Digital Cash

- Here are some features of physical cash:
- Cash can be distributed between individuals whether you know them or not.
- It is extremely difficult to counterfeit cash.
- If the person who is spending or receiving cash can remain anonymous if they want.
- Although banks are convenient for storing cash, individuals can exchange cash without involving a bank.
- We will look at a situation in which we are really distributing files thought of as individual digital coins.
- The goal is to find a scheme for distributing digital coins so that as much as possible they replicate the above properties of cash.

The set-up

- We are going to need a few things to set-up a digital cash scheme; their purpose will be explained along the way.
- Fix two primes p and q so that $2q + 1 = p$. Usually one picks two such primes that are very large. It is not known if there are infinitely many such primes but there are definitely pairs of large such primes.
- We also pick g to be the square of a generator of F_p.
- We also pick two random powers of g. Call them g_1 and g_2. We destroy the exponents otherwise the system will be compromised. The numbers g, g_1 and g_2 are made public.
- We also fix two hash functions H and H_0 which take 5-tuples and 4-tuples of integers and produce numbers mod q.
- What's a hash function?

Hash functions

- In general, a function $f : A \to B$ is said to be a (strong) *hash function* it is easy to compute, the inverse is hard to compute and it is difficult to computationally show that f is not one-to-one.
- If f isn't one-to-one then by an inverse, we will mean that it is difficult to compute, for any element in the range b, any a in the domain such that $H(a) = b$.
- In practice, since the sets A and B we use need to be somewhat constrained and practical matters interfere, we need to loosen this definition. For us, A will be either 5(or 4)-tuples of integers mod p. B will be integers mod q.
- We will demand that we have two functions, one from Z_p^5 and one from Z_p^4 to Z_q which are computationally difficult to show are not one-to-one, are easy to compute and for which it is extremely difficult to compute "an inverse".
- Such functions exist and are commercially available.

The accounts

- An entity which wants to act as the bank chooses a secret number x and creates the numbers

$$h = g^x, h_1 = g_1^x \text{ and } h_2 = g_2^x \bmod p$$

  public. The numbers h, h_1 and h_2 act as the name of the bank. The bank does not reveal x.
- Anyone who wants to create a bank account chooses a secret number u and computes a bank account number $I = g_1^u$ which they send to the bank. The bank sends $z' = (Ig_2)^x \bmod p$ in response.
- The account holder does not reveal their number u to the bank or anyone else.

Making a purchase

- There are really three parts to this process:
- First of all, the person who wants to spend some money requests money from the bank. They wind up in possession of a file or number which represents a coin which they can use independent of a connection with the bank.
- Second, at some point after they get a coin, they can make a transaction with another person, again independent of the bank.
- Third, after the other party (the store or merchant) receives the coin from the purchaser, they can deposit it in the bank.

Withdrawing a coin

- A coin is represented by a tuple (A, B, z, a, b, r) where A, B, z, a and b are 5 numbers mod p and r is a number mod q.
- In order to create a coin, you request a coin from your account I. The bank picks a random number w which differs for each coin and computes

$$g_w = g^w \text{ and } \beta = (Ig_2)^w \bmod p.$$

  and sends these to you.
- You choose a random 5-tuple of integers $(s, x_1, x_2, \alpha_1, \alpha_2)$ and compute

$$A = (Ig_2)^s, B = g_1^{x_1} g_2^{x_2}, z = (z')^s, a = g_w^{\alpha_1} g^{\alpha_2} \text{ and } b = \beta^{s\alpha_1} A^{\alpha_2}$$

  all mod p. We don't allow $A = 1$ and so at the very least we don't allow $s = 0 \bmod p - 1$.

Withdrawing a coin, cont'd

- Notice that it is reasonably important to keep even your account number as secret as possible since anyone who knows the account number can request a coin. As we will see, they won't be able to spend it but they could mount a DoS attack on you and/or the bank.
- Now you compute $c = \alpha_1^{-1} H(A, B, z, a, b) \bmod q$ and send c to the bank.
- The bank computes $c_1 = cx + w \bmod q$ and sends you c_1. You now compute $r = \alpha_1 c_1 + \alpha_2 \bmod q$.
- The coin the bank just produced for you is (A, B, z, a, b, r).

Spending a coin

- You have a coin (A, B, z, a, b, r) and you give it to someone else we'll call the Merchant.
- The Merchant can check if this is a valid coin by computing and seeing if

$$g^r = ah^{H(A,B,z,a,b)} \text{ and } A^r = z^{H(A,B,z,a,b)}b \bmod p.$$

  If so, then this is a real coin. Why?
- The question now becomes: are you trying to spend this coin more than once?
- To check this, the Merchant uses the other hash function and computes

$$d = H_0(A, B, M, t) \bmod q$$

  where M is their account number with the bank and t is the time of the transaction. They send d to you.

Spending a coin, cont'd

- You compute

$$r_1 = dus + x_1 \text{ and } r_2 = ds + x_2 \bmod q$$

where u is the original number you used to create your account and s, x_1 and x_2 were part of your secret key to create the coin. You now send r_1 and r_2 to the Merchant.

- The Merchant checks if

$$g_1^{r_1} g_2^{r_2} = A^d B \bmod p$$

and accepts the coin if it is. If not, they reject it.

Depositing a coin

- If the Merchant wants to deposit the coin to their bank account, they tell the bank their account number, the coin (A, B, z, a, b, r) they are depositing and the numbers (r_1, r_2, d) that were produced during the transaction.
- The bank checks if they have the coin already deposited. If not, then the bank checks if

$$g^r = ah^{H(A,B,z,a,b)}, A^r = z^{H(A,B,z,a,b)} \text{ and } g_1^{r_1} g_2^{r_2} = A^d B$$

mod p. If this all checks out, the Merchant's account is credited with the money.