

On the Fair Resource-Sharing in Large Peer-to-Peer Systems¹

Chunhua Tang, T. C. Lam and Jyh-Charn Liu²
{ctang, tbl4427, liu}@cs.tamu.edu
Computer Science Department
Texas A&M University
College Station, TX 77843-3112

Technical Report 2003-8-7

August 14, 2003

Abstract

Accountability is becoming a major concern for the Internet. Wide spread of misconduct by profit driven hackers makes trust (faith)-based applications vulnerable to malicious attacks. An evidence based resource management system can guarantee that collaborative parties will share their resources based un-deniable evidence, so that community members cannot abuse their privileges without being caught. In this paper we propose fair resource sharing scheme for peer to peer (P2P) networks based on modified e-coin algorithms. A computing node joins a resource sharing community through the initial brokerage of a credential authority, which issues unforgeable yet anonymous transaction tokens, so that it can use the token to generate evidences of resource sharing with other nodes. We study offline fair sharing of resources in P2P systems, in which fraudulent activities can be detected based on the undeniable evidences generated in the transactions. These evidences can be used for unveiling the identity of a participant attempting to forge the evidences in transactions.

Keywords – fair resource sharing, P2P systems, e-coin, un-deniable evidence.

¹ This work is supported in part by an NSF ITR grant, EIA-0081761, and by a TAMU-CONACYT grant

² Correspondence author

1. INTRODUCTION

Resource sharing is a major benefit of distributed computing systems. Computing resources, such as processor cycles, disk space, and other computing resources can be shared among a group of computer nodes. The wide success of peer-to-peer (P2P) networking applications [1]-[3] suggests that the P2P architecture is a much more flexible architecture for resource sharing. In current practice, nodes are usually under the control of a centralized administrative entity to determine whether or not anyone can access system resources. The centralized authorization, authentication and access control of computing resources are not so flexible as a distributed architecture for the management of computing resources.

A main concern of resource sharing in a P2P network is how to guarantee fairness of service exchange. Without the support of un-deniable evidences, it is difficult to implement any kind of accounting systems in a P2P network [4]. A decentralized resource-sharing scheme is needed to support autonomous and secure management of distributed computing resources for large scale P2P systems.

In this paper, we propose a modified e-coin algorithm to implement a P2P based fair resource-sharing scheme. We are primarily interested in offline resource sharing schemes, where the credential authority is only responsible for issuing *transaction tokens* (e-coins) to nodes. After that, nodes directly exchange resources without the involvement of an online central arbiter. Although the scheme is applicable to any kind of resource sharing, for simplicity we focus on sharing of storage space in this paper. In this case, the notion of fairness is simple:

F₁: No peer can use more space than it shares.

F₂: Any peer must serve a valid quest if the space it commits to share has not been entirely utilized by others.

Both F₁ and F₂ can easily be achieved in the traditional online quota based systems where every peer gets permission from a central authority before it can access the shared space. But for a P2P system, to achieve F₂ every node would have to prove to others its storage records [5]. It is highly desirable for a P2P resource-sharing scheme to guarantee fairness while still to keep the privacy of the nodes.

In the absence of a central on-line authority, one cannot prevent fraudulent conducts in a P2P network. However, one can create undeniable evidences to document the transactions between P2P users, with minimal intervention from the credential authority. The anonymous payment mechanisms of the e-coin based banking systems [6]-[10][23] are well suited for our needs. An e-coin is a file that contains the signature of

the “bank,” or the credential authority. In a typical e-coin system, only the owner of the e-coin can use it to make payment for acquisition of services. The authenticity of the e-coin is guaranteed by the bank’s signature, even though the bank may not know the identity of the e-coin owner. When the e-coin owner tries to spend an e-coin twice, its identity can be revealed by the bank.

The traditional e-cash schemes are mostly based on centralized management architecture, in which all the histories of spending need to be validated by the bank for culprit detection. Moreover, allowing an issued e-coin to be used just once for resource trading will incur excessive overhead for resource sharing. To eliminate the deficiencies of existing systems, in this paper we propose a distributed credit-evidence distribution scheme on the e-coin algorithm in [8][9], while to make the e-coins transferable, divisible and reusable. When a node uses the resources owned by others, it trades a portion of its resources to the service provider. After a node provides resources to others, it can use the e-coin received from the service requester to access other resources later. Every node only needs to withdraw one transaction token from credential authority (CA) and every node is responsible for detection of double spending to prevent other nodes from accessing its resources improperly.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 introduces the system model and adversarial models. Section 4 gives our modified e-coin algorithm and the details of our resource-sharing scheme and section 5 gives the proof of the correctness. Section 6 gives the divisible version of our scheme. Finally, section 7 concludes this work.

2. RELATED WORK

The P2P architecture is largely divided into two categories: *pure* P2P, in which all nodes are considered identical, and *hybrid* P2P, in which some nodes act as super peers. Recent work in P2P systems, e.g., overlay networks, Patry [11], Chord [12], Tapestry [13], and CAN [14] aimed at P2P applications. In Freenet [15], Free Heaven [16] and Gnutella [17], queries are routed between nodes until results are found, or when the limit on the number of hops is reached.

Several fair resource management schemes have been proposed for P2P systems. Servers in Tangler [18] are required to obtain certificates from other servers to publish files for a limited time. New servers must first provide good services to existing members (to provide its trustworthiness) before they can receive any services. It does not provide accurate fairness, F₁ and F₂.

PAST [19] proposed to use smart card to guarantee fairness. The Eternity Service [20] uses e-cash to solve

the fair sharing problem. But a document cannot be deleted, even if requested by the owner. The service uses e-cash to prevent service denials and provides a limited degree of fairness.

Ngan *et al* proposed a random audit scheme in [5] to enforce fair sharing. The purpose of the scheme is to provide F_1 and F_2 simultaneously. Every node publishes a signed file containing its capacity to be shared, stored files and published files. To prevent cheating, every node needs to conduct random auditing, through an anonymous channel, and the capture rate of cheating nodes is estimated to be 95% after three rounds of auditing, at some communication costs.

A multi-payment scheme for fair exchange was studied in [21]. The scheme makes e-cash transferable at cost of anonymity. The node gets an Anonymous Authentication Certificate (ACC) from Anonymity Server (AS) such that only AS knows the owner of the ACC. Then node uses this ACC to spend the e-cash by signing the cash. When double spending occurs, AS can find the culprit through the public/private key pair. All transactions are not anonymous to AS.

GridBank [25] was proposed to add accountability to a large resource-sharing platform Grid [24]. They deployed the Grid-wide banking service using e-cash for service payment.

3. SYSTEM MODEL

Our scheme is essentially an offline and decentralized credit conservation system that enforces fair resource sharing in the P2P environments. The credit is the amount of the local storage space that a node commits to share with its peers. A node can trade its credit for the remote storage space. The credit can also be transferred between the peer nodes. When a valid credit is presented to a peer node, the request to access the space associated with the credit cannot be denied. If the node does reject a legitimate request, the requester can submit the evidence to the credential authority. Our scheme guarantees both F_1 and F_2 properties in the P2P systems.

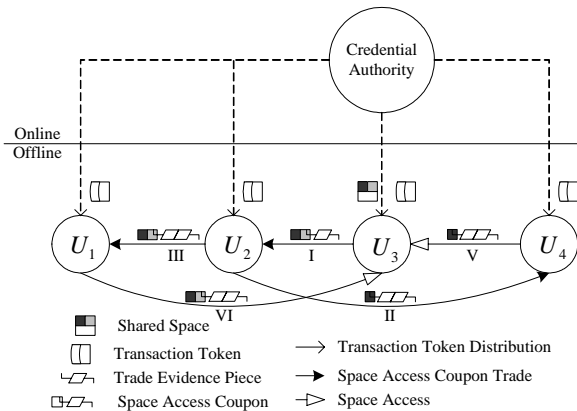


Figure 1. System Architecture.

Figure 1 illustrates the system architecture of our scheme. The P2P nodes (e.g. U_1, U_2, U_3 and U_4) register their shared space with the CA. After registration, each node will get a *Transaction Token* (TT) and a *Certified Storage Space Document* (CSD) that together represent the space that is committed to share with the group by the node. A node can access the storage space of another node by giving the other node the access right to its own space. This process is called space trading. The access right is represented by a *space access coupon* (referred as *coupon* later), which is a CSD appended with the *evidence pieces* of the trading process. The node that holds a coupon can access the space stated in the CSD of the coupon. For example, in Figure 1, U_3 trades half of its shared space to U_2 in step I. After step I, U_2 generates a space access coupon with the evidence piece of step I. Then U_2 can use the coupon to access space in the redemption phase or trade it for other coupons. In step II and step III, U_2 trades its entire coupon to U_1 , and then half of the coupon to U_4 . The evidence pieces of these two transactions are appended to the coupon to generate another two new coupons. In steps IV and V, U_1 and U_4 present their coupons to U_3 to access the space. Step IV and V illustrate a *double spending* problem within the distributed P2P resource sharing system. However, if every node does not trade coupons more than their value, the fairness is archived.

A more detailed description about each component in the architecture is given as follows. When U_i is permitted to join the P2P resource trading, it receives CSD_i from CA. Obviously, CA will verify the relevant information provided by U_i before CSD_i can be issued. A CSD includes four parts – *space descriptor*, a *token identity*, a value K and the *CA signature* of this document. The space descriptor describes the attributes (size, location, etc.) of the space that the node is committed to share. The token identity is the unique ID of each token. K is computed from a secret k by the use of a one-way function and it provides one-time anonymity to the transaction token, that is, if the space has ever been spent more than once, the node identity in the token will be revealed. The CA signature guarantees the integrity of the CSD.

The transaction token associated with each CSD consists of three elements – a unique *token identity*, *node identity* and *secret signed by CA*. The *transaction token* TT_i issued by the CA to U_i is used for generating the undeniable *evidence piece* in trading and redemption of storage space. The evidence pieces are computed from the node identity and the signed secret, but they will not reveal the node identity as long as U_i does not violate the fair trading policy.

Given CSD_i and TT_i , the node U_i can directly trade for space with other nodes in the offline mode. After the

space trading is completed, that is, a node exchanges the access right to some space with the other node, it leaves each node an evidence piece. The evidence piece is essentially an un-forgeable *challenge-response* pair generated by the use of the transaction tokens. Since this step binds new properties to the coupon via challenge x and response r , which also are used in coupon trading, the generation process of the evidence piece is also called *x-r-binding*.

Figure 2 gives the details of the trading steps between U_2 and U_4 . First, U_2 sends its coupon to U_4 . After verifying the coupon, U_4 computes the challenge x from its token identity and a randomly selected secret k . U_2 binds the coupon to the response, which is sent as the evidence piece to U_4 to generate a new coupon. The newly generated coupon can only be used (redeemed or traded) by U_4 . One-time anonymity is added to the coupon because of the unknown secret k . The challenge x makes the generated coupon only be used by the owner of TT_4 with the secret k . The response r guarantees that the evidence piece can only be appended to the coupon.

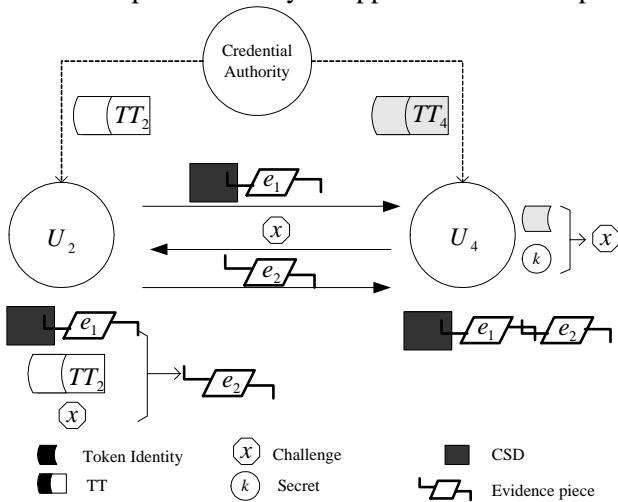


Figure 2. Trading of a space access coupon.

The evidence piece itself does not reveal the node identity. Once a coupon is traded more than its value, the evidence pieces of the transactions can be used to identify the offending node.

Because the storage sizes committed to share by each node are usually different, it is difficult for a node to find another perfectly matched node to trade their storage space. To minimize the interactions between P2P nodes and the CA, we add divisibility to our scheme so that a portion of a coupon can be used for trading. This divisibility attribute uses a binary hash tree rather than a single value k as the secret. Details of this step are discussed in Section 6.

Space trading is offline. We cannot prevent a node from overspending its CSD or coupons. For the example III and IV in Figure 1 U_1 and U_4 do not know that U_2 has

overspent its received coupon. Using of the CSD by a node to meet its current storage needs implies that the node will need to serve some other nodes in the future. Thus, a node may prefer to use coupons even when the node could use its TT and CSD to access the remote storage. For efficiency and fairness, we propose to check for double spending violations at redemption of each coupon. For example, referring to Figure 1, if U_4 redeemed the coupon received in (III) and when U_1 tries to redeem the coupon received in (IV,) U_3 will find that a double spending situation has occurred, and it can figure out U_2 's identity from the two received coupons.

4. DESIGN DETAILS

In this section, we first discuss the non-divisible storage space sharing scheme, and then will discuss the divisible version later. Both schemes use the x-r-binding scheme to make the coupon transferable.

Each node obtains a transaction token t from CA by submitting its unique node identity to the credential authority via the token withdrawal protocol (section 4.1). Then, the node submits its shared space information to CA, CA signs it and returns the CSD to node via a CSD sign protocol. Nodes obtain coupons via the space trading protocol (section 4.2). When a node wants to insert a file to the system, it consumes the space via the space access protocol (section 4.3). File deletion is also processed in the space access protocol.

All cryptographically computations are based on modulo n unless otherwise stated. The RSA public key v of CA and the modulus n are known to everyone. Credential Authority keeps the private key $1/v$ as its secret. Parameters g_1, g_2, g_3 are published large numbers in the multiplicative group of Z_n . The mapping f is a suitable one-way function from Z_n^* to Z_v^* . h_c and h_b are public known elements of order n from F_p where $p-1$ is a multiple of n . In our later discussion, $A = ag_1^{f(a)}, B = bg_2^{f(h_b^b)}, C = cg_3^{f(h_c^c)}$.

A TT is a 6-tuple (a, b, c, U, S, T) that satisfies $S^v = (CA), T^v = (C^U B)$, where (a, b, c) is the token identity, U the node identity, and (S, T) the secrete signature of CA. A CSD is a 6-tuple $W = (L, a, b, c, K, G)$, where L is the space descriptor that collectively represents the space location and the amount of the space being certified. (a, b, c) is the token identity, K is computed from the secret k and G is the signature of CA on this certificate.

The construction of TT is identical to that of Ferguson's e-coin [8][9], except that we do not have secret k in the withdrawal protocol of Ferguson's e-coin. To make a TT re-usable, we add a new secret k via the

x-r-binding that is depicted in Figure 2. The secret k will add another one-time anonymity to TT, see Figure 3. To respond to a challenge x , node U computes $r = Ux + k$ and $R = S^x T^k$, and then sends them with $K = A^{k \bmod v}$ to another node U' . U' can verify the response with $R^v = C^r B^x K$, but cannot calculate the U . In our later discussion, $K = A^k$ means $K = A^{k \bmod v}$.

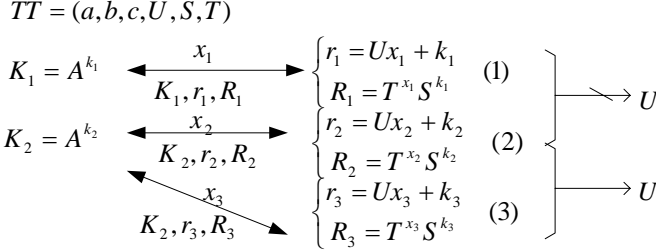


Figure 3. One-time anonymity provided by secret k .

For the construction of a CSD, the node sends to CA its trading space descriptor L , the token identity (a, b, c) , and computes K from a pre-selected secret k . CA returns a signature G on it after verifying the information.

An evidence piece is a 7-tuple (a, b, c, x, r, R, K) that satisfies the condition $R^v = C^r B^x K$. A space access coupon is an integral 9-tuple $P = (W, \ell, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{r}, \mathbf{R}, \mathbf{K})$ where W is a CSD, $\mathbf{a} = (a_1, a_2, \dots, a_\ell)$, $\mathbf{b} = (b_1, b_2, \dots, b_\ell)$, $\mathbf{c} = (c_1, c_2, \dots, c_\ell)$, $\mathbf{x} = (x_1, x_2, \dots, x_\ell)$, $\mathbf{r} = (r_1, r_2, \dots, r_\ell)$, $\mathbf{R} = (R_1, R_2, \dots, R_\ell)$ and $\mathbf{K} = (K_1, K_2, \dots, K_\ell)$ are integer vectors of length ℓ . ℓ is an integer and represents the length of the coupon. A coupon P is valid if $(a_i, b_i, c_i, r_i, R_i, K_i)$ is an evidence piece for $1 \leq i \leq \ell$, and $x_{\ell-1} = f(W, r_{\ell-2}, a, b, c, K_\ell)$ for $3 \leq i \leq \ell$.

4.1 Token Withdrawal Protocol

When a node wants to join the system, it contacts the credential authority and obtains a TT, (a, b, c, U, S, T) . The protocol is almost identical to the Ferguson's e-coin withdrawal protocol [8][9]. The following summarizes the main features:

- Node presents its identity, U , to CA.
- Node and CA jointly generate a, b, c, S, T , but they are known only to node, not to CA.
- Substituting k_1 and k_2 with 1, TT is a Ferguson's e-coin with $k = 1$.
- It is hard for node to forge TT without knowing $1/v$.

4.2 Space Trading Protocol

When two nodes (U_1 and U_2) want to trade space for coupons, they execute the space trading protocol as in Figure 4. They both have new coupons after a successful

execution of the protocol. The new coupons can be used for further trade or redemption.

It takes three challenge-response steps between U_1 and U_2 to generate one evidence piece, and thus one new coupon for U_2 . It takes another three more steps to generate the other coupon in the opposite direction. First, U_1 sends its coupon P to U_2 . After verifying the validity of P , U_1 randomly selects a unique secret k and calculates the challenge x by hashing $K' = A^{k'}$, (W, r_ℓ) in P and its TT_1 identity (a, b, c) . In the last step, U_1 computes the response with its TT_2 and the secret k , which is used as the challenge to receive P , and sends them to U_2 . U_2 verifies the response, concatenates them to the vectors $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{r}, \mathbf{R}, \mathbf{K})$ in P and generates the new coupon P' .

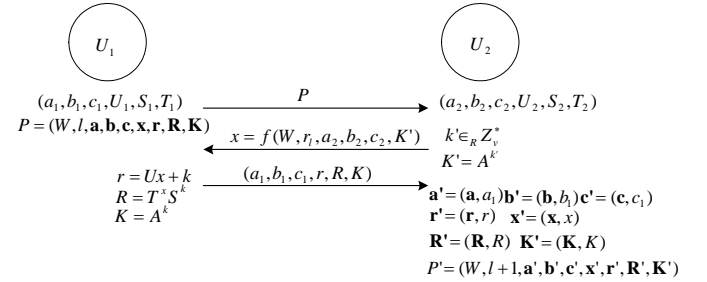


Figure 4. Details of space trading protocol.

The challenge-response binds TT_2 and secret k to P' , and thus P' can be traded once. The verification of the response includes verifying $x_\ell = f(W, r_{\ell-1}, a_1, b_1, c_1, K)$, which prevents U_1 using an other TT or secret to generate the evidence piece.

4.3 Space Access Protocol

A node U accesses the space in W via the storage access protocol. U first finds the location of the space – node U' via the underlying P2P network. They then execute the following authentication:

1. U selects a secret k' and calculates $K' = A^{k'}$, then presents K , which is used for receiving the coupon P , and P and its TT identity (a, b, c) of its transaction token to the U' .
2. U' verifies the validity of P and sends a challenge x to U .
3. U computes response $r = Ux + k'$ and $R = T^x S^{k'}$, based on its TT and the secret k' . U sends the response to U' .
4. U' verifies
 - I. $R^v = C^r B^x K'$
 - II. $x_\ell = f(W, r_\ell, a, b, c, K)$.

The verification I in step 4 proves node U has the legitimate TT, whose identity is (a, b, c) . The

verification II proves U is the owner P . U' will also check the double spending offense. If the length of coupon with the same W is changed, the file belonged to former owner in the space will be removed. The node U can manipulate files in the space. Since U selects a different k' in every authentication, the node identity U remains anonymous to U' .

5. CORRECTNESS

Our scheme is designed to work in P2P systems to prevent three major types of adversarial attacks [5]:

- **No collusion:** Nodes, acting on their own, wish to gain an unfair advantage over the network, but they have no peers with which to collude.
- **Minority collusion:** A subset of the P2P network is willing to form a conspiracy to lie about their resource usage.
- **Minority bribery:** The adversary may choose specific nodes to join the conspiracy, perhaps offering them a bribe in the form of unfairly increased resource usage.

No collusion and minority bribery are special cases of the minority collusion. The attack methods of minority collusion and minority bribery are the same, so our scheme focuses on the first two models and we assume the underlying P2P network is correct under these adversarial models.

5.1 Double Spending Detection

When a node spends the same coupon more than once to other nodes, it is called *double spending*. Conventionally, double spending is detected by the central authority. However, in a fair trading system, it is much more efficient to detect double spending at the redemption of coupons. When a host receives a redemption request of a coupon $P = (W, \ell, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{r}, \mathbf{R}, \mathbf{K})$, it checks the last coupon $P' = (W, \ell', \mathbf{a}', \mathbf{b}', \mathbf{c}', \mathbf{x}', \mathbf{r}', \mathbf{R}', \mathbf{K}')$ with the same W . If $P = P'$, then the coupon has not been traded since last trading. If this space has been traded, the new coupon associated with this space is generated in the finalized phase of the space trading protocol, which only increases the vectors of P' . Thus, if $P = P'$ or the vectors in P' are the prefix of the vectors in P , the coupon has not been doubly spent. Otherwise, the host executes the culprit identification protocol.

5.2 Culprit Identification Protocol

When the host finds two coupons P and P' have the same W and the vectors in P are not the prefixes of the vectors in P' and vice versa, the host needs to invoke the culprit identification protocol.

Let i be the smallest positive integer such that $(a_i, b_i, c_i) = (a'_i, b'_i, c'_i)$ and $x_i \neq x'_i$ (Figure 5). Our scheme closely follows that of the Ferguson's e-cash scheme, and thus every TT has a unique identity (a, b, c) , with a negligible probability of being replicated. Thus when $(a, b, c) = (a_i, b_i, c_i)$ there is only one TT that can generate them, and the node which withdraws the TT is the culprit.

In the space trading protocol, the node has to spend the coupon with the same TT and secret k , which are used as the challenge, to receive the coupon. In order to receive the coupon, the node has to send $x = f(W, r_\ell, a, b, c, K)$ to the previous owner to receive the new coupon. Assuming f is a suitable one-way function, it is hard for node to create another (a, b, c) or K which can meet $x = f(W, r_\ell, a, b, c, K)$.

By comparing $r_i = Ux_i + k_i$ and $r'_i = Ux'_i + k'_i$, we can get $k_i = k'_i$ and solve for U .

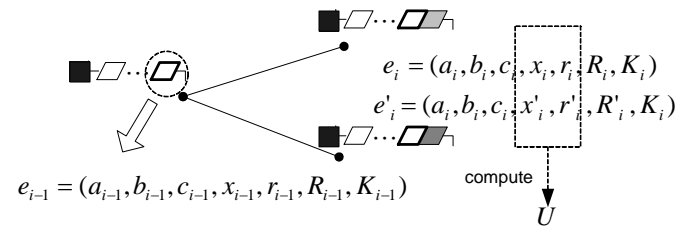


Figure 5. Double spending checking and culprit identification.

5.3 Security

If a node can forge a TT, it can use it to carry out trading without being caught. For the 1024-bit RSA implementation, it will be cost prohibitive to forge a TT. We consider the possibility to forge a valid coupon under different adversarial modes. To forge a valid coupon is equal to forge an evidence piece (a, b, c, x, r, R, K) , and append it to an existing valid coupon $P = (W, \ell, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{r}, \mathbf{R}, \mathbf{K})$ that satisfies the condition $x_i = f(W, r_{i-1}, K, a, b, c)$.

Given that x_ℓ is fixed, to forge a valid coupon, the node has to find (a, b, c, K, W) satisfying the one-way function $x_\ell = f(W, r_{\ell-1}, K, a, b, c)$. Forging a valid coupon is at least as hard as cracking the one-way function, *i.e.*, no conclusion. Next, we consider that the possibility that two nodes can work together to forge the evidence piece. If neither of them has a valid coupon, then it is as hard as in the case of the no collusion model to forge a valid coupon.

If a node U_1 has a valid coupon P and it wants to help U_2 to forge an evidence piece, then they need not crack the one-way function $x_\ell = f(W, r_{\ell-1}, K, a, b, c)$ since U_1

knows (W, K, a, b, c) . But they do need to find a 4-tuple (x, r, R, K) satisfying $R^v = C^r B^x K$. K is hard to forge after x is sent to get the evidence piece. x is fixed because it is calculated from r_t , which is calculated in the former trade protocol. If r is fixed, then the problem is reduced to find R satisfying $R^v = Y$, which is as hard as cracking the RSA signature scheme. The only way to forge an evidence piece is to calculate an r for a fixed R . In [10], Ferguson gave the proof that forging r is as hard as the RSA problem.

Now we consider the case that more than two nodes work together to forge a coupon. From the above discussion, only K can be calculated in the $R^v = C^r B^x K$ after the selection of R, r, x . There is a possibility for some nodes to perform a timed attack as the following steps:

1. U_3 sends x' , calculated from pre-select k , to U_2 .
2. U_2 calculates K' from x' after fixing r' and R' .
3. U_2 sends x , computed from K' , to another node U_1 , which has a valid coupon P .
4. U_1 returns an evidence piece (x, r, R, K) and concatenates the two evidence pieces with the original coupon to forge a coupon.

To prevent such an attack, our challenge x is calculated from the last response of the coupon, which means that x' cannot be pre-computed without evidence piece (x, r, R, K) . It also means that the evidence piece e' in the evidence chains can only be generated after the generation of the evidence piece e , which is before e' . If e is generated, then the scenario is same as the two nodes. Overall, to forge a coupon is as hard as RSA problem or to solve the one-way function.

5.4 Privacy

Except double spending, every secret k is only used once to generate $r = Ux + k$. Even having more (x, r) pairs cannot solve U by linear functions since the unknown factors are more than the number of functions, for example, two functions with three unknowns in (1).

$$\begin{cases} r_1 = Ux_1 + k_1 \\ r_2 = Ux_2 + k_2 \end{cases} \quad (1)$$

If a node does not double spend any of its coupons, its identity will not be revealed. In a round of challenge-response, a node U will send $r = Ux + k$ to the other node. If one were to deduct U , it would have to compute k from $K = A^{k \bmod v}$, which is as hard as RSA problem proved in [10].

6. SPACE SPLIT

The in-divisible scheme discussed above can guarantee fairness of space sharing, but every TT has to be signed by credential authority. Furthermore, from the user's perspective, a node only wants to trade for the minimal amount of space in each trade. In this section, we introduce a method to make the coupon divisible, which can reduce the interaction between the node and CA. We use the binary hash tree proposed by Otkmato [6] to achieve this goal. U_1 generates a binary hash tree and sends the root K as a challenge to another node to get the evidence piece. Then U_1 can split the new coupon.

The central idea in tracking the splitting of a coupon is to tie the root of a binary hash tree with the identity $(w = (a, b, c))$ of the TT. Each tree node is associated with a secret value, represents certain shares of coupon. If the node violates the double spending rules, one can compute its identity from the exposed secrets and the evidence pieces in the coupons.

A binary hash tree is constructed for t levels. Each node of the tree is represented by a secret value, called the k -value. The k -value of the root is denoted by k_0 . The left child and the right child of a node, which is denoted by k_j , are represented by k_{j_0} and k_{j_1} respectively, where the *resource index* j is a binary string. Given the suitable one-way function H , the k -values are assigned as follows:

Leaf node assignment:

$k_j = H(\varepsilon \parallel j)$, where ε is the random seed of this binary coupon tree, and “ \parallel ” stands for the concatenation of two entities.

Non-leaf node assignment:

$$k_j = H(H(A^{k_{j_0}}) \parallel H(A^{k_{j_1}}))$$

In the subsequent discussions, the root of the tree is referred to as $K_0 = A^{k_0}$, unless explicitly specified otherwise. Figure 6 demonstrates an example of the binary coupon tree construction.

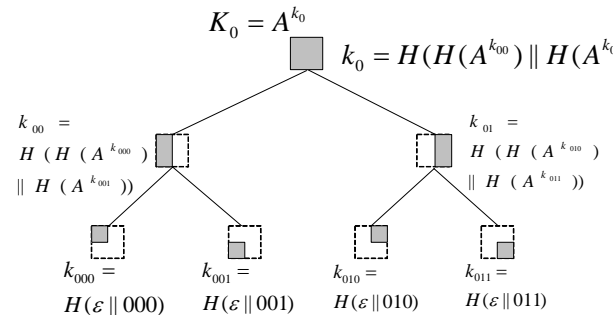


Figure 6. Binary tree construction for the splitting operation.

If a coupon can be divided into M shares, then a node at the i -th level represents the portion of $M / 2^{i-1}$ shares. To trade a portion of coupons at tree node k_j , the node reveals the “need-to-know information” to the other node to compute the tree root K_0 . The need-to-know information, which actually reveals the k -values of all ancestors of the node k_j , includes $K_j = A^{k_j}$ and $H(A^{k_j})$ if k_j is a direct offspring of an ancestor of the node k_j , but not on the route from the node k_j to the root node. Figure 7 illustrates an example to give the need-to-know information corresponding to the node k_{000} .

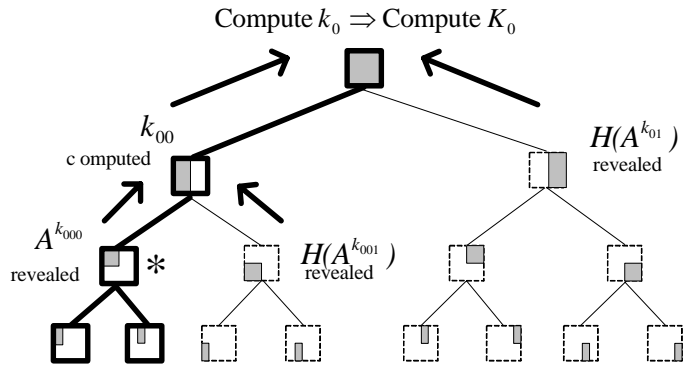


Figure 7. H-value revealed for spending.

For the coupon splitting operation, the evidence piece is generated as follows. U_1 first sends its coupon and tree root K_0 to U_2 . U_2 verifies the coupon is and generates a binary tree with root \bar{K}_0 (the level is determined how the node wants to split the receive coupon), then sends the consumer a challenge $x = f(W, a, b, c, r_t, \bar{K}_0)$. U_2 incorporates the k -value into the response. This k -value corresponds to the portion of shares that the U_2 intends to spend. U_2 sends the response $r = Ux + k$, $R = T^x S^k$ and the index j of the corresponding shares to U_1 . U_1 verifies that the response is valid and also verifies that the correct amount of shares by checking the number of level t to re-construct the root K_0 from the need-to-know information and appends the evidence piece (r, R, x) and the need-to-know information to the received coupon to generate the new coupon. The value of the new coupon is $1/2^t$ of the old coupon. Then, U_1 can trade the coupon in the same way, spending one of the nodes in the tree, whose root is \bar{K}_0 .

The coupon is doubly spent if the traverse path of transfers violates the *route node* or *same node rule* [6][26]. The route node rule states that if a node is used for delegation, then all ancestors and all descendants of this node cannot be used for delegation anymore. The

same node rule states that no node can be used for transfer more than once. When a node spends one node of the tree, it has to reveal all the secret k -values of the node’s ancestors. If the node trades its ancestor also (violates the route node rule), the identity of the node can be calculated because the k -value associated with the ancestor is revealed already. The detection and culprit identification of the same node rule is the same as the indivisible version (the same k -value is traded twice).

These two rules make sure the coupon cannot be spent more than once, but the total value is equal to the original coupon. *Double spending checking* in the redemption phase is enough to provide the fairness. The split operation can significantly reduce the interaction between the credential authority and the nodes, which makes the scheme scalable.

We need to include the need-to-know information in the evidence piece to change the in-divisible version to this divisible version and still maintain correctness of the scheme. The *double spending check* also needs a few modifications. The node should compare all coupons associated with the same W to check if anyone violates the same node rule or route node rule and identify the culprit if there is one. Since we can identify any culprit and every secret k -value will be used at most once, the security and privacy properties are as same as that in the in-divisible version.

7. CONCLUSION

In this paper we propose a fair resource-sharing scheme based on a modified e-coin algorithm. It inherits the security and privacy-protected properties of e-cash and our scheme adds transferability, re-usability and divisibility to the fair trading system. We also distribute the responsibility of double spending to all nodes, making our scheme suitable to solve the fair space sharing in large P2P computing systems. The scheme can easily be applied to sharing of heterogeneous resources.

REFERENCES

- [1] Napster: <http://www.napster.com/>.
- [2] The Gnutella protocol specification, 2000. <http://dss.clip2.com/GnutellaProtocol04.pdf>.
- [3] Kazza: <http://www.kazza.com/>.
- [4] E. Adar and B. Huberman. *Free Riding on Gnutella*. First Monday, 5(10), Oct, 2000.
- [5] T. Ngan, D. S. Wallach, and P. Druschel. *Enforcing Fair Sharing of Peer-to-Peer Resources*. In Proc. IPTS 03, Berkeley, Feb. 2003.
- [6] T. Okamoto and K. Ohta. *Universal Electronic Cash*. In Advances in Cryptology – CRYPTO’91, pp. 725-729, 1991.
- [7] T. Eng and T. Okamoto. *Single-Term Divisible Electronic Coins*. In Advances in Cryptography – Proceedings of EUROCRYPT ’94, LNCS 950, pp. 306 – 319, Springer-Verlag, 1995.
- [8] N. Ferguson. *Single Term Off-line Coins*. In Advances in Cryptology – Proceedings of EUROCRYPT ’93, LNCS 765, pp. 318 – 328, Springer Verlag, 1993.

- [9] N. Ferguson. *Single Term Off-line Coins*. Technical Report CS-R9318, CWI (Centre for Mathematics and Computer Science), Amsterdam, 1993.
- [10] N. Ferguson. *Extensions of Single-term Coins*. In *Advances in Cryptology – CRYPTO '93*, LNCS 773, pp 292 – 301, Springer Verlag, 1993.
- [11] A. Rowstron and P. Druschel *Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems*. *Lec. Notes. Comp. Sci.*, vol. 2218, pp. 329-- 350, 2001.
- [12] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. *Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications*. In *Proceedings of the 2001*.
- [13] B. Zhao, J. Kubiatowicz, and A. D. Joseph. *Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing*. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.
- [14] S. Ratnasamy, P. Fransis, M. Handley, R. Karp, and S. Shenker. *A Scalable Content-addressable Network*. In *Proceeding of ACM SIGCOMM 2001*, August 2001.
- [15] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. *Freenet: A Distributed Anonymous Information Storage and Retrieval System*. In *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, California, June 2000.
- [16] R. Dingledine. *The Free Haven Project: Design and Deployment of an Anonymous Secure Data Haven*. MIT Master's Thesis, June 2000.
- [17] Gnutella website. <http://gnutella.wego.com>.
- [18] M. Waldman and D. Mazieres. *Tangler: A Censorshipresistant Publishing System Based on Document Entanglements*. In *Proc. 8th ACM Conf. on Computer and Communications Security*, Nov. 2001.
- [19] P. Druschel and A. Rowstron. *Past: A Large-scale Persistent Peer-to-Peer Storage Utility*. In *Proc. of the Eighth IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, Schoss Elmau, Germany, May 2001.
- [20] R. Anderson. *The Eternity Service*. In *Proceedings of Pragocrypt 1996*.
- [21] H. Pagnia, and R. Jansen. *Towards Multiple-payment Schemes for Digital Money*. First International Conference, FC '97, Anguilla, British West Indies, 24-28 Feb.
- [22] H. Y. Wong. *Issues in Electronic Payment Systems: A New Off-line Transferable E-coin Scheme and a New Off-line E-check Scheme*. Master's Thesis, Dep. of Information Engineering, The Chinese University of Hong Kong (2001).
- [23] S. Brands. *Untraceable Off-line Cash in Wallet with Observers*. In *Advances in Cryptology – CRYPTO '93*, pp. 302 – 318, 1993.
- [24] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, 1998.
- [25] A. Barmouta and R. Buyya. *GridBank: A Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration*. 17th Annual International Parallel & Distributed Processing Symposium (IPDPS 2003) Workshop on Internet Computing and E-Commerce, 2003.
- [26] T. C. Lam and J. C. Liu. *On the Evidence Based Peer-to-Peer Resource Management in Distributed Computing System*. Technical Report 2003-7-2, Department of Computer Science, Texas A & M University, 2003.