An Anonymous Credit Card System

Elli Androulaki and Steven Bellovin

Technical Report CUCS-010-09

Abstract. Credit cards have many important benefits; however, these same benefits often carry with them many privacy concerns. In particular, the need for users to be able to monitor their own transactions, as well as bank's need to justify its payment requests from cardholders, entitle the latter to maintain a detailed log of all transactions its credit card customers were involved in. A bank can thus build a profile of each cardholder even without the latter's consent. In this technical report, we present a practical and accountable anonymous credit system based on ecash , with a privacy preserving mechanism for error correction and expense-reporting.

1 Introduction

Motivation: Credit Cards vs. Consumer's Privacy. Credit cards have many useful properties. Apart permitting delayed payment, they provide users with logs of their own transactions, receipts, and the opportunity to challenge and correct erroneous charges. However, these same benefits are a privacy risk: banks can use the same information to build and sell profiles of their customers. We need a system that preserves the benefits of credit cards without violating users' privacy.

In the context of e-commerce, privacy of an entity is being able to transact with other entities without any unauthorized outsider being able to acquire any transaction-related information. In addition, no party should be able to build profiles of any other party based on purchases without the latter's consent. Being closely related to their owners'identities, credit cards' extended use constitutes a serious threat to consumers' privacy: *Frequent occurrences of credit card losses, credit card number based-impersonation attacks as well as human nature errors, i.e. overcharge of a client, make it necessary for cardholders to be able to monitor their own transaction activity and for merchants to provide banks with detailed description of each credit card transaction. Under the umbrella of the need of immediate charge justification/correction, each bank, which is no more trusted than the people operating it, acquires a global view of its customers' transaction activity.* None of the currently deployed credit card systems offer consumer's privacy towards banks. Given the fact that the percentage of credit card-based purchases is increasing, a deployable privacy preserving credit card system has become quite important. This is the problem we have solved.

Privacy Preserving Payment Mechanisms. Anonymous Credit Cards were introduced in 1994 by Low etl. [LPM94]. However, their scheme involves many trusted parties and offers no expense report or error correction service. Current schemes have some of the privacy problems mentioned earlier:

Ecash. Ecash [CHL05,DCN90] is a substitute of money on the Internet which cannot be faked, copied or spent more than once. It is known to provide absolute anonymity, namely no one can relate a particular ecash coin (ecoin) with its owner. One would argue that ecash could solve the problem we described before. Consumers can indeed buy anonymous ecoins from a bank/mint and use them in their online transactions. However, ecash is a prepayment based — as opposed to the most popular credit based — scheme, and used strictly for online transactions; additionally, the complete anonymity it guarantees gives no opportunities for error correction or expense reporting.

Anonymous Debit Cards(ADCs). Anonymous Debit Cards are prepaid ecash -based cards, which are recharged by cardholders and used to pay for goods anonymously. However, their use is very limited; among the reasons are the lack of error correction and proof of purchase mechanisms; additionally, they operate in a debit rather than a credit fashion, i.e. the amount of money paid by it, is subtracted from one's account, when the card is initially obtained.

Our Contribution. In this technical report we introduce a privacy-preserving credit card mechanism which can be applied in current credit card systems. In particular, we present a realizable protocol to support "credit card"-based online and offline transactions, in which banks, unless authorized by the cardholder, do not acquire any knowledge of their customers' transactions. At the same time, cardholders are provided with detailed reports regarding their purchases and may participate on any type of merchant credit card offers. For the purposes of our system, we made use of a combination of two types of the compact cash [CHL05] scheme for payments, and a combination of blind [CL02] and plain digital signatures for the rest of our system's operations.

In the following sections we will briefly present our system's main functions. For space reasons, we have put the more detailed presentation of all the services provided by our scheme in [**?**]. It also includes full security proofs.

2 System Architecture

A typical credit card mechanism consists of cardholders (consumers), merchants (sellers), Acquiring Banks, Card-Issuing Banks and Credit Card Associations.

When eligible to issue a credit card, a consumer collaborates with a *Card Issuing Bank* she maintains an account with. The *Card Issuing Bank* charges the *cardholders* for outpayment and bears the risk of fraudulent use of the card. On the other hand, *Merchants*, who are eligible to receive credit card payments, are in agreement with an *Acquiring Bank* authorized to receive payments on behalf of them. Banks are organized in *Credit Card Associations*, that set the transaction rules between *Card Issuing* and *Acquiring Banks.*

For convenience, in the following sections, we will refer to merchant as 'he', to client as 'she' and to any type of Bank as 'it'. In addition, we will use the following acronyms: CIB for Card Issuing Banks, ABs for Acquiring Banks, ACCA for our Anonymous Credit Card Association and ACCs for the anonymous credit cards of our system.

As we aim to to create a realizable system, we assume that our adversary has all the powers and motives a real Bank/merchant/cardholder or groups of them would have. In addition, we assume that all parties have as main objective to increase their profit and would not act against it. More specifically,

- All types of Banks are "honest but curious", namely while they are trusted to do their functional operations correctly, they may collude and combine the information they possess to track their customers' activities.
- Merchants are mostly interested in receiving their payment. However, they may try to "deceive" a customer into making her pay more. In addition, for advertising purposes, merchants may be motivated — if cost effective — to collaborate with Banks to profile their customers. In offline transactions the merchant knows the customer's face. However, any attempt to identify a customer manually (i.e. by comparing pictures online) is not cost effective and is thus highly unlikely.
- Cardholders may try to "cheat", i.e. not to pay, to attempt to forge an ACC or to frame another customer by accusing her of having cheated.

3 Requirements

Privacy and the rest of our system requirement will be presented in this section. For simplicity and as we assumed collaboration between Banks, we will refer to all of them as if they constituted a unite organization, a general Bank.

Privacy. Given a simple (online) cardholder-merchant transaction, no (client/merchant)-unauthorized third party — including the Bank — should be able to gain any information regarding that particular transaction or link it to a particular cardholder even with merchant's collaboration (*Customer Anonymity w.r.t Bank* *and merchants*). In addition, given a cardholder, tracing her ACC based transactions should be impossible (*non-Traceability*). However, we require that the privacy provided in our system is *conditional*: guaranteed for honest cardholders, but immediately revoked for dishonest ones.

As mentioned before, one of our fundamental requirements is system's **Deployability**, i.e. our protocols should be possible to be applied on the current credit card systems' architecture described in previous section. **Credit Card Unforgeability** and **non-Transferability** are also required, as our cards should not be forgeable or used by any third party. It should be possible for cardholders to track their transactions (**Expense Report Service**) and provide an undeniable proof of any mischarge (**Error Correction Service**) without endangering their privacy. Privacy preserving **Loss Recovery** of the card and **Special** interest rate **offers** — i.e. card promotion agreements between the Bank and the merchants — common in current credit card systems should also be supported.

4 Anonymous Credit Card System

Our credit mechanism can be viewed as a long term loan. The cardholder is credited the amount she borrows to transact, while credit limit is the highest price that loan can get. The amount borrowed is outpaid in a predefined interest rate as in current credit card systems. To avoid charges for a bigger amount than the one she has spent, the cardholder is required — at the end of each predefined time period, which is usually a month — to provide undeniable proof of the amount of money she has spent.

Identities-Master Secrets. Entities in our system (cardholders, banks, merchants) are identified by the signature keys they issue after entering the system. However, the transaction activity of each cardholder is entirely enabled by a master secret, Ms , which is part of her secret signature-related information. As impossible to be certified without a bank's collaboration, proof of possession of a valid Ms denotes a valid cardholder and any attempt to cheat would result in recovery of the identity and Ms of the misbehaving entity.

Bank Data Management. CIBs maintain a large database consisting of their customers' account information. In particular, CIBs manage D_{debit} for customers' debit accounts, D_{credit} for the credit accounts, D_{anon} for the temporary anonymous accounts used only in online ACC transactions and D_{hist} which is used as a log of D_{credit} and D_{anon}. ABs — which may/may not be CIBs — are connected to merchants' debit accounts.

System Operation. ACCs are issued with the collaboration of a CIB and the intended cardholder, who has already opened an account with the former. ACC's functionality is based on two types of ecash wallets, which are withdrawn at the ACC issuing procedure: the payment wallet W_p and the identity one W_{id}. The two wallets have the same number of ecoins , which is in proportion to the credit limit of the cardholder. However, they differ to their traceability mechanism in case of double-spending. If double-spent, W_p only reveals the identity of the double-spender. On the contrary, if part of W_{id} is double-spent, Ms of the double-spender and all the ecoins withdrawn by the latter are revealed. W_p's traceability attributes are used in the loss recovery protocol, while W_{id}'s are used in identifying malicious parties.

Payment procedure basically consists of a merchant authentication phase followed by two ecash spending procedures. In offline purchases, the cardholder uses merchant's machine to spend the same number of ecoins from both wallets, W_p and W_{id}. Spent ecoins ' value corresponds to the price of the product. Transaction details are signed by merchant and stored (as receipt) in the card. Merchant is paid when he deposits the ecoins he has obtained through his customers to his AB. Apparently, any attempt on cardholder's side to cheat, i.e. by using two copies of the same card will reveal — because of the W_{id} double-spending defense mechanism — the cheater's Ms . To enable online purchases, customer sets up an anonymous account with her CIB. In particular, she makes use of an ATM machine where she spends the amount of ecash she would like to use for the online purchases from both wallets. Both parties agree on secret information which would enable a cardholder to authorize the intended merchant's AB to substract safely money from that anonymous account. Secret information is renewed so that replay attacks can not succeed. At the end of each month, the cardholder deposits in person the unspent part of her wallets to her CIB, so that the former is charged accordingly.

Each cardholder is encouraged each fixed time interval to back up the content of her ACC. In case of loss, the backed up ACC content is simply copied to another card. Faking ACC's loss and making use of both ACCs (the new and the old) will reveal customer's identity, who is imediately charged[1]. In all cases, the content of the card is encrypted so that only its legal owner can read its content. Expense Report Service is achieved through a decryption of the transaction-part of the card which can take place locally, at client's machine.

ACC promotion offers are realized through coupons obtained by each merchant from the ACCA. Coupons are stored in the ACC at the end of the transaction and deposited by the cardholder to her CIB, for the former to obtain better interest rates. To preserve cardholder anonymity coupons' deposit procedure is done in two phases.
In what follows we will use the following notation:

- $\mathsf{Sig}_{\mathsf{C}}(\mathsf{Msg})$ $(\mathsf{Sig}^{\mathsf{H}}_{\mathsf{C}}(\mathsf{Msg}))$ for the signature of C on Msg $(H(\mathsf{Msg}))$.
- $\mathsf{B[G]Sig}_{\mathsf{M}}(\mathsf{Msg})$ $(\mathsf{B[G]Sig}^{\mathsf{H}}_{\mathsf{M}}(\mathsf{Msg}))$ for the blind [group] signature of Mon $\mathsf{Msg}(H(\mathsf{Msg}))$.
- $\{\mathsf{Msg}\}_K$ for the encryption of Msg under key K. For efficiency, we induct every asymmetric encryption a symmetric one. Therefore, $\{\mathsf{Msg}\}_{PK}$ denotes $\{K\}_{PK}||\{\mathsf{Msg}\}_K$ for a random K.

4.1 Building Blocks

ECash. An ecash system consists of three types of players: a *bank*, *cardholders* and *merchants*. Below are the supported procedures (see [CHL05]).

- $(pk_{\mathsf{B}}, sk_{\mathsf{B}}) \leftarrow \mathsf{EC.BKeyGen}(1^k, params)$ is the key generation algorithm for the bank.
- $(pk_{\mathsf{C}}, sk_{\mathsf{C}}) \leftarrow \mathsf{EC.UKeyGen}(1^k, params)$ is the key generation algorithm for cardholders.
- $\langle W, \top \rangle \leftarrow \mathsf{EC.Withdraw}(pk_{\mathsf{B}}, pk_{\mathsf{C}}, n)$ $[\mathsf{C}(sk_{\mathsf{C}}), \mathsf{B}(sk_{\mathsf{B}})]$. The cardholder C withdraws a wallet W of n coins from the bank B.
- $\langle W', (S, \pi) \rangle \leftarrow \mathsf{EC.Spend}(pk_{\mathsf{M}}, pk_{\mathsf{B}}, n)$ $[\mathsf{C}(W), \mathsf{M}(sk_{\mathsf{M}})]$. The cardholder C spends an ecoin by giving it to the merchant M. C gets the updated wallet W, and M obtains an ecoin (S, π) where S is a serial number and π is a proof.
- $\langle \top/\bot, L' \rangle \leftarrow \mathsf{EC.Deposit}(pk_{\mathsf{M}}, pk_{\mathsf{B}})$ $[\mathsf{M}(sk_{\mathsf{M}}, S, \pi), \mathsf{B}(sk_{\mathsf{B}}, L)]$. M deposits (S, π) into its account in the bank B. L' is the updated list of the spent coins (i.e. (S, π) is added to the list).
- $(pk_{\mathsf{C}}, \Pi_G) \leftarrow \mathsf{EC.Identify}(params, S, \pi_1, \pi_2)$. Given two coins with the same serial number, i.e. (S, π_1) and (S, π_2), B finds the identity of the double-spender pk_{C} and the corresponding proof Π_G.
- $\top/\bot \leftarrow \mathsf{EC.VerifyGuilt}(params, S, pk_{\mathsf{C}}, \Pi_G)$. It verifies the proof Π_G that the cardholder pk_{C} is guilty of double-spending coin S.

In [CHL05], except for the conventional ecash system, Camenisch, Hohenberger and Lysyanskaya introduce an ecash system with an extended double-spending detection mechanism: apart from the identity of the double-spender, all the digital coins she has withdrawn are revealed. In particular, in addition to the aforementioned procedures (which in the second scheme are slightly different), the [CHL05]'s second scheme is extended with the following procedures:

- $((S_1, \Pi_1), \ldots, (S_m, \Pi_m)) \leftarrow \mathsf{EC.Trace}(params, S, pk_{\mathsf{C}}, \Pi_G, D)$. Given a double-spender pk_{C} of a digital coin with serial S and a proof of guilt Π_G, such that $\mathsf{EC.VerifyGuilt}(params, S, pk_{\mathsf{C}}, \Pi_G)$ accepts, $\mathsf{EC.Trace}$ outputs the serial numbers $(S_1, \ldots, S_m)$ of all the digital coins pk_{C} has withdrawn along with the corresponding proofs of ownership $(\Pi_1, \ldots \Pi_m)$.
- $\top/\bot \leftarrow \mathsf{EC.VerifyOwnership}(params, S, \Pi, pk_{\mathsf{C}})$. Given a double-spender pkc, $\mathsf{EC.VerifyOwnership}$ verifies that a digital coin with serial S has been withdrawn by him.

Secure e-cash scheme satisfies the following conditions: (1) *Correctness.* If an honest cardholder runs $\mathsf{EC.Withdraw}$ with an honest bank, then neither will output an error message. If an honest cardholder runs $\mathsf{EC.Spend}$ with an honest merchant, then the merchant accepts the coin. (2) *Balance.* No collection of cardholders

[1] Anonymity revocation in this case is not absolute; only the identity of the person having double-spent is revealed and not her master secret.

and merchants can ever spend more coins than they withdrew. (3) *Identification of double-spenders.* Suppose the bank B is honest, and M_1 and M_2 are honest merchants who ran the $\mathsf{EC.Spend}$ protocol with the adversary whose public key is pk_{C}. Suppose the outputs of M_1 and M_2 are (S, π_1) and (S, π_2) respectively. This property guarantees that, with high probability, $\mathsf{EC.Identify}(params, S, \pi_1, \pi_2)$ outputs a key pk_{C} and proof Π_G such that $\mathsf{EC.VerifyGuilt}$ $(params, S, pk_{\mathsf{C}}, \Pi_G)$ accepts. (4) *Anonymity of cardholders.* B, even when cooperating with any collection of malicious cardholders and merchants, cannot learn anything about a cardholder's spendings other than what is available from side information from the environment. (5) *Exculpability.* When S is a coin serial number not double-spent by cardholder $\mathsf{C}$with public key pk_{C}, the probability that $\mathsf{EC.VerifyGuilt}(params, S, \Pi_G, pk_{\mathsf{C}}, n)$ accepts is negligible. The extended ecash version, we mentioned before also suppports (6) *Full Transaction traceability of double-spenders.* In particular, given a double-spender pk_{C} on an ecoin with serial S and proof of guilt Π_G, $\mathsf{EC.Trace}(params, S, pk_{\mathsf{C}}, \Pi_G, D)$ outputs $((S_1, \Pi_1), \ldots, (S_m, \Pi_m))$ so that for $\mathsf{EC.VerifyOwnership}(params, S_i, \Pi_i, pk_{\mathsf{C}})$ accepts with high probability for $i = 1, \ldots, m$.

We will use the first of the two compact ecash schemes presented here, for the payment wallet W_p and the second one for the identity one W_{id}. To differentiate the procedures of the two schemes that have the same name, we will use the subindex p when we refer to the procedures of the first scheme and the subindex id, for the procedures of the second one.

Blind Signature Schemes. In this section, we describe the definition and security of the blind signatures. See,[JLO97,CL02] and references within for more details.

Players. For our context, we can say there are two types of players: the *bank* and the *cardholders*. A cardholder requests the bank to generate a signature on a message m. Then the bank generates a signature on m without knowing the message m.

Procedures. A blind digital signature scheme consists of three procedures:

- $(pk_{\mathsf{B}}, sk_{\mathsf{B}}) \leftarrow \mathsf{BS.KeyGen}(1^k)$. This is a key-generation algorithm that outputs a public/secret key pair $(pk_{\mathsf{B}}, sk_{\mathsf{B}})$.
- $\langle \top/\bot,\ \sigma/\bot \rangle \leftarrow \mathsf{BS.Sign}(pk_{\mathsf{B}})[\mathsf{B}(sk_{\mathsf{B}}), \mathsf{C}(m)]$. At the end of this interactive procedure, the output of the bank B is either *completed* or *not-completed* and the output of the cardholder C is either the signature (σ) or a failure sign $(\bot)$.
- $\top/\bot \leftarrow \mathsf{BS.Verify}(m, \sigma, pk_{\mathsf{B}})$ is a verification algorithm.

Security Properties. A blind signature scheme is said to be secure if it satisfies the following requirements:

- *Unforgeability.* Only the bank who owns the secret key sk_{B} can generate valid signatures.
- *Blindness.* The bank B does not learn any information about the message m on which it generates a signature σ.

Blind Group Signature Schemes. Here we will describe the definition and security of group blind signatures. See [LR98] for more details.

Players. For our context, we can say there are two types of players: the ACCA, the merchants and the cardholders. In particular, the ACCA is the manager of the merchants' group and merchants are required to provide signatures on behalf of their group to cardholders, on quantities which are blind to them.

Procedures.

- $(\mathsf{bgpk}, \mathsf{bgsk}) \leftarrow \mathsf{BGS.Setup}(1^k)$. This algorithm generates a group public key bgpk and the ACCA's master secret key bgsk .
- $\langle \mathsf{bgsk}_{\mathsf{M}}, \mathsf{JLog}_{\mathsf{M}} \rangle \leftarrow \mathsf{BGS.Join}(\mathsf{bgpk})[\mathsf{M}, ACCA(\mathsf{bgsk})]$. When this interactive join procedure ends, the merchant M obtains a secret signing key $\mathsf{bgsk}_{\mathsf{M}}$, and the ACCA (group manager) logs the join transcript in the database D.

- $\sigma \leftarrow \mathsf{BGS.Sign}(\mathsf{bgpk})[\mathsf{M}(\mathsf{bgsk}_{\mathsf{M}}), \mathsf{C}(m)]$ and $\top/\bot \leftarrow \mathsf{BGS.Verify}(\mathsf{bgpk}, m, \sigma)$, the signature and signature verification algorithm of message m.
- $\mathsf{M} \leftarrow \mathsf{BGS.Open}(\mathsf{bgsk}, \sigma, D)$. This algorithm determines the identity of the party having generated the signature σ. It can only be performed by the group manager, i.e. the ACCA.

Security Properties Apart from the *Unforgeability* and *Blindness* properties, which we have already introduced, blind group signatures offer *Signer Anonymity* — since noone but the ACCA can determine which group member produced it — and *Undeniable Signer Identity*, i.e. the ACCA can always detect the exact signer of a signature.

4.2 System Operations

Setup. Every Bank B participating in the ADS generates a signature key pair: $(pk_{\mathsf{B}}^{s}, sk_{\mathsf{B}}^{s})$, which identifies it. To facilitate the withdrawal of the ecash wallets mentioned before, B makes the appropriate setup to support the two compact ecash schemes in [CHL05] and runs EC.BKeyGen to generate $(pk_{\mathsf{B}}^{w},\ sk_{\mathsf{B}}^{w})$. For the purposes of authentication, efficiency and accountability in online transactions, each CIB publishes an array of hashes $\underline{\mathsf{Hash}}_{ot}$ and a gateway communication hash H_{GB}.

Every **merchant** M collaborates with his AB, AB, to issue a signature key pair $(pk_{\mathsf{M}}^{s}, sk_{\mathsf{M}}^{s})$ with the corresponding certificate $\mathsf{Cred}_{\mathsf{M}} = \mathsf{Sig}_{\mathsf{B}}(pk_{\mathsf{M}}^{s})$.

On the other hand, each **customer** C who opens an account with a CIB CIB collaborates with the latter in EC.UKeyGen procedure of both compact ecash schemes [CHL05]), mentioned in subsection 4.1, to issue two signature key pairs:

$$(pk_{\mathsf{C}}^{p}, sk_{\mathsf{C}}^{p}), (pk_{\mathsf{C}}^{id}, sk_{\mathsf{C}}^{id}).$$

Although each one of the pairs above individually identifies C, for convenience, we will refer to both pairs as $(pk_{\mathsf{C}}^{s}, sk_{\mathsf{C}}^{s})$.

The **ACCA** chooses and publishes the transaction related hashes H_{ot}, H_t and H_r.

ACC Issue. Let that customer C is eligible for a credit limit $\mathsf{L}_{\mathsf{credit}}$ with a CIB CIB. CIB and C collaborate in $EC.Withdraw_p(sk_{\mathsf{C}}^{p},\ \mathsf{L}_{\mathsf{credit}})$ and $\mathsf{EC.Withdraw}_{\mathsf{id}}(sk_{\mathsf{C}}^{id},\ \mathsf{L}_{\mathsf{credit}})$ for the latter to withdraw the payment W_p and identity W_{id} wallets. In both withdrawal procedures, C provides a sk_{C}^{s} related password, $pass_{pin}$. In the issued ACC is also stored public information regarding the Banks participating in ACCA (params).

Apart from $pass_{pin}$, the cardholder chooses a set of passwords: a backup $pass_e$ password — from which her backup encryption key pair $(pk_{\mathsf{C}}^{e}, sk_{\mathsf{C}}^{e})$ — is derived and $pass_e^t$, $pass_e^w$, $pass_e^c$ — which correspond to three encryption key-pairs

$$(pk_{\mathsf{C}}^{e_t}, sk_{\mathsf{C}}^{e_t}), (pk_{\mathsf{C}}^{e_w}, sk_{\mathsf{C}}^{e_w}) \text{ and } (pk_{\mathsf{C}}^{e_c}, sk_{\mathsf{C}}^{e_c}),$$

that serve for encryption of transaction, wallet and coupon part of the card as we will describe later on. Each cardholder also agrees on two hashes with Bank H_K and H_{CB}.

It is noticeable that no C-related identification information is included in the ACC, other than the information one can infer from the credit limit of a cardholder (which equals the value in the wallets). On the other hand, because of ecash *properties, only the individual who issued the ACC, i.e. withdrew the wallets, is able to use it, i.e. spend part of the wallets consisting it.*

Payment

Offline Payment. It takes place between the cardholder C, who participates through her ACC (anonymously), and a merchant M.

1. M provides $\mathsf{Cred}_{\mathsf{M}}$ to C, who checks its validity using params.
2. M signs a hash of the transaction details, $T_{det} = \{date,\ time,\ product,\ price\}$, and provides C with

$$\mathsf{Cred}_{\mathsf{M}},\ T_{det},\ \mathsf{Sig}_{\mathsf{M}}^{H_t}(T_{det}),$$

where *product* is all the product related info stored in its bar code.

3. C verifies the *product* information and inserts her $pass^w_e$ to have her W_p and W_{id} wallets decrypted,
4. C enters her $pass_{pin}$ to run $\mathsf{EC.Spend_p}(sk^p_{\mathsf{C}},\ price)$ - $\mathsf{EC.Spend_{id}}(sk^{id}_{\mathsf{C}},\ price)$ and spends *price* value from both wallets. Let W'_p and W'_{id} be the remaining wallets.
5. M calculates $\mathsf{Rec_T} = \mathsf{Sig}^{H_r}_{\mathsf{M}}(T_{det}{-}fin)$ and sends it to C, while he provides C with a printed transaction record.
6. C encrypts $\mathsf{Rec_T}$ and $W_p - W_{id}$ using her $pass^t_e$, $pass^w_e$ respectively into:

$$E_{T_{det}} = \{T_{det}||Rec_T\}_{pk^{e_t}_{\mathsf{C}}} \text{ and } E_{W_{p,id}} = \{W'_p||W'_{id}\}_{pk^{e_w}_{\mathsf{C}}}.$$

To receive his payment, M simply deposits to his AB, AB, the ecoins he has received from his customers. AB contacts each customer's CIB[2], CIB, and collaborates with the latter to run $\mathsf{EC.Deposit_p}$ and $\mathsf{EC.Deposit_{id}}$ (see section 4.1). Depending on whether double-spending has occurred, M may receive credit for each pair of payment-identity ecoin deposited. If no double-spending is detected, CIB and AB make the required transfer to M's bank account. If there a double-spending is detected, CIB runs $\mathsf{EC.Identify_{id}}$ on the double-spent identity ecoins to reveal the double-spenders' pk^{id}_{C}. In addition, CIB may run $\mathsf{EC.Trace_{id}}$ to trace all the identity ecoins pk^{id}_{C} double-spender has withdrawn.

Online Payment. It is performed in two stages: initially the cardholder C collaborates with her CIB, CIB, to set up one or more Anonymous Accounts (*Anonymous Account Setup*), which C addresses to make her online purchases (*Transaction Payment*).

Anonymous Account Setup. The two parties involved, C and CIB collaborate through her ACC and an ATM repsectively. Let that C has decided to open an anonymous account of M_{ot} value for her online transactions.

1. C interacts with CIB into
$$\mathsf{EC.Spend_p}(sk^p_{\mathsf{C}}, M_{ot}) \text{ and } \mathsf{EC.Spend_{id}}(sk^{id}_{\mathsf{C}}, M_{ot})$$
to spend M_{ot} from her W_p, W_{id} wallets respectively.
2. C chooses m, h_{ot}, a random R_{ot} number and acknowledges CIB.
3. Chashes R_{ot} m times using hash $H_{ot} = \underline{\mathsf{Hash}}_{ot}[h_{ot}]$: $(H^{(i)}_{ot}(R_{ot}), i = 1\ldots, m)$. Let
$$A^x_{\mathsf{C}} \ = \ H^{(x)}_{ot}(R_{anon}).$$
A^m_{C} will be the anonymous account number for the first transaction and in general the $A^{m-(i-1)}_{\mathsf{C}}$ will be the account number for the i^{th} online transaction. C sends $A^m_{\mathsf{C}},\ m, h_{ot}$ to CIB.
4. C establishes a pseudonym P^C_α, i.e. the public part of a key-pair (pk_P, sk_P), for which she gets the corresponding certificate:
$$Cert_{P^{\mathsf{C}}_\alpha} = \mathsf{Sig}_{\mathsf{CIB}_{ot}}(pk^{\mathsf{C}}_\alpha),$$
where CIB_{ot} is the online transaction section of CIB.
5. CIB stores in D_{anon}
$$\alpha_{\mathsf{C}}(m) = \{A^m_{\mathsf{C}},\ hi_{ot},\ M^m_{ot},\ m,\ \mathsf{P}^{\mathsf{C}}_\alpha\},$$
where for consistency, we use $M^m_{ot} \ = \ M_{ot}$ to express the balance of the A^m_{C} account.
6. CIB confirms account's validity with
$$\mathsf{Rec}_{\alpha_{\mathsf{C}}(m)} = \mathsf{Sig}_{\mathsf{CIB}_{ot}}(H^r(\alpha_{\mathsf{C}}(m)),\ Cert_{\mathsf{P}^{\mathsf{C}}_\alpha},\ time),$$
which serves as a proof of ownership of $\alpha_{\mathsf{C}}(m)$ and of the corresponding pseudonym $\mathsf{P}^{\mathsf{C}}_\alpha$.
7. C using $pass^t_e$ encrypts $\alpha_{\mathsf{C}}(m)$ and $\mathsf{Rec}_{\alpha_{\mathsf{C}}(m)}$ and stores them in her ACC.

Transaction Payment. Let that the cardholder C wants to purchase a product of value V_p from a merchant M. We assume that C has logged in to M's website anonymously.

[2] CIBs may be identified by the form of the ecash deposited.

1. C provides M's website — or the gateway G behind it — with the current number of her anonymous account (A_{C}^{x}), while she uses A_{C}^{x-1} to calculate the following:
$$\mathsf{TInfo}_{\mathsf{CIB}} = \{\{\mathsf{Cred}_{\mathsf{M}},\ T_{det}, A_{\mathsf{C}}^{x-1}\}_{K_\alpha},\ A_{\mathsf{C}}^{x}\}_{pk_{\mathsf{CIB}}^{e}},$$
where $K_\alpha = H_K(A_{\mathsf{C}}^{x})$ a key derived from A_{C}^{x} and $H_K = \underline{\mathsf{Hash}}_{ot}[h_k]$ used for efficiency purposes.
2. G sends $\mathsf{TInfo}_{\mathsf{CIB}}$ to C's CIB(CIB) as follows:
$$\mathsf{Sig}_{\mathsf{G}}(H_{GB}(T_{det}, \mathsf{Cred}_{\mathsf{M}}), time)$$
.
3. CIB decrypts $\mathsf{TInfo}_{\mathsf{CIB}}$ and checks whether it matches the information provided by G. CIB verifies that A_{C}^{x} is an active anonymous account number in its D_{anon} and that the current account balance is enough for the payment included in T_{det}. If there is an error, CIB sends G a transaction rejection message:
$$\mathsf{Rej}_{\mathsf{CIB}} = \mathsf{Sig}_{\mathsf{CIB}_{ot}}(\mathsf{Cred}_{\mathsf{M}},\ T_{det},\ A_{\mathsf{C}}^{x},\ reject),$$
which is then forwarded to C. To avoid any replay attacks, CIB updates the $\alpha_{\mathsf{C}}(x)$ entry of D_{anon} to
$$\alpha_{\mathsf{C}}(x-1) = \{A_{\mathsf{C}}^{x},\ h_{ot},\ h_k,\ M_{ot}^{x-1},\ x-1,\ Cert_{P_{\alpha}^{\mathsf{C}}}\},$$
where $M_{ot}^{x-1} = M_{ot}^{x}$ since no purchase took place.
4. CIB sends to G a signed endorsement on $\mathsf{Cred}_{\mathsf{M}}$, T_{det} with check of value V_p addressed to M's AB, AB:
$$\mathsf{Paym}_{\mathsf{CIB}\to\mathsf{M}} = \mathsf{Sig}_{\mathsf{CIB}_{ot}}(H_{MT}(\mathsf{Cred}_{\mathsf{M}},\ T_{det}),\ A_{\mathsf{C}}^{x},\ \mathsf{AB},\ time),\ Cred_M,\ T_{det}$$
where H_{MT} is a hash agreed between Banks, used to reduce the signed text. CIB also updates ($\alpha_{\mathsf{C}}(x)$ entry in its D_{anon} to
$$\alpha_{\mathsf{C}}(x-1) = \{(A_{\mathsf{C}}^{x-1},\ h_{ot},\ h_k,\ M_{ot}^{x-1},\ x-1\}$$
where $M_{ot}^{x-1} = M_{ot}^{x} - V_p$.
5. G sends to CIB a confirmation of the valid transaction
$$\mathsf{Rec}_{\mathsf{T}} = \mathsf{Sig}_{\mathsf{G}}^{H_r}(\mathsf{Cred}_{\mathsf{M}}, T_{det} - fin)$$
and forwards $\mathsf{Paym}_{\mathsf{CIB}\to\mathsf{M}}$ message to M's AB for M's account to be credited accordingly. As G is paid in proportion to V_p, it has no motivation not to submit $\mathsf{Paym}_{\mathsf{CIB}\to\mathsf{M}}$ to AB.
6. CIB stores $\mathsf{Rec}_{\mathsf{T}}$, T_{det} at its D_{hist} database.

We need to note that here A_{C}^{x-1} is used for authentication purposes, since C who generated R_{ot} is the only one, who knows the pre-image of A_{C}^{x} on H_{ot}. M sends a receipt of the purchase to one of C's email addresses as well as to CIB, which stores everything under the anonymous account entry. When closing the entry in D_{anon}, C, through her ACC, demonstrates knowledge of $\mathsf{P}_{\alpha}^{\mathsf{C}}$, m and R_{ot} to CIB and the latter updates the content of C's ACC accordingly (with $\mathsf{Rec}_{\mathsf{T}}$, T_{det} and any additional wallets).

ACC BackUp. It takes place between a cardholder C, who participates in person, i.e. after having identified herself to her CIB, and her CIB, CIB.

1. Cgenerates a random number N_b, creates the $K = H_K(N_b,\ pass_{pin})$ and sends
$$BackUp = \{N_b\}_{pk_{\mathsf{C}}^{e}}, \{ACCcontent||date - time\}_K$$
to CIB. *K is used for efficiency purposes, since symmetric encryption is considered to be much faster. We can see that only the valid owner of the card knows how to create K given N_b.*
2. Both, C and CIB, hash and sign the $BackUp$ into
$$BackUp_x = \mathsf{Sig}_x^{H_{CB}}(BackUp),\ x = \{\mathsf{C}, \mathsf{B}\},$$
where $ACContent$ is the content of the anonymous credit card and $date - time$ is the timestamp of the backup.
3. CIB updates her D_{hist}.

Loss Recovery. It takes place between a cardholder C, who participates as an identity and her CIB, CIB.

1. C declares the loss of her ACC and is provided with the most recent BackUp of her ACC, *BackUp*.
2. C verifies that *BackUp* matches the most recent $BackUp_B$ of her and decrypts it.
3. C and CIB collaborate in $\mathsf{EC.Spend_p}(sk^p_{\mathsf{C}}, |W'_p|)$ to spend the remaining payment wallet of the *BackUp* (W'_p).
4. CIB credits C's credit account for the amount spent till the *BackUp* had been taken ($\mathsf{L_{credit}}$ - $|W'_p|$) and waits till the merchants' deposit time passes. For each double-spent digital coin detected, CIB runs $\mathsf{EC.Identify_p(S)}$ to confirm its owner and credits C's credit account accordingly. In this way, we avoid overall exspose of C if the last *BackUp* kept is not up-to-date with the C's most recent transactions.
5. CIB infers C's overall spending amount and collaborates with her in $\mathsf{EC.Withdraw_p}$ and $\mathsf{EC.Withdraw_{id}}$ procedures for the latter to issue new payment and identity wallets W_p and W_{id}.

As we can see, in this case there is a small breach in anonymity provided in our system: Bank will be able to see with who customer interacted to spend the double-spent part of her payment wallet. In the following section, we will elaborate on this privacy breach.

Monthly Payment Calculation . At the end of every month, each cardholder C proves to her CIB, CIB, the amount of money she has spent throughout that month. To calculate C's monthly payment, CIB applies the formula used in current Credit Card Systems on C's overall credits.

1. C enters her $pass^w_e$ to decrypt the remaining of her W_{id} wallet (W'_{id}) and interacts with CIB into $\mathsf{EC.Spend_{id}}(sk^s_{\mathsf{C}}, |W'_{id}|)$ to spend it entirely.
2. CIB updates C's entry in D_{credit} with the amount of money spent by C: $\mathsf{L_{credit}}$ - $|W'_{id}|$.
3. CIB bills C based on the latter's D_{credit} entry.
4. If C is still eligible for an ACC, she interacts with CIB for issuing a new W_{id} and additional W_p according to C's new credit limit.

It is obvious that any attempt on C's part to lie for the remaining W_{id} wallet, i.e. by presenting a former version of the card, a part of W_{id} would be double-spent (to CIB and to a merchant) and EC.Trace$_{id}$ would reveal sk^s_{C}.

Expense Report-Error Correction.

Expense Report. The cardholder C enters her $pass^t_e$ to decrypt the transaction related part of her ACC to obtain the detailed chain of transactions. C may request for an expense report of her active anonymous accounts by providing her CIB, CIB, anonymously with the following:

$$\{expense-report, time, A^{x-1}_{\mathsf{C}}\}_{K_\alpha},\ A^x_{\mathsf{C}},$$

where A^x_{C} is the current anonymous account number and A^{x-1}_{C} is used for authentication purposes. CIB provides the report of A^x_{C}'s transaction activity and — for security purposes — updates D_{anon} by replacing $\alpha_{\mathsf{C}}(x)$ with $\alpha_{\mathsf{C}}(x-1)$.

Error Correction. It takes place between a cardholder C, who participates as an identity, and the customer service section of ACCA or of the merchant M involved. There are two error cases we examine here: (a) C detects a mischarge at her expense report, in which case she contacts the ACCA in person, and (b) C requests to cancel a transaction with a merchant M, in which case she contacts M in person. In both cases C uses $\mathsf{Rec_T}$ as a proof of purchase.

For now we will refer to case (a). C contacts the ACCA and the ACCA contacts M. M can either accept or reject the refund-request. If he rejects, his customer service department, ($\mathsf{M\text{-}CS}$), sends to the ACCA:

$$\mathsf{Rej_M} = \mathsf{Sig_{M-CS}}(Merchant,\ T_{det},\ refund-reject,\ time).$$

ACCA, depending on its policy, may provide C with the refund:

$$\mathsf{RefCoup}_{\mathsf{ACCA}\to\mathsf{C}} = \mathsf{Sig}_{\mathsf{ACCA-ref}}(\mathsf{C},\ \mathsf{Rej}_{\mathsf{M}},\ T_{det},\ price,\ time),$$

where $\mathsf{ACCA\text{-}ref}$ is the refund section of ACCA. C interracts with her CIB, CIB, to deposit $\mathsf{RefCoup}_{\mathsf{ACCA}\to\mathsf{C}}$ and to withdraw additional payment and identity wallets of equivalent value. *It is noticeable that in this case, we do not care whether* C*'s identity is revealed, since no purchase was actually done.* If M accepts the return/mischarge:

1. M provides the ACCA with signed endorsements of the amount to be removed from his account and added to C's card/account. Signatures of this type may be special type of refund signatures issued by M's CS:
$$\mathsf{Ref}_{\mathsf{M}} = \mathsf{Sig}_{\mathsf{M-CS}}(Merchant,\ T_{det},\ refund-accept,\ refund,\ time),$$
where $Merchant$ has all the merchant related account and certificate information.
2. The ACCA sends $\mathsf{Ref}_{\mathsf{M}}$ to M's AB, AB, for validation.
3. AB sends back to ACCA a confirmation:
$$\mathsf{RefConf}_{\mathsf{AB}} = \mathsf{Sig}_{\mathsf{AB}}^{H_{\mathsf{AB}-ref}}(Ref_M - accept),$$
where $H_{\mathsf{AB}-ref}$ is a refund-specific hash of AB.
4. For C to collect the payment, ACCA issues a digital check to C:
$$\mathsf{RefCoup}_{\mathsf{ACCA}\to\mathsf{C}} = \mathsf{Sig}_{\mathsf{ACCA-ref}}(\mathsf{RefConf}_{\mathsf{AB}},\ \mathsf{Ref}_{\mathsf{M}},\ T_{det},\ refund,\ time, \mathsf{C}).$$
5. C then deposits $\mathsf{RefCoup}_{\mathsf{ACCA}\to\mathsf{C}}$ to CIB, which contacts AB to make the appropriate transfers. C interacts with CIB to issue $W_{id} - W_p$ wallets of $refund$ value.

In case (b), as mentioned before, C contacts M directly. If M accepts the return of the product, it provides C with $\mathsf{Ref}_{\mathsf{M}}$, which C deposits to CIB. CIB makes the appropriate interbank communications to verify Ref_M's validity and updates C's account accordingly.

Timestaps are used for replay attacks to be avoided. It is noticeable that all messages exchanged here are signed/timestamped. In this way replay and intersection attacks are avoided. Also, we make use of different digital signature key-pairs than in the rest of the ACC protocols. This is done to avoid any type of reflection attacks involving different protocols which use the same signature key-pairs. Another important point in the expense report protocol is that privacy is not a concern and thus, no encryption is used.

ACC Promotion Offers. This is the case where the Anonymous Credit Card Association (ACCA), some CIBs and merchants have made an agreement, so that the CIBs provide better payment interest rates to cardholders when they make many purchases from the participating merchants, i.e. the cardholder may be eligible of paying out an amount V_p in N_{M} parts without any interests applied on it. To support this mechanism in our system we introduce coupons implemented with the blind group signature scheme.

The Anonymous Credit Card Association (ACCA), as the ACCA manager, makes the appropriate setup for the blind group signature scheme [LR98], which will be used to instantiate the group of the promotion participating merchants. In particular ACCA runs $\mathsf{BGS.Setup}$ mentioned in subsection 4.1 to generate $pk_{\mathsf{M}}^{bg}, sk_{\mathsf{M}}^{bg}$, i.e. the public and secret administration information of the merchants-group. Merchants, who participate in the promotion offers, interact with the ACCA in $\mathsf{BGS.Join}$ to obtain a membership blind group signature secret information sk_{M}^{bg}. In addition, participating CIBs issue a plain blind signature key pair [O06,CL02]: $(pk_{\mathsf{B}}^{b}, sk_{\mathsf{B}}^{b})$.

Assuming coupons of value v_p^M and after having interacted with a merchant M in a transaction of value V_p, a cardholder C contributes randomness $r_1, \ldots, r_N$ to obtain from M $N = \lfloor \frac{V_p}{v_p^M} \rfloor$, blind signatures, the *credit coupons*:

$$\mathsf{GBSig}_M(r_1), \ldots, \mathsf{GBSig}_M(r_N).$$

Credit coupons are separately encrypted and stored in the ACC with $pass_c$ and should be deposited within a month after the transaction has taken place. This may be enforced by changing group's administration information every month.

When she decides to deposit her coupons, C contacts her CIB through her ACC. If participating in the ACC offers, C's CIB, CIB checks coupons' validity by contacting the ACCA. The ACCA runs $\mathsf{BGS.Open}$ procedure to recover merchant's name, updates M-related statistics and checks M agreement details ($N_{merchant}$). CIB, using the technique in A.1 calculates the overall amount of money the C has to be favored in general, *CreditReduction*, and issues a number of *debit coupons* of equivalent value. Debit Coupons are plain CIB blind signatures on quantities chosen by C $(r'_1, \ldots, r'_N)$ and are – thus – unlinkable to particular credit coupons:

$$\mathsf{BSig}_{\mathsf{CIB}}(r'_1), \ldots, \mathsf{BSig}_{\mathsf{CIB}}(r'_N).$$

C deposits the debit coupons at her convenience to CIB to reduce her credit amount by *CreditReduction*.

5 System Considerations

In this section, we will emphasize on particular system issues. For detailed Security Definitions and the corresponding proofs, see Appendix B. Here, we will only sketch how the most important of our system's requirements are satisfied.

Cardholder Anonymity w.r.t Bank and the merchant/ Cardholder non-Traceability. Both of these two properties are satisfied through ecash anonymity and unlinkability properties mentioned in 4.1. Each payment procedure is a typical ecash $\mathsf{EC.Spend}$ procedure from W_p and W_{id} wallets and can thus not be linked to the cardholder C who issued the ACC or to any other spending (transaction) from the same wallets. On the other hand, the anonymity provided is conditional: if C tries to spend more money than his credit-limit, i.e. more ecoins from the initial amount in the two wallets, or lie at the monthly payment calculation procedure — by providing a non-updated version of her ACC —, a part of W_{id} will inevitably be double-spent: to her CIB (CIB) at the $\mathsf{CreditUpdate}$ procedure and to a merchant M in a $\mathsf{Payment}$ protocol. After M's deposit procedure, CIB detects the double-spending and runs $\mathsf{EC.Identify}_{\mathsf{id}}$ on the double-spent identity ecoins to reveal each double-spender's pk_{C}^{id}. In addition, CIB may run $\mathsf{EC.Trace}_{\mathsf{id}}$ on the same ecoins to trace all the identity ecoins withdrawn by each double-spender.

In the case of ACC promotion offers, the two aforementioned properties are satisfied through the blindness property of blind (group) signatures. When deposited, *credit coupons* are linked to the merchant M who issued them since C's CIB, CIB, contacts the ACCA which is the manager of the group and can identify the coupons' creator (signer). However, because of the blindness property, the ACCA or/and CIB(if colluding with the ACCA) does not know the exact transaction the cardholder participated in, even when colluding with M. In any case, *Credit coupons* are deposited anonymously and the *debit coupons*, which are issued in response to the valid *credit* ones — and deposited by C in person — are blind to the CIB who issued them and thus unlinkable to any particular *credit coupon*.

There are two cases, where we accept a small breach in a cardholder's anonymity/transaction unlinkability: (a) in the *Loss Recovery* and (b) in the *Online Payment* scenarios. At Loss Recovery process, when the most recent *BackUp* is not up-to-date, C inevitably doublespends a part of her W_p wallet: to the merchants she interacted with and to her CIB. pk_{C}^{s} is then revealed and Bank knows who C interacted with. However, this anonymity breach becomes less important if we require that backups are taken regularly. In the *Online Payment* case, Bank can obviously trace what type of transactions a particular anonymous account is involved in through D_{hist}. However, thanks to the unlinkability property of the ecash spent at the anonymous account setup phase, linking that profile to a particular identity is impossible. In any case, the cardholder may open as many anonymous accounts she wishes, in order to avoid transaction linkability.

Authorized Anonymity Revocation. This is the case where a cardholder, C, is a suspect of a offence and Judge requests a detailed description of that C's ACC related transactions. In our system, this can be achieved only with C's consent and in a way such that the later cannot lie for her transactions:

a. Cis asked to provide sk^s_{C} for all her transactions to be revealed, which we want to avoid.
b. Cis asked to enter her $pass^t_e$ to decrypt the transaction related part of her ACC and "spend" the rest of her W_p to CIB. Transaction details of each transaction are signed by a merchant or C's CIB (CIB) — in the case of Anonymous Account Setup — and are, thus, impossible to be forged. To check for any deceptive deletion of a transaction on cardholder's side, CIB may use D_{hist} to check whether the overall amount spent matches the aggregated amount in the backed-up transaction details.

On Security of Online Transactions. We can study it in two phases: (a) the security of the anonymous account setup and (b) the security of the management of the anonymous account, which includes the transaction payments and the expense reports issued.

In case (a) cardholder Cauthentication is achieved through the $pass_{pin}$,Cis required to enter to create the $\alpha_{\mathsf{C}}(m)$ account. As in the offline payment procedure, only the owner of the ACC may spend ecoins from the wallets in it. Double-spending tracing mechanism restricts Cfrom using the spent part of her wallets elsewhere.

Security in case (b) is achieved through the non-invertibility property of hash functions and the unforgeability of the digital signature schemes. More specifically, it is not possible for an unauthorized party to use the balance of an account even if she knows the current anonymous account number (A^x_{C}); knowledge of H_{ot} and A^x_{C}'s pre-image w.r.t. H_{ot} is required. Signed endorsement $Rec_{\alpha_{\mathsf{C}}(m)}$ of CIB on the initial account's value, prevent CIB from cheating. In addition, as T_{det} and $credm$ are part of $\mathsf{Info}_{\mathsf{CIB}}$, i.e. encrypted with a key only CIB and Cmay derive, Gcannot lie for the *price* or the merchant M the payment is for. As timestamps are included in every message and account numbers change in every authorized request, replay attacks or offline account guessing attacks cannot succeed. Expense Report authorization is achieved in an exact similar way.

Error Correction Issues. As mentioned before, in order to request a refund for an charging error having occurred in the expense report, the carholder C is required to present $\mathsf{Rec}_{\mathsf{T}}$ to the ACCA in person. In offline transactions, the latter does not constitute a problem since $\mathsf{Rec}_{\mathsf{T}}$ is completely unlinkable to other transactions. However, if $\mathsf{Rec}_{\mathsf{T}}$ refers to an online transaction, then — since CIB has all the information regarding the (online) transaction activity of the anonymous account — C is automatically linked to all the transactions of that particular anonymous account. We address this problem in two ways:

- We reduce the amount of transaction information linked to each account by encouraging cardholders to open several — as opposed to one — anonymous accounts for their online purchases. This security measure becomes even more attractive if we consider the fact that *Anonymous Account Setup* is similar in terms of computation to an offline payment procedure,.
- We reduce the likelihood of an error taking place by introducing additional security measures at the online transaction payment procedure. Apart from the current C authentication procedure in *Transaction Payment*, we require that the CIB obtains an email-based transaction confirmation by the owner of the anonymous account. Therefore, we make the following changes:
  - At the *Anonymous Account Setup*. In addition to $\mathsf{P}^{\mathsf{C}}_{\alpha}$, C, provides the CIB with an email address, $\mathsf{email}_{P^{\mathsf{C}}_{\alpha}}$, which is added to $\alpha_{\mathsf{C}}(m)$. It is critical that $\mathsf{email}_{P^{\mathsf{C}}_{\alpha}}$ contains no C-identification information.
  - At the *Transaction Payment*. At step 1, a nonce *nonce* is added to the $\mathsf{TInfo}_{\mathsf{CIB}}$ send by C to CIB:

$$\mathsf{TInfo}_{\mathsf{CIB}} \;=\; \{\{\mathsf{Cred}_{\mathsf{M}},\ T_{det},\ nonce\}_{A_C^{x-1}},\ A_C^x\}_{pk^e_{CIB}},$$

  while after step 4 a confirmation email is sent to $\mathsf{email}_{P^{\mathsf{C}}_{\alpha}}$address with T_{det} and a function F of *nonce*, to which C is required to respond for her purchase to be completed. If no confirmation is received within a particular time interval a transaction rejection procedure on behalf of CIB takes place.

However, we do need to emphasize on the fact that involving electronic mail procedures in the online purchase procedure, is likely to introduce other privacy related concenrs: Because of source tracing information they may include, account owners' confirmations may enable Banks/Merchants to link individual transactions from different accounts as having been done by the same person.

ACC Unforgeability is satisfied through the Correctness and Unforgeability properties of the underlying ecash schemes. **ACC non-Transferability** is also satisfied, sicne sk_{C}^{s} is required for the card to be used in both offline and online purchases.

Bank Dishonesty. $BackUp_B$ is used to avoid any attempt of a CIB to trick a cardholder into tracing more of the latter's transactions: Assuming the CIB provided a less recent backup, then a bigger part of W_p would be double-spent and more merchants would be directly linked to the cardholder. $BackUp_B$ will act as an undeniable proof of the date and integrity of the backup kept.

Merchant's Honesty. Coupons's deposit procedure enables a merchant-cardholder to use the coupons he can issue to his customers for his own favor. We address this problem by enforcing ACCA to grant a particular number of coupons to each merchant. Coupons' number is proportionate to merchant's sales and, if restricted, merchant will be motivated to use it to attract customers as opposed to use them for his own purposes.

ACC Organization. ACCs' content is organized in the following way: $\{E_{T^1}, \ldots, E_{T^\ell}, padding, E_{W'_{p,id}}\}$, where ℓ is the number of transactions a cardholder has participated in and *padding* is used to avoid any information leakage regarding ℓ. This modular way of encryption is necessary for each of the procedures mentioned before to be able to be executed individually.

Computing power. Credit card customers in our system lack in computing power: not all of them have or know how to install software able to encrypt/ decrypt text, verify hashes, which are used in our system. A solution on this problem would be that CIBs provide their customers with special machines dealing with card encryption/decryption issues. The extra cost of these devices, may be provided by the cardholder as an extra price for her privacy.

6 Conclusion

In this paper, we addressed e-commerce Context Privacy. In particular, we presented a deployable credit card system which guarantees cardholder anonymity and transaction unlinkability even towards to Credit Card Associations or Card Issuing Banks. In special circumstances the transactions of a party may be revealed but only with that party's consent. Undeniably, there are still issues to be dealt, such as password loss recovery, operations' transparency w.r.t. cardholders. However, we do believe that this paper is a good start for real time privacy in current credit card systems.

References

[CHL05] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321. Springer-Verlag, 2005.

[CL02] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *International Conference on Security in Communication Networks – SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer Verlag, 2002.

[DCN90] A. F. D. Chaum and M. Naor. Untraceable Electronic Cash. 1990.

[JLO97] A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures (extended abstract). In *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 150–164. Springer-Verlag, 1997.

[LPM94] S. H. Low, S. Paul, and N. F. Maxemchuk. Anonymous credit cards. In *CCS '94: Proceedings of the 2nd ACM Conference on Computer and communications security*, pages 108–117, New York, NY, USA, 1994. ACM.

[LR98] A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In *In Financial Cryptography (FC)*, pages 184–197. Springer-Verlag, 1998.

[O06] T. Okamoto. Efficient blind and partially blind signatures without random oracles. In *TCC*, pages 80–99, 2006.

A Appendix

A.1 Coupon Number Calculation.

Let the following notation:

- r the interest applied in the card remaining amount each month. We basically need to "undo" interest rate application for the coupon-amount.
- t_p^M: Threshold value for issueing coupon agreed with a merchant M.
- v_p^M: Value of each coupon issued by Merchant M.
- N_M: Number of times the amount of money spend by client can be payed out in without interest.
- A_C: Amount of money customer C spent in merchant M. It is obvious that for a coupon to be issued, it should be: $A_C \geq t_p^M$.

A customer who takes advantage of the offer, obtains $n_C = \lfloor \frac{A_C}{v_p^M} \rfloor$ coupons. As mentioned before, depending on the agreement merchant has made with the ACCA, outpayment of a coupon will be destributed in N_M months. Thus, if the customer payed the normal interest rate r for outpaying amount $A'_C = n_C \times v_p^M$, he would have to pay:

$$CustomerOvercharge =$$

$$r \times \{1 + \frac{N_M - 1}{N_M} + \frac{N_M - 2}{N_M} + \ldots + \frac{1}{N_M}\} \times A'_C =$$

$$r \times \{\frac{N_M \times (N_M - 1)}{2 \times N_M}\} \times A'_C =$$

$$r \times \frac{N_M - 1}{2} \times A'_C$$

The main concept is here that we make customer pay this additional amount per month but after having removed this amount from his credit card in advance. Namely, during the montly credit card amount update procedure, his credit amount gets "enhanced" [3] by the additional amount she will have to pay during the following months. However, there is a chaveat: the amount substracted from customer's credit account should be calculated in a fair way towards the Bank. In particular, assume that *CreditReduction* is the amount removed from customer's credit account, Bank will lose all the interest payments on it. Namely, Bank will lose at most

$$CreditReduction \times (1 + r)^{N_M} - CreditReduction.$$

Thus *CreditReduction* should satisfy the following:

$$CreditReduction = CustomerOvercharge - CreditReduction \times \{(1 + r)^{N_M} - 1\} \iff$$

$$CreditReduction \times \{(1 + r)^{N_M} - 1\} = CustomerOvercharge \iff$$

$$CreditReduction = \frac{CustomerOvercharge}{(1 + r)^{N_M} - 1} \iff$$

[3] By enhanced we mean in favor of the customer, namely the amount customer is credited is reduced.

$$CreditReduction = r \times A'_C \times \frac{1}{2} \frac{N_M - 1}{(1+r)^{N_M} - 1}$$

Given the fact that r ranges from 0.01 to 0.05, and N_M from 3 to 20, we can see how $CreditReduction$ is about 30-40% of the initial amount paid. The reason we decided that this amount is added to customer's credit account as opposed to her savings one, is the way Banks operate, depositing actual money to customers savings accounts would require from them cash that may not be willing to pay in advance.

B Security

Security is a principal requirement in our system. In this section, we provide strict definitions of our system's operations and we base on them to define our system's security properties. Some of the definitions presented in this section are inspired by previous work on other primitives, such as [**?**,CHL05,**?**].

In what follows, we assume the typical Cardholder-Merchant-Bank system architecture.

B.1 Operations

When an operation is an interactive procedure (or a protocol consisting of multiple procedures) between two entities C and B, we denote it by $\langle O_{\mathsf{C}}, O_B\rangle \leftarrow \mathsf{Pro}(I_{\mathsf{C},\ B})[\mathsf{C}(I_{\mathsf{C}}),\ B(I_B)]$, where Pro is the name of the procedure (or protocol).

O_{C} (resp. O_B) is the private output of C (resp. B), $I_{\mathsf{C},\ B}$ is the common input of both entities, and I_{C} (resp. I_B) is the private input of C (resp. B). We also note that depending on the setup, some operations may require additional global parameters (i.e. some common parameters for efficient zero-knowledge proofs, a modulus p, etc). Our system will need these additional parameters only when using underlying schemes that use such parameters, e.g., ecash systems or blind group signatures. To simplify notation, we omit these potential global parameters from the inputs to all the operations.

- $(pk_{\mathsf{B}}, sk_{\mathsf{B}}) \leftarrow \mathsf{Bkeygen}(1^k)$ is the key generation algorithm for Bank B. We denote by pk_{B} all public information regarding B, namely her public ([group]blind) signature and encryption keys and by sk_{B} the overall secret information of B.
- $(pk_{\mathsf{C}}, sk_{\mathsf{C}}) \leftarrow \mathsf{Ckeygen}(1^k)$ is the key generation algorithm for cardholders. We denote by pk_{C} all public information regarding a cardholder C, namely her public signature and encryption keys and by sk_{C} the overall secret information of C.
- $\langle(W_p, W_{id}), (T_{p,id}, D_{\mathsf{credit}}{}'))/\langle\perp,\perp\rangle \leftarrow$ $\mathsf{AnonymousCreditCardIssue}$ $(pk_{\mathsf{B}}, pk_{\mathsf{C}}, \mathsf{L}_{\mathsf{credit}})$ $[\mathsf{C}(sk_{\mathsf{C}}),\ \mathsf{B}(sk_{\mathsf{B}},\ D_{\mathsf{credit}},\ D_{\mathsf{debit}})]$. A customer Cissues an ACC of credit limit $\mathsf{L}_{\mathsf{credit}}$. value ecoins in the form of two wallets W_p and W_{id} from Bank B. Bank, using D_{debit}, checks if C is eligible for that. If so, ecash withdrawal procedure procedure is carried out for C to acquire two $\mathsf{L}_{\mathsf{credit}}$ valued wallets W_p and W_{id}. B's $D_{\mathsf{credit}}$is updated accordingly while B maintains $T_{p,id}$ to trace double-spenders in case of emergency.
- $\langle(W'_p,\ W'_{id}), (S_p,\ \pi_p,\ S_{id},\ \pi_{id})\rangle/\langle\perp,\perp\rangle \leftarrow \mathsf{OfflinePayment}(pk_{\mathsf{M}},\ pk_{\mathsf{B}},\ T_{det})$ $[C(W_p, W_{id}, sk_{\mathsf{C}}),\ \mathsf{M}(sk_{\mathsf{M}})]$. Customer C, using her ACC spends V_p value from each wallet to M. Spent ecoins are pairs of (S,π), where S is a serial number and π is the proof of its validity. For convenience, we will denote with $(S_p,\ \pi_p)$ and $(S_{id},\ \pi_{id})$ the set of serial numbers and spending-proofs of ecoins spent from W_p and W_{id} wallets, respectively.
- $\langle\top, (D_{\mathsf{debit}}{}', D_{\mathsf{hist}}{}')\rangle/\langle\perp,\perp\rangle \leftarrow$ $\mathsf{MerchantPayment}$ $(pk_{\mathsf{M}}, pk_{\mathsf{B}})$ $[\mathsf{M}(sk_{\mathsf{M}},\ S_{p,id},\ \pi_{p,id}),\ \mathsf{B}(sk_{\mathsf{B}},\ D_{\mathsf{debit}},\ D_{\mathsf{hist}})]$, where for convenience
$$(S_{p,id},\ \pi_{p,id}) = (S_p,\ \pi_p) \cup (S_{id},\ \pi_{id}).$$
A merchant M deposits the ecoins he has received throughout his selling activity. If the ecoins deposited $(S_{p,id},\ \pi_{p,id})$ are valid and not double-spent, Bank B updates the entry of M in its debit database D_{debit}. Deposited ecoins are then stored in the history database D_{hist}.
- $\langle(W'_{p,id},\ \alpha_{\mathsf{C}}(m)), (S_{p,id},\ \pi_{p,id},\ D_{\mathsf{anon}}{}')\rangle/\langle\perp,\perp\rangle \leftarrow \mathsf{AnonymousAccountSetup}(pk_{\mathsf{B}},\ M_{ot},\ m,\ R_{ot})[\mathsf{C}(W_{p,id},\ sk_{\mathsf{C}}), \mathsf{B}(sk_{\mathsf{B}},\ D_{\mathsf{anon}})]$. In this procedure, C interacts through her ACC with Bank B to setup an anonymous account for online transactions. Both parties know B's public information (pk_{B}), the new account's balance M_{ot}, m and R_{ot}. The main output is the new D_{anon} entry for both parties.

- $\langle \top, \mathsf{Paym}_{\mathsf{CIB}},\ (\ \mathsf{D}'_{\mathsf{anon}},\ \mathsf{Rec}_{\mathsf{T}}) \rangle / \langle \mathsf{Rej}_{\mathsf{CIB}},\ \mathsf{Rej}_{\mathsf{CIB}},\ D_{\mathsf{anon}}{}' \rangle \leftarrow$
  $\leftarrow\ \mathsf{OnlinePayment}(pk_{\mathsf{G}},\ pk_{\mathsf{CIB}},\ V_p)[C(\alpha_{\mathsf{C}}(x)),\ \mathsf{G}(sk_{\mathsf{G}}),\ \mathsf{CIB}(\mathsf{D}_{\mathsf{anon}}, sk_{\mathsf{CIB}})]$. In this procedure, C uses her anonymous account information to pay for the purchase of a product of value V_p. The CIB-related outputs are the updated D_{anon} and the transaction receipt, $\mathsf{Rec}_{\mathsf{T}}$. Gateway G receives the payment $\mathsf{Paym}_{\mathsf{CIB}}$, while if something goes wrong only CIB's D_{anon}changes.
- $\langle \top, \top,\ D_{\mathsf{debit}}{}' \rangle / \langle \bot, \bot,\ \bot \rangle \leftarrow\ \mathsf{OnlinePaymentDeposit}(pk_{\mathsf{G}},\ pk_{\mathsf{CIB}},\ pk_{\mathsf{AB}})$
  $[\mathsf{G}(sk_{\mathsf{G}},\ \mathsf{Paym}_{\mathsf{CIB}}),\ \mathsf{CIB}(sk_{\mathsf{CIB}}),\ \mathsf{AB}(D_{\mathsf{debit}},\ sk_{\mathsf{AB}})]$. Gdeposits $\mathsf{Paym}_{\mathsf{CIB}}$ to M's AB, AB. AB, G and C's CIB, CIB, collaborate for M's debit account to be updated (D_{debit}').
- $\mathsf{Identify}$ operation consists of two suboperations:
  - $(pk_{\mathsf{C}}, \mathit{\Pi}_p^G)/\bot \leftarrow \mathsf{Identify}_{\mathsf{p}}(S_p, \pi_p^1, \pi_p^2)$ and
  - $(pk_{\mathsf{C}}, \mathit{\Pi}_{id}^G)/\bot \leftarrow \mathsf{Identify}_{\mathsf{id}}(S_{id}, \pi_{id}^1, \pi_{id}^2)$.

  If an ecoin from W_p (or W_{id}) is double-spent, with (S^p, π_p^1) and (S_p, π_p^2) (or (S_p, π_p^1) and (S_p, π_p^2)), Bank can find the customer who double-spent the ecoin with serial S_p (S_{id}) using $\mathsf{Identify}_{\mathsf{p}}$ (or $\mathsf{Identify}_{\mathsf{id}}$). $\mathit{\Pi}_p^G$ $(\mathit{\Pi}_{id}^G)$ is a proof that pk_{C} double-spent the ecoin with the serial number S_p (S_{id}).
- $\mathsf{VerifyGuilt}$ operation also consists of two suboperations, one for each wallet in the anonymous card:
  - $\top/\bot \leftarrow \mathsf{VerifyGuilt}_{\mathsf{p}}(S_p,\ \mathit{\Pi}_p^G,\ pk_{\mathsf{C}})$ and
  - $\top/\bot \leftarrow \mathsf{VerifyGuilt}_{\mathsf{id}}(S_{id},\ \mathit{\Pi}_{id}^G,\ pk_{\mathsf{C}})$.

  Both operations output $\top$ if the customer C (represented by pk_{C}) double-spent S_p from W_p and the corresponding ecoin S^{id} from wallet W_{id}. $\mathit{\Pi}_p^G$ and $\mathit{\Pi}_{id}^G$ are the proofs of guilt of C, i.e. the outputs from the preceding $\mathsf{Identify}$ procedure.
- $\langle S_{id}^i,\ \mathit{\Pi}_{id}^i \rangle \leftarrow \mathsf{Trace}_{\mathsf{id}}(\mathsf{S}_{\mathsf{id}},\ pk_{\mathsf{C}},\ \mathsf{\Pi}_{\mathsf{id}}^{\mathsf{G}},\ D_{\mathsf{hist}},\ \mathsf{L})$, where $i = 1 \dots L$. This algorithm first checks whether $\mathsf{VerifiyGuilt}_{\mathsf{id}}(S_{id},\ \mathit{\Pi}_{id}^G,\ pk_{\mathsf{C}})$ accepts. If yes, the procedure outputs all L ecoins $(S_{id}^i, i = 1 \dots L)$ issued by the customer pk_{C} along with the proof $\mathit{\Pi}_{id}^i$ of pk_{C}'s ownership. In any other case, this algorithm does nothing.
- $\top/\bot \leftarrow \mathsf{VerifyOwnership}_{\mathsf{id}}(S_{id},\ \mathit{\Pi}_{id},\ pk_{\mathsf{C}},\ L)$. This algorithm allows to publicly verify the proof $\mathit{\Pi}_{id}$ that an ecoin with serial number S_{id} belongs to a doublespender with public key pk_{C}. We need to emphasize on the fact that the latter two procedures can only be applied to ecoins spent from W_{id} type of wallets.
- $\langle (BackUp_{\mathsf{B}}, (D_{\mathsf{hist}}{}',\ BackUp_{\mathsf{C}} \rangle / \langle \bot, \bot \rangle \leftarrow\ \mathsf{BackUp}\ (pk_{\mathsf{C}},\ pk_{\mathsf{B}},\ date)[\mathsf{C}(sk_{\mathsf{C}},\ ACContent), \mathsf{B}(sk_{\mathsf{B}}^s,\ D_{\mathsf{hist}})]$. Customers follow this procedure in order to back their ACC up. Cardholders provide the encrypted content of their card $ACContent$ as well as a backup encryption password $pass_e$, while both parties output obtain a signed receipt of the final form of the backup stored $BackUp_{\mathsf{B}}$, $BackUp_{\mathsf{C}}.D_{\mathsf{hist}}$ is also updated with the new backup-record.
- $\langle ACC', (\mathsf{D}'_{\mathsf{credit}},\ \mathsf{D}'_{\mathsf{hist}},\ T_{p,id}) \rangle /\ \langle \bot, \bot \rangle \leftarrow\ \mathsf{CardLossRecovery}(pk_{\mathsf{C}},\ pk_{\mathsf{B}})[\mathsf{C}(sk_{\mathsf{C}}), \mathsf{B}(sk_{\mathsf{B}}^s,\ BackUp)]$. C checks whether $BackUp$ provided by her CIB B, is valid and collaborates with B to generate a new ACC. B provides proof of $BackUp$ validity. If the BackUp is not up-to-date, B checks $D_{\mathsf{hist}}$for double-spending case.
- $\langle (W'_{p,id}, monthly\_payment), (S_{id},\ \pi_{id},\ T'_{p,id},\ \mathsf{D}_{\mathsf{hist}}{}',\ \mathsf{D}_{\mathsf{credit}}{}') \rangle / \langle \bot, \bot \rangle\ \leftarrow\ \mathsf{CreditUpdate}(pk_{\mathsf{B}},\ pk_{\mathsf{C}})[\mathsf{C}(sk_{\mathsf{C}}),\ \mathsf{B}(sk_{\mathsf{B}},$ $D_{\mathsf{hist}},\ D_{\mathsf{credit}})]$. $\mathsf{CreditUpdate}$ operation takes place between customer and Bank in order for customer to update her credit account according to her purchases. Customer provides her secret information (passwords, sk_{C}^s and wallets) and her outputs involve new wallets and her monthly bill. Bank checks D_{hist} for double-spending and if everything is fine, it updates its D_{credit}. Bank also obtains double-spending related tracing information for the new wallets withdrawn $(T'_{p,id})$.

- $\mathsf{ErrorCorrection}$ operation has two versions.
  - *Version 1:* This is the case, where an error has been detected in the expense report and we have two steps:
    1. $\langle \mathsf{RefCoup}_{\mathsf{ACCA} \rightarrow \mathsf{C}},\ \top, (\mathsf{Ref}_{\mathsf{M}},\ \mathsf{RefConf}_{\mathsf{AB}}), \mathsf{Ref}_{\mathsf{M}} \rangle / \langle \mathsf{RefCoup}_{\mathsf{ACCA} \rightarrow \mathsf{C}}, \top, \mathsf{Rej}_{\mathsf{M}}, - \rangle\ \leftarrow$
       $\leftarrow\ \ \mathsf{RefundConf}(pk_{\mathsf{M}},\ pk_{\mathsf{CIB}},\ pk_{\mathsf{AB}})[\mathsf{C}(\mathsf{Rec}_{\mathsf{T}}),\ \mathsf{M}(sk_{\mathsf{M}}),\ ACCA(sk_{ACCA}),\ \mathsf{AB}(sk_{\mathsf{AB}},\ \mathsf{D}_{\mathsf{debit}})]$, where the C declares a charge error, i.e. requests a refund for a purchase she did not participate in. C obtains $\mathsf{RefCoup}_{\mathsf{ACCA} \rightarrow \mathsf{C}}$, which is basically a payment check either from merchant's AB, AB, or from the ACCA if merchant M rejects $(\mathsf{Rej}_{\mathsf{M}})$ the refund. ACCA obtains (a) a proof of whether merchant accepted to provide the refund $(\mathsf{Ref}_{\mathsf{M}})$ or not $(\mathsf{Rej}_{\mathsf{M}})$, (b) a proof from AB of M's account balance, ($\mathsf{RefundConf}$). AB simply updates $D_{\mathsf{hist}}$with $\mathsf{Ref}_{\mathsf{M}}$.
    2. $\langle W'_{p,id}, (T'_{p,id}[,\ \mathsf{D}_{\mathsf{hist}}{}']), (\mathsf{D}_{\mathsf{debit}}{}',\ \mathsf{D}_{\mathsf{hist}}{}') \rangle / \langle \bot, \bot,\ \bot \rangle \leftarrow$
       $\leftarrow\ \ \mathsf{RefCoupDeposit}(pk_{\mathsf{C}},\ pk_{\mathsf{CIB}}, pk_{\mathsf{AB}})[\mathsf{C}(sk_{\mathsf{C}},\ RefCoup_{ACCA \rightarrow \mathsf{C}}),\ \mathsf{CIB}(sk_{\mathsf{CIB}}),\ \mathsf{AB}(sk_{\mathsf{AB}},\ \mathsf{D}_{\mathsf{hist}},\ \mathsf{D}_{\mathsf{debit}})]$. C interacts with her CIB, CIB, who contacts the AB/ACCA involved to deposit $\mathsf{RefCoupDeposit}$. C's outputs are the new wallets. CIB updates D_{hist} if the error refers to an online transaction. M's entry in AB's $D_{\mathsf{debit}}$is updated.
  - *Version 2:* It refers to a plain purchase cancelation procedure:
    $\langle \mathsf{Ref}_{\mathsf{M}}, \top, \mathsf{D}_{\mathsf{debit}}{}' \rangle / \langle \bot, \bot \rangle \leftarrow\ \mathsf{PurchaseCancelation}(pk_{\mathsf{M}},\ pk_{\mathsf{C}})[\mathsf{C}(\mathsf{Rec}_{\mathsf{T}}),\ \mathsf{M}(sk_{\mathsf{M}})]$, where C obtains a refund $\mathsf{Ref}_{\mathsf{M}}$ from M, which he deposits to her CIB.
- $\langle (T_{det}^i, i = 1 \dots x), \mathsf{D}'_{\mathsf{anon}} \rangle \leftarrow\ \mathsf{OnlineExpenseReportRequest}[\mathsf{C}(\alpha_{\mathsf{C}}(x)),\ \mathsf{B}(\mathsf{D}_{\mathsf{anon}})]$. C requests for an expnse report of her anonymous account with Bank B and provides the $\alpha_{\mathsf{C}}(x)$-realted secret information. $\mathsf{OnlineExpenseReportRequest}$ operation outputes the series of A_{C}^m online transactions of C.

- $\langle {W_{\mathsf{coupons}}}', N_{\mathsf{M}}^{issued} \rangle / \langle \perp, \perp \rangle \leftarrow$ $\mathsf{CouponIssue}(bgpk, N_{\mathsf{coupons}})[\mathsf{C}(Randomness), \mathsf{M}(gbusk_{\mathsf{M}}, N_{\mathsf{M}}^{issued})]$. Merchant M and cardholder C collaborate for the latter to obtain N_{coupons} discount coupons (credit coupons). At the end of this procedure, C obtains a wallet W_{coupons} of N_{coupons} while M updates the number of coupons he has provided (N_M^{issued}).
- $\langle \top, {D_{\mathsf{credit}}}', \mathsf{D}'_{\mathsf{merchants}} \rangle / \langle \perp, \perp \rangle \leftarrow$ $\mathsf{CouponDeposit}(pk_{\mathsf{C}},\ pk_{\mathsf{B}})$ $[\mathsf{C}(W_{\mathsf{coupons}},\ sk_{\mathsf{C}},\ N), \mathsf{B}(sk_{\mathsf{B}},\ \mathsf{D}_{\mathsf{credit}},\ \mathsf{D}_{\mathsf{merchants}})]$. In this operation, customer C deposits N coupons from coupon wallets W_{coupons}. Bank updates the merchants' statistics database as well as the customer's credit account according to N.

B.2 Security Properties

Correctness.

1. If an honest customer C, eligible to own an anonymous credit card, runs AnonymousCreditCardIssue with an honest Bank B, then neither will output an error message; if an honest customer C, who has collaborated with a CIB CIB into a AnonymousCreditCardIssue procedure to issue an anonymous credit card, runs OfflinePayment protocol using that card with an honest merchant M, then M accepts the payment; if M runs MerchantPayment with an honest Bank AB, to deposit the payments he received from honest clients, then M's account will be increased by the value of the ecoins deposited.
2. If an honest customer C, owning a valid ACC, collaborates with her honest CIB CIB, into a AnonymousAccountSetup procedure to create an anonymous account A_{C}, then CIB accepts. In addition, if C
   - runs OnlinePayment protocol using A_{C}'s secret information, then merchant's gateway G accepts the payment: when G runs OnlinePaymentDeposit, the cardholder's CIB CIB accepts and the merchant's account will be increased by the price of the product purchased.
   - runs OnlineExpenseReportRequest with CIB, for her A_{C} account, C receives a detailed report of her A_{C} related online transactions.
3. If an honest cardholder C runs CreditUpdate with an honest Bank B, C's credit account will be increased by the aggregated amount of her purchases since her last CreditUpdate. C will be billed accordingly.
4. If an honest cardholder C detects a charging error and runs either version of ErrorCorrection protocol with honest merchants and/or Banks, then all parties involved will accept and C obtains refund wallets.
5. If an honest cardholder C runs Loss Recovery protocol with an honest Bank B, then she obtains an ACC with the same balance and transaction history as the one lost.
6. If an honest cardholder C obtains discount coupons W_{coupons} from an honest merchant M through CouponsIssue procedure and runs CouponDeposit with an honest Bank, ACCA accepts and updates C's credit account accordingly.

No OverSpending.
No cardholder should be able to spend more money than her credit limit.

Offline ACC Use. – No collection of cardholders should be able to spend more ecoins than the ones contained in their anonymous cards. Suppose that N cardholders $\mathsf{C}_1, \dots, \mathsf{C}_N$ collude together, and that the sum of the amount of ecoins allowed to them is

$$N_C = \sum_{i=1}^{N} {\mathsf{L}_{\mathsf{credit}}}^{\mathsf{C}_i}.$$

Then, the number of different serial numbers of ecoins that can be spent to merchants — in OfflinePayment procedures — and/or Banks — in AnonymousAccountSetup/sf CreditUpdate — procedures is at most N_C.

– Suppose that one or more colluding peers run the OfflinePayment protocol with two merchants M_1 and M_2, such that M_1 gets $({S_{p,id}}^1, {\pi_{p,id}}^1)$ and M_2 gets $({S_{p,id}}^2, {\pi_{p,id}}^2)$, where ${S_{p,id}}^1$, ${S_{p,id}}^2$, ${\pi_{p,id}}^1$ and ${\pi_{p,id}}^2$ the sets of serial numbers of pairs of spent payment-identity ecoins and the corresponding proofs of validity, i.e.

$${S_{p,id}}^i, {\pi_{p,id}}^i = (S_p^i, \pi_p^i) \cup (S_{id}^i, \pi_{id}^i), i = 1, 2.$$

Assume that $S_{p,id} = {S_{p,id}}^1 \bigcap {S_{p,id}}^2 \neq \emptyset$, and $\pi_{p,id}$ the set of spending validity proofs that correspond to $S_{p,id}$. Then, we require that $\mathsf{Identify}_{\mathsf{x}}(S_x, \pi_x), x = p, id$ outputs a public key pk_{C} and a proof of guilt Π^G such that $\mathsf{VerifyGuilt}_x(pk_{\mathsf{C}},\ S_x,\ \Pi^G)$ accepts.

- Each ecoin pair contained in payment-identity wallets that is accepted but not double-spent in the MerchantPayment protocol increases by its exact value merchant's AB's D_{debit} irrespective of the beneficiary of the ecoins. However, we don't regard it as a breach of security when a merchant M_1 received an ecoin but passed it to M_2, who deposited it into his debit account; in any event, this is just another form of collusion. Another justification is that the peer M_1 sacrifices his money.

Online ACC Use. – No collection of cardholders should be able to spend more in online purchases than the sum of their current active anonymous accounts' balance. Suppose that N cardholders $\mathsf{C}_1, \ldots, \mathsf{C}_N$ collude together, and that of the balance $(M_{ot}^{m_i})$ of their active online accounts is

$$N_{AC} = \sum_{\ell=1}^{N} M_{ot}^{m_i}.$$

Then the overall amount of money that can be spent to merchants in online transactions or returned to their owners is N_{AC}.

Error correction may constitute an exception in this property if the cardholder lies for the error having taken place and acquires from the ACCA — if we assume that merchant does not accept to provide a refund — new wallets. We consider this to be a general risk credit card associations take. In addition, as in error correction case customers identify themselves, it is a reasonable assumption may not attempt to cheat in this way multiple times.

Credit Card Unforgeability.

- No customer/merchant or collection of customers and/or merchants should be able to create a credit card of the form of the Anonymous Credit Card described before, which when used to run OfflinePayment or AnonymousAccountSetup protocols with an honest merchant or Bank respectively noone outputs error message.

BackUp Integrity.

- Let that a cardholder C has run BackUp protocol with her CIB, CIB, at a particular date. There should be impossible for C to run Loss Recovery protocol with CIB, with a different backup than the most recent one.

No Unauthorized Use of ACC.
We require that an unauthorized individual may not make use of an ACC's functionalities. In particular, we consider the following cases:

Non Frameability. – No coalition of customers, even with Bank, can forge a proof Π_G that $\mathsf{VerifyGuilt}_{\mathsf{p}}(pk_{\mathsf{C}},\ S,\ \Pi_G)$ or $\mathsf{VerifyGuilt}_{\mathsf{id}}(pk_{\mathsf{C}},\ S,\ \Pi_G)$ accepts where pk_{C} is the public key of an honest cardholder, i.e. a cardholder who did not double-spent an ecoin with the serial number S.

ACC Payment. – (Offline Payments) No cardholder or merchant or collection of customers and/or merchants should be able to use another customer's ACC in OfflinePayment, or AnonymousAccountSetup protocol without possessing sk_{C}.

- (Online Payments) No cardholder or merchant or collection of cardholders and/or merchants should be able to run Online Payment protocol on an anonymous account they do not possess the secret of succesfully, i.e. without error message.

Expense Report Request. – No cardholder or merchant or collection of customers and/or merchants should be able to run OnlineExpenseReportRequest for an account A_{C} successfully, without possessing A_{C}'s secret information.

Conditional Non Traceability.

- *Non Tracing of non-double-spenders.* Given that a cardholder C has issued an ACC with an honest CIB CIB through an AnonymousCreditCardIssue protocol, participated in an OfflinePayment or AnonymousAccountSetup procedure, it should computationally impossible for any merchant or CIB or any collusion between the two to infer any information regarding sk_{C} or pk_{C}. In addition, given the outputs of two different offline spending procedures of C it should be impossible to link one to the other as having been done by the same person.
- (Non Frameability)No coalition of customers, even with Bank, can forge a proof Π_G that $\mathsf{VerifyGuilt}_{\mathsf{p}}(pk_{\mathsf{C}}, S, \Pi_G)$ or $\mathsf{VerifyGuilt}_{\mathsf{id}}(pk_{\mathsf{C}}, S, \Pi_G)$ accepts where pk_{C} is the public key of an honest cardholder, i.e. a cardholder who did not double-spent an ecoin with the serial number S.
- *Tracing of double-spenders.* Given that a cardholder C is shown guilty of double-spending ecoin S_p by a proof Π_G such that $\mathsf{VerifyGuilt}_{\mathsf{p}}$ accepts, this property guarantees that $\mathsf{Trace}_{\mathsf{id}}(\mathsf{params},\ S_{id},\ pk_{\mathsf{C}},\ \Pi_G,\ D_{\mathsf{hist}})$ will output the serial numbers $S_{id_1}, \ldots, S_{id_m}$ of all coins that belong to customer along with proofs of ownership $\Pi_{id_1}, \ldots,\ \Pi_{id_m}$ such that for all i with high probability, $\mathsf{VerifyOwnership_id}(S_{id_i},\ \Pi_{id_i},\ pk_{\mathsf{C}})$ also accepts.

Coupon Security

- (Correctness) If a customer obtains discount coupons W_{coupons} from an honest merchant M through CouponsIssue procedure and runs CouponDeposit with an honest Bank, ACCA accepts and increases M's popularity and customer's credit account accordingly.
- (Balance) No collection of customers should be able to deposit more coupons than the ones legaly issued by merchants $\mathsf{M}_1, \ldots, \mathsf{M}_m$ they interacted. Suppose that N customers $\mathsf{C}_1, \ldots, \mathsf{C}_n$ collude together, and that the sum of the coupons allowed to them is $N_{coupons}$. Then, the number of different coupons that can be accepted when deposited to Bank is at most $N_{coupons}$.
- (Unforgeability) No customer or coalition of customers should be able to create coupon-wallet W_{coupons}, such that when provided to CouponDeposit are accepted by Bank as issued by an honest merchant *merchant*, without M's collaboration.
- (Anonymity) Bank or any coalition of merchants should not be able to link a particular coupon to a particular identity.

B.3 Security Proof

The following theorem states the correctness and security of our general scheme.

Theorem 1. *If the underlying primitives (*ecash *system, blind signatures and group blind signatures) are secure, then our scheme satisfies* correctness*,* no overspending*,* credit card unforgeability*,* backUp integrity*,* no unauthorized use of ACC*,* conditional non-traceability *and* coupon security*.*

Lemma 1. *If the underlying primitives (blind signatures, blind group signatures and E-Cash system) are secure, then our scheme satisfies* correctness*.*

Proof sketch: From the correctness of the secure E-cash scheme and the secure blind signature scheme, our scheme satisfies the first (1) condition of the correctness. Hash functions' invertibility satisfies the second (2) and fourth (4) parts of correctness definition, while the balance property of the E-cash scheme guarantees the third ACC correctness property. The combination of correctness and anonymity revocability attribute of the underlying compact ecash scheme [CHL05]satisfy the fifth (5) condition. The correctness of the secure group blind signature scheme guarantees and of the plain blind signature scheme satisfy the last (6) correction condition (see coupons' security proof analysis for more details). □

Lemma 2. *If underlying primitives (digital signatures, blind signatures, blind group signaturs and E-Cash system) are secure, then our scheme satisfies* Credit Card Unforgeability*.*

Proof sketch: From the unforgeability and consistency of the secure E-cash scheme, our scheme guarrantees that no valid wallets can be created without Bank's collaboration. Thus, Anonymous Credit Card Unforgeability is satisfied. □

Lemma 3. *If the underlying primitives (blind signatures and E-Cash system) are secure, then our scheme satisfies* No OverSpending.

Proof sketch: *Offline Payment.* Being ecash based, the offline conditions of No OverSpending definition are all satisfied by the no overspending property of the underlying secure compact ecash scheme [CHL05].
Online Payment. This is basically controlled by the bank, where the anonymous account is situated. After each transaction, bank substracts from the anonymous account, the amount spent, while it rejects any transaction requiring more money than the anonymous account's balance. □

Lemma 4. *If the underlying primitives (digital signatures) are secure, then our scheme satisfies* BackUp Integrity.

Proof sketch: It is satisfied by the unforgeability property of digital signature. In particular, in the end of the backup procedure, both Bank and customer sign a hash of the encrypted content of the card concatenated with the corresponding date. Customer has no motivation to lie (because of double-spending consequences) and Bank's signature on the hash of the backup acts as undeniable proof of authenticity on client's behalf. □

Lemma 5. *If the underlying primitives (E-cash, digital signatures, hash functions) are secure, then our scheme satisfies* No Unauthorized Use of ACC.

Proof sketch: We will address each ACC use seperately. *Non framability* is satisfied from the exculpability of the secure E-cash scheme. *No unauthorized offline payment* requirement is satisfied through the all or nothing non transferability property of the underlying E-cash scheme. In particular, to make a valid offline payment using an ACC's wallets, the secret key of the ACC owner is required. *No unauthorized online payment* property is also satisfied by the non invertibility properties of hashes. To authenticate herself the person attepting to make a payment is required to provide the pre-image of the current anonymous account number she only knows. Automatic changes of the account number and timestamps prevent a third party to succeed in replay attacks. Authentication in the *Expense Report Request* case is achieved in a similar case as in online payment protocol. □

Lemma 6. *If the underlying primitives (blind signatures and blind group signatures) are secure, then our scheme satisfies* Coupons Security.

Proof sketch:

- (Correctness) Verifiability of a secure goup blind signature scheme guarantees that an "honest but curious" Bank will accept all the credit coupons generated by honest merchants and update the merchant statistics database accordingly. Verifiability of a secure blind signature scheme guarantees that an "honest but curious" Bank will accept all the debit coupons generated by honest customers and update the credit database accordingly.
- (Balance) Credit coupons are subjected to double-use check when deposited and debit coupons are only issued in a rate one for each valid credit coupon. Since debit coupons are also subject to double-using check when deposit, no customer or coalition of customers can eventually deposit more coupons (debit) than the ones initially obtained by merchant.
- (Unforgeability) Unforgeability of blind group signatures guarantee that no customer or coalition of customers can forge credit coupons. Unforgeability of blind signatures guarantees debit coupons' unforgeability.
- (Anonymity) Blindness property of blind signature scheme guarantees that Bank cannot link a debit coupon or a set of debit coupons to a credit or set of credit coupons. Thus *Anonymity* property is satisfied.

□

Lemma 7. *If the underlying primitives (blind signatures, group blind signatures and E-Cash system) are secure, then our scheme satisfies* Credit card conditional non traceability*.*

Proof sketch: Simply from traceability attributes of the secure E-cash scheme, our scheme satisfies the *conditional non Traceability.*

From the E-cash scheme properties, the only case for a customer's identity to be revealed, is when she double-spends part of her wallets to exceed her credit limit (identity or payment). As the two wallets are spent concurrently in all payment procedures, double-spending a part of the payment wallet implies double-spending the corresponding part of the identity one as well. Thus, the double-spender's identity is revealed.

In our scheme, there are three cases, where a customer C double-spends parts of her wallet(s):

1. Closes her card, reports her loss but in the period between the last card backup and the loss of the card, C has used the card to make purchases. In this case, C spends the rest of *only* the payment wallet to Bank. Thus, although C's transactions within this time period (critical) get revealed — when merchants deposit the double-spent parts of C's wallet — her transaction activity preceeding and following the critical period remains secret.
2. C copies her card, and uses two cards at the same time. In this case, identity wallet is spent twice and because of the tracing attribute of tracing enabled E-casht scheme, the serial numbers of all coins C has withdrawn are revealed.
3. C copies her card, updates her credit limit using one copy and uses the other copy to make purchases. This is another case of on purpose double-spending,similar to case 2.

□