Building Trust For Distributed Commerce Transactions *

Jiawen Su
Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
sjw@cs.cmu.edu

Daniel Manchala
Xerox Coporation
ESAE-231
701 S. Aviation Blvd.
El Segundo, CA 90245
manchala@cp10.es.xerox.com

Abstract

The explosive growth of the Internet exposes unprecedented commercial opportunities over the network. The vast computer networks easily bring together customers and vendors who are physically distributed in different continents. Current research in electronic commerce mainly focuses on payment mechanisms. However, the global presence of customers and vendors makes it difficult to build trusts among the parties involved in a transaction. In this paper, we will develop systematic mechanisms to address the trust problem in network commerce environment. We first develop protocols for single-customer single-vendor case where one customer buys goods from one vendor. Then we derive protocols for more complicated environments such as when a customer wants to buy goods from several vendors as part of a single transaction. Most of our protocols require some trusted third-party to intermediate transactions. However the workload and requirements placed on the trusted third-party vary from the whole goods and payment store-and-forward to very simple book-keepings of encrypted keys. Our contributions also include an algorithm to find the optimal trusted agent.

1 Introduction

The explosive growth of the Internet exposes unprecedented commercial opportunities over the network. The vast computer networks easily brings together customers and vendors who are physically distributed in different continents. Through the World Wide Web, a starting vendor could achieve global presence which was privileged only for the largest international conglomerates a couple of years ago. The cheap and easy start-up (anyone who has a PC with Internet access is able to run his own Web server) also allows single-person shops to sell goods world wide. For example, an amateur artist may sell some interesting electronic post cards she created.

Current research in electronic commerce mainly focuses on payment mechanisms. However, the global presence of customers and vendors also poses some unprecedented challenges beyond simple payment mechanisms which bear no resemblance in the real world commerce: the customer and vendor may have never met or conducted business before and they may not trust each other in commerce transactions. Some physical distributions of the customers and vendors even aggravate the situation. For example, the customer and vendor may live in different countries governed by hostile governments and different law systems. This suggests that a lot of customers and vendors may even not want to conduct commerce if there is no mechanism to build trust among the parties involved in the commerce transaction. It is not surprising if we look at the real financial world: besides the bank industry, there are whole insurance and security industries to reduce the inherent risks in financial transactions.

In this paper, we will develop systematic mechanisms to address the trust problem in network commerce environment. Most of our protocols rely on a trusted third-party, which could be an independent service provider to facilitate network transactions. The role of the trusted third-party in our protocols varies from a fully trusted entity to a merely backup storage. We also study various factors, including various requirements on the trusted third party, payment arrangements, signature methods, fault-tolerant considerations, number of vendors involved in the transaction, and different degree of risks placed on the parties involved in the transaction. By varying different factors, we have developed a group of protocols which provide different trade-offs suitable for different environments. We also present an algorithm to find the optimal trust agent.

In the open network environment, messages are easy to be injected, forged, intercepted and modified. Thus, in order to build any trust it is necessary to apply strong cryptographic techniques to safeguard against potential attacks. Most of our protocols use some form of cryptographies, which is how our work differs from some earlier works[1]. We believe it is impossible to build trust without any security measures. Our protocols frequently use some cryptographic techniques and maintain the accountability[6]. We have shown earlier that protocols lacking of accountability will suffer from serious security problems if malicious attackers exploit computer and communication failures[2].

We first develop protocols for single-customer single-vendor case where one customer buys goods

*To appear in *Proceedings of 17th IEEE International Conference on Distributed Computing Systems*, Baltimore, May 1997

from one vendor. Then we derive protocols for more complicated environments such as when a customer wants to buy goods from several vendors as part of a single transaction. The customer will buy goods if and only if all the vendors involved in the transaction supply goods as promised. This situation is also studied in [1] , however [1] only gives a feasibility algorithm.

The rest of the paper is organized as follows. Section 3 describes our system model. Section 4 presents a simple protocol prototype to establish trust between one customer and one vendor. Section 5 develops several techniques to overcome the problems with the simple protocol. Section 6 then classifies a group of protocols used between a single customer and a single vendor by combining all techniques developed so far. Section 7 extends it to purchases involving one customer and multiple vendors and also presents algorithms to find a commonly trusted agent. Section 2 relates our works to others. The paper is concluded in section 8.

2 Previous Works

The trust problem in electronic commerce was first noticed by Ketchpel and Garcia-Molina in [1], where they developed a technique called sequencing graph to identify if there is feasible way for a customer to acquire some goods from multiple vendors. However the trusted agents in [1] could only be trusted by no more than two parties (it is not explicitly stated in [1], but all examples and methods described there have this property). Moreover, no cryptographic techniques are employed at all, which we believe are fundamental tools to build trust in network environment where messages could be easily intercepted, modified, injected or forged. Also our protocols are based on the assumptions that a common trusted agent could be found in a transaction. Nevertheless, their technique is still useful when there is no trusted agent commonly trusted by all parties involved in a transaction.

Yahalom, Klein and Beth studied the trust relationships in authentication protocols [4]. They classified seven classes of trusts involved in authentication, for example, one player may trust another in generating good- quality session key but may not trust it in maintaining some secrets. By reasoning about the trust relations, they identified problems in some well-known authentication protocols e.g. Kerberos. Though trust in exchange is different from those in authentications, their works could help us reason about fine-grain trust in electronic commerce. The trust described in this paper is coarse-grain. It may be useful to classify different classes of trust as well as possibility trust.

Lai, Medvinsky and Newman observed in [5] that traditional techniques like insurance, endorsement, and license will be useful to increase confidence of a client in a less-known distributed server. They also described how to implement such a certificate in cryptography so that the client could verify a server's certificate before talking to it. Their ideas could be used in our protocols, for example, to increase the confidence in a trusted agent. But certificates could only increase the overall confidence in a server, not every transaction. On the other hand, our work guarantees the trust in every transaction.

Our earlier works[2] found that most existing electronic payment protocols have fault-tolerance problems, that is, a malicious player could easily steal goods or money by exploiting the fault-tolerant weaknesses, e.g. force a failure during the transaction. We also proposed a couple of building blocks which could be used in electronic commerce protocols to remedy the problem. Works here take one step further to reduce or avoid risks for a customer to interact with foreign vendors though we may already have fault-tolerant electronic payment mechanisms.

Slightly less related works lie in payment mechanisms and protocols. Payment systems study the problem on how to transfer money from one party to another. Most of the proposed payment systems fall into three broad categories: electronic currency[10, 11, 12], debit/credit instrument[13] and secure credit card [14, 15]. Most of currency systems do not provide transferable cash, which means the merchant has to deposit the digital cash tokens to get the real money before he could spend them elsewhere. In an electronic currency system, money is denoted as a token which is a sequence of digital bits. In debit/credit instrument systems, customers and merchants register accounts with payment servers. When a customer buys goods from a merchant, she signs a payment instrument directed to her payment server. Secure credit card systems base the payment on traditional credit cards by sending an encrypted credit card number over the network. Almost all of the payment systems do not provide transferable cash, which means the merchant has to deposit or get the digital cash tokens/instrument/credit-transfer to get the real money before he could spend them elsewhere. Our trust protocols are independent of any payment systems and we could employ any suitable payment system for payment in our models. There is one particularly interesting payment system called NetBill[13] , which provides delivery guarantee to safeguard against failures. The certified delivery could be viewed as a form of trust, where the payment server serves as the trusted third party. However, NetBill employs a central payment server, incurring two fundamental problems from our point of view in trust. First, it implies a global trusted agent. As we argued later, it is very difficult to build a global trusted agent on the Internet. Second, the system has a serious scalability problem, one single central server has no way to handle all the traffic on the Internet.

3 System Model

There are several players in our model: a customer(C), a vendor(V), a bank (B) and a trusted agent(TA). The players could be physically located anywhere. They are interconnected through a communication network. There are processes running on computers on behalf of each player and they can communicate with the others by sending messages over the network. We also assume that there are authentication services available so that two players could set up a secure channel between each other [1].

The bank provides financial or payment services; for example, it maintains money for a customer at his account. The customer could either buy electronic currency from the bank or spend money deposited in the bank. When a customer buys goods from a vendor, he needs to present electronic coins or sign some kind of payment order to the vendor.

The trusted agent is a party who provides intermediating service in the commerce transaction. It is modeled after the middle-man in the real world. It is likely that the agent will charge some fee for each transaction, either as some percent of the transaction or a fixed fee for every transaction or other charging methods like a monthly subscription. The customer and vendor may place some restrictions on different agents or vice-versa; for example, a particular entity may be only trusted for transactions below $100. The trusted agent could also faciliate a kind of conditional anonymity for the customer and vendors so that they do not need to disclose their true identities to each other. The anonymity here is conditional in sense that the trusted agent still knows their identities. It is safe for the customer and verdors to conduct business using pseudonyms because the actual exchange is performed and guaranteed by the trusted agent. If required for some special purpose, e.g. an order form a court, the trusted agent may expose the identities of the parties involved in a transaction. This property is important in some countries such as USA because it is required by the laws.

In our investigation of trust protocols, a player's computer or the communication media may fail. In particular, the customer and vendor may also mount any malicious behaviors to force any failures if they could benefit from the failures. Thus we need carefully design our protocols to safeguard failures. We view the players differently in term of failure-resistance. The trusted agent could be built using redundant hardware and software. This is a reasonable assumption because there may be much less trusted agents than the customers and vendors. The cost for extra reliability of the trusted agent could be also offset by charging customers and merchants for transaction fee. On the other hand, the customer may use any kind of computers, including a laptop with very limited resources, thus we could not assume much for them. So does the vendor. Some bigger vendor may be able to employ more hardware and software for higher reliability, but it will be prohibitively expensive for a small vendor. In short, our protocols assume that the trusted agent will be highly reliable and that the customer or vendor may fail at any time. We observed earlier that most of existing payment protocols have problems in the occurrence of failures because it is not clear whether a receiver has received anything or whether the sender has sent out[2]. Since in our trust models, all transactions are done through a trusted agent, much of the unclarity will be eliminated. As a result, most of our protocols here could be directly used to safeguard fault-tolerant problems. However special protections are still needed when we relax the requirements on the trusted agent, as we will see later in this paper.

[1] Because all our players may be globally distributed, it seems that only public-key based cryptography is feasible to setup such a secure channel

3.1 Notations

Throughout the paper, we use the following notations:

- $A \rightarrow B : m$ Player A sends a message m to player B
- $m1, m2$ the concatenation of messages m1 and m2
- m_k encrypted message m using encryption key k
- $[m]_A$ a digital signature on message m by player A. The dig ital implicit ly includes some context information such as time stamp to make the message unique.
- MD(m) a cryptographic digest of message m.

3.2 Security Considerations

There are many cryptographic and security considerations that must be satisfied to make a safe, correct, and secure electronic commerce protocol. To make our discussion more focused, we do not explicitly consider these issues in the protocols discussed below. The readers are strongly recommended to apply well-known security practice for the final protocol if any segment of protocols here are to be integrated for practical use, including checking the final protocols using various reasoning logics. Some excellent suggestions for designing secure protocols are documented in [7].

4 A Simple Protocol

In this section we describe a general protocol framework to which we will add various components later. As illustrated in Table 1, the agent works as an intermediary between the customer and the vendor: it accepts payment from the customer and goods (denoted by m) from the vendor. It then exchanges them: goods to the customer and payment to the vendor. The customer and vendor could freely negotiate details about the transaction e.g. the price. After they agree to initiate the purchase, they will perform the exchange through an agent trusted by both the parties. Either of the parties could specify Texp , the expiration time for the transaction. The trusted agent could terminate the transaction if the other party doesn`t submit the promised payment or goods within the specified period of time.

1.	C $\rightarrow$ V:	price-qury, TID
2.	V $\rightarrow$ C:	price, TID
3.	C $\rightarrow$ TA:	payment, TID, T_{exp}
4.	V $\rightarrow$ TA:	item, TID, C
5.	TA $\rightarrow$ C:	item, TID, V
6.	TA $\rightarrow$ V:	payment, TID, C

Table 1: **A General Protocol**

There are a number of inherent problems with this simple protocol. First, the size of information goods may be very large, e.g. some software could easily reach several megabytes or more. There is a big network overload involved in sending the information goods through the trusted agent, since it will be sent over the network twice (once from the vendor to the trusted agent and other from the trusted agent to the customer). Second, the information goods need to be stored at the trusted agent until payment is available from the customer. Thus, communication bandwidth and storage requirements limit any trusted agent to be only able to handle a modest amount of transactions and drive the per transaction cost high. This two problems constitute the *communication and storage bottleneck* problem.

Third, there is a risk that the agent will also know the content of the goods. This is undesirable, especially if the goods are considered private information. Fourth, the trusted agent should be able to authenticate the customer`s identity and control access of the goods to the customer who has paid the money. All these require authentication and privacy which can be achieved through cryptographic techniques. However, this requires substantially more computing power. These two problems constitute the *authentication and privacy* problem. Much of our works are motivated by these problems. In the following sections, we will develop several techniques to solve those problems.

Above all, there must be some ways for the customer and vendor to find an agent trusted by both. The easiest solution is to set up some globally trusted agents. But two primary factors prohibit the establishment of global agents. First, the global agent has to be located in some physical place governed by the local law system, and it is less likely that the local law system is compatible with the rest of the world (as a quick example, US laws ban exports of strong cryptography products, but many countries allow it). The second factor is that the global trusted agents will inevitably be a performance bottleneck and cannot handle overwhelming requests. Thus we believe a web of trusted agents is a more practical approach.

It is easy to find a commonly trusted agent for transactions involved only one customer and a single verndor, but not trivial for transactions involved several vendors. For this reason, we defer the algorithms to section 7 where we discusss transactions requiring multiple vendors.

5 Relaxing Requirement on Trusted Agents

Now we develop techniques to address the problems of the simple protocol described in section 4. Our first goal is to reduce the communication bandwidth and storage requirement placed on the trusted agent and thus avoid sending the information goods twice over the network. The solution we offer is to let the vendor first deliver encrypted goods to the customer (Table 2, step 2), then the customer submits the payment to the trusted agent (step 3). The trusted agent may then notify the vendor and get the encryption key from the vendor (step 4). The exchange ends when the TA sends the encryption key to the customer and payment to the vendor, as shown in step 5 and 6 respectively. Note that the deliveries of the encrypted goods and encryption should be done in private. Any traditional symmetric- key algorithms such as DES and IDEA could be used for encryption.

1.	C $\rightarrow$ V:	price-qury, TID
2.	V $\rightarrow$ C:	price, $\{m\}_k$, TID
3.	C $\rightarrow$ TA:	payment, TID, T_{exp}
4.	V $\rightarrow$ TA:	k, TID
5.	TA $\rightarrow$ C:	k, TID
6.	TA $\rightarrow$ V:	payment, TID

Table 2: **Protocol to Solve the Bottleneck Problem**

However, the protocol does introduce a fundamental problem. The vendor may not have delivered the encrypted goods at all, or the customer could claim that the encrypted goods is a garbage message . The simple protocol shown in Table 1 does not have this problem because the trusted agent keeps a copy of the goods so that the content of goods could not be denied. We need some mechanisms which could prove the content of the message if any disputes arise. Digital signature is a perfect way to do it. We require either the customer or the vendor sign the encrypted goods. The reason to sign the encrypted goods is that the signature could be verified before TA distributes the payment or encryption key. Some context information will be required to make the signature unique so that attacks replaying old signature could be detected. For the purposes of efficiency, we only need to sign a digest of the message[9]. The digest algorithms have to be cryptographically strong. Examples of the algorithms include MD5 and SHA. If the customer and vendor have a detailed agreement about the goods before the payment (the agreement had better bear signatures of both parties), later a judge could easily decide whether the delivered goods match the agreement. Because the digest function is cryptographically strong, it is almost impossible for the vendor to produce two goods with the same digest, thus the vendor could not deliver one goods while presenting another to fool the judge.

This protocol is shown in Table 3. Steps 1 and 2 of this protocol are similar to the earlier protocols. The vendor sends a message digest of the encrypted goods to the customer and TA as shown by steps 3. and 4. In addition, the vendor also sends encryption key, context information, expiration time of the transaction to the TA as part of step 3. and encrypted goods to the customer as part of step 2. The customer performs a message digest on the encrypted information goods; compares it with the message digest sent by the vendor (integrity check on the information goods sent by the vendor to prevent spurious insertions/deletions/modifications to the message by an intruder), signs on the message digest, and sends it to the TA (step 4). The TA compares the message digests received from the customer, and vendor (the TA verifies the veracity of the goods sent to the customer). If the message digests are the same, payment is made to the vendor (step 6), and the key given to the customer (step 5).

1.	C → V:	price-qury, TID
2.	V → C:	price, $\{m\}_k$, MD(m), TID
3.	C → TA:	payment,[MD$\{m_k\}$]$_C$,TID,T_{exp}
4.	V → TA:	k, [MD$\{m_k\}$]$_V$, TID
5.	TA → C:	k, TID
6.	TA → V:	payment, TID

Table 3: **Protocol to Solve the Cheating Vendor Problem**

The workload of the trusted agent could be further reduced by delegating to the vendor the responsibility to deliver goods or encryption key. That is, after the vendor receives the payment, he will be responsible to deliver the encryption key to the customer. However, the trusted agent still need a copy of the encryption key to safeguard possible failures of the vendor, resulting in the transaction in a unacceptable situation to the customer.

Payment may be also considered a special kind of good, and hence we could apply the techniques we developed so far to the payment. As shown in Table 4, both the customer and vendor deliver an encrypted form of goods or payment. Then the trusted agent exchange the encryption key for them. To avoid possible disputes, the customer and vendor are required to send the signatures on received messages.

1.	C → V:	price-qury, TID
2.	V → C:	price, $\{m\}_k1$, MD($\{m\}_{k1}$), TID
3.	C → TA:	$\{payment\}_{k2}$, [MD$\{m_k\}$]$_C$, k2, T_{exp},TID
4.	V → TA:	k1, [MD$\{m_k\}$]$_V$, TID
5.	TA → C:	k1, TID
6.	TA → V:	k2, TID

Table 4: **Symmetric Exchange Protocol**

All the above schemes do not consider the details of payment mechanisms. There will be many further improvements if we take the payment mechanisms into account. For example, payments could carry transaction information and are verified by the vendor (such as digital cash). Thus we may avoid delivering the payment through the trusted agent and let the vendor only deliver the encryption key after he has verified the payment The important thing is that the payment mechanism has to be able to bind the transaction information with the payment so that the vendor could not deny it later. Payment mechanisms stressing anonymity could not be used for this purpose, because there is no trace to relate a transaction to a payment and the vendor could just get the payment without-delivering the goods. Most of currently proposed payment mechanisms do not consider it.

It is important to note that we did not exploit the message and communication optimizations in this paper. Many of the protocols described in this paper could be further improved by carefully combining some steps without affecting the properties in trusts. The readers are recommended to tune up a specific protocol if they intend to build it into a system.

6 Classification of Protocols

In last section, we introduced several techniques to address problmes in the simple protocols. Most of those techniques are orthogonal and they could be combined into different ways to provide different trade-offs. In this section, we systematically classify all possible protocols by combining the techniques developed so far. Protocols are classified on system aspects like storage, fault-tolerance, and cryptographic properties like authentication, privacy, non-repudiation, etc. We call each of these properties a dimension. Each dimension may take different values. For example, the dimension on *signature on the information goods*, has values - the customer or the vendor or both could perform the signature. By varying the parameter value for each dimension, we get groups of different protocols, including all protocols described earlier and many new ones. Different protocols address different requirements which are suitable for different situations. After we describe the dimensions and trade-offs for them, we will show several protocols by varying the parameters and demonstrate what kinds of trade-off are achievable from them.

6.1 Dimensions and Parameters

- *Storage of information goods or payment at TA.* This parameter addresses requirements regarding whether the TA stores information goods or payment to be forwarded to the customer or the vendor respectively. Storage of information goods or payment at TA requires the TA to have more communication bandwidth and enough storage space. The same message must be sent across the network twice, which will be very expensive for large information goods. Though it may be possible for the trusted agent to delete the message after the transaction, for practical reasons, the trusted agent needs to keep the message for a longer period of time in order to tolerate possible failures at the customer site or the communication breakdowns (the vendor may also be contacted to in case of failures at the customer site in which case the customer does not receive the goods, but the customer may not trust the vendor. In order to avoid sending information goods twice over the network, and minimize the communication bandwidth requirements, but preserve the trusts in the transaction at the same time, the goods should be only delivered in encrypted form only. This requires the delivery of a separate encryption key, resulting in more number of messages.

- *Signature on the information goods or payment.* If the encryption goods or pay ment is delivered directly from the vendor to the customer, some forms of signatures are required to avoid potential disputes (please see section 5 for possible problems without the signature. The four possible options are 1) none 2) the customer signs 3) the vendor signs 4) both the customer and vendor sign. If neither the customer nor the vendor sign on the message, TA has to keep a clear copy of the information goods or payment. Vendors are usually busier than customers. Thus it is preferable for the customer to do the signature because the signature is a pretty expensive operation. A minor advantage for the vendor to do the signature is that it requires less from the customer. Both signatures add a degree of security by clearly proving who is sending the message and who is receiving the message. It may be useful for critical transactions e.g. transactions with very high values. But a secure channel between the customer and vendor usually gives sufficient security . However from the point of view of trust, the receiver of the signature should submit the signature because it serves as a proof that the message is received and there is no transmission error (transmission errors could be detected by comparing the digest in the signature against the message).

- *Processing of payments through the TA.* Some proposed electronic payment systems such as the electronic check have the ability to bind transaction information to the payment (however this also implies the payment mechanism does not preserve anonymity) and the payment is verifiable after the transaction . This may obligate the need to process the payment through TA.
- *Ways to handle information goods and payment.* The information goods and payment could freely use any combination. For example, information goods may apply encryption-based, TA-delivered, customer-signed method, while payment applies a different method.

6.2 Dimension Space

By varying parameter values in different dimensions, we have roughly total 256 potential protocols. Our further investigation shows about half of the protocol space are not practical, but the remaining protocols still provide a wide variety of trade-offs. Before we try some protocols spanned by the space, let us give an example about the space *position* of a protocol described earlier. Let us consider the parameter values for the protocol in Table 3, it is [Doesn not store information goods in TA (with encryption), customer's signature on encrypted goods, vendor delivers the encryption key; Store the payment in TA (no encryption), No signature, TA delivers the payment; Payment goes through TA]. The protocol of Table 2 is not practical because it may incur some unresolvable disputes (see earlier section for details).

The dimensions described in the previous subsection spans more than hundreds of protocols. We have seen some of them in earlier sections, here we give a couple of new examples in the universe. Given the space limit of the paper, it is impossible to investigate every possible protocol, however it should be very straightforward to infer the remaining protocols and their advantages and disadvantages from our discussions in section 6.1.

The first example we give here is to change the signing party from the protocol described in Table 3. Illustrated in Table 5, the vendor signs on the encrypted message. We also let the vendor deliver the encryption key rather than the TA. We also perform a communication optimization here by combining the payment submission and signature submission in one message from the customer to TA. By comparing Table 5 and Table 3, we could clearly see there is a substantial reduction on the communications involved with TA. The parameter for this protocol is [Does not store information goods in TA (with encryption), Vendor's signature on the goods, Vendor delivers the goods; Store the payment in TA (no encryption), No signature on the payment, TA delivers the payment; Payment goes through TA].

1.	C $\rightarrow$ V:	price-qury, TID
2.	V $\rightarrow$ C:	price, $\{m\}_k$, $[\mathrm{MD}\{m_k\}]_V$, TID
3.	C $\rightarrow$ TA:	payment, $[\mathrm{MD}\{m_k\}]_C$, TID, T_{exp}
4.	V $\rightarrow$ TA:	k, TID
5.	TA $\rightarrow$ C:	k, TID
6.	TA $\rightarrow$ V:	payment, TID

Table 5: **The vendor signs on the goods**

The second example delivers the payment in encryption form (Table 6). The customer will do both signatures on the encrypted goods and payment. TA will not deliver neither of the payment and encryption goods. The parameter for this protocol is [Does not store information goods in TA (with encryption), Customer's signature on the goods, Vendor delivers the goods; Does not store the payment in TA (with encryption), Customer's signature on the payment, Customer delivers the payment ; Payment goes through TA].

1.	C $\rightarrow$ V:	price-qury, TID
2.	V $\rightarrow$ C:	price, $\{m\}_k1$, TID
3.	C $\rightarrow$ TA:	$\{payment\}_{k2}$, $[\mathrm{MD}\{m_{k1}\}]_C$, $[\mathrm{MD}\{payment_{k2}\}]_C$,TID
4.	C $\rightarrow$ TA:	k1, $[\mathrm{MD}\{m_k\}]_C$, $[\mathrm{MD}\{payment_{k2}\}]_C$,TID
5.	V $\rightarrow$ TA:	k1, TID
6.	TA $\rightarrow$ C:	k2, TID

Table 6: **Customer signs both the goods and payment**

7 Transactions requiring multiple vendors

In this section, we extend our works to situations where a transaction is meaningful only when all the vendors present their part of the goods. Our approach is based on an assupmtion that a commonly trusted agent could be found. If such an agent could not be found, some additional measures are required such as security deposit or indemnity, as discussed in [1]. We first need to find if there is an agent which could be trusted by the customer and all involved vendors. We describe a simple algorithm based on minimum-path algorithm to find a node (a trusted agent from the web of trusted agents) between the customer node and all the vendor nodes. The customer and each of the vendors may each have a list of agents they could trust, either from the past business experience or from some other sources.

For each purchase, let us constitute a graph G=(N,E) where N is the set of nodes and E is the set of edges. The customer, the vendor and all related agents are nodes in the graph. An edge will be added to the graph if the two adjoining nodes trust each other in the transaction. The customer and vendor may pose some restrictions on the transaction which will affect the addition of edges. The varying degree of trusts between agents may also change the connectability from transaction to transaction. A customer or vendor may be willing to perform a certain kind of purchases only through some trusted agents. Since a trusted agent may charge for a transaction, each node may be associated with a weight. To minimize the cost to perform the transaction via trusted agents, the cheapest path of trusted agents is desired.

For a transaction involving only a single vendor, such a cheapest path of trusted agent could be found using the classical Dijkstra single-source shortest path algorithm. However, for a transaction involving several vendors, we need to find a trusted agent commonly trusted by the customer and all the vendors while minimizing the total costs. This problem is usually classified as the Steiner tree problem[16]. Formally speaking, let A be a set of the nodes in gragh G, called terminals (In our case, the customer, vendor and all TAs nodes are terminals). A steiner tree for A in G is a connected subgraph of G containing all the nodes of A. The Steiner tree problem is to find a minimum-cost Steiner tree. Unfortunatelly, the problem has been proved to be NP-Complete by Karp. Nevertheless, there are several good polynomial-time approximation algorithms. In our problem, it is natural to be formalized as node -weighted Steiner tree because each trusted agent will charge a transaction fee (a node-edged Steiner tree is more general , edges there could be associated with costs too). But it works equally well if we move the weight of a node to the edge(s) leading to the node. Because the edge-weighted approximation algorithms have better performance then node-weighted counterparts, we will modify an approximation algorithm based on edge-weighted Steiner tree. If there is some cost associated with each edge (for example, there may be communication cost), the node-weighted variation is more desirable and the approximation algorithm could refer to [17].

The approximation algorithm is based on so-called minimum cost paths heuristic, Let us denote the set of terminals as TERM and p denotes the number of terminals. The algorithm always maintain a tree T which contains a subset TERMk of TERM; at each step, we add such a node v from TERM - TERMk to T that v has the minimum cost path connecting T among all nodes in TERM - TERMk. More specifically, the algorithm is as follows:

```
randomly choose v from TERM;
T={v}; TERMk = {v};
for  I= 1 to p-1 do {
      mimimum = +infinity;
      for each node j in TERM - TERMk {
          cost[j] = shortest path to T;    (1)
          Path[j] = such a shortest path; (1)
          if (cost[j]) < minimum) {
              minimum = cost[j];
              node = j;
          }
      }
     TERMk = TERMk + {node};
     T = T + Path[node];
}
```

The cost of the spanning tree found by the above approximation algorithm is proved to be bounded within 2 - 2/p factor of the optimal tree in the worst case for any graph G and any terminals TERM. The approximation algorithm itself costs at most O(pn^2) time, since minimum cost paths can be determined by for example Dijkstra's algorithm in at most O(n^2) time. Interested readers may refer to [16] for proofs of the results.

There does exist a problem with the algorithm: it may find a minimum Steiner tree where one of terminal is not a leaf node, which means one of terminals (either a customer or one of the vendors) may be placed trusts. This could be remedied by replacing (1) of the algorithm as follows: if the shortest path is conneted to one of terminals, we choose the second shortest path.

The above algorithm does not directly output a specific agent who is trusted by all parties and will perform the actual exchange. It is not important for customer and vendors to know the agent because they only interact with their own local agents. But some agent has to know it is responsible for a transaction It is easy to derive such an agent. We could add following steps into the algorithm: associate each agent a counter to record the number of the customer and vendors place trust on it, whenever a new path is merged to form a new tree, we modify the counters for the corresponding nodes. At the end of the algorithm, any node whose counter equals to the total numbeof the customer and vendors is what we want (tie could be broken in any order).

Once such a trusted agent is found, we could extend all the protocols in the earlier sections to incorporate multiple vendors. The agent will not perform the exchange until all the vendors required by the customer submit their part of the goods. Some protocols based on encryption will give the customer the opportunity to decide if he should pay. For example, after the customer receives encrypted goods from all vendors, he may be convinced that it is safe for him to send the payment to the trusted agent.

8 Conclusions and Future Works

We have developed various protocols to build trusts in network commerce environment in this paper. Most of our protocols require some trusted third-party to intermediate transactions. However the workloads and requirements placed on the trusted third-party varies from the whole goods and payment store-and-forward to very simple book-keepings of encrypted keys used in the transaction. Our contributions are:

- Developed a broad category of trust protocols to build trusts among the parties involved in the transaction which guarantee customers that he will receive the right kind of goods once he pays and guarantee the vendor that she will receive correct amount of payment once she delivers the goods to the customer. Those protocols provide a rich set of trade-offs suitable for different environment.

- Proposed an algorithm to find a common trusted agent from a web of available agents.

- A simple solution for a transaction requiring multiple vendors to present their goods at the same time.

Regardless of the success of our protocols, we believe it's essential to explore and understand solutions to build trust in network commerce environment. The successful deployment of commercial transactions on the global Internet hinges on finding some workable approaches.

In this paper, we assume the two parties involved in the transaction either fully trust each other or not trust at all. However, trust in the real world is most likely to fall in between the two extremes. In an incoming works, we will formalize a model called trust matrix based on possibility and fuzzy logics to measure trust and study how to accumulate and propagate trust[3]. Measurement of trust could dramatically reduce the cost associated with transactions. For example, after two parties accumulate full trusts through transactions performed by trusted agent, they do not need to process transaction through trusted agents. A less degree of trust may process transactions through trusted agents randomly at a certain possibility.

References

[1] Steven P. Ketchpel, Hector Garcia-Mooina. Making Trust Explicit in Distributed Commerce Transactions. in *Proceedings of 16th International Conference on Distributed Computing Systems*, Hong Kong, 1996

[2] Jiawen Su, J.D. Tygar. Building blocks for Atomicity in Electronic Commerce. in *Proceedings of 6th USENIX security symposium*, San Jose, 1996.

[3] Daniel W. Manchala, Jiawen Su. Risk-based approach towards possibility theoretic electronic payment verification. to submit for publication. Also appeared as Xerox IP961036.

[4] R. Yahalom, B. Klein, Th. Beth. Trust Relationships in Secure Systems – A Distributed Authentication Perspective. in *Proceedings of IEEE Symposium on Research in Security and Privacy*, 1993

[5] C. Lai, G. Medvinsky, and B,C. Newman. Endorsing, Licensing and Insurance for Distributed System Services. in *Proceedings of 2nd ACM Conference on Computer and Communication Security*, November 1994

[6] Rajashekar Kailar. Accountability in Electronic Commerce Protocols. in *IEEE Transactions on Software Engineering*, Vol 22, No. 5, May 1996

[7] M. Abadi and R.M. Needham. Prudent Engineering Practice for Cryptographic Protocols. in *Proceedings of the IEEE symposium on Research in Security and Privacy*, May 1994

[8] R. L. Rivest, A. Shamir, and L. Adleman. A method for Obtaining Digital Signatures and Public-Key Cryptosystems. in *Comm. ACM*, Vol. 21, No.2, pp. 120-126, Feruary 1978

[9] J. Rmpel. One-way Functions are Necessary and Sufficient for Secure Signatures. *Proceedings of 31st IEEE Symp. on Foundation of Computer Science*, 1991

[10] D. Chaum, A. Fiat, and M. Naor. Untraceable Electronic Cash. *Advances in Cryptology-Crypto'88.*

[11] G. Medvinsky and B.C. Newman. NetCash: A Design for Practical Electronic Currency on the Internet. In *Proceedings of First ACM Conference on. Computer and Communication Security*, pp. 102-106, November 1993

[12] S. Brands. Untraceable Off-Line Cash in Wallets with Observers. *Advances in Cryptology-Crypto'93*

[13] M. Sirbu, J.D. Tygar. NetBill: An Internet Commerce System Optimized for Network Delivered Services. in *IEEE Personal Communications*, pp. 6-11, August 1995

[14] M. Bellare et al. iKP - A family of Secure Electronic Payment Protocols. in *Proceedings of 1st USENIX workshop on Electronic Commerce*, July 1995

[15] Visa and MasterCard. The Secure Electronic Transaction (SET) Specification. http://www.visa.com:80/cgi-bin/vee/sf/standard.html.

[16] P. Winterr. Steiner Problem in networks: a Survy. in *Networks*, Vol 17, 1987

[17] P. Klein and R. Ravi. A Nearly Best -Possible Approximation Algorithm for Node-Weighted Steiner Trees. in *Jornal of Algorithms*, Vol. 19, N0. 1, July 1995

[18] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner tree problem on network. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*,