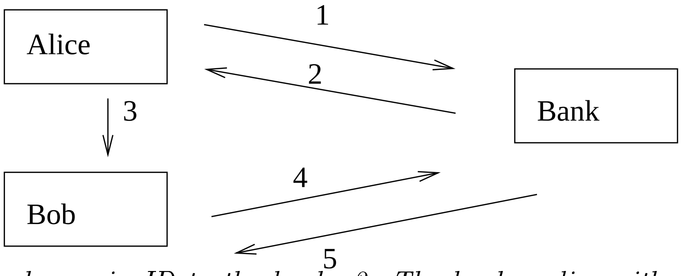
Electronic Cash

With the growth of the Internet there has been interest in commerce on it. Currently this is done with credit cards in combination with SSL to encrypt and authenticate Web connections. We would like functionality more like physical mechanisms, like checks. Check-like payment schemes can be done, and one present company providing such a system is PayPal. PayPal requires online transactions, but it can be made into an offline protocol by signing instructions to the bank, and giving them to the other party.

However, this is not an anonymous protocol: both parties must know who the other is, and the bank knows everything. Ideally, we would like something more like cash.

Proposal 1 **Strawman Scheme**



Above, 1. Alice sends a coin ID to the bank. 2. The bank replies with a signature and debits Alice's account 1 cent. 3. Alice sends the coin ID and its signature to Bob. 4. Bob sends the coin ID and signature to the bank. 5. The bank confirms the transaction and credits Bob's account 1 cent.

Problems and issues with this scheme:

1. Alice's identity is hidden from Bob, and vice versa.
2. The bank has full knowledge of the transaction
3. Double spending: Alice can give the authenticated coin ID to Bob and Carol. We can handle this by having the bank keep a list of used coin IDs, and disallowing multiple withdrawals. Then we have a race between two recipients of a coin, so Bob needs to go online immediately to deposit the coin.

Proposal 2 **David Chaum's e-cash**

Withdrawal occurs via blind signatures, using RSA. A valid coin is (u, v) such that $v^3 = H(u) \bmod n$. The bank's public key is $(3, n)$. 3 is not special—it could be any other valid RSA exponent.

Withdrawal: Alice chooses u and r and sends $x = H(u)r^3$ to the bank (all arithmetic modulo n). The bank deducts 1 cent from Alice's account and sends $y = x^{1/3}$ to Alice. Alice computes $v = y/r$ and then (u, v) is a valid coin.

Deposit: Bob sends (u, v). The bank checks that $v^3 = H(u)$ and credits Bob 1 cent. (It checks a double spending list along the way.)

Since u and r are chosen uniformly at random and independently, the bank gains no information. For this scheme to work, u must be chosen at random from a large set. The hash must be secure: otherwise Alice gets signatures for 2 and 3 and then has a signature of 6 by multiplication.

Properties of this scheme:

1. The double spending list may get arbitrarily large; to prevent this the bank must cycle public keys every once in a while.
2. The bank, Alice, and Bob by themselves learn nothing.
3. Bob and the bank colluding learn nothing about Alice (only she knows r).
4. Alice and the bank together can learn Bob's identity, since Alice can tell the bank u. That is, the system is payer-anonymous but not payee-anonymous.

One trick to do with this scheme is to support multiple denominations. We can use multiple public keys, one for each denomination, or use this variant: let $(3, n)$ be the public key for 1 cent coins, $(5, n)$ be the public key for 5 cent coins, and $(7, n)$ be the public key for 10 cent coins. For 25 cent coins we would need to use $(11, n)$ not $(9, n)$ (that is, we take increasing primes, not odd numbers) since otherwise one could get a valid (u, v) that formed a 25 cent coin and cube both u and v to get a valid 1 cent coin, getting a free cent. With this proviso, the scheme is secure assuming the security of the underlying e-cash protocol, due to a result from Shamir. Roughly stated, given oracles for e_ith roots, $i = 1 \ldots n$, the most we can learn is zth roots where $z = \prod_{i \in S} e_i$, where S is an arbitrary multiset.

At a higher level, how do multiple denominations affect security? In terms of anonymity, e-cash systems only give users safety in numbers. More denominations reduce anonymity. In particular, very large denominations are problematic, since few users use them at a time.

As to payer versus payee anonymity, it turns out that one can build a system that is both payer and payee anonymous. Consider the following two systems.

PROPOSAL 3 **Payer-anonymous system** *Let Bob choose an arbitrary u and give it to Alice, who blinds it with an r, gives it to the bank to sign, and gives the unblinded signature back to Bob.*

This system is payer-anonymous but not payee anonymous.

PROPOSAL 4 **Payee-anonymous system.** *Let Bob choose u and r, give the blinded u to Alice, who obtains the signature and returns it to Bob, who unblinds the signature.*

This system is payee-anonymous (Alice never learns r) but not payer-anonymous (since Bob can tell the bank ur^3, and the bank can watch and see who sends it for a signature).

These can be combined by having each party produce a blinding factor.

PROPOSAL 5 **Totally anonymous system** *Let Bob choose u and r, give ur^3 to Alice, who chooses s and sends ur^3s^3 to the bank. Then Alice receives the signature, divides by s, and gives the result to Bob, who divides by r.*

To be secure, timing has to be handled carefully. In Chaum's original system, Alice withdrew money before the transaction and Bob deposited it immediately upon the transaction close. In this system Alice withdraws in realtime and Bob deposits late. (Here, if

Bob and the bank collude, timing issues might compromise Alice's identity. To be secure a real implementation would have to address this, perhaps with a random delay factor.)

At the time that Chaum's company Digicash implemented this system, it was payer-anonymous only. However, one could circumvent this by building a *moneychanger*, a device that has an account with the bank and accepts a coin in exchange for a new coin of the same denomination. A vendor could then accept coins without having an account with the bank, and use the moneychanger to convert the coins into new coins to be spent elsewhere.