of 22th Annual ACM Symp. on Principles of*

Distributed Computing (*PODC'03*), pp. 172-181. ACM Press, 2003.

[35] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In: *CRYPTO 1991*, LNCS 576, pp. 129-140. Springer-Verlag, 1991.

[36] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, Feb. 1978, 21(2): 120-126.

[37] V. Shoup. Practical threshold signatures. In: *EUROCRYPT 2000*, LNCS 1807, pp. 207-220. Springer-Verlag, 2000.

[38] J. Zhou and D. Gollmann. A fair non-repudiation protocol. In: *Proc. of the IEEE Symposium on Security and Privacy*, pp. 55-61. IEEE Computer Press, 1996.

[39] J. Zhou and D. Gollmann. Certified electronic mail. In: *Computer Security - ESORICS'96*, LNCS 1146, pp. 160-171. Springer-Verlag, 1996.