

Constructing Fair-Exchange Protocols for E-commerce Via Distributed Computation of RSA Signatures

Jung Min Park
School of Electrical and
Computer Engineering
Purdue University
West Lafayette, IN
47907-1285
parkjm@ecn.purdue.edu

Edwin K. P. Chong
Dept. of Electrical and
Computer Engineering
Dept. of Mathematics
Colorado State University
Fort Collins, CO 80523-1373
echong@colostate.edu

Howard Jay Siegel
Dept. of Electrical and
Computer Engineering
Dept. of Computer Science
Colorado State University
Fort Collins, CO 80523-1373
hj@colostate.edu

ABSTRACT

Applications such as e-commerce payment protocols, electronic contract signing, and certified e-mail delivery require that fair exchange be assured. A fair-exchange protocol allows two parties to exchange items in a fair way so that either each party gets the other's item, or neither party does. We describe a novel method of constructing very efficient fair-exchange protocols by distributing the computation of RSA signatures. Specifically, we employ multisignatures based on the RSA-signature scheme. To date, the vast majority of fair-exchange protocols require the use of zero-knowledge proofs, which is the most computationally intensive part of the exchange protocol. Using the intrinsic features of our multisignature model, we construct protocols that require no zero-knowledge proofs in the exchange protocol. Use of zero-knowledge proofs is needed only in the protocol setup phase—this is a one-time cost. Furthermore, our scheme uses multisignatures that are compatible with the underlying standard (single-signer) signature scheme, which makes it possible to readily integrate the fair-exchange feature with existing e-commerce systems.

Categories and Subject Descriptors

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

General Terms

Algorithms, Security, Design

Keywords

Fair-exchange protocols, e-commerce, multisignatures, RSA signatures, zero-knowledge proofs.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PODC'03, July 13–16, 2003, Boston, Massachusetts, USA.
Copyright 2003 ACM 1-58113-708-7/03/0007...\$5.00.

1. INTRODUCTION

Fueled by the exponential growth in the number of people with access to the Internet, e-commerce transactions via the Internet have become a major part of our economy. To date, the majority of e-commerce transactions involve exchanging the customer's credit-card number for the merchant's guarantee of merchandise delivery (i.e., an electronic receipt). In such transactions, items being exchanged have no intrinsic value, and thus the importance of ensuring a "fair exchange" has not received widespread recognition. In a fair exchange, either each player gets the other player's item, or neither player does. In the foreseeable future, data with significant intrinsic value, such as financial data, medical data, software, and electronic forms of money (e.g., electronic checks and electronic cash), will be exchanged regularly over the Internet. In such instances, ensuring fairness is critical if the participants are to be protected from fraud. Furthermore, future applications such as payment protocols via electronic money [13, 18], electronic contract signing [4, 21], and certified e-mail delivery [1, 6, 7] require that fair exchange be assured. As more business is conducted over the Internet, the fair-exchange problem is gaining greater importance.

Significant effort has been devoted to the study of the fair-exchange problem. For an overview of fair-exchange protocols, we refer the reader to [29]. Fair-exchange protocols can be broadly categorized into three types: (i) gradual exchange protocols, (ii) protocols requiring an online trusted third party (TTP), and (iii) protocols requiring an offline TTP. There are some protocols [18, 31] (not included in the three categories above) that do not ensure fairness but provide a weaker form of protection: the gathering of evidence during execution so that if one party obtains the other's item without sending his, the dishonest party can be prosecuted using the evidence. These protocols are efficient, but, as mentioned above, do not guarantee fairness.

In gradual exchange protocols [10, 21], the probability of fair exchange is gradually increased over several rounds of message exchanges. These protocols are impractical because extensive amounts of communication are needed. Furthermore, proofs of their security rely on both parties having equal computational power, which is unrealistic for most applications.

In fair-exchange protocols with an on-line TTP [7, 18], a TTP is directly involved in every exchange, and must be available for the entire duration of the exchange. The protocol itself is relatively simple and computationally efficient. However, maintaining a TTP that needs to be on-line constantly can be expensive. Moreover, the TTP can become a bottleneck, and pose scalability problems.

Protocols with an offline TTP provide a method for ensuring fair exchange while avoiding the problems faced by the two other types of protocols. In this type of a protocol, the TTP is involved in the protocol only if one of the parties behaves unfairly or aborts the protocol prematurely; otherwise the TTP is never involved in the protocol. Hence, these protocols are also known as “optimistic” fair-exchange protocols. In terms of practicality, optimistic protocols seem to be most appealing, and hence we focus on this approach.

Most, if not all, previously proposed optimistic fair-exchange protocols in the literature are largely based on two different types of protocol frameworks—one is the framework proposed by Asokan et al. [3, 4], and the other is the framework used by Bao et al. [8]. However, each protocol employs a different technique for constructing the cryptographic primitive that ensures fairness—this primitive is the cornerstone of the fair-exchange protocol, and its design poses the greatest technical challenge. To clarify how such a *fairness primitive* is used in an exchange protocol, we present an example of a basic optimistic fair-exchange protocol. The protocol framework is essentially the same as what is proposed in [8] excluding the specific fairness primitive used in the protocol.

Let Alice and Bob be two players trying to exchange their respective digital signatures σ_A and σ_B on a message M (known a priori to both parties), and let Charlie represent a TTP. In the first step of the protocol, one of the items that Alice sends to Bob is her “commitment” to the transaction, which we denote as c_A . This value c_A has no intrinsic value, but serves as Alice’s commitment to the exchange. Along with c_A , Alice has to send a “voucher” V that provides proof of the following: (i) there exists a tamperproof one-to-one link between c_A and Alice’s signature σ_A , and (ii) Charlie can compute σ_A using c_A if needed. We call c_A and V collectively the fairness primitive. The protocol is executed as follows:

EXCHANGE.

1. Alice sends Bob c_A and its associated voucher V .
2. Bob verifies V and c_A , and, if valid, sends his signature σ_B to Alice. Otherwise, he stops the protocol.
3. If σ_B is valid, Alice sends σ_A to Bob, otherwise she stops the protocol.
4. If σ_A is valid, Bob ends the exchange protocol. Otherwise, Bob initiates the dispute resolution protocol.

DISPUTE RESOLUTION.

1. If σ_A is invalid (in Step 4), or if Bob fails to receive anything from Alice (in Step 3), he initiates a dispute resolution protocol with Charlie, and sends c_A , V , and σ_B to him.

2. Charlie verifies whether c_A , V , and σ_B are valid. If they are valid, Charlie computes σ_A (which he computes using c_A), and sends it to Bob. He also forwards σ_B to Alice.

Note that, in the last step of the dispute resolution protocol, Bob’s signature σ_B needs to be forwarded to Alice in case Bob is dishonest. This is necessary to prevent the scenario where Bob attempts to obtain σ_A via the dispute resolution protocol after intentionally aborting the exchange protocol after Step 1.

Although the example illustrates a signature-exchange protocol, this basic model can readily be adapted to almost any exchange protocol for e-commerce. For instance, in an e-commerce payment protocol, the signature of Alice (acting as a customer) would correspond to her electronic check, and the signature of Bob (acting as a merchant) would be replaced by some digital merchandise (e.g., MP3 music files, MPEG-4 media files, or e-books). In such a scenario, message M would include information such as Alice’s unique identity, Bob’s unique account number, price of merchandise, description of merchandise, and date of transaction.

Observe that the above protocol framework does not ensure a *timely termination*, and, as a result, does not ensure fairness when time-sensitive items are exchanged. Consider the following scenario: after Step 1 of the exchange protocol, Bob aborts the exchange, and initiates a dispute resolution protocol after a long delay. Unaware of Bob’s intentions, Charlie sends σ_A to Bob, and forwards σ_B to Alice. If the intentional delay (caused by Bob) is sufficiently long and σ_B is time-sensitive, then Alice suffers a loss. For example, if σ_B represents a digital airline ticket with an expiration date, the ticket is useless to Alice after that date. To prevent such cases, a *timestamp* can be attached to c_A that specifies an expiration time. After this expiration time, Charlie would refuse to execute the dispute resolution protocol with Bob. Unlike the protocol framework used by Bao et al. [8], the framework proposed by Asokan et al. [3, 4] ensures timely termination without the use of timestamps. However, it is more complicated and requires the TTP to store state information. Although our techniques can be applied to *both* frameworks, we choose the framework of Bao et al. to illustrate how our techniques (based on multisignatures) are applied to construct efficient optimistic protocols. We emphasize that our main contribution is the construction of a novel fairness primitive, and is not the proposal of a new protocol framework.

In this paper, we present a novel approach for constructing fair-exchange protocols. We employ RSA-based *multisignatures* to construct efficient optimistic protocols. Note that our multisignature paradigm is quite different from the conventional model of multisignatures, and has distinctive features that are used to create the fairness primitive. Our approach ensures fairness by splitting an RSA private key into two parts—the signer holds both parts while the TTP holds just one of the parts. Similar key-splitting techniques can be used in a variety of applications. For instance, MacKenzie and Reiter [25] use key splitting to minimize the vulnerability of devices that perform networked private-key operations (i.e., signatures or decryptions) against attacks.

To date, all optimistic protocols require the use of zero-knowledge proofs (via zero-knowledge protocols) in the exchange protocol, with the exception of schemes based on one-time tokens (see Section 2). This technique, however, is not a truly optimistic protocol—it only supports a limited number of exchanges, and the TTP has to be involved intermittently to replenish the exhausted one-time tokens. Zero-knowledge protocols constitute the most computationally intensive part of the exchange protocol. In our approach, we do not use any zero-knowledge proofs in the exchange protocol, which significantly increases efficiency. To the best of our knowledge, our scheme is the only optimistic fair-exchange protocol to date that achieves this without suffering the drawbacks mentioned above (i.e., intermittent interactions with the TTP). In our approach, use of zero-knowledge proofs is needed, however, in the protocol setup phase (called the registration protocol), but this is only a one-time cost. This phase needs to be executed only once, after which it can support any number of exchanges. The end result is a fair-exchange protocol that is very efficient in terms of computation and communication overhead.

In the next section, we discuss some related work relevant to optimistic protocols. We discuss the technical background in Section 3—an overview of multisignatures, our multisignature model, the RSA signature algorithm [30], and our RSA-based multisignature scheme is given. In Section 4, we describe how our multisignature paradigm can be used to construct optimistic protocols. In Section 5, we compare the efficiency of our fair-exchange protocol with two other schemes in terms of computation and communication overhead. The concluding remarks are given in Section 6. In the Appendix, we discuss certain security issues involved in key splitting.

2. RELATED WORK

Optimistic fair-exchange protocols can be categorized into four types depending on how the commitment and voucher (see Section 1) are created: (i) protocols using *verifiable escrows*, (ii) protocols using *verifiable encryptions*, (iii) protocols using *convertible undeniable signatures*, and (iv) protocols using *one-time tokens*.

In [3, 4], Asokan et al. propose an optimistic protocol that uses cryptographic primitives called verifiable escrows to produce the fairness primitive. A verifiable escrow of a signature is created as follows: first, the signer reduces her signature to a particular homomorphic pre-image of the signature; then, a cut-and-choose interactive zero-knowledge proof, in combination with an ordinary escrow scheme, is used to verifiably escrow the homomorphic pre-image. This technique can be applied to almost any signature scheme as long as there is a way to reduce the signature into a homomorphic pre-image. Its drawback is that it requires Alice and Bob to execute considerable amounts of computations during the interactive zero-knowledge proof. Furthermore, communication overhead is relatively high—the amount of data transmitted (by both parties) is on the order of several thousand bytes. The approach of Asokan et al. was later generalized by Camenisch and Damgard [14], but the computational burden remains.

Fair-exchange protocols using verifiable encryption was pro-

posed by Ateniese [5] and Bao et al. [8] independently. These protocols apply ad-hoc techniques to create an encryption of a signature that conforms to the signature type. For instance, a verifiable encryption of an RSA signature [30] is created by encrypting it via the ElGamal encryption scheme [20] using the TTP’s public key. To prove that the verifiable encryption was correctly generated without revealing the signature itself, the signer uses a zero-knowledge proof. Note that the verifiable encryption corresponds to Alice’s commitment, and the zero-knowledge proof corresponds to the voucher. The primary advantage of protocols based on verifiable encryption is its efficiency. Ateniese [5] shows that his schemes require considerably less computation and communication overhead than the approach of Asokan et al. However, because the verifiable encryption approach is based on ad-hoc techniques, it can be vulnerable to security flaws—the verifiable encryption of Guillou-Quisquater signatures proposed in [8] was shown to be insecure [5, 13].

In [13, 17], the authors show that fair-exchange protocols can be constructed efficiently using cryptographic primitives known as convertible undeniable signatures. Convertible undeniable signatures [11, 19, 28] require the signer’s assistance for verification, but have a feature that allows the signatures to be converted into “universally verifiable” signatures. That is, the original signatures can be converted into signatures that can be verified by anyone without the assistance of the signer. Using this feature, fair-exchange protocols can be constructed. This approach also utilizes zero-knowledge proofs in the exchange protocol—in [13], a zero-knowledge proof is used to prove the validity of the signer’s convertible undeniable signature. The problem with this approach is that most converted signatures are not *compatible* with standard signatures. That is, the converted signature does not have the same form as a standard signature, and hence needs a verification algorithm that is different from the algorithm used to verify standard signatures. The scheme used in [13] is an exception (i.e., it is compatible with standard RSA signatures), but their approach is limited to the RSA signature scheme.

Another method of constructing optimistic protocols is to use one-time verifiable encryptions constructed from one-time tokens (also called coupons) issued by the TTP [4, 5]. The drawback of this scheme is that the signer has to contact the TTP for new tokens once all the previously issued tokens are exhausted. Protocols based on one-time tokens require the use of zero-knowledge proofs for some (but not all) signature types (see Section IX of [4]).

It should be noted that the difference between the techniques categorized above is, in some cases, obscure, and certain schemes can belong to more than one category.

3. TECHNICAL BACKGROUND

3.1 Multisignatures

If multiple signers need to sign a single message, the naive approach would be to let the signers create their own signature and concatenate the individual signatures. Obviously, this causes an expansion in the signature data. A *multisignature* scheme [12, 23, 27] allows multiple signers to sign a single message efficiently such that the resulting multisignature has little or no difference in size with an individual

signature.

The players in a typical multisignature scheme are $n \geq 2$ signers and a verifier. For each signer i , there is a private key sk_i , which we call the signer's *partial private key*. Depending on the multisignature scheme, a *partial public key* pk_i might also be assigned to each signer. After each signer creates its partial signature σ_i using its partial private key, these are combined to form the multisignature σ , and given to the verifier. After receiving the multisignature, the verifier verifies its correctness using the *joint public key* pk corresponding to the *joint private key* sk . The joint private key has an explicit algebraic relation with the partial private keys, that is,

$$sk = sk_1 \oplus \dots \oplus sk_n,$$

where \oplus denotes some binary operation (e.g., addition or multiplication). This relation depends on how the multisignature is constructed. If the partial and joint keys are created correctly, it should be infeasible to create a multisignature using only a proper subset of the partial private keys; more precisely, if the joint private key is $sk = sk_1 \oplus \dots \oplus sk_n$, then it should be infeasible to create a multisignature with a proper subset of $\{sk_1, \dots, sk_n\}$.

Unlike most convertible undeniable signatures, multisignatures are compatible with the underlying (single-signer) signature scheme. That is, the algorithm for verifying the multisignature is identical to the algorithm for verifying the underlying signature scheme.

3.2 Our Multisignature Paradigm

For our application, we use a modified version of the typical multisignature model. In our model, the players in a multisignature scheme include a *primary signer*, a *cosigner*, and a verifier—we fix the number of signers to two.¹ The cosigner has knowledge of his partial private key sk_2 only, while the primary signer has knowledge of her partial private key sk_1 as well as the cosigner's partial private key. The keys sk_1 and sk_2 are used to generate the primary signer's partial signature σ_1 and the cosigner's partial signature σ_2 , respectively. Such assignment of keys enables the primary signer to compute the multisignature σ without any cooperation from the cosigner, but prevents the cosigner from computing it on his own. (In contrast, in a typical multisignature scheme, each signer only has knowledge of his own partial private key so that every signer has to cooperate to compute a multisignature.) In addition, in our framework, there exists a partial public key pk_1 that is used to verify the primary signer's partial signature. However, there is no partial public key associated with the cosigner because the cosigner's signature does not need to be verified (in our model).

In our model, the cosigner acts as a sort of “arbitrator,” who holds his partial private key so that he has the ability to settle disputes that might arise between the primary signer and the verifier. In contrast, the primary signer is

¹We associate the primary signer with the female gender, and associate the cosigner and verifier with the male gender. The notations associated with the primary signer and cosigner have subscripts of one and two, respectively.

the “owner” of the multisignature, that is, the multisignature provides non-repudiation of the message signed by the primary signer. Imagine a scenario where Alice (primary signer) and Bob (verifier) are trying to exchange signatures in a fair way. Alice wants to commit to the transaction by providing “a token of commitment” but not her signature itself. Alice wants to convince Bob that she has made an honest commitment and that he has the guarantee of receiving her signature if he carries out the protocol honestly. If Charlie (cosigner) is trusted by both parties, it is possible to construct a fair-exchange protocol using our multisignature paradigm—a multisignature can be used as Alice's signature, and a corresponding partial signature can be used as her token of commitment. By providing the partial signature, Alice can make a commitment without giving an advantage to Bob (assuming the partial signature is of no value to Bob). Moreover, Bob is protected from Alice's dishonest behavior once he receives her partial signature because Charlie has the ability to create the multisignature using his partial private key sk_2 and Alice's partial signature σ_1 . Thus, after verifying the correctness of Alice's partial signature, Bob can safely send his signature to Alice.

Another important property of our model is that a trusted party (i.e., Charlie) guarantees the algebraic relation between the keys. The correct relationship between the keys ensures that Charlie can generate the multisignature using his partial private key and Alice's partial signature. In addition, it is assumed that *only* the joint public key pk (and *not* the partial public key pk_1) is certified by the certification authority (CA). Because the partial public key pk_1 is not certified, the only role that the partial signature can play is as a commitment to the transaction in which it is used and not as a cryptographic primitive for providing non-repudiation. For a signature to provide non-repudiation, there has to be a tamperproof link between the public key and the signer's identity, and this link is provided by the public key's certificate, issued by the CA. In contrast, the joint public key pk is certified, and hence the multisignature provides non-repudiation of the message signed by the entity that holds all the partial private keys—in our model, this is the primary signer.

The above properties of our multisignature paradigm enable us to construct efficient optimistic fair-exchange protocols without relying on costly cryptographic techniques.

3.3 The RSA Signature Scheme

Before we discuss our RSA-based multisignature scheme, we review the RSA (single-signer) signature scheme [30]. We consider the standard “hash-then-decrypt” variety.

The signature space is the set of integers modulo N , denoted as \mathbb{Z}_N where N is a product of two distinct primes p and q . The parameter N is chosen such that $2^{k-1} \leq N < 2^k$ holds for some security parameter k . The public key² is obtained by selecting a random integer e , $1 < e < \phi(N)$, such that $\gcd(e, \phi(N)) = 1$. Here, ϕ is the Euler totient function.³

²In some literature, (e, N) and d are called the public and private keys, respectively.

³ $\phi(N)$ denotes the number of integers in the interval $[1, N]$ that are relatively prime to N .

The private key is generated by finding the unique integer d , $1 < d < \phi(N)$, such that

$$ed \equiv 1 \pmod{\phi(N)}.$$

A signature σ on message M is created by computing

$$H(M)^d \pmod{N},$$

where $H: \{0, 1\}^* \rightarrow \mathbb{Z}_N$ is a public hash function. (Here, $\{0, 1\}^*$ denotes a binary string of arbitrary finite length.) To thwart attacks, the hash function should have the preimage, second-preimage, and collision-resistance properties (see p. 323, [26]). The signing space⁴ is \mathbb{Z}_N . The signature σ is considered to be a valid signature of M if

$$\sigma^e \pmod{N} = H(M).$$

3.4 Our RSA-Based Multisignature Scheme

Boyd [12] proposed an RSA-based multisignature scheme that allows two signers to compute a multisignature efficiently. The core idea behind his scheme is to *multiplicatively* split the private key d into two partial keys d_1 and d_2 , each associated with a different signer. That is,

$$d \equiv d_1 d_2 \pmod{\phi(N)}.$$

The multisignature computation is then based on the equation

$$H(M)^d \equiv H(M)^{d_1 d_2} \pmod{N}.$$

In [9], Bellare and Sandhu give security analyses for a set of signature protocols based on Boyd's scheme.

In our fair-exchange protocol, we employ an RSA-based multisignature scheme that also splits d into two partial keys, but the splitting is done *additively* instead of multiplicatively. Our multisignature scheme has the following features that set it apart from Boyd's scheme.

We restrict the modulus N to be a product of *safe primes* p and q , that is, p and q are primes such that $p = 2p' + 1$ and $q = 2q' + 1$ with p' and q' primes. Let \mathbb{Z}_N^* denote the multiplicative group of integers modulo N . The signing space is the set of quadratic residues modulo N , denoted as Q_N . By definition, $Q_N \subset \mathbb{Z}_N^*$ is the set of elements $a \in \mathbb{Z}_N^*$ such that there exists an $x \in \mathbb{Z}_N^*$ with $x^2 \equiv a \pmod{N}$. Note that Q_N is a cyclic subgroup of \mathbb{Z}_N^* , and that

$$|Q_N| = \frac{|\mathbb{Z}_N^*|}{4} = p'q', \quad (1)$$

where $|\cdot|$ denotes the order of a group. The private key d is split additively. That is,

$$d \equiv d_1 + d_2 \pmod{\lambda},$$

where $\lambda = p'q'$. Observe that the modulus used above is λ instead of $\phi(N)$ because the signing space is Q_N . The multisignature computation is then based on the congruence

$$H(M)^d \equiv H(M)^{d_1} \cdot H(M)^{d_2} \pmod{N},$$

where $H: \{0, 1\}^* \rightarrow Q_N$ is a public hash function. In addition to splitting d , we need to create a partial public key e_1 associated with the partial private key d_1 that satisfies

$$d_1 e_1 \equiv 1 \pmod{\lambda}.$$

⁴The set of elements to which the signature transformation is applied.

The key e_1 is used by the verifier to verify the primary signer's partial signature, and is published information. For the following discussions, we use the results of the following lemma proven in [22].

LEMMA 1. Let $N = pq$, where $p = 2p' + 1$, $q = 2q' + 1$, and p, q, p' , and q' are all prime numbers. We assume, without loss of generality, that $p < q$. Then,

1. The order of elements in \mathbb{Z}_N^* is one of the integers in the set $\{1, 2, p', q', 2p', 2q', p'q', 2p'q'\}$.
2. Given an element $a \in \mathbb{Z}_N^* \setminus \{-1, 1\}$ such that $\text{ord}(a) < p'q'$, then either $\text{gcd}(a - 1, N)$ or $\text{gcd}(a + 1, N)$ is a prime factor of N . ($\text{ord}(\cdot)$ denotes the order of a group element.)

As a consequence of Lemma 1, any element $a \in \mathbb{Z}_N^* \setminus \{-1, 1\}$ selected by a party that does not know the factorization of N satisfies $\text{ord}(a) \geq p'q'$ with overwhelming probability. We can directly use the results of Lemma 1 to gain useful insight into the characteristics of the elements of Q_N . By Lagrange's Theorem⁵ and (1), the order of elements in Q_N is one of the integers in the set $\{1, p', q', p'q'\}$. However, as a consequence of Lemma 1, any element $b \in Q_N$, $b \neq \pm 1$, such that $\text{gcd}(b + 1, N)$ and $\text{gcd}(b - 1, N)$ are not prime factors of N , satisfies

$$\text{ord}(b) = p'q'.$$

This fact will play a vital role in the registration stage of our fair-exchange protocol. The details of the key and multisignature generation processes are given below.

KEY GENERATION. Let KG denote the key generation algorithm for the multisignature scheme. The algorithm KG , on input of some security parameter k , first selects two safe primes p and q such that their product N satisfies $2^{k-1} \leq N < 2^k$. The joint public key is obtained by selecting a random integer e , $1 < e < \lambda$, such that $\text{gcd}(e, \lambda) = 1$. The joint private key is generated by finding the unique integer d , $1 < d < \lambda$, such that

$$ed \equiv 1 \pmod{\lambda}.$$

Now, the partial public key e_1 , $1 < e_1 < \lambda$, is chosen randomly such that $\text{gcd}(e_1, \lambda) = 1$. The corresponding (primary signer's) partial private key is generated by finding the unique integer d_1 , $1 < d_1 < \lambda$, such that

$$e_1 d_1 \equiv 1 \pmod{\lambda}.$$

Next, the (cosigner's) partial private key d_2 is computed as

$$d_2 = d - d_1 \pmod{\lambda}.$$

The keys satisfy the following relations:

$$\begin{aligned} ed &\equiv 1 \pmod{\lambda}, \\ e_1 d_1 &\equiv 1 \pmod{\lambda}, \\ d_1 + d_2 &\equiv d \pmod{\lambda}. \end{aligned}$$

⁵Lagrange's theorem states that if G is a finite group and H is a subgroup of G , then $|H|$ divides $|G|$. Thus, if $a \in G$, then $\text{ord}(a)$ divides $|G|$.

SIGNATURE GENERATION. The primary signer and co-signer create their respective partial signatures,

$$\sigma_i = H(M)^{d_i} \bmod N, \quad i = 1, 2,$$

using their respective partial private keys d_1 and d_2 . These are multiplied modulo N to form the multisignature σ . That is,

$$\sigma = H(M)^{d_1+d_2} \bmod N.$$

The partial signature σ_1 is considered valid if and only if

$$\sigma_1^{e_1} \bmod N = H(M).$$

The multisignature σ is verified in the same way using the joint public key e instead of e_1 .

Remark 1. Recall that in Boyd's scheme, d was split multiplicatively, whereas in our scheme, we split d additively. It is necessary to take the latter approach to avoid compromising the security of our protocol. Specifically, if d is split multiplicatively, the cosigner is able to create multisignatures on his own without the help of the primary signer. The compromise in security is due to the fact that the cosigner can use the three keys available to him—partial private key d_2 , partial public key e_1 , and the joint public key e —to compute d_1 . In the Appendix, we give details on how this can occur. Note that, in the fair-exchange protocol of Boyd and Foo [13], an RSA-based convertible undeniable signature scheme, which splits d multiplicatively, is used. In this signature scheme, however, the partial public key e_1 does not exist. Thus, although d is split multiplicatively, it does not cause any security problems there.

4. THE FAIR-EXCHANGE PROTOCOL

In the following protocol, Alice is the customer (or primary signer), Charlie is the TTP (or cosigner), and Bob is the merchant (or verifier). Using the following optimistic protocol, Alice purchases digital goods from Bob, and Bob receives her electronic check in return. We assume that the public keys of the CA and Charlie are known to all parties involved in the transaction.

The central theme of the registration protocol is Alice proving to Charlie that the following congruences hold without revealing d_1 and λ (with rest of the keys known to Charlie):

$$e_1 d_1 \equiv 1 \pmod{\lambda}, \quad (2)$$

$$(d_1 + d_2)e \equiv 1 \pmod{\lambda}. \quad (3)$$

This is done by using a *reference message* ω and its corresponding *reference signature* Ω , and is described in the following steps. A similar approach is used to construct an undeniable signature scheme in [22].

REGISTRATION. The registration protocol needs to be performed only *once*, after which it can support any number of exchanges. Note that most optimistic fair-exchange protocols require a registration stage (e.g., [5, 13]). We assume that the registration protocol is performed via confidential and authenticated channels. In practice, such channels can be implemented by using message authentication codes (MAC) in conjunction with encryption schemes.

1. Alice first generates the parameters N , p , and q and the keys e , e_1 , d , d_1 , and d_2 . Alice then contacts the

CA to get the joint public key e certified. At this stage, Alice has to prove to the CA that N is a product of safe primes (without revealing p and q). This can be done using the zero-knowledge protocol of Camenisch and Michels [15]. After verifying the construction of N , the CA issues a signed certificate C_{CA} . This certificate certifies the joint public key e and the modulus N . We assume that the values of e and N can be extracted from C_{CA} .⁶

2. Alice sends C_{CA} , e_1 , and d_2 to Charlie. Note that d_2 is Charlie's partial private key, and e_1 is the partial public key associated with Alice's partial private key d_1 .
3. Charlie checks the validity of C_{CA} .
4. Charlie randomly chooses an integer $\bar{\omega} \in \mathbb{Z}_N^* \setminus \{-1, 1\}$, and checks that $\gcd(\bar{\omega}+1, N)$ and $\gcd(\bar{\omega}-1, N)$ are not prime factors of N . He then computes $\omega = \bar{\omega}^2 \bmod N$. Note that ω is a generator of Q_N (see Section 3.4). Charlie sends the reference message ω to Alice.
5. Alice computes the reference signature

$$\Omega = \omega^{d_1} \bmod N,$$

and sends this to Charlie.

6. Now, Alice proves to Charlie that Ω is a power of ω without revealing d_1 . This can be done using the zero-knowledge protocol of Chaum et al. [16]. See [16] for details.
7. Charlie checks that Ω is constructed correctly by verifying the following congruence relations:

$$\Omega^{e_1} \equiv \omega \pmod{N},$$

$$\Omega^e \cdot \omega^{d_2 e} \equiv \omega \pmod{N}.$$

8. If the verifications (of Steps 3, 6, and 7) are passed, Charlie accepts Alice's claim that the congruence relations of (2) and (3) hold. He then creates a voucher V_C by signing on e_1 . We assume that e_1 can be extracted from V_C . Charlie stores his partial private key d_2 , and sends V_C to Alice.

Remark 2. The certificate C_{CA} certifies e and N . It might also include descriptions of the group Q_N and the hash function $H: \{0, 1\}^* \rightarrow Q_N$.

Remark 3. After Step 6, Charlie can assume that $\Omega = \omega^\delta \bmod N$ for some integer δ (at this point, he cannot be sure that $\delta = d_1$). Because ω is a generator (i.e., $\text{ord}(\omega) = p'q'$), the congruences in Step 7 imply that $\delta e_1 \equiv 1 \pmod{\lambda}$ and $(\delta + d_2)e \equiv 1 \pmod{\lambda}$, respectively. Thus, $\delta = d_1$ must be true if the verifications of Steps 6 and 7 are passed.

⁶The plaintext of the certificate can be extracted from the certificate, either because the signature scheme (used for generating the certificate) is capable of message recovery, or because the plaintext is concatenated with the signature.

Remark 4. The voucher V_C is a signed statement from Charlie that assures the following: (i) e_1 is Alice's valid partial public key, and (ii) the algebraic relations between the keys have been verified, and, as a result, Charlie can generate a multisignature from the corresponding partial signature. The first statement is explicitly shown by the content of the voucher, and the second statement is implicitly assumed to be true—Charlie will not create the voucher without verifying the relations of the keys. Therefore, the voucher implicitly conveys the following important semantics: *Charlie can convert any signature on some arbitrary message M , which is verified using (e_1, N) , to a signature on M that is verified with (e, N) .*

Remark 5. If the number of users is large, it requires Charlie to securely store a correspondingly large number of partial private keys d_2 (one for each primary signer). This can be avoided by using the following technique: Charlie concatenates d_2 and Alice's unique identification, ID_A , to form $d_2||ID_A$, and then encrypts this value via some symmetric-key encryption algorithm $E_\psi(\cdot)$, where ψ denotes the secret key. Charlie then creates a signature of the concatenated value of e_1 and $E_\psi(d_2||ID_A)$. That is,

$$Sig_C(e_1||E_\psi(d_2||ID_A)),$$

where $Sig_C(\cdot)$ denotes Charlie's signature algorithm. This value is used as the voucher V_C . Now, Charlie can extract d_2 from V_C (using ψ), and only needs to securely store ψ .

EXCHANGE. Alice initiates the protocol with Bob. We assume that Alice and Bob have gone through a negotiation process to agree on the purchase information M (which might contain Alice's unique identity, Bob's unique account number, price of the merchandise, description of the merchandise, and date of transaction) prior to the start of the exchange protocol. This process may be as simple as Alice choosing fixed-priced goods from Bob's website. Note that Alice's digital signature on M (which is her multisignature σ) acts as her digital check. In addition, Alice and Bob agree on a session key using some key-agreement protocol (e.g., Diffie-Hellman key agreement). The session key is used to encrypt the digital merchandise to deter eavesdropping. We use the basic optimistic protocol of Section 1 as the protocol framework. The following steps describe the exchange protocol.

1. Alice computes her partial signature σ_1 (using d_1), and sends Bob C_{CA} , V_C , and σ_1 .
2. Bob, using C_{CA} , verifies N and e . He then checks the validity of V_C , and verifies whether

$$\sigma_1^{e_1} \bmod N = H(M).$$

If everything is in order, Bob encrypts the digital merchandise μ with some symmetric encryption algorithm $E_\gamma(\cdot)$, where γ is the secret encryption key (i.e., the session key). The encrypted merchandise $E_\gamma(\mu)$ is sent to Alice. However, if any one of the items received from Alice is invalid, Bob does not send the merchandise, and stops the protocol.

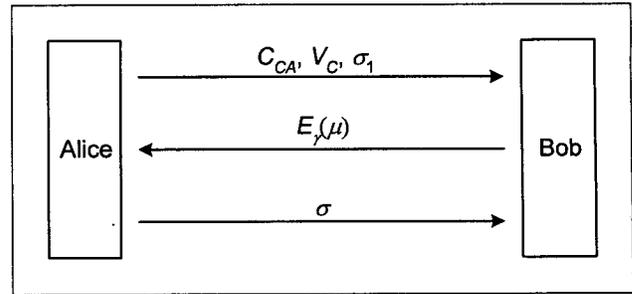


Figure 1: The exchange protocol.

3. Alice decrypts and verifies the merchandise. If Alice is satisfied with the merchandise, she computes the multisignature σ , and sends it to Bob. Otherwise, Alice stops the protocol.
4. Bob verifies σ , and if it is valid, ends the protocol. Otherwise, Bob initiates the dispute resolution protocol.

Figure 1 shows the messages exchanged between Alice and Bob in the exchange protocol when both parties act honestly.

Remark 6. Note that σ_1 and V_C constitute the fairness primitive.

Remark 7. The exchange protocol above requires the use of *timestamps* and *reliable channels*⁷ (see [2]) to ensure timely termination. Note that the protocol framework of Asokan et al. [3, 4] does not require timestamps for timely termination, and only requires *resilient channels*⁸. We can also apply our approach to their framework. Doing so is straightforward—replace their multistep verifiable escrow procedure with the transmission of a partial signature and its associated voucher (i.e., the fairness primitive). Of course, the primary signer and the TTP would have to go through a one-time registration protocol a priori to the exchange protocol.

Remark 8. The above exchange protocol does not require confidential and authenticated channels. Observe that Alice's digital check (i.e., σ) is of no value to an eavesdropper because it specifies the intended recipient of the check. Moreover, the digital merchandise is encrypted before transmission, and hence it is useless to an eavesdropper.

DISPUTE RESOLUTION. If Bob does not receive the multisignature σ , or if σ is invalid, he initiates a dispute resolution protocol by contacting Charlie. We assume that reliable channels exist between the parties.

⁷A channel that is always operational with a known upper bound of the time delay. An attacker cannot delay any messages beyond the known upper bound.

⁸A channel that is normally operational, but an attacker can succeed in delaying messages by an arbitrary but finite amount of time.

1. Bob encrypts the session key γ as $AE_{pk_C}(\gamma)$, where pk_C is Charlie's public key, and $AE_{pk_C}(\cdot)$ is an asymmetric encryption algorithm. Bob then sends C_{CA} , V_C , σ_1 , M , $E_\gamma(\mu)$, and $AE_{pk_C}(\gamma)$ to Charlie.
2. Charlie decrypts $AE_{pk_C}(\gamma)$, and uses γ to recover μ . Next, he extracts all the system parameters and keys from C_{CA} and V_C , and then verifies σ_1 using those values. If everything is in order, Charlie generates the multisignature σ using σ_1 and his partial private key d_2 via the relation

$$\sigma = \sigma_1 \cdot H(M)^{d_2} \bmod N.$$

The multisignature is sent to Bob, and the (encrypted) merchandise is forwarded to Alice. If any of the items received from Bob is invalid, Charlie halts the dispute resolution protocol without sending anything to either party.

5. EFFICIENCY EVALUATIONS

The most computationally expensive part of an optimistic exchange protocol is creating and verifying the fairness primitive, that is, the "commitment" c_A and its associated "voucher" V (see Section 1). Moreover, modular exponentiation is the most costly operation required to create and verify those items. Recall that in the exchange protocol of Section 4, σ_1 corresponds to a commitment, and V_C corresponds to a voucher.

In this section, we evaluate the efficiency of our scheme in terms of two criteria: (i) number of modular exponentiations required for creating/verifying the fairness primitive (i.e., σ_1 and V_C) in the exchange protocol and (ii) size of the fairness primitive (in bytes). In Table 1, we compare our scheme with the verifiable-escrow scheme of Asokan et al. [4] and the verifiable-encryption scheme of Ateniese [5]. For the comparison, we make the same assumptions made in [5]. Specifically, we exclude the overhead related to the CA's certificate, and assume a 1200-bit RSA modulus N and a 128-bit hash function. Specific to our fair-exchange protocol, we also assume that Charlie creates V_C using a signature scheme with the message recovery feature that requires one exponentiation modulo a 1280-bit number for verification (e.g., RSA).

Note that the numbers of Table 1 are approximate figures—the numbers can vary depending on the implementation. The numbers for the verifiable escrow and verifiable encryption schemes were taken from [5].

6. CONCLUSIONS

We presented a novel method for constructing efficient optimistic fair-exchange protocols using RSA-based multisignatures. We used a novel multisignature paradigm, which is quite different from the conventional model. The intrinsic features of the new paradigm enabled us to construct an exchange protocol that is very simple and efficient. Unlike the vast majority of previously proposed protocols, our approach does not use any zero-knowledge proofs in the exchange protocol, and thus avoids most of the costly computations. Use of zero-knowledge proofs is needed only in the registration phase—this is a one-time cost. As seen from Table 1, the resulting protocol is extremely efficient—in fact, it

Table 1: Comparison of efficiency

criterion	verifiable	verifiable	multi-
	escrow	encryption	signature
number of exponentiations	75	7.5	3
fairness primitive size	8000	400	310

requires even less computation and space overhead than the verifiable-encryption scheme [5], which is one of the most efficient protocols. Our approach has other advantageous features: (i) unlike protocols using convertible undeniable signatures, our scheme uses multisignatures that are compatible with the underlying (single-signer) signature, which implies that implementing the fair-exchange feature on top of an existing e-commerce system is less complicated, (ii) our approach can be applied to constructing fair-exchange protocols via multisignatures based on signature algorithms other than RSA, and (iii) our technique is flexible enough so that it can be used with the protocol framework of Asokan et al. [3, 4] as well as the framework of Bao et al. [8].

7. ACKNOWLEDGEMENTS

This research was supported in part by the Colorado State University George T. Abell Endowment, and by NSF under grants 0098089-ECS, 0099137-ANI, and ANI-0207892. The authors would like to thank the anonymous PODC reviewers for their insightful comments that improved the presentation of the paper.

8. ADDITIONAL AUTHORS

Additional authors: Indrajit Ray, Department of Computer Science, Colorado State University, 601 S Howes Street, Fort Collins, CO 80523, email: indrajit@colostate.edu.

9. 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 *International World Wide Web Conference Proceedings*, pages 387–395, May 1991.
- [2] N. Asokan. *Fairness in Electronic Commerce*. Department of Computer Science, University of Waterloo, 1998.
- [3] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In *Advances in Cryptology—EUROCRYPT '98*, pages 591–606, 1998.
- [4] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):593–610, 2000.
- [5] G. Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. In *Proceedings of ACM Conference on Computer and Communications Security*, pages 138–146, November 1999.
- [6] G. Ateniese and C. Nita-Rotaru. Stateless-recipient certified e-mail system based on verifiable encryption. In *Proceedings of RSA 2002*, February 2002.

- [7] A. Bahreman and J. D. Tygar. Certified electronic mail. In *Proceedings of Symposium on Network and Distributed Systems Security*, pages 3–19, February 1994.
- [8] 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*, pages 77–85, May 1998.
- [9] M. Bellare and R. Sandhu. *The security of practical two-party RSA signature schemes*. unpublished manuscript, 2001, available at <http://www.cs.ucsd.edu/users/mihir/papers/splitkey.html>.
- [10] M. Blum. How to exchange (secret) keys. *ACM Transactions on Computer Systems*, 1(2):175–193, 1983.
- [11] J. Boyar, D. Chaum, and I. Damgard. Convertible undeniable signatures. In *Advances in Cryptology—CRYPTO '90*, pages 189–205, 1990.
- [12] C. Boyd. Digital multisignatures. *Cryptography and Coding*, pages 241–246, 1989.
- [13] C. Boyd and E. Foo. Off-line fair payment protocols using convertible signatures. In *Advances in Cryptology—ASIACRYPT '98*, 1998.
- [14] J. Camenisch and I. Damgard. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *Advances in Cryptology—ASIACRYPT '00*, pages 331–345, 2000.
- [15] J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *Advances in Cryptology—EUROCRYPT '99*, pages 106–121, 2000.
- [16] D. Chaum, J. H. Evertse, and J. van der Graaf. An improved protocol for demonstrating possession of a discrete logarithm and some generalizations. In *Advances in Cryptology—EUROCRYPT '87*, pages 127–141, 1987.
- [17] L. Chen. Efficient fair exchange with verifiable confirmation of signatures. In *Advances in Cryptology—ASIACRYPT '98*, pages 286–299, 1998.
- [18] B. Cox, J. D. Tygar, and M. Sirbu. Netbill security and transaction protocol. In *Proceedings of 1st USENIX Workshop on Electronic Commerce*, pages 77–88, July 1995.
- [19] I. Damgard and T. Pedersen. New convertible undeniable signature schemes. In *Advances in Cryptology—EUROCRYPT '96*, pages 372–386, 1996.
- [20] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
- [21] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.
- [22] R. Gennaro, H. Krawczyk, and T. Rabin. RSA-based undeniable signatures. In *Advances in Cryptology—CRYPTO '97*, pages 132–149, 1997.
- [23] L. Harn. Group-oriented (t, n) threshold digital signature scheme and digital multisignature. *IEEE Proceedings—Computers and Digital Techniques*, 141(5):307–313, 1994.
- [24] N. Koblitz. *A Course in Number Theory and Cryptography*. Springer-Verlag, New York, New York, 1987.
- [25] P. MacKenzie and M. K. Reiter. Networked cryptographic devices resilient to capture. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 12–25, May 2001.
- [26] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, Florida, 1996.
- [27] S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures. In *Proceedings of ACM Conference on Computer and Communications Security*, pages 245–254, November 2001.
- [28] M. Michels and M. Stadler. Efficient convertible undeniable signature schemes. In *Proceedings of Annual Workshop on Selected Areas in Cryptography (SAC '97)*, pages 231–243, August 1997.
- [29] I. Ray and I. Ray. Fair exchange in e-commerce. *ACM SIGecom Exchange*, 3(2):9–17, May 2002.
- [30] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
- [31] J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 55–61, May 1996.

APPENDIX

A. INSECURITY OF SPLITTING THE PRIVATE KEY MULTIPLICATIVELY

We show that splitting the joint private key d multiplicatively renders our protocol insecure. Specifically, if d is split multiplicatively, the cosigner can create multisignatures on his own without the help of the primary signer. That is, the cosigner is able to use the three keys available to him—partial private key d_2 , partial public key e_1 , and the joint public key e —to compute d_1 . Although the cosigner is a trusted entity, it is important that he does not have the ability to forge multisignatures so that the multisignature's non-repudiation property is preserved. If we split d multiplicatively, then the keys satisfy the following relations:

$$d \equiv d_1 d_2 \pmod{\lambda}, \quad (4)$$

$$ed \equiv 1 \pmod{\lambda}, \quad (5)$$

$$e_1 d_1 \equiv 1 \pmod{\lambda}, \quad (6)$$

where $\lambda = p'q'$. Using (4), we can insert $d_1 \cdot d_2$ in place of d in (5) to obtain

$$ed_1d_2 \equiv 1 \pmod{\lambda}.$$

Now, multiplying both sides of the above congruence relation with e_1 and using (6), we obtain

$$ed_2 \equiv e_1 \pmod{\lambda}.$$

The above congruence relation implies that $ed_2 - e_1$ is a multiple of λ .

Recall that the cosigner is given the values d_2 , e_1 , and e by the primary signer. This means that the cosigner can readily calculate the value of $ed_2 - e_1$, which is a multiple of λ . With this knowledge, the cosigner can factor N efficiently using Koblitz's probabilistic algorithm (see p. 91 of [24]). Once N is factored, the cosigner can readily compute d_1 via the extended Euclidean algorithm, and thus compute the multisignature without the primary signer's help.