# CS174 Spr 99 Lecture 28 Summary John Canny

## Blinded Digital Signatures

We need one more general concept before we dive into digital cash. A blinded digital signature is something like signing a blank check, or actually a blank piece of paper. It doesnt sound like a good idea, but as we will see it is very useful. Blinded signatures are one of several techniques used to preserve privacy with digital cash. They make it harder to trace the path of some cash.

Let $M$ be a message which is a bank note or check. A bank could sign this in the RSA signature scheme by first computing a secure hash $H(M)$ of the message, and then computing $H(M)^d(modn)$ as the signature, where $d$ is the banks private RSA key. Anyone could verify given $M$ and the signature, that the bank had meant to sign this note. The problem with this is that it gives banks great power to trace your spending. Since they issue the note to you in the first place, but will receive the note for collection from a merchant, they could match the two. That is, they could save $M$ in a database indexed with your name and then match the $M$ that the merchant brings in against it. This happens with credit card transactions, and people often prefer cash because it is not traceable this way.

The anonymity is accomplished with a blinding factor $k$, which is known to you only. You ask the bank to sign:

$$r = H(M)k^e (\bmod\ n)$$

where $e$ is the bank's *public* RSA key. What you get back is

$$r^d = H(M)^d k (\bmod\ n)$$

and you can multiply this by $k^{-1}$ to get back $H(M)^d (\bmod\ n)$ which is the signed note. The bank only saw $H(M)k^e$ which is difficult to distinguish from random data.

But how do you convince a bank to sign a blank piece of paper? They might be commiting themselves to pay a huge amount of money. The answer is very similar to a zero-knowledge proof (c.f. the ZKP of graph coloring). Instead of one note, you present the bank with many notes, all with the same value (say $10). That is, you give the bank:

$$r_i = H(M)k_i^e (\bmod\ n)$$

for $i = 1, \ldots, k$, and the same message $M$. The blinding factors $k_i$ are different in each case. Then the bank picks one of the notes at random and sets it aside and asks you to unblind the other $k-1$. You send the bank the $k_i$ factors for notes it has asked for. The bank unblinds those notes by multiplying by $k_i^{-e} (\bmod\ n)$, and checking for a match with $H(M)$. The bank discovers through this process that you were honest in filling out all $k-1$ of these notes. So it agrees to sign the other note (that it set aside) without seeing it. The bank doesnt know what this note actually contains. But there can be at most one bad note, and the probability that the bank picks it is

$\frac{1}{k}$

This is not much protection, but we will see some other schemes for protecting the bank later.

## Digital Checks

If we made a digital version of a classical personal check, it would look something like this:

$\text{Signed}_{Owner}(BankName, Owner'sName, Amount, RecipientName)$

In order to cash the check, someone would present it to a bank. In general, the bank where the check is collected will be different from the bank that actually provides the funds, which is normally your bank. The chain of signatures that gets applied to the check looks like this:

$\text{Signed}_{Owners\ bank}(\text{Signed}_{Collection\ bank}(\text{Signed}_{Recipient}(\text{Signed}_{Owner}(Data..))))$

The nesting of the signatures proves that they were applied in the correct order. This is both stronger and weaker than an ordinary check. Stronger because it is hard to forge digital signatures, but weaker because it is easy to make multiple copies of the check at any stage.

### Adding Privacy

There are several reasons for having a cash-like form of digital money. The first which we have already noted is privacy. We would like to hide the identities of the parties to a digital exchange. A first attempt might be to use a check made out to "cash".

$\text{Signed}_{Owner}(BankName, Owner'sName, Amount, Cash)$

This provides some protection for the recipient, but it is still hard to hide their identity because someone must present the check to a bank for collection. The collection bank could record information about that person at the time of collection.

Protecting the owner's identity is harder. One solution is to use the blinded signature technique described earlier. Through that scheme, the bank signs a check for which it has no record of the Owner's name (you would probably have to give the bank real cash in order for them to do this). This is good protection for the owner and recipient, but because of that, it is more susceptible to fraud, especially double spending. The recipient of a normal check is identified on the check and so runs a risk if he/she tries to spend it twice. But with anonymous or cashier's checks, the recipient might feel emboldened to cheat.

### Traceable Anonymous Cash

Traceable anonymous cash sounds like an oxymoron. But in the digital world, remarkable things are possible. Specifically, we can hide someone's identity in a 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.