Digital Cash
Jayadev Misra
08/24/2012

1 Introduction

We propose a scheme for management of digital cash that mimics the current physical management. In particular, any one can verify the authenticity of a digital bill, no one can manufacture or double-spend a bill, a transaction between two parties does not reveal the identities of the parties to others, and the bills are maintained by a distributed set of (trusted) servers that may belong to many financial institutions, much like the banks of today. Additionally, failure of any one client or server does not affect the rest of the system.

2 Implementation

2.1 Logical Implementation

A *scrip* is the electronic counterpart of a physical bill. It is stored at a unique address (given by its URL) and its contents specify the amount of the bill. The authenticity and uniqueness of a scrip are guaranteed by its contents. The contents of scrip at address c is a signed document of the form: "This is a \$5 bill stored at c"; the document is signed by a trusted authority (such as the treasury secretary; the public key required for decryption of the signed document is publicly available). Its authenticity is guaranteed by the signature. Uniqueness of the document is guaranteed by the reference to the scrip address within the document. It cannot be duplicated and stored at another address.

A scrip is publicly readable, though it is password-protected in the following sense. The password is known only to the *owner* of the scrip who owns the corresponding amount. There is just one method available to an owner: change the password. The owner may transfer the money to another party by giving the latter the password. (The new owner is expected to change the password promptly.)

A scrip is thus *permanent* —it continues to be stored at the same address— and *immutable* —its contents never change. Later, we relax both assumptions, but the changes are transparent to an owner.

2.2 Physical Implementation

The total value of all scrips is the amount of electronic money supply. We expect the number of scrips to be in hundreds of trillions since even the smallest denomination has to be stored as a scrip. Unlike the limited number of denominations used for physical bills, the Fed may issue scrips in a large number of denominations, possibly a billion dollar scrip, or in uneven amounts.

We expect that the scrips will be distributed over many servers each of which protects the password and ensures that each scrip is permanent and immutable. We call such servers *banks*, though any trusted institution can own such servers.

There are four different kinds of players in electronic banking: (1) *clients* (individuals or organizations) each of whom owns a number of scrips and may execute electronic transactions to transfer or receive scrips, (2) *banks* that store and guard the scrips and enforce password-protection, (3) *Fed* that issues new scrips, and (4) a *scrip name server* (SNS) that directs a client request for a scrip to the appropriate bank. The clients are unaware of the distribution of scrips over banks.

We assume that a client runs an app to manage his scrips. The app stores a file keeping the address, amount and password for each scrip. A transaction may require the app to assemble a number of scrips for the proper amount. Conversely, the app may also receive a number of scrips, authenticate their contents, change their passwords and add them to the client scrip file. As a background task, an app may negotiate with other apps, say at a bank, to change a set of scrips into a scrip denoting a larger amount, or conversely. An owner can write an electronic check as he would a physical check today.

A scrip may migrate from one bank to another, which is tracked by the SNS. One bank sends a scrip to another and informs SNS, the recipient bank receives the scrip and informs SNS. All the completed migrations are reflected in the mapping update. At some point both banks have copies of the scrip, but only one receives the client requests because SNS directs all requests to the bank that appears in its mapping. After SNS switches the mapping of the scrip, all requests are directed to the new bank and none to the old bank. SNS informs the old bank to destroy the scrip after the mapping; it is non-time-critical.

2.3 Points of Vulnerability

We do not address side channel attacks here. So, all the attacks orginate online and are directed towards the nodes (computers) or communication links. There are 4 possible points of vulnerability corresponding to the four kinds of players. we examine each separately.

2.3.1 Vulnerability at a client

A client may lose all his money if his scrip file is hacked. This is no different from the vulnerability in electronic banking today. Most importantly, no one else would lose any money, nor is the money supply compromised in any way.

Some of the traditional safeguards are applicable here. The stored passwords should be encrypted and only encrypted passwords are transmitted. We recommend that when an owner changes password of a scrip, the new password be chosen by the machine rather than the client, to ensure that different scrips belonging to a client have different passwords.

2.3.2 Vulnerability at a bank

A bank may lose all the money belonging to its customers if its servers are hacked. Again, the situation is no different from what it is today. And, as is the case for a client, failure is contained within the bank and does not spread over the system.

The bank may adopt a scheme, similar to the one suggested for the SNS, below, in which the bank's machine that interacts with the users can not modify the contents of scrips (though it has to modify their passwords) by write-locking the storage device where scrips are stored.

2.3.3 Vulnerability at Fed

The Fed adds to the money supply by creating new scrips whose owner is the Fed itself; we call this a *print run*. Later, it may transfer the scrips to the banks, exactly as for a physical print run. Unlike a client or a bank, if the private key used for creating scrips is compromised, the entire electronic money system becomes compromised. The hacker can then print money. This is so critical that we suggest: *each print run uses a new private key and that key is destroyed following the print run*[1].

An effect of this policy is that scrips in circulation may have been signed using different private keys. So, each scrip must store, in plaintext, an identification of the print run (perhaps the date of the run) so that the correct public key can be applied for its decryption. All the public keys corresponding to the scrips in circulation have to be available along with the identification of the print run. This is not burdensome; assuming a print-run every quarter, only 400 such entries need be stored over a century.

Another policy is to use a single public key at any time by retiring old scrips. This can be done as follows. A print run includes creation of not just new scrips but also scrips to replace old scrips. The Fed sends the replacement scrips to each bank over a period of time. On completion of this transfer, the Fed makes an announcement that a new public key is operational. The bank is expected to (1) use old scrips until it receives the announcement, (2) new scrips after the announcement, and (3) destroy all old scrips after the announcement. Observe that the banks and clients may receive the announcements at different times. A client may request to read a scrip before it receives the announcement and then find that the scrip is signed with a new key, and conversely. To avoid such problems, both the old and new public keys of the Fed should be available for a duration following the announcement that allows every bank to update the old scrips by the replacement scrips.

The two protocols may be combined so that a small number of public keys may be in circulation at any time.

[1] I owe this lovely insight to Arthur Peters and David Kitchin

2.3.4 Vulnerability at SNS

SNS provides a mapping between an address and a bank. If this mapping is altered by a hacker, the client requests may be directed to a phony bank. The following protocol makes this impossible. SNS includes two (logical) machines, one that is online and receives requests for name service, and another that is offline and performs backup tasks. The online machine is connected to a storage device (disk) from which it can only read the mapping information; the disk is manually write-locked. Therefore, it is impossible to manipulate the mapping.

The offline machine is used to create changes in the mapping, which is necessary to allow migration of scrips between banks. The backup disk can not be affected by any online attack. At some point, the backup disk is write-locked manually and switched with the online disk.

2.4 Extension

An owner can prove that he has the money without actually transferring it. This is implemented by extending a scrip's functionality. There is a read-password associated with each scrip that allows a client to read the document, but does not allow him to change any (read or write) password. A read password is use-once. The system generates a new read-password after the old one is used. The owner of the scrip can read the read-password and convey it to another client to prove his assets.

Different kinds of currencies, or items for barter, may be available; the exchange rate is either set or calculated instantaneously.