**Adwebsec Lecture 1 – 4. November 2020**

**Part 3 – Untracable Electronic Cash**

Traceability possible with e-cash compared to physical cash.

**How can we achieve anonymity with e-cash?**

1. Alice contacts the bank to withdraw eCoin
2. Bank sends Alice the eCoin
3. Alice transfers the eCoin to Bob
4. Bob deposits or redeems the eCash back to the bank

**eCash Basic Principles:**

1. Only the bank can generate eCash
2. Prevent double spending / money cloning
3. ..achieve same untraceability as physical cash

How to make sure that the bank doesn’t know that the money that Bob deposits is not the same coins that Alice withdrew? We can do this by using blind signatures.

**How can the bank be the only one to generate eCash?**

The solution is to use signatures, the bank has a secret key and each key has signatures.

**How to prevent double spending?**

1. Report to the bank every coin spent.
2. Remove anonymity only when double spending

**How to prevent expense tracing?**

We need the bank to sign the coin, but the bank should not know what coin.

**Blind RSA signature – Coin withdrawal**

Alice holds eCash X and bank hold signing secret key sk = d.

1. In order to spend x, she has to get a signature
2. Picks a random message r, to blind the signatures, so the bank doesn’t know what it is signing
3. Takes the r and multiplies it by the hash of the coins, called the value B.
4. Sends B to the bank, the bank has the secret key d, uses this to sign the blinded message B, creating the signature S-bar
5. Sends S-bar to Alice.
6. Alice can now extract a signature for coin x from this S-bar.
7. By multiplying S-bar times the inverse of the randomness r that she chose in the beginning to blind the message.
8. Alice has successfully withdrawn the coin x(S is a valid signature for x, but the bank has never seen coin x).

**Spending coins. (Alice, Bob and Bank)**

1. Alice wants to pay Bob for an item or service. She has coin x and signature S.
2. Send signed coin (x, S)
3. Verify signature to make sure it is a legit coin
4. Send (x, S) and ask if it is and if has been spent already or not.
5. Bank responds yes or no

Not practical to store every single coin value at the bank. It would be a massive database keeping track of this. The bank doesn’t know who coin x belongs to either, so punishing the double spender is impossible as they are anonymous. We need to change the withdrawal protocol to change this feature.

**Cut and choose:**

Cut and choose is an interactive protocol to properly securely check that a party has produced data in the way it was supposed to.

This party is called the “verifier”. We have an prover who wants to prove something to the verifier   
if the verifier trusts the “prover” there is no need to check the “send message”  
but if we say the prover can misbehave, the verifier can open the message/box.

Instead of opening the original box/message, ask the prover to send multiple messages cut up, the prover will then pick some of these boxes/messages and send these back, asking the prover to open these “boxes” (say 1 and 4). It will send the opened content, which then the verifier can verify with the boxes it received earlier. It still doesn’t know what is in the other unopened boxes, but the chances of there being malicious content is low, as the prover didn’t know beforehand which content the verifier would choose.

A better untraceable eCash protocol – withdrawal

Alice:

1. Pick 2k 4-tuples of random numbers (ai, bi, ci, ri)2k i=1
2. Then she computes two values:
   * Xi = h(ai, bi)
   * Yi = h(ai xor IDA, Ci)
3. Now Alice computes the blinding factors. 1 blinding factor per fordable
   * Bi = rei h(Xi, Yi) mode n.
4. The bank probabilistically verify that Alice has put her identity in every blinded value using the Cut-and-Choose technique.
   * Pick k random indexes I = {i1, I2, … ik}
5. Alice must reveal the asked values provided in indexes I from the bank. She must reveal the information inside these tuples which means that she would need to unblind values before sending them.
6. Re-compute the Bi for each i in I and check that they contain IDA. If Alice did not cheat, sign the blind value. And the Bank returns S-bar
7. Alice can unblind the S-bar by inversion and now she has a coin and a signature for the coin.

Spending:

1. Alice has coin x and signature X.
2. Alice send signed coin to Bob.
3. Not Bob will contact Alice on a random set of bits. (k random bits, each bit is between {0,1}).
4. Alice discloses the value depending whether each bit is 0 or 1, and using the index of the list to also find the index in Alice tuples.
5. Rj =

* (xj, aj xor IDA, Cj) If zj = 0
* (aj, bj, yj) if zj = 1

1. Alice responses does not leak her identity but revels some info about the Blinding.
2. Bob sends the list R to the bank to get it verified.

The bank can now figure out if the coin that Alice is using is being used for double spending, by using some probability calculations. Since Alice has provided either the a, b, c values or the xor’ed version of these values, then the chances are that if this coin has been used before, on of the corresponding values to (a, b, c) or (a xor ID) will have been sent in the messages when the coin was sent the first time.

Since the ID is xor’ed with the value aj, then we can find the ID by xor’ing it back. IDA hold the identity of Alice. This way the only way for the bank to find the identity of Alice can only happen if Alice double spends.