Versatile Anonymous, Fair and Off-line Hybrid Payment System

Arrianto Mukti Wibowo[1], *Kwok Yan Lam*[2], *Gary S. H. Tan*[2]

[1]arrianto@comp.nus.edu.sg, amwibowo@yahoo.com
[2]{lamky, gtan}@comp.nus.edu.sg

Contact author:
Mr. Arrianto Mukti Wibowo
telephone: +65 874 2735

Mailing address:
c/o Graduate Office, S15-5
Computer Science Department, School of Computing
National University of Singapore
10 Kent Ridge Crescent
Singapore 119260

Abstract. There has been a lot of off-line electronic cash systems proposed so far, which are based on the 'digital coins' paradigm. Digital coins are anonymous, but not without drawbacks. Among the drawbacks are: large database requirement to record spent coins, customers must 'carry' the coins prior payments, time consuming client side processing at payment (especially with multiple coins), and coin divisibility problems. Other proposals suggested account-based payment systems that are based on wire fund transfer, therefore the customers do not need to carry the coins nor divide coins. However, they only provide limited (or easily compromised) anonymity. In this paper we propose a novel solution, which we baptize '*hybrid payment system*' (*HiPS*). As its name implies, it is a hybrid of digital coin system and account-based payment system, and has the best features of both systems. Moreover, a judge can revoke complete anonymity in HiPS whenever required. (version: 9 June 2000)

Keywords: financial cryptography, electronic payment system, electronic cash, account-based payment system.

1 Introduction

1.1 Background & Problems

Since the day Chaum, Fiat and Naor invented the off-line electronic cash [10], nearly all of the off-line electronic cash systems are based on the 'digital coin' paradigm (e.g., [2, 12, 13, 20, 21] and their derivatives). The basic form of an off-line electronic cash system consists of three major parties, which is a bank, its customers, and its merchants. Generally, there are four important phases in electronic cash systems. First, a customer must *setup an account* to be used at the 'minting' process. Then, the customer can *withdraw or* '*mint*' *digital coins* from the bank, in which bank will deduct his account equal the value of the coins he withdraws. After possessing the coins, the customer can *use the coins to pay* to the merchant. Finally, the merchant *deposits the coins* to the bank, which then the bank credits merchant's account. Electronic cash systems use a certain blind signature protocol to achieve customer anonymity.

Despite their anonymity features, they have some major drawbacks. Generally, they have large database requirement to record coins which have been spent; expensive client side computation at payment phase due to multiple coin processing; storage requirement problem (because client must carry the coins); and (coin) divisibility problem. Of course, several papers [8, 12, 20, 21, 25] have suggested solutions to the divisibility problem, but not without increasing the complexity of the protocol at payment phase.

Other proposed payment systems such in [3, 17, 18, 19, 23] use a bank account paradigm, which also tried to achieve some limited anonymity level. Account-based payment systems do not have the problems of digital coin systems just mentioned above, because they are based on ordinary wire fund transfer.

Proposed anonymous bank account scheme from Camenish, Pivetau and Stadler [3] is an on-line debit payment scheme. It has several drawbacks. First, since in each payment transaction the customer discloses his anonymous account number, therefore the payments are linkable. Second, because the scheme requires the customer to transfer some funds from his conventional (identifiable) account to his anonymous account which are linked with a common 'synchronous counter', the bank can conduct a traffic analysis and still have a chance to statistically link the inter-account fund transfer.

Other proposals of anonymous account based payment system, which is based on the credit card transaction scheme, was presented by Low, Maxemchuck and Paul in [17, 18]. Their proposed schemes suffer from possible collusion of principal parties against the customer that may reveal the true identity of the customer.

In the Visa/MasterCard SET protocol [23], the merchants are also not allowed to know the credit card number of the cardholder (customer). Cardholder's certificate does not bear the credit card number, but rather a commitment to the credit card number (Personal Account Number). SET uses HMAC SHA1 to 'blind' the credit card number in cardholders certificate. SET also suffers from the fact that payment transactions made by the same cardholder are linkable. Krawczyk analyzed the blinding schemes of SET in [16].

Mu and Varadharajan also proposed another anonymous credit card [19]. Although the payments are anonymous, the payments are still linkable. No attempt was made to make the whole system truly anonymous, since at the end, the bank will have to send the bill to the customer's conventional bank account.

No anonymity revocation was considered in all of the proposed account-based anonymous payment system so far.

1.2 Objective

Initially, we were motivated by the need to have a smartcard based payment system with a fairly good degree of anonymity, but without the problems of digital coin systems, most notably the divisibility problem. We also assume that the customer is willing to do additional pre-payment processing to achieve payment-phase convenience. The use of smartcard also implies that at payment phase, the customer has only a limited storage.

In designing our newly proposed payment system, we extend the objectives to: minimize spent-coins database space requirement; minimize client computation time at payment phase; minimal client side storage (so the customer does not need to 'carry' coins at payment phase); and totally eliminate the divisibility problem. At the same time we also fix the problems of the account-based payment system which has been proposed so far: more protection from eavesdropper's traffic analysis; provide a certain degree of payment-unlinkability; higher degree of collusion-resistant anonymity; maintaining the off-line payment capability (albeit in a rather different meaning from digital coin systems); and also achieve a *fair* payment system (anonymity revocation capability).

1.3 Organization

In section 2, we sketch our proposed scheme and briefly illustrate how our payment system mends the problems of previous works. Next, we describe the protocol building blocks in section 3. Our payment system is then completely described in section 4. Finally, we conclude the paper in section 5 with discussions on the analysis of the scheme, the implementation issues and some open problems.

2 Overview of the HiPS

Our proposed payment system is a hybrid between bank account paradigm and digital coin paradigm. We use the digital coin protocol at the initial phase our payment protocol and the anonymous bank account is used at later phases, notably for the actual payment phase. Therefore, for our newly proposed ***Hybrid Payment System***, we invent the term **HiPS**[1].

HiPS consists of three *major* (*principal*) *parties*, which are the bank *B*, the customer (or the client) *C*, and the merchant *M*. Other non-principal parties include a trusted mix-resender *T* (much like the mix-remailer for untraceable mail in [9]) and an escrow agent *E* (e.g., a judge) to revoke the anonymity whenever required. We shall now briefly sketch our proposed scheme.

Initially, a customer with a conventional bank account, anonymously creates an anonymous bank account (step 1). The customer then transfers his money from his conventional account to the anonymous account, such that, even if all major parties collude together against the customer, they still cannot identify which customer did a particular inter-account fund transfer (step 2, 3, 4). At the payment phase, the customer uses his newly created anonymous account to pay the merchant, in such way that the merchants will know that the payment is valid, but without the customer surrendering his anonymous account number to the merchant (step 5, 6). Because the payment phase uses a bank account paradigm, the customer does not need to 'carry coins'. Refer to figure 1 for the scheme diagram.

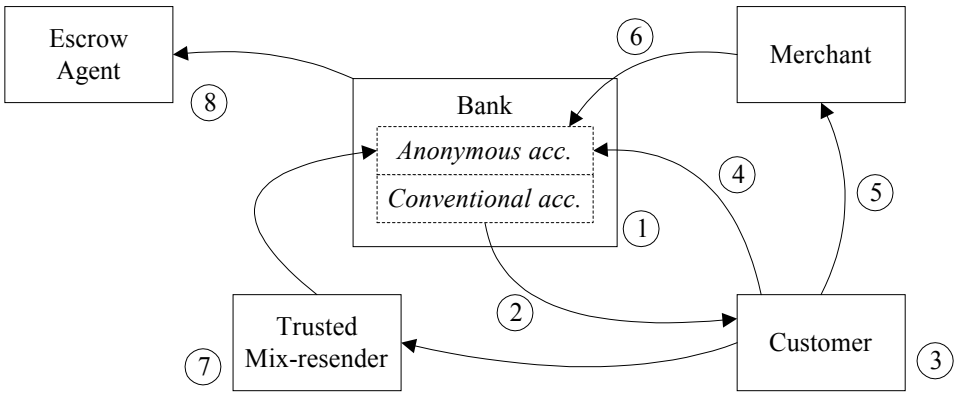

Figure 1. *Scheme diagram of Hybrid Payment System (HiPS)*

HiPS solves the weaknesses of previous works on coin-based and account-based payment system. We use a digital coin protocol to completely obfuscate the fund transfer from

[1] Actually [3] may also be categorized as a hybrid payment system, since the fund transfer from the conventional account to the anonymous account uses a blind signature protocol.

customer's conventional bank account to his anonymous account. We remove the synchronous counter of [3] on both accounts. We also divide the fund transfer into several fixed 'coin'[2] denominations to disallow statistical correlation using traffic analysis of the inter-account fund transfer. To further obfuscate the fund transfer from traffic analysis, those coins should be transferred into the anonymous account in such unpredictable time. Optionally, the user can trust a mix-resender to automatically do the fund transfer process (step 7).

Off-line payment capability with pseudo payment-unlinkability is achieved by using a *group signature* scheme [6]. Unlike [17, 18], HiPS also protects the true identity of the customer, even if all major parties collude together. Anonymity revocation (step 8) can be achieved in several ways, but in this paper we choose *fair off-line e-cash* (FOLC) in [14] as the foundation for the inter-account fund-transfer anonymity revocation, and *fair blind signature* protocol derived from FOLC to generate the required blinded identity.

3 Basic Protocol Building Blocks

Here we describe several basic protocol building blocks for HiPS. We opt not to rewrite the FOLC protocol and the group signature protocol in detail due to space limitation. However, to give a clear picture, we elaborate the fair blind signature protocol, which is actually Brands's restrictive blind signature scheme in [2] with anonymity control of FOLC.

To let our paper be uncomplicated and clear, we encapsulate all possible complex protocols into functions with few parameters as possible. For example, from the function parameters we shall omit: prime modulo; random numbers generated at run-time; generators from most discrete log based setup protocols; etc. Even on the later part of the paper such in section 4, for simplicity, we will further omit obvious parameters such as the appropriate public key from the function.

3.1 Interactive Proof of A Discrete Logarithm Representation Knowledge

In this protocol, the prover P convinces verifier V that P knows how to represent $I = g^u$ with respect to g. This could be done by using Schnorr-type identification scheme [22].

1. P : $x' \in_R \mathbf{Z}_q$, $y' = g^{x'}$
   $P \rightarrow V$: I, y'
2. V : $c \in_R \mathbf{Z}_q$
   $V \rightarrow P$: c
3. P : $r_c = uc + x'$

[2] Later in section 3.3.3 we shall change the term '*coin*' into '*anonymous cashier's check*' because the term 'coin' does not fit nicely with large amount fund transfer. Nevertheless, both terms refer to the same thing.

$P \rightarrow V$: r_c

V : $g^{r_c} \stackrel{?}{=} I^c y'$

We encapsulate this process into:

$P \leftrightarrow V$: ***ProofLogarithmRepresentationKnowledge***(I, g)

This protocol is used in the fair blind signature protocol. Although not described in detail in this paper, this protocol is also used in the FOLC protocol and the group signature protocol used in HiPS.

3.2 Equality of Logarithm

To provide anonymity revocation (fairness) for owner tracing, blind identity tracing, and coin tracing, we require an efficient *blind proof* of equality of logarithms. This important building block is used in the fair blind signature protocol, FOLC and even in the group signature scheme in in HiPS.

In the equality of logarithms protocol, the prover P basically proves to verifier V the equality of two logarithm, which is $\log_a A = \log_b B$. We define a, b, G_1, G_2 as generators of G_q, where G_q is a subgroup of prime order q of the multiplicative group $\boldsymbol{Z}_p^*$ for some large prime p. The prover P is assumed not to know the relative discrete logarithms of a, b, G_1, G_2. Secret input to P is x, v, $w \in G_q$, such that $A = a^x G_1^v, B = b^x G_2^w$. All of the calculations here are done in modulo p, and all exponential calculation are done in modulo q, unless explicitly mentioned.

The prover P now proves to V that $A = a^x G_1^v, B = b^x G_2^w$ (i.e. $\log_a A = \log_b B$).

1. P : $y, s_1, s_2 \in_R \boldsymbol{Z}_q$, $A' = a^y G_1^{s_1}, B' = b^y G_2^{s_2}$

   P $\rightarrow$ V : A', B'

2. V : $c \in_R \boldsymbol{Z}_q$

   V $\rightarrow$ P : c

3. P : $r = cx + y$, $r_1 = cv + s_1$, $r_2 = cw + s_2$ (computations in mod q)

   P $\rightarrow$ V : r, r_1, r_2

4. V : $a^r G_1^{r_1} \stackrel{?}{=} A^c A'$ and $b^r G_2^{r_2} \stackrel{?}{=} B^c B'$

For simplicity, the whole process is encaplsulated into a function ***EqLog***((A, a), G_1, (B, b), G_2) in the original paper. And because the process is a two way process between two parties, the simplified notation for the protocol is:

$P \leftrightarrow V$: ***EqLog***((A, a), G_1, (B, b), G_2)

However, since the challenge c from the verifier V apparently came out to be important in HiPS, we modify the notation to include challenge c from V. Hence, to accommodate the challenge, the function can also be written as ***EqLog***((A, a), G_1, (B, b), G_2, c), where c is the challenge from V:

$P \leftrightarrow V$: ***EqLog***$((A, a), G_1, (B, b), G_2, c)$

The equality logarithm function can also be extended into proving more than two values. We can also define ***EqLog***$((A, a), G_1, (B, b), G_2, (C, c), G_3)$ to prove equality between the respective logarithms of A, B and C.

3.3 Fair Off-Line e-Cash (FOLC) Protocol

3.3.1 Brief Description

Since we achieve account anonymity by advancing the 'cash' inter-account fund transfer before the payment phase, we still need to use a certain digital coin protocol. In this paper we use Frankel, Tsiounis and Yung fair off-line e-cash (FOLC) [14] which is an extension of Brands' electronic cash [2]. The described protocol in [14] also incorporates fixes by Chan, Frankel, MacKenzie and Tsiounis [7] over Brands' original protocol.

Our reason to use FOLC is because its design supports anonymity revocation. Anonymity revocation includes owner tracing (to reveal C's true identity in suspicious coin deposits and suspicious payments) and coin tracing (to reveal the corresponding coin of a suspicious withdrawal transaction).

3.3.2 Minor Modification on FOLC's Payment Phase Protocol

We do not change any FOLC [14] building block protocols, except the payment phase protocol. Recall that in most digital coin protocols, including FOLC, at the payment phase, the recipient (or the merchant in our context) asks the customer to prove his ownership of the digital coins. The recipient does so by giving an interactive challenge c to the customer, where c is a random string. Alternatively, under the random oracle model [1], the challenge c can also be in the form of the hash of recipient's identity, description of the payment, and payment date/time. Note that if we adopt the random oracle model for c, and the customer C already know the recipient of the coin, we can make the protocol non-interactive. C can affix the destination of his coin himself without anyhelp from the recipient.

We now describe the modification in detail. From the withdrawal protocol of FOLC, C recieves the object *Coin*, which is the pair (A, B) with the bank B's signature $\sigma(A, B) = (z, a, b, r)$ on it. Readers are refered to [14] for the notations and explanation of A, B, z, a, b, r and the detailed withdrawal protocol, because we use exactly the same notations too. Readers may also note that the withdrawal protocol of FOLC is analogous to the blind certificate generation in section 3.4.4 in this paper.

At payment phase, C provides to the recipient R that he is the owner of *Coin*, and supplies sufficient information to R (which is later forwarded to the bank B) so if a coin is double-spent, the identity of C will be revealed.

1. C : $A_1 = g_1^{u_1} g_2^{s}\ [= A / g_3]$, $A_2 = f_2^{s}$
   $C \rightarrow R$: $A_1, A_2, A, B, (z, a, b, r)$
2. R : $c =$ (*RecipientIdentity*, *PaymentDescription*, *PaymentDateTime*)
3. $C \leftrightarrow R$: ***EqLog***$((A_1, g_2), g_1, (A_2, f_2),$ nil, $H(c))$
4. R : $A_1 \overset{\gamma}{\neq} 1$, $A_1 g_3 \overset{\gamma}{=} A$
   $\sigma(A, B) = (z, a, b, r)$

The transcript of the protocol is an object *SpentCoin*.

3.3.3 Naming Equivalences in HiPS

We shall see later in section 4 that the FOLC protocol is used in HiPS in the inter-account anonymous fund transfer. Therefore, we need to rename and map the FOLC protocol and object names into the corresponding HiPS protocol and object names.

Note that we replace the term 'coin' with 'anonymous cashier's check', since we assume that the customer would conduct the fund transfer in much larger amount compared to the much smaller (and much more often) payments.

FOLC Naming	**HiPS Naming Equivalence**
Coin object	*AnonymousCashiersCheck*
SpentCoin object	*PredestinedAnonymousCashiersCheck*
Bank setup protocol (performed once by B)	*B* : ***BankSetupInterAccount-AnonymousFundTransfer***
Escrow agent setup protocol (performed once by E)	*E* : ***EscrowSetupInterAccount-AnonymousFundTransfer***
Customer setup protocol (performed once for each C)	$C \leftrightarrow B$: ***CustomerSetupInterAccount-AnonymousFundTransfer***
Withdrawal protocol (over an authenticated channel between B and C)	$C \leftrightarrow B$: ***WithdrawCheck*** = *AnonymousCashiersCheck* The transcript of this protocol is specified as *CheckWithdrawalTranscript*
Payment protocol (performed between C and recipient R over an anonymous channel)	$C \leftrightarrow R$: ***AssignCheckDestination***(*AnonymousCashiersCheck*, *RecipientChallange*) = *PredestinedAnonymousCashiersCheck*
Deposit protocol (performed between recipient R and B over an authenticated channel)	$R \rightarrow B$: *PredestinedAnonymous-CashiersCheck* *B* : ***VerifyCheck*** (*Predestined-AnonymousCashiersCheck*)
Owner tracing protocol (performed between B and E over an authenticated channel)	$B \leftrightarrow E$: ***TraceOwner*** (*Predestined-AnonymousCashiersCheck*) = *Identity*
Coin tracing protocol (performed between B and E over an authenticated channel)	$B \leftrightarrow E$: ***TraceAnonymousCashiers-Check*** (*CheckWithdrawal-Transcript*) = *PredestinedAnonymousCashiersCheck*

Table 1. *FOLC naming equivalence in HiPS*

3.4 Fair Blind Signature Protocol

The fair blind signature protocol is used by bank *B* to create a fair blindly signed digital certificate (fair blind certificate, or just blind certificate for short) for customer *C*, which the escrow agent *E* takes over the anonymity revocation. There are several fair blind signature protocols, such as magic-ink signatures from Jakobsson [15] and also from Stadler, Piveteau and Camenish [24], but we opt for the FOLC [14] variant because they use the same primitives.

3.4.1 Bank Setup Protocol (performed once by B)

The bank *B* chooses primes *p* and *q* where $|p - 1| = \delta + k$ for a specified constant δ, and $p = \gamma q + 1$, for a specific integer γ. Then a unique subgroup G_q of prime order *q* of the multiplicative group $\boldsymbol{Z}_p^*$ and generators g, g_1, g_2, g_3, g_4 of G_q are defined. A secret key $X_B \in_R \boldsymbol{Z}_q$ is generated too. Hash function *H* from a family of collision intractible hash functions are also defined. *B* publishes *p, q, g, g1, g2, g3, g4, H,* and its public keys $h = g^{X_B}$, $h_1 = g_1^{X_B}$, $h_2 = g_2^{X_B}$, $h_3 = g_3^{X_B}$. The protocol is simplified into:

B : ***BankSetupFairBlindCertificate***$(X_B) = (h, h_1, h_2, h_3) = Y_B$

3.4.2 Escrow Agent Setup Protocol (performed once by E)

The escrow agent publishes his public keys $f_2 = g_2^{X_E}$, $f_3 = g_3^{X_E}$, where $X_E \in_R \boldsymbol{Z}_q$ is escrow agent's private key.

E : ***EscrowSetupFairBlindCertificate***$(X_E) = (f_2, f_3) = Y_E$

3.4.3 Customer Setup Protocol

The bank *B* associates customer *C* with *Identity* = $I = g_1^{u_1}$, where $u_1 \in G_q$ is generated by *C* and $g_1^{u_1} \neq 1$. The protocol also forces *C* to prove to *B* that *C* knows how to represent *I* with respect to g_1.

$C \leftrightarrow B$: ***ProofLogaritmRepresentationKnowledge***(I, g_1)

Readers may take note that at customer setup phase, *C* does not need to contact escrow agent *E*. Also notice that bank may allow a customer to have a certain number of blind certificate (depending on bank's policy), but nevertheless *C* only need to conduct this protocol once. We encapsulate the protocol into:

$C \leftrightarrow B$: ***CustomerSetupFairBlindCertificate***$(u_1) = I$

3.4.4 Blind Certificate Generation (over an authenticated channel between B and C)

Customer creates an intermediate value $I' = I^{s^{-1}} g_3^{s^{-1}} g_4^{t}$, then continues to construct $E_1 = g_2^{s} f_3^{m}$, $E_2 = g_3^{m}$ of g_2^{s} and proceeds to prove its correct construction with respect to *I'*. The construction of *I'*, E_1, E_2 are proven using proofs of equality logarithms. Note that during the verification step, *C* is expected to is expected to present a new blinded identity of a

specific structure. It forces *C* to use the committed value *s* as the blinding factor. Thus the coin contains $g_2^{\,s}$ and can be traced by decrypting (E_1, E_2).

1. C : $m, s, t \in_R \mathbf{Z}_q$, $I' = g_1^{u_1 s^{-1}} g_3^{s^{-1}} g_4^{\,t}$, $E_1 = g_2^{\,s} f_3^{\,m}$, $E_2 = g_3^{\,m}$
   $C \to B$: I', E_1, E_2
2. $C \leftrightarrow B$: ***EqLog***$((E_1, f_3), g_2, (E_2, g_3), \text{nil})$
   EqLog$((g_3, I'), (g_1, g_4), (E_1, g_2), f_3, (I, I'), (g_3, g_4))$
3. B : $w \in_R \mathbf{Z}_q$, $a' = g^w$, $b' = (Ig_2)^w$, $b'' = g_4^{\,w}$
   $B \to C$: a', b', b''
4. C : $A = (I' g_2 g_4^{\,t^{-1}})^s = g_1^{u_1} g_2^{\,s} g_3$, $z = h_1^{u_1} h_2^{\,s} h_3 \,[= A^{X_B}]$,
   $u, v \in_R \mathbf{Z}_q$, $a = (a')^u g^v$, $b = (b' b''^{\,t^{-1}})^{su} A^v \,[= A^{wu+v}]$,
   $c = H(A, z, a, b)$, $c' = c / u$
   $C \to B$: c'
5. B : $r' = c' X_B + w$
   $B \to C$: r'
6. C : $r = r'u + v \bmod q$
   $g^r \overset{?}{=} h^c a$, $A^r \overset{?}{=} z^c b$

The object *BlindCertificate* is *A* and the bank *B*'s signature $\sigma(A) = (z, a, b, r)$, with *s* as the secret part for *C*. *A* itself is the the blinded indentity of the customer (*BlindIdentity* $= A = Ig_2^{\,s} g_3$).

$C \leftrightarrow B$: ***GenerateFairBlindCertificate***$(I, Y_B, Y_E) =$ *BlindCertificate*, where Y_B and Y_E are bank's and escrow agent's public key, respectively.

3.4.5 Blind Certificate Verification (performed by C to any verifier V)

C proves the knowledge of ownership of his blind certificate by also guaranteeing *V* that he does so in a fair way (involving *E*'s public key).

1. C : $A_1 = g_1^{u_1} g_2^{\,s} \,[= A / g_3]$, $A_2 = f_2^{\,s}$
   $C \to V$: $A_1, A_2, A, (z, a, b, r)$
2. $C \leftrightarrow V$: ***EqLog***$[(A_1, g_2), g_1, (A_2, f_2), \text{nil}, c]$
3. V : $A_1 \overset{?}{\neq} 1$, $A_1 g_3 = A$
   $\sigma(A) \overset{?}{=} (z, a, b, r)$

We encapsulate the protocol into:

$C \leftrightarrow B$: ***VerifyFairBlindCertificate***(*BlindCertificate*)

3.4.6 Identity tracing protocol (performed between B and E over an authenticated channel)

The protocol is executed whenever bank *B* wants to know the true identity of a blind certificate. In this case, *B* simply sends the blind certificate verification transcript to the escrow agent *E*. Then *E* uses his private key to decrypt the ElGamal encryption (A_1, A_2) and

sends the decrypted value (i.e., $I = g_1^{u_1}$) back to *B*. Knowing *I* is sufficient for *B* to know the true identity of *C*. We simplify the protocol into:

$B \leftrightarrow E$: ***TraceIdentity***(*BlindCertificateVerificationTranscript*) = *Identity*

3.4.7 Blind certificate tracing protocol (performed between B and E over an authenticated channel)

Whenever *B* wants to know which blind certificate did *C* create after a blind certificate generation, *B* sends the suspicious blind certificate generation transcript to *E*. The escrow agent *E* decrypts the ElGamal encryption (E_1, E_2) to obtain the value g_2^s, and sends it to *B*. Then *B* performs a search to find the related blind certificate by comparing *A* from the the list of anonymous accounts with their associated blind certificates, with $A = Ig_2^s g_3$, where *I* is *C*'s identity (known from the certificate generation process). We encapsulate the protocol into:

$B \leftrightarrow E$: ***TraceBlindCertificate*** (*BlindCertificateGenerationTranscript*) = *BlindIdentity* or *BlindCertificate*

3.5 Group Signature Protocol

3.5.1 Brief Description

The concept of group signature scheme was invented by Chaum and van Heyst in [11]. It allows a group member to sign messages anonymously on behalf of the group. The signatures can be verified with respect to a single public key of the group and do not reveal the identity of the signer (the member who signed the corresponding message). The membership manager is responsible for system setup and for registering group members. The revocation manager has the authority to revoke the anonymity of a signature, to reveal the which group member actually signed the signature.

Some group signature schemes [11, 4] are not suitable for large groups. Other scheme such in [5] are a lot more efficient, where the length of signatures and the size of the group's public key is independent of the number of group members. In HiPS we choose the group signature scheme from Camenish and Michels [6] which is more efficient than [5].

3.5.2 Naming Equivalence in HiPS

Since we do not change any protocols in the group signature [6] that we use in HiPS, we only need to map the entity name and the protocols within [6] into HiPS. Notice that the bank *B* acts both as the membership manager and the revocation manager.

Group Signature [6] **Entities**	**HiPS Entities Equivalence**
Membership manager and revocation manager	*Bank B*
Group member	*Customer C, using a blinded identity*
Verifier	*Merchant M*

Table 2. *Group signature entities equivalence in HiPS*

Group Signature [6] **Naming**	**HiPS Naming Equivalence**
Membership manager setup protocol	*B* : ***BankSetupAnonymousAccount***
Revocation manager setup protocol	
Member registration protocol (over a secure anonymous channel)	*B* ↔ *C* : ***SetupAnonymousAccount*** (*CustomerIdentity*), or to be more precise in HiPS, ***SetupAnonymousAccount*** (*BlindIdentity*)
Group signature generation protocol	*C:* ***GroupSign***(*Message*) = *Signature*
Group signature verification protocol	*M* : ***VerifyGroupSignature***(*Message*, *Signature)*
Group signature tracing protocol	*B* : ***TraceGroupSignature***(*Message*, *Signature) = BlindIdentity*
Tracing verification (vertracing) protocol	Not used in HiPS

Table 3. *Group signature protocol equivalence in HiPS*

4 Complete Protocol Description

Here we describe the complete protocol of HiPS. The protocol consists of several major phases:

1. Bank, escrow agent and customer setup phase
2. Anonymous cashier's check withdrawal/generation
3. Assigning/affixing the destination/beneficiary of the anonymous cashier's check
4. Depositing the anonymous cashier's check
5. Actual payment phase, where the customer pays the merchant
6. Merchant deposit phase

Optionally, the customer can trust a mix-resender to deposit his checks to his anonymous bank account. Refer back to figure 1 for the scheme diagram of HiPS.

4.1 Setup Phase

Several steps must be conducted in the setup phase. The bank and the escrow agent must execute the appropriate setup protocols. The first setup protocols executed are the setup protocols to allow the bank to issue a fair blind certificate, and the escrow agent to revoke the issued blind certificate:

B : ***BankSetupFairBlindCertificate***

E : ***EscrowSetupFairBlindCertificate***

Then, in order to allow anonymous inter-account fund transfer using the anonymous cashier's check based on the FOLC protocol, both bank and the escrow agent must execute the setup protocol of FOLC:

B : ***BankSetupInterAccountAnonymousFundTransfer***

E : ***EscrowSetupInterAccountAnonymousFundTransfer***

Since the actual account-based anonymous payment process is done by using a group signature protocol, the bank must also execute the group signature setup protocol. In our scheme, the bank acts both as the group manager and as the revocation manager.

B : ***BankSetupAnonymousAccount***

Only after those setups has been completed, the customer can open a conventional bank account, linked to customer's true identity. The customer executes the customer setup protocol, which is actually the customer setup protocol in fair blind signature/certificate protocol (for creating a fair blind certificate) *and* customer setup protocol in FOLC (for fund transfer).

$C \leftrightarrow B$: ***CustomerSetupFairBlindCertificate***
(over a secure authenticated channel)

$C \leftrightarrow B$: ***CustomerSetupInterAccountAnonymousFundTransfer***
(over a secure authenticated channel)

The customer should then request a fair blind certificate to the bank, without the need to contact the escrow agent. He did this by proving to the bank that he had encrypted his identity with escrow agent's public key.

$C \leftrightarrow B$: ***GenerateFairBlindCertificate***(*Identity*) = *BlindCertificate*
(over a secure authenticated channel)

After receiving the blind certificate, preferably at some random time after receiving the certificate, the customer should be ready to create an anonymous bank account. The customer authenticates himself to the bank with his new blind certificate. With his new blind certificate, the customer can establishes a new anonymous bank account at the bank.

$C \leftrightarrow B$: ***VerifyFairBlindCertificate***(*BlindCertificate*)
(over a secure anonymous channel)

Immediately, he setup his anonymous account by executing the setup phase of the group signature protocol using his blinded identity in the blind certificate, making him as a group member of 'authenticated' anonymous customers at the bank.

$C \leftrightarrow B$: ***SetupAnonymousAccount***(*BlindIdentity*)
(over a secure anonymous channel)

4.2 Withdrawal of Anonymous Cashier's Check

Next, the customer needs to transfer some funds from his conventional bank account into his newly created anonymous account. At withdrawal phase, the customer creates several anonymous cashiers' checks (or just 'checks' for short) in several denominations from the bank to the amount he wants to transfer. For example, if the customer wants to transfer $500, he might choose to create one $250 check, two $100 checks and one $50 check. Since the check withdrawal protocol is actually a digital coin withdrawal protocol, the resulting check is untraceable.

$C \leftrightarrow B$: ***WithdrawCheck*** = *AnonymousCashiersCheck*
(over a secure authenticated channel)

4.3 Assigning the Destination of the Anonymous Cashier's Check

Immediately after withdrawal, the customer sets the 'destination' of his checks to his anonymous account number. In our case, we simply let the customer fill in the random challenge *c* as the hash of his anonymous account number and the check itself (no checks are created the same). His anonymous account number in the challenge actually attaches a commitment of the check to his anonymous account.

C : c = (*CustomerAnonymousAccountNumber*, *AnonymousCashiersCheck*)
$C \leftrightarrow C$: ***AssignCheckDestination***(*AnonymousCashiersCheck*, $H(c)$) =
PredestinedAnonymousCashiersCheck

4.4 Depositing the Anonymous Cashier's Check

Now the customer can deposit his checks anytime he wants. To obfuscate eavesdropper, the customer should not deposit all of his checks into his anonymous account at the same time. We suggest that the customer should deposit his checks in several *different* random times, such that the deposit times are randomly distributed over the average inter-withdrawal time span. Of course, it is possible to let the customer deposit his payment as soon as possible, but with the risk of possible statistical linking of withdrawal and deposit. Anyway, the algorithm to determine when should each check be deposited, really depends on the user requirement.

$C \rightarrow B$: *PredestinedAnonymousCashiersCheck*
(over a secure authenticated channel)

The bank then verifies the validity of the predestined checks, and increases the balance of the corresponding anonymous bank account. Readers should take note, that our deposit-into-anonymous-account phase is analogous to the payment phase in the digital coin protocols.

B : ***VerifyCheck*** (*PredestinedAnonymousCashiersCheck*)

4.5 Actual Anonymous Payment Phase

At payment phase, the merchant creates a payment slip containing necessary payment information, but without any customer identity information written on it. Then the merchant asks the customer to sign the payment slip using customer's private group signature key. The merchant then verifies the signature using bank's public group signature key. Upon verifying the signature, merchant will accept that the signature is a valid signature of a group member, without knowing who actually signed the payment slip.

$M \rightarrow C$: *PaymentSlip*

C : ***GroupSign***(*PaymentSlip*) = *PaymentSlipSignature*

$C \rightarrow M$: *PaymentSlipSignature*

M : ***VerifyGroupSignature***(*PaymentSlip*, *PaymentSlipSignature*)

(all of them are conducted over secure anonymous channel between C and M)

Note that it is possible for the customer conduct a payment even before the inter-account fund transfer (section 4.2 – 4.4), since HiPS is an off-line credit payment system (he can 'pay' the bills charged to his anonymous bank account later).

4.6 Merchant Deposit Phase

Later, the merchant can clear the signed payment slip to the bank. Bank will open the signed payment slip using his revocation key to reveal the blinded identity of the customer. Then the bank bills the customer. Alternatively, the bank can debit customer's corresponding anonymous account if there is sufficient balance, and then credits merchant's bank account.

$M \rightarrow B$: *PaymentSlip*, *PaymentSlipSignature*
(over secure authenticated channel between M and B)

B : ***TraceGroupSignature***(*PaymentSlip*, *PaymentSlipSignature)* = *BlindIdentity*

4.7 Delegating the Deposit Process to A Trusted Mix-resender

Alternatively, the customer can rely on a trusted mix-resender, especially if the customer wants to go off-line after the withdrawal process or to further obfuscate his network address. The customer slips the predestined checks into a digital envelope that can only be opened by the bank. Then the customer sends the digital envelope to the trusted mix-resender, which acts as a trusted party on behalf of the customer. The customer places his trust on the trusted mix-resender to forward each of his checks at preassigned times.

4.8 Anonymity revocation

Whenever there is a need to revoke suspicious withdrawal, deposit or payment transaction, the bank consults the escrow agent. All of these revocation protocols are conducted over a secure authenticated channel between the bank and the escrow agent.

4.8.1 Tracing a Blind Certificate

To reveal which fair blind certificate was generated from a suspicious blind certificate generation (recall that blind certificate generation is conducted over an authenticated channel), the bank gives the blind certificate generation transcript (*BlindCertificateGenerationTranscript*) to the escrow agent.

$B \leftrightarrow E$: ***TraceBlindCertificate***(*BlindCertificateGenerationTranscript*) = *BlindIdentity*

After the escrow agent recovers the blinded identity (which is also included in the fair blind certificate), the escrow agent submits the recovered blinded identity to the bank, which then will find the matching blind certificate.

4.8.2 Tracing the Owner of an Anonymous Account

Because HiPS uses a digital coin protocol for the inter-account fund transfer, it is possible that a customer gives his anonymous cashier's check to other customer's anonymous account. In case of such suspicious fund transfer and the bank wishes to know who is the owner of the anonymous account (which receives the fund), the bank must give the *BlindCertificate* verification transcript which is linked to that anonymous account, to the escrow agent.

$B \leftrightarrow E$: ***TraceIdentity***(*BlindCertificateVerificationTranscript*) = *Identity*

4.8.3 Tracing an Anonymous Cashier's Check

To know which anonymous cashier's check was created after a particular check withdrawal, the bank and the escrow agent executes the trace anonymous cashier's check protocol together.

$B \leftrightarrow E$: ***TraceAnonymousCashiersCheck*** (*CheckWithdrawalTranscript*) = *PredestinedAnonymousCashiersCheck*

4.8.4 Tracing the Owner of an Anonymous Cashier's Check

Continuing the example (section 4.8.2), the bank wishes to know the owner of the anonymous cashier's check in a suspicious fund transfer. Given a particular *Predestined-AnonymousCashiersCheck*, the bank and the escrow agent can recover the *Identity* of the owner.

$B \leftrightarrow E$: ***TraceOwner*** (*PredestinedAnonymousCashiersCheck*) = *Identity*

4.8.5 Tracing the Payer's True Identity of a Particular Payment

Recall that the payment phase uses a group signature protocol. To recover the true identity of the payer of a suspicious payment, the merchant submits the payment information to the bank as described in section 4.6, and let the bank executes the revocation protocol in section 4.8.2.

5 Discussion

5.1 Analysis of HiPS

We shall now analyze in detail how we achieve the stated requirement of HiPS. Smaller database storage requirement of spent 'coins' (or 'checks' in HiPS's context) can be achieved because we assume that the denomination of the checks for inter-account fund transfer are much higher than the digital coins which is used for direct payment. Therefore, for the same amount of money, we claim that we use fewer 'coins' in HiPS.

To make traffic analysis of inter-account fund transfer harder to statistically relate check withdrawal and check deposits, we suggest that the check withdrawal should be 'sliced' into smaller fixed-value checks and deposit each check at unpredictable times. To further obfuscate the fund transfer, the customer can trust a mix-resender to deposit the checks.

Since the only thing that the customer must do at payment phase is to sign a payment slip with a group signature protocol, it avoids multiple coin processing at payment phase. Moreover, the customer obviously does not need to carry the 'coins'. In addition, because any price or value can be affixed on the payment slip, HiPS does not have any divisibility problem.

By using a group signature protocol for the payment phase, we can achieve pseudo payment-unlinkability, where the merchant cannot discern whether two payments are made by the same customer or not. However, the bank must know which anonymous account to be billed, therefore from the bank's perspective, the payments made by a customer are linkable.

The blind signature protocol used extensively in HiPS provides strong anonymity level, and guarantees that the bank and the merchant cannot collude to reveal the anonymity of the customer. However, since we are using a 'fair' blind signature, the bank with the help of a trusted escrow agent can control the anonymity in case of a suspicious transaction. Notice that a malicious mix-resender can reveal the true identity of the customer.

HiPS is basically an off-line *credit* payment system, which means at the actual payment phase the merchant does not know whether the customer is lying or not about the balance in customer's anonymous account. Only after the merchant deposits the signed payment slip, the merchant will know the customer was lying or not. If the balance is not sufficient to cover the signed payment slips, the bank may choose to give some time for the customer to settle the payment bills, or revoke the identity of the anonymous bank account if the customer refuses to credit his anonymous bank account.

Since HiPS is a credit system, one may argue how HiPS is comparable to off-line electronic *cash* systems. Our argument is that even in electronic cash systems, at payment phase the merchant also does not know whether the customer is lying about his payment (i.e.

double spending his coin) or not. Double spending detection is only done by the bank after the merchant deposits the coins, and automatically recover the identity of the customer.

Readers should also note that the customer may 'double-deposit' his checks, notably to other customer's anonymous account. Since in HiPS we have much fewer 'coins' (checks) to manage, the double-deposit detection process would be much faster.

5.2 Implementation Scenarios

There are several possible scenarios to implement HiPS, but due to space limitation, we only elaborate a few of them. In one scenario, the customer is assumed to have a PC connected to the Internet. The customer manages all of his conventional and anonymous bank account over the Internet (section 4.1 – 4.4). But to shop in the real world or on the Internet, he would *only* need his smartcard with group signature capability (section 4.5). The customer can even start shopping with his smartcard before he has credited his anonymous account.

HiPS can also be modified into an on-line debit system if the merchant has an on-line connection to the bank and the customer always maintain enough balance in his anonymous bank account.

5.3 Open Problems

There is a lot of room for improvements for HiPS. First, optimal algorithms to obfuscate traffic analysis of check withdrawals and check deposits must be devised. Note that the algorithm depends on the real-world requirement, which must be explored in detail. Second, the possibility to use time-dependent cryptography to lower customer's trust to the mix-resender should be examined. Third, to achieve higher degree of payment-unlinkability, the possibility to use multiple anonymous bank accounts (therefore with multiple blinded identity) with their proper balance management algorithm should be explored. The balance management of those anonymous accounts relates to the buying pattern of the consumer. The use of multiple anonymous accounts can also be used for loyalty program with privacy protection.

6 References

[1] Mihir Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communication Security*. Association for Computing Machinery, 1993.

[2] Stefan Brands. An efficient off-line electronic cash system based on the representation problem. Report CS-R9323, Centrum voor Wiskunde en Informatica, March 1993.

[3] Jan L. Camenish, Jean-Marc Piveteau and Markus A. Stadler. An Efficient Electronic Payment System Protecting Privacy. In *Proceedings of ESORICS '94.* Lecture Notes in Computer Science 875, Springer-Verlag, 1995.

[4] Jan L. Camenish. Efficient and generalized group signatures. In *Advances in Cryptology – proceedings of EUROCRYPT 97*, Lecture Notes in Computer Science 1233, Springer-Verlag, 1997.

[5] Jan L. Camenish and Markus A. Stadler. Efficient Group Signatures Schemes for large Groups. In *Advances in Cryptology – proceedings of CRYPTO 97,* Lecture Notes in Computer Science 1294, Springer-Verlag, 1997.

[6] Jan Camenisch and Markus Michels. *A Group Signature Scheme Based on an RSA-Variant*, BRICS RS-98-27. Basic Research In Computer Science, Dept. of Computer Science, University of Aarhus, Denmark, 1998.

[7] Agnes Chan, Yair Frankel, Philip MacKenzie and Yiannis Tsiounis. Mis-representation of identities in e-cash schemes and how to prevent it. In *Advances in Cryptology – proceedings of ASIACRYPT 96.* Lecture Notes in Computer Science 1163, Springer-Verlag, 1996.

[8] Agnes Chan, Yair Frankel and Yiannis Tsiounis. Easy come – easy go divisable cash. In *Advances in Cryptology – proceedings of EUROCRYPT 98.* Lecture Notes in Computer Science 1403, Springer-Verlag, 1998.

[9] David Chaum. Untraceable electronic mail, return address, and digital pseudonyms. In *Communications of the ACM*, volume 24, number 2, February 1981.

[10] David Chaum, Amos Fiat and Moni Naor. Untraceable electronic cash. In *Advances in Cryptology – proceedings of CRYPTO 88,* Lecture Notes in Computer Science 403, Springer-Verlag, 1990.

[11] David Chaum and Eugene van Heyst. Group signatures. In *Advances in Cryptology – proceedings of EUROCRYPT 91*, Lecture Notes in Computer Science 547, Springer-Verlag, 1991.

[12] Tony Eng and Tatsuaki Okamoto. Single term divisible electronic coins. In *Advances in Cryptology – proceedings of EUROCRYPT 94.* Lecture Notes in Computer Science 960, Springer-Verlag, 1995.

[13] Niels Ferguson. Single term off-line coins. In *Advances in Cryptology – EUROCRYPT 93,* Lecture Notes in Computer Science 765, Springer-Verlag, 1994.

[14] Yair Frankel, Yiannis Tsiounis and Moti Yung. Fair Off-Line e-Cash Made Easy. In *Advances in Cryptology – proceedings of ASIACRYPT 98.* Lecture Notes in Computer Science 1514, Springer-Verlag, 1998.

[15] Björn Markus Jakobsson. *Privacy vs. Authenticity*. Ph.D. thesis. University California, San Diego. 1997.

[16] Hugo Krawczyk. Blinding Credit Card Numbers In SET. In In *Financial Cryptography, 3nd International Conference, FC 99,* Lecture Notes in Computer Science 1648, Springer, 1999.

[17] Stephen H. Low, Nicholas F. Maxemchuk and Sanjoy Paul. Anonymous Credit Cards. In *Proceedings of the 2nd ACM Conference on Computer and Communication Security.* 1994.

[18] Stephen H. Low, Nicholas F. Maxemchuk and Sanjoy Paul. *Anonymous Credit Cards amd Its Collusion Analysis*. AT&T Bell Laboratories. Murray Hill, NJ, 1994.

[19] Y. Mu and V. Varadharajan. A new scheme of credit based payment for electronic commerce. In *Proceedings of 23rd Local Area Networks Conference*, IEEE Computer Society, 1998.

[20] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In *Advances in Cryptology – CRYPTO 91.* Lecture Notes in Computer Science 576, Springer-Verlag, 1992.

[21] Tatsuaki Okamoto. An efficient divisible electronic cash scheme. In *Advances in Cryptology – CRYPTO 95.* Lecture Notes in Computer Science 963, Springer-Verlag, 1995.

[22] Claus P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology,* no.4 vol.3, 1991.

[23] *Secure Electronic Transaction.* Documentations. Available at *http://www.setco.org/*

[24] Markus Stadler, Jean-Marc Pivateau, Jan L. Camenish. Fair Blind Signatures. In *Advances in Cryptology – proceedings of EUROCRYPT 95.* Lecture Notes in Computer Science 921, Springer-Verlag, 1995.

[25] Yiannis S. Tsiounis. *Efficient electronic cash: new notions and techniques.* Ph.D. thesis. Department of Computer Science, Northeastern University, 1997.

[26] Yacov Yacobi. Efficient electronic money. In *Advances in Cryptology – ASIACRYPT 94.* Lecture Notes in Computer Science 917, Springer-Verlag, 1995.