A Flexible Payment Scheme and Its Role-Based Access Control

Hua Wang, Jinli Cao, *Member*, *IEEE Computer Society*, and Yanchun Zhang

Abstract—This paper proposes a practical payment protocol with scalable anonymity for Internet purchases, and analyzes its role-based access control (RBAC). The protocol uses electronic cash for payment transactions. It is an offline payment scheme that can prevent a consumer from spending a coin more than once. Consumers can improve anonymity if they are worried about disclosure of their identities to banks. An agent provides high anonymity through the issue of a certification. The agent certifies reencrypted data after verifying the validity of the content from consumers, but with no private information of the consumers required. With this new method, each consumer can get the required anonymity level, depending on the available time, computation, and cost. We use RBAC to manage the new payment scheme and improve its integrity. With RBAC, each user may be assigned one or more roles, and each role can be assigned one or more privileges that are permitted to users in that role. To reduce conflicts of different roles and decrease complexities of administration, duty separation constraints, role hierarchies, and scenarios of end-users are analyzed.

Index Terms—Electronic-cash, anonymity, integrity, trace ability, hash function.

1 Introduction

Recent advances in the Internet and WWW have enabled rapid development in e-commerce. More and more businesses have begun to develop or adopt e-commerce systems to support their selling/business activities. While this brings convenience for both consumers and vendors, many consumers have concerns about security and their private information when purchasing over the Internet, especially with electronic payment or e-cash payment. Consumers often prefer to have some degree of anonymity when shopping over the Internet. There are a number of proposals for electronic cash systems [1], [2], [3]. Probably, it is accurate to say that most of them lack flexibility in anonymity. Chaum [1], for example, first proposed an online payment system that guarantees receiving valid coins. This system provides some anonymity against a collaboration between shops and banks. However, consumers have no flexible anonymity and banks have to keep a very large database for consumers and coins.

The systems mentioned above are online payment systems. They need sophisticated cryptographic functions for each coin, and require additional computational resources for banks to validate purchases. Forcing banks to be online at payment is a very strict requirement. Online payment systems protect merchants and banks against customer fraud since every payment needs to be approved by the customer's banks. This increases the computation cost, proportional to the size of the database of spent coins. If a large number of people start using a payment system, the size of this database could become very large and unmanageable. Keeping a database of all the coins ever spent in the system is not a scalable solution. Digicash [4] plans to use multiple banks, each minting and managing their own currency, with interbank clearing to handle the problems of scalability. It seems likely that the host bank machine has an internal scalable structure so that it can be set up not only for a 10,000 consumer bank, but also for a 1,000,000 consumer bank. Under the current circumstances, the task of maintaining and querying a database of spent coins is probably beyond today's state-of-the-art database systems [5].

In an offline protocol, merchants verify payments using cryptographic techniques, and commit payments to payments authority later, in an offline batch process. Offline payment systems were designed to lower the cost of transactions due to the delay in verifying batch processes. However, they suffer from the potential of double spending, whereby the electronic currency might be duplicated and spent repeatedly.

The first offline anonymous electronic cash was introduced by Chaum et al. [24]. The security of their scheme relied on some restricted assumptions, such as requiring a function which is similar to random oracle, and has to map onto a special range. There is also no formal proof attempted. Although hardly practical, their system demonstrated how offline e-cash can be constructed and laid the foundation for more secure and efficient schemes. In 1995, Chan et al. [7] presented a provable secure offline e-cash scheme that relied only on the security of RSA [8]. This scheme extended the work of Franklin and Yung [9] who aimed to achieve provable security without the use of general computation protocols. The anonymity of consumers is based on the security of RSA and it cannot be

- *H. Wang is with the University of Southern Queensland, Toowoomba, QLD 4350, Australia. E-mail: wang@usq.edu.au.*
- *J. Cao is with the Department of Computer Science and Computer Engineering, La Trobe University, Bundoora, Melbourne Vic. 3083, Australia. E-mail: jinli@cs.latrobe.edu.au..*
- *Y. Zhang is with the Internet Technologies and Application Research Lab, School of Computer Science and Mathematics, Victoria University, Ballarat Road, Footscray PO Box 14428, Melbourne City MC VIC 8001, Australia. E-mail: yzhang@csm.vu.edu.au.*

Manuscript received 14 May 2004; accepted 4 Aug. 2004; published online 19 Jan. 2005.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0131-0504.

changed dynamically after the system is established. NetCents [10] proposed a lightweight, flexible, and secure protocol for micro payments of electronic commerce over the Internet. This protocol is designed only to support purchases ranging in value from a fraction of a penny and up. In 2000, Pointcheval [11] presented a payment scheme in which the consumer's identity can be found any time by a certification authority. So, the privacy of a consumer cannot be protected.

Furthermore, the issue of privilege management across multiple domains poses a number of additional challenges. First, there is a need to represent different types of privileges for different services in a purchase process. Our payment scheme is able to support a range of access policies, including role-based access control (RBAC). RBAC has gained much popularity in access control, though the idea of partitioning privileges in terms of specific job functions and roles has been well known for several decades. Second, there is a need to verify the correctness and the validity of the privileges at the time of access to a service. The security management model should be flexible enough to specify a range of privileges and to evaluate them to make appropriate decisions in an efficient manner.

To demonstrate the importance of management policies, consider the unconstrained actions of Nicholas Leeson, who led to the bankruptcy of England's oldest investment firm in 1995 [12]. Leeson was granted the right to do both the financial derivatives trading operation and back office functions where trades were settled. This is a mix of roles that can be dangerous. In any firm serious about preventing fraud in its operations, this arrangement is conflicting. Management at Barings PLC should never have had the same person making and settling trades. Such a conflict of interest policy can be specified centrally by management, and enforced effectively using RBAC technology.

Role-based access control (RBAC) has been widely used in database system management and operating system products [13], [14], [15]. Many RBAC practical applications have been implemented [16], [17], [18] because RBAC has many advantages such as reducing administration cost and complexity. Another use of RBAC is to support integrity. Integrity is a requirement that data and processes be modified only in authorized ways by authorized users. Within a role-based system, the principal concern is protecting the integrity of information: who can perform what acts on what information. However, there has been little research done on the usage of RBAC in payment scheme management [19].

As mentioned above, online e-cash payments need more computing resources. Most of the previously designed offline schemes are only for micro payments. They rely on the heuristic proofs of security and, therefore, do not formally prevent fraud and counterfeit money. Under these conditions, most online and offline payment schemes do not provide efficient anonymity for consumers. Hence, a new payment scheme for the purchases over the Internet with the following properties are very useful and important:

1. Offline: It spends less computational resources and reduces network communication.
2. Flexible anonymity: It can provide high-level anonymity for some consumers with high security requirements.
3. Untrace ability: No one can trace consumers from cash. However, the identity of a consumer can be found if he/she uses a coin twice.
4. RBAC management: It can decrease the complexities of a system and improve system security such as integrity.

In this paper, we analyze the electronic-payment model first, then propose a new offline electronic cash scheme in which the anonymity of consumers is scalable. Consumers can get the required anonymity without showing their identities to any third party. Furthermore, to reduce administration cost and complexity and to improve the security of management, we analyze how to use RBAC to manage the new payment scheme.

The paper is organized as follows: In the first two sections, some basic definitions and simple examples are reviewed. The payment model and the anonymity provider agent are described in Section 3. The design of a new offline electronic cash scheme and its security analysis, including a logical proof for security, are detailed in Section 4. The RBAC management of the new scheme such as the relationships of roles and end-user use scenarios is presented in Section 5. An example of the new e-cash scheme and how to use it for Internet purchases are given in Section 6. Conclusions are included in Section 7.

2 Some Basic Definitions

2.1 Hash Functions

$H(x)$ is a hash function. For a given value W, it is computationally hard to find an x such that $H(x) = W$, i.e., collisions are hard to find, where x might be a vector.

Hash function is a major building block for several cryptographic protocols, including pseudorandom generators [20], digital signatures [21], and message authentication.

2.2 DLA and ElGamal Encryption System

Discrete Logarithm Assumption (DLA) is an assumption that the discrete logarithm problem is believed to be difficult.

The discrete logarithm problem is as follows: Given an element g in a group G of order q, and another element y of G, find x, where $0 < x < q - 1$, such that y is the result of multiplying g with itself x times, i.e., $y = g^x$. In some groups, there exist elements that can generate all the elements of G by exponentiation (i.e., applying the group operation repeatedly) with all the integers from 0 to $q - 1$. When this occurs, the element is called a generator and the group is called cyclic. Rivest [22] has analyzed the expected time to solve the discrete logarithm problem both in terms of computing power and cost and shown that it is computationally hard to get y from x.

For this reason, it has been used for the basis of several public-key cryptosystems, including the famous ElGamal encryption system. The ElGamal encryption system [23] is a public key encryption scheme which provides semantic security. Let us briefly recall it.

TABLE 1
ElGamal Encryptiom Scheme

Step 1. The system needs a group G of order q, and a generator g. The secret key is an element $sk \in Z_p = \{0, 1, ..., q-1\}$ and the public key is $pk = g^{sk}$. Step 2. For any message $m \in G$, the cipher text of m is $c = (g^r, pk^r m)$, for a random $r \in Z_p - \{0\}$. Step 3. For any cipher text $c = (a, b)$, the message m can be retrieved by $m = b/a^{sk}$.

Table 1 indicates that the message m can be obtained only by the person who has the secret key sk.

2.3 Undeniable Signature Scheme and Schnorr Signature Scheme

The undeniable signature scheme, devised by Chaum and Antwerpen [6], is a non-self-authenticating signature scheme, where signatures can only be verified with the signer's consent. However, if a signature is only verifiable with the aid of a signer, a dishonest signer may refuse to authenticate a genuine document. Undeniable signatures solve this problem by adding a new component, called the disavowal protocol, in addition to the normal components of signature and verification.

An undeniable proof scheme consists of the following algorithms:

1. The key generation algorithm K which outputs random pairs of secret and public keys (sk, pk).
2. The proof algorithm $P(sk, m)$ which inputs a message m, returns an "undeniable signature" S on m. However, this proof "S" does not convince anybody by itself. To be convinced of the validity of the pair (m, S), relative to the public key pk, one has to interact with the owner of the secret key sk.
3. The confirmation process confirms (sk, pk, m, S), which is an interactive protocol between the signer and the verifier, where the signer tries to convince of the validity of the pair (m, S).
4. The disavowal process is an interactive protocol between the signer and the verifier, where the signer can prove that a forged signature really is a forgery.

We simply recall the following example shown in Table 2 that is an undeniable signature scheme [25]. The disavowal process is detailed below.

TABLE 2
Schnorr Signature Scheme

The system needs primes p and q such that q is divided by $(p-1)$, i.e. $q\|(p-1)$, $g \in Z_p$ with order q, i.e. $g^q = 1(mod\ p), g \neq 1$. A consumer generates by himself a private key sk which is a random number in Z_q. The corresponding public key pk is the number $pk = g^{-sk}(mod\ p)$.
To sign message m with the private key sk the consumer performs the following steps: 1. Computes $x = g^r(mod\ p)$, where $r \in Z_q$ is a random number. 2. Computes $R = H(x, m)$, where H is a hash function. 3. Computes $y = r + sk * R(mod\ q)$ and output the signature $S = (R, y)$.
To verify the signature $S = (R, y)$ for message m with the public key pk a verifier computes $\overline{x} = g^y pk^R(mod\ p)$ and checks $R = h(\overline{x}, m)$.

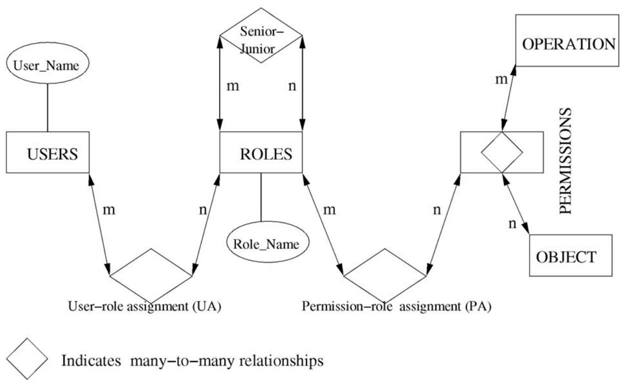
User_Name
Senior–Junior
OPERATION
m
n
m
USERS
ROLES
PERMISSIONS
m
n
m
n
n
Role_Name
OBJECT
User–role assignment (UA)
Permission–role assignment (PA)
Indicates many–to–many relationships

Fig. 1. RBAC relationship.

The signer is not able to deny the signature $S = (e, y)$ because the signature can be produced by the owner of the secret key sk only. The signature $S = (e, y)$ is confirmed only if $\overline{x} = g^y pk^e(mod\ p)$ and checks $e = h(\overline{x}, m)$ with the public key pk. In the disavowal process, the verifier chooses a random number R_1 and sends to the signer. The latter picks a random r_1, computes $h(x_1), y_1 = r_1 + S^* R_1$, where $x_1 = g^{r_1}$, sends $h(x_1)$ and y_1 to the verifier, the verifier computes $\overline{x} = g^{y_1} pk^{R_1}$ and checks if $h(x_1) = h(\overline{x})$.

2.4 Role-Based Access Control

RBAC is described in terms of individual users being associated with roles as well as roles being associated with permissions (each permission is a pair of objects and operations). As such, a role is used to associate with users and permissions. A user in this model is a human being. A role is a job function or job title within the organization associated with the authority and responsibility.

Permission is an approval of a particular operation to be performed on one or more objects. The relationship between roles and permissions is shown in Fig. 1, arrows indicate a many-to-many relationship (i.e., a permission can be associated with one or more roles and a role can be associated with one or more permissions). The RBAC security model has two components: MC_0 and MC_1. Model component MC_0, called the RBAC authorization database model, defines the RBAC security properties for authorization of static roles. Static properties of an RBAC authorization database include role hierarchy, inheritance, cardinality, and static separation of duty. MC_1, called the RBAC activation model, defines the RBAC security properties for dynamic activation of roles. Dynamic properties include role activation, permission execution, dynamic separation of duties, and object access. A *session* is a mapping of one user to possible roles. In other words, a user establishes a session during which the user activates some subset of roles that he/she is a member of. In particular, the RBAC model supports the specification in Table 3.

We note that properties b and c have to be decided when a system is designed. This is because the relationship of different roles may be in conflict, compromising the integrity of the system. Properties b and c can decrease the complexities of management and reduce the conflicts of

TABLE 3
Specification of RBAC

a. User/role associations; the constraints specifying user authorizations to perform roles,
b. Role hierarchies; the constraints specifying which role may inherit all of the permissions of another role,
c. Duty separation constraints; these are role/role associations indicating conflict of interest: c1. Static separated duty (SSD); a constraint specifying that a user cannot be authorized for two different roles, e.g. auditor and account representative in a bank; c2. Dynamic separated duty (DSD); a constraint specifying that a user can be authorized for two different roles but cannot act simultaneously in both, e.g. auditor and seller in a supermarket;
d. Cardinality; the maximum number of users allowed, i.e. how many users can be authorized for any particular role (role cardinality), e.g., only one manager.

different roles. Therefore, this paper focuses on relationships of roles such as Role hierarchies, SSD, and DSD.

3 New Payment Model

We show the basic payment model and then discuss the new payment model in this section.

3.1 Basic Payment Model

Electronic cash has sparked wide interest among cryptographers ([22], [26], [27], etc.). In its simplest form, an e-cash system consists of three parts (a bank, a consumer, and a shop) and three main procedures, as shown in Fig. 2 (withdrawal, payment, and deposit). In a coin's life-cycle, the consumer first performs an account establishment protocol to open an account with the bank.

The consumer and the shop maintain an account with the bank, while

1. The consumer withdraws electronic coins from his account, by performing a withdrawal protocol with the bank over an authenticated channel.
2. The consumer spends a coin by participating in a payment protocol with a shop over an anonymous channel.
3. The shop performs a deposit protocol with the bank, to deposit the consumer's coin into his account.

The system is *offline* if the shop does not communicate with the bank during payment. It is *untraceable* if there is no p.p.t. TM (probabilistic polynomial-time Turing Machine) that can identify a coin's origin even if one has all the information of withdrawal, payment, and deposit transactions. It is *anonymous* if the bank, in collaboration with the shop, cannot trace the coin to the consumer. However, in the absence of tamperproof hardware, electronic coins can be copied and spent multiple times by the consumer. This has been traditionally referred to as double-spending. In an online e-cash system, double-spending is prevented by having banks check if a coin has been deposited before. In an offline e-cash system, however, this solution is not possible; instead, as proposed by Chaum et al. [24], the system guarantees that, if a coin is double-spent, the consumer's identity is revealed with overwhelming probability.

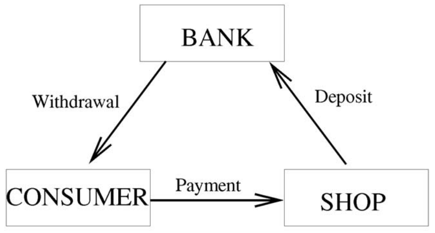

Fig. 2. Basic processes of an electronic cash system.

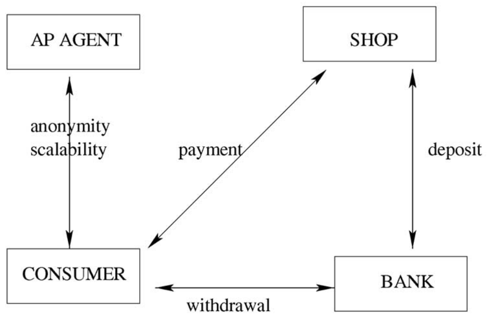

Fig. 3. New electronic cash model.

There are also three additional processes, such as the bank setup, the shop setup, and the consumer setup (account opening). They describe the system initialization, namely, creation and posting of public keys and opening of bank accounts. Although they are certainly part of a complete system, these are often omitted as their functionalities can be easily inferred from the description of the three main procedures. For clarity, we only describe the bank setup and the consumer setup for the new scheme in the next section because the shop setup is similar to the consumer setup.

Besides the basic participants, a third party, the named Anonymity Provider (AP) agent, is involved in the scheme. The AP agent helps the consumer to get the required anonymity, but is not involved in the purchase process. The new model is shown in Fig. 3. The AP agent gives a certificate to the consumer when he/she needs a high level of anonymity.

3.2 Anonymity Provider Agent

Here, we explain what an AP agent is. Assume a consumer owns a valid coin $c = \varphi(pk_B, pk_u, y)$ with its certificate $Cert_c$, where $\varphi(pk_B, pk_u, y)$ is a function of the public keys of the bank, the consumer, and a variable y, i.e., (pk_B, pk_u, y). The certificate $Cert_c$ guarantees correct withdrawal from a bank, whether a coin is valid or not depends on its certificate. $Cert_c$ is a coin certificate and does not contain any information about a customer. After the following processes with the AP agent, the consumer owns a new valid coin $c' = \varphi(pk_B, pk_u, y + v)$ with its certificate $Cert_{c'}$, where v is a variable. Please note that a certificate can be a signature of a issuer:

$$\{Value : IssuerName : IssuerSerial : IssuerNumber \\ : Time\}.$$

TABLE 4
Proof of Validity of a Coin $c = Y^r I^s$

1. Consumers choose a random $k \in Z_p$, then compute $t = Y^k g^s (mod\ p)$ and $e = H(m,t)$ where m is a mixed message of c, current time etc,
2. Then compute $u = k - re$, $v = s - x_u e$, and $t_1 = g^{(s-1)x_u e} (mod\ p)$,
3. The signature finally consists of (e, u, v, t_1), the signature and (m,t) are sent to a verifier,
4. In order to verify it, one has just to compute $t' = Y^u g^v b^e (mod\ p)$ and check whether $t' = tt_1 (mod\ p)$ and $e = H(m, t'/t_1)$.

Therefore, certificates made by a bank and an AP agent are different and do not include a customer's identity.

1. The consumer reencrypts the coin c into
$$c' = \varphi(pk_B, pk_u, y + v).$$
2. The consumer provides an undeniable signature S, and the equivalence between c and c'. This equivalence is guaranteed by the variable v.
3. The consumer confirms the validity of this signature S to the AP agent.
4. The AP agent certifies the new coin c' and sends $Cert_{c'}$ to the consumer.

Indeed, after Steps 2 and 3, the AP is convinced that the conversion has been performed by the owner of the coin c; c' is equivalent to c. The owner of c is not able to deny S (the relation between c and c'). The AP agent should be an electronic notarized participant in the system. It does not need to know any private information about consumers; it only verifies the information of consumers.

3.3 Proof of Ownership of a Coin

This section shows how consumers prove the ownership of a coin. Let us assume that Y is the public key of a bank, x_u is a secret key of a consumer, and $I = g^{x_u}$ is the identity of a consumer. $H(x,y)$ is a hash function. A coin is the encryption of I: $c = (a = g^r, b = Y^r I^s)$, which is afterward certified by the bank, where r, s are random numbers. With the certificate of the bank, one knows that the encryption is valid. Therefore, in order to prove his ownership, the consumer has just to convince of his knowledge of (x_u, r, s) such that $b = Y^r I^s$. This can be expressed as in Table 4.

Then, a scrambled coin is gotten by multiplying both parts of the old one by the respective bases, g and Y, both to the same random exponent ρ:

$$c' = (a' = g^\rho a, b' = Y^\rho b) = (g^{r+\rho}, Y^{r+\rho} I^s).$$

Then, if the owner of the old coin has certified the message $m' = h^\rho$, the equivalence of both coins can be proven with the proof of equivalence of three discrete logarithms:

$$log_h m' = log_g(a'/a) = log_Y(b'/b),$$

where h is a public variable.

4 Anonymity Scalable Payment Scheme

In this section, we propose an anonymity self-scalable payment scheme. The new payment scheme has two main features. The first is that a consumer can have a high level of anonymity himself and the second is that the identity of a consumer cannot be traced unless the consumer spends the same coin twice.

Our scheme includes two basic processes in system initialization (bank setup and consumer setup) and three main protocols: a new withdrawal protocol with which a consumer withdraws electronic coins from a bank while his account is debited, a new payment protocol with which a consumer pays the coin to a shop, and a new deposit protocol with which the shop deposits the coin to the bank and has his account credited. If a consumer wants to get a high level of anonymity after getting a coin from the bank (withdrawal), he/she can contact the AP agent.

4.1 System Initialization

The bank setup and the consumer setup are described as follows, and the details of the shop setup are omitted (because the shop setup is similar to the consumer setup).

4.1.1 Bank Setup (Performed Once by the Bank)

Primes p and q are chosen such that $|p-1| = \delta + k$, for a specified constant δ, and $p = \gamma q + 1$, for a specified small integer γ. Then, a unique subgroup G_q of prime order q of the multiplicative group Z_p and generator g of G_q are defined. Secret key $x_B \in_R Z_q$ for a denomination is created, where $a \in_R A$ means that the element a is selected randomly from the set A with uniform distribution. Hash function H from a family of collision intractable hash functions is also defined. The bank publishes p, q, g, H, and its public key $Y = g^{x_B} (mod\ p)$.

The secret key x_B is safe under the DLA. The hash function is used in payment transactions.

4.1.2 Consumer Setup (Performed for Each Consumer)

The bank associates the consumer with $I = g^{x_u} (mod\ p)$, where $x_u \in G_q$ is the secret key of the consumer and is generated by the consumer.

After the consumer's account and the shop's account are opened, the new payment scheme can be described.

4.2 New Offline Payment Scheme

We now describe the new anonymity scalable electronic cash scheme, which includes withdrawal, payment, and deposit.

4.2.1 Withdrawal (Consumers Withdraw Coins from the Bank)

Usually, an anonymous coin is a certified message which embeds the public key of a consumer. In our scheme, the message is an encryption of this consumer's public key, using the public key Y of the bank.

Instead of using intricate zero-knowledge proofs to convince the bank of the validity of the encryption, the consumer shows some information to the bank, including a signature. So, the bank certifies the encryption with full confidence.

The consumer $I = g^{x_u}$ constructs a coin $c = (a = g^r, b = Y^r I^s)$ using the public key Y of the bank, where s is a secret key of the coin which is kept by the consumer and r is a random number in Z_q. Using the private key x_u, the consumer signs a Schnorr signature S on the message of c together with the date, etc. He/she sends (c, S) to the bank together with r, I. Then, the bank can check the correct encryption. With the signature of the coin and the date, only the legitimate consumer could have done it. After having modified the consumer's account, the bank sends back a certificate $Cert_c$. The consumer needs to remember $(r, s, Cert_c)$ only.

4.2.2 Anonymity Scalability (Performed between Consumers and the AP Agent)

The consumer can use the coin now without a high level of anonymity since the bank can easily trace any transaction performed through the coin. This is because some information of the consumer such as $I, Cert_c$ has been known by the bank. To solve this problem, an AP agent is established to help the consumer to achieve a high level of anonymity: The consumer can derive a new encryption of his identity in an indistinguishable way. However, the consumer needs a new certificate for a new issued cipher text. The AP agent can provide this new certificate. Before certifying, the consumer requires both the previous coin $(c, Cert_c)$ and the proof of equivalence between the two cipher texts. Details are described below.

The consumer contacts the AP agent if he/she needs to get a high level of anonymity. The consumer chooses a random ρ and reencrypts the coin:

$$c' = (a' = g^{\rho} a, b' = Y^{\rho} b).$$

1. The consumer generates a Schnorr signature S on $m = h^{\rho}$ using the secret key x_u, as shown in Section 2.3. Because of S, the consumer is not able to deny his knowledge of ρ later. Furthermore, nobody can impersonate the consumer at this step since the discrete logarithm x_u of I is required to produce a valid signature. So, there is no existential forgery.
2. The consumer also provides a designated verifier proof of equality of discrete logarithms

$$log_h m = log_g(a'/a) = log_Y(b'/b). \qquad (1)$$

3. The consumer finally sends c, c', S, m to the AP agent.
4. The AP agent checks the certificate $Cert_c$ on c, the validity of the signature S on the message m, then certifies c', and sends back a certificate $Cert_{c'}$ to the consumer.

After these processes, the consumer gets a new certified coin $c' = (a' = g^{\rho} a, b' = Y^{\rho} b)$ and a new certification $Cert_{c'}$, which is now strongly anonymous from the point of view of the bank. The AP agent has to keep (c, c', m, S) to be able to prove the link between c and c', with the help of the consumer. If the number of consumers gets large, databases for the AP agent and banks is huge, data back up is required. We may backup data once a month or once every two weeks to support the system.

Following the process, the AP agent can also give many smaller new coins for an old one since the amount of a new one can be embedded in the certificate $Cert_{c'}$.

4.2.3 Payment (Performed between the Consumer and the Shop over an Anonymous Channel)

When a consumer possesses a coin, he/she can simply spend it at shops by proving knowledge of the secret key (x_u, s) associated with the coin c or c'. This proof is a signature $S = (e, u, v, t_1)$ of the new certificate $Cert_{c'}$, purchase, date, etc., with the secret key (x_u, s) associating the coin to the receiver (which is later forwarded to the bank).

4.2.4 Deposit (The Receiver Deposits a Coin to a Bank)

Since the system is offline, the shop sends the payment transcript to the bank later. The transcript consists of the coin c or c' (if the consumer applied a high level of anonymity), the signature, and the date/time of the transaction. The bank verifies the correctness of payment and credits the coin into the shop's account.

The receiver (shop) deposits the coin into its bank account with a transcript of the payment. If the consumer uses the same coin c twice, then the consumer is traced: Two different receivers send the same coin c to the bank. The bank can easily search its records to ensure that c has not been used before. If the consumer uses c twice, then the bank has two different signatures. Thus, the bank can isolate the consumer and trace the payment to the consumer's account I.

4.3 Security Analysis

Based on the point of view with electronic cash system, an offline e-cash scheme is secure [9] if the following requirements are satisfied:

1. *Unreusable*: If any consumer uses the same coin twice, the identity of the consumer can be computed.
2. *Untraceable*: With n withdrawal processes, no p.p.t. (probabilistic polynomial time) Turing Machine can compute $(n+1)\mathrm{th}$ distinct and valid coin.
3. *Unforgeable*: With any number of the customer's withdrawal, payment, and deposit protocols, no p.p.t. Turing Machine can compute a single valid coin.
4. *Unexpandable*: With any number of the customer's valid withdrawal, payment, and deposit protocols, no p.p.t. Turing Machine can compute a legal consumer's identity.

The security in the e-cash scheme is based on the hardness of Discrete Logarithms [28] and hash functions. The system preserves the above four requirements.

4.3.1 Unreusable

When a consumer spends a coin, he/she hands over the coin, together with a signature $S = (e, u, v, t_1)$, to a shop. If the consumer uses a coin twice, then we have two signatures $S_1 = (e_1, u_1, v_1, t_{11})$ and $S_2 = (e_2, u_2, v_2, t_{12})$, where

$$u_1 = k_1 - re_1 (mod\ q),\ v_1 = s - x_u e_1 (mod\ q).$$

$$u_2 = k_2 - re_2 (mod\ q),\ v_2 = s - x_u e_2 (mod\ q).$$

Then, $(v_2 - v_1)/(e_1 - e_2) = x_u$; this is the secret key of the consumer I. So, a coin in the new scheme cannot be reused.

4.3.2 Untraceable

When a consumer constructs a coin, he/she uses the secret keys x_u and s, both of which are not shown to any other parts in the purchase process. So, no one can trace the consumer and the coin.

4.3.3 Unforgeable

We first discuss whether the bank and the AP agent can forge a valid coin or not. To produce a valid coin, the first requirement is making a encryption $c = (a = g^r, b = Y^r I^s)$ of I. The second requirement is using the secret key x_u of the consumer to sign a Schnorr signature of c together with the current time. The bank can do the first step, but cannot do the second step since it does not know the secret key x_u. This means the bank cannot forge a valid coin. Similarly, the AP agent cannot forge a valid coin either. It should be noted that, even though both the bank and the AP agent know a valid coin, they still could not use it. This is because there is a signature in the payment process which can only be done by the consumer.

As already seen, the secret key x_u of a consumer is never revealed, only used in some signatures. A consumer is therefore protected against any impersonation, even from collusion of the bank, the AP agent, and the shop. Only the consumer can construct a valid coin since there is an undeniable signature embedded in the coin. To prevent the bank from framing the consumer as a multiple spender in the scheme, we use digital signature I^s for s which is known only by the consumer. Then, the system is unforgeable.

4.3.4 Unexpandable

For a legal consumer and a valid coin, the secret key x_u and the random number s are never shown to others at any time. Furthermore, the random number s is changed for different coins. With n withdrawal proceedings, the random number s is changed n times. Then, no one can compute the $(n+1)\text{th}$ distinct and valid coin even if they see n procedure withdrawals.

The AP cannot cheat shops and banks. For example, AP knows c, c', S, m from a consumer but not the secret key (x_u, s). In the payment process, a signature is needed when the coin c or c' is used. The secret key (x_u, s) is used to produce the undeniable signature. It means that the AP cannot use the old coin c to the new coin c' and, thus, the AP cannot cheat shops.

4.4 Logical Proof for Security

We have demonstrated that the e-cash system satisfies secure requirements. This section examines the system issues and use stages using logical proof [29]. Logical proof allows analysis of trust between principals involved in authentication. The e-cash approach is about who has rights to access and not about authentication, hence the analysis is different from analysis of authentication protocols. The goal is not to prove an identity of principals but to prove that a user has been granted privilege to use a coin. The use of logical proof requires the transformation of the e-cash approach into an idealized protocol and then refinement until the required trust is obtained. The idealized protocol associated to a coin for logical proof is as below:

- *M1*: $Bank \rightarrow Consumer : I$.
- *M2*: $Consumer \rightarrow Bank : (c, S_1, r, I)$.
- *M3*: $Bank \rightarrow Consumer : Cert_c$.
- *M4**: $Consumer \rightarrow AP : (c, c', S_2, m)$.
- *M5*: $AP \rightarrow Consumer : Cert_{c'}$.
- *M6*: $Consumer \rightarrow Shop : S_3 = (e, u, v, t_l)$.

We have the following assumptions with PKI technology, where K_{BC}, K_{AC}, and K_{SC} means shared keys between Bank and Consumer, AP agent and Consumer, and Consumer and Shop, respectively.

- *A1*: Bank $\rightarrow$ Consumer: Bank $\overset{K_{BC}}{\longleftrightarrow}$ Consumer ($A1$ is used in $M3$).
- *A2*: AP agent believes: AP agent $\overset{K_{AC}}{\longleftrightarrow}$ Consumer.
- *A3*: Consumer believes $Cert_c$ and $Cert_{c'}$ are fresh.
- *A4*: Consumer believes: Consumer $\overset{K_{SC}}{\longleftrightarrow}$ Shop.
- *A5*: Bank believes c and S_l are fresh.
- *A6*: AP agent believes c' and S_2 are fresh.
- *A7*: Shop believes S_3 is fresh.

To verify the approach, we need to reach the following states:

- *P1*: Bank believes Consumer believes (c, S_1, r, I).
- *P2*: AP agent believes Consumer believes
$$(c, c', S_2, m).$$
- *P3*: Consumer believes $Cert_c$ and $Cert_{c'}$.
- *P4*: Shop believes Consumer believes $S_3 = (e, u, v, t_l)$.

$P1, P2, P3$ relate to the e-cash creation process of the system and the other one relates to payment processes. We prove these states as below.

Proof. After $M1$ and $M2$:

- *R1*: Bank sees (c, S_1, r, I).
  Since S_1 is an undeniable signature of the Consumer, from $R1$, and the *message meaning rule* [29].
- *R2*: Bank believes the Consumer once said (c, S_1, r, I).
  Since Bank believes c, S_l are fresh ($A5$) and using the *nonce verification rule* [29].
- *R3*: Bank believes Consumer believes (c, S_1, r, I).
  ($P1$) has been proven.
  After $M4^*$,
- *R4*: AP agent sees (c, c', S_2, m).
  Since S_2 is a Schnorr signature of Consumer, from $R4$ and $A2$ and the *message meaning rule*.
- *R5*: AP agent believes Consumer once said (c, c', S_2, m).
  Since AP agent believes c', S_2 are fresh ($A6$) and using the *nonce verification rule*.
- *R6*: AP agent believes Consumer believes (c, c', S_2, m).
  ($P2$) has been proven.
  After $M3$ and $M5^*$,
- *R7*: Consumer sees $Cert_c$ and $Cert_{c'}$, from *R7* and *A1*, *A2*, and the *message meaning rule*.

- *R8*: Consumer believes Bank and AP agent once said $Cert_c$ and $Cert_{c'}$.
  Since Consumer believes, $Cert_c$ and $Cert_{c'}$ are fresh. Using $M3$ and the *nonce verification rule*.
- *R9*: Consumer believes Bank and AP agent believe $Cert_c$ and $Cert_{c'}$.
  From $M3, M5^*$,
- *R10*: Consumer believes Bank and AP agent control $Cert_c$ and $Cert_{c'}$.
  From $R9$, $R10$, and the *jurisdiction rule* [29],
- *R11*: Consumer believes $Cert_c$ and $Cert_{c'}$.
  $P3$ has been proven.
  After $M6$, Shop sees S_3. From $A4$ and the *message meaning rule*.
- *R12*: Shop believes Consumer said S_3, using the *nonce verification rule* and $A7$.
- *R13*: Shop believes Consumer believes S_3.
  $P4$ has been proven. □

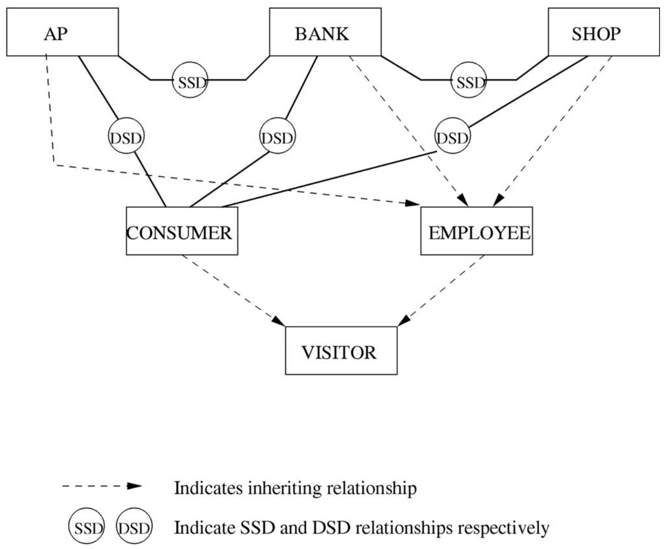
AP
BANK
SHOP
SSD
SSD
DSD
DSD
DSD
CONSUMER
EMPLOYEE
VISITOR
Indicates inheriting relationship
SSD DSD Indicate SSD and DSD relationships respectively

Fig. 4. The relationships of the roles in the new scheme.

5 RBAC Management

As we mentioned before, this paper focuses on the relationships of roles such as Role hierarchies, SSD, and DSD. In this section, we analyze RBAC management for the new system, which includes two parts duty separation constraints and the scenario of end-users.

5.1 Duty Separation Constraints

With RBAC, users cannot associate with permissions directly. Permissions must be authorized for roles and roles must be authorized for users. In RBAC administration, two different types of associations must be managed, i.e., associations between users and roles and associations between roles and permissions. When a user's job position changes, only the user/role associations change. If the job position is represented by a single role, then, when a user's job position changes, only two user/role associations need to be changed: the removal of the association between the user and the user's current role and the addition of an association between the user and the user's new role.

Both SSD and DSD relations must be included in the implementation. Once implemented, they may be configured for the specific application policy and may be altered at any time during the operational life of the system. SSD relations are enforced at administration time. DSD relations are enforced during runtime.

Now, we consider the RBAC management of the new payment scheme. There are four major roles in the system, the AP agent, the consumer, the bank, and the shop. The AP agent, the bank, and the shop would be companies comprised of many participants. We do not discuss the relationships of all the participants in these companies since they are beyond the scope of this paper. We only consider duty separation constraints. In Fig. 4, for example, since all staff in the AP agent, the bank, and the shop are employees, their corresponding roles inherit the employee role. It is similar to the relationship of the consumer and the visitor. The roles AP agent, the shop, and the bank have a DSD relationship with the role consumer. This indicates that an individual consumer cannot act the roles of the AP agent, the shop, or the bank simultaneously. The staff in these three companies have to first log out if they want to register as consumers. For example, a consumer who is a staff member of the AP agent can ask the AP agent to help him to get high anonymity. But, as a consumer, since the shop and the bank need to check the new coin's certificate $Cert_{c'}$ and signature, he/she cannot give herself/himself a new certificate $Cert_{c'}$ of a coin when he/she works for the AP agent. Another staff member of the AP agent can help this person.

The AP agent has an SSD relationship with the bank. This is because the duty of the AP agent is to help a consumer to get high anonymity. The bank knows the old coin $c = (g^r, Y^r I^s)$ and its certificate $Cert_c$. The AP agent sends the new certificate $Cert_{c'}$ of the new coin $c' = (g^{r+\rho}, Y^{r+\rho} I^s)$ to the consumer. The bank knows the new certificate $Cert_{c'}$ when the AP agent and the bank are authorized by a staff member common in both. If so, the consumer cannot have the required anonymity of coin. The shop also has an SSD relationship with the bank since the bank verifies the payment as well as depositing the coin to the shop's account. The SSD relationship is also a conflict of interest relationship like the DSD relationship, but much stronger. If two roles have an SSD relationship, then they may not even be authorized to the same individual. Thus, these three roles may never be authorized to the same individual.

After analyzing duty separation constraints, we see what a user needs to do when using the system on the Internet.

5.2 Scenarios of End-Users

End-users need to establish a session before accessing the AP agent, bank, or shop. In establishing the session, end-users choose a current active role set (ARS). The ARS determines what activities the users can perform. An ARS remains active until the users establish a new one. When a user is assigned roles which have DSD relationships, the session manager enables users to choose the subset of their assigned role set that they wish to have in their ARS. The user is presented with a list of items in the subset which do

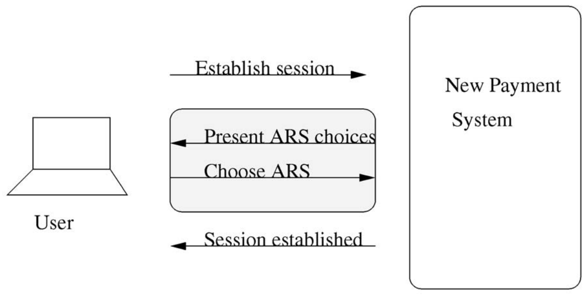

Fig. 5. An end user's scenario.

not violate any DSD relationships and asked to choose. The subset in the list, taken from the set of all possible assigned roles, contains the largest subset, which does not violate any DSD relationships. Once the choice is made, the session is established with all authorized roles (i.e., assigned roles along with all roles which the assigned roles inherit) being placed in the ARS. The ARS automatically establishes all authorized roles if there are no DSD relationships among the roles assigned to the user.

For example, in Fig. 5, a user is assigned the consumer's role, the ARS is {Consumer} since the role of a consumer has a DSD relationship with the AP agent, the shop, and the bank. The user can perform a consumer's activities only. If a user is assigned a manager in the AP agent, the set of ARS is {Manager, Operator, Quality controller}, as shown in Fig. 6. This means the user can perform as a manager, an operator, and a Quality controller. In the AP agent, the manager inherits the operator and the quality controller. However, the ARS of the Operator in the ARS at the AP agent is {Operator} since the operator has a DSD relationship with the quality controller. The ARS of the manager in the bank is {Manager, Account_rep, Auditor, Teller} because the Manager inherits the Account_rep (account representative), the Auditor, and the Teller. The manager can act as a manager and an account representative, an auditor, and a teller. The details are in Table 5.

TABLE 5
The Scenario of End-Users

Role	Active role set (ARS)
CONSUMER	{Consumer}
QUALITY CONTROLLER	{Quality controller }
OPERATOR	{Operator}
MANAGER1	{Manager, Quality controller, Operator }
AUDITOR1	{ Auditor in banks }
ACCOUNT_REP	{ Account representative }
TELLER	{ Teller }
MANAGER2	{Manager, Auditor in banks, Account representative, Teller }
AUDITOR2	{ Auditor in shops }
SELLER	{Seller}
MANAGER3	{Manager, Auditor in shops, Seller }

6 An Example and Implementations

In this section, we give a simple example and analyze two different purchase procedures. We show how to use the new e-cash for Internet purchases and how to get some smaller coins from the AP agent. This demonstrates the efficiency of the payment protocol.

6.1 An Example

This example shows the main steps in the e-cash scheme. We omit the details of two undeniable signatures in withdrawal and the scalable anonymity process because they are only used for verifying the user. In the following example, the module 47 is used.

6.1.1 Bank Setup

Suppose $(p, q, \gamma, k) = (47, 23, 2, 4)$, then $G_q = \{0, 1, 2, \ldots, 22\}$ is a subgroup of order 23. $g = 3$ is a generator of G_q. The bank's secret key $x_B = 4$ and hash function $H(x, y) = 3^x * 5^y$. The

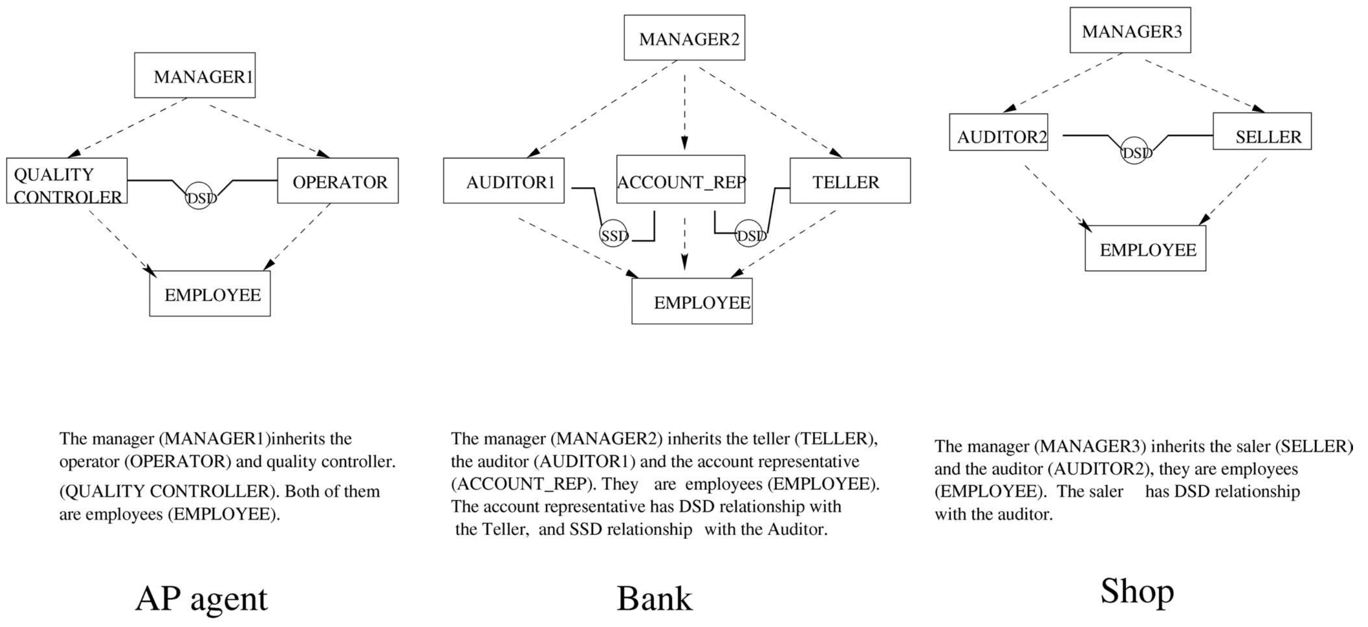
MANAGER1
QUALITY CONTROLER
DSD
OPERATOR
EMPLOYEE
The manager (MANAGER1)inherits the operator (OPERATOR) and quality controller. (QUALITY CONTROLLER). Both of them are employees (EMPLOYEE).
AP agent
MANAGER2
AUDITOR1
ACCOUNT_REP
TELLER
SSD
DSD
EMPLOYEE
The manager (MANAGER2) inherits the teller (TELLER), the auditor (AUDITOR1) and the account representative (ACCOUNT_REP). They are employees (EMPLOYEE). The account representative has DSD relationship with the Teller, and SSD relationship with the Auditor.
Bank
MANAGER3
AUDITOR2
DSD
SELLER
EMPLOYEE
The manager (MANAGER3) inherits the saler (SELLER) and the auditor (AUDITOR2), they are employees (EMPLOYEE). The saler has DSD relationship with the auditor.
Shop

Fig. 6. Role relationships.

bank publishes $H(x, y)$ and $\{p, q, g\} = \{47, 23, 3\}$. The public key of the bank is $Y = g^{x_B} = 34\ (module 47)$.

6.1.2 User Setup

We assume the secret of a user is $x_u = 7$ and the user sends $I = g^{x_u} = 25\ (module 47)$ to the bank. After checking some identifications, like social security card or driver's licence, the bank authorizes the user (consumer) with I.

After the bank setup and the user setup, the user can purchase.

6.1.3 Withdrawal

The user chooses $(r, s) = (2, 3)$ and computes

$$c = (g^r, Y^r I^s) = (9, 24)\ (module 47),$$

then signs a Schnorr signature S for the message $m = (c, currenttime)$. The user sends $c = (9, 24)$ and S to the bank; the latter sends back a certificate $Cert_c$.

The user contacts the AP agent if he/she needs a high level of anonymity or uses the coin in a shop directly (see Payment). The user and the AP agent follow the processes below. We suppose $h = 37$ is a public number.

6.1.4 Anonymity Scalability

The user reencrypts the coin c, chooses $\rho = 4$, and computes $c' = (a' = g^\rho a, b' = Y^\rho b) = (24, 16)\ (module 47)$ and signs a Schnorr signature S on $m = h^\rho = 36\ (module 47)$. Finally, the user sends (c, c', S, m) to the AP agent. The latter verifies the Schnorr signature S and (1) and sends a certificate $Cert_{c'}$ to the user if they are correct.

Since the new coin $c' = (24, 16)$ and its certificate $Cert_{c'}$ have no relationship with the bank, the user has high anonymity.

6.1.5 Payment

The user signs a signature $S = (e, u, v, t_1)$ of a message m which includes $c', Cert_{c'}$ and purchase time, etc., to prove the ownership of the new coin. For convenience, we assume $m = 11$. The user chooses $k = 5$, then computes

$$t = Y^k g^s = 1\ (module 47),$$
$$e = H(m, t) = 20\ (module 47),$$
$$u = 11\ (module 23),$$
$$v = 1\ (module 23),$$
$$t_1 = 34\ (module 47).$$

The shop who is convinced that the user is the owner of the coin computes $t' = 34\ (module 47)$ if the equation of $t' = tt_1$ and the signature S is successful. He/she does not know who the user is.

6.1.6 Deposit

The bank puts the money into the shop's account when the checking of the coin $C' = (24, 16)$ and the signature $S = (e, u, v, t_1) = (20, 11, 1, 34)$ are correct. The shop can also see that the money in his account is added.

When the user uses the old coin $(9, 24)$ again, then another signature $S_1 = (e_1, u_1, v_1, t_{12})$ is produced. Suppose $m = 3$ and $k = 1$, we have $e_1 = 30$ and $v_1 = 0$. Therefore, $(v_1 - v)/(e - e_1) = 7 = x_u(mod\ 23)$. It means no double spending of the old coin and the new coin.

6.2 Purchase Procedures

6.2.1 Purchase Procedure 1

In purchase procedure 1, a consumer decides how much money should be paid to the shop, withdraws the money from the bank, and pays it to the shop.

1. *Consumer to shop*: The consumer wants to buy some goods in a shop, so he/she contacts the shop for the price.
2. *Consumer to bank*: The consumer gets the money from the bank, the amount being embedded in the signature.
3. *Anonymity scalability*: If the consumer wants to maintain a high level of anonymity, he/she can ask the AP agent to certify a new coin which can be then used in the shop.
4. *Consumer to shop*: The consumer proves to the shop that he/she is the owner of the money and pays it to the shop. Then, the shop sends the goods to the consumer.
5. *Shop to bank*: The shop deposits the e-cash in the bank. The bank checks the validation and that there is no double spending of the coin. The bank transfers the money to the shop's account.

6.2.2 Purchase Procedure 2

In purchase procedure 2, the consumer does not have to ask the bank to send money since the consumer already has enough e-cash in his "wallet." All the consumer needs to do is to get some smaller e-cash from the AP agent to pay the shop.

There are four steps in purchase procedure 2. They are: 1) *consumer to shop*, 2) *consumer to AP agent*, 3) *consumer to shop* again, and (4) *shop to bank*. Step 2, *consumer to AP agent*, is different from Step 3 in procedure 1, but the other three steps are similar to those in procedure 1. Therefore, we focus only on Step 2, *consumer to AP agent*. It should be noted that electronic-cash is a digital message and a certification. We say that the AP agent can provide certificates of coins then provide a service in changing to small coin.

Consumer to AP agent: The consumer advises the AP agent of the amount of money to pay the shop from his wallet. He/she then asks the AP agent to make some smaller coins. By doing this, the consumer can also get a high level of anonymity. After checking the old money sent by the consumer, the AP agent creates some new coins of an equivalent value to the original coin. One of these new coins can be used in the shop.

We have already seen that the consumer can keep money in his wallet or get money from the bank. In both purchase procedures 1 and 2, most computations are done by the consumers, so the system is very convenient for Internet purchases.

7 Conclusions

In this paper, a new electronic cash scheme is described and its RBAC management is analyzed. The new scheme provides different degrees of anonymity for consumers.

Consumers have a low level of anonymity if they want to spend coins directly after withdrawing them from the bank. They can achieve a high level of anonymity through the AP agent without revealing their private information and are more secure in relation to the bank because certification comes from the AP agent, who is not involved in the payment process. There is little research available on RBAC for electronic cash schemes. We first canvas RBAC for payment schemes in this paper. With RBAC, the conflicts of different roles can be reduced and the complexities of the scheme can be decreased. The duty separation constraints of roles, role hierarchies, and scenarios of end-users are discussed in details.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their constructive suggestions and comments.

REFERENCES

[1] D. Chaum, "Blind Signature for Untraceable Payments," *Proc. Advances in Cryptology–Crypto '82,* pp. 199-203, 1983.
[2] B. Cox, J.D. Tygar, and M. Sirbu, "NetBill Security and Transaction Protocol," *Proc. First USENIX Workshop Electronic Commerce,* 1995.
[3] MastercardVisa, SET 1.0—Secure Electronic Transaction Specification, http://www.mastercard.com/set.html, 1997.
[4] D. Chaum, "DigiCash, an Introduction to E–Cash," http://www.digicash.com, 1995.
[5] H. Wang, J. Cao, and Y. Kambayashi, "Building a Consumer Anonymity Scalable Payment Protocol for the Internet Purchases," *Proc. 12th Int'l Workshop Research Issues on Data Eng.: Eng. E-Commerce/E-Business Systems,* pp. 159-168, 2002.
[6] D. Chaum and H. Antwerpen, "Undeniable Signatures," *Proc. Advances in Cryptology–Crypto '89,* pp. 212-216, 1990.
[7] A. Chan, Y. Frankel, and Y. Tsiounis, "An Efficient Off-Line Electronic Cash Scheme as Secure as RSA," NU-CCS-96-03, Northeastern Univ., 1995.
[8] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM,* vol. 21, pp. 120-126, 1978.
[9] M. Franklin and M. Yung, "Secure and Efficient Off-Line Digital Money," *Proc. 20th Int'l Colloquium Automata, Languages, and Programming,* pp. 265-276, 1993.
[10] T. Poutanen, H. Hinton, and M. Stumm, "NetCents: A Lightweight Protocol for Secure Micropayments," *Proc. Third USENIX Workshop Electronic Commerce,* 1998.
[11] D. Pointcheval, "Self-Scrambling Anonymizers," *Proc. Financial Cryptography,* pp. 259-275, 2001.
[12] D. Ferraiolo, J. Cugini, and D. Kuhn, "Role-Based Access Control (RBAC): Features and Motivations," *Proc. Computer Security Applications Conf.,* pp. 241-248, 1995.
[13] D. Ferraiolo, R. Sandhu, S. Gavrila, R. Kuhn, and R. Chandramouli, "Proposed NIST Standard for Role-Based Access Control," *ACM Trans. Information System Security,* vol. 4, no. 3, pp. 224-274, 2001.
[14] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST Model for Role-Based Access Control: Towards a Unified Standard," *Proc. Fifth ACM Workshop Role-Based Access Control,* pp. 47-63, http://doi.acm.org/10.1145/344287.344301, 2000.
[15] V. Gligor, S. Gavrila, and D. Ferraiolo, "On the Formal Definition of Separation-of-Duty Policies and Their Composition," *Proc. 19th IEEE CS Symp. Research in Security and Privacy,* http://citeseer.ist.psu.edu/gligor98formal.html, 1998.
[16] J. Barkley, K. Beznosov, and J. Uppal, "Supporting Relationships in Access Control Using Role Based Access Control," *Proc. Fourth ACM Workshop Role Based Access Control,* pp. 55-65, 1999.
[17] D. Ferraiolo, J. Barkley, and D. Kuhn, "Role-Based Access Control Model and Reference Implementation within a Corporate Intranet," *ACM Trans. Information and System Security (TISSEC),* vol. 2, pp. 34-64, 1999.
[18] R. Sandhu, "Role Activation Hierarchies," *Proc. Third ACM Workshop Role Based Access Control,* http: www.list.gmu.edu/confrnc/rbac/ps-ver/r98hier.ps, 1998.
[19] R. Sandhu, "Future Directions in Role-Based Access Control Models," *Proc. Int'l Workshop Information Assurance in Computer Networks,* pp. 22-26, 2001, http://www.list.gmu.edu/confrnc/misconf/.
[20] M. Bellare, O. Goldreich, and H. Krawczyk, "Stateless Evaluation of Pseudorandom Functions: Security beyond the Birthday Barrier," *Proc. 19th Ann. Int'l Cryptology Conf. Advances in Cryptology,* pp. 270-287, 1999.
[21] R. Canetti, O. Goldreich, and S. Halevi, "The Random Oracle Methodology" *Proc. 30th ACM Symp. Theory of Computing (STOC '98),* pp. 209-218, 1998.
[22] R. Rivest, "The MD5 Message Digest Algorithm," Internet RFC 1321, 1992.
[23] T. ElGamal, "Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Trans. Information Theory,* vol. 31, pp. 469-472, 1985.
[24] D. Chaum, A. Fiat, and M. Naor, "Untraceable Electronic cash," *Proc. Advances in Cryptology–Crypto '88,* pp. 319-327, 1990.
[25] C. Schnorr, "Efficient Signature Generation by Smart Cards," *Cryptology,* vol. 4, pp. 161-174, 1991.
[26] T. Yiannis, "Fair Off-Line Cash Made Easy," *Proc. Advances in Cryptology–Asiacrypt '98,* pp. 240-252, 1998.
[27] T. Okamoto, "An Efficient Divisible Electronic Cash Scheme," *Proc. Advances in Cryptology–Crypto '95,* pp. 438-451, 1995.
[28] T. Yiannis and M. Yung, "On the Security of ElGamal-Based Encryption," *Proc. Int'l Workshop Practice and Theory in Public Key Cryptography,* http://www.ccs.neu.edu/home/yiannis/papers/eg.ps, 1998.
[29] M. Burrows, M. Abadi, and R. Needham, "A Logic of Authentication," *Proc. Royal Soc.,* http://citeseer.nj.nec.com/burrows90logic.html, 1996.
[30] H. Wang and Y. Zhang, "Untraceable Off-Line Electronic Cash Flow in E-Commerce," *Proc. 24th Australian Computer Science Conf. (ACSC 2001),* pp. 191-198, 2001.
[31] H. Wang, Y. Zhang, J. Cao, and V. Varadharajan, "Achieving Secure and Flexible M-Services through Tickets," *IEEE Trans. Systems, Man, and Cybernetics,* special issue on M-Services, vol. 33, no. 6, pp. 697-708, 2003.
[32] D. Goldschlag, M. Reed, and P. Syverson, "Onion Routing for Anonymous and Private Internet Connections," *Comm. ACM,* vol. 24, pp. 39-41, 1999.

Hua Wang received the PhD degree in computer science from The University of Southern Queensland in 2003. He is a lecturer at the University of Southern Queensland, Australia. He has been active in the areas of information systems management, distributed database management systems, access control, software engineering, and electronic commerce. He has participated in research projects on mobile electronic systems, Web services, and role-based access control for electronic service systems. He has published more than 30 research papers in refereed international journals and conference proceedings. Dr. Wang is a technical PC member of APWeb '05 and ACSC 2005.

Jinli Cao received the PhD degree in computer science from the University of Southern Queensland in 1997. She is a senior lecturer in the Department of Computer Science and Computer Engineering at La Trobe University, Australia. She was a lecturer at the James Cook University, and University of Southern Queensland from 1998 to 2002. She is a member of the IEEE Computer Society. Her research interests include transaction management, distributed databases, mobile database management, Internet and Web information systems, electronic commerce, and Web services. She has served a number of international conference as a program committee member. She is a member of the editorial reviewer board for the *International Journal of Distance Education Technologies*.

Yanchun Zhang received the PhD degree in computer science from the University of Queensland in 1991. He is an associate professor in the School of Computer Science and Mathematics and the director of the Internet Technologies and Applications Research Lab (ITArl) at Victoria University of Technology. He is a founding editor and co-editor-in-chief of *World Wide Web: Internet and Web Information Systems (WWW Journal)*, and a founding book series editor on Web information systems engineering and internet technologies for Kluwer Academic Publishers, and a cochairman of the Web Information Systems Engineering (WISE) Society and steering committee chair of the WISE conference series. His research areas cover database and information systems, distributed databases and multidatabase systems, database support for cooperative work, electronic commerce, Internet/Web information systems, Web data management, Web mining, Web search, and Web services. He has published more than 100 research papers in refereed international journals and conference proceedings and has edited more than 10 books/proceedings and journal special issues. He has been a key organizer of several international conferences such as the APWeb '05 program committee cochair, APWeb '03 and APWeb '04 publication chair, RIDE '02 PC cochair, WISE '01 publication chair, WISE '00 general cochair, CODAS '99 cochair, etc. He is a member of the IFIP Working Group WG 6.4 on Internet Applications Engineering.