A Tracable Proactive Anonymous Electronic Cash

Chang Yu Cheng[1], Kamaruzzaman Seman[2] ,Senior Member IEEE ,and Jasmy Yunus[3]

Faculty of Electrical Engineering,
University Technology Malaysia,
Kampus Skudai, Karung Berkunci 791,
80990 Johor Bahru, Malaysia.
Tel: (607) 5505106 Fax:(607) 5566272
Email: [1] cyu_cheng@hotmail.com
[2] kzaman@suria.fke.utm.my
[3] jasmy@suria.fke.utm.my

Abstract- **In this paper, we propose a traceable anonymous electronic cash, based on Digital Signature Scheme(DSS), but with difference whereby the coin's anonymity can be revoke by a quorum of the server. Our system, unlike previous anonymous systems, can prevent crimes from successfully being perpetrated, and employs revocation to do so. We introduce this traceable electronic cash that require a quorum of servers to produce a signature to the receiver, and a (possibly different) quorum to be unblended the coin. We also applied the method of hints to our protocol to enable tracing of correctness of the signature, and the corresponding coins.**

1.0 Introduction

The traceable electronic cash is a method to enable the generation of blind signatures, which can later be unblended by the signer. It is analogy that the cash money. Where each cash money has a serial number. The particular cash money can be identified by checking on the serial number whether it has been black listed. This traceable electronic cash is in sharp contrast to traditional blind signatures, which are information theoretically blinded to the signer. The typical application where the need for this functionality arises is for cases where privacy of individuals is assured until some criminal or otherwise unusual activities is detected. Upon detection, identification of the origin of a signature becomes important in identifying the source of the unwanted activity. This is applied to private access tokens, authorized anonymous account, and here, electronic money.

Both the generation and the tracing are performed under quorum action. The generation protocol is distributed to increase the availability and security of the system; the tracing protocol is distributed in order to increase the availability of the system, and to introduce control. We present a method to trace the electronic cash for DSS signatures.

2.0 Digital Signature Standard (DSS)

We use DSS [1] as the underlying signature algorithm. Note that, since we use different moduli at different times, we use $[op]_z$ to denote the operation op mod z.

Key generation. A DSS key is composed of public information p,q,g, a public key y and a secret key x, where:

1. p is a prime number of length l, where l is a multiple of 64 and $512 \le l \le 1024$.
2. q is a 160 bit prime divisor of p-1.
3. g is an element of order q in Z^*_P .The triple (p,q,g) is public.
4. $x \in_u Z_q$ is the secret key of the signer.
5. $y = [g^x]_p$ is the public verification key.

Signature Algorithm. Let $m \in Z_q$ be a hash of the message to be signed. The signer picks a random number $k \in_u Z_q$, calculates $[k^{-1}]_q$ (w.l.o.g k and k^{-1} values compare to DSA description are interchanged), and sets

$$r = [[g^{k^{-1}}]_p]_q$$

$$s = [k(m + xr)]_q$$

The pair (r,s) is the signature on m w.r.t the signer whose public key is $y = [g^x]_p$.

Verification Algorithm. A signature (r,s) of a message m can be publicly verified by checking that

$r \stackrel{?}{=} [[g^{ms^{-1}} y^{rs^{-1}}]_p]_q$.

The verifier have (m,r,s) and y. So he can proof this equation.

Note that:

$$[[g^{ms^{-1}} y^{rs^{-1}}]_p]_q = [[g^{ms^{-1}} g^{xrs^{-1}}]_p]_q$$

$$= [[g^{(m+xr)s^{-1}}]_p]_q = [[g^{k^{-1}}]_p]_q = r$$

From

$$s = [k(m + xr)]_q \Rightarrow s^{-1} = [k^{-1}(m + xr)^{-1}]_q$$

$$\Rightarrow k^{-1} = (m + xr)s^{-1}$$

3.0 Requirements

We wish to obtain a signature scheme where blind signatures can be distributively produced by a quorum of trustees, and these signatures always can be unblended by (possibly different) quorum of trustees. Let the signature key x be distributed using a (t_S, n) secret sharing scheme, and the tracing key x_t distributed using (t_t, n) secret sharing scheme. We want a signature scheme with the following properties:

Requirement 1:Correctness
Signatures can be correctly generated using a ($3t_S$, n) threshold scheme, by any size ($3t_S$ + 1) quorum out of the n trustees, even in the case where a coalition of up to t_S dishonest participants attempt to disrupt the generation of valid transcripts.

Requirement 2: Quorum tracing
Valid signatures can always be unblended, i.e. signatures matched to signing session or vice versa, by any size (t_t + 1) quorum out of the n servers. In other words, no coalition of less than t_S + 1 signature servers, interacting with honest servers a polynomial number of times in the signing and tracing protocols, can produce a valid signature that will not be traced to the tag of one of the signing session by any group containing at least t_t + 1 honest server in the tracing protocol [2].

4.0 Tools

Let us briefly describe what existing tools we will use in the protocol specification before describing these in more detail:

1. Polymonial Interpolation Secret Sharing
This is the well-known result in which a secret s is shared by choosing at random a polynomial f(x) of degree t, such that f(0)=s, and distributing n points on this polynomial. Given any t+1 out of the points, the entire polynomial f can be reconstructed, and specifically, f(0) can be produced. However, given only t such points on the polynomiual, each secret s = f(0) is equally likely. We call this a (t,n) sharing of s.

2. Join Zero Secret Sharing
This protocol generates a sharing of a "secret" whose value is zero. Such a protocol is that each players deals a sharing of the value zero [3].

3. Computing Reciprocals
Given a secret $k \in Z_q$, shared among players P_1,..... P_n, we want to generate the value $[g^{k^{-1}}]_p$ without revealing information on k or k^{-1}.

5.0 Description of the Traceable electronic cash protocol

5.0.1 Foundation of the DSS

-Given a distributed held value k, shared using (t_S, n) threshold scheme, we want to calculate $g^{k^{-1}}$ without revealing any secret information.

-The server share a secret a, using a (t_S, n) secret sharing scheme, a value b=0 using ($2t_S$, n) secret sharing scheme. Each server S_i calculate $v_i = [k_i a_i + b_i]_q$, $A_i = [g^{a_i}]_p$. A is calculated as the (t_S, n) interpolation of the shares of A. v as ($2t_S$, n) interpolation of shares of v. Result $[A^{v^{-1}}]_p = [g^{a[k^{-1}a^{-1}]}] = [g^{k^{-1}}]_p$ is output.

5.0.2 Signature Generation

istributed using (t_S, n) secret

distributed using (t_t, n) secret sharing scheme.

1. Server distributively generate
   - A random secret x using (t_S, n) secret sharing scheme.
   - A random secret x_t using (t_t, n) secret sharing scheme. x_t is link with identity of the receiver R.

2. Each server publish his share of public key $y_i = [g^{x_i}]_p$, and $h_i = [g^{x_{ti}}]_p$.

3. Each server S_i then prove knowledge of his secret share x_i to the other server.

4. f server, generate two random secret k, a $\in Z_q$, secret share using (t_S, n) secret sharing scheme. Let k_i, a_i denote the share held by S_i.

5. S_i also distributively generate b, in ($2t_S$, n) zero-sharing, where server S_i has share b_i. S_i also distributively generate ($3t_S$, n) zero sharing on c, where server S_i has share c_i.

6. Server compute $r = [g^{k^{-1}}]_p$ by following steps s follows:
   - Server S_i compute $v_i = [k_i a_i + b_i]_q$, $A_i = [g^{a_i}]_p$. The server publish (v_i, A_i).
   - The participants compute A by (t_S, n) interpolation of A_i.
   - The participants compute v as ($2t_S$, n) interpolation of v_i's and calculate $r = [A^{v^{-1}}]_p$. r send to R

7. The receiver R has a message $m \in Z_q$ that wants to signed. R then compute (t_S, n) secret sharing (m_1, m_2 ..... m_n) for m, with public information (M_1, M_2 ..... M_n) = (g^{m_1}.....g^{m_n}) and a (t_S, n) secret sharing (r_1, r_2 ..... r_n) of r, with public information (R_1, R_2 ,..... R_n) = (h^{r_1}.....h^{r_n}) = ($g^{x_1 r_1}$,.....$g^{x_1 r_n}$).

8. -Server (t_S, n) interpolate tag (g^m, h^r).
   -Server generate $s_i = [k_i(m_i + x_i r_i) + c_i]_q$. For method of verifying shares of s_i, refer to section 5.0.1.1
   and section 5.0.1.2.
   -Server (t_S, n) interpolate $s = [k(m + xr)]_q$ and send S to R.

9. R verified (m,r,s) is a valid DSS on m.

5.0.3 Proof during complain on Signature Server

If a complain receive by signature servers, then server S_i publish s_i, m_i, r_i, $[g^{k_i}]_p$,$[g^{b_i}]_p$,$[g^{c_i}]_p$. And proof to other server that :

$$\log_{g^{k_i}} (g^{s_i} g^{-c_i} g^{k_i(-m_i)})^{r_i^{-1}} = \log_g y_i = x_i$$

$$\log_{g^{k_i}} (g^{v_i} g^{-b_i}) = \log_g A_i = a_i$$

-Each server verifies that above proofs are valid and that $M_i = g^{m_i}$, $R_i = h^{r_i}$.
-Then, server verifies (as above) the correctness of $[g^{k_i}]_p$,$[g^{b_i}]_p$,$[g^{c_i}]_p$.
-Any server who fails/refuses is replaced, and the signature generation restarts [4].

5.0.3.1 To trace the coin

Note that, the User ID is link with x_t. The series that store by signature server is in the form below: $tag = (g^m, h^r)$, coin (i.e. m,r,s).
For example, server S want to trace user id for a particular coin (m,r,s). When tracing coin (m,r,s), For example the server know what tag to be traced. Example, see whether the tag x_t is match the $x_t = 10$. So the procedure as below:

Given a description of (m,r,s), and its related tag, $tag = (tag_a, tag_b) = (g^m, h^r)$, we check this coin's secret tag number.

$$\log_{tag_a^{rm^{-1}}} tag_b = \log_{g^{mrm^{-1}}} h^r = \log_{g^r} g^{rx_t} = x_t$$

Check the value of x_t (e.g. $x_t \stackrel{?}{=} 10$) . This holds iff coin corresponds to the tag. The value x_t is distributed over the server. Then the server can check the ID of user for x_t = 10. This is to prevent money laundering where the user ID will be identified based on the coin and tag.

5.0.3.2 Modified method to trace a coin

Another way of tracing is for example, set x_t,= 10 for a certain user, trace the coin (m,r,s) with its tag , $tag = (g^m, h^r)$

i.e. check $h^r \stackrel{?}{=} g^{x_t r}$ i.e. $h^r \stackrel{?}{=} g^{10\,r}$ where the value h^r is from the tag, r is from the coin. This is to prevent blackmailing, where the coin owner is known, but to trace the coin of such owner.

6.0 Hint-generation

6.0.1 Hint generation using El Gamal encryption : Previous work done

Details of this section kindly refer to [5]. Let x_h be a private key distributively held by the tracing servers; $y_h = [g^{x_h}]_p$ is the corresponding public key.

1. The receiver calculates an ElGamal encryption of *m*: He chooses a $\gamma \in Z_q$ and calculates (a,b) = (mg^{γ} , y_h^{γ}). This pair is sent to the servers.

2. (a) The servers distributively compute $hint_i = a^{x_{hi}} / b$. The jointly generated from *hint*$_i$ by a quarom of server, *hint*, will be linked with the identity of the receiver. x_h is always fixed value hold by the server

   (b) In order to prove that every server has performed the correct exponentiation the server run a protocol for proving valid exponentiation. This is proof that $\log_a(hint_i . b) = \log_g(y_{hi}) = x_{h_i}$ for a given quadruple (a, g, (hint$_i$.b), y_{hi}).

   (c) The servers compute *hint* as the Lagrange-weighted product of the shares *hint*$_i$ of the servers in the quorum. (This value equals $[m^{x_{hi}}]_p$ if R did not cheat). Note that, after reconstruction,

   $$hint_i = [\frac{(mg^{\gamma})^{x_{hi}}}{(g^{x_h})^{\gamma}}]_p = [m^{x_{hi}}]_p .$$

   $$hint = [\sum_{r=1}^{r=k} hint_i] = m^{x_h} .$$ Where $[\sum_{r=1}^{r=k} hint_i]$ is Reconstruction of Secret Sharing by k servers in the quorum. And $m = [m^{x_h}]^{1/x_h}$.

6.0.2 Hint generation using RSA cryptography : Modified method

Let x_h be a private key distributively held by the tracing servers; $y_{h_i} = [g^{x_{h_i}}]_p$ is the corresponding public key.

3. The receiver calculates an RSA encryption of *m*: He chooses a d $\in Z_q$ and calculates (a,b) = ($[m^d]_p$, y_{h_i}). This pair is sent to the servers.

4. (a) The servers distributively compute $hint_i = [((m^d)^{x_{hi}})^e]_p = [m^{x_{hi}}]_p$. The jointly generated from *hint*$_i$ by a quarom of server, *hint*, will be linked with the identity of the receiver. x_h is always fixed value hold by the server

   (b) In order to prove that every server has performed the correct exponentiation the server run a protocol for proving valid exponentiation. This is proof that $\log_m(hint_i) = \log_g(y_{hi}) = x_{h_i}$ for a given quadruple (a, g, (hint$_i$), y_{hi}).

   (c) The servers compute *hint* as the Lagrange-weighted product of the shares *hint*$_i$ of the servers in the quorum. (This value equals $[m^{x_{hi}}]_p$ if R did not

cheat). Note that, after reconstruction, $hint_i = [((m^d)^{x_{h_i}})^e]_p = [m^{x_{h_i}}]_p$.

$hint = [\sum_{r=1}^{r=k} hint_i] = m^{x_h}$. Where $[\sum_{r=1}^{r=k} hint_i]$ is Reconstruction of Secret Sharing by k servers in the quorum. And $m = [m^{x_h}]^{1/x_h}$.

6.0.3 Tracing from known signed message to signing session

Given a description *(m,r,s)* , the tracing servers compute a value $trace_c = [m^{x_h}]_p$ from (m,r,s) and a particular fixed value x_h. Then they compare $trace_c$ with the stored hints. *hint* can be computed by a quarom of server based on $hint_i$.

If $trace_c \equiv hint$ for a particular record, then the signed message corresponds to the signing session of this record. Since server link hint with receiver ID, then server can trace the receiver ID based on (a,b)= (mg^γ, y_h^γ) of receiver in El Gamal encryption, and (a,b) = $([m^d]_p, y_{h_i})$ in RSA cryptography.

7.0 Protocol for E-Money System

7.0.1 Withdrawing a Coin

Here, we represent a coin by a pair (x,s) where x = secret key of the coin pair. Also, $s=Sig_{Bank}(y)$, and y=public key of the coin pair. In order to perform the withdrawal of a coin, the following protocol is executed:

1. withdrawer, whom we will call Alice, select a secret key, x, by himself. And using the RSA code, to generate a public key y. She proves her identity to the Bank, where she shows the id, *id* to bank, which bank verified she is the customer.
2. By cooperation of a quorum of Bank to compute a signature $s = Sig_{Bank}(y)$. And the bank servers get a tag *tag*, linked to this signed message, but not linkable by less than a quorum of bank servers. The tag is stored in secure database.

7.0.2 Spending a Coin

A coin (x,s), where x is the secret key. This secret key can produce a signature signed by Alice, Sig_{Alice}.

1. The spender of the coin, Alice, send (y, $Sig_{Bank}(y)$) to the payee, Shop. The shop verifies the validity of (y, $Sig_{Bank}(y)$) by using Bank's public key.
2. The shop sends the challenge c to Alice. Which is a value set by Shop.
3. Alice sends the answer $Sig_{Alice}(c)$ to the Shop. The shop can verified this signature by using Alice's public key y. The shop will store (y, $Sig_{Bank}(y)$, c, $Sig_{Alice}(c)$).

7.0.3 Depositing a Coin

A spent coin is deposited by forwarding the transcript (y, $Sig_{Bank}(y)$, c, $Sig_{Alice}(c)$) to the Bank. The Bank verifies $Sig_{Bank}(y)$ and also verifies $Sig_{Alice}(c)$. The bank also double check whether this transcript hs been deposited before. If everything is okey, the Bank will debits to Shop and credits the Alice's account. The bank then saves the transcript in a database containing all transcripts, and checks of overspending of the coin [6].

7.0.4 Tracing the coin

The Bank can trace the signature $Sig_{Bank}(y)$ to get its tag *tag*. This tag will expose the identity *id* of the withdrawer, i.e. Alice. The Bank can also input the *id* of Alice, to verified whether it is correct signature $Sig_{Bank}(y)$ that need to be verified. Details kindly refer to 5.0.3.

8.0 Conclusion

In this paper, we based on a signature scheme Digital Signature Scheme, but further modified to enable the coin to be traced. I.e. we enclosed a tag to each coin. The serial number of the coin can be traced by checking on the coin and its tag. This serial number can identified certain coin that has been misused/blackmail. However, such task requires the join quorum of certain amount of server to do it. The signature of the coin also, is being generated by another quorum of server.

References

[1] Bruce Schneier "Applied Cryptography, Secnd Edition", John Wiley & Sons. 1996.

[2] Markus Jakobsson "Privacy vs Authenticity"PhD thesis.Unversity of Carlifornia,San Diego, 1997.

[3] R.Gennaro, S. Jarecki, H.Krawczyk and T.Rabin, "Robust Threshold DSS Signature",Advances in Cryptology-Proceeding in Eurocrypt'96. pp 356-371.

[4] M.Jakobsson and M.Yong, "Revocable and Versitile Electronic Money", 3rd ACM Conference on Computer and Communication Security, 1996. pp76-87.

[5] M.Jakobsson and Joy Muller "Improved Magic Ink Signatures Using Hints. IEEE 1998.

[6] M.Jakobsson and Monti Yung, "Applying Anti-Trust Policies to Increase Trust in a Versatile E-Money System. IEEE 1998.

[7] Stanislaw Jarecki. "Proactive Secret Sharing and Public Key Cryptosystems". Master Thesis. MIT, 1995.

[8] Rosario Gennaro. "Theory and Practice of Verifiable Secret Sharing", PhD thesis, MIT. May1996.