**CS174** **Lecture 28** **John Canny**

## Digital Checks

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

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

We'll assume that the signature includes the original document, i.e. that $\text{Signed}_{Owner}(X)$ means $(X, R(H(X), d))$ where $R(Y, d) = Y^d (\bmod\ n)$ is the RSA en(de)cryption function, and $H(X)$ is a secure hash function such as MD5. The Owner's signature proves to the bank that they want to withdraw this money.

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. For example, the recipient signs the string $(X, S_O(X))$, where $S_O(X)$ is the owner's signature. This nesting scheme proves that the signatures were applied in the correct order, not some other order that might enable forgery.

The serial number is important to make each check unique. Without it, it would be impossible to create two checks from the same owner to the same person for the same amount. The second check would be an identical bit string, including the Owner's signature. It would look exactly like an illegal copy of the first check.

This digital check 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.

If we assume that clearing a digital check is like clearing a regular check, i.e. it takes several days[1], then this scheme is quite safe. Although the check may be copied, it will not be cashed a second time because the Owner's bank will refuse to clear the funds. The second "check" will bounce and prove that someone cheated.

### Adding Privacy: First version of digital cash

There are several reasons for having a cash-like form of digital money. The first and most important is privacy. We dont want a lot of agencies to be able to monitor how the user is spending their

[1] In the digital world, there should be no need for delays like this. Assuming bank databases are reliably online and have suitable automatic protocols for approving overdrafts etc., it should be possible to reduce this to matter of minutes with the same series of confirmations

money. So our goal as digital cash designers is to hide the identities of the parties to a digital exchange. A first attempt might be to use a check made out to "cash".

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

This provides some protection for the recipient, but it doesnt prove to a merchant that that person has the money in their bank account. It also discloses the identity of the spender. Ideally, we want to remove the owner's name and their signature. We have to provide some other authentication to prove that this note is worth something (i.e. someone is committed to paying the stated amount). The owner's bank's signature is suitable.

To remove the owner information, we would create a note like this, and send it to the bank:

$$\text{Signed}_{Owner}(BankName, Amount, SerialNumber)$$

The Owner's name does not appear in the note body, but is still in the signature. This is necessary to prove to the bank that the owner really wants to spend the money, and in effect authorizes the bank to dispense that money from the owner's account. Then the bank would issue a check which is like a cashier's check to the owner:

$$\text{Signed}_{Bank}(BankName, Amount, SerialNumber)$$

Note that the bank has signed the original check less the owner's signature. This is necessary to preserve the privacy of the owner. Now there is no information in the check about the owner at all.

When the bank issues this check, it should debit the owner's account, rather than waiting for collection as with a normal check. That's because the owner won't be identified at collection time, and the bank won't know which account to debit. Where this money "goes" is an esoteric question. In practice, its likely that when the check is issued, money is moved from the owner's account (really a pool of owner accounts) to a general "payment" account which is used for payment of cashier's checks.

There is one complication with this scheme: because we dont know the owner, the serial number must be unique across all bank customers and all checks. So we should pick a serial number at random from a large enough set that no other user is likely to pick the same number. By the birthday paradox, this probability is low as long as the set of possible numbers is larger than the square of the number of possible checks. e.g. $256$-bit serial numbers should be enough.

The bank's signature verifies that there is money in its payment account to honor this check. When the bank receives notes for collection, it checks the serial numbers against the ones that were presented before. If it is not a duplicate, the bank will pay the stated amount to the recipient.

This note is reasonably safe against fraud (via serial number checks) and appears to protect the identities of owner and recipient. But the anonymization of the owner is illusory. The serial numbers give the bank an easy way to keep track of what checks each user has withdrawn. It can be combined with other information, e.g. collected by online vendors, to track what each user is buying. Banks are motivated to share this information with other banks and with certain merchants because it provides some protections against fraud and allows them to gather useful information about their customers such as creditworthiness.

To take the idea of anonymous cash further, we would like to use cryptographic ideas to ensure that the bank cannot track notes at all. You might argue that it would be easier to use legislation to prevent the banks from sharing this information. This is some merit to this argument, but laws can change and e.g. even if this data is protected by default, the protection may be overridden if the data is relevant to a criminal prosection. Also, the class of agencies that can serve as “banks” is much larger if we can build protocols that enforce good behavior, rather than depending on law enforcement and regulation of those agencies.

The key to preventing the bank from tracing the notes is to “blind” the bank’s signature, in effect making the bank sign a note without seeing it. Thus the bank doesnt know the serial number, and has no way to trace the note.

## Blinded Digital Signatures

A blinded digital signature is something like signing a blank check, or like signing the envelope containing the check. 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 $H(M)^d (\mathrm{mod}\ n)$ 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 apply its RSA decryption function to:

$$r = H(M) k^e (\mathrm{mod}\ n)$$

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

$$r^d = H(M)^d k (\mathrm{mod}\ n)$$

and you can multiply this by $k^{-1}$ to get back $H(M)^d (\mathrm{mod}\ n)$ which is the signature part of 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 discrete log). Instead of one note, you present the bank with many notes, all with the same value (say $10). That is, you give the bank:

$$r_i = H(M) k_i^e (\mathrm{mod}\ n)$$

for $i = 1, \ldots, m$, 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 $m-1$. You send the bank the $k_i$ factors for notes it has asked for. The bank unblinds those notes by multiplying by $k_i^{-e} (\mathrm{mod} \ \ n)$, and checking for a match with $H(M)$. The bank discovers through this process that you were honest in filling out all $m-1$ 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

$$\frac{1}{m}$$

This is not much protection for a given amount of effort. We can turn it into an exponential amount of protection by requiring the note itself to consist of multiple copies for the same amount. e.g. suppose in the scheme above the final “note” consisted of half of the $m$ pieces described above. The bank would set $m/2$ aside, ask to see the other $m/2$, and assuming they were all OK, it would sign the $m/2$ that it had set aside. The final note would be a union of the $m/2$ pieces. When it was spent, all m/2 pieces would have to have the same information except for serial number, or the note would not be accepted. Therefore if any of the $m/2$ pieces were invalid the note would be invalid. Check for yourself that the probability of the owner being able to cheat succesfully is $2^{-\Omega(m)}$.

## Anonymous Digital Cash

To have real cash, we need better protection of the owner’s identity so that their spending cannot be tracked. One solution is to use the blinded signature technique described earlier. Through that scheme, the bank signs a note for which it has no record of the Owner’s name. In more detail, you the spender create $m$ notes of the form

$$M_i = (BankName, Amount, Serial_i)$$

and you choose $m$ random blinding factors $K_i$, one for each note. The serial number is different for each note. You keep all this information secret from the bank. To prove to the bank you arent cheating, you would bit-commit to these values by sending the bank hashes of each pair:

$$H(M_1, K_1), \ldots, H(M_m, K_m)$$

You also send to the bank the data that you want it to “sign” (by decryption), namely:

$$K_1^e H(M_1), \ldots, K_m^e H(M_m)$$

The bank choose one of these notes, say the $l^{th}$ note, and asks you to unblind the other $m-1$. You give the bank the blinding factors $K_i$ and the serial numbers for those notes only. The bank checks that the unblinded notes are all of the form

$$\mathrm{Signed}_{Owner}(BankName, Amount, Serial_i)$$

by checking that the hashes of these strings match the bit committed hashes. You can also check that the signature data which should be of the form $K_i^e H(M_i)$ are valid.

The bank takes this as strong evidence that the $l^{th}$ note is also of this form, so it signs that note by applying RSA decryption to it, and returns it to you. You remove the blinding factor, and you now have a note of the form

$$\mathrm{Signed}_{Bank}(BankName, Amount, SerialNumber)$$

for which the bank does not know the serial number. The bank cannot infer the serial number either. Although the bank sees the hash $H(M_l, K_l)$, it doesnt know the blinding factor (which comes from a huge space), it cant find the serial number by enumeration. This is a truly anonymous note.

As we argued in class, digital protocols are most suitable for single-step transactions. i.e. you would use an anonymous note to pay another person, who would deposit that note themselves. It is not a good idea to pass such a note to third person because it is impossible to tell if it has been spent except by trying to spend it. The more people who touch the note, the more chances for fraud, and the harder to trace the dishonest party.

So far we have a method that gives very good privacy protection for the owner and recipient, assuming they are the only two people involved. But because of that, it is still tempting for one of those parties to cheat, especially by double spending. The serial numbers provide some bookkeeping to show that someone has cheated, but its impossible to tell who duplicated the note. Next time we will describe a form of digital cash which is both anonymous *and* (surprisingly) traceable! It uses an interesting form of secret-sharing embedded in the note itself.