International Journal of Combined Research & Development (IJCRD)
eISSN:2321-225X;pISSN:2321-2241
Volume: 1; Issue: 2; June –2013

A simple approach of Peer-to-Peer E-Cash system

Mr. Dharamvir , Mr. Rabinarayan Panda
Asst. Professor, Dept. of MCA,
The Oxford College of Engineering Bangalore, India.

Abstract-With the popularization of the networking, the network payments have become a trend. Ecash is a payment system designed and implemented for making purchases over open networks such as the Internet. In this paper we review some of the main cryptographic techniques used throughout the ecash system.

Keywords: e-cash, electronic commerce, Electronic Payment Systems.

1. Introduction

Behind the scenes banks, credit-card companies, and other financial institutions have been processing transactions electronically for several decades now. Two important developments that will open up the field of electronic payment systems are now taking place. First, the prospect of electronic commerce over the Internet is creating a large demand for electronic payment methods for open networks. Second, the introduction of nation-wide electronic purse schemes is creating many more places and situations where smart cards can be used for cost-effective off-line payments. The aim of this work is to choose and implement a payment system, among the main expected characteristics, this electronic purse must allow direct client-to-client payments, e-shopping over Internet, and the transactions must be achieved anonymously.

The e-cash system uses different cryptography tools, namely asymmetric cryptography to establish a secure channel between participants, encryption to protect data from unauthorized readers, hash functions to increase efficiency, and digital signatures to authenticate the notes. It will finds its roots in the work by Chaum, who invented the notion of electronic (or digital) coins as well as the basic protocols for electronic cash. Electronic coins possess similar properties as metal coins, among which is the unique feature that a payment transaction leaves no trace about the identity of the payer.

2. Preliminaries

2.1 Secret key cryptography

This method has been invented 2000 years ago, by Julius Caesar. It consists of using the same secret key to encrypt and decrypt data.

2.1.1 Blowfish

Blowfish is one of the most common symmetric block ciphers implemented in Java. Blowfish and Twofish (Twofish is the version after Blowfish) were invented by one of the most famous cryptography authors, Bruce Schneier. The success of this algorithm in Java implementations can partly be explained by the fact that it is a non-patent and free algorithm to use. It takes a variable-length key from 32 bits to

448 bits. Since the key can be varied from a low to a high range, it is ideal for exporting; you just need to adapt the key-size to the exportations regulations and the local rules.

2.2 Public key cryptography

Symmetric cryptography is fast and secure. But how is it possible to exchange the secret key in a secure manner? We can for example exchange it through another channel, like the phone. Or store the private key in a floppy and give it physically to the receiver. Of course they are a lot of solutions, but they are not very handy. And what if you need to communicate securely with someone you've never met, or how to establish a secure channel between you and your bank to make e-banking?

The solution is called asymmetric cryptography, and is certainly one of the most innovations in the field. The concept was introduced by Whitfield Diffie and Martin Hellman in 1975. The idea is to use two keys, one for encryption and the other for decryption.

A user generates mathematically two keys. They are created so that the public key encrypts a plaintext and that the private key decrypts the resulting cipher text. The security resides in the fact that it is computationally infeasible to deduce the private key from the public key if you don't know the secret values used to build them. Once a key pair is created, the user published the public key and keeps the private key secret. Therefore everybody is able to encrypt a message using the public key, but only the owner of the corresponding private key is able to decipher the message. The need for sender and receiver to share secret keys via some secure channel is eliminated; all communications involve only public keys, and no private key is ever transmitted or shared. Some examples of public-key systems are Elgamal, RSA, and DSA. The most popular is RSA, which is used in the SSL protocol.

2.3 Hash functions

A hash function is a one-way function that transforms an arbitrary long message into a fixed size fingerprint. Such a function ensures that if the information is changed in anyway, even by just one bit, an entirely different output value is produced. It is computationally infeasible to find two different texts that produce the same message digest (fingerprint), and it is of course impossible to compute the original text from its fingerprint.

2.3.1 SHA

SHA stands for Secure Hash Algorithm and was developed by the NIST (National Institute of Standards) and the NSA (National Security Agency). It is closely modeled after the MD4 algorithm and was designed for use with the DSA (Digital Signature Algorithm) in mind.

2.4 Digital signatures

Asymmetric cryptography provides the possibility for everybody to encrypt a message (using the public key) and to be sure that only the owner of the corresponding private key will be able to decrypt it. But it doesn't provide authentication; all the messages are anonymous. What about Oscar how passes of as the bank and pleases Alice to identify her by sending her bank account number and her password? Digital signatures allow a person to digitally sign a document. An easy way consists simply in encrypting (signing) a document with the private key,

and so everybody can decrypt (checking the signature) with the corresponding public key:

3. The Basic Model of the System

The electronic cash protocol contains following stages, initialization, withdrawing, and payment, depositing. There are three participants in this scheme: the Bank, the customer, the merchant.
In the initialization stage, the bank publishes its public keys and some related information. In the withdrawing stage, a customer withdraws an e-cash from the bank. The customer then spends his /her e-cash in the payment stage. In the depositing stage, the merchant deposits the e-cash to the bank and equivalent money credited to his/her account.

There are four phases in the proposed protocol:

Initial Phase: The initial phase includes the following sub procedures. (1) Key-pair Setup: setup bank's public / private key. (2) Account Establishment: the users' (including customer and merchant) accounts in the bank are needed to be established before starting a transaction.

Withdrawal Phase: The customer in this phase can acquire electronic cash from the bank by running the blind signature. The withdrawal phase is required to be designed to protect customer's privacy.

Payment Phase: In this phase, customer and merchant aim to exchange their money and goods.

Deposit Phase: The merchant can change his coins received from the customer into real money via the bank.

Basic model of E-cash System

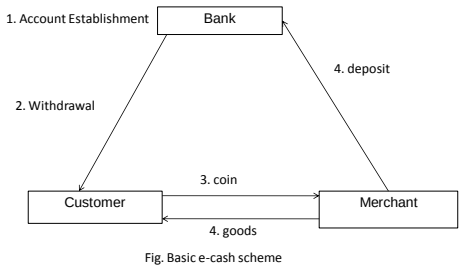

Fig. Basic e-cash scheme

4. The e-cash algorithm

The first point consists in generating valid bank notes, recognizable by the bank. To stay anonymous, the buyer creates a note, hides it in an envelope, and asks the bank to sign it. This process is called blind signature; the bank signs something without seeing it. Asymmetric cryptography is used to achieve this goal. The bank generates a pair of key. The public key is (*e*, *n*) and the private key is (*d*, *n*). These letters stand for:

- *e*: public encryption exponent;
- *d*: private decryption exponent;
- *n*: the modulus.

Let's take a sequence of bits that identifies the client, for example it's bank account. Then we split this value with a XOR function, so that

$$x \oplus x' = User_{id}$$

Let's repeat that several time, so that:

$$x_i \oplus x_i' = User_{id} \forall i$$

This sequence of x is called the RIS (Random Identity String).

A hash function applied to each item of the RIS will produce a sequence of y:

$$y_i = H(x_i) \ , \ y_i^{'} = H(x_i^{'})$$

A bank note will looks like:

$$M = (amount, y_1, y_1^{'}, y_2, y_2^{'}, ..., y_p, y_p^{'})$$

Because a hash is a one-way function, it is impossible to deduce the identity hided in the bank note.

In this project, *M* is a Big Integer that contains all the values in an Indian file. Because it is quite a huge value, the cryptographic operations are made on a hash of *M*:

$$m = H(M)$$

The buyer generates a random number *k*, the blinding factor, and sends the envelope to the bank:

$$Buyer \rightarrow Bank : (m \cdot k^e) \bmod n$$

The bank returns the signed envelope to the buyer:

$$Bank \rightarrow Buyer : (m \cdot k^e)^d \bmod n$$

The client computes:

$$\frac{(m \cdot k^e)^d}{k} \bmod n = \frac{m^d \cdot k^{ed}}{k} \bmod n = \frac{m^d \cdot k}{k} \bmod n = m^d \bmod n = signedNote$$

At this point the client owns a note signed by the bank, despite the latter never seen it. The question is now how to be sure that the note the bank signs corresponds really to the amount announced be the client? They are two ways to solve this problem. The first is simply to use a different signature for each amount. It's very simple, but it is also limited; a pair of keys is needed for each different amount, what makes impossible the creating of notes of any amount. The second solution is the one discussed previously: Make several notes (all with the same amount), send them together, let the bank choose one of them and finally reveal all the information (RIS and the blinding factors *k*) used to build the other notes; if everything is correct, the bank signs blindly the last envelope and returns it. This second approach has been chosen for this project.

The buyer can now pay a seller by sending him the note *M* and the signed hash of *M*:

$$Buyer \rightarrow Seller : M, signedNote$$

The seller verifies that the signature is correct, i.e. the bank has actually signed the note:

$$H(M) = signedNote^e \bmod n$$

The seller sends a random challenge to the client to make sure that the identity written on the note is correct:

$$Seller \rightarrow Buyer : r \ (\text{p bits})$$

The buyer has to reveal a part of his RIS (enough to check its validity but just not enough to find out his identity):

$$Buyer \rightarrow Seller : RISpart = [x_i \text{ if } r_i = 0, \ x_i^{'} \text{ if } r_i = 1 \ \forall i \in (1..p)]$$

The seller verifies that:

$$\text{if } r_i = 0 : H(x_i) = y_i$$
$$\text{if } r_i = 1 : H(x_i^{'}) = y_i^{'}$$

If everything is correct, the seller accepts the note. It is important to notice that this kind of money is one-time use; once a seller got a note, he needs to send it to the bank in order to credit his bank account.

The last step consists in depositing the note at the bank. In addition to the note, the seller sends also the *RISpart* and the challenge *r*:

$$Seller \rightarrow Bank : M, signedNote, RISpart, r$$

The bank first verifies that the signature and the values given in the *RISPart* are correct. Then if this note has not already been deposited, the latter is accepted and the account of the seller is credited. But if this note is already in the used-notes list, the bank will have to discover the cheater:

- If both *RISpart* are the same, then it is the seller who cheats; he tries to deposit twice the same note.
- If the *RISpart* is different for each note, then it is the buyer who cheats. Only a buyer is able to generate a valid RIS. Therefore, if the bank gets two different valid *RISpart*, it means that they have been generated in response to the two different challenges, [r_{11}, r_{12}, r_{13}..., r_{1p}] and [r_{21}, r_{22}, r_{23}..., r_{2p}], imposed by the sellers. This allow the bank to identify the cheater; because the challenges are random, they are different from each other (with probability $1\text{-}2^{-p}$), and therefore it exists at least one position *i* where

  $$r_{1i} \neq r_{2i}$$

  which allows the bank to identify the cheater:

$$Cheater = x_{1i} \oplus x'_{2i} \text{ or } x_{2i} \oplus x'_{1i}$$

4. CONCLUSION

This paper presents a lightweight electronic cash payment scheme. It has the flexibility that one would expect from present day paper based cash schemes, without the complexity that is sometimes associated with electronic payment systems. It combines the benefits of user anonymity and reusability of coins.

11. References

1. The Design of a Novel E-cash System with the Fairness Property and its Implementation in Wireless Communications.
2. D. Chaum, "Blind signature systems," In Advances in Cryptology - Proceedings of Crypto '83, pp.153-156, 1983.
3. S. J. Kim and H. K. Oh, "Efficient anonymous cash using the hash chain," IEICE Transactions on Communications, Vol.E86-B, No.3, pp. 1140-1143, 2003.
4. T. Okamoto and K. Ohta, "Universal electronic cash," In Advances in Cryptology – Proceedings of Crytpo'91, pp. 324-337, 1998.
5. S. J. Kim and H. K. Oh, "Efficient anonymous cash using the hash chain," IEICE Transactions on Communications, Vol.E86-B, No.3, pp. 1140-1143, 2003.