Final Report on Anonymous Digital Cash

— from a designer's perspective

Hua Yu and Zhongtao Wang
{hyu@cs, zhongtao@andrew}.cmu.edu

1 Introduction

Here we report our analysis of different anonymous Ecash systems, we'll focus on Chaum's system ([CFN90]) and Brands' system ([Bra93]). We've gone through a long (and hard) way to understand these systems, and it's our hope that this report will give a very clean and unified view on the design issues, like:

- why do they use this hash function here?
- how exactly did the authors come up with such a seemingly daunting system?
- how to introduce new features into the existing systems?

Since this is an analysis, we aren't going to re-introduce these systems in full length. The reader is advised to keep a copy of at least the above-mentioned 2 papers at hand, although we have tried to make this report as self-contained as possible.

The lifecycle of a coin is basically 3 protocol:

1) withdrawal: Alice withdraws ecash from Bank
2) payment: Alice uses ecash to purchase from Merchant
3) deposit: Merchant deposits the received coin to Bank for credit

In [OO91], the following 6 criteria are put forward for ideal cash system:

1) Independence: The security of ecash cannot depend on any physical condition. Then the cash can be transferred through networks.
2) Security: The ability to copy and forge cash must be prevented
3) Privacy (untraceability): the purchase should be anonymous
4) Off-line payment: payment can be made without Bank's participation
5) Transferability
6) Divisibility

Now we'll see how to achieve these properties step by step.

2 Chaum's anonymous Ecash scheme [CFN90]

Here we assume the reader has some knowledge of how this scheme works. (refer to [CFN90] in case not). Let's take a designer's view of Chaum's scheme. This gives us some insight on how to achieve those properties, which also applies to other Ecash schemes.

Step 1 Independence

For the coin to be legitimate, it has to bear a signature of the bank. The public key signature is clearly the choice here, since everyone can verify it. Now how can we prevent others from copying and using Alice's coin? Suppose we don't want to physically lock the coin in a safe or something. Of course we'll have to keep some information secret to ourselves (if Eve knows as much of the coin as Alice, she can do whatever Alice can do with the coin), but not necessarily the coin itself.

The basic idea is to split the coin into two parts (A,B), where there's some mutual information between A and B. Let's call B the coin, which we don't intend to guard physically, but we definitely keep A secret.

When the Merchant sees B, he'll ask for information about A to verify the ownership, thus the knowledge of A signifies ownership.

In this view, there's nothing so great about [CFN90] scheme or any other digital cash scheme we've seen so far. We still need to protect something, so the independence assumption is actually not fulfilled.

In [CFN90] scheme, that A is the construction details of B. Back to the root, (A,B) is just (x, f(x)), where f() is a one-way hash function. This way Eve can't figure out the secret A (=x) from observing B (=f(x)).

Here we note that other forms of (A,B) are possible.

Step 2 Untraceable (Blind Signature)

Although independence is what we can never achieve in the exact sense, we have somehow by splitting the information, alleviated the difficulty. Now let's move on to the untraceability criterion.

Of course the Bank has to know Alice's identity before she can withdraw the coin, blind signature gives us a solution to get a coin but keep the issuer (Bank) from knowing its existence. Here any blind signature scheme fits in:

1) Alice generate a candidate upon which she would like to get the Bank's signature
2) She blinds it using some blinding factor r
3) Alice sends the blinded candidate to Bank
4) Bank signs it, return it to Alice
5) Alice unblinds it, getting a signature on the original message

In [CFN90], the signature takes the form of cubic root, i.e. Bank's signature on any number x is $x^{1/3} \pmod n$. Here n is the public key of the Bank, whose factorization is only known by the Bank, with the underlying assumption that it's difficult to calculate the cubic root without the knowledge of the factorization of n. Thus, the signature here is $f(x)^{1/3}$.

Step 3 Double spending

Now let's see how to prevent Alice from spending the coin twice. We note that Alice has to reveal some information of A when spending the coin. The idea is to have Alice reveal different information for different purchase, so that in the first time the information she revealed will just be enough to prove her ownership of the coin, in the second time the additional information she provided will lead to her identity being disclosed.

So Alice's identity, call it u, is used in construction of B. We use secret splitting method to split u into 2 pieces. Each time when she tries to spend the coin, she'll have to reveal 1 of the pieces; after double spending has happened, we have gathered both pieces of the secret, thus we can rebuild u, revealing Alice's identity.

The easiest way of secret splitting is xor: using random number r, $u = r\oplus(u\oplus r)$, the resulting pieces r and $u\oplus r$ are also random. Other ways of secret splitting are also possible.

Summary: The first (simplest) working protocol

So far, we have decided the digital money to be of the form (A,B), where B is the public part, everyone can see it; A is the secret part, where Alice's identity u is part of it. The Bank will sign on B. B is a one-way hash of A, so it's ok for others to see B. Every time when making a purchase, Alice reveals a piece of A, which is enough to verify she knows A, but not enough to reveal A (thus her identity). Specifically, in [CFN90], the coin takes the form: $B = f(g(r), g(r\oplus u))$ where g is a publicly known one-way hash function. The 1st time Alice making a purchase, she reveals r and $g(r\oplus u)$ to Merchant; the 2nd time she reveals g(r) and $(r\oplus u)$ to the Merchant. In this way we achieved both the goals:

1) Merchant can verify Alice knows A: from r he calculates g(r), combined with $g(r \oplus u)$ he can verify $f(g(r), g(r \oplus u)) = B$
2) Merchant can't deduce u from r and $g(r \oplus u)$ since g is one-way function.

The role g(.) plays here is like bit-commitment. Alice can hide half of the secret by telling only g(r), since it's not trivial to reverse g(.); when Alice reveals the secret r, Merchant can check she's not cheating by telling a random number, since g(r) has to be a certain number for the checking to go through.

Then comes the question: how to ensure Alice doesn't always tell the same piece of information again and again? Random challenge comes into play here. The Merchant generates a random challenge string every time, Alice has to comply by revealing the corresponding piece of secret.

Here we emphasize that this protocol is basically a clever combination of some simple crypto techniques: secret splitting, one-way function, random challenge. [CFN90] is only one possible implementation of the ideas sketched above.

Step 4 Prove double spending, and other stuff

Now let's be careful: the Bank can frame Alice up as a double-spender! This is a direct result that Bank can produce coin by itself even if Alice didn't ask for it, since Bank can generate random numbers and also know Alice's identity u. Suppose a bad Bank employee Mallory generates such a coin, then he can either spend it, or double-spend it, with Alice always being the victim.

The counter-approach is either to protect u or to sign the coin Alice has withdrawn. [CFN90] chose the second approach. Here we'll not go into details, except that Alice's signature is not forgeable.

By adding more and more constructions into the basic scheme, we can achieve more and more desirable properties. [CFN90] has another 2 sections on untraceable check and blacklisting withdrawals, which is clever but tedious for us to go into details. Keeping in mind the design of the basic scheme, tweaking it into more complicated schemes is a business somehow like constructing a proof for a math. problem. IMHO, providing a formal verification tool for such E-cash schemes is also kinda like verifying the correctness of a math proof.

Step 5 Trustee-based Tracing

For anonymous Ecash to work in real life, the government would like to trace any purchase of interest. So [BGK95] introduced trustee-based tracing: trustee is a third party trusted by both Alice and government, with the cooperation of which government can always trace a purchase even if Alice didn't double-spend.

The basic idea is to give one piece of the secret A to trustee, so that when Alice reveals another piece when first spending the coin, the combined knowledge leads to her identity. In the simplest form in lieu of [CFN90] scheme, we can just ask Alice to conduct a payment protocol with Trustee, then he knows a piece of the secret. When Alice does the actual purchase, it's technically double-spending already, so we can trace Alice with trustee cooperating. In case we want multiple trustee, (to make the big brother smaller), just make the piece of information each trustee needs to know smaller, by splitting the secret to multiple smaller pieces.

3 [OO92] an ideal ecash system

The ecash system proposed in [OO92] is based on [CFN90] scheme, with the underlying crypto techniques changed to Blum number and Williams integer for the main purpose of achieving divisibility. It used a hierarchical structure table to subdivide a coin into smaller pieces, so that Alice can have some flexibility to pay for some value less than C. It's a pain going through all those number theory to see how it works, but the basic design remains unchanged.

The system achieved all 6 properties listed above, and is claimed it can be implemented efficiently.

4 Brands' model [Bra93]

Again, we assume the reader has some knowledge of Brands's basic scheme [Bra93], although it's definitely not a fun to read it. [CP92] is the precursor of [Bra93]. Here we give a much cleaner explanation of the model, following roughly the same steps as before.

4.1 Signature Scheme [CP92]

The reader is advised to get familiar with Schnorr's authentication and signature scheme first, which is based on the difficulty of calculating discrete logarithms, to better understand the basic signature scheme used here:

Bank chooses two primes p and q such that $q|(p-1)$. Let $g\in Z_q^*$ be an element of order q, the group generated by g is denoted as G_q. Bank's secret key is $x\in Z_q^*$, it publishes the public key $h=g^x$ together with p, q, and g. The signature on message $m\in G_q$ is $z=m^x$ plus a proof for $\log_g h=\log_m z$. The proof goes as following:

1) Bank chooses $w\in_R Z_q$, computes $(a,b) = (g^w,m^w)$, sends (a,b) to Alice
2) Alice chooses a random challenge $c\in_R Z_q$, sends it to Bank
3) Bank sends back $r=w+cx \pmod q$
4) Alice verifies if $g^r=ah^c$ and $m^r=bz^c$

This minimum-knowledge proof can be easily turned into a signature scheme by fixing c=H(m,z,a,b), where H(.) is a publicized one-way hush function (as in Fiat-Shamir scheme, see [FS87]). As noted above, the signature on m is simply z ($=m^x$), but to convince others it's a valid signature, we need to present (z,a,b,r). Later on we'll just say Sign(m)=(z,a,b,r).

Now we can already use (m, Sign(m)) as a basic digital coin, except that it has no way to deal with proving ownership, protecting anonymity, etc.

4.2 Blind Signature [CP92]

To blind message m, Alice chooses a random number t and raises m to the t^{th} power, $m_0=m^t$. Bank signes m_0, i.e. $z_0=m_0^x$. Of course it's easy for Alice to extract $z=m^x$, but she needs also to extract Sign(m)=(z,a,b,r), from $\text{Sign}(m_0)=(z_0,a_0,b_0,r_0)$ provided by Bank:

1) Bank chooses $w\in_R Z_q$, computes $(a_0,b_0) = (g^w,m_0^w)$, sends (a_0,b_0) to Alice
2) Alice chooses at random u,v and computes:
   $a=a_0^u g^v$, $b=(b_0^{1/t})^u(m_0^{1/t})^v$, also we remind the reader that $m=m_0^{1/t}, z=z_0^{1/t}$.
   Alice calculates the challenge c=H(m,z,a,b), blinds it to $c_0=c/u \pmod q$, sends c_0 to Bank.
3) Bank sends back $r_0=w+c_0x \pmod q$
4) Alice verifies if $g^{r0}=a_0h^{c0}$ and $m_0^{r0}=b_0z_0^{c0}$
5) Alice computes $r=u(r_0+v) \pmod q$, the signature on m is then (z,a,b,r).

It's straightforward to verify (z,a,b,r) is a signature on m. Since t is unknown to the Bank, the Bank has no way of knowing the existence of coin (m, Sign(m)).

4.3 Restrictive Blind Signature [Bra93]

Brands calls his restrictive blind signature a new primitive, but actually it's almost the same as the blind signature scheme described above. The difference comes from the observation that the Bank need only sign once on m_0 (i.e. m_0^x), while Alice can extract m^x for any message $m=m_0^s$, with s be anything. So rather than having the Bank computes m_0^x for a different m_0 each time, we can fix m_0 when opening the account, have the Bank sign on it (i.e. m_0^x). Later on Alice can only get signatures on message $m=m_0^s$(not any message), but she can get it easily by herself. It's just $m^x=(m_0^x)^s$. This has great computational advantage over the cut'n'choose protocol used in [CFN90]. Of course she will still need to conduct the protocol to get a different proof each time. The protocol is exactly the same as above once we substitute 1/t for s.

To understand why Brands call it restrictive blinding, let's see how Alice's identity is encoded in the coin. To this end Bank chooses another 2 generators g_1,g_2. Suppose Alice's identity (account number) is u_1, then the fixed number $m_0=g_1^{u1}g_2$, the actual message Alice uses is $m=m_0^s=g_1^{s*u1}g_2^s$. In the payment protocol, Alice has to reveal some information of the representation of m, i.e. $(s*u_1,s)$. Comparing with the representation of m_0, which is $(u_1,1)$, we find $(s*u_1)/s=u_1/1$, which means Alice's identity is still there: we blinded the appearance of the coin, but not the internal structure of the coin. Hence the name "restrictive blinding".

4.4 Double spending

The same as in discussion of [CFN90], here we need to encode the user's identity into the digital coin, and force Alice to reveal half of this piece of information each time she makes a purchase.

In the previous section, we showed how Alice's identity is encoded in m_0 $(=g_1^{u1}g_2)$, thus in any m $(=m_0^s=g_1^{s*u1}g_2^s)$. Later when using a coin, the Merchant in addition to verify the Bank's signature on the coin, will also ask Alice to prove her knowledge of representation of m, i.e. to reveal part of her identity information.

The basic protocol for proving knowledge of representation is as follows. Suppose Alice knows $m=g_1^{a1}g_2^{a2}$, she wants to prove her knowledge of representation to Merchant:

1) Alice chooses random numbers $x_1,x_2\in_R Z_q$, computes $B= g_1^{x1}g_2^{x2}$, sends B to Merchant
2) Merchant generates a random challenge $d\in_R Z_q$, sends to Alice
3) Alice computes $r_1=x_1+d*a_1$, $r_2=x_2+d*a_2$ (mod q), sends r_1,r_2 back to Merchant
4) Merchant checks $B*m^d=g_1^{r1}g_2^{r2}$

Note this basic protocol is a zero-knowledge proof. The random number B here protects Alice's secret. Merchant can learn nothing more than that Alice knows the representation of m. To give some information out, we can fix x_1 and x_2 (but still keep them secret), i.e. Alice always uses the same B in every purchase. This way after 2 executions of the payment protocol, Merchant can solve for Alice's identity using gathered information: suppose we collect (r_1,r_2) and (r_1',r_2'): $r_1'=x_1+d'*a_1$, $r_2'=x_2+d'*a_2$ (mod q). It's easy to verify $(r_1'-r_1)/(r_2'-r_2)=a_1/a_2$.

4.5 The complete Brands' system

Ok, time for the complete system:

Setup:

Bank's private key: x
Bank's public information: p, q, (g_1,g_2,g), $h=g^x$

Opening an account:

1) Alice chooses a random number $u_1\in_R Z_q$, sends $I=g_1^{u1}$ to the Bank
2) Bank stores I with Alice's identity, computes $m_0=I*g_2=g_1^{u1}g_2$, $z=m_0^x$, sends them back to Alice

Withdrawal protocol:

Here we use the restrictive blind signature scheme sketched above, the only change here is Alice generates random number x_1,x_2, computes B= $g_1^{x1}g_2^{x2}$, and includes B in:

$$C = H(m,B,z,a,b)$$

Now we can say sign(m,B)=(z,a,b,r), since $\log_g h=\log_m z$. Note the protection factor B used in the payment protocol is fixed here. The coin takes the form of (m,B,sign(m,B)), everything about the coin can be made public except the representation of m and B, i.e. (s, u_1, x_1, x_2). As a reminder, s is the blinding factor, ($u_1 s$, s) is the representation of m, (x_1,x_2) is the representation of B. The knowledge of the representation proves ownership of the coin, and thus should be kept secret.

Payment protocol:

1) Alice shows to Merchant (m,B,sign(m,B))
2) Merchant checks the signature is valid
3) Merchant then generate a random challenge d, sends to Alice
4) Alice computes $r_1=x_1+d*(s*u_1)$, $r_2=x_2+d*s$ (mod q), sends r_1,r_2 back to Merchant
5) Merchant checks $B*m^d=g_1^{r1}g_2^{r2}$, i.e. Alice knows the representation of the coin.
6) Merchant stores the coin and (r_1,r_2), later sends them to Bank for credit.

If Alice double-spends, $(r_1'-r_1)/(r_2'-r_2)=u_1$ reveals her identity.

4.6 Extension to trustee-based tracing cash

[BGK95] extended Brands' system to include trustee-based tracing. The idea is the same. The user has to split the coin's representation into two parts, and gives each public trustee one part. During the payment step, the user has to reveal part of representation to the shop. The shop is going to send this part information with the coin together to the bank during deposit step. Thus the two trustees and the bank together can trace the user, even for the user who only uses a coin once.

The paper doesn't have any new idea. The security, the user's privacy and the system' functionality are the same as that of Brands, but it needs more computational effort in each step of building the system.

5 Wallet with observer

Chaum has made a clear introduction to wallet with observer in [Cha92]. It can be incorporated into any of the above anonymous ecash systems. Here we only introduce how to implement it in [Bra93].

1. During the opening account step, the user generates at random u_1, the bank store a random number o_1 in the observer. The user's identity is $I = g_1^{u1+o1}$.
2. During the withdrawal step, whenever the user wants to get a coin (a signature) from the bank, the user chooses a number m such that $m = m_0^s$, where $m_0 = g_1^{u1+o1}g_2$. Since the user doesn't know o_1, he needs the observer's help. The observer, without reveal its secret number o_1, generates at random a number o_2, sends the value g_1^{o2} to the user.
   A coin (m, B, sign(m,B) = (z, a, b, r)) is the same as before, where
   $$B = g_1^{x1}g_2^{x2}g_1^{f(o1)}g_1^{o2}$$
   Here $f(o_1)$ is some function of o_1 . The signature is only on m in the sense: $\log_g h = \log_m z$.
   The role of B is to protect the user and the observer's secret number o_1 and u_1. B is different from the basic cash system, it's for the merchant to later check if it's a correct coin and if the user knows the representation of the coin using the formula:
   $$g_1^{r1}\, g_2^{r2} = m^dB$$

3. During the payment step, when the user wants to spend a coin, the observer is going to check if this number o_2 is still in the memory. If it's not, this means the user tries to double spend this coin, the observer just locks, the user can't spend the coin; if the number o_2 is still in the memory, the observer sends the necessary information to the user, the user can spend the coin. This can prevent double spending.
4. For the honest user, the privacy is computational protected; for double spender, the bank can compute his account number even the user broke the observer and double spent the coin.

6 Conclusion

Basically we've vivisected several anonymous digital cash systems, it should be clear now how to construct a digital cash system based on some low level crypto techniques. We extended [CFN90] system to include trustee-based tracing as an example, although it may not always be that trivial. As said before, formal ways of verifying the soundness of an ecash system are, we believe, as difficult as theorem proving.

[Bra93] and [OO92] gave some discussion of efficiency, like in [Bra93], he argued the three-move withdrawal protocol is more efficient than cut'n'choose protocol. But we feel it immature to give any discussion on feasibility or efficiency issues, simply because that's heavily dependent on the underlying crypto techniques. Things like difficulty of calculating discrete logarithms are just untested assumptions, they're research topics on their own.

The independence property, as we pointed out, is actually not fulfilled, unlike what's been claimed in some of the papers; it's simply impossible.

Wallet with observer and trustee-based tracing are two important properties for ecash system to be successfully deployed and accepted in real world.

7 On the real life front

A dutch company Digicash, which Chaum is affiliated with, has already launched a trial run of their system. Deutsch Bank, with its 1500 customers, and 25 online vendors are experimenting with it now. From their website (http://www.digicash.com) I can say it's based on Chaum's scheme, the actual underlying signature scheme might be cubic root or representation problem in groups of prime order. Some interesting points there are:

- coin denomination distribution: There're coins of different face value, and it's the user's responsibility to maintain small changes in their pocket. But they provided a coin denomination distribution as a reference for the user to follow.
- dispute resolving: When dispute occurs between customer and merchant, the user can choose to cancel a payment. It in effect equals double-spending, but is necessary since the user has to prove her ownership of the coin in order to cancel. The merchant has no way of knowing whether the user received the goods or not from the Ecash system, so there has to be other means to prevent Alice from cheating.

Cybercash (http://www.cybercash.com) also offers a variety of different ecash solutions, including both online and offline ecash systems. Cybercash recently acquired the right to use NetBill system by CMU, an online payment system dedicated to facilitate micropayment.

References

- [Cha83] D. Chaum. Blind Signature for Untraceable Payments. Advances in Cryptology - Proceedings of CRYPTO 82, pp199-203.

- ✓ [CFN90] D. Chaum, A. Fiat, M. Naor. Untraceable Electronic Cash. Advances in Cryptology - Proceedings of CRYPTO 88, pp319-327.
- ✓ [Cha92] D.Chaum. Achieving Electronic Privacy. Scientific American, Aug. 92
- ✓ [CP92] D. Chaum, T. Pedersen. Wallet Databases With Observers. Preproceedings of Crpto92, pp89-105.
- ✓ [OO92] T. Okamoto, K. Ohta. Universal Electronic Cash. Advances in Cryptology - Proceedings of CRYPTO 91, 1992, pp324-337.
- ✓ [Bra93] S. Brands. Untraceable off-line Cash in Wallets with Observers. Proceedings of CRYPTO 93, pp303-318.
- ✓ [Bra93bis] S. Brands. An Efficient Off-line Electronic Cash System Based On the Representation Problem. CWI Technical Report CS-R9323, April 11, 1993
- ✓ [BGK95] E. Brickell, P. Gemmell, D. Kravitz. Trustee-based Tracing Extensions to Anonymous Cash and the Making of Anonymous Chang. 1995