Optimistic Fair Exchange Based on Publicly Verifiable Secret Sharing

Gildas Avoine and Serge Vaudenay

EPFL

http://lasecwww.epfl.ch

Abstract. In this paper we propose an optimistic two-party fair exchange protocol which does not rely on a centralized trusted third party. Instead, the fairness of the protocol relies on the honesty of part of the neighbor participants. This new concept, which is based on a generic verifiable secret sharing scheme, is particularly relevant in networks where centralized authority can neither be used on-line nor off-line.

1 Introduction

A two-party fair exchange protocol is a protocol in which two participants, an originator P_o and a recipient P_r, wish to exchange items m_o and m_r in a fair way, i.e. such that no party can gain any advantage by cheating. There are two major kinds of two-party fair exchange protocols: those which rely on a Trusted Third Party (TTP) to ensure the fairness of the protocol, and those which do not. Even and Yacobi proved however in 1980 [9] that it is impossible to ensure a perfect fairness between only two participants without using a TTP. Protocols without TTP (e.g. [3,5,7]) are therefore only able to ensure fairness in a gradual or probabilistic way. The probability of fairness increases with the number of messages which are exchanged between the two participants. This implies a communication complexity cost which is too important for most of the practical applications. Finally, these protocols usually require that the involved parties have roughly equivalent computational powers, an assumption which is difficult to guarantee when dealing with heterogeneous networks. The second category of protocols rely on the use of a TTP to guarantee the fairness of the protocol. In all protocols the TTP acts as a central authority. TTP can be *on-line*, i.e. it is required in every exchange, or *off-line* (e.g. [1,2,11,15]). In this latter case, Asokan, Schunter, and Waidner invented the notion of *optimistic protocol* [1], where TTP is required only in case of conflict among participants. Fairness is ensured in a deterministic way and with few exchanges, but items must be either revocable or generatable by the TTP. Unfortunately, many environments, such as mobile ad hoc networks (MANET), do not allow either the use of a centralized authority (even in an optimistic case) for topologic reason, nor the usage of gradual protocols, which generate huge communication overheads. It is therefore important to design protocols based on other concepts. We propose a protocol which relies on the honesty of some "neighbor participants": when the exchange

H. Wang et al. (Eds.): ACISP 2004, LNCS 3108, pp. 74–85, 2004.

runs well, no communication overhead is required to any third party, but when a conflict occurs, neighbor participants are requested to restore fairness, by recovering the unreceived item. The recovery protocol relies on a publicly verifiable secret sharing scheme (PVSS). Indeed, a secret sharing scheme allows to share a secret among participants such that only certain subsets of participants can recover the secret. When the scheme is publicly verifiable, anybody is able to check whether the distributed shares are correct.
In what follows we define the protocol requirements, and the communication and security model. In Section 2, we recall briefly the verifiable secret sharing concept and describe such a practical protocol. We propose then our optimistic two-party fair exchange protocol based on a generic PVSS. A security analysis is finally provided in Section 4.

2 Communication and Security Model

2.1 Requirements

Several (different) definitions for the fair exchange are available in the literature; most of them are context-dependent. We use the following common definition.

Definition 1 (exchange protocol). *An exchange protocol between an originator P_o and a recipient P_r is a protocol in which P_o and P_r own some items m_o and m_r respectively and aim at exchanging them. We define the security properties as follows.*

1. **Completeness***: when there is no malicious misbehavior, P_o gets m_r and P_r gets m_o at the end of the protocol.*
2. **Fairness***: when at least one of the two participants follows the protocol, the exchange terminates so that either P_o gets m_r and P_r gets m_o (success termination), or P_o gets no information about m_r and P_r gets no information about m_o (failure termination).*
3. **Timeliness***: the exchange eventually terminates.*
4. **Privacy***: no other participant gets any information about m_o and m_r.*
5. **Optimism***: no other participant is involved when P_o and P_r are honest.*

We say that the protocol is a fair exchange protocol when it is complete, fair, and timely.

According to this definition, we design in this paper an optimistic fair exchange protocol between two parties, that do not require centralized TTP, implies reasonably low communication overhead, and protects privacy.

2.2 Threat Model

We say that an active participant is *honest* if he follows the protocol; otherwise he is *dishonest*. Note that due to the communication assumptions (below), all messages from honest participants are eventually delivered. The fairness of the protocol is ensured when either P_o or P_r is dishonest. Note that we do not

need to consider the case where both P_o and P_r are dishonest: in this case they obviously always have the ability to halt the exchange on an unfair termination. We now consider the passive participants' behaviors.

- $\mathcal{B}_1$: participants who honestly collaborate with both P_o and P_r.
- $\mathcal{B}_2$: participants who may harm P_o by colluding with P_r.
- $\mathcal{B}_3$: participants who may harm P_r by colluding with P_o.
- $\mathcal{B}_4$: participants who do not collaborate at all.

Note that $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$, and $\mathcal{B}_4$ form a partition of the passive participants $\mathcal{P}$. We denote $b_i = |\mathcal{B}_i|$ where $1 \leq i \leq 4$. We assume that participants cannot move from one set to another, focusing on the "honesty status" at the time of the recovery protocol only.

2.3 Communication Model and Hypothesis

Definition 2 (secure channel). *A channel is said to be secure if it ensures confidentiality, integrity, authentication, sequentiality, and timeliness.*
Confidentiality *ensures that the message is kept secret from any third party.* Integrity *ensures that the message cannot be modified by any third party.* Authentication *ensures that no third party can insert a forged message in the channel.* Sequentiality *ensures that the sequence of messages received by one party is equal to the sequence of messages sent by the other party in the same ordering at some time.* Timeliness *ensures that a message inserted into the channel is eventually delivered.*

Remark 1. Sequentiality ensures that no messages are swapped, dropped, or replayed.

Definition 3 (environment). *We say that two entities P and P' are in the same environment if and only if P and P' are able to communicate through a secure channel. We let Env_P denote the set of all the entities which are in the same environment as P*

Remark 2. The relation $P' \in Env_P$ between P and P' is symmetric but not transitive due to the timeliness requirement.

Definition 4 (participant). *We say that an entity which is involved in the protocol, either during the exchange stage or the recovery stage, is a participant. Participants which are only involved in the recovery stage are said* passive*; otherwise they are said* active*.*

Remark 3. In an optimistic *two-party* fair exchange, only participants P_o and P_r are active.

Hypothesis 1. In what follows we assume that $P_r \in Env_{P_o}$; P_r knows a subset of passive participants $\mathcal{P}$ of $Env_{P_o} \cap Env_{P_r}$ and a constant $T_{\max} < +\infty$ such that messages from P_r to any participant in $\mathcal{P}$ are always delivered within a time delay less than $T_{\max}$; $b_1 > 0$; and P_o and P_r know some constant k such that $b_2 < k \leq b_2 + b_1$.
We give here two examples in order to illustrate this assumption.

Example 1. If P_o and P_r know that there is a majority of honest participants in the network i.e. $b_1 > \frac{n}{2}$ then we take $k = \left\lceil \frac{n}{2} \right\rceil$.

Example 2. If P_o knows that at least 40% of the network is honest with him (i.e. $b_1 + b_3 \geq \frac{2n}{5}$) and P_r knows that at least 70% of the network is honest with him (i.e. $b_1 + b_2 \geq \frac{7n}{10}$) then we can take k such that $\left\lfloor \frac{6n}{10} \right\rfloor < k \leq \left\lceil \frac{7n}{10} \right\rceil$. For instance, if $n = 100$, k is chosen such that $60 < k \leq 70$. We show in Section 3 that k is actually the threshold of the secret sharing.

3 Optimistic Two-Party Fair Exchange Protocol

In this section, we first recall the notion of publicly verifiable secret-sharing; then we give an optimistic two-party fair exchange protocol. The main idea consists of sharing items among n participants such as k participants are enough to recover these items in case of conflict. The constraints on k will be analyzed in Section 4.

3.1 Publicly Verifiable Secret Sharing

Secret sharing [4,13] allows to share a secret m among several participants such that only some specific subsets of participants can recover m by collusion. In the Shamir secret sharing scheme, there is a threshold k so that only subsets of at least k participants can reconstruct m. A drawback of the Shamir scheme is that participants cannot verify that the distributed shares effectively allow to recover the secret m. In other words, the basic secret sharing scheme assumes that the dealer is not malicious. *Verifiable* secret sharing [6,10,12,14] resists to a malicious dealer who sends wrong shares: each participant can indeed check his own share. In *Publicly* verifiable secret sharing [12,14], introduced by Stadler in 1996, anybody can perform this verification and not only the participants. Below we describe a model for non-interactive publicly verifiable secret sharing (PVSS).

Distribution stage. The dealer generates the shares m_i of m and then publishes the encrypted values $E_i(m_i)$ such that only the participant P_i is able to decrypt $E_i(m_i)$. The dealer also publishes an information Δ containing $\theta = \mathcal{W}(m)$ where $\mathcal{W}$ is a one-way function. This information allows to prove that the distributed shares are correct i.e. they allow to recover some m such that $\mathcal{W}(m) = \theta$.

Verification stage. Given the P_is' public keys, the $E_i(m_i)$s, Δ, and a verification algorithm, anybody can verify that the shares allow to recover some m such that $\mathcal{W}(m) = \theta$.

Reconstruction stage. The participants decrypt their share m_i from $E_i(m_i)$ and pool them in order to recover m.

3.2 A Practical Publicly Verifiable Secret Sharing Scheme

We describe in this section a practical publicly verifiable secret sharing scheme which has been proposed by Stadler in [14]. This scheme relies on both ElGamal's public key cryptosystem [8] and on the double discrete logarithms assumption [14]. Let p be a large prime number so that $q = (p-1)/2$ is also prime, and let $h \in (\mathbb{Z}/p\mathbb{Z})^*$ be an element of order q. Let G be a group of order p, and let g be a generator of G such that computing discrete logarithms to the base g is difficult. Let $m \in \mathbb{Z}/p\mathbb{Z}$ be the secret and let $\mathcal{W}(m) = g^m$. As in Shamir's scheme, we assume that a publicly known element $x_i \in \mathbb{Z}/p\mathbb{Z}$, $x_i \neq 0$, is assigned to each participant P_i. We assume also that each participant P_i owns a secret key $z_i \in \mathbb{Z}/q\mathbb{Z}$ and the corresponding public key $y_i = h^{z_i} \bmod p$.

Distribution stage. The dealer chooses random elements $a_j \in \mathbb{Z}/p\mathbb{Z}$ $(j = 1, ..., k-1)$ and publishes the values $A_j = g^{a_j}$ $(j = 1, ..., k-1)$ in Δ. Then he securely computes the share

$$m_i = m + \sum_{j=1}^{k-1} a_j x_i^j \mod p \tag{1}$$

for P_i and he publishes the value g^{m_i} in Δ $(1 \leq i \leq n)$. He uses the ElGamal encryption: he chooses a random value $\alpha_i \in \mathbb{Z}/q\mathbb{Z}$, computes the pair

$$E_i(m_i) = (\sigma_i^1, \sigma_i^2) = (h^{\alpha_i}, m_i^{-1}.\, y_i^{\alpha_i}) \mod p$$

and publishes it in Δ $(1 \leq i \leq n)$. The precise content of Δ is described below.

Verification stage. The first step of this procedure consists in verifying the consistency of the shares. Anybody is able to perform this step by checking whether $g^{m_i} = g^m . \prod_{j=1}^{k-1} A_j^{x_i^j}$, obtained by exponentiating (1), is satisfied in G (Note that Δ includes g^m, g^{m_i}, A_j).

The second step consists in verifying that the pairs (σ_i^1, σ_i^2) really encrypt the discrete logarithms of public elements g^{m_i}. This verification is based on the fact that the discrete logarithm of $\sigma_i^1 = h^{\alpha_i}$ to the base h equals the double discrete logarithm of $g^{m_i \sigma_i^2} = g^{(y_i^{\alpha_i})}$ to the bases g and y_i. One may give a zero-knowledge interactive verification procedure between the dealer and participants as described on Fig. 1; we describe here the non-interactive version which is obtained by simulating the verifier by a hash function. We assume that the dealer randomly picked some values $w_{i,\ell} \in \mathbb{Z}/q\mathbb{Z}$ $(1 \leq \ell \leq L$ where $L \approx 100$ from [14]) for each share m_i and computed:

$$\begin{aligned} \delta_{i,\ell} &:= h^{w_{i,\ell}} \bmod p \\ \gamma_{i,\ell} &:= g^{y_i^{w_{i,\ell}}} \\ r_{i,\ell} &:= w_{i,\ell} - c_{i,\ell}\alpha_i \bmod q \end{aligned}$$

where $c_{i,\ell}$ denotes the ℓ-th bit of $c_i = \mathcal{H}(g^{m_i} \| \sigma_i^1 \| \sigma_i^2 \| \delta_{i,1} \| \gamma_{i,1} \| ... \| \delta_{i,L} \| \gamma_{i,L})$ with $\mathcal{H}$ a hash function from $\{0,1\}^*$ to $\{0,1\}^L$. Participants have therefore to check whether $\delta_{i,\ell} = h^{r_{i,\ell}} {\sigma_i^1}^{c_{i,\ell}} \mod p$ and $\gamma_{i,\ell} = (g^{1-c_{i,\ell}} g^{m_i c_{i,\ell} \sigma_i^2})^{y_i^{r_{i,\ell}}}$ for all ℓ.

Δ finally contains g^m, the g^{m_i}s, the $r_{i,\ell}$s, the $\delta_{i,\ell}$s, the $\gamma_{i,\ell}$s, and A_j.

Reconstruction stage. Each participant P_i decrypts his own share m_i by computing

$$m_i = \frac{(\sigma_i^1)^{z_i}}{\sigma_i^2} \mod p.$$

A subset of k participants can then recover m by using the Lagrange's interpolation formula.

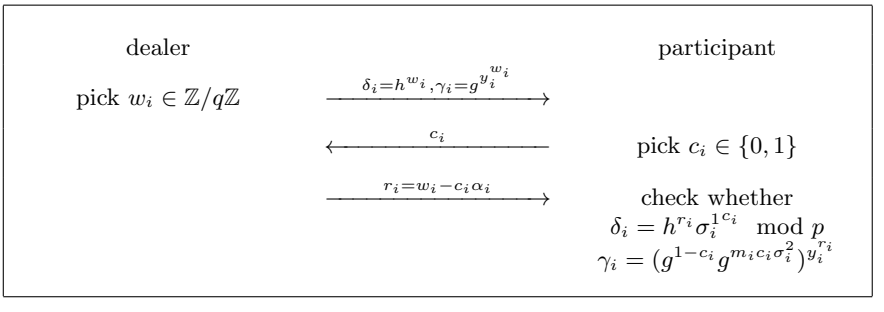


Fig. 1. Interactive verification procedure

3.3 Primitives

We define in this section some primitives which will be used in our protocol. These primitives are not related to a particular PVSS.

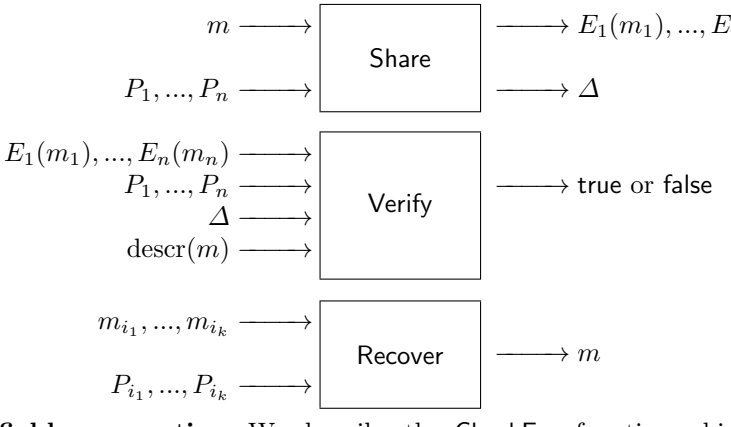
Signature. We consider P_o's authentication function S_o which, given a message m, outputs the signed version $m' = S_o(m)$ and the corresponding verification function V_o. Note that S_o is either a signature with message recovery or the concatenation of the message with the signature.

Item description. As is typically assumed in the literature, we suppose that P_o and P_r have committed to their items beforehand. We consider therefore that P_o and P_r established a legal agreement linking the authentic human-readable descriptions of the items with mathematical descriptions of these items $\text{descr}(m_o)$ and $\text{descr}(m_r)$. For instance, $\text{descr}(m) = \mathcal{W}(m) = g^m$. According to the fact that the authentic descriptions match the mathematical descriptions (a conflict at this layer can only be resolved by legal means), participants will be satisfied if they receive an item m which is consistent with its description $\text{descr}(m)$. To check

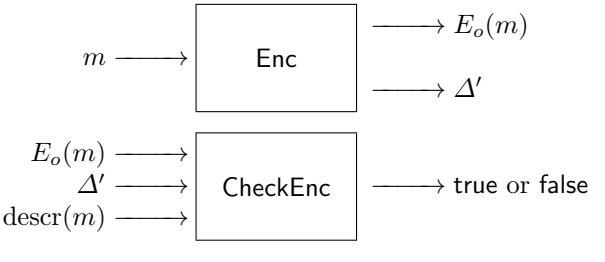
that, we consider the public contract $\Omega = S_o(P_o \| P_r \| \text{descr}(m_o) \| \text{descr}(m_r) \| T)$, where T is the expiration date after what the exchange has to be considered as null and void.

Encryption. We consider P_i's encryption function E_i which, given a message m, outputs the encrypted message $m' = E_i(m)$ for P_i and the corresponding decryption function D_i such that, given an encrypted message m', outputs the plain message $m = D_i(m')$.

Verifiable secret sharing. We consider a publicly verifiable secret sharing scheme using the functions Share, Verify, and Recover. Given a message m and some participants P_i $(1 \leq i \leq n)$, Share outputs the encrypted shares $E_1(m_1)$, ..., $E_n(m_n)$, and the proof Δ, as described in Section 3.2; given a list of encrypted shares, a list of participants, Δ, and descr(m), Verify outputs true if the shares allow any subset of k participants to recover m and false otherwise; given some shares m_{i_1}, ..., m_{i_k} and participants P_i $(i \in \{i_1, ..., i_k\})$, Recover outputs the message m.



Verifiable encryption. We describe the CheckEnc function which is pretty similar to the Check function except that its input is $E_o(m)$ rather than m. This function is used by the *passive* participants. We will not detail this function for the sake of simplicity but it could come from a PVSS with a single participant. We only sketch the primitives related to CheckEnc:



3.4 Description of the Protocols

Two participants P_o and P_r wish to exchange items m_o and m_r in a set of participants $\mathcal{P}$ such that $\mathcal{P} \subset Env_{P_o} \cap Env_{P_r}$ with $|\mathcal{P}| = n$. Note that for the sake of simplicity, we exclude P_o and P_r from $\mathcal{P}$, although it is not mandatory; so $\mathcal{P}$ contains only passive participants. Here are the exchange protocol and the recovery protocol.

Exchange protocol. The exchange protocol (Fig. 2) implies only the active participants, P_o and P_r, and consists in exchanging items m_o and m_r after a commitment step. This commitment gives to P_r the ability to restore fairness of the exchange, helped by the passive participants, in case of conflict with P_o.

- *Step 1:* P_o picks a random element a and computes b such that $m_o = a + b$. He computes $\mathsf{Share}(a, P_1, ..., P_n)$ and sends $E_1(a_1), ..., E_n(a_n), \Delta, \Omega, b$ to P_r.
- *Step 2:* P_r checks that $\mathsf{Verify}(E_1(a_1), ..., E_n(a_n), P_1, ..., P_n, \Delta, \mathrm{descr}(a))$ is true where $\mathrm{descr}(a)$ is deduced from $\mathrm{descr}(m_o)$ (extracted from Ω) and b, e.g. $g^a = g^m \times g^{-b}$; if the test succeeds then he sends m_r to P_o; otherwise he has just to wait until the expiration date T to give up the exchange.
- *Step 3:* P_o checks that m_r is correct running $\mathsf{Check}(m_r, \mathrm{descr}(m_r))$. If it is the case then P_o sends m_o to P_r. Otherwise, he has just to wait until the expiration date T in order to give up the exchange.
- *Step 4:* If P_r does not receive m_o or if $\mathsf{Check}(m_o, \mathrm{descr}(m_o))$ is false then he runs the recovery protocol.

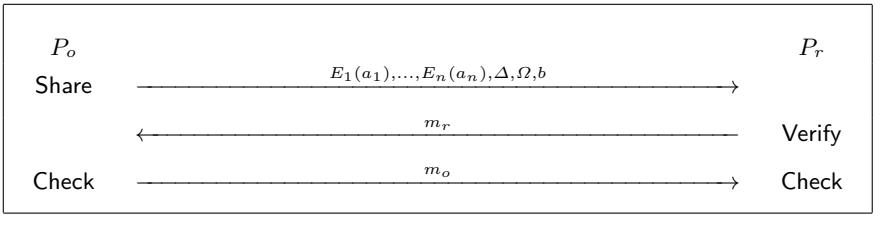


Fig. 2. Exchange protocol

Recovery protocol. The recovery protocol (Fig. 3) is started before $T - T_{\max}$ by the recipient, P_r, when he is injured, that is if the third message of the exchange, m_o, is wrong or missing.

- *Step 1:* P_r encrypts m_r for P_o and sends $E_i(a_i)$, $E_o(m_r)$, Δ', and Ω to P_i.

– *Step 2:* P_i computes $\mathsf{CheckEnc}(E_o(m_r), \mathrm{descr}(m_r), \Delta')$ where $\mathrm{descr}(m_r)$ is extracted from Ω; if the output is true and if the expiration date, contained in Ω, has not expired, P_i sends a_i to P_r and $E_o(m_r)$ to P_o.

– *Step 3:* after having received k shares, P_r runs $\mathsf{Recover}$. From a he computes $m_o = a + b$.

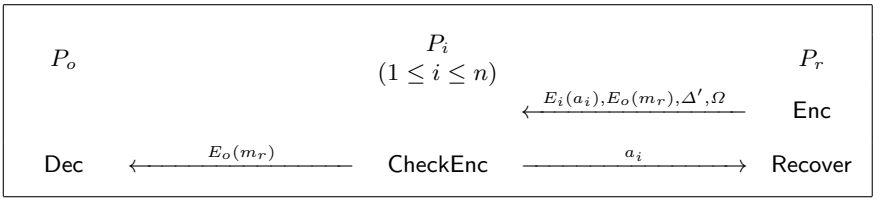


Fig. 3. Recovery protocol

Remark 4. In optimistic fair exchange protocols using TTP (e.g. [11]), the TTP is stateful: the TTP keeps in mind whether the recovery or abort protocol has already been performed. Due to our distributed architecture we cannot use this model here and we prefer using expiration dates.

4 Security Analysis

We prove in this section that our protocol is complete, fair, timely and respects the privacy property even in case of misbehaviors. We recall that the security parameter k, which is the threshold of the PVSS, is such that $b_2 < k \le b_2 + b_1$. We defined in Section 2.2 the set of passive participants $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$, and $\mathcal{B}_4$. We rewrite these definitions here according to our protocol defined in Section 3.4: $\mathcal{B}_1$: participants who honestly collaborate with both P_o and P_r; $\mathcal{B}_2$: participants P_i such that when P_r sends $E_i(a_i)$ to $P_i \in \mathcal{B}_2$, P_i decrypts $E_i(a_i)$ and sends a_i to P_r (even if the date is expired) but does not send m_r to P_o; $\mathcal{B}_3$: participants P_i such that when P_r sends $E_i(a_i)$ to $P_i \in \mathcal{B}_3$, P_i sends $E_o(m_r)$ to P_o (even if the date is expired) but does not decrypt $E_i(a_i)$ for P_r; $\mathcal{B}_4$: participants who do not collaborate at all. We denote M_1, M_2, and M_3 the three messages of the exchange protocol consisting respectively of $[E_1(a_1), ..., E_n(a_n), \Delta, \Omega, b]$, $[m_r]$, and $[m_o]$.

4.1 Completeness of the Protocol

Proving that the protocol is complete when P_o and P_r are honest is straightforward. In case of late discovery of M_3 due to communication protocol reasons, P_r runs the recovery protocol. Since $b_2 < k \le b_2 + b_1$ we have at least k participants who will collaborate with P_r and at least one who will collaborate with P_o.

4.2 Fairness of the Protocol

We saw in the previous section that the protocol is fair if both active participants are honest. We explained furthermore in Section 2.2 that the case where both active participants are dishonest is not relevant.

P_o is honest and P_r is dishonest. Since P_o is honest, M_1 is correctly formed. On the other hand M_2 is wrong (or missing) otherwise both P_o and P_r would be honest. Here P_o can detect that M_2 is wrong using $\mathsf{Check}(m_r,\mathrm{descr}(m_r))$; therefore he does not transmit M_3 and waits for T. If P_r does not run the recovery protocol then nobody can obtain anything valuable on the expected items and the exchange is trivially fair. If P_r starts the recovery protocol after T then he cannot obtain m_o since $b_2 < k$. If P_r starts the recovery protocol before T (note that if he only contacts participants in $\mathcal{B}_2$ or $\mathcal{B}_4$ before T then we fall into the previous case) then P_o receives m_r from a passive participant either in $\mathcal{B}_1$ or $\mathcal{B}_3$; therefore the protocol is fair iff P_r can obtain m_o that is if and only if $b_1 + b_2 \geq k$.

P_o is dishonest and P_r is honest. Since P_o is dishonest, we consider the case where M_1 is wrong (or missing) and the case where M_1 is correct but M_3 is not (or missing). If M_1 is wrong (or missing), the exchange will die after T; indeed P_r can perform $\mathsf{Verify}(E_1(a_1), ..., E_n(a_n), P_1, ..., P_n, \Delta, \mathrm{descr}(a))$ and detects so that M_1 is wrong; he decides therefore not to disclose m_r. The exchange ends therefore on a trivially fair termination after T. Secondly, if M_1 is correct but M_3 is not (or missing): P_r can detect such a wrong M_3 using $\mathsf{Check}(m_o,\mathrm{descr}(m_0))$ and therefore start the recovery protocol. The fairness of the exchange relies thus on the ability of the passive participant to supply a to P_r, that is if and only if $b_1 + b_2 \geq k$. The fairness is so ensured since P_o has already received m_r in M_2.

4.3 Timeliness of the Protocol

Timeliness of the protocol is straightforward.

4.4 Privacy of the Protocol

Privacy of the protocol is straightforward. If the recovery protocol is not performed, then only information between P_o and P_r are exchanged and passive participants receive nothing. If the recovery protocol is used, then some participants receive shares of a_i. However, although k participants colluding can recover a, they cannot recover m_o since they do not know b. Obviously, they cannot discover m_r either. Privacy is here a great improvement with regard to previous optimistic fair exchange protocol where we usually assume that the trusted third party is able to regenerate expected items.

4.5 Complexity of the Protocol

When both P_o and P_r are honest, the complexity in terms of exchanged messages is very small since only the three messages of the exchange protocol are

sent. When somebody misbehaves, the complexity obviously increases since the recovery procedure is performed. In the worst case, the n passive participants are contacted by P_r, each receives one message and sends at most two messages, so the complexity is only $O(3n)$ in terms of exchanged messages.

5 Conclusion

We proposed an optimistic two-party fair exchange protocol using publicly verifiable secret sharing. Our protocol is the first optimistic fair exchange protocol which does not rely on a centralized trusted third party. This concept is therefore particularly suitable for ad-hoc networks. We proved that our protocol ensures fairness and privacy even in quite dishonest environment and implies only low communication overheads. Our protocol works assuming that a majority of participants are honest or that only one is honest but we can estimate the number b_2 of participants who may harm P_o by colluding with P_r.

Acknowledgment. The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

References

1. N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *Proceedings of 4th ACM Conference on Computer and Communications Security*, pages 7–17, Zurich, Switzerland, April 1997. ACM Press.
2. N. Asokan, Victor Shoup, and Michael Waidner. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 86–99, Oakland, California, USA, May 1998. IEEE Computer Society Press.
3. Gildas Avoine and Serge Vaudenay. Fair exchange with guardian angels. In Kijoon Chae and Moti Yung, editors, *The 4th International Workshop on Information Security Applications – WISA 2003*, volume 2908 of *Lecture Notes in Computer Science*, pages 188–202, Jeju Island, Korea, August 2003. Springer-Verlag.
4. George Robert Blakley. Safeguarding cryptographic keys. In *National Computer Conference*, volume 48 of *American Federation of Information*, pages 313–317, 1979.
5. Ernest F. Brickell, David Chaum, Ivan B. Damgård, and Jeroen van de Graaf. Gradual and verifiable release of a secret. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 156–166, Santa Barbara, CA, USA, August 1988. IACR, Springer-Verlag.
6. Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of the 26th IEEE Annual Symposium on Foundations of Computer Science – FOCS'85*, pages 383– 395, Portland, OR, USA, October 1985. IEEE.

7. Richard Cleve. Controlled gradual disclosure schemes for random bits and their applications. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 573–588, Santa Barbara, California, USA, August 1990. IACR, Springer-Verlag.
8. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, July 1985.
9. Shimon Even and Yacov Yacobi. Relations amoung public key signature systems. Technical Report 175, Computer Science Department, Technicon, Haifa, Israel, 1980.
10. Eiichiro Fujisaki and Tatsuaki Okamoto. A practical and provably secure scheme for publicly veriable secret sharing and its applications. In Nyberg Kaisa, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 32–46, Helsinki, Finland, May–June 1998. IACR, Springer-Verlag.
11. Olivier Markowitch and Shahrokh Saeednia. Optimistic fair exchange with transparent signature recovery. In *Financial Cryptography – FC'01*, Lecture Notes in Computer Science, Cayman Islands, February 2001. IFCA, Springer-Verlag.
12. Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In Michael Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 148–164, Santa Barbara, California, USA, August 1999. IACR, Springer-Verlag.
13. Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.
14. Markus Stadler. Publicly verifiable secret sharing. In Ueli Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 190–199, Saragossa, Spain, May 1996. IACR, Springer-Verlag.
15. Holger Vogt. Asynchronous optimistic fair exchange based on revocable items. In Rebecca N. Wright, editor, *Financial Cryptography – FC'03*, volume 2742 of *Lecture Notes in Computer Science*, Le Gosier, Guadeloupe, French West Indies, January 2003. IFCA, Springer-Verlag.