



# **A new electronic check system with reusable refunds**

S. Kim and H. Oh



# Agenda

---

- ▶ Clarification of the subject
- ▶ Desirable properties of electronic money
- ▶ Partially blind signature
- ▶ The protocol
  - ▷ Account setting
  - ▷ Withdraw
  - ▷ Payment
  - ▷ Refund
- ▶ The resolve protocol
- ▶ What to pursue further

# Clarification of the subject

<b>Real-life check</b>	<b>E-check in this paper</b>
Not anonymous	Anonymous
Transactions are linkable	Transactions are unlinkable
Usually paid in the exact price of goods	Paid in a (pre-decided) check value and receive a refund check as change.
Post-paid	Pre-paid

# Clarification of the subject

<b>Real-life cash</b>	<b>E-check in this paper</b>
Fixed face values	Users decide the check values
Lifetime extends beyond a single transaction	Just for one transaction.
Can be used by any person, not resistant to theft.	Can only be used by the owner, resistant to theft.

# Desirable properties

---

An ideal payment system is a divisible-cash system with the following properties:

- ▶ One coin is used for each payment.
- ▶ Subcoins are unlinkable.
- ▶ Protocols are computationally efficient.

Not feasible with current technology . . .

# Desirable properties

---

This paper describes an alternative solution:  
A check system with an efficient reusable refund mechanism.

- ▶ A refund is unlinkable to the paid check.
- ▶ A refund check is reusable as payment
- ▶ Receiving a refund is efficient.

# Partially-blinded signature

- ▶ The receiver receives a signature  $Sig(m, c)$  on  $m$  (clear part) and  $c$  (blinded part).
- ▶ The signer does not get any information about  $m$  or the resulting signature  $Sig(m, c)$ .
- ▶ Allows a signer and a receiver to agree upon the clear part in the blinded signature.
- ▶ In this paper
  - ▷ Signer is the bank; receiver is the client.
  - ▷  $c$  is the face value and the valid period;  $m$  is the serial number of a check.

# Partially-blinded signature

An RSA-based partially-blinded signature:

- ▶  $n = p \cdot q$ , ( $p$  and  $q$  are large primes)
- ▶ A public exponent generating function  $f$  satisfying:
  - ▷  $f(c)$  must be different for different values of  $c$ .
  - ▷  $f(c)$  must be relatively prime to  $\phi(n)$ .
  - ▷ Each output has at least one unique prime factor.

Abe and Camenisch suggested:  $f(c) = 2 \cdot h(c \parallel \delta) + 1$   
in which  $\delta$  is a random value and  $h : \{0, 1\}^* \rightarrow \{0, 1\}^l$   
is a collision free hash function.

- ▶  $(n, f)$  is the public key, and the factorization of  $n$  is the private key.

# Partially-blinded signature

The signature protocol goes like this (Alice as receiver, Bob as signer):

- ▶ Alice chooses  $r \in_R \mathcal{Z}_n^*$  and the desired  $c$ , then computes  $e_c = f(c)$  and  $m' = m \cdot r^{e_c} \pmod n$
- ▶ Alice sends to Bob:  $m', c$ .
- ▶ Bob computes:

$$e_c = f(c)$$

$$d_c = e_c^{-1} \pmod{\phi(n)}$$

$$s' = m'^{d_c} = (m \cdot r^{e_c})^{d_c} = m^{d_c} \cdot r \pmod n$$

# Partially-blinded signature

- ▶ Bob sends to Alice:  $s'$
- ▶ Alice computes the signature

$$s = s'/r = (m^{d_c} \cdot r)/r = m^{d_c} \pmod n$$

and verifies

$$m \stackrel{?}{=} s^{e_c} \pmod n$$

# System setup

---



- ▶ Three players: client, shop, and bank
- ▶ The bank maintains the accounts for the client and the shop.
- ▶ The bank publishes:
  - ▷  $(n, f)$ , in which  $n$  is an RSA modulus and  $f$  is an appropriate public exponent generating function.
  - ▷  $(n_b, e_b)$  which is a standard RSA public key.



# Account settings

Shop account setting:

- ▶ A shop  $S$  chooses an RSA modulus  $n_s = p_s \cdot q_s$  and generates RSA key pair  $(e_s, d_s)$ .
- ▶ The shop sends  $(e_s, n_s)$  to the bank to establish an account  $acct_s$ .

An client  $U$  performs the same steps:

- ▶ Chooses  $n_u$  and generates  $(e_u, d_u)$ .
- ▶ Sends  $(e_u, n_u)$  to the bank to establish account  $acct_u$

# The withdraw protocol

A client  $U$  requests the bank to perform an RSA-based partially-blinded signature to withdraw a check:

- ▶  $U$  generates a blind factor  $r$  and two secret keys  $K_b$  and  $K_s$ :
  - ▷  $K_b$ : used as the serial number of a check.
  - ▷  $K_s$ : used to exchange data with the shop (later).
- ▶ The clear part of the signature  $c = V \parallel T$  encodes the face value ( $c.V$ ) and the expiration date ( $c.T$ ) of the check.
- ▶ The client obtains a check of the form:

$$s = (h(K_b) \parallel h(K_s))^{d_c}$$

# The withdraw protocol

Client  $\mathcal{U}$

$$r \in_R \mathbb{Z}_n^*$$

$$K_b, K_s \in_R \mathbf{K}$$

$$m = \{h(K_b) \parallel h(K_s)\}$$

$$c = \{V \parallel T\},$$

$V$  : check value,

$T$  : expiration date

$$e_c = f(c)$$

$$\tilde{m} = mr^{e_c} \pmod n$$

$$s = \tilde{s}/r \pmod n$$

$$m \stackrel{?}{=} s^{e_c} \pmod n$$

Bank  $\mathcal{B}$

$$\xrightarrow{\text{acct}_u, \tilde{m}, c, \text{Sig}_{d_u}(\tilde{m} \parallel c \parallel T_u)}$$

verify  $\text{Sig}_{d_u}(\tilde{m} \parallel c \parallel T_u)$

$$e_c = f(c)$$

$$e_c d_c \equiv 1 \pmod{\lambda_n}$$

$$\tilde{s} = \tilde{m}^{d_c} \pmod n$$

$$\xleftarrow{\tilde{s}}$$

# The payment protocol

The payment protocol is essentially a variation of the withdraw protocol with the shop playing an intermediary role. This is an outline:

- ▶  $U$  sends the shop  $S$  the identifier  $I$  of the desired item, a check, and some authentication information.
- ▶  $S$  verifies that the check is authentic and its face value is enough for the purchase.  $S$  then sends the purchased item encrypted by a random key  $K'$  (we're only talking about electronic goods) and a receipt to  $U$ .

# The payment protocol

- ▶  $U$  verifies the receipt and sends  $S$  a change request order  $O_u$  which says “I purchase  $I$  with  $\$price$  from shop  $S$ ”.  $O_u$  also includes the information necessary to construct the new refund check (same as in the withdraw protocol).
- ▶  $S$  generates a deposit order  $O_s$  which says “I sold something priced  $\$price$  which was paid for by check  $s$ ”.
- ▶  $S$  sends  $s$ ,  $O_s$  and  $O_u$  to the bank  $B$ .

# The payment protocol

- ▶  $B$  verifies that the check  $s$  is valid, then constructs and sends  $S$  a payment certificate  $pay - cert$  and a new check  $\tilde{s}$  if the value of  $s$  is greater than the price of  $I$ .  $B$  also updates  $S$ 's account.
- ▶  $S$  verifies  $pay - cert$ , and sends the refund  $\tilde{s}$  and the key  $K'$  to the client  $U$ .
- ▶  $U$  verifies that  $\tilde{s}$  is valid, then decrypts and verifies the purchased item.  $\tilde{s}$  can be used for purchasing as a new check.

# The payment protocol

Client  $\mathcal{U}$

$$r' \in_R \mathbb{Z}_n^*$$

$$K'_b, K'_s \in_R \mathbf{K}$$

$$m' = \{h(K'_b) \parallel h(K'_s)\}$$

$$e_{c'} = f(c')$$

$$\tilde{m} = m' r'^{e_{c'}} \pmod n$$

$$k_b = K_b^{e_b} \pmod{n_b}$$

$$k_s = K_s^{e_s} \pmod{n_s}$$

Shop  $\mathcal{S}$

$$k_b, h(K_b), k_s, h(K_s), \\ s, c, I$$

$$K_s = k_s^{d_s} \pmod{n_s}$$

$$e_c = f(c)$$

$$m = \{h(K_b) \parallel h(K_s)\}$$

$$m \stackrel{?}{=} s^{e_c} \pmod n$$

$$c.V \stackrel{?}{\geq} \text{price}_s$$

$$K' \in_R \mathbf{K}$$

$$K = K' \oplus K_s$$

$$\text{receipt} = \text{Sig}_{d_s}(\{\text{item}\}_K \parallel I \parallel \text{price}_s \parallel s)$$

verify receipt

$$O_u = \{\text{price}_u \parallel \text{acct}_s \parallel \tilde{m}\}_{K_b}$$

$$\xleftarrow{\{\text{item}\}_K, \text{receipt}}$$

$$\xrightarrow{O_u}$$

# The payment protocol

Shop  $\mathcal{S}$

$$O_s = \text{Sig}_{d_s}(\text{price}_s \parallel s)$$

verify pay-cert

Client  $\mathcal{U}$

$$s' = \tilde{s}/r' \pmod n$$

$$m' \stackrel{?}{=} s'^{e_{c'}} \pmod n$$

decrypt and verify the item

$$k_b, h(K_s), s, c, O_u, O_s \\ \text{price}_s, \text{acct}_s$$

Bank  $\mathcal{B}$

$$K_b = k_b^{d_b} \pmod{n_b}$$

$$e_c = f(c)$$

$$m = \{h(K_b) \parallel h(K_s)\}$$

$$m \stackrel{?}{=} s^{e_c} \pmod n$$

verify  $O_s, O_u$

$$c'.V = c.V - \text{price}_s$$

$$e_{c'} = f(c')$$

$$e_{c'} d_{c'} \equiv 1 \pmod{\lambda_n}$$

$$\tilde{s} = \tilde{m}^{d_{c'}} \pmod n$$

$$\text{pay-cert} = \text{Sig}_{d_b}(\text{price}_s \parallel \text{acct}_s \parallel s)$$

Shop  $\mathcal{S}$

$$\tilde{s}, K'$$

# The refund protocol

A client can use the following protocol to deposit unused checks:

Client  $\mathcal{U}$

$$R_u = \{\text{acct}_u \parallel c \parallel K_s\}_{K_b}$$

$$k_b = K_b^{e_b} \pmod{n_b}$$

$k_b, R_u, s$

Bank  $\mathcal{B}$

$$K_b = k_b^{d_b} \pmod{n_b}$$

decrypt  $R_u$

$$e_c = f(c)$$

$$m = \{h(K_b) \parallel h(K_s)\}$$

$$m \stackrel{?}{=} s^{e_c} \pmod{n}$$

# The recovery protocol

- ▶ *Withdraw atomicity*
  - ▷ May be violated if the client falsely claims not receiving  $\tilde{s}$  before the state information  $(K_b, K_s, r, c)$  has been lost.
  - ▷ Solution: force the clients to use stable storage. (*Is this a solution at all?*)
- ▶ *Payment-delivery atomicity*
  - ▷ The payment protocol may be broken due to malfunction.
  - ▷ The client re-initiates the protocol. The shop resumes the protocol unless a dispute occurs.

# The recovery protocol

## ► *Payment-delivery atomicity (cont.)*

- ▷ In case of a dispute, the following resolve protocol is performed:
  - ★ The client presents the check to the TTP and proves the ownership of the check.
  - ★ The TTP asks the bank to examine whether the shop has been credited.
  - ★ If yes, the TTP asks the shop to send the key  $K'$  to the client. Otherwise, the TTP notifies the bank not to accept the check and contacts the shop to determine the cause of the dispute.
  - ★ The shop complies to the TTP's decision or faces a legal action.

# The recovery protocol

---



- ▶ *Refund atomicity:*

The client re-initiates the refund protocol. The bank credits the client's account if this check has not been refunded before.



# Conclusions

---

This paper described an online check system with the following features:

- ▶ flexible check value
- ▶ user anonymity
- ▶ reusable refund
- ▶ unlinkability between the initial checks and the refunds
- ▶ efficiency (?)

# What to pursue further

These are problems I think worth pondering (on e-commerce in general):

- ▶ We need a trust model that closely reflects the trust in the real world.

## Known Theoretical model

Players don't trust each other.

Anonymity: *All* information about the users' transactions should be kept secret from other parties if possible.

## Real world

People *do* have trust on each other.(credit cards, checks, online shopping).

Most people are not concerned that their credit card companies have their purchase history.(Why? And exactly what information should be kept secret from whom?)

# What to pursue further

---

- ▶ Anonymity and accountability
  - ▷ Problem with perfect anonymity: perfect crimes.
  - ▷ Anonymity should be revocable.
- ▶ Anonymous post-paid service (Consider pre-paid phone cards vs. your cellphone plan).
  - ▷ Clients want to keep their activities private.
  - ▷ Service providers need to keep accounting information (to calculate how much the clients should be charged).
  - ▷ Is anonymous post-paid service possible in spite of the conflicting goals?