ELECTRONIC WALLET

by

S. Even, O. Goldreich and Y. Yacobi

Technical Report #276

June 1983
ABSTRACT

The electronic wallet looks like a small pocket calculator with a plug which enables electronic connection with another wallet, either directly or through telephone line. It stores an unforgeable amount of money, and enables unforgeable transactions with other wallets. It is safer than cash money since only the legitimate owner who knows the password can operate it.

It can replace credit-cards, checks and traveler's checks and saves the paper work involved in their use. In most transactions there is no need for the intervention of the bank at all.

In remote transactions it provides reliable identification.
We describe an electronic wallet assuming that our device is mechanically constructed in a way that ensures that any attempt to open it will destroy the secret information stored in it. Under this assumption we could do well with conventional cryptosystems. However, since in real life nobody can 100% guarantee this assumption, we take an extra precaution and implement our system using public key cryptosystem. As a result, even if one succeeds to penetrate a wallet he cannot endanger the security of other wallet.

1. SPECIFICATION

In the description of a basic transaction there is no difference between a bank and a customer. We require that a basic transaction have the following properties:

(a) A transaction may take place only if both parties agree on it,
(b) The balance should be unforgeable,
(c) The sum of balances before and after the transaction should be the same,
(d) The paid party should have a proof of the legitimacy of the transaction,
(e) The paying party should have an unforgeable receipt,
(f) A transaction may take place only if the paying party’s balance is greater than or equal to the payment.

2. ASSUMPTIONS

(2.1) Any mechanical attempt to penetrate the wallet will destroy the information stored in it.

(2.2) There is a secure PKCS. The PKCS of user i is a pair of operators \((D_i, E_i)\) s.t. \(D_i\) is secret and \(E_i\) is publically known,
but doesn't reveal $D_i$, and $D_iE_i \neq E_iD_i = 1$, the identity operator. In order to operate $E_i$ one must use the public key $e_i$, and in order to operate $D_i$ one must use the secret key $d_i$. RSA seems to currently be sufficient. For more details consult [RSA].

(2.3) Even the owner of a wallet doesn't know the secret private key stored in it.

### 3. The Data Stored in the Wallet

(3.1) $j$'s current balance, $B_j$

(3.2) $e_x$ and $d_j$, where $e_x$ is the public key of the bank and $d_j$ is $j$'s secret key.

(3.3) $D_x (e_j)$, $j$'s public key certified by the bank.

(3.4) $j$'s password,

(3.5) A list of cancelled wallets,

(3.6) Audit trail.

### 4. Denotations

Let $t$ represent the real time. 30 bits are sufficient to represent real time with resolution of seconds over a period of 30 years; $v$ represents the value of the current transaction; $(DATA)\rightarrow$ denotes the transmission of DATA to the current counterpart (the arrow may point to the left). We assume that the error message $(err.\ mess.)\rightarrow$ always implies halting; $n$ denotes random noise.

### 5. The Transaction Protocol

Suppose $i$ agrees to pay the amount of $v$ dollars to $j$. $i$ and $j$ first identify themselves to their corresponding wallets using their passwords. Then they plug their wallets back to back (or via
telephone line, i types PAY($v_i$) and j types GET($v_j$). If they are honest then $v_i = v_j$ (even). The protocol proceeds as follows:

\[ \text{if } v_i \neq v_j \text{ (err.mess.) -->} \]

If $v_i$ not equal $v_j$ (err.mess.) -->

\[ \text{if } v_i \neq v_j \text{ then } \]

\[ \text{<----- (err. mess.)} \]

\[ \text{part-b: protection against forged wallets} \]

\[ \{ \text{the process continues iff } v_i = v_j (=v) \} \]

\[ \text{If } v < B_i \text{ then} \]

\[ B_i := B_i - v_i; \]

\[ (i,D_i(v,t,n)) --> \]

else (err. message) -->

\[ \{ \text{Let } C \text{ denote the received signed triple and } (v',t',n') = E_i(C) \} \]

\[ \text{If } (v',t',n') = (v,t,n) \text{ then} \]

\[ \text{<----- (j,D_j(-v,t,n))}; \]

\[ B_j = B_j + v; \]

\[ \text{else (err. message);} \]

6. ANALYSIS

Clause (a) of the specifications hold since both parties first verify that they typed the same payment, $v$. Also, we assume that in case they both type GET($v$) the process stops.

One cannot increase his balance without the corresponding decrease of his counterpart's balance, by the definition of the protocol. This takes care of (c). It also takes care of (b) as long as one is not up to cheating himself. (d) is true since the paid party has \((i,D_i,(v,t,n)), \)
and (e) is true since the paying party has \((j,D_j(v,t,n))\). (f) holds
since a transaction takes place only if \(v < B_i\). Therefore, the
protocol meets the specifications. Also \((i,D_i(v,t,n))\) and \((j,D_j(v,t,n))\)
are proofs of identification of the corresponding parties. The
description of the password check mechanism is omitted.

REMARKS:

(0) We could do well with conventional cryptosystems. We use public
key cryptosystems as an extra safeguard so that in case (2.1) is
violated only a restricted damage may occur, i.e. the violator
may duplicate his wallet only.

(1) Even \(i\) doesn't know \(D_i\) therefore he cannot forge a similar wallet
which doesn't perform \(B_i = B_i - v\) in the course of his payment
transaction.

(2) One cannot pay the same money simultaneously to two wallets using
some illegal extra hardware due to the first step of the protocol,
which contains unpredictable noise. (This is a probabilistic
safeguard. We could make it deterministic by adding to the signed
triples a fourth component: the payee's id.)

(3) The real time \(t\) serves to inhibit cheating using replays.

(4) The fact that \(i\) knows \(D_x(e_i)\) (and vice versa) doesn't risk \(j\),
since \(i\) cannot forge \(D_j(v,t,n)\).

7. NON BASIC BANK-WALLET PROTOCOLS

The following protocols between the bank and the wallet may also
be needed: (each protocol starts with a mutual identification of the
bank and the wallet).
(1): Transfer of audit trail from wallet to bank: The wallet transfers its record of past transactions to the bank and erases its record.

(2) Update list of cancelled wallets: The certification of a key will include a version number. A key with a wrong version number is automatically illegal. In addition there is a list of wallets with correct version number which are declared invalid, i.e. they are reported to be stolen, or known by past audit trail to be forged, or ill-used. The bank transfers the updated list of cancelled wallets to the wallet.

(3) Update bank public key and the bank certification to the wallet's public key.

Remark: Note that the purpose of checking the audit trails is to detect forged wallets.

REFERENCE