Secure Group Barter: Multi-Party Fair Exchange with Semi-Trusted Neutral Parties

Matt Franklin
AT&T Labs Research
Florham Park, NJ 07932
franklin@research.att.com

Gene Tsudik*
USC Information Sciences Institute
Marina del Rey, CA 90292
gts@isi.edu

February 6, 1998

Abstract

The recent surge in popularity of e-commerce prompted a lot of activity in the area of electronic payments. Solutions have been developed for cash, credit card and check-based electronic transactions. Much less attention has been paid to non-monetary commerce such as barter. In this paper we discuss the notion of "secure group barter" or multi-party fair exchange. We develop a classification of types of barter schemes and present new cryptographic protocols for multi-party exchange with fairness. These protocols assume the presence of a "semi-trusted neutral party".

1 Introduction

This paper is concerned with the barter of digital goods among groups of participants in the electronic world. The kind of barter we envision is an instantaneous, one-time, discrete trade arrangement by an *ad hoc* group of participants. A crucial issue for this kind of barter situation is "fairness". This is a kind of atomicity property for the exchange, whereby no participant gives anything away unless she gets everything she wants.

The problem of multi-party fair exchange has received some attention, but it has not been studied as widely as the more fundamental problem of 2-party fair exchange. In a 2-party fair exchange, each party holds a digital document. They want to engage in a protocol to swap the documents fairly, in the sense that neither party should gain an informational advantage by quitting early or otherwise misbehaving. In the next subsection, we consider a simple but useful taxonomy for 2-party fair exchange.

1.1 A Taxonomy of Protocols for Two-Party Fair Exchange

In a 2-party fair exchange, there is nothing to prevent a "malicious party" from behaving honestly throughout the protocol, while contributing a digital document which contains meaningless or valueless junk. To address this, it is typical to assume that each main party has committed to its document beforehand, e.g., by publishing a cryptographic checksum thereof. Then we assume that each main party is satisfied if it receives a digital document that is consistent with the checksum. In some sense, this commitment-before-exchange approach defers rather than solves the issue of document quality or value. This issue, which also arises in the certification of public keys, is outside the scope of this paper.

Protocols for 2-party fair exchange can be divided into two categories, on the basis of the possible use of an additional, neutral participant. A number of theoretically important protocols, beginning with [6, 14, 16], avoid the use of a neutral party altogether. A number of practically important protocols use a neutral party, with the

*Research supported by the Defense Advanced Research Project Agency, Information Technology Office (DARPA-ITO), under contract DABT63-97-C-0031.

requirement that the neutral party should not learn any useful information about the documents being exchanged by the two main parties.

Within the category of neutral party protocols, it useful to make the further distinction of "on-line" versus "off-line" participation by the neutral party. Some protocols, such as [2, 1], require the neutral party to participate actively only when there has been misbehavior by one of the two main parties ("off-line"). These off-line protocols are also called "optimistic", because they are most efficient in the (hoped-for) case when the main parties are honest. Other protocols, such as [15, 7, 11], require the neutral party to participate in every exchange ("on-line").

The class of on-line fair exchange protocols can be further subdivided into those offering "certified delivery" versus "expected delivery". A protocol achieves certified delivery [15] if each main party gets either the document it expects or proof that the other main party misbehaved. A protocol achieves expected delivery if each main party gets the document it expects. The protocols in [11] achieve expected delivery under the assumption that the neutral party does not collude with either of the main parties; they call this a "semi-trusted neutral party".

Given a choice of protocols that were otherwise equally efficient, we might prefer expected delivery to certified delivery, and prefer optimistic to on-line, and no neutral party whatsoever most of all. However, the best protocols in the literature for each category are not otherwise equally efficient. In fact, the on-line protocols with certified delivery require the least computation and communication, and the perfect protocols require the most. On-line protocols with expected delivery require a few more public key operations (e.g., modular multiplications) than for certified delivery. Optimistic protocols such as [1] require two to three orders of magnitude more public key operations than for on-line protocols. Perfect protocols perform the exchange one bit at a time, or even a fraction of a bit at a time, with a lot of cryptographic computations required by both parties to prove that they are behaving properly; these protocols are typically not not considered suitable for practical applications.

1.2 Contribution of Our Work

Work in multi-party fair exchange can be categorized as in the 2-party case. Asokan et al. [2] proposed an optimistic multi-party fair exchange protocol fashioned after an optimistic 2-party fair exchange protocol developed earlier in [3]. In the same vein, work is on-going in the context of the EC-sponsored project SEMPER to refine and extend the results in [3]. In addition, Ketchpel and Garcia-Molina [13] developed multi-party fair exchange protocols that are on-line and achieve certified delivery.

This paper is concerned with the case of multi-party fair exchange that is on-line and achieves expected delivery. We envision an environment where large groups of parties are involved in complex transactions and continuous surveillance by an on-line or an in-line neutral party is impossible or impractical. There may not even be a designated *full-time* neutral party to rely on. Finally, parties are mutually suspicious and cannot be assumed to have any prior association. This, coupled with the consideration that different parties can be subject to very different administrative, legal and political controls, makes this kind of an environment unsuitable for optimistic fair exchange.

All this leads us to extend the "semi-trusted neutral party" setting from 2-party fair exchange [11] to multi-party fair exchange. A semi-trusted neutral party can be selected on a case-by-case basis and asked to aid in the execution of a fair multi-party exchange. However, while an semi-trusted neutral party is trusted to ensure the *fairness* of the multi-party exchange, it is not trusted with the actual commodities involved in the exchange. This means that a malicious semi-trusted neutral party must be unable to cheat as long as the other parties remain honest.

The remainder of the paper is organized as follows. In Section 2 we review the previous work on 2-party fair exchange with a semi-trusted neutral party. A classification of multi-party exchanges is given in Section 3. In Section 4, our protocols for multi-party fair exchange are specified and analyzed. Additional security issues are considered in Section 5: concealment of the exchange topology, defense against passive conspiracies, and identification of barter opportunities. Section 6 gives some conclusions and open problems.

2 Two-Party Fair Exchange with a Semi-Trusted Neutral Party

In this section, we review the semi-trusted neutral party setting and the 2-party fair exchange protocol of [11]. We begin with some useful notation and acronyms.

FE	Fair Exchange
TNP	Trusted Neutral Party
STNP	Semi-Trusted Neutral Party
OWF	One-Way Function
P_i	Protocol participant (indexed)
K_i	Secret quantity held by P_i

The 2-party FE protocol with STNP works through 2-out-of-2 verifiable secret sharing [9, 10]. It can be built from any one-way function f with certain algebraic properties, based for example on the hardness of factoring, discrete log, or graph isomorphism. For convenience, we assume the example based on the hardness of factoring: $f(x) = x^2 \bmod N$, where N is the product of two large distinct primes. For notational convenience, we will omit $\bmod N$ whenever it is obvious from context.

Initially, party X holds a secret key K_X and party Y holds a secret key K_Y. Somehow, X and Y agree on (i.e., select) a semi-trusted neutral party Z that will aid them in the exchange. All three parties (X, Y and Z) know K_X^2 and K_Y^2. X chooses random x_1, x_2 such that $x_1 x_2 = K_X$. Y chooses random y_1, y_2 such that $y_1 y_2 = K_Y$. X sends x_1 to Y and Y sends y_1 to X. Then X sends x_2, y_1^2 to Z and Y sends y_2, x_1^2 to Z. This puts Z in a position to *verify* the consistency of the shares: Does $K_X^2 = x_1^2(x_2)^2$? And does $K_Y^2 = y_1^2(y_2)^2$? If so, then Z completes the exchange by sending y_2 to X and sending x_2 to Y.

At the end of the protocol, if all three parties are honest, X learns $K_Y = y_1 y_2$ and Y learns $K_X = x_1 x_2$. If X and Z are honest, then Y learns nothing useful about K_X unless X learns K_Y. If Y and Z are honest, then X learns nothing useful about K_Y unless Y learns K_X. If X and Y are honest, then Z learns nothing useful about K_X or K_Y. These are the properties needed for a fair exchange.

The protocol can be optimized so that there are normally only four flows: (1) from X to Y; (2) from Y to Z; (3) from Z to Y; and (4) from Y to X. It is also not necessary for Z to know K_X^2, K_Y^2 at the start. It is also possible to *blind* the protocol so that Z cannot recognize different exchanges of the same document (unlinkability).

In this paper, we will have n main parties in the exchange, together with a single semi-trusted "neutral party". The trust assumptions for this more general case will be discussed in a later section.

3 Classification of Multi-Party Exchanges

Let $\{K_1, \ldots, K_d\}$ be the types of commodities that will be involved in the exchanges, with a well-defined notion of unit for each commodity. We will express trades and trading preferences as d-dimensional vectors of integers. A positive integer $+m$ in the ith position indicates that m units of commodity K_i are received or desired. A negative integer $-m$ in the ith position indicates that m units of K_i are given or offered.

First, we define a single-unit cyclic exchange (see also Figure 1):

> $\boxed{\textit{Definition:}}$ *An n-party single-unit cyclic exchange is a cycle of length n where for every $i \leq n$, the corresponding party P_i, trades one unit of commodity K_i by offering it to $P_{(i+1)}$ for one unit of commodity $K_{(i-1)}$ offered by $P_{(i-1)}$. (Note: all indices are* mod *n.)*

The trade performed by party P_i in a single-unit cyclic exchange can be expressed as a d-dimensional vector, all of whose entries are zero except for -1 in the ith location, and +1 in the $i-1$st location. Next, we define a single-unit general exchange:

> $\boxed{\textit{Definition:}}$ *An n-party single-unit general exchange is a permutation σ on $\{1 \ldots n\}$, where each party P_i offers a single unit of commodity K_i to $P_{\sigma(i)}$, and receives a single unit of commodity $K_{\sigma^{-1}(i)}$ from $P_{\sigma^{-1}(i)}$.*

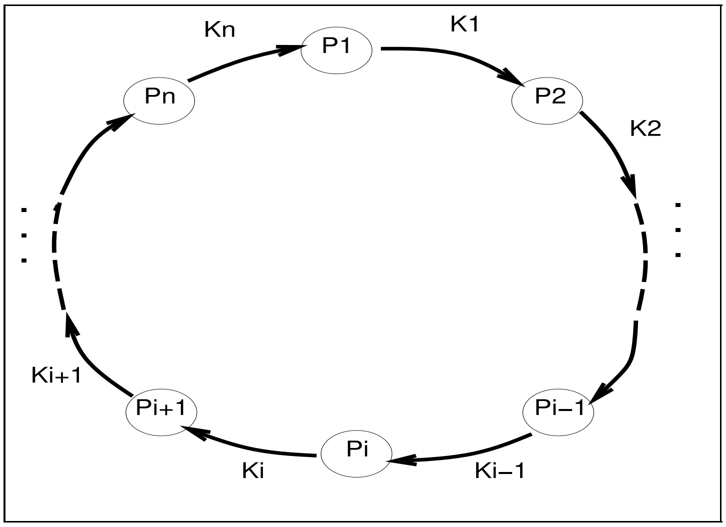

Figure 1: Single-Unit Cyclic Exchange

The trade performed by party P_i in a single-unit general exchange can be expressed as a d-dimensional vector, all of whose entries are zero except for -1 in the ith position, and +1 in the $\sigma^{-1}(i)$th location. Next, we define a multi-unit cyclic exchange and a multi-unit general exchange. A "basket" of commodities is a d-dimensional vector all of whose entries are nonnegative.

$\boxed{\textit{Definition:}}$ *An n-party multi-unit cyclic exchange is a cycle of length n where for every $i \leq n$, the corresponding party P_i, trades a basket of commodities B_i by offering it to $P_{(i+1)}$ for a basket of commodity $B_{(i-1)}$ offered by $P_{(i-1)}$.*

$\boxed{\textit{Definition:}}$ *An n-party multi-unit general exchange is a matrix of baskets, where the entry B_{ij} in row i and column j is the basket of goods given by P_i to P_j.*

The trade performed by party P_i in a multi-unit cyclic exchange is the vector $B_{i-1} - B_i$. The trade performed by party P_i in a multi-unit general exchange is the vector $(\sum_j B_{ji}) - (\sum_j B_{ij}))$.

Any permutation can be decomposed into disjoint cycles. Given a protocol for (single-unit, multi-unit) cyclic exchange, we can construct a protocol for (single-unit, multi-unit) general exchange by performing each disjoint cyclic exchange independently. For our model, the fairness properties of the cyclic exchange will carry over to the general exchange. In the remainder of this paper, we focus on the problem of fair single-unit cyclic exchange.

4 Fair Single-Unit Cyclic Exchange

In this section, we consider fair protocols for single-unit cyclic exchange. We assume that, initially, each party P_i is in sole possession of a secret quantity K_i. The ideal outcome of a group barter is the simultaneous exchange of secret quantities as depicted in Figure 1, i.e., each P_i trades its K_i to P_{i+1} in return for K_{i-1} from P_{i-1}. The desired properties of a group barter are essentially nothing more than the properties of 2-party FE extended to groups:

1. If all parties are honest, then, for $0 < i \leq n$, P_i obtains K_{i-1}.

2. If all P_i-s are honest, then Z (the STNP) cannot obtain any K_i.

3. Let $\mathcal{P} = P_1, \ldots, P_n$. $\forall\, \mathcal{S} = \mathcal{P} - P_j \quad (0 < j \leq n)$:
   if all parties in $\mathcal{S}$ and Z are honest, then:

   (a) P_j obtains K_{j-1} iff
       $\forall P_i \in \mathcal{S}$, P_i obtains P_{i-1}.

   (b) P_j cannot obtain any K_i for $i \neq j$ and $i \neq j-1$.

These properties collectively translate into 1-resilience, i.e., as long at most one of protocol parties is dishonest, it can gain no advantage over the rest. For the moment, we consider this to be sufficient.

Similar to the 2-party case described in [11], we require that all protocol messages be **private** and **authentic**. Also, as in [11], we select an OWF $f : G \to G$ where G is group in which membership testing, group operation and inverse computation are efficient.

We also require an n-variable function $F_n(X_1, \ldots, X_n)$ with a property that:

$$F_n(X_1, f(X_2), \ldots, f(X_n)) = f(X_1 X_2 \cdots X_n)$$

One example of a suitable $< F_n, f >$ function pair is:

- $F_n = (X_1^2) X_2 \cdots X_n$
   and
- $f = X^2 mod N$
  where N is a product of two large distinct primes.

Each party P_i holds a secret quantity K_i and knows all $f(K_j)$ $(0 < j \leq n)$.

4.1 Protocol for Fair Single-Unit Cyclic Exchange

Our protocol (called SUCEX-1) begins with each P_i generating at random a quantity $R_i \in_R Z_q$ and computing its inverse R_i^{-1}. Then, each P_i sends R_i in secret to P_{i+1}, the intended future recipient of K_i. (See also Figure 4.1.)

Before contacting STNP Z, each P_i also computes the following:

1. $A_i = F_n(K_i, f(K_1), \ldots, f(K_{i-1}), f(K_{i+1}), \ldots, f(K_n))$
2. $C_i = K_i \cdot R_i^{-1}$

Then, each P_i forwards to Z its $< A_i, C_i, f(R_i) >$. In turn, Z does the following:

1. Matches all A_i fields; if everything matches, proceed to next step. Otherwise output ERROR and halt.
2. Computes $C = C_1 \cdots C_n$ and then computes

   $$F_{n+1}(C, f(R_1), \ldots, f(R_n)) = f(K_1 R_1^{-1} \cdots K_n R_n^{-1} \cdot R_1 \cdots R_n) = f(K_1 \cdots K_n)$$

   and compares to A_i. (The choice of i is unimportant since all A_i's are the same.) The function F_{n+1} is an $(n+1)$-variable version of F_n.

The first step is basically a sanity check; it establishes that all K_i's and all R_i's are consistent and have been properly committed. The second step establishes the coherence of all K_i values, i.e., C_i actually proves P_i's possession of K_i and R_i^{-1}.

In the last protocol round, Z broadcasts to all parties the set $\mathcal{C} = \{C_j \mid 0 < j \leq n\}$. This enables each P_i to compute $K_{i-1} = R_{i-1} \cdot C_i = R_{i-1} \cdot K_{i-1} \cdot R_{i-1}^{-1}$.

P_i $\qquad\qquad\qquad$ P_{i+1}

$$\xrightarrow{R_i}$$

P_i $\qquad\qquad\qquad$ Z

$$A_i = F_n(K_i, f(K_1), \ldots, f(K_{i-1}), f(K_{i+1}), \ldots, f(K_n))$$
$$C_i = K_i \cdot \ R_i^{-1}$$

$$\xrightarrow{A_i, C_i, f(R_i)}$$

$$A_i \stackrel{?}{=} A_j$$
$$C = C_1 \cdots \ C_n$$
$$F_{n+1}(C, f(R_1), \ldots, f(R_n)) \stackrel{?}{=} A_i$$

$$\xleftarrow{\mathcal{C} = \{C_j \mid 0 < j \leq \ n\}}$$

$$\widehat{K_{i-1}} = C_{i-1} \cdot \ R_{i-1}$$
$$f(\widehat{K_{i-1}}) \stackrel{?}{=} f(K_{i-1})$$

Figure 2: SUCEX-1: Fair Cyclic Exchange with STNP

4.2 Analysis

Correctness: (sketch) we claim that protocol SUCEX-1 satisfies the requirements stated above:

1. If everyone is honest, then at the end of the protocol each P_i can compute:
   $K_{i-1} = R_{i-1} \cdot C_{i-1} = R_{i-1} \cdot K_i \cdot \ R_i^{-1}$
2. If all parties in $\mathcal{P}$ are honest (but Z is not) and, as required, all R_i values are pre-distributed securely, Z cannot obtain any K_i.
3. Suppose that exactly one (say, P_j) protocol participant is dishonest:
   (a) P_j has only $f(K_{j-1})$ and R_{j-1} in its possession unless the protocol terminates normally, i.e., it receives C_{j-1} in the last round. Then, P_j obtains K_{j-1}. However, the last round includes the atomic distribution of all C_i values; consequently, K_j is at the same time computed by P_{j+1}. (Since in this case Z is honest.)
   (b) P_j similarly learns nothing about any other K_i where $i \neq \ j-1$. This is because P_j's *view* of any such K_i is equivalent to that of Z; it only "sees" C_i in round 3 and never learns any R_i which is transmitted in secret to R_{i+1} in round 1.

4.3 Neutral Party Variations

There are a number of possible variations on the STNP scenario. The role of the STNP could be distributed among two or more neutral parties. Alternatively, the role of the STNP could be played by one or more of the n parties themselves. We have focused on the setting of a single STNP primarily for reasons of efficiency. The communication pattern is particularly simple. The applicability of modular squaring as the underlying cryptographic mechanism simplifies the computational effort by all participants as well.

5 Other Security Issues

In this section, we consider some other security issues associated with multi-party fair exchange: hiding the cycle topology, defending against passive conspiracy, and identifying barter opportunities.

5.1 Hiding the Cycle Topology

In a 2-party exchange, the *topology* of the exchange is obvious: P_1 trades K_1 for K_2 with P_2. This is evident to anyone, including, of course, an STNP. In the multi-party case, however, **the barter topology cannot be inferred from the indentities of the participants.** In fact, for a group of size n, there are $(n-1)!$ distinct barter topologies.

This observation leads us to consider the topology as an additional and unique property of group barter. Furthermore, the topology is something that barter participants may not want revealed to outside parties. We also note that an STNP is essentially an outside party; its task is to ensure the fairness of the exchange and it has no inherent "need-to-know" as far as the topology.

Claim: Although it has not been an explicit design goal, the SUCEX-1 protocol keeps barter topology secret from the STNP.

Proof: (sketch) To support this claim consider that the construction of the barter topology is assumed to have been completed amongst the participants ahead of time. Moreover, the first protocol round (distribution of R_i values) takes place before the STNP (Z) is contacted or perhaps even selected.

In the second round, Z is made aware of the participants' number and identities. The computations of both C and F_{n+1} requires Z to assemble all indexed C_i and $f(R_i)$ values. However, the order of assembly is unimportant due to commutativity of multiplication. Finally, in round 3, Z needs to broadcast a set $\mathcal{C}$ to all participants. Once again, the ordering within the set can be arbitrary since each P_i is able to discern its C_{i-1} by testing (for a given C_j):

$$f(C_j \cdot R_{i-1}) \stackrel{?}{=} f(K_{i-1})$$

Therefore, SUCEX-1 prevents Z from learning the topology.

A related issue is that of group size, i.e., the number of barter participants. While it appears difficult to hide this parameter from an STNP altogether, it is possible to *pad* it. Nothing prevents a *bona fide* participant from enlisting the help of one or more dummy participants whose only function is to make the barter group appear larger than it really is. It is also possible for a participant to take on "multiple personalities", i.e., to play multiple roles (in the SUCEX-1 protocol) only one of which is real.

We have shown that SUCEX-1 hides the barter topology from the STNP, although each main participant P_i knows the complete topology. It is possible to modify SUCEX-1 so that the opposite is true. Only Z learns the barter topology, while P_i is concerned only with the local knowledge that it is giving K_i and K_{i-1}. This is probably not a useful design goal when the STNP is a *stranger* randomly pulled out of the crowd. However, this reverse restriction of the cycle information may be important under other circumstances. The modified protocol, SUCEX-2, is shown in Figure 3.

SUCEX-2 is, in fact, a straightforward extension of the original 2-party STNP protocol. Each P_i goes through the same steps and performs the same computations as in the 2-party case. We also no longer need the cumbersome F_n construct; a much simpler F_2 suffices. (Choice of F_2 is the same as in the 2-party protocol.)

5.2 Defending Against Passive Conspiracies

A *passive conspiracy* (PC) occurs whenever a dishonest party (or a group thereof) *conspires* with an honest party without the latter's consent. PCs were first brought up by Beaver in [4]. The 2-party STNP paper also mentions their relevance to STNP-based fair exchange. The following PC scenarios are given:

1. P_1 could misbehave in a way that causes Z to learn K_1.
2. Z could (without any consent) favor P_1 over P_2 and cause it to learn K_2 without P_2 learning K_1.

P_i $\qquad\qquad\qquad\qquad\qquad$ P_{i+1}

$$P_i \xrightarrow{R_i} P_{i+1}$$

P_i $\qquad\qquad\qquad\qquad\qquad$ Z

$$\begin{aligned} C_i &= K_i \cdot \; R_i^{-1} \\ D_i &= f(R_{i-1}) \\ E_i &= f(K_i) \\ G_i &= f(K_{i-1}) \end{aligned}$$

$$P_i \xrightarrow{C_i, D_i, E_i, G_i} Z$$

$$\begin{array}{l} for\ each \text{ i } do \\ \quad E_i \stackrel{?}{=} G_{i-1} \\ \quad E_i \stackrel{?}{=} F_2(C_{i-1}, D_i) \\ endfor \end{array}$$

$$P_i \xleftarrow{\mathcal{C} = \{C_j \mid 0 < j \leq \; n\}} Z$$

$$\begin{aligned} \widehat{K_{i-1}} &= C_{i-1} \cdot \; R_{i-1} \\ f(\widehat{K_{i-1}}) &\stackrel{?}{=} f(K_{i-1}) \end{aligned}$$

Figure 3: SUCEX-2: Fair Cyclic Exchange with STNP

Both types are still applicable in a group setting. Furthermore, a new PC type occurring only in groups is:

3. P_i conspiring to reveal K_i to P_{i+1} before SUCEX-1 completion thereby "framing" P_{i+1}.

The first PC type is not of interest to us since, while Z can be tricked into learning some K_i, it will remain utterly oblivious to this fact (unless it is dishonest which puts it out of the realm of passive conspiracies).

In contrast, PC types 2 and 3 are of some concern. Type 3 can be effectively countered by requiring that each participant only accept a message in round 3 if it is signed by Z. This is not an additional requirement since message privacy and authenticity are already assumed in SUCEX-1.

Recall that in round 3 of SUCEX-1, Z is supposed to broadcast: $\widehat{\mathcal{C}} = \{\widehat{C_j} \mid 0 < j \leq \; n\}$

We consider two sub-types of type 2: one in which Z selectively broadcasts (i.e., directs its message to some participants but not to others) and one in which Z intentionally sends corrupt $\mathcal{C}$, i.e., some C_i values are genuine and some are fake.

In order to prevent an honest participant P_i from becoming and unwilling co-conspirator of Z, the following measures need to be taken:

Before proceeding to search for C_{i-1} in $\widehat{\mathcal{C}}$, P_i computes $C' = \Pi(\mathcal{C})$, i.e., a product of all elements in $\mathcal{C}$. Next, P_i computes $F_{n+1}(C', f(R_1), \ldots, f(R_n))$ and compares it to A_i computed earlier.

In the event that $\mathcal{C}$ does not pass muster, an honest P_i must halt the protocol and not compute K_{i-1}. Otherwise, P_i proceeds to locate C_{i-1} within $\mathcal{C}$ and compute K_{i-1}. Finally, it broadcasts $\mathcal{C}$ to all other participants. This is done for the benefit of other participants who may not have received $\mathcal{C}$ perhaps because of a misbehaving Z.

This procedure ensures that a participant P_i does not learn its barter secret P_{i-1} while some other participant is denied the opportunity to learn its secret. In other words, P_i checks the integrity of the entire set $\mathcal{C}$ – the correctness of all C_i values – before computing K_{i-1}.

5.3 Identifying Barter Opportunities

At the start of a multi-party exchange, the parties to the exchange are fixed, as are the goods that will be exchanged. It is interesting to consider how the parties might arrive at this starting point. After all, there may be many potential barterers, each holding many items for potential exchange. There are a number of ways in which barter opportunities may be identified out of the larger population. In an open outcry system, desired trades might all be public, and possible exchanges could be broadcast to the larger population. Anyone could suggest new exchanges, or insinuate himself into a previously suggested exchange (e.g., "breaking" a link in a cycle to create a larger cycle). Mechanisms for identifying exchanges out in the open, although possibly inefficient, are easy to devise.

If individuals desired privacy, so that trading preferences remained secret whenever possible, then cryptographic techniques could be useful. Recall that potential trades are expressed as d-dimensional vectors over a space of commodities. An exchange is a subset of trades that sum to the all-zero vector. If each trade in the subset has all zeros except for one +1 entry and one -1 entry, then the potential exchange will decompose as a collection of single-unit cyclic exchanges. In principle, finding such subsets can be done using techniques from secure multiparty computation [12, 5, 8]. Since the bulk of the computations are simple sums and comparisons, it is possible that efficient protocols could be found for this problem using existing techniques. This is an interesting area for further investigation.

6 Conclusions

In conclusion, we have presented a classification of barter scenarios. For the case where each participant gives a single item and receives a single item, we have constructed fair exchange protocols in the semi-trusted neutral-party setting. One important open question is how to extend these protocols to the problem of multi-item exchange. Another interesting question concerns the *identification* of barter opportunities. Another interesting open question is to develop efficient protocols for identifying barter opportunities with privacy, as discussed in the preceding section. It would also be valuable to develop algorithms for composing barter cycles (and groups thereof) in the context of both multi-unit and single unit general exchanges.

References

[1] N. Asokan, V. Schoup, and M. Waidner. Optimistic fair exchange of digital signatures. Research Report RZ 2892 (# 90840), IBM Research, December 1997.

[2] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for multi-party fair exchange. Research Report RZ 2892 (# 90840), IBM Research, December 1996.

[3] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In *ACM Conference on Computer and Communication Security*, April 1997.

[4] D. Beaver. Security, fault tolerance and communication complexity in distributed systems. Ph.D. Thesis, Harvard University, May 1990.

[5] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault tolerant distributed computation. In *Proceedings of ACM STOC 1988*, 1988.

[6] M. Blum. How to exchange (secret) keys. *ACM Transactions on Computer Systems*, 1:175–193, 1983.

[7] J. Camp, M. Harkavy, J. Tygar, and B. Yee. Anonymous atomic transactions. In *2nd USENIX Workshop on Electronic Commerce*, pages 123–134, November 1996.

[8] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proceedings of ACM STOC 1988*, 1988.

[9] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *IEEE Fondations of Computer Science*, 1985.

[10] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *IEEE Fondations of Computer Science*, 1987.

[11] M. Franklin and M. Reiter. Fair exchange with a semi-trusted third party. In *ACM Conference on Computer and Communication Security*, April 1997.

[12] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of ACM STOC 1987*, 1987.

[13] S. Ketchpel and H. Garcia-Molina. Making trust explicit in distributed commerce transactions. In *IEEE ICDCS*, 1996.

[14] M. Luby, S. Micali, and C. Rackoff. How to simultaneously exchange a secret bit by flipping a symmetrically-biased coin. In *IEEE Symposium on Foundations of Computer Science*, pages 11–21, November 1984.

[15] J. Tygar. Atomicity in electronic commerce. In *ACM Symposium on Principles of Distributed Computing*, pages 8–26, 1996.

[16] U. Vazirani and V. Vazirani. Trapdoor pseudo-random number generators, with applications to protocol design. In *IEEE Symposium on Foundations of Computer Science*, pages 23–30, 1983.