Improved Conditional E-Payments

Marina Blanton

Department of Computer Science and Engineering
University of Notre Dame
mblanton@cse.nd.edu

Abstract. Conditional e-cash or conditional e-payments have been introduced by Shi et al. as the means for enabling electronic payments to be based on the outcome of a certain condition not known in advance. In this framework, a payer obtains an electronic coin and can transfer it to a payee under a certain condition. Once the outcome of the condition is known, if it was favorable to the payee, the payee can deposit the coin; otherwise, the payer keeps the money. In this work, we formalize conditional payments and give a scheme to achieve conditional e-payments that outperforms the original solution in several respects.

1 Introduction

Recently Shi et al. [24] introduced conditional electronic payments which allow a participant to anonymously cash bank-issued electronic coin at a future time if a certain agreed-upon condition is satisfied. That is, a payer engages in a protocol with a payee after which the payee has a (conditional) payment that can later be cashed only if a certain public condition is satisfied (or a certain event happens). Such scenarios arise in several contexts including, for instance, trading of financial securities and prediction markets. The conditional nature of electronic payments can, however, be taken more broadly with the outcome of the condition determined by the performance of the payee in carrying out a certain task or by a combination of different conditions.

Such conditional payments have similarities with traditional e-cash systems, but the requirements placed on interaction between entities in a conditional e-payment protocol are different enough for an e-cash scheme to be adopted to this problem. Thus, new tools need to be developed to meet the requirements of conditional payments. To illustrate why e-cash solutions are not sufficient for conditional e-payments, we list some distinctive features of conditional e-cash next. In conditional payments, a payer obtains an electronic coin and anonymously transfers it to an anonymous payee. In traditional e-cash systems, however, the coin is normally bound to the identity of the merchant during the transfer, and the merchant cannot remain anonymous. Furthermore, in conditional payments the payee should be unable to spend the coin until after the outcome of the condition is determined and only if the outcome is favorable to the payee. Additionally, the payer should have the ability to cash the payment in case of an unfavorable to the payee outcome of the condition, which cannot be done in

traditional e-cash. Thus, existing e-cash schemes do not provide an adequate solution to the conditional e-payment setting with the above requirements.

Shi et al. [24] defined the model and necessary properties for conditional e-payments, as well gave the first solution to the problem. Unfortunately, this solution lacks efficiency due to the use of expensive cut-and-choose techniques, and in this work we show that recent advances in electronic cash systems allow conditional payments to be implemented in a more efficient manner completely avoiding cut-and-choose techniques. More precisely, the solution of [24] requires $O(n_1 n_2 k)$ computation and communication, where n_1 and n_2 are cut-and-choose parameters and k is a security parameter for RSA-based systems. Recall that in cut-and-choose techniques with a parameter n a dishonest user can cheat with probability $1/n$, therefore a protocol that has the overhead of $O(n^2 k)$ is likely to be too computation and communication heavy for practical use, which we remedy in this work with a faster solution. In our solution, probability of cheating drops exponentially with the increase in computation and communication. More precisely, we achieve $O(k' \log n_2)$, where k' is a security parameter for groups with bilinear maps, which significantly lowers $O(n_1 n_2 k)$. (The logarithmic factor in our solution is due to the use of verifiable encryption.)

Our contributions. The basis of construction is an e-cash system of Camenisch and Lysyanskaya that follows from CL-signatures with protocols. We modify it to (i) permit payers and payees to stay anonymous during the transfer protocol while maintaining the ability of the bank to trace dishonest payers, and to (ii) incorporate the conditional nature of the transfer. Compared to the solution of Shi et al., our simple scheme has the following advantages:

- Our scheme has lower computation and communication overhead.
- Shi et al.'s solution requires the payee to contact the bank at the time of coin transfer to verify the validity of the coin, while in our scheme all transfers between the payer and the payee are performed off-line, with no participation of the bank or other entities.
- Shi et al.'s solution gave only informal arguments regarding the security of the solution. In this work, we formally state security requirements for a conditional e-payment scheme and provide proofs of security.

Another important contribution of this work is an extension that permits payees to further transfer (conditional) payments to other payees. In this case, double-spending by both payers and payees is addressed.

The rest of this paper is organized as follows. We first give a more detailed description of the model in section 2 and list preliminaries and building blocks in section 3. Our conditional payment scheme is presented in section 4. An extension for handling additional transfers is given in section 5. And section 6 concludes the paper.

2 The Model

A conditional e-payment scheme involves several parties, namely: a bank that issues electronic coins; a publisher that announces public conditions and later

their outcomes; a payer who obtains an electronic coin from a bank and can conditionally transfer it to a payee. In this model, the payer withdraws a certain amount of money from his bank account and obtains an electronic coin for the amount of the withdraw. The publisher announces future events by posting information about them. The payer can conditionally transfer his coin to a payee using the public conditions announced by the publisher. When the outcome of the public condition becomes known, the publisher is trusted to correctly publish the outcome of the event and any other information associated with it. After the event, the payee will be able to validate her coin and cash it only if the published outcome of the event was favorable to her. Otherwise, the payee's coin remains unspendable, and the payer cashes it back.

In the rest of the paper, we use the following terminology: when the payer contacts the bank to have an electronic payment issued to him, we will refer to the token that the payer receives from the bank as a coin. Once a coin is transferred to a payee, we will refer to the token that the payee receives as a conditional payment. Once the outcome of the condition on the payee's conditional payment is announced, it can be transferred into a validated (or casheable) coin. Finally, in the event of unfavorable outcome of the condition, the conditional payment becomes uncasheable.

The properties that the conditional e-cash system in [24] was (informally) shown to have are as follows:

Anonymity. The bank is unable to associate its previously issued coins with the identities of principals cashing them (payers or payees).

Double spending. It is either infeasible to achieve or the identity of a payer will be uncovered if (i) the payer transfers a coin to more than one payee and the payees cash it or (ii) one payee cashes the coin and the payer cashes it as well.

Conditional transfer. In the case of an unfavorable outcome, the payer can cash back his coin. If the payee accepts the coin during the transfer protocol, in the case of a favorable outcome, she will be able to cash the coin.

Deniability. Neither payer nor the payee can prove to outside parties that they participated in a conditional payment protocol.

Limited information flow. The bank cannot infer any event-specific details. The publisher cannot infer any information about bank-payer-payee interactions through the protocol.

In some interactions in the conditional payments scheme, a participant is to stay anonymous. We then assume that in such cases the participant will use a network anonymizer (e.g., [16]) to hide her location information or engage in a protocol using other anonymous means (e.g., a protocol between the bank and an anonymous user can take place at a bank's kiosk).

We show the security of our solution with respect to all properties stated above. The bank and the publisher are trusted to perform their function in the protocol correctly (i.e., the bank issues electronic payments and cashes its previously issued valid coins and the publisher announces the events correctly),

but we also design the protocols to be resilient to collision between different participants. For example, a payer, a payee, and the publisher should not be able to conspire to over-spend a bank-issued payment. Similarly, the bank, the publisher, and payees should not be able to conspire to link a coin to the identity of an honest payer.

As stated above, there are four players in the conditional e-payment system: the bank, the publisher, the payer, and the payee. We will assume that the bank will setup is public-private key pair prior to any payment can be made. The functionality of the system and the interaction between the bank, a payer, and a payee can be described using the following algorithms (the publisher does not interact with other parties, it has a passive role of announcing events and their outcomes):

Payment Generation: A protocol between the bank and a payer that allows the payer to obtain electronic payments from the bank. The bank withdraws from the payer's account the value of the electronic coin it issued.

Conditional Transfer: A protocol between a payer and a payee during which the payer transfers an electronic payment to the payee in such a way that the payee will be able to cash the payment only after the favorable outcome of the agreed-upon condition.

Validating the Payment: After the publisher announces the outcome of the event, if the outcome was favorable to the payee, the payee will use the information posted by the publisher to transform the conditional payment into a casheable coin.

Cashing the Payment: Cashing can be done by either the payee in case of favorable outcome of the condition or by the payer otherwise. In either case the claimant anonymously submits an electronic payment to the bank and receives cash in the amount of the payment.

Identifying Double-Spenders: This algorithm is invoked by the bank on input a coin's serial number s and two validity proofs for it. If a payee does not attempt to spend the same coin twice (with the same proof), this algorithm will reveal the identity of the payer (who either transferred the coin to more than one payee or cashed the coin that was also casheable by a payee).

The solution of [24] also included a Payment Activation protocol: a protocol between the bank and the payer, where the payer anonymously contacts the bank and activates the electronic payment issued during the payment generation protocol. This protocol, however, is not required.

We now give a more formal definition of a conditional e-payment system and its security properties.

Correctness. If an honest payer generates a conditional payment and engages in a conditional transfer protocol with a payee, then the payee will accept. If a payee is honest and accepts during the conditional transfer protocol, then in case of the favorable outcome the payee will be able to cash the payment. If a payer is honest and does not transfer the payment or transfers the payment and the outcome is unfavorable to the payee, the payer will be able to cash the payment.

Anonymity. When addressing anonymity of users, we need to consider two different cases: anonymity of payers and anonymity of payees. Anonymity of payers means that other participants (e.g., the bank, the publisher, and the payee) cannot link the coin an (honest) payer transfers to a payee to the payer's identity (assuming that the payer does not double-spend the coin). Thus, we model the adversary $\mathcal{A}$ as colluding bank, payees, and the publisher. $\mathcal{A}$ will be able to create the bank's private and public keys and engage in queries, where $\mathcal{A}$ executes the payment generation protocol with various users id_i. Then $\mathcal{A}$ engages in a challenge conditional transfer and the consecutive payment validation, where it is interacting either with a real user or a simulator with no access to any user information. The anonymity requirement for the payer is such that, for any adversary $\mathcal{A}$, given all information $\mathcal{A}$ receives during payment generation, transfer, and validation, $\mathcal{A}$ is unable to distinguish between a real payer and a simulator with more than negligible probability.

To model anonymity of payees, the adversary $\mathcal{A}$ will represent the bank colluding with payers. In this case, $\mathcal{A}$ will be able to create the bank's key, create users, and issue coins. To ensure that a payee's identity is not revealed at any point during the protocols, we require that $\mathcal{A}$ cannot distinguish between a real user id_j and a simulator (with no access to user-specific information) with more than negligible probability during both the conditional transfer protocol and cashing payment protocol.

Balance. There are two parts to this property. From the bank's point of view, we would like to assure that no coalition of dishonest payers and payees can cash more coins than they withdrew. This is often shown by treating the payment generation protocol as a proof where the user plays the role of the prover and the bank plays the role of the verifier. Let x denote the user's input to the payment generation protocol. Then if the bank accepts at the end of the protocol, there will be a knowledge extractor that extracts a witness $w = s$, which is the serial number associated with the issued payment. Let the adversary $\mathcal{A}$ engage in a number of payment generation, payment transfer, and payment cashing protocols. Let on ith successful execution of the payment generation protocol the extractor's output be (x_i, s_i). We say that adversary wins if, for any number n of payment generation protocol executions, $\mathcal{A}$ is able to cash a payment with a serial number $s \notin \{s_1, \ldots, s_n\}$. We say that the balance property holds if $\mathcal{A}$ has at most negligible probability of winning.

From the payer's point of view, if the payment was transferred to a payee and the outcome is unfavorable to the payee, that payee should not be able to produce a valid casheable payment. To be able to show this, we let $\mathcal{A}$ act as either the payer or the payee with various users (by possibly corrupting other users) and, when acting as the payee, request favorable outcome for its events. Then $\mathcal{A}$, acting as the payee, engages in a challenge conditional transfer protocol with an uncorrupted payer, such that the outcome of this event is unfavorable. We say that the protocol is secure if any $\mathcal{A}$ can recover a casheable token with at most negligible probability.

Double-spending. In this case the adversary $\mathcal{A}$ represents a payer possibly in coalition with one or more payees; the bank is assumed to be honest. $\mathcal{A}$ engages in the payment generation, conditional transfer, and cashing protocols as many times as it likes. $\mathcal{A}$ wins the game if the bank accepts two requests to cash the same payment with serial number s without being able to recover the identity of $\mathcal{A}$. We say that identification of double-spenders is achieved if any $\mathcal{A}$ can succeed at double-spending with at most negligible probability.

Deniability. Deniability was defined in [24] as the inability of a payer or a payee to prove to outside parties that he or she participated in a conditional payment protocol. This issue, however, requires further investigation to specify what constitutes a proof of engagement in such a protocol. If, for example, the format or the use of coins issued for a conditional payment protocol differs from coins issued by the bank for different purposes, then the mere fact that a user requests the bank to issue a conditional e-payment token to her indicates the intent to engage in a conditional transfer protocol. In this case, both the solution of Shi et al. and our solution fail to provide deniability. Furthermore, we argue that in many applications of conditional e-payments deniability might not be a requirement or a different type of deniability might be preferred (where, e.g., a participant – a payer or a payee – cannot provide a proof that its peer engaged in a conditional transfer protocol). Thus, we do not further treat deniability in this work.

3 Preliminaries

In this section, we review certain cryptographic primitives used as building blocks in our solution, as well as their security guarantees.

3.1 Zero-knowledge proofs of knowledge

Prior literature provides efficient zero-knowledge proofs of knowledge (ZKPK) for a variety of statements, with many efficient proofs being based on the discrete logarithm problem (see, e.g., [14, 13, 11, 3, 4]). Camenisch and Stadler [12] introduced notation for various proofs of knowledge and we follow their notation here. For example,

$$PK\{(\alpha, \beta, \gamma) : A = g^{\alpha}h^{\beta} \wedge B = g^{\alpha}h^{\gamma} \wedge (\alpha \geq a)\}$$

denotes a ZKPK of integers α, β, and γ, where $A = g^{\alpha}h^{\beta}$, $B = g^{\alpha}h^{\gamma}$, and $\alpha \geq a$.

ZKPKs used in our protocols are proof of knowledge of the discrete logarithm representation, equality of discrete logarithms, and linear equations on the discrete logarithms, solutions to which are well known. We also utilize proof of knowledge that a discrete logarithm is the product of two other committed values [11].

3.2 Signature schemes

In this work, we use signature schemes with protocols due to Camenisch and Lysyanskaya [8, 9]. These schemes have two protocols associated with them: (i)

they allow a user to obtain a signature on a committed value without revealing that value to the signer; and (ii) they enable the user to convince a third party that she possesses a signature on a certain message.

The commitment scheme used is the Pedersen commitment scheme [22]. Recall that in this scheme the public parameters are a group G_q of prime order q such that the discrete logarithm problem in G_q is hard and generators g_0, g_1, ..., g_k. In order to compute a commitment to $x_1, \ldots, x_\ell \in \mathbb{Z}_q$, we choose $r \in \mathbb{Z}_q$ at random and compute $com(x_1, \ldots, x_\ell; r) = g_0^r \prod_{i=1}^{\ell} g_i^{x_i}$. This commitment is unconditionally hiding (i.e., $com(x_1, \ldots, x_\ell; r)$ reveals no information about $x_1, \ldots, x_\ell$) and is computationally binding (assuming that the discrete logarithm problem is hard in G_q, the sender cannot open the commitment to values other than $x_1, \ldots, x_\ell$).

Then given a commitment $com(x_1, \ldots, x_\ell; r)$, it is possible obtain the signer's CL signature $\sigma(x_1, \ldots, x_\ell)$ without revealing any information about the values $x_1, \ldots, x_\ell$ to the signer. Furthermore, possession of $\sigma(x_1, \ldots, x_\ell)$ allows its owner to use commitments to $x_1, \ldots, x_\ell$ to prove to other parties that she has the signer's signature on the values included in the commitments without revealing additional information about the signed values themselves. If this protocol is combined with a ZK proof that the values included in these commitments satisfy certain properties, it becomes possible to convince a third party that the prover possesses a CL signature that meets these conditions without disclosing additional information about the signed values.

The signature scheme [8] relies on the Strong RSA assumption for its security. The scheme [9] relies on LRSW assumption in groups with bilinear maps. Our and other e-cash solutions built on such schemes require certain ZK proofs to be non-interactive, which is normally done by applying Fiat-Shamir heuristic [18]. Such non-interactive ZK proofs are, however, secure only in the random oracle model. Recent work by Belenkiy et al.[2] is the first to give new signatures with protocols, called P-signatures, that have non-interactive protocols without relying on the random oracle model. In particular, they utilize techniques of Groth and Sahai [19] to permit non-interactive zero-knowledge proofs that the contents of a commitment has been signed and that a pair of commitments are committed to the same value. The protocols used in this work, however, involve proofs of more general statements than equality, and more research is needed to determine what types of statements about discrete logarithms can be proven non-interactively using techniques of Groth and Sahai.

3.3 Verifiable encryption

Verifiable encryption is a protocol between an encryptor and verifier that allows the encryptor to convince the verifier that encryption was performed correctly. Given a public encryption key pk and a commitment $C = com(x)$, this protocol allows the encryptor to produce encryption, $E_{pk}(x)$, of the opening of C, such that the verifier can accept an invalid encryption only with a negligible probability. Then given the corresponding decryption key sk and the protocol transcript for $E_{pk}(x)$, opening of C can be computed efficiently. Camenisch

and Damgard [5] provide techniques for converting any semantically secure encryption scheme into a verifiable encryption scheme, and this is what adds the logarithmic factor to the complexity of our scheme. For our purposes, any secure verifiable encryption scheme will suffice.

4 Conditional E-Payments

4.1 Description and Intuition

With the recent advances in e-cash systems, a natural place to seek alternatives for expensive cut-and-choose techniques of Shi et al. [24] conditional e-cash is to look at other e-cash solutions. Recent e-cash systems such as, e.g., [6, 7, 10] use CL-signatures with constant overhead as their building block. Using CL-signatures, it is possible to construct an e-cash system using a known method as follows: to generate a payment, a user obtains the bank's signature on (id, s, t), where id is the user's identity, s is the serial number of the coin, t is the blinding value without the bank knowing s or t. Then when the user would like to spend the coin at a merchant, the user provides the merchant with commitments to id, s, and t and a ZK proof that she possesses a signature on these values. The merchant chooses a random value R (computed as a function of the merchant's identity and additional information provided by the merchant) and communicates R to the user. The user reveals to the merchant the serial number s and the result of evaluating the double-spending equation $D = id + R \cdot t \bmod q$ on R, as well as provides a ZK proof of correctness of both s and D (i.e., that s appears in the signature, and D is computed using id and t from the signature). Now, if only a single value D is computed for a coin, it does not reveal any information about the user's identity id (since t was chosen at random); but revealing two such values for different R's reveals the identity of the user.

The biggest difference between the regular e-cash setting and conditional e-payments is that (i) in e-cash a coin transferred to a merchant carries a validity proof that is bound to the merchant (and only that merchant can cash it), while in conditional e-payments each payee is to stay anonymous, and (ii) the only way for a user to spend a coin in e-cash is through a merchant, while in conditional e-payments, the payer is able to cash coins herself. To address the first item, we can require each (anonymous) payee to challenge the payer on a randomly chosen value R, which, unlike the e-cash systems, is not bound to the payee's identity. This will ensure that if the payer transfers the coin to more than one payee, the payer's identity can be recovered. In case when the payer does not transfer the payment or the outcome of the event is unfavorable, the payer can transfer a coin to himself and cash it anonymously. This, however, creates a problem, as the payer will be able to quickly cash the coin using the payee's R before the payee learns the outcome of the condition. And when the (honest) payee attempts to cash the same payment, the request will be denied by the bank as duplicate (without the ability to uncover the payer's identity).

This could be fixed in the same way as in [24]: a trusted authority (e.g., the bank) keeps track of payment transfers.[1] We, however, are interested in a solution that can be performed completely off-line, without the need of the bank to participate in payment transfers. Even though both participants are to stay anonymous, we let the payee to obtain a proof of the payment in an oblivious way, without the payer knowing the value of R. That is, the payee sends to the payer a commitment to R, g^R, and the payer then transfers that commitment into g^D and also provides a proof that the value was formed as prescribed. Note that knowledge of R is required to be able to cash the payment, and the payer is prevented from doing so. Given commitments to the double spending equation evaluated on different values, it is still possible to recover the identity of the payer (more precisely, now the bank will be able to recover g^{id} instead of the value of id itself, but this does not affect the security of the solution[2]).

In the above solution, the payee's participation in the conditional transfer and subsequent cashing protocols is linkable due to the use of the payer's validity proofs during cashing. Thus, to prevent colluding payer and the bank from recovering the payee's identity, the payee must stay anonymous during the deposit protocol. This means that either the payer cashes the e-payment anonymously or exchanges it for another anonymous token such that its generation and deposit protocols cannot be linked together (and this is what was suggested in [24]). Furthermore, the techniques we are utilizing do not seem to permit a payee's participation in the conditional transfer and cashing protocols to be unlinkable, as the payee is unable to modify the ZK proof generated by the payer. We leave this unlinkability issue as an open problem.

The final issue that remains to be addressed is the conditional nature of the transfer, where the payee's payment must remain uncasheable at the time of the transfer while still ensuring that the payee can verify all of the payer's proofs. We solve this by conditionally hiding only partial information which is necessary for cashing the payment. In particular, the payee will be able to obtain access to the serial number s only in case of favorable outcome of the condition.

We model events as follows: When event $\mathcal{E}$ is announced, the publisher posts the public key $pk_{\mathcal{E}}$ associated with this event. Later, when the outcome of the event is determined, if the outcome satisfies the condition, the publisher releases the corresponding private key $sk_{\mathcal{E}}$. In case of unfavorable outcome, the publisher does not release any additional information. At the time of transfer, the payer encrypts the payment-related information using $pk_{\mathcal{E}}$, and the payee will be able

[1] In more detail, a coin consists of two halves (the right half is used for conditional transfers and the left half is used to cash unspent payments) such that spending both halves reveals the identity of the payer. The payer first (anonymously) activates a payment using its serial number s. During the transfer protocol, the payee contacts the bank with s and obtains a secret value that must be presented to the bank upon cashing the right half of the coin. Thus, the payer can cash only the left half.

[2] Furthermore, in previous e-cash schemes [6, 7, 10], each user was identified by a public key instead of her identity. Then for user U, the secret key is $sk_U \in \mathbb{Z}_q$ and the corresponding public key is $pk_U = g^{sk_U}$. Using our terminology, the secret key is id and the public key is g^{id}.

to decrypt it and produce a casheable coin only if the private key $sk_{\mathcal{E}}$ is announced. To ensure that the payer does not cheat during the encryption process, we employ verifiable encryption. More precisely, the payer forms a commitment to the serial number s and verifiably encrypts s using $pk_{\mathcal{E}}$. The payee will accept the payment only is she is able to verify the correctness of the encryption.

We also make use of another key pair ($pk_{\bar{\mathcal{E}}}$, $sk_{\bar{\mathcal{E}}}$) that correspond to the event $\bar{\mathcal{E}}$ opposite of $\mathcal{E}$ (that is, $sk_{\bar{\mathcal{E}}}$ is published in case of unfavorable outcome of the event $\mathcal{E}$). The key $pk_{\bar{\mathcal{E}}}$ enables encryption of the value R to the payer, which he decrypts in case of unfavorable outcome and cashes the coin back.

4.2 The Scheme

Our scheme is as follows: The common parameters to all players consist of a group G of prime order q (chosen according to a security parameter 1^κ) and generators $g_0, g_1, \ldots$. The bank generates its private signing key sk_B and the corresponding verification key pk_B using one of the CL-signature algorithms. This key will be used to sign e-payments the bank issues.

Payment Generation: The interaction between a payer and the bank is as follows:

1. The payer identifies herself as having id. The bank and the payer also agree on the amount of withdraw v. (We assume that $id, v \in \mathbb{Z}_q$.)
2. The payer with the help of the bank computes a commitment to s, t, id, and v, where s is a random value used as the payment's serial number (jointly generated by the payer and the bank). To achieve this:
   (a) The payer chooses $s_P, t \in \mathbb{Z}_q$ at random, forms a commitment $C_1 = com(s_P, t; r) = g_0^r g_1^{s_P} g_2^t$, and sends it to the bank along with a ZK proof of knowledge of the representation of C_1 with respect to bases g_0, g_1, and g_2.
   (b) The bank chooses $s_B \in \mathbb{Z}_q$ at random and sends it to the payer.
   (c) Both the payer and the bank use s_B, id, and v to locally compute $C = com(s, t, id, v; r)$, where $s = s_P + s_B$. This is done by setting $C = C_1 \cdot g_1^{s_B} \cdot g_3^{id} \cdot g_4^v$.
3. The payer and the bank run a protocol for obtaining a signature on a committed value, at the end of which the payer has bank's signature $\sigma_B(s, t, id, v)$.
4. The bank debits the payer's account with amount v.
5. The payer stores $(id, s, t, v, \sigma_B(s, t, id, v))$ as his e-payment token.

When the payer would like to transfer the payment to a payee, both of them need to agree on the public condition $\mathcal{E}$ announced by the publisher. The public key $pk_{\mathcal{E}}$ is then used during the conditional transfer.

In the conditional transfer protocol, the payer needs to convince the payee in the validity of the payment, which is done by using protocols associated with CL-signature and additional ZKPKs. Both participants stay anonymous during the protocol.

Conditional Transfer: The protocol between a payer with $(id, s, t, v, \sigma_B(s, t, id, v))$ and a payee proceeds in the following steps:

1. The payer forms commitments $C_1 = com(s; r_1)$, $C_2 = com(t; r_2)$, and $C_3 = com(id; r_3)$, and sends them along with v to the payee.
2. The payer sends to the payee a proof of knowledge π_1 of a CL signature from the bank on the openings of C_1, C_2, and C_3 as well as the value v.
3. The payee chooses $R \in \mathbb{Z}_q$ at random and sends $A = g^R$ to the payer.
4. The payer computes $B = g^{Rt+id}$, sends it to the payee, and executes a proof of knowledge π_2 of well-formness of B. More precisely, the ZKPK is:

$$PK\{(\alpha, \beta, \gamma_1, \gamma_2) : B = A^{\alpha} g^{\beta} \wedge C_2 = g_1^{\alpha} g_0^{\gamma_1} \wedge C_3 = g_1^{\beta} g_0^{\gamma_2}\}$$

5. The payer verifiably encrypts s under the key $pk_{\mathcal{E}}$ using C_1. The payee obtains $E_{pk_{\mathcal{E}}}(s)$, which includes the proof that the encryption was formed properly.
6. The payee verifiably encrypts R under the key $pk_{\bar{\mathcal{E}}}$ using A. The payer obtains $E_{pk_{\bar{\mathcal{E}}}}(R)$, which includes the proof that the encryption was formed properly.
7. As the result of this transfer, the payee stores $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, E_{pk_{\mathcal{E}}}(s))$ and the payer stores $(C_1, C_2, C_3, v, \pi_1, E_{pk_{\bar{\mathcal{E}}}}(R), A, B, \pi_2, s)$.

After the outcome of the event is announced, if it was favorable to the payee, the payee obtains the private key $sk_{\mathcal{E}}$ corresponding to $pk_{\mathcal{E}}$ and performs the following payment validation procedure.

Validating the Payment: On input $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, E_{pk_{\mathcal{E}}}(s))$ and $sk_{\mathcal{E}}$, the payee decrypts $E_{pk_{\mathcal{E}}}(s)$ obtaining s and stores $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, s)$ as a casheable payment.

In case of unfavorable outcome, the payer transfers his coin into a casheable payment using $sk_{\bar{\mathcal{E}}}$ (i.e., given $(C_1, C_2, C_3, v, \pi_1, E_{pk_{\bar{\mathcal{E}}}}(R), A, B, \pi_2, s)$, the payer decrypts $E_{pk_{\bar{\mathcal{E}}}}(R)$ to obtain $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, s)$. In case the payer did not engage in any conditional transfer protocol and would like to cash the coin back, the payer engages in the transfer protocol with himself without encrypting the serial number. In this case, the casheable coin is the same as in other cases.

The following protocol is performed anonymously.

Cashing the Payment: The claimant's input consists of a casheable coin $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, s)$.

1. The claimant submits the payment $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, s)$ to the bank.
2. The bank verifies all proofs and, in particular, that $g^R = A$. The bank also checks whether the pair (s, R) was previously submitted to the bank. If the proofs are correct and (s, R) is fresh, the bank credits the claimant's account with amount v; otherwise, it rejects the payment.
3. The bank records the payment $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, s)$ in the database of spent payments and searches for another record with s. If s has been used before, it invokes the identification algorithm.

Identifying Double-Spenders: If a coin with a serial number s can be found in the bank's database more than once, the bank identifies the guilty payer as follows:

- The bank locates two records (s, R_1, B_1) and (s, R_2, B_2). (Recall that each $B_i = g^{R_1 t + id}$ for some value t unknown to the bank.)
- It computes the identity of the payer by first computing $d_1 = (B_1/B_2)^{(R_1 - R_2)^{-1}}$ and then $d_2 = B_1/(d_1^{R_1})$. The value of d_2 will correspond to g^{id}, where id is the identity of the double-spender.
- The bank searches its database for an id that matches g^{id}.

We briefly show that d_2 indeed corresponds to the value g^{id}:

$$d_2 = \frac{B_1}{d_1^{R_1}} = B_1 \left(\frac{B_2}{B_1}\right)^{(R_1 - R_2)^{-1} R_1} = g^{R_1 t + id} \left(g^{R_2 t + id - R_1 t - id}\right)^{(R_1 - R_2)^{-1} R_1}$$
$$= g^{R_1 t + id} \left(g^{-t(R_1 - R_2)}\right)^{(R_1 - R_2)^{-1} R_1} = g^{R_1 t + id} g^{-R_1 t} = g^{id}$$

Efficiency. Computation and communication in all of our protocols are constant (more precisely, linear in the security parameter), with the exception of verifiable encryption used in the transfer protocol. To achieve probability of cheating 2^{-k}, $O(k)$ encryptions must be performed during the transfer protocol. Additionally, to decrypt a value during payment validation, $O(k)$ decryptions are to be performed.

4.3 Security Analysis

Theorem 1. *Assuming the security of the CL-signature scheme and the verifiable encryption scheme, the above scheme is a secure conditional e-payment scheme in the random oracle model.*

Proof. We analyze the scheme w.r.t. the properties given in section 2.

Correctness. This property can be shown by examination.

Balance. We first show that dishonest users cannot cash more coins than what they withdrew. Let X_{pg} denote a knowledge extractor in the payment generation protocol that acts as the bank and X_{pk} denote a knowledge extractor for the proof of knowledge executed in step 2(a) of the protocol. Then during each execution of the payment generation protocol by the adversary $\mathcal{A}$, X_{pg} executes the protocol as the bank would with the exception that it also executes X_{pk} to obtain values id, s, t, v. After n executions of the protocol, X_{pg} has the knowledge of n serial numbers $s_1, \ldots, s_n$. Now note that the soundness property of the proofs of knowledge prevents $\mathcal{A}$ from successfully producing a valid serial number $s' \notin \{s_1, \ldots, s_n\}$. Therefore, $\mathcal{A}$ will attempt to deposit a coin for which it cannot honestly generate a valid proof. For $\mathcal{A}$ to succeed in convincing the bank that it has a valid coin with serial number s', $\mathcal{A}$ must produce a proof that either $\mathcal{A}$ knows the bank's signature on openings of C_1, C_2, and C_3 or B is well-formed. The former can happen only with a negligible probability assuming that the

CL-signatures are secure, and the latter can also happen only with a negligible probability assuming that the discrete logarithm problem is hard. Therefore, $\mathcal{A}$ can succeed in winning this game with at most negligible probability.

To show that the adversary $\mathcal{A}$, acting as a payee, is unable to spend its coin in case of unfavorable outcome of the event, let $c_1, \ldots, c_n$ denote the sequence of casheable coins $\mathcal{A}$ acquires in the first part of the game (acting as both payers and payees), where the pairs $(E_{pk_{\mathcal{E}_1}}(s_1), s_1), \ldots, (E_{pk_{\mathcal{E}_n}}(s_n), s_n)$ are the corresponding encryptions of coin serial numbers produced during the conditional transfer protocols and their decryptions. $\mathcal{A}$ then engages in a challenge transfer protocol with an honest payer and obtains coin $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, E_{pk_{\mathcal{E}}}(s))$. Now assume that $\mathcal{A}$ is able to recover a casheable payment from this coin without access to $sk_{\mathcal{E}}$. Then $\mathcal{A}$ must either recover the correct value of s or produce a valid coin using another serial number s'. As was argued above, the latter is possible only with a negligible probability. $\mathcal{A}$ will then attempt to recover s from the commitment C_1 and the associated proof of knowledge or encryption $E_{pk_{\mathcal{E}}}(s)$. Since the commitment is unconditionally hiding and the proof of knowledge is zero-knowledge, $\mathcal{A}$ must use $E_{pk_{\mathcal{E}}}(s)$. Finally, assuming the security of the verifiable encryption scheme, $\mathcal{A}$ can recover s from $E_{pk_{\mathcal{E}}}(s)$ with at most negligible probability. Therefore, $\mathcal{A}$ can succeed in producing a casheable coin in this game with at most negligible probability.

Anonymity. Anonymity of a payee is trivially achieved, since at no point in the protocol the payee is required to present her identity or include it in the information exchanged between the payee and the payer or the bank.

To show payer anonymity, let the game setup be as described in section 2. Adversary $\mathcal{A}$ represents other participants colluding together, who can create users and engages in payment generation protocols. During the challenge, $\mathcal{A}$ is asked to engage in a conditional transfer protocol and execute the consecutive payment validation procedure with either a real user id_j or a simulator S without access to user id_j's information. Our simulator S participates in a conditional transfer protocol by performing the following steps:

1. S chooses $id, s, t, v \in \mathbb{Z}_q$ and produces commitments $C_1 = com(s; r_1)$, $C_2 = com(t; r_2)$, and $C_3 = com(id; r_3)$.
2. S produces a simulated proof of knowledge π_1 of a CL signature from the bank on the openings of C_1, C_2, C_3, and the value v. This requires usage of the corresponding simulator of CL-signatures.
3. After obtaining $A = g^R$, S computes $B = g^{Rt+id}$ and produces a proof of knowledge π_2 of well-formness of B.
4. S verifiably encrypts s under the key $pk_{\mathcal{E}}$ using C_1 producing $E_{pk_{\mathcal{E}}}(s)$.

At the end of this interaction, $\mathcal{A}$ obtains $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, E_{pk_{\mathcal{E}}}(s))$. After executing the challenge transfer protocol (with either a real user or a simulator), $\mathcal{A}$ is given $pk_{\mathcal{E}}$ and validates the coin obtaining $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, s)$. We next argue that a casheable coin produced by a real payer is indistinguishable from a casheable coin produced by the simulator.

First, the payment generation protocol does not allow the bank to learn any information about the coin-specific values s and t. When a payer is issued a CL-signature $\sigma_B(s, t, id, v)$ by the bank, all values in it are information-theoretically hidden (i.e., the signature is issued on the values (s, t, id, v, r) for a random value r rather than (s, t, id, v); this r is obtained from the commitment that the payer submits and information-theoretically hides the values both in the commitment and in the signature). Therefore, the values s and t that S chooses are indistinguishable from those chosen by real users.

Other values produced by S in the transfer protocol are commitments C_1, C_2, C_3 (which perfectly hide the values), real proofs of knowledge π_2 (which therefore are indistinguishable from a payer's proofs), real verifiable encryption $E_{pk_{\mathcal{E}}}(s)$ which consequently is decrypted, and one simulated proof of knowledge π_1. This simulated proof differs from a real proof of knowledge of a CL signature, but due to the security of CL-signatures, $\mathcal{A}$ can distinguish between a real and simulated proofs only with a negligible probability.

Double spending. There are two different ways for $\mathcal{A}$ (representing a coalition of dishonest users) to double-spend a coin with serial number s: (1) by transferring it to two different (honest) payees, both of whom are able to cash it later, or (2) by transferring it to a single (honest) payee, recovering the payee's challenge R, and cashing the coin while the payee has a casheable coin.

In the first case, suppose the bank accepts two casheable coins $c_1 = (C_1, C_2, C_3, v, \pi_1, R, A,\ B, \pi_2, s)$ and $c_2 = (C_1', C_2', C_3', v, \pi_1', R', A', B', \pi_2', s)$ with the same serial number s. Since there are knowledge extractors for all proofs included in the payments, $\mathcal{A}$ knows $\sigma_B(s, t, id, v)$ and $B = g^{id+Rt}$ (resp., $B' = g^{id+R't}$) (more precisely, these conditions can fail with at most negligible probability). Now, because R and R' can happen to be the same only with a small probability, the protocol for identifying double-spenders in this case will succeed in recovering the identity id.

Next, consider the case where $\mathcal{A}$ transfers its payment to a single honest payee, and the payee is later able to obtain a casheable coin (i.e., the condition outcome is favorable). $\mathcal{A}$ might attempt to produce a casheable coin using the payee's challenge R from the transcript of the conditional transfer protocol. $\mathcal{A}$ in this case has $(C_1, C_2, C_3, v, \pi_1, E_{pk_{\bar{\mathcal{E}}}}(R), A, B, \pi_2, s)$. To produce a casheable coin, $\mathcal{A}$ will have to recover R from either $A = g^R$ or $E_{pk_{\mathcal{E}}}(R)$. However, in the first case $\mathcal{A}$ will have to solve the discrete logarithm which, by our assumption, is infeasible, and in the first case to circumvent verifiable encryption scheme $E(\cdot)$, which is assumed to be secure. Therefore, in this case $\mathcal{A}$ also succeeds with at most negligible probability.

5 Further Transfer of Coins

In this section, we sketch how a payee can further transfer a conditional payment to another payee. Solutions to achieve transferable e-cash, including off-line systems, have been previously proposed in the literature (see, e.g., [15, 21, 1, 20, 23]), but to the best of our knowledge, no solution of the kind we propose (or

even another solution based on CL-signatures) has previously appeared in the literature.

Now each payee who would like to be able to further transfer conditional payments will need to be registered with the bank to permit recovering her identity in case of double-spending. Note that a payee will be able to transfer a coin under the same condition as the one used in issuing the payment to that payee.[3] To transfer a payment, each payee can use the bank's signature on the payee's identity and other information similar to the coins themselves. This, however, will require a signature per transfer since using it more than once reveals the identity of its owner. To enable a user to perform multiple transfers using the same credential from the bank, we modify the double-spending equation: the property we now desire is that using this credential on a unique serial number once will leave the user anonymous, but using it twice on the same serial number will allow the identity of the user to be uncovered. More precisely, evaluating this equation on a pair (s_1, R_1) and (s_2, R_2), where $s_1 = s_2$ and $R_1 \neq R_2$ will lead to recovery of the identity, but observing the results of evaluating the equation on any number of values $(s_1, R_1), (s_2, R_2), \ldots, (s_n, R_n)$ where all s_i's are unique does not reveal information about the user. To achieve this, our idea is to have the double-spending equation in the same form $D_i = id + R_i \cdot t$, but make t dependent on s_i. This means that different values of s_i's will result in different values of t and therefore different equations, but evaluating it on the same s and two values of R will permit recovery of id. The function $t = f(s)$ should be deterministic and such that, knowing s, computing t is difficult. For example, we can compute t using a pseudo-random function and a secret key u known to the user only. Then the user will obtain the bank's signature on a pair (id, u) without revealing u to the bank, and during a coin transfer compute $t = f_u(s)$ and $D = id + Rt$. The function f should be such that the user is able to convince the other protocol participant in zero-knowledge that both t and D were computed as prescribed (and according to the bank's signature on id, u).

User Registration: The interaction between a user and the bank is as follows:

1. The user identifies herself as having id and sends to the bank a commitment $C' = com(u; r)$ for a randomly chosen $u \in \mathbb{Z}_q$.
2. The user proves to the bank in zero-knowledge that she knows the representation of C' with respect to bases g_0 and g_1.
3. Both the user and the bank use id to locally compute $C = com(u, id; r)$ by setting $C = C' \cdot g_2^{id}$.
4. The user and bank run a protocol for obtaining a signature on a committed value, at the end of which the user has bank's signatures $\sigma_B(u, id)$.
5. The user stores $(id, u, \sigma_B(u, id))$ as her payment transfer credential.

[3] Since the payee does not have access to the serial number s, she cannot use other conditions for hiding it. Furthermore, even when the payment becomes casheable after successful outcome of the original event, that payee still will not be able to construct verifiable encryption of s using the (payer's) commitment C_1.

The payment generation protocol and the conditional transfer protocol between the payer and the first payee remain mostly unchanged. The only difference in the first conditional transfer protocol is that, in order for the payee to further transfer her coin, she will need to prove statements involving the coin's serial number (to prove that the double-spending equation was constructed correctly). Therefore, we modify the commitment C_1 generated in the first step of the conditional transfer protocol to be $C_1 = com(s)$. This means that $com(s)$ is of the form g^s and does not unconditionally hide the value of s. The knowledge of s, however, is required for cashing the payment, and recovering it from g^s requires solving the discrete logarithm problem. Therefore, we relax the hiding property of the commitment to permit further transfer of payments.

Now suppose that a payee would like to transfer her conditional payment to another payee. This operation can be performed any number of times, and we denote the chain of payees associated with some payment s as $P_1, P_2, \ldots$. When P_1 transfers her conditional payment to P_2 using coin $c = (C_1, C_2, C_3, v, \pi_1, R_0, A, B, \pi_2, E_{pk_\mathcal{E}}(s))$, the new challenge R_1 is computed as a one-way function of the previous value R_0 and randomness r_1 contributed by P_2. That is, $R_1 = f(R_0, r_1)$ and the idea is to chain the sequence of challenges $R_0, R_1, R_2, \ldots$, so that future values in the chain are unpredictable to earlier participants and P_i cannot completely control the value of R_i. For P_2 to convince P_1 that R_1 was computed correctly, the computation must be verifiable in zero-knowledge without disclosing R_1 or r_1. Therefore, we can, for example, use the verifiable random function $f_k(x) = g^{1/(k+x)}$ of Dodis and Yampolskiy [17] (which is the most efficient construction) as $R_i = f(R_{i-1}, r_i) = g^{1/(r_i + R_{i-1})}$.

Going back to evaluating the double-spending equation on coin- and user-specific values, we have $t = f_u(s)$. The above DY function can also be used here to compute t. But since computation of this function would require access to the serial number s, which is not known at the time of the transfer, commitment $C_1 = g^s$ can be used instead. If all participants use an agreed-upon method of mapping any g^s to an element s' of the appropriate group (i.e., $\mathbb{Z}_q^*$), the computation will be of the form $t = f_u(s')$.

We next describe the protocol for transferring a payment from P_1 to P_2. Further transfers from P_i to P_{i+1} are performed analogously, and each new transfer maintains information about all previous transfers. P_1's input is the original coin $c = (C_1, C_2, C_3, v, \pi_1, R_0, A, B, \pi_2, E_{pk_\mathcal{E}}(s))$.

Additional Transfer:

1. P_2 chooses r_1 and sends $C = com(r_1; r')$ to P_1. P_1 sends R_0 to P_2.
2. P_2 computes $R_1 = f(R_0, r_1)$ and sends g^{R_1} and a proof of its well-formness to P_1.
3. P_1, who is in possession of $\sigma_B(u, id_{P_1})$, sends to P_2 commitments $C_1^{(1)} = com(u; r_1')$, $C_2^{(1)} = com(id_{P_1}; r_2')$ and proves possession of the bank's signature on the openings of these commitments with proof $\pi_1^{(1)}$.
4. P_1 computes $t_1 = f(s', u)$ and $g^{D_1} = g^{id_{P_1} + R_1 t_1}$. P_1 sends to P_2 g^{D_1} and a proof $\pi_2^{(1)}$ of its well-formness.

5. P_1 transfers c to P_2. P_2 verifies all of the proofs in c and that R_0 was used in c and pays P_1. P_2 stores c and $c_1 = (r_1, R_1, C_1^{(1)}, C_2^{(1)}, g^{D_1}, \pi_1^{(1)}, \pi_2^{(1)})$.

Each additional transfer adds $c_i = (r_i, R_i, C_1^{(i)}, C_2^{(i)}, g^{D_i}, \pi_1^{(i)}, \pi_2^{(i)})$ to the coin. This information is used to recover the identity of P_i in case of double spending. Payment validation is performed as before, by decrypting the serial number s, but now only the last person in the chain is entitled to the payment.

During the cashing protocol, the bank will challenge the user on a newly chosen value of R computed in the same way as in the above additional transfer protocol. The user cashing the coin will evaluate the double spending equation $D = id + R \cdot f_u(s') \bmod q$ using her transfer credential $\sigma_B(u, id)$. This applies to every user, including the original payer (note that in payer's case, transfer is performed using the coin $\sigma_B(s, t, id)$, while the cashing protocol also requires from that user transfer credential $\sigma_B(u, id)$).

Cashing the Payment:

1. The banks sends to the claimant a randomly chosen value r.
2. The claimant presents a coin c and transfer values $c_1, c_2, \ldots, c_i$ for $i \geq 0$ (i.e., the claimant is participant P_{i+1} in the chain).
3. The bank first verifies all values and proofs in c and each c_i. It then retrieves the value R_i from c_i and computes $R_{i+1} = f(R_i, r)$.
4. Similar to the additional transfer protocol, the claimant proves possession of transfer credentials $\sigma_B(u, id_{P_{i+1}})$ and sends to the bank $g^{D_{i+1}}$ along with a proof of its well-formness.
5. The bank stores all values and issues to the claimant a payment for amount v recorded in c.

Double-spending now can happen in different ways (in all of the following cases double-spending occurs only if the coins in question become casheable): (1) as before, a payer can transfer his coin to more than one payee, (2) a payee can transfer her coin to more than one payee, or (3) a payee can transfer her coin to another payee and cash the coin herself. In case (1), the payer's identity is recovered as before using the serial number s and two different values for R_0 and B. The case of dishonest payees is handled as follows. Suppose payee P_i transfered c to P_{i+1} and P'_{i+1} (case (2) above), and now there are two chains $P_1, \ldots, P_i, P_{i+1}, \ldots, P_n$ and $P_1, \ldots, P_i, P'_{i+1}, \ldots, P'_m$. When P_n and P'_m cash the payments, the bank will be able to retrieve g^{D_i} and $g^{D'_i}$ computed using R_i and R'_i, respectively, and recover id_{P_i} in the same way. Case (3) above is handled in a similar way. The difference is that the bank obtains one value g^{D_i} that encodes id_{P_i} when user P_n cashes the payment. Another value $g^{D'_i}$ is obtained during the cashing protocol, where P_i is challenged on a new value R'_i.

6 Conclusions

Conditional e-payments, first introduced by Shi et al. [24], allow an electronic payment between two parties to be based on the outcome of a condition not

known in advance. This work formalizes the security of a conditional e-payment scheme and gives a solution based on CL-signatures. Compared to work of [24], our scheme completely avoids cut-and-choose techniques, thus exponentially reducing the protocols' overhead. We also eliminate the need for the bank to be involved in all conditional transfer protocols by making the protocol off-line.

Another significant contribution of this work is an extension that permits a conditional payment to be further transfered to other users. This extension builds a chain of users involved in a transfer of a particular coin, and each chain contains enough information to prevent double-spending by each user in the chain.

While our solutions satisfy the stated goals, there are a few directions that can be pursued in refining the properties we achieve. In particular, it would be interesting to see if the information used in the conditional payment protocol and the consecutive cashing protocols could be made unlinkable (this, however, is likely to require drastically different tools). Also, some of our constructions use commitments of the form g^x, which do not unconditionally hide x. Thus, it would be desirable to achieve stronger hiding properties.

References

1. R. Anand and C. Madhavan. An online, transferable e-cash payment system. In *International Conference on Progress in Cryptology*, volume 1977 of *LNCS*, pages 93–103, 2000.
2. M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Non-interactive anonymous credentials. In *To appear in Theory of Cryptography Conference (TCC'08)*, 2008.
3. F. Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology – EUROCRYPT'00*, volume 1807 of *LNCS*, pages 431–444, 2000.
4. E. Bresson and J. Stern. Proofs of knowledge for non-monotone discrete-log formulae and applications. In *Information Security Conference (ISC'02)*, volume 2433 of *LNCS*, pages 272–288, 2002.
5. J. Camenisch and I. Damgard. Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes. In *Advances in Cryptology – ASIACRYPT'00*, volume 1976 of *LNCS*, pages 331–345, 2000.
6. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *Advances in Cryptology – EUROCRYPT'05*, volume 3494 of *LNCS*, pages 302–321, 2005.
7. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Balancing accountability and privacy using e-cash. In *Security and Cryptography for Networks (SCN)*, 2006.
8. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *Security in Communication Networks (SCN)*, volume 2576 of *LNCS*, pages 268–289, 2002.
9. J. Camenisch and A. Lysyanskaya. Singature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO'04*, volume 3152 of *LNCS*, pages 56–72, 2004.
10. J. Camenisch, A. Lysyanskaya, and M. Meyerovich. Endorsed e-cash. In *EEE Security and Privacy*, 2007.

11. J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *LNCS*, pages 107–122, 1999.
12. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology – CRYPTO'97*, volume 1296 of *LNCS*, pages 410–424, 1997.
13. J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical Report No. 260, ETH Zurich, 1997.
14. D. Chaum, J.-H. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *Advances in Cryptology – EUROCRYPT'87*, volume 304 of *LNCS*, pages 127–141, 1988.
15. D. Chaum and T. Pedersen. Transferred cash grows in size. In *Advances in Cryptology – EUROCRYPT'92*, volume 658 of *LNCS*, pages 390–407, 1993.
16. R. Dingledine, N. Mathewson, and P. Syverson. TOR: The second generation onion router. In *USENIX Security Symposium*, 2004.
17. Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *Public Key Cryptography*, volume 3386 of *LNCS*, pages 416–431, 2005.
18. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identifcation and signature problems. In *Advances in Cryptology – CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, 1987.
19. J. Groth and A. Sahai. Efficient non-iteractive proof systems for bilinear groups. IACR ePrint Archive 2007/155. `http://eprint.iacr.org/2007/155`.
20. I. Jeong, D. Lee, and J. Lim. Efficient transferable cash with group signatures. In *International Conference on Information Security*, volume 2200 of *LNCS*, pages 462–474, 2001.
21. T. Okamoto and K. Ohta. One-time zero-knowledge authentications and their applications to untraceable electronic cash. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E81-A(1):2–10, 1998.
22. T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO'91*, volume 576 of *LNCS*, pages 129–140, 1991.
23. A. Saxena, B. Soh, and D. Zantidis. A digital cash protocol based on additive zero knowledge. In *Computational Science and Its Applications (ICCSA'05)*, volume 3482 of *LNCS*, pages 672–680, 2005.
24. L. Shi, B. Carbunar, and R. Sion. Conditional e-cash. In *Financial Cryptography and Data Security (FC'07)*, 2007.