A Method for Exchanging Valuable Data: How to Realize Matching Oblivious Transfer

Shin'ichiro Matsuo (*,**) Wakaha Ogata(*)

(*)Tokyo Institute of Technology
2-12-1 O-okayama, Meguro-ku, Tokyo, 152-8552 Japan
matsuo@crypt.ss.titech.ac.jp, wakaha@ss.titech.ac.jp
(**)NTT DATA Corporation
890-12 Shin-Kawasaki Mitsui Bldg. West Tower 15F,
Kashimada, Kawasaki-Shi, Saiwai-ku, Kanagawa, 212-0058 Japan
matsuosn@nttdata.co.jp

Abstract: Financial services over the Internet, the exchange of valuable digital data, such as digital money, payment information and digital contents are expected to become major business. These services require privacy for the participant's behavior, security against malicious users and fairness for matching. The concept "Matching Oblivious Transfer (MOT)" [MO03] was proposed to satisfy these requirements.

In this paper, we propose a practical protocol of MOT based on public key cryptography and VSS. In this protocol, once a participant sends an order to the market, no interaction between each participant and the market is needed. Thus our protocol is practical.

Keywords: Electronic market, matching, oblivious transfer and secret sharing.

1 Introduction

Due to the widespread penetration of the Internet, many types of financial transactions, i.e. Internet banking, online auction, trading stocks, online shopping and so on, are being conducted over the Internet. In particular, the scale of online stock trading business is growing rapidly. In the near future, we will trade other valuable digital data as music or movie files, vouchers and coupons over the Internet. Thus, we need to construct a secure and efficient

scheme for trading such kinds of digital data. The trading system, hereafter we call it the 'market' system, must support multiple sellers and buyers at one time. The sellers want to obtain digital money in return for providing some kind of digital data, and the buyers want to obtain the digital data in return for the digital money. The value and price of these digital data varies according to time and occasion. Everyone have her/his own intended price for these digital data. The market matches digital data to digital money according to sellers/buyer's intended order price and some reasonable rules.

When we construct such a market system, we must consider several security features. The market system must preserve the privacy of seller/buyer's order price. A buyer obtains only matched digital data, and a seller obtains only matched digital money. The concept of "Matching Oblivious Transfer (MOT)" which can achieve this was described in [MO03]. In this paper, we propose a practical and secure MOT protocol by using a public key cryptosystem and distributed servers. In this protocol, we use a communication model that uses broadcasting, and once a participant sends an order to the market, no interaction between each participant and the market is needed. This significantly lightens the round complexity of this protocol. The protocol in [MM01] has $O(nl)$ round complexity and communication complexity for matching, where n is the number of participants and l is the range of prices. Our protocol has only $O(1)$ round complexity and $O(l) + O(n)$ communication complexity for matching and exchange, both of which are quite small. Our protocol has fairness, confidentiality of digital information, privacy, and anonymity as security features.

Related Work: To send only specified data to a party while preserving privacy, we can use the oblivious transfer (OT) protocol (see Appendix A). In 1-out-of-N OT, the sender knows N values and would like to let the receiver choose any one of them in such a way that the receiver does not learn of the other values, and the sender remains oblivious to the value chosen by the receiver. 1-out-of-N OT was introduced by Brassard et al. in [BCR86]. Aiello et al. proposed a protocol called "priced oblivious transfer" [AIR01] in which a receiver can buy valuable data while hiding his choice and ensuring financial privacy; the provider can sell valuable data while preserving confidentiality of the data not chosen by the receiver. However, none of these schemes provide a matching mechanism.

On the other hand, there are many matching protocols for one seller and multiple buyers that preserve the buyer's privacy. These protocols are variants of the "sealed-bid auction protocols", which was introduced by Frankrin

et al. in [FR96]. The first matching protocol for multiple sellers and buyers was proposed by Matsuo and Morita [MM01]. Crescenzo proposed a stock purchase protocol which keeps hidden the sell/buy amounts [Cre01]. However, none of these protocols have an exchange mechanism.

There is no research or published protocol that can realize the market system, i.e. that can securely match multiple sellers/buyers and exchange digital data.

2 Matching Oblivious Transfer [MO03]

Matching oblivious transfer (MOT) is a multiparty protocol for the market M of sellers and buyers. In MOT, sellers and buyers trade valuable digital data. In this paper, we focus on the stock market and denote these valuable data as 'stock'.

We assume the following situation. There are some sellers $U_{S_i}(i = 1, 2, \cdots)$ who have digitized stocks of the same company. Each seller U_{S_i} wants to sell her stock d_i for some price P_{S_i} in the market. On the other hand, there are some buyers $U_{B_j}(j = 1, 2, \cdots)$ who want to buy the digital stock in return for digital money m_j for some price P_{B_j}. For such trades, each seller makes a sell-order $O_{S_i} = (d_i, P_{S_i})$ for some i, and posts it to the market. Similarly, each buyer makes a buy-order $O_{B_j} = (m_j, P_{B_j})$ and posts it to the market.

The market accepts sell-orders from sellers and buy-orders from buyers, and then matches a sell-order to a buy-order according to some relation R. That is, the market evaluates R for all sell/buy orders, and matches a sell order O_{S_i} and buy order O_{B_j} if $R(O_{S_i}, O_{B_j})$ holds. If $R(O_{S_i}, O_{B_j})$ holds, only U_{S_i} gets m_j and only U_{B_j} gets d_i from the market.

When a protocol conducted by sellers, buyers and the market meets the following requirements, this protocol is an MOT protocol.

Correctness of exchange: If the sell-order O_{S_i} from seller U_{S_i} and the buy-order O_{B_j} from buyer U_{B_j} are matched, then U_{S_i} gets the digital money in O_{B_j} and U_{B_j} gets the digital stock in O_{S_i}.

Confidentiality of digital data: The market cannot get any digital money or any digital stocks. No seller can get any digital money nor any digital stock except the digital money that matches her sell-order. No buyer can get any digital money nor any digital stock except the digital stock(s) that match his buy-order.

Confidentiality of prices: No one knows the prices in each order and the number of orders for each price.

Anonymity: No one knows who got which digital money/stock.

Robustness: The market can reject irregular orders and seller/buyer's irregular actions in this protocol.

In this paper, we consider the following matching rules:

1. If a sell-order and a buy-order have the same price, then these orders should be matched.
2. If more than one sell/buy orders which have the same price, then the first order should be matched to a buy-order/sell-order.

These rules are used in many actual markets, and are quite reasonable.

3 Proposed Scheme

In this section, we show first a high level description of our scheme to illustrate how we assure security against customer's abuse. We then show a detailed protocol to explain how we assure the privacy of customer's choice with distributed servers.

3.1 High-level description

We assume that there is one trusted server in the market and prices of stocks are in the range of $1 \leq i \leq l$. The market server M has variables for sellers $C_i^{(S)}$ and for buyers $C_i^{(B)}$ for each $i = \{1, 2, \ldots, l\}$, which indicates a current identifier of sell/buy order for price i. The initial value of each variable is a random number, however $C_i^{(S)}$ and $C_i^{(B)}$ begin with the same number for each i.

The public key $PK_{i,c}^{(S)}$ and the private key $SK_{i,c}^{(S)}$, which are used to encrypt and decrypt digital stock, are computed deterministically from the price i and the value $c = C_i^{(S)}$ of the order. Similarly, the public key $PK_{i,c}^{(B)}$ and the private key $SK_{i,c}^{(B)}$, which are used to encrypt and decrypt digital money, are computed deterministically.

To avoid heavy round complexity, we use broadcast communication for matching and exchange. The high-level description is as follows.

Setup: M initializes all variables. M generates random numbers r_i and

$$C_i^{(B)} = C_i^{(S)} = r_i \quad (i = 1, \ldots, l). \tag{1}$$

Sell-order subprotocol: We suppose seller U_S wants to make an order to sell her stock at price P_S. Let d be the digital stock.

Step1. M performs the following procedures for all $i \in \{1, 2, \ldots, l\}$. M makes public key $PK_{i,c}^{(S)}$ from $c = C_i^{(S)}$. M then creates $X^{(i)}$ which indicates order price i and computes digital signature $sig^{(i)}$ for $X^{(i)}$. M sets $EX^{(i)} = (X^{(i)}, sig^{(i)})$. Next, M makes private key $SK_{i,c}^{(B)}$ from c. M then chooses partial keys $SK_{i,c}^{(B),(1)}$ and $SK_{i,c}^{(B),(2)}$ randomly and ensures that they satisfy

$$SK_{i,c}^{(B),(1)} + SK_{i,c}^{(B),(2)} = SK_{i,c}^{(B)}.$$

Finally, M sets $DK_i = (PK_{i,c}^{(S)}, EX^{(i)}, SK_{i,c}^{(B),(1)})$.

M then executes $(1, l)-$OT with U_S. In this OT, U_S receives only DK_{P_S}.

Step2. U_S encrypts d with $PK_{i,c}^{(S)}$, and gets EE as a result of this encryption. She then sends EE, $X^{(P_S)}$ and $sig^{(P_S)}$ to M.

Step3. M verifies the signature $sig^{(P_S)}$ by using $X^{(P_S)}$ and its public key. M then executes $(1, l)$-OT with U_S. In this OT, U_S receives only $SK_{P_S,c}^{(B),(2)}$. U_S can get $SK_{P_S,c}^{(B)}$ from $SK_{P_S,c}^{(B),(1)}$ and $SK_{P_S,c}^{(B),(2)}$.

Next, M inserts EE into its DB_S. This database is for selling stock. M then updates $C_{P_S}^{(S)}$ as

$$C_{P_S}^{(S)} = C_{P_S}^{(S)} \times r_{P_S}.$$

Buy-order subprotocol: The higher description of this protocol is the same as the Sell-order subprotocol.

Broadcast subprotocol: This subprotocol is performed periodically.

Step1. M broadcasts all information in DB_B and DB_S to all buyers and sellers.

Step2-S. U_S decrypts each encrypted digital money in DB_B using $SK_{i,c}^{(B)}$ and checks the validity of the obtained digital money. If it is valid digital money, she gets it as matched digital money m.

Step2-B. Similarly, U_B decrypts each encrypted stock in DB_S using $SK_{i,c}^{(S)}$ and checks the validity of the obtained stock.

Remember that all keys are distinct. Then for each encrypted digital stock EE in DB_S, the result of decryption in step2-B is d for matched data, random data for other cases. This means that U_B with $SK_{i,c}^{(S)}$ can get only the digital stock d that matches his buy order. Similarly, U_S with $SK_{i,c}^{(B)}$ can get only the digital money m that matches her sell order.

If there is no matching stock for U_B, U_B cannot obtain any stock. However, if some seller makes, at a later time, a sell-order that matches U_B, the order will be inserted into DB_S and U_B will be able to obtain it at a later broadcast. Similarly, U_S can obtain matched digital money when a buy order that matches U_S is inserted into DB_B.

3.1.1 How to make secure encryption and decryption keys

In our scheme, all keys must be deterministic with price i, variable c and order kind, sell or buy. Further, they must be distinct and chosen such that no one can guess $SK_{i,c}^{(S)}/SK_{i,c}^{(B)}$ from the other keys.

To meet these requirements, we construct public and private keys as follows.

Let p and q be large primes s.t. $p-1|qL$ where $L \geq 2l$ and consider a multiplicative group generated by a generator g with order p. Let α be the L-th root of 1 on mod p, r be a random number with order q on mod p. The public keys and private keys are calculated as follows.

$$SK_{i,c}^{(S)} = \alpha^i r^{C_i^{(S)}} \bmod p, \quad PK_{i,c}^{(S)} = g^{SK_{i,c}^{(S)}}$$

$$SK_{i,c}^{(B)} = \alpha^{i+l} r^{C_i^{(B)}} \bmod p, \quad PK_{i,c}^{(B)} = g^{SK_{i,c}^{(B)}}$$

$C_i^{(S)}$ and $C_i^{(B)}$ are calculated as,

$$C_i^{(S)} = r_i^j \bmod q, (j = 1, 2, \ldots)$$

$$C_i^{(B)} = r_i^k \bmod q (k = 1, 2, \ldots).$$

Here, r_i is a random number. Then, all of the above requirements are satisfied. These keys can be used in an ElGamal cryptosystem.

3.2 Protocol with distributed market servers

3.2.1 How to assure security and privacy

To achieve confidentiality of prices, we use a verifiable secret sharing scheme (VSS) and we split the functionality of the market between server M_{p} and multiple servers $M_1, \ldots, M_n$. M_{p} only makes distributed information of price information $X^{(i)}$ to avoid creating irregular orders by seller/buyer and obtaining order price by the market server. M_{p} does not deal with any information sent from seller/buyer. $M_1, \ldots, M_n$ manage variables $C_i^{(S)}, C_i^{(B)}$ and creates encryption and decryption keys in distributed manner. There are up to $t-2$ malicious servers, and $n = 2t$. Order prices P_S and P_B are distributed using VSS, and all operations for $C_i^{(S)}, C_i^{(B)}$ are conducted by using a homomorphic (t, n) threshold VSS scheme [GRR98]. Updates of variables are executed by using the homomorphism of VSS and multiplication of secrets over VSS. We use the $(1, l)$-OT protocol to deliver only appropriate keys while preserving buyer's and seller's privacy.

3.2.2 Setup

The market servers publish g, r and α as public parameters.

To initialize $C_i^{(S)}, C_i^{(B)}$, n servers jointly generate random numbers r_i, and set them as eq. (1). The values of $C_i^{(B)}$ and $C_i^{(S)}$ are always distributed among the servers, and less than t servers cannot know them. r_i is also distributed to all servers.

3.2.3 Sell-order subprotocol

Step1-1. Let c_i be the current value of $C_i^{(S)}$ and $c_{i,j}$ be the j-th share of the current value of $C_i^{(S)}$. As we showed in 3.1.1, the public key $PK_{i,c}^{(S)}$ equals $g^{\alpha^i r^{c_i}}$. Here,

$$c_i = \sum_{0 \le j \le t} c_{i,j} \lambda_j$$

where λ_j is a Lagrange coefficient. Thus,

$$g^{\alpha^i r^{c_i}} = (g^{\alpha^i})^{r^{c_i}} = (g^{\alpha^i})^{\prod_j (r^{c_{i,j}})^{\lambda_j}}.$$

To make the public key while hiding its from any $t-2$ servers, the market servers execute the following procedure.

For all i, each of t servers of $M_j (1 \leq j \leq n)$ locally calculates $r^{c_{i,j}}$. Two servers then jointly calculate

$$PK_{i,c}^{(S),(1)} = (g^{\alpha^i})^{\prod_j (r^{c_{i,j}})^{\lambda_j}},$$

and encrypt it with seller's public key PkU_S. They then gets $EPK_{i,c}^{(S),(1)}$ as a result. On the other hand, each of the remaining $t-2$ servers encrypts $(r^{c_{i,j}})^{\lambda_j}$. They jointly collect $EPK_{i,c}^{(S),(2)} = (Enc\langle PkU_S\rangle((r^{c_i,j})^{\lambda_j}), \cdots)$ for all i. Here, $Enc\langle key\rangle(mes)$ represents the ciphertext of message mes created using the public key key.

Step1-2. For all i, let $X^{(i)} = (X_1^{(i)}, \ldots, X_l^{(i)})$ and $X_k^{(i)}$ be 1 if $i = k$ and 0 otherwise. $M_{\rm p}$ makes n shares of all $X_k^{(i)}$ for all $i, k \in \{1, 2, \ldots, l\}$. Let $x_{k,j}^{(i)}$ be a share of $X_k^{(i)}$ for M_j. $M_{\rm p}$ adds its digital signatures $sig^{(i,j,k)}$ to all $x_{k,j}^{(i)}$. Next $M_{\rm p}$ encrypts $(x_{k,j}^{(i)}, sig^{(i,j,k)})$ for all i, j, k with U_S's public key PkU_S. Then $M_{\rm p}$ calculates the hash value $H_j^{(i)} = H(x_{1,j}^{(i)}||\cdots||x_{l,j}^{(i)})$ for all i, and shuffles them. Here H is a collision intractable one-way hash function. Finally $M_{\rm p}$ sends

$$EX_j^{(i)} = (Enc\langle PkU_S\rangle(x_{1,j}^{(i)}, sig^{(i,j,k)}), \ldots, Enc\langle PkU_S\rangle(x_{l,j}^{(i)}),\ H_j^{(\pi(i))})$$

for all i to $M_j (1 \leq j \leq n)$.

Step1-3. Each server M_j generates keys $SK_{i,j,c}^{(B)}$ for all i where c is the current value of $C_i^{(S)}$.

$$SK_{i,j,c}^{(B)} = \alpha^i r^{c_{i,j}}$$

M_j chooses partial keys $SK_{i,j,c}^{(B),(1)}$ and $SK_{i,j,c}^{(B),(2)}$ randomly but ensures that

$$SK_{i,j,c}^{(B)} = SK_{i,j,c}^{(B),(1)} + SK_{i,j,c}^{(B),(2)}.$$

Finally, each M_j encrypts $SK_{i,j,c}^{(B),(1)}$ and $SK_{i,j,c}^{(B),(2)}$ with U_S's public key PkU_S. The market servers collect $ESK_{i,c}^{(B),(1)}$ as the set of all ciphertexts of $SK_{i,j,c}^{(B),(1)}$ for j, and $ESK_{i,c}^{(B),(2)}$ as the set of all ciphertexts of $SK_{i,j,c}^{(B),(2)}$ for j.

Step1-4. The market servers set

$$DK_i := (EPK_{i,c}^{(S),(1)}, EPK_{i,c}^{(S),(2)}, EX_1^{(i)}, \ldots, EX_n^{(i)}, ESK_{i,c}^{(B),(1)}).$$

Market servers jointly execute $(1, l)$-OT with U_S. In this OT, U_S receives only DK_{P_S}.

Step2. Let $i = P_S$. U_S decrypts $EPK_{i,c}^{(S),(1)}$ and $EPK_{i,c}^{(S),(2)}$, and reconstructs the public key from $PK_{i,c}^{(S),(1)}$ and $PK_{i,c}^{(S),(2)} = (a_1, \cdots, a_{t-2})$ as follows.

$$PK_{i,c}^{(S)} = (\ldots(PK_{i,c}^{(S),(1)})^{a_1})^{a_2}\ldots)^{a_{t-2}}$$

Next, U_S encrypts his digital stock with $PK_{i,c}^{(S)}$ and gets EE as a result. Then, U_S decrypts all items of $EX_j^{(P_S)}(1 \leq j \leq n)$ to obtain $x_{k,j}^{(i)}$ and $sig^{(i,j,k)}$ for all j, k. Finally, U_S sends EE, $x_{k,j}^{(P_S)}$ and $sig^{(P_S,j,k)}$ for all k to $M_j(1 \leq j \leq n)$.

Step3. Each server verifies the validity of $sig^{(P_S,j,k)}$. They then verify that there exists a stored hash value $H_j^{\pi(i)}$ that equals the hash value of the concatenation of all $x_{k,j}^{(P_S)}$.

Next, the servers store EE into DB_S. The servers then send $ESK_{P_S,c}^{(B),(2)}$ to U_S by using $(1, l)$-OT. U_S can get $SK_{P_S,c}^{(B)}$ from $ESK_{P_S,c}^{(B),(1)}$ and $ESK_{i,c}^{(B),(2)}$ by using the reconstruction protocol of VSS.

Finally, all servers jointly update the variables as follows.

$$C_i^{(S)} := C_i^{(S)} X_i^{(P_S)} r_i - C_i^{(S)} X_i^{(P_S)} + C_i^{(S)} \quad (i = 1, \ldots, l)$$

3.2.4 Buy-order subprotocol

The buy-order subprotocol is similar to the sell-order subprotocol.

3.2.5 Broadcast subprotocol

This subprotocol is same as the one described in3.1. Each M_j can see all broadcast information. However, the contents in the database are encrypted and since no M_j knows the keys, the market learns nothing.

3.3 Analysis

Security: Correctness of exchange is assured because encryption/decryption keys are delivered accoring to $C_i^{(S)}/C_i^{(B)}$ of each sell/buy order and digital data/money are delivered by broadcasting. Confidentiality of digital data is assured because broadcasted digital data/money is encrypted, and one key

can not be guessed easily from the other keys. Confidentiality of price is assured through the use of VSS and key delivery via $(1, l)$-OT. Anonymity is assured through the use of broadcasting and VSS. Robustness is assured by the check procedure in step3.

Efficiency: In this protocol, we use $(1, l)$-OT where l is the range of prices and exchanges are broadcast. In our scheme, the round complexity is $O(1)$ and the communication complexity is $O(l) + O(n)$ for matching and exchange, where n is the number of customers. However, the existing scheme[MM01] has $O(nl)$ round complexity and communication complexity for matching. Our scheme offers improved efficiency.

4 Conclusion

We proposed a practical protocol to realize "matching oblivious transfer." It can exchange digital data and digital money according to some rules as in an actual market while preserving privacy. Its security is based on public key cryptography, VSS and OT. Because this protocol uses the broadcast communication model, we have significantly reduced the round complexity and communication complexity.

References

[AIR01] B. Aiello, Y. Ishai and O. Reingold, *Priced Oblivious Transfer; How to sell Digital Goods*, Advances in Cryptology - EUROCRYPT 2001, Lecture Notes in Computer Science, Vol.2045, pp. 119-135, 2001.

[BCR86] G. Brassard, C. Crépeau and J.-M. Robert, *All-or-Nothing Disclosure of Secrets*, Advances in Cryptology - Crypto '86, Lecture Notes in Computer Science, Vol.263, pp.234-238, 1987.

[BGW88] M. Ben-or, S. Goldwasser, and A. Widgerson, *Completeness theorems for non-cryptographic fault-tolerant distributed computation*, STOC '88, pp.1-10, 1988.

[Cre01] G. D. Crescenzo, *Privacy for the Stock Market*, In Pre-Proceedings of Financial Cryptography '01, Grand Cayman, BWI, February, 2001.

[EGL85] S. Evan, O. Goldreich and A. Lampel, *A Randomized Protocol for Signing Contracts*, Communications of the ACM, Vol.28, No.6, pp. 637-647, 1985.

[FR96] M. Franklin and M. Reiter, *The Design and Implementation of a Secure Auction Service*, IEEE Trans. on Software Engineering, Vol.22, No.5 (1996).

[GRR98] Gennaro and Rabin and Rabin, *Simplified VSS and Fast-track Multiparty Computations with Applications to Threshold Cryptography*, PODC: 17th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, 1998.

[MM01] S. Matsuo and H. Morita, *Secure protocol to construct electronic trading*, IEICE Transactions on Fundamentals of Electronics, Communication and Computer Sciences, VOL.E84-A, No.1, pp.281-288, January 2001.

[MO03] S. matsuo and W. Ogata, *Matching Oblivious Transfer: How to Exchange valuable data*, IEICE Transactions on Fundamentals of Electronics, Communication and Computer Sciences, VOL.E86-A, No.1, pp.189-193, January 2003.

[NP01] M. Naor and B. Pinkas, *Efficient Oblivious Transfer Protocols*, In proceedings of SODA 2001.

[NP99] M. Naor, B. Pinkas, *Oblivious Transfer and Polynomial Evaluation*, Proc. of 31st ACM Symposium of Theory on Computing, pp.245-254, 1999.

[Sha79] A. Shamir, *How to Share a Secret*, Communications of the ACM, Vol.22, No.11, pp.612-613, 1979.

A Oblivious transfer

Oblivious transfer (OT) protocol is two-party protocol such that sender Alice sends messages then chooser Bob receives one of them with hiding which message he gets and anything about the other messages.

1-out-of-2 oblivious transfer, which we denote as $(1, 2)$-OT, is popular type of OT protocols. In $(1, 2)$-OT, Alice sends message (m_1, m_2) to Bob. Bob chooses index of message $i \in \{1, 2\}$ which he wants to receive. After $(1, 2)$-OT finished, Bob gets message m_i, and receives nothing about another

message. At the same time, Alice cannot know about i. $(1,2)$-OT was suggested by Evan et al. [EGL85].

1-out-of-N oblivious transfer [BCR86], which we denote as $(1,N)$-OT, is expanded from $(1,2)$-OT. In $(1,N)$-OT, Alice sends messages $(m_1, m_2, \ldots, m_N)$ to Bob. Bob chooses index of message $i(1 \leq i \leq N)$ which he wants to receive. After $(1,N)$-OT finished, Bob gets message m_i and receives nothing about other messages. Alice cannot know about i. Efficient $(1,N)$-OT protocol was proposed by Naor and Pinkas [NP99, NP01]. This $(1,N)$-OT can be used to send a message of any data structure.

B Verifiable secret sharing

A secret sharing scheme (SS) is an important primitive to construct cryptographic protocols. In SS, a dealer has secret information s and creates n shares of this secret. Then he delivers each share to each player. Each player cannot know anything about secret s from his share. However, only when a qualified subset cooperate, they can reconstruct s from their shares. Most of SSs are (t,n) threshold secret sharing, in which up to $t-1$ players cannot reconstruct shared secret, but with cooperation of t or more players, they can reconstruct it. The most popular (t,n) threshold SS is Shamir's scheme proposed in [Sha79].

Verifiable secret sharing scheme (VSS) is a kind of secret sharing scheme in which players can verify whether his share is correctly calculated from s, Ben-or et al. proposed a popular VSS in [BGW88]. We use (k,n) threshold VSS to construct trusted servers in the market.

Most SSs have homomorphism, which is important feature for cryptographic protocols. Let $(S_1(s_1), S_2(s_1), \ldots, S_n(s_1))$ be shares of secret s_1 and $(S_1(s_2), S_2(s_2), \ldots, S_n(s_2))$ be shares of secret s_2. Then $s_1 + s_2$ is reconstructed from $(S_1(s_1) + S_1(s_2), S_2(s_1) + S_2(s_2), \ldots, S_n(s_1) + S_n(s_2))$ in homomorphic SS.

To realize multiplication of secrets which are shared by using VSS, we can use a protocol proposed by Gennaro et al [GRR98].