IT486 v3.0

Before Bitcoin: 1990s Ecash

First Attempt

- Let's try to come up with a cryptographic protocol for digital cash
- Paper money started out as a promise by a bank:

"I promise to pay the bearer on demand the sum of . . ."

First Attempt

- Can we have a digital equivalent of this?
  1. Alice goes to bank, handing over \$1000
  2. Bank creates message m: pay bearer \$1000
  3. Bank signs m with its private key
  4. Bank gives signed m to Alice

First Attempt

- Can we have a digital equivalent of this?
  1. Alice goes to bank, handing over \$1000
  2. Bank creates message m: pay bearer \$1000
  3. Bank signs m with its private key
  4. Bank gives signed m to Alice
- Alice can now spend it
- A merchant can check the validity (assuming the public key of the bank is known)
- Merchant cashes in the signed note with the bank

Attack: Double Spending

- Now we have a problem
  - What prevents Alice from spending it multiple times?
  - Or Merchant from showing up with the same message multiple times to cash it in?

Attack: Double Spending

- Now we have a problem
  - What prevents Alice from spending it multiple times?
  - Or Merchant from showing up with the same message multiple times to cash it in?
- Bank could add a serial number to the message m
- Each “digital bank note” is allowed to be cashed in only once

Attack: Double Spending

- Now we have a problem
  - What prevents Alice from spending it multiple times?
  - Or Merchant from showing up with the same message multiple times to cash it in?
- Bank could add a serial number to the message m
- Each “digital bank note” is allowed to be cashed in only once
- Everything fine now?

Anonymity is broken

- The bank knows which messages were spent by Alice
  - it generated the serial number

Anonymity is broken

- The bank knows which messages were spent by Alice
  - it generated the serial number
- To restore anonymity, we need a so-called "blind signature"
- Let's illustrate this with paper documents

Blind Signatures

1. Alice creates x messages for \$1000 each
2. Puts each one in a separate envelope lined with carbon paper and gives them to the bank

Blind Signatures

1. Alice creates x messages for \$1000 each
2. Puts each one in a separate envelope lined with carbon paper and gives them to the bank
3. Bank randomly opens $x - 1$ of them, verifying that each is for \$1000
4. Bank signs the last envelope blindly without opening it, deducts \$1000 from Alice's bank account
5. Bank hands unopened envelope to Alice

Blind Signatures

1. Alice creates x messages for \$1000 each
2. Puts each one in a separate envelope lined with carbon paper and gives them to the bank
3. Bank randomly opens $x - 1$ of them, verifying that each is for \$1000
4. Bank signs the last envelope blindly without opening it, deducts \$1000 from Alice's bank account
5. Bank hands unopened envelope to Alice

- Payment and cashing in as before

Blind Signatures

- Bank never sees what it signs
- Assume Alice wants to cheat (by putting different amounts in the envelopes):
  - Bank opens envelopes randomly
  - With a probability of $\frac{x-1}{x}$ bank can detect this fraud

Blind Signatures

- Bank never sees what it signs
- Assume Alice wants to cheat (by putting different amounts in the envelopes):
  - Bank opens envelopes randomly
  - With a probability of $\frac{x-1}{x}$ bank can detect this fraud
- This is anonymous, but we have re-introduced the double spending

User generated serial number

- Alice can add the serial number herself
  - Every message is appended by a different long random string
- By opening $x - 1$ envelopes bank can see those serial numbers
  - but not the one it actually signs
- Now, if someone tries to cash in a message twice, bank can detect it

Blind Signatures

- What do blind signatures look like in terms of cryptography?
- Protocol between (R)eceiver and (S)igner:

Blinding	R prepares blinded message $M' = b(M)$.
Signing	R sends M' to S.
	S replies with signature σ' on M'.
Unblinding	R extracts signature σ for M from σ'.

Further Issues

- The bank is protected from cheaters, but we cannot identify them
- When double-spending is detected, we don't know whether
  - someone tried to cheat a merchant
  - a merchant tried to cheat the bank

Further Issues

- The bank is protected from cheaters, but we cannot identify them
- When double-spending is detected, we don't know whether
  - someone tried to cheat a merchant
  - a merchant tried to cheat the bank
- There are more complicated versions of the protocol that try to identify cheater
- Another solution is for the merchant to check with the bank before accepting a message

Recap: Commitment

- Temporarily hide a value, but ensure that it cannot be changed later
- 1st stage: commit
  - Sender electronically "locks" a message in a box and sends the box to the Receiver
- 2nd stage: reveal
  - Sender proves to the Receiver that a certain message is contained in the box

Properties of Commitment Schemes

- Commitment must be hiding
  - At the end of the 1st stage, no adversarial receiver learns information about the committed value
- Commitment must be binding
  - At the end of the 2nd stage, there is only one value that an adversarial sender can successfully "reveal"

Bit commitment from a hash function

	Alice		Bob
	To commit(b):		
1.	Choose random string r. Compute $c = H(r \parallel b)$.	$\xrightarrow{c}$	c is commitment.
	To open(c):		
2.	Send (r, b).	$\xrightarrow{r,b}$	Verify that: $c = H(r \parallel b)$.

Why is r needed?

Chaum's offline ecash (1983)

Basic idea:

- Make the user embed his identity in each coin (hidden).
- Introduce a challenge-response phase in payment protocol such that:
  - A single run of the protocol leaks no information about user
  - The result of two runs of the protocol leaks the user identity

Chaumian ecash – Withdrawal phase

- User prepares k \$1000 dollar bills:

  $M_i = \{$ \$1000 bill, $\sharp\{serial\}_i,\ y_{i,1},\ y'_{i,1},\ y_{i,2},\ y'_{i,2},\ \ldots,\ y_{i,n},\ y'_{i,n}\ \}$

  where $y_{i,j} = H(x_{i,j})$; $y'_{i,j} = H(x'_{i,j})$ with randomly chosen pairs $x_{i,j}$, $x'_{i,j}$ such that

$$x_{i,j} \oplus x'_{i,j} = id \ \ \forall i, j$$

Chaumian ecash – Withdrawal phase

- User prepares k \$1000 dollar bills:

  $M_i = \{$ \$1000 bill, $\sharp\{serial\}_i$, $y_{i,1}$, $y'_{i,1}$, $y_{i,2}$, $y'_{i,2}$, $\ldots$, $y_{i,n}$, $y'_{i,n}$ $\}$

  where $y_{i,j} = H(x_{i,j})$; $y'_{i,j} = H(x'_{i,j})$ with randomly chosen pairs $x_{i,j}$, $x'_{i,j}$ such that

  $$x_{i,j} \oplus x'_{i,j} = id \;\; \forall i, j$$

- User blinds all M_i and sends them to bank
- Bank asks to unblind $k - 1$ random M'_i
- User also returns all $x_{i,j}$, $x'_{i,j}$ for unblinded M_i.
- Bank checks that amount is correct and that

  $$x_{i,j} \oplus x'_{i,j} = id \;\; \forall i, j$$

- Bank returns signature on remaining blinded M_i

Chaumian ecash – Payment phase

- User pays Payee with bill M_i
- Payee sends random n-bit string $b_1, \ldots, b_n$ (Challenge)
- For each bit b_j, user replies with $c_j = x_{i,j}$ if $b_j = 0$, and $c_j = x'_{i,j}$ otherwise (Response)
- Payee accepts if $H(c_j) = y_{i,j}$ (or $H(c_j) = y'_{i,j}$, respectively) for all j and the signature is valid.

Chaumian ecash - Deposit Phase

- Payee gives bill M_i, bitstring b and openings c_j to bank
- Bank checks signature and looks for serial number in database
- If signature is invalid or openings are invalid then bank aborts

Chaumian ecash - Deposit Phase

- Payee gives bill M_i, bitstring b and openings c_j to bank
- Bank checks signature and looks for serial number in database
- If signature is invalid or openings are invalid then bank aborts
- If serial number is not in database, bank credits payee account and adds serial number, bitstring and openings to database.

Chaumian ecash - Deposit Phase

- Payee gives bill M_i, bitstring b and openings c_j to bank
- Bank checks signature and looks for serial number in database
- If signature is invalid or openings are invalid then bank aborts
- If serial number is not in database, bank credits payee account and adds serial number, bitstring and openings to database.
- If serial number is in database:
  - if openings are the same: Bank knows payee misbehaved
  - if openings are different: Bank can learn user identity

Chaumian ecash - Security

- If H is one-way, righteous users remain anonymous
- If H is one-way, no one besides user can solve challenge
- If H is collision-resistant, a misbehaving user is caught with probability $1 - 2^{-n}$.

Review Q

True or False:

- Suppose a particular merchant X has faulty software that causes him to always send the same challenge bitstring to the customer. If the customer notices this and is dishonest, he could conspire with another merchant Y to make it look (to the bank) like merchant X is trying to cash the same coin twice.

Review Q

True or False:

- Suppose the customer by mistake emails to Alice the coin withdrawn by him from the bank (not yet spent by the customer). The customer is unaware that he made this mistake, and does not send Alice anything other than that coin. Then Alice can rush to a merchant and, if she does so, before the customer spends his coin, she can successfully spend the coin, and the customer is later accused of cheating when he tries to spend the same coin.

Required Reading

- Ogiela and Sulkowski, Improved Cryptographic Protocol for Digital Coin Exchange, SCIS&ISIS 2014.
- Can be found in class folder or at the following link
  - https://doi.org/10.1109/SCIS-ISIS.2014.7044796