Conditional E-Payments with Transferability

Bogdan Carbunar, Larry Shi, and Radu Sion [1]

Abstract

We introduce a novel *conditional* e-cash protocol allowing future anonymous cashing of bank-issued e-money only upon the satisfaction of an agreed-upon public condition. Payers are able to remunerate payees for services that depend on future, yet to be determined outcomes of events. Moreover, *payees are able to further transfer payments* to third parties. Once payment complete, any double-spending attempt by the payer will reveal its identity; no double-spending by any of payees in the payee transfer-chain is possible. Payers can not be linked to payees or to ongoing or past transactions. The flow of cash within the system is thus both correct and anonymous. We discuss several applications of conditional e-cash including online trading of financial securities, prediction markets, and betting systems.

1. Introduction

Electronic cash (e-cash) instruments allow digital payment for goods and services. Desirable properties of such protocols include: the ability to effect anonymous payments, the detection and prevention of malicious behavior (e.g., double spending), as well as the transactional consistency of the participants' financial state. A multitude of e-cash protocols have been proposed in the recent past. The main desiderata in such efforts has often been achieving digitally, levels of similarity and ease of use comparable to physical cash.

There are scenarios however, where basic e-cash properties are not sufficient. Here we consider the case of payments conditional on unknown future outcomes. In such settings, payers require the ability to anonymously remunerate payees for items that depend on future, yet to be determined outcomes of events. Prominent examples include trading of financial market instruments such as futures and securities (K. J. Arrow and G. Debreu, 1954; Balasko, 1986; Samuelson, 1947), and other online protocols involving deferred conditional payments such as betting.

Correctness assurances are essential. Payees need to be confident that payment *will* occur with certainty for favorable future event outcomes. Payers should be able to cash back un-cashed issued conditional payments for events with unfavorable outcomes. Overall monetary consistency needs to be preserved.

We note that trivial designs for such mechanisms can be envisioned, e.g., involving the e-cash issuing institution (i.e., bank) as a trusted arbitrator. Such assumptions, however, are rarely desirable. Requiring knowledge about the semantics of each and every considered future event at the bank is not scalable for even moderate transaction throughputs, considered events, and number of parties[2]. Moreover, an important concern in such scenarios is the privacy of participants. It is important to protect the privacy of interactions between payer and payee entities. Revealing identities should only be possible as a counter-incentive for faulty behavior (e.g., double spending) and specifically not during a correct run of the protocol.

Thus, one of the main challenges of a sound design is assuring participants' privacy while guaranteeing the conditional nature of payments. Payers and payees will naturally know each other, either by knowing each other's identity or at least by having access to a pre-authenticated channel through which to transfer public keys. No other party however should be able to associate them with each other and the conditional payments. While many existing e-cash protocols provide for participant anonymity, they cannot be directly deployed for payments of a conditional nature.

In this paper we introduce a new *conditional* e-cash protocol featuring the following properties. A payer can ask her bank to issue an anonymous payment token that can be cashed by any potential payee, *once* and if and only if a trusted publisher[3] will publish a specific secret (which only the publisher can do) in the future. In effect, payers are now able to remunerate payees (e.g., merchants) anonymously, for services that depend on future, yet to be determined outcomes of events. Moreover, the payee, can further transfer the payment to a third party with full assurances. Once payment complete, any double-spending attempt by the payer will reveal its identity. Moreover, no double-spending by any of the payees (in the payee-to-payee

[2]Additionally, arguably, very few banks would enter such an arbitration business.

[3]The publisher can be considered a "manager" of events – e.g., a stock market administrator, a race organizer.

transfer chain) is possible. Payers can not be linked to payees or to ongoing or past transactions. The flow of cash within the system is thus both correct and anonymous.

We explore a series of applications for conditional payments, including the online trading of securities, prediction markets, and online betting protocols.

The paper is organized as follows. We discuss the operational and adversarial models in Section 2 and related work in Section 3. We introduce and analyze the basic payment protocol in Section 4 and discuss transferrability in Section 5. We explore applications such as anonymous online betting in Section 6 and conclude in Section 7.

2. Model

A payer remunerates a payee by providing a payment token that can be activated and cashed at a specific bank, but only when a secret is published by a trusted publisher upon the completion of a certain agreed-upon event with a "favorable" outcome (e.g., stock price below given threshold, horse won race). Events with two possible outcomes will be considered ("favorable" – payment should be honored, and "unfavorable"). No other party but the publisher can generate the secret (under computational intractability assumptions). Without sacrificing generality, we will consider a single such event/secret combination, but there may be possibly many payees and payers exchanging conditional payments for a single event. The protocol guarantees the following:

P1. The bank is not able to associate previously issued conditional payments (to payers) with identities of principals (payers or payees) cashing them later.

P2. Double spending by both the payers and the payees is prevented. Moreover, if a payer re-uses the payment token for a different payee, its identity is revealed to the bank.

P3. The payer is able to cash back the payment token in the case of an unfavorable outcome.

P4. Once the payee accepts the conditional payment from the payer, she will be able to cash it in with high probability in the case of a favorable outcome, when the publisher publishes the associated enabling secret. In this case, if the payer attempts to spend the payment token the payer's identity will be revealed to the bank (this is discussed in P2).

P5. The publisher cannot infer any information about the existence of payer-payee-bank interactions solely through the protocol.

P6. The bank cannot infer any event-specific details (other than its outcome).

P7. Neither the payer nor the payee should be able to prove to outside parties that they interacted in a conditional payment protocol (deniability).

2.1. Operational Model

Let A be the payer, C the payee, B the bank and T the trusted publisher. Factoring large composite numbers is hard. There exists a PKI infrastructure based on RSA. For any party X, we denote by $id(X)$ its identity, N_X its public RSA modulus, e_X its public key and d_X its private key.

In our work we use the concept of a network anonymizer, or mix network (Dingledine et al., 2004). Mix networks consist of serially composed servers, each transforming a set of input messages into a permuted and re-encrypted set of output elements. Mix networks ensure that as long as an adversary is unable to corrupt more than a certain number of servers, they are not able to link the communicating parties. This includes the fact that the destination cannot infer the identity of the communication's source. Chaum (Chaum, 1981) was the first to formalize the concept of mix networks, while numerous other works (Abe, 1998; Park et al., 1994; Dingledine et al., 2004), have provided various solutions. We assume such tools can be deployed by both A and C to communicate with B and with each other. Let Mix be a notation for such an anonymizer.

Whenever possible point-to-point communication will be encrypted semantically secure [4], including links passing through an anonymizer towards the bank. These will be encrypted with no forward security by using a session key generated by the anonymous party (e.g., C, when communicating with B). The bank B manages client accounts and assists clients by generating or cashing traditional and conditional e-cash payments.

Let b denote the public "name" of the considered future event. Let t be the corresponding secret published by T in the case of a favorable (for payment) outcome. Without loss of generality we will consider b to be a large prime number, and $t = b^{-1} \ mod \ \phi(N_T)$, where $\phi()$ is Euler's totient (this is discussed further in Section 4.3). If the event's outcome is not favorable, T

[4] With keys being generated using authenticated DH or equivalent.

is trusted to immediately discard any information that could enable other parties to reconstruct t or portions thereof. We stress it is important for T to not collude with the payee to reveal the payer's identity by publishing t and allowing C to cause a payer double-spending condition. The publishing process of T could be as simple as maintaining an authenticated website. For scalability, outside of the publishing process, no interaction between T and other participants is required by the protocol.

2.2. Adversary

As discussed above we are concerned with a computationally bounded adversary. Because the message exchanges are encrypted, and the protocol only uses anonymizers when no authentication is required, we will consider here mainly the insider threat. Both the bank and the publisher should not be able to infer any additional information about ongoing or past conditional payment transactions. Specifically, without their direct cooperation, A and C should not be identifiable as conditional payment partners. Additionally, no subset of participants should be able to collude and violate any of the properties above.

2.3. Crypto Tools

For completeness, we will briefly discuss blind signature protocols.

Let a party A engage in a blind signature protocol with B (B is the signing party). At the end of a correct run of the protocol, A will be in the possession of a "well-formed" (e.g., "\$10") message signed by B, such that B does not know the message contents but is (sufficiently) confident of its "well-formed"-ness. It can be considered that B's signature semantics in fact speak only about the fact that the message is "well formed". Thus, the "blind" signature should not be interpreted to mean anything else. We now overview an instance, namely the **cut-and-choose** protocol (Chaum, 1982) (Chaum, 1985) (Chaum, 1988) (Chaum et al., 1990).

Let $S_B(M)$ denote B's signature on message M. A generates n "well-formed" messages $\{M_1, \ldots, M_n\}$, such that any of them signed by B (i.e., any of $\{S_B(M_1), \ldots, S_B(M_n)\}$) would satisfy A as an end-result. A "blinds" all n messages with different blinding factors and sends them to B. A blinded message cannot be read unless the corresponding blinding factor is known. B requests $n-1$ randomly chosen blinding factors from A. It un-blinds the corresponding messages and verifies that they are "well formed". B is now convinced that with probability $1 - 1/n$, the remaining message is also

well formed. By making n arbitrarily high, this confidence can also be made sufficiently high. B then signs the remaining blinded message M_j and sends it back to A, who simply un-blinds it. The blinding mechanism is designed such that a message first blinded by A, then signed by B, can be transformed into its simple signed (un-blinded) corresponding message $S_B(M_j)$ by A, knowing the blinding factor. We say that B blindly signed M_j for A.

For illustration purposes, we consider B's signature to be simple RSA exponentiation with private key d_B. The blinding mechanism of a message M can then be $M \times s^{e_B}$. The corresponding un-blinding process is simply division by the blinding factor s. We note that blind signature protocols can be run through anonymizers (with simple precautions). Moreover, also for illustration purposes we use the cut-and-choose and blinding protocol combination of Chaum et al. (Chaum et al., 1990). However, for real life implementations, more efficient blind signature protocols (e.g., the work of Brands (Brands, 1993)) can be used.

3. Related Work

E-cash. The use of blind signatures and of the cut-and-choose protocol for providing untraceable electronic cash payments was proposed in (Chaum, 1982) (Chaum, 1985) (Chaum, 1988) (Chaum et al., 1990). The problem of transferable e-cash was analytically studied first by Chaum and Pedersen (Chaum and Pedersen, 1992). The work of Brands (Brands, 1993) proposes a primitive called *restrictive blind signatures* to replace the high cost of blind signatures that use the cut-and-choose technique. While in our work we have used the latter technique to illustrate our protocol, for real implementations, Brands's solution could also be employed.

Franklin and Yung (Franklin and Yung, 1993) propose the use of a trusted entity (trustee) that collaborates with the bank during withdrawal and deposit to provide a computation efficient on-line cash system. Trustees (either on-line or off-line) were proposed to provide variable degrees of anonymity for e-cash (M. et al., 1995) (Davida et al., 1997) (Camenisch et al., 1996) (Frankel et al., 1996). Stadler et al. (M. et al., 1995) introduced the notion of coin tracing and introduced several tracing mechanisms, requiring the trustee to be on-line at withdrawal. Camenisch et al. (Camenisch et al., 1996), Frankel et al. (Frankel et al., 1996) and Davida et al. (Davida et al., 1997) proposed payer and coin tracing mechanisms using off-line trustees. In contrast, in our work the requirement of preserving the payer and payee anonymity is

essential. Moreover, our work also requires the bank to be unable to link the payer and payee even when colluding with one of them.

Simon (Simon, 1996) proposes a simple e-cash protocol in a network where anonymous communication is possible. The payer generates the e-cash by having the bank sign $f(x)$ where x is the payer's secret and f is a one-way function. The e-cash can be transferred by revealing x to the payee. The payee can then either cash the money with the bank or further transfer it by providing the bank with x and asking it to sign $f(y)$ for which it knows y. If the communication between the payee and the bank is anonymous, the payee remains anonymous and can transfer the money further. The bank can link the start and end points of a transfer chain, however, for long chains this information may be meaningless. Moreover, the end point of a transfer chain may repeat this protocol with itself, to artificially increase the length of the chain. Even though we also require the use of anonymizers, the solution of (Simon, 1996) does not provide support for conditional transfers. Even if conditional transfers would be provided, the payer could easily spend the e-cash transferred to the payee before the condition is satisfied – as the e-cash does not encode any information about the payer for anonymity reasons.

Camenish et al. (Camenisch et al., 2005) propose an efficient off-line anonymous e-cash protocol that prevents linking payers to payees, even when the bank and all other users collude. In addition, the solution allows a user to withdraw not one, but 2^l coins, where each coin can be spent anonymously. The result is interesting in that the storage required for the 2^l coins is only $O(l+k)$, where k is a security parameter. Moreover the solution is extended to allow the traceability of the coins without a trusted third party. That is, once a user double spends any of its coins, all its spendings, of any of the 2^l coins, can be traced. The coin storage cost of this extension is only $O(lk)$. Note however that unlike our work, this solution does not allow coins to be spent conditionally, based on desired events.

Shi et. al (Shi et al., 2007) proposed the initial conditional e-cash solution discussed in this paper. This paper extends our previous work with (i) an offline solution for the initial problem (see Section 4.6) and with (ii) a transferability property (see Section 5). For the latter contribution, our extensions allow a payee to anonymously transfer a received payment, before the condition's outcome is published, while still satisfying the properties discussed in Section 2.

Camenish et al. (Camenisch et al., 2007) proposed the concept of "Endorsed E-Cash", which is related to our notion of conditional e-cash. The

proposed endorsed e-coins are composed of a lightweight endorsement x and the rest of the coin which is meaningless without x. They allow users to exchange e-cash by exchanging endorsements. The use of endorsed e-cash is exemplified in two scenarios. First, an optimistic and unlinkable fair exchange of e-cash for digital goods and services is proposed and second, onion routing with incentives and accountability for the routers is provided. We note that this work considers only a weaker form of anonymity, specifically not considering payees. In the scenarios considered in our work, e.g. anonymous services, online betting etc, payees also need to preserve their privacy. In our work we provide payer and payee privacy as well as prevent linking payers to payees.

Blanton (Blanton, 2008) uses Camenish and Lysyanskaya (Camenisch and Lysyanskaya, 2004) type signatures to provide a new conditional e-cash scheme that improves on the performance of our online solution. However, the solution proposed does not allows for transferability of e-cash tokens.

Carbunar and Tripunitara (Carbunar and Tripunitara, 2008, 2010) have extended our work by providing a secure payment component for distributed computation markets (e.g., cloud and volunteer computing environments). Existing solutions in this area provided computation outsourcers with the ability to verify the correctness and completeness of returned computation results. This work complements such solutions by simultaneously allowing workers to verify the correctness of received payments, while preventing them from caching the payments before completing the alloted computation. Thus, the work from (Carbunar and Tripunitara, 2008, 2010) can be viewed as an implementation of a conditional payment, where the condition consists of completing a job.

Time release encryption. Dodis and Yum (Dodis and Yum, 2005) introduce a novel problem called the *time capsule signature*. It allows for the construction of a signature that becomes valid at a time in the future when a trusted third party publishes a trapdoor associated with the current time. The time capsule signature allows the recipient of the signature to immediately verify its validity. Moreover, the third party has no interaction with the generator or recipient of the signature. It may seem possible to use the time capsule signature to solve the conditional payment problem. The payer could ask the bank to generate a time capsule signature on a blinded e-cash such that the capsule can be removed only if a certain event occurs. Besides the technical difficulty of the payer un-blinding the time capsule, this solution would require the bank's knowledge of the event, its publishing pro-

cedure and ultimately the identity of the publishing institution. However, for privacy reasons, the conditional payment problem requires the decoupling of the publishing institution from all other participants. In particular, the bank's operation should be oblivious of the nature of the event determining the condition.

Blake and Chan (Blake and Chan, 2005) propose a protocol for transferring time-encrypted messages between users. A message becomes valid only after a trusted server publishes a signed piece of information on a specific time value. Their solution requires no interaction between the trusted server and the users and also preserves the user's privacy from the server. Cathalo et al. (Cathalo et al., 2005) propose a more efficient solution for this problem, that also improves the user's anonymity. However, none of these schemes allows the receiver of a timed release message to verify its validity before release time, making them unsuitable for conditional e-cash transfers.

4. Conditional Anonymous Payments

The solution is composed of a set of logical sub-components: the generation of conditional payments, the validated transfer of the payments from the payer to the payee, and their spending by the payee in the case of a successful event outcome, or the cashing of the un-spent payments by the payer otherwise (see Figure 1). All the above will also be designed to prevent double spending by both the payer and the payee.

As mentioned in Section 2.1 for any party X, we denote by $id(X)$ its identity, N_X its public RSA modulus, e_X its public key and d_X its private key. Let b denote the public "name" of the considered future event. Let t be the corresponding secret published by T in the case of a favorable (for payment) outcome. Without loss of generality we will consider b to be a large prime number, and $t = b^{-1} \ mod \ \phi(N_T)$, where $\phi()$ is Euler's totient (this is discussed further in Section 4.3).

In the following we detail each of these components.

4.1. Payment Generation (PG)

Let n_1 and n_2 be security parameters. Assuming the payer A holds an account with the bank B, to generate the conditional payment, A will contact the bank B as follows (see Figure 2).

A generates $2n_1$ random numbers $X_1, .., X_{n_1}$ and $R_1, .., R_{n_1}$. Using a standard secret sharing algorithm (Shamir, 1979), A constructs n_2 shares

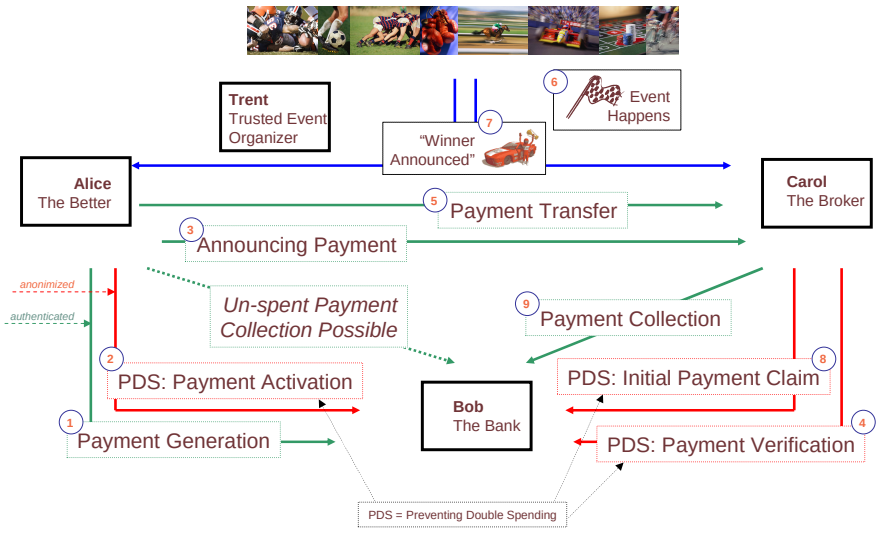

Figure 1: Solution Overview. Arrows with boxes indicate protocol steps, with the direction of the arrow indicating the protocol's initiator and target parties.

for each of the values $X_i \oplus id(A)$, for $i = 1..n_1$. We denote the j-th share corresponding to $X_i \oplus id(A)$ by $share_{ij}$ for $j = 1..n_2$. For any $X_i \oplus id(A)$, all its n_2 shares are required for its correct reconstruction.

A then constructs n_1 blocks, each of $n_2 + 1$ fields. The i-th block consists of

$$m_{iL} = [X_i, R_i, v, \text{“}left\text{”}], \quad m_{ijR} = [share_{ij}, R_i, v, \text{“}right\text{”}] \tag{1}$$

where v represents the value and currency of the payment (i.e. \$1). “$left$” and “$right$” are text messages used to differentiate between the m_{iL} value and the m_{ijR} shares.

Next, A asks the bank to blindly sign one of the n_1 message pairs using the cut-and-choose protocol discussed in Section 2.3. In this specific case however, the bank signature consists of a signature on both m_{iL}, and all m_{ijR} as well as on each and every $share_{ij}$ in m_{ijR}. The bank will do so after

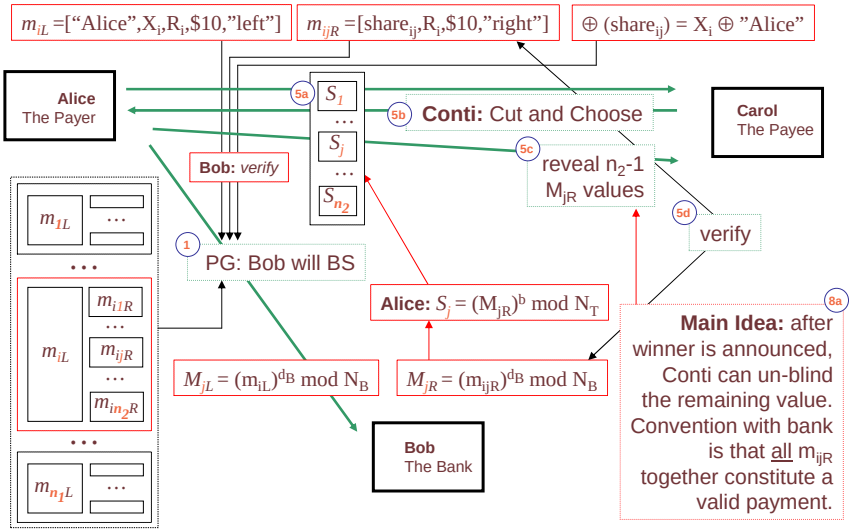

Figure 2: Overview of Payment Generation and Conditional Transfer steps.

verifying “well formed”-ness of $n_1 - 1$ random pairs as well as their associated shares. Specifically, the bank will verify

- that after using each set of n_2 shares in the $n_1 - 1$ “right” messages m_{ijR} to reconstruct the corresponding $X_i \oplus id(A)$ values, XOR-ing these reconstructed values $X_i \oplus id(A)$ with the first fields of m_{iL} yield indeed $id(A)$.
- that the second field of m_{iL} is equal to the second field of m_{ijR}. This value, R_i associates the messages later on.
- the correctness of the enclosed currency value (v).

If any check fails, B aborts the protocol. Otherwise, A's account is debited in an amount of v and A is able to retrieve (after un-blinding) the following payment document (signed by the bank B):

$$M_L = m_{lL}^{d_B} \ mod \ N_B, \quad M_{jR} = m_{ljR}^{d_B} \ mod \ N_B \tag{2}$$

where $j = 1..n_2$ and $l \in [1, n_1]$ was randomly chosen by B.

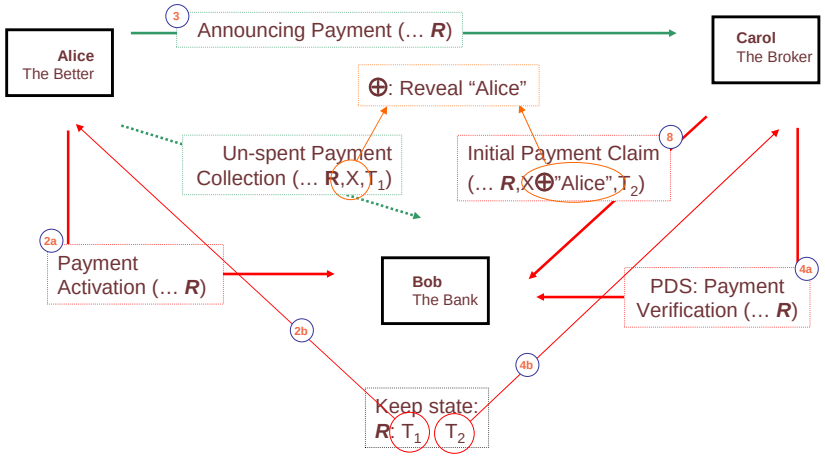

Figure 3: Double spending prevention using activation tokens.

Intuitively, M_L can be later used by A to cash any un-spent payment in the case of an un-successful event outcome (see Section 4.4), while the n_2 bank signed e-cash shares, M_{jR}, can be used by A for payments to potential payees such as C (see Section 4.3).

4.2. Preventing Double Spending (PDS)

Before we proceed with describing the actual transfer of these shares to payees, we will first discuss a simple token attribution mechanism designed as one of the tools we will use to prevent the payer from double spending (see Figure 3). Specifically, A will be prevented from transferring the payment to more than one payee. Moreover, at the completion of this step, at most two participants, one being A, will be able to cash the payment.

To achieve this, B will issue two unique "use tokens" for each signed payment (identified so anonymously by its unique R_l value). Each of these tokens will be issued on-demand, in an online interactive protocol, through an anonymizer. Specifically, before interacting with C but after retrieving the signed payment document $\{M_L, M_{jR}\}$ from B, A will use the anonymizer Mix to send B the currency amount v and the R_l value occurring both in M_L and M_{jR}, $j = 1..n_2$. B will respond with a fresh random token $token_L$. B will also store an association between R_l and this token $R_l : \{token_L\}$ for

future reference. We call the payment "activated" once this happens. If B has already seen R_l it ignores the message.

Before transferring the actual payment document, A sends R_l and v to C. C then forwards R_l anonymously to B who proceeds as follows:

- if B does not find any record of R_l it notifies C and then simply ignores the message as the payment has not been activated yet.
- otherwise, if R_l is associated with a *single* token $token_L$, B generates a new random token $token_R$ (payment cashing token), associates it also with R_l ($R_l : \{token_L, token_R\}$), and sends it to C. We now call the payment "cash-able". It is important to note that only C and B know $token_R$. C will use $token_R$ later to cash the payment upon a successful event outcome, as will be discussed later.
- if B already stores *two* tokens associated with R_l, it notifies C, who in turn then aborts the protocol, knowing that A attempts to double spend.

4.3. Conditional Transfer (CT)

The PDS protocol above allows C to assert the fact that the payment that will follow from A has been activated and has not yet been spent. In this section we discuss achieving the "conditional" properties of the protocol. We introduce here a randomized probabilistic solution (see Figure 2).

The main idea is for A to generate a quantity that can both (i) convince C to accept this payment because it is indeed valid cash-able money signed by B, (ii) allow its cashing only if t is published by T. A uses event b and T's modulus N_T (see Section 2) to blind each $M_{jR} = m_{ljR}^{d_B} \; mod \; N_B$, $j = 1..n_2$, separately, by computing

$$S_j = M_{jR}^b \; mod \; N_T.$$

A and C then engage in a cut-and-choose protocol (see Section 2.3) through which C becomes convinced that with $1 - 1/n_2$ probability, all of the S_j values are indeed well formed and signed by B, as follows.

A sends all such S_j values to C, along with the R_l value and currency amount v. C selects a random one of them (e.g., S_u) and asks A to prove that all the remaining ones are indeed valid M_{jR} messages. To do so, A sends

C all $M_{jR} = m_{ljR}^{d_B} \; mod \; N_B$ values for all $j \in [1, n_2] \setminus \{u\}$ and C can verify that indeed $S_j = M_{jR}^b \; mod \; N_T$ for these values.

At this point, C will verify the "well-formed"-ness of all revealed M_{jR} values. After removing B's signature from M_{jR}, C verifies that the fourth field of m_{ljR} equals the constant string "right" and that the second and third fields equal the R_l and v values previously sent by A for the present transaction. This verification prevents A from re-using shares from different protocol instances. C also verifies that there are no duplicates among the first fields ($share_{lj}$) of the $n_2 - 1$ m_{ljR} values recovered. As a reminder, all n_2 shares are required for the reconstruction of the corresponding $X_i \oplus id(A)$ value later on. If any of these checks fails, C aborts the protocol.

Later, for a successful event outcome, T will publish

$$t = b^{-1} \; mod \; \phi(N_T)$$

Since b is prime (see Section 2), it has an inverse mod $\phi(N_T)$. Only T can compute this inverse, knowing the factorization of N_T. Using t, C can retrieve the missing M_{uR} value as

$$M_{uR} = S_u^t \; mod \; N_T$$

By removing B's signature from M_{uR}, C obtains the last unknown share, $share_{lu}$, allowing it to construct the secret $X_l \oplus id(A)$.

Note that given b, once $t = b^{-1} \; mod \; \phi(N_T)$ is published, N_T can be factored. Thus, a new modulus N_T needs to be created for each event. To avoid this, instead of RSA, an identity based encryption scheme (Boneh and Franklin, 2001) can be used. Then, the condition b is the identity (public) and t is the private key corresponding to b. t can then safely published by T in case of a favorable outcome.

4.4. Spending The Money (SM)

In the case of a favorable event outcome, C should be able to interact with B and get her account credited appropriately. To achieve this, we propose a three-stage protocol. In the first stage C contacts B anonymously and provides proof of credit. The proof of credit consists of the secret $X_l \oplus id(A)$, the value R_l and credit value v, along with all the values M_{jR}, $j = 1..n_2$. As mentioned before, each value M_{jR} is signed by the bank and contains one share of $X_l \oplus id(A)$.

In the second stage, also performed over the anonymous channel, C and B engage in a blind signature protocol (see Section 2.3) in which B blindly signs an un-traceable piece of currency of equivalent value to the credit that was proved in the first stage. This is performed as follows. C first generates n_1 pairs of random values U_i, SN_i, for $i = 1..n_1$. C then sends to B n_1 obfuscated values of format $U_i^{e_B}(SN_i, v)\ mod\ N_B$. Note that the value of v needs to coincide with the value of v from the first step. B chooses randomly $u \in [1..n_1]$ and asks C to reveal all pairs U_i, SN_i, $i \in [0.., n_1] - u$. B then verifies that the revealed pairs indeed reconstruct the previously received obfuscated values. Then, B signs the uth value, producing

$$S = U_u(SN_u, v)^{d_B}\ mod\ N_B$$

C uses S to retrieve the un-traceable piece of currency, $(SN_u, v)^{d_B}\ mod\ N_B$.

In the final stage, the payee C directly contacts the bank B through an authenticated channel and exchanges this piece of currency for credit to her account. For an unfavorable event outcome, to cash an un-spent payment, A proceeds identically.

Note that B is unable to link the user that has performed the first two steps over the anonymous channel to C. This is because B blindly produces the value $S = U_u(SN_u, v)^{d_B}\ mod\ N_B$, without seeing the underlying currency $(SN_u, v)^{d_B}\ mod\ N_B$. This in effect disconnects the user involved in a conditional transfer from the user that deposits the cash in its account.

We note that, technically, the three-stage anonymous protocol is apparently superfluous here for purposes of providing anonymity, as this has already been ensured by previous anonymization and the lack of any information about A's identity in the proof of credit. Nevertheless, we chose to discuss it here for ease of presentation. Its purpose will become apparent later when we discuss specific applications of conditional payments such as online betting.

We now detail the above. C uses the anonymizer Mix to send to B the message

$$token_R, M_{jR}, j = 1..n_2,$$

containing the n_2 shares recovered from A and T. Similar to C (see Section 4.3), B immediately verifies the validity of each share M_{jR}. If at least one share does not verify, B aborts the protocol. Otherwise, it uses the shares to recover $X_l \oplus id(A)$. B then verifies that $token_R$ is the *second* token associated

with the R_l value contained in all M_{jR} shares. If the check fails, B aborts the protocol.

Next, B investigates potential double spending. If the M_{jR} shares have been previously spent, it simply drops the message, knowing that C double spends. If the left part of the payment, M_L has been spent (by A), B can immediately recover A's identity by computing the XOR of the first field of the corresponding m_{lL}, X_l with $X_l \oplus id(A)$.

At this point, B has proof to believe that C is entitled to a credit equal to the v value stored in the third field of M_{jR}. Now C and B can anonymously engage in a blind signature protocol in which B blindly signs an un-traceable temporary piece of uniquely identifiable currency of equivalent value to this credit.

Finally, the payee C directly contacts the bank B through an authenticated channel and exchanges this piece of currency for credit to her account. B will first verify if this currency has been already spent, credit C's account, and store the unique identifier of the currency for future double spending detection.

4.5. Analysis

In this section we informally discuss the security properties of the above protocol.

Double spending (P2). The payer could try to double spend during the PDS step by registering with the bank the same e-cash under different R_l values and transferring each value to a different payee. This is prevented during the CT step, by having the payee verify that the R_l value encoded in the e-cash matches the R_l value received during the PDS step.

Alternately, during the SM step, the payer could try to spend her e-cash (using M_L) even in the case of a favorable outcome published by T. However, once the payee performs her SM step, the payer's identity will be immediately revealed. The payer could also try to recover the e-cash she sends to the payee and then attempt to spend it before the payee has a chance to do it. This would be a good strategy since in case of double spending the payer's identity would not be revealed. To succeed in this attack, the payer has to obtain the $token_R$ value associated with the unique R_l of the e-cash, that is being shared only between the payee and the bank. Since the interaction between the payee and the bank is encrypted, this is impossible. If the payer, before transferring the e-cash to the payee, impersonates a possible payee and goes to the bank to obtain a $token_R$ value, during its initial interaction with the

bank the payee will know that the payer has tried to double spend and abort the protocol. This is because the bank will report to the payee the fact that two values, $token_L$ and $token_R$ have already been generated for the R_l value the payee has received from the payer (see Section 4.2).

The payee cannot double spend, since both her shares (M_{jR}) and the unique identifier generated at the end of the SM step (see Section 4.4) are recorded by the bank. The payer and the payee could try to collaborate in order to double spend e-cash without having their identities revealed. This is prevented by the fact that the e-cash generated during the PG step ensures w.h.p. $(1-1/n_1)$ the fact that spending both M_L and the M_{jR} shares reveals the payer's identity. Moreover, both M_L and the M_{jR} shares can only be spent once.

Probabilistic Payment or Rollback (P3,P4). During the cut-and-choose sequence of the CT step, the payee receives $n_2 - 1$ shares of its choice of the payee's e-cash. If event b occurs and the corresponding t value is published by T, the payee can recover the missing share and spend the e-cash. If event b does not occur, the payer is certain that the payee is unable to recover the e-cash. The payer can then safely cash back its payment, without fear of double spending. At this point T is trusted to never reveal the factoring of the current N_T value. We stressed before the existence of a collusion vulnerability: T can collude with the payee to reveal the payer's identity by publishing t and allowing C to cause a payer double-spending condition.

Note that due to the use of a cut-and-choose protocol to generate a payment, A has probability $1/n_1$ to make the payment unusable. A system-wide n_1 parameter may not work well, as transactions may have different risk levels. We address this issue by proposing the use of different levels of assurance. For instance, for high-risk transactions (e.g., involving larger sums of money), the value of n_1 could be also set to a sufficiently large value to ensure sufficiently small cheating probabilities. We also propose to use only a small, fixed number of mappings of risk levels to n_1 values, since otherwise the bank can link transactions (a payment of \$19.37 with $n_1 = 171$ can easily link the user performing a deposit to the user that earlier withdrew the money). This extension can be implemented by requiring the payer A to choose a risk level and value of n_1 during the payment generation step. Then, if during the conditional transfer step the payee C is not comfortable with the assurance level provided by the value of n_1, it can abort the protocol.

Un-linkability and deniability (P1,P5,P7). The payer obtains the payment signed by the bank, containing a R_l value that is unknown to the bank. Moreover, the payee cannot prove payment origin to other parties as no non-repudiable identification tokens are revealed in any steps outside of double spending. This prevents the bank from colluding with payees to trace payments to their payer.

The payer could collaborate with the bank and attempt to reveal the identity of the payee. To achieve this, the payer could spend her e-cash (M_L) or the payee's e-cash (the M_{jR} shares) in order to signal the bank the moment when her e-cash will be spent by the payee. However, before spending the e-cash in person, the payee performs two additional stages, both through an anonymizer (see Section 4.4). The second additional stage generates the anonymous e-cash the payee will spend then in person.

Since the publisher does not directly interact with any participants, except possibly for publishing event outcomes, property P5 is trivially satisfied.

Event Privacy (P6). P6 is satisfied by construction. None of the public tokens issued by T are linked to any event-specific details other than its outcome. This ensures that events can be kept and run privately if so desired. By only observing the published tokens and transaction transcripts third parties are not offered additional information about any event details (e.g., such as "Horse X won race Z.").

The Use of Anonymizers. Note that our solution employs a mix network twice. First, this occurs during the prevention of double spending step, when the payer registers a unique random number with a payment and the payee uses the random number to verify the absence of double spending. Then, by using an anonymizer, the payer and payee effectively prevent the bank from linking them to the payment and to each other. Second, the payee uses an anonymizer when it deposits a payment with the bank. The deposit operation requires the payee to present a payment and an account number to the bank. In order to prevent the bank from trivially learning an association between the payee and the payment, the deposit operation takes place in two steps. First, the payee uses the anonymizer to present the payment and exchange it for a blindly signed (using cut-and-choose) equivalent new payment. Then, the payee contacts the bank over an authenticated channel and deposits the new payment into its account. Thus, the anonymizer disconnects the payee from the original payment.

4.6. Offline Version

The previously described PDS module requires the existence of an online entity, the bank B for instance, to prevent A from spending a payment with multiple payees. To understand this issue, assume a malicious user A that spends the same currency – signed by B but using different conditions – with users C and D simultaneously. If later, both conditions come up favorably for C and D, they are able to retrieve the e-cash. When both go to the bank, with the same e-cash instance, we would like to be able to retrieve A's identity. Note that the previous, online solution detects double spending and allows the bank to retrieve A's identity after sending e-cash to C, only if both A and C try to spend the money.

We now briefly describe an offline solution to this problem, using a construct borrowed from the work of Chaum et al. (Chaum et al., 1990). The solution works in the following manner. During the payment generation stage, A generates n_1 payment strings (SN_i, v), $i = 1..n_1$. A uses then a $(2n_2,n_2)$ threshold splitting scheme to split each payment into $2n_2$ shares, such that any but not less than n_2 of them are sufficient to re-construct the original payment. For each such payment string (SN_i, v), A produces the following n_2 blocks, each having a "left" and a "right" part,

$$m_{ijL} = [j, X_{ij}, share_{ijL}, SN_i, v, \text{“}left\text{”}],$$
$$m_{ijR} = [j, X_{ij} \oplus id(A), share_{ijR}, SN_i, v, \text{“}right\text{”}],$$

where $j = 1..n_2$ is also used to index the blocks. Using the blinding and cut-and-choose processes previously described, A convinces B to blindly sign the $2n_2$ blocks corresponding to one of the payment strings. Let that be the lth payment string, $l \in \{1..n_1\}$ and let M_{ljL} and M_{ljR} denote the corresponding signed messages, $j = 1..n_2$. The semantics of these shares is that if any participant can present to B n_2 shares of the same payment string, each validly signed by B, the bank will convert the shares into v currency.

Similar to the initial solution, A uses the event b to encrypt each of the $2n_2$ retrieved shares signed by B, producing messages S_{ljL} and S_{ljR}, $j = 1..n_2$. A then sends to C only n_2 of the S_{ljL} and S_{ljR} values, in the following way. First, C sends to A n_2 challenge bits. If the jth bit is 0, A sends to C the value S_{ljL}, otherwise it sends the value S_{ljR}, $j = 1..n_2$. C then asks A to reveal n_2-1 of these shares, randomly chosen. That is, C reveals the corresponding M_{ljL} or M_{ljR} value, $j = 1..n_2$.

When C receives these values, it first verifies that the S values where produced from the corresponding M values using the pre-agreed upon event b. It then verifies B's signature on all the revealed M values, producing the corresponding m values. C then verifies that each revealed m value is indeed the "left" or "right" half, according to the challenge bits. C then verifies that each m value has the correct index, j, and that all m values have the same serial number SN_i and are for the same currency value v.

At this stage C is confident that if the event b occurs, it will be able to retrieve the last unrevealed share and be able to cash them using a protocol similar to the one presented in Section 4.4. However, when C brings the payment shares to the bank, B stores them, under the serial number SN_l.

In case the event b does not occur, A can go to the bank and cash its payment using a procedure similar to the one used by C. That is, B generates n_2 challenge bits, for which A has to produce either the left M_{ljL} or the right M_{ljR} halves of the payment blocks. B verifies its signature on each of them and verifies the well-formedness of each (the index j, the serial number SN_l and the currency amount). Then, B looks to see if the serial number SN_l was already spent. If this is the case, B looks at the values stored with it. For each index $j = 1..n_2$ it looks to see if it has both a "left" and a "right" half. If it does, it uses the X_{lj} and $X_{lj} \oplus id(A)$ fields to reveal A's identity. Since the challenge bits of B and C are very likely to differ in at least one bit, A's identity will very likely be revealed. This is correct, since if C has already cached its copy of the payment it means that the event b occurred. Thus, A should not have tried to spend its own payment copy.

Note that in this solution, after spending a payment with C, A could impersonate C with the bank and try to redeem the payment before the event b comes up. This is easy to do since the redemption step is done over an anonymous channel and A already knows which shares it has revealed to C. Then, it can redeem the payment by revealing to B exactly the shares it has revealed to C. If later event b occurs, C will try to reveal the same shares to B. Thus A's identity will not be discovered even though double spending is detected.

We prevent this by extending the above protocol with an oblivious transfer step. That is, A produces $2n_2$ pairs of format $[(S_{ljL}, E_{k_j}(m_{ljL})),(S_{ljR}, E_{k_j}(m_{ljR}))]$, $j = 1..n_2$, where k_j is a symmetric encryption key. Using the above solution, A sends these pairs to C, then A and C engage in a 1-out-of-2 oblivious transfer for each of the n_2 pairs. This allows C to retrieve exactly one half of each pair, that is, either the left half, $(S_{ljL}, E_{k_j}(m_{ljL}))$, or the right half,

$(S_{ljR}, E_{k_j}(m_{ljR})$, without A knowing which. Then, C specifies which $n_2 - 1$ shares it wants revealed and A sends the corresponding keys k_j. Thus, A will not know which shares C has. If A wants to impersonate C and redeem its payment early, it will have to *guess* the shares C has and its guess will be correct with probability at most $1/2^{n_2}$.

5. E-Cash Transferability

We discussed above a solution providing single-hop e-cash payments. We now turn to the issue of multi-hop transfers. This is important in a multitude of scenarios, e.g., in financial securities/options trading where securities and options are subject to multiple sell-buy cycles before maturation.

We introduce a mechanism that allows C to anonymously transfer the e-cash payment from A [5] to a participant D, while still satisfying the properties discussed in section 2. In particular, according to property **P4** described in Section 2, our solution allows C to convince D that in the event of a favorable outcome, the shares will construct w.h.p. valid e-cash.

To this end, we extend the operational model of section 2.1 as follows. Let k be a security parameter. Its purpose will become apparent in section 5.5 when we discuss A-with-C collusion attacks. The bank B is assumed to possess an RSA key pair in addition to the $K = (e_B, d_B, N_B)$ key-pair used to authenticate e-cash. Let this key pair be $\bar{K} = (\bar{e_B},\bar{d_B},\bar{N_B})$. Intuitively, B will use this key-pair to sign "having seen" a set of blinded payment shares – this will then convince D of the fairness of transfer.

The new protocol modules are discussed below, extending the previous online solution. For brevity of exposition, only the differences in functionality are outlined.

5.1. Payment Generation

In the payment generation (PG) phase (see equation (1) in section 4.1) A deploys a $(n_2 + k,n_2)$ secret splitting algorithm to construct $n_2 + k$ shares $share_{ij}$ for each of the n_1 values $X_i \oplus id(A)$, $i = 1..n_1, j = 1, .., n_2$, such that *no less than* n_2 of the shares can reconstruct $X_i \oplus id(A)$. Next, for each $X_i \oplus id(A)$, A generates $n_2 + k$ values

$$m_{ijR} = [share_{ij}, R_i, v, \text{“}right''], j = 1..(n_2 + k).$$

[5] Before the condition's outcome is published.

The payment generation protocol then continues briefly as in section 4.1, allowing B to verify the well-formedness of $n_1 - 1$ of the n_1 $X_i \oplus id(A)$ values. Eventually, B will blindly sign the remaining value and return M_L and M_{jR} (see equation (2)), where each m_{ljR} value is signed by B with key K:

$$M_{jR} = m_{ljR}^{d_B} \; mod \; N_B, j = 1..(n_2 + k)$$

A then encrypts each M_{jR} with b, yielding values $S_j = M_{jR}^b \; mod \; N_T$. Now however, instead of sending the S_j's to the payee C, A sends all the S_j values in clear to B, through the Mix anonymizer. Since B has blindly signed the m_{ljR} values, it was never able to see the resulting M_{jR}s. Thus, even if B would know all possible events b it cannot link the S_j values to the identity of A. B generates a new random number R_B and uses the received $n_2 + k$ values and the private exponent of the key pair $\bar{K}$ to produce the following certification message

$$Cert = H(S_1, S_2.., S_{n_2+k}, R_B)^{\bar{d_B}} \; mod \; \bar{N_B}.$$

That is, B signs a hash of a sequenced concatenation of the shares S_j received. The intuition behind this step is that B certifies of having seen all the S_j values at the same time and in a certain order, while at the same time not revealing or learning anything more. As we will show later, this construct is essential in preventing adversaries from re-using S_j values from multiple un-related protocol runs illicitly, or changing the order of these values before transferring them. Finally, B sends the $Cert$ value to A.

Note that even though B is required to blindly sign random looking values received from A, this constitutes no security vulnerability, as it does so using solely the key-pair $\bar{K}$. The *only* semantics of this signature is that B certifies having seen the values S_j at the same time and in a certain order. This assurance is required in the conditional transfer as described below.

5.2. Initial Conditional Transfer

To perform the initial conditional transfer to C, A sends R_B, all S_j values, $j = 1..n_2 + k$ and their associated $Cert$ value to C. When C receives these values, it first verifies the order of the S_j values, by computing $H(S_1, S_2.., S_{n_2+k}, R_B)$ and comparing it with the value $Cert^{\bar{e_B}} \; mod \; \bar{N_B}$. C continues the protocol only if the verification succeeds.

In the next step, A and C both generate and exchange random values, R_A and R_C. This step could be preceded by a pre-commit (to R_A and R_C)

phase. Using R_A, R_B and R_C, they both compute $n_2 - 1$ *distinct* and *random* values ("indexes") in the space $[0..(n_2 + k - 1)]$, e.g., as follows (where $H()$ is a one-way crypto hash): $I_1 = H(R_A, R_B, R_C)$ *mod* $(n_2 + k)$; remove I_1 from the set, compute $I_2 = H(H(R_A, R_B, R_C))$ *mod* $(n_2 + k - 1)$ and so on, until $n_2 - 1$ distinct index values are computed. The intuition behind these values is that they are both agreed upon and no one party can control or a-priori predict them (thus ensuring fairness of the cut and choose protocol below. This is similar to certain Non-Interactive Zero Knowledge (NIZK) proof protocols where the verifier's challenges are simulated using one-way crypto hashes of inputs generated by the prover.

Then, similar to the CT module of section 4.3, A proves to C that $n_2 - 1$ of the shares are well-formed (signed shares of the same e-cash instance). This time however, *the* $n_2 - 1$ *shares to be revealed are chosen according to the* $n_2 - 1$ *indexes previously computed.* That is, $n_2 - 1$ out of a total of $n_2 + k$ shares are revealed. The randomness of the computed indexes ensures fairness of the cut and choose. Also, $n_2 - 1$ shares are not enough to reconstruct the e-cash.

To later construct the complete e-cash payment, C will proceed as in section 4.4, this time however, *only being required to recover exactly one more out of the remaining* $k + 1$ *unrevealed shares.* This is so because of the $(n_2 + k,n_2)$ secret-splitting (discussed in section 5.1) specifying that n_2 out of $(n_2 + k)$ shares are sufficient for full reconstruction.

5.3. Subsequent Conditional Transfers

The above mechanisms set up the stage for further conditional transfers, e.g., from C to D, once C is convinced of the validity of the shares, before the publication of the condition's outcome. The transfer protocol discussed here can then be deployed unaltered by D for further transfers.

The transfer proceeds exactly as in the previous section, except for the exchange of the R_A and R_C values. That is, instead of C and D generating new random numbers, C transfers R_A and R_C to D. As before, D verifies the authenticity of the S_j shares, the well-formedness of the $n_2 - 1$ shares revealed by A to C, and finally the correct generation of their respective indexes, $I_1, .., I_{n_2-1}$ by one-way hashing from R_A, R_B and R_C, as described in Section 5.2.

Observation. When the payment was generated by A, the value $Cert$ certifies the fact that B has seen the values S_j in a certain order and at the

same time. This, together with the fact that the generation of the indexes I_1 ensures that they are both *random* and (a-priori) *unpredictable* (due to the use of B's R_B value) constitutes proof of fair cut and choose (thus not requiring a separate cut and choose interaction between C and D). A and C are unable to cheat D by re-arranging the S_j values so that only specific shares should be revealed. This is because B has already tied each S_j value with a certain index, in the $Cert$ value. Moreover, A and C are unable to cheat D by replacing some or all the unrevealed S_js with shares generated during older protocol runs. This is because the $Cert$ value certifies the fact that all the S_js were seen at the same time. The cut-and-choose protocol between A and C ensures that the randomly revealed S_js correspond to m_{ljR} values that have the same sequence number R_l, thus, w.h.p. the remaining unrevealed shares also encode the same number R_l.

5.4. Preventing Double Spending (PDS)

This module remains unchanged in spirit as described in section 4.2. However, now, once conditional e-cash is transferred to multiple participants, we need to ensure that only a single participant, i.e., the last in the chain of transactions, should be able to spend in the case of a favorable outcome. The bank can prevent double spending by validating only the first use of the e-cash (as discussed in section 4.2). Then, since all intermediaries are able to construct the e-cash when the condition's outcome is published by T, all intermediaries could race to the bank, potentially preventing the (last) rightful participant from spending its copy of the e-cash.

To avoid this situation and accommodate multi-hop transfers we propose the following. Let $token_R^C$ denote the payment cashing token held by C for the e-cash identified by R_l. This token was denoted by $token_R$ in section 4.2. Before transferring the e-cash to D, C contacts B with a *transfer activation* message containing $\{R_l, v, token_R^C\}$. B verifies that an association $(R_l : \{token_L, token_R^C\})$ already exists. If it does, it invalidates $token_R^C$, by replacing the association with a new one, $(R_l : \{token_L, [token_R^C]\})$. The semantics of the $[token_R^C]$ construct is that the $token_R^C$ is invalid, but still stored for authentication purposes. Once this happens, we call the payment "transferable". In the process, the tokens kept by the bank effectively implement a "barrier" or mediator between different payees.

To transfer the e-cash to D, C now sends $\{R_l, v, token_R^C\}$. D then forwards these values anonymously to B. If B stores an association $(R_l : \{token_L, [token_R^C]\})$, it generates and replies with a new token $token_R^D$. It

then also replaces the association with $(R_l : \{token_L, token_R^D\})$. Otherwise, it refuses to issue a new token and D should not accept the transfer and abort.

If D decides to abort the protocol after the payment was made "transferable" but before the actual transfer takes place, C can perform the same protocol as above to recover ownership of the cash-able payment.

5.5. Collusion Analysis

In addition to the basic properties defined in section 2 which are explored in section 4.5, the transfer protocol requires an additional assurance of non-collusion. Specifically, A and C should be prevented from colluding in an attempt to cheat D, e.g., by deciding a priori which shares not to reveal. Let $AI_1, .., AI_{k+1}$ denote the indexes of these shares. Then, during the payment generation step, A can replace each of these shares with e.g., a random number. B then signs all shares, including the now useless ones. However, the success probability of this can be upper bound; since B's own random input R_B is used to generate the indexes of the actual shares to be revealed, *after* the $AI_1, .., AI_{k+1}$ indexes have been agreed upon, the chance of A and C to cheat by computing random R_A and R_C values is small and can be upper bound by a choice of k:

$$\frac{n_2-1}{n_2+k}\frac{n_2-2}{n_2+k-1}\cdots\frac{1}{k+2} = \frac{(n_2-1)!}{(n_2+k)!/(k+1)!} =$$

$$\frac{1}{\binom{n_2+k}{k+1}} \leq \frac{1}{(\frac{n_2+k}{k+1})^{k+1}} = (\frac{k+1}{n_2+k})^{k+1},$$

where the inequality is derived from $\binom{n}{k} \geq (\frac{n}{k})^k$ (Motwani and Raghavan, 1995). For $n_2 = k$ this is approximately $2^{-(k+1)}$. If $n_2 = k = 100$, this is $\approx 2^{-101}$. In effect this reduces the problem from a full hash length to a 100 bit space, arguably still strong enough (otherwise the value of k can be increased to a desired value).

5.6. Off-line Transferability?

In Section 4.6 we have proposed an offline solution for the conditional e-cash problem without transferability. The natural question is then if we can similarly have the transferable conditional e-cash solution work offline. The following result shows that if C receives the payment from A, before transferring it to D, it has to contact the bank.

In the following, we call a conditional e-cash scheme *fair* if the payee is able to cash the payment in the event of a favorable outcome.

Claim 1. *An offline transferable conditional e-cash solution cannot be fair, preserve the anonymity of well-behaved participants and prevent double spending simultaneously.*

Proof. Let us consider the previous example, where A generates the payment, sends it to C who then in turn transfers it to D. The first observation is that at the time when it generates the payment, A cannot predict the its trajectory. Thus, the initial payment message cannot include any information about C's identity.

Furthermore, assume that after receiving the payment, but before transferring it to D, C is not contacting B (we want off-line behaviour). Then, after the transfer takes place, assume C double-spends by further transfering the same payment to other payees or using it itself (in the event of a favorable outcome). If the bank only satisfies the first valid payment request, D may be cheated out of its money by C performing a "race" attack (see Section 5.4) or by other honest payees that have received the same payment from C.

The above double-spending scenario cannot be prevented by revealing the identity of C. This is first because as shown above, the initial payment message (generated by A) cannot contain C's identity. Second, since any information can added to the payment message by C only offline, no one can force C to include its identity in the payment transfered to D (e.g., as in the previous on-line version where this was enforced by B). Any protocol that would allow D to verify the fact that C's identity is included in the payment transferred, would be equivalent to allowing D to prove to outside parties C's involvement in this protocol. This would however violate property **P7** described in Section 2).

Moreover, A or D cannot provide trusted, world-verifiable payment correctness assurances (e.g., through a signature scheme, acting as B, to certify the payment's properties, as in the online version of the protocol, shown at the beginning of this section) without being trusted not to collude, which is an un-desirable arbitrarily (as A and D could be any parties in the space of participants) strong assumption that would render the protocol un-usable in real scenarios. □

6. Applications

In this section we briefly overview just a few of the application scenarios requiring conditional e-cash payments: financial securities, prediction markets, and anonymous online betting.

6.1. Securities Trading

A particularly relevant application scenario for conditional payments can be found in trade systems involving (atomic) securities. Securities are financial instruments that deliver future value as a function of event outcomes. A simple illustrative instance is the following contract:

> "The Smart Financial Group will pay the bearer of this certificate $50 at the end of the financial year of 2006-2007, if and only if the DOW Jones will increase by 5% since the end of financial year 2005-2006."

Financial institutions can now sell such securities online with full privacy and assurances of payment for their clients.

6.2. Prediction Markets

Yet another application for conditional payments is in prediction markets (IEM, 2007)(New, 2007)(Int, 2007)(Tra, 2007) (Str, 2007). Prediction markets generate assets whose value is conditioned by specific events. For instance, IEM is an educational prediction market of University of Iowa, based on real money, where payoffs are based on real-world events such as political or economic outcomes. Intrade and TradeSports allow their members to speculate for real money on the outcome of a multitude of future events, ranging from politics to sports and pop culture.

Companies such as Hewlett-Packard, Eli Lilly, Microsoft and Google use internal prediction markets, where employees trade futures contracts on sales and profits, success of products or supplier behavior (Kiviat, 2004) (Saporito, 2005). The Iowa Health Prediction Market (IHP, 2007) attempts to forecast the future activity of a wide variety of infectious diseases and related phenomena, by using the unique and fresh knowledge of health-care workers. University of Miami released a Hurricane Futures Market in an attempt to better understand the information that people rely on when forecasting hurricanes.

Conditional payments can enable novel applications for prediction makers and companies with an interest in future outcomes of events. Prediction makers can receive rewards for accurate predictions, while allowing companies to purchase safety for important decisions. For example, manufacturers may use futures markets to direct investments. Additionally, a sense of confidence can be gained if conditional monetary transactions are involved. A prediction maker can express its confidence in a prediction by associating a payment to the manufacturer that is to take place if the outcome of the prediction is unfavorable. In return, the manufacturer agrees to reward the prediction maker if the outcome of the prediction is favorable.

For instance, the Smart Motors Company (SM) may propose the following trade to any willing prediction maker:

> "If crude oil is traded at under \$60 a barrel until the end of 2007, the Smart Motors Company will pay \$6. If the price goes above \$60, SM will be paid \$10. No money changes hands now."

SM and a prediction maker may sign as many of such contracts as they desire.

Manufacturers and prediction makers signing such contracts online are now able to preserve their interactions private, even from the financial institution handling the money. This is important in cases where manufacturers want to hide certain decisions from the competition and where prediction makers may posses insider information.

6.3. Online Betting

Interestingly, the conditional payment mechanisms discussed here can be deployed in the design of anonymous online betting protocols. We briefly outline how.

Without loss of generality, we will consider A as being the betting party and C the "bookie" (the party taking bets). Then, a simple online betting protocol can be constructed as a symmetrical conditional payment scenario. For example, A will provide a conditional (on a certain race outcome) \$1 to C, while C will reciprocate with \$10 conditional on the negated outcome. The race organizer T will publish different t values t_{win} and t_{lose} for a win or a loss respectively.

Even though the payments sent are conditional, either C or A may choose not to reciprocate if the other party sends its payment first. One simple (yet more costly) solution to address this issue is to break each payment into

multiple smaller payments. For instance, for a 2:1 bet for \$100, A may initiate a 10 step protocol, by sending C a \$10 conditional payment. A then waits to receive a \$20 conditional payment from C before sending the next payment. While imposing a larger communication overhead, this ensures that no participant may loose more than 1/10th of the expected value. We also designed a few lower-overhead solutions (of increased exposition complexity) we will not discuss here.

Full Anonymity. The above solution provides a simple betting protocol geared toward achieving anonymity of both C and A with respect to B or T. Often however, online betting protocols would benefit from one additional property, namely full anonymity:

> **P8.** The payer and payee should not be required to know each other's identities nor should they be able to infer these identities from the betting protocol.

This is particularly important in hostile environments with concerns of collusion (of either C or A) with outside parties with incentives to reveal participation in the protocol of either the better or the bookie.

To achieve full anonymity we will require the interaction between A and C to be performed either through a special anonymous IP rendez-vous point, similar to the ones in Tor (Dingledine et al., 2004) or through IRC channels as follows.

C anonymously advertises its public key as well as the service it provides. C also registers its public key along with several introduction points in a lookup service (built to be censorship resilient (Waldman et al., 2000)).

A finds the advertisements and then uses the lookup service to retrieve the introduction points of the bookie. It then chooses an anonymous rendez-vous point as the place where the transaction is to take place and registers its coordinates (encrypted with the public key of the bookie) on one or several of the introduction points. If the bookie decides to accept the better, it retrieves the bet anonymously from the rendez-vous point while it reciprocates with its own conditional payment or engages in a more complex multi-step *simultaneous* payment protocol as above.

A simpler idea is to use IRC channels and messages steganographed into posted media files to also achieve plausible deniability of participation claims in the case of compromised rendez-vous points.

7. Conclusions

In this paper we introduce a novel conditional payment protocol that allows future anonymous cashing of bank-issued e-money only upon the satisfaction of an agreed-upon public condition. Moreover, such payments can be anonymously transferred further by any payee, before their respective condition outcome is known. Application scenarios including online trading of financial securities, prediction markets, and betting systems.

, 2007. Intrade: A trade exchange network company. http://www.intrade.com/.

, 2007. Iowa electronic markets. http://www.biz.uiowa.edu/iem/.

, 2007. Iowa health prediction market. http://fluweb.biz.uiowa.edu/fluhome/index.html.

, 2007. Newsfutures. http://us.newsfutures.com/home/home.html.

, 2007. Strategy page. http://www.strategypage.com/prediction_market/default.asp.

, 2007. Tradesports. http://www.tradesports.com/.

Abe, M., 1998. Universally verifiable mix-net with verification work indendent of the number of mix-servers. In: EUROCRYPT.

Balasko, Y., 1986. Foundations of the Theory of General Equilibrium.

Blake, I. F., Chan, A. C. F., 2005. Scalable, server-passive, user-anonymous timed release cryptography. In: ICDCS '05. IEEE Computer Society, Washington, DC, USA, pp. 504–513.

Blanton, M., 2008. Improved conditional e-payments. In: ACNS'08.

Boneh, D., Franklin, M. K., 2001. Identity-based encryption from the weil pairing. In: CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology.

Brands, S., 1993. Untraceable off-line cash in wallets with observers (extended abstract). In: CRYPTO.

Camenisch, J., Hohenberger, S., Lysyanskaya, A., 2005. Compact e-cash. In: EUROCRYPT '05. Vol. 3494.

Camenisch, J., Lysyanskaya, A., 2004. Signature schemes and anonymous credentials from bilinear maps. In: CRYPTO '04.

Camenisch, J., Lysyanskaya, A., Meyerovich, M., 2007. Endorsed e-cash. In: SP '07.

Camenisch, J., Maurer, U. M., Stadler, M., 1996. Digital payment systems with passive anonymity-revoking trustees. In: ESORICS '96. Springer-Verlag, London, UK, pp. 33–43.

Carbunar, B., Tripunitara, M., 2008. Conditional payments for computing markets. In: CANS '08.

Carbunar, B., Tripunitara, M., 2010. Fair payments for outsourced computations. In: IEEE SECON.

Cathalo, J., Libert, B., Quisquater, J.-J., 2005. Efficient and non-interactive timed-release encryption. In: ICICS. pp. 291–303.

Chaum, D., 1982. Blind signatures for untraceable payments. In: CRYPTO '82. Plenum Press, pp. 199–203.

Chaum, D., 1985. Security without identification: transaction systems to make big brother obsolete. Communications of the ACM 28 (10), 1030–1044.
URL `citeseer.ist.psu.edu/chaum85security.html`

Chaum, D., 1988. Privacy protected payments: Unconditional payer and/or payee untraceability. In: SmartCard 2000.

Chaum, D., Fiat, A., Naor, M., 1990. Untraceable electronic cash. In: CRYPTO '88. Springer-Verlag, London, UK, pp. 319–327.

Chaum, D., Pedersen, T. P., 1992. Transferred cash grows in size. In: EUROCRYPT.

Chaum, D. L., 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24 (2).

Davida, G. I., Frankel, Y., Tsiounis, Y., Yung, M., 1997. Anonymity control in e-cash systems. In: FC '97. Springer-Verlag, London, UK, pp. 1–16.

Dingledine, R., Mathewson, N., Syverson, P., 2004. Tor: The second-generation onion router. In: USENIX Security Symposium '04.

Dodis, Y., Yum, D. H., 2005. Time capsule signature. In: FC '05. pp. 57–71.

Frankel, Y., Tsiounis, Y., Yung, M., 1996. "indirect discourse proof": Achieving efficient fair off-line e-cash. In: ASIACRYPT '96. Springer-Verlag, London, UK, pp. 286–300.

Franklin, M. K., Yung, M., 1993. Secure and efficient off-line digital money (extended abstract). In: ICALP '93. Springer-Verlag, London, UK, pp. 265–276.

K. J. Arrow and G. Debreu, 1954. The Existence of an Equilibrium for a Competitive Economy. Econometrica 22.

Kiviat, B., 2004. The end of management? Time - Inside Business: `http://www.time.com/time/insidebiz/printout/0,8816,1101040712-660965,00%.html`.

M., S., J.-M, P., J., C., 1995. Fair blind signatures. In: EUROCRYPT '95. pp. 209–219.

Motwani, R., Raghavan, P., 1995. Randomized algorithms. Cambridge University Press.

Park, C., Itoh, K., Kurosawa, K., 1994. Efficient anonymous channel and all/nothing election scheme. In: EUROCRYPT '93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology.

Samuelson, P. A., 1947. Foundations of Economic Analysis. Harvard University Press.

Saporito, B., 2005. Place your bets! Time: `http://www.time.com/time/magazine/article/0,9171,1118373,00.html`.

Shamir, A., 1979. How to share a secret. Commun. ACM 22 (11).

Shi, L., Carbunar, B., Sion, R., 2007. Conditional e-cash. In: Financial Cryptography.

Simon, D. R., 1996. Anonymous communication and anonymous cash. In: CRYPTO '96. Springer-Verlag, London, UK, pp. 61–73.

Waldman, M., Rubin, A. D., Cranor, L. F., August 2000. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In: USENIX Security Symposium '00. pp. 59–72.