Traceable Anonymous Cash

Recall from last time we described traceable anonymous cash. The idea is to hide the spender's identity in the note in such a way that if they spend the note once, they will remain anonymous. But if they try to spend multiple times, their identity will almost certainly be revealed.

Not surprisingly, we use secret sharing to do this. Recall that it is possible to take an n-bit message M and make two n-bit messages each of which contain no information about M but which together completely define M. Here is the procedure:

1. The customer makes j copies of his/her identity and splits each one in two halves. That is, the copies are $\{Id_1, \ldots, Id_j\}$ and $\{Id'_1, \ldots, Id'_j\}$. The owner's identity will be computable from any pair (Id_i, Id'_i) but from no other combination.

2. These Identities are encrypted each with a different key and become part of the users "Identity info". Each result would look like $f(Id_iG, K_i)$ or $f(Id_iG, K'_i)$, where f is an encryption function, and G is a "recognizable" string like the name of the bank. It is needed because Id_i is itself just a random string, and decrypting it with a false key would be hard to detect. The keys K_i and K'_i are bit-committed (e.g. by hashing) but kept secret.

3. A customer requesting a unit of cash creates k samples containing:
   $(Bank'sName, Amount, SerialNumber, Identityinfo)$
   where the serial number is unique to each bill, and the identity info is computed separately for each note using steps 1 and 2 above. The k units of cash are blinded using blinding factors $b_1, \ldots, b_k$ and presented to the bank.

4. The bank selects one unit at random and sets it aside. Then it asks the customer to unblind the other $k-1$. The customer provides the blinding factors and all the keys K_i and K'_i for those notes, and the bank looks inside. For each note, the bank checks that all the identity halves Id_i and Id'_i gives the owners identity.

5. The bank signs the unit it had set aside, and gives it to the customer.

6. The customer unblinds this unit and is ready to spend.

Now when the customer takes this note to a merchant, an interactive transaction takes place. In order for the merchant to accept the note, the customer must prove that it is their money. That is, they must expose some of the secret keys K_i and K'_i.

To make a purchase, the customer would conduct this transaction:

1. Customer arrives at the store with the bill. Merchant tosses a coin j times and tells the customer the outcomes.
2. for the ith coin toss, the customer reveals Id_i if the toss was heads, or Id'_i if it was tails by revealing the encryption key K_i or K'_i. The merchant gets one half of each identity, but that reveals nothing.
3. The merchant checks the banks signature and forwards the bill along with the coin flips and the keys K_i or K'_i that were revealed to the bank.

The customer can't avoid revealing half of each identity for each coin toss. First of all, the identity pieces are almost certainly real. The bank checked the other $k-1$ notes before it signed this one, including checking that the identity halves matched. It is very unlikely the customer could have cheated when they created the note and got away with it.

The customer can't cheat at step 2 by using a false K_i or K'_i because the keys were bit committed when the note was created (and before the bank signed it). The string G confirms that the decryption worked.

Suppose the customer tries to spend the same note again. Another merchant will toss a coin j times. The customer must reveal j identity halves, and the merchant will forward these (actually their keys) to the same bank. Unless the customer is extremely lucky, the sequence of coin tosses by the second merchant will be different from the first merchant. Suppose toss i is different between merchant 1 and merchant 2. That means the customer revealed both Id_i and Id'_i. When the bank receives those pieces, it will be able to recover the customers identity. It will also know that the customer deliberately tried to spend twice.

What if the merchant tried to cheat? If they just copy the information that they got from the customer and go to the bank with it, the bank will suspect the merchant. It is very unlikely that a real transaction would happen where the merchant came up with exacltly the same sequence of coin tosses. Can the bank forge some identity keys? There is really no way to do this. The keys K_i are all bit committed using a hash function, and the result is signed by the bank. So the merchant can't fake other keys with the same hash value.

The real risk to the customer is that the keys K_i must be stored in some kind of computer. If that is stolen, a criminal could try to spend the money twice and only the identity of the original owner would be revealed. But overall this is a pretty secure approach.