Easy come – easy go divisible cash

Agnes Chan * Yair Frankel † Yiannis Tsiounis ‡

October 16, 1996

Abstract

Recently, there has been an interest in making electronic cash protocols more practical for electronic commerce by developing e-cash which is divisible (e.g., a coin which can be spent incrementally but total purchases are limited to the monetary value of the coin) [DC94, EO94, OO92, Pai93, Oka95]. In Crypto'95, T. Okamoto presented the first practical divisible, untraceable, off-line e-cash scheme, which requires only $O(\log \mathcal{N})$ computations for each of the withdrawal, payment and deposit procedures, where $\mathcal{N}$ =(total coin value)/(smallest divisible unit). However, Okamoto's set-up procedure is quite inefficient (on the order of 4000 "multi-exponentiations" and depending on the size of the RSA modulus).

We formalize the notion of "range-bounded commitment," originally used in Okamoto's account establishment protocol, and present a very efficient instantiation which allows us to construct the first truly efficient divisible e-cash system. Our scheme only requires the equivalent of one (1) exponentiation for set-up, less than two (2) exponentiations for withdrawal and around 20 for payment, while the size of our coin remains about 300 Bytes. Hence, our withdrawal protocol is 3 orders of magnitude faster than Okamoto's, while the rest of our system remains equally efficient, allowing for implementation in smart-cards. Similar to Okamoto's, our scheme is based on proofs whose cryptographic security assumptions are theoretically clarified.

Keywords: Electronic cash, efficient, anonymity, divisibility, range-bounded commitment, tracing, linking, provable, Williams integers.

1 Introduction

Off-line untraceable electronic cash has sparked wide interest among cryptographers ([CFN90, FY93, Oka95, PW92, Bra93b, DC94, EO94, OO92, Pai93], etc). In its simplest form, an anonymous off-line e-cash system consists of three parties (a bank $\mathcal{B}$, a user $\mathcal{U}$, and a receiver $\mathcal{R}$) and four main procedures (account establishment, withdrawal, payment and deposit). The user $\mathcal{U}$ performs an *account establishment protocol* to open an account with bank $\mathcal{B}$. To withdraw money, $\mathcal{U}$ performs a *withdrawal protocol* with $\mathcal{B}$ over an authenticated channel. User $\mathcal{U}$ spends a coin by

*College of Computer Science, Northeastern University, Boston, Massachussetts.
†Sandia Laboratories, New Mexico.
‡College of Computer Science, Northeastern University, Boston, Massachussetts. e-mail: yiannis@ccs.neu.edu

This work was supported by the United States Department of Energy under Contract DE-AC04-94AL 85000.
Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy.

DISCLAIMER

participating in a *payment protocol* with the receiver $\mathcal{R}$ over an anonymous channel. $\mathcal{R}$ then performs a *deposit protocol* with the bank $\mathcal{B}$, to deposit the user's coin. The system is *anonymous* if the bank $\mathcal{B}$, in collaboration with the receiver $\mathcal{R}$, cannot trace the coin to the user. The system is *off-line* if during payment the receiver $\mathcal{R}$ does not communicate with the bank $\mathcal{B}$. However, if a coin is double spent, the user's identity is revealed with overwhelming probability.

More recently, there has been a strong effort in developing secure divisible untraceable off-line electronic cash protocols [DC94, EO94, OO92, Pai93, Oka95]. With *divisible e-cash* a coin of value $\$x$ can be spent in several increments but the total amount can not exceed $\$x$, unless the user is willing to be identified with high probability. This paper develops a new efficient off-line divisible e-cash protocol.

In Crypto '95, Okamoto [Oka95] presented the first divisible e-cash scheme in which all procedures can be performed efficiently (i.e., in $O(\log \mathcal{N})$, where $\mathcal{N}$ = (total coin value)/(smallest divisible unit)). Furthermore, all protocols, *except establishing an account*, are of comparable efficiency with the most efficient untraceable off-line e-cash systems available. Hence, [Oka95] was a major break-through in electronic cash research.

The protocol presented in this paper, in comparison, has an efficient "account establishment" protocol, hence its functionality can be included in every withdrawal. Therefore, unlike [Oka95], we have no trade-off between the degree of unlinkability among coins and efficiency attained. Having no compromise between coin linkability and practicality is a significant difference between ours and [Oka95]. As the name of this paper "easy come / easy go divisible cash" suggests, our protocols are efficient in providing the user with the coin, as well as efficient in allowing the user to spend the coin.

Efficiency of our scheme:

The major advantage of our system is that the construction of the electronic license (the bulk of the computation in [Oka95]'s "user account establishment" protocol) requires the equivalent of *less than* two (2) modular exponentiations of [Oka95], while [Oka95] requires more than 4000. Furthermore, in contrast to our scheme, the number of exponentiations in [Oka95] depends on the length of the RSA modulus (which is an insufficient 512 bits in their efficiency calculations), impairing the scalability of the system. In our scheme, account establishment is used to exchange authentication keys to be used later in withdrawal. Similar to ([Bra93b, EO94, OO92, Pai93, CFN90], etc.) we put the functionality of licensing into the withdrawal protocol.

It should be noted, as noted by [PW92], the more the user uses the same license the more likely he can be traced by other means (i.e., correlating various payments' locality, date, type, frequency, etc.) The cost of not performing the account establishment protocol at each withdrawal is that withdrawals of coins using the same license can be linked. Hence efficient license generation impacts annonymity.

Our withdrawal protocol requires the user and the bank to perform the equivalent of less than two (2) and one (1) exponentiations respectively. During payment, the user and the bank perform around 20 exponentiations. The size of our coin remains around 300 bytes. Hence, our scheme can be implemented in current (PC-based) smart cards, while allowing for coins to be divisible.

As in the Okamoto scheme, our scheme is based on proofs whose "cryptographic security assumptions are theoretically clarified".

Organization: We present our range-bounded commitment in Section 2. We proceed with an overview of Okamoto's [Oka95] scheme in Section 3, focusing on the account establishment and withdrawal protocols. In Section 4 we sketch our idea and present our scheme in Section 5. Next we discuss the scheme's security (Section 6), based on a formal security model, and its efficiency (Section 7).

2 Range-bounded commitment

The idea of checking whether a number is in a specific range and a protocol for its instatiation were first proposed by T. Okamoto in [Oka95] for the inefficient license generation. We propose to call such protocols *range-bounded commitments*. Here we formalize the notion of a range-bounded commitment and present an efficient instatiation based on the Discrete Logarithm Assumption *(DLA)*; informally, this is a protocol between a prover, $\mathcal{P}$, and a verifier, $\mathcal{V}$, with which $\mathcal{P}$ can commit to a string, x, and prove to $\mathcal{V}$ that x is in a predetermined range. Formally,

Definition 2.1 *A* **range-bounded commitment** *scheme consists of a pair of probabilistic polynomial-time interactive machines, denoted $(\mathcal{P},\mathcal{V})$ (for* prover *and* verifier*), satisfying:*

- Input specification: *The private input to the prover is a string x for which $|x| \leq H$. The common input is a* bit commitment *$b(x)$ on x, an integer k presented in unary (the* security parameter*), an integer H (the specified* range*) and a fraction δ (the* accuracy*).*
- Completeness: *The prover can prove that x is in the specified range. Namely, for every x such that $|x| \leq H$ and for every polynomial $p(\cdot)$,*

$$Prob[(\mathcal{P},\mathcal{V})(b(x)) = 1] \geq 1 - \frac{1}{p(k)} ,$$

  where the probability is taken over the coin tosses of $\mathcal{P}$ and $\mathcal{V}$.
- Soundness: *The verifier is convinced that x is in the range of H. Namely, for every probabilistic polynomial-time machine $\mathcal{P}^*$ (a* cheating prover*), every x such that $|x| > (1+\delta)H$ and for every polynomial $p(\cdot)$,*

$$Prob[(\mathcal{P}^*,\mathcal{V})(b(x)) = 1] \leq \frac{1}{p(k)} ,$$

  where the probability is taken over the coin tosses of $\mathcal{P}^*$ and $\mathcal{V}$.
- Secrecy: *The verifier does not obtain any information about x, other than $b(x)$: for every probabilistic polynomial-time machine $\mathcal{V}^*$ (a* cheating verifier*), there exists a probabilistic polynomial-time machine $\mathcal{M}^*$ (a* simulator*) so that with probability—taken over the coin tosses of $\mathcal{P},\mathcal{V}^*$ and $\mathcal{M}^*$—overwhelming in k the following two ensembles are polynomially indistinguishable*
  - $\{(\mathcal{P},\mathcal{V}^*)(b(x))\}_{|x|<H}$, *and*
  - $\{\mathcal{M}^*(b(x))\}_{|x|<H}$.

We now present an efficient range-bounded commitment protocol based on the DLA. An interesting point to note is that while computations are normally performed

modulo a prime Q computations involving exponents are not performed modulo $Q-1$ but on the integers, so that the range of the numbers can be checked.

Setup: Define security parameters H, S, k, $\delta > (3k+2)/H$, $k \leq \epsilon < \delta H - 2k$, where S is the desired length of the output of the random oracle-like hash function if the protocol is made non-interactive, ϵ is the length of the verifier's challenge if the protocol is interactive and the other parameters conform to the definition of the range-bounded commitment above.

Given a prime Q with $|Q| = 2(1+\delta)H + 6$, and $X = g^x \bmod Q$, prover $\mathcal{P}$ will prove to verifier $\mathcal{V}$ that $|x| \leq (1+\delta)H$.

The protocol:

(1) $\mathcal{P}$ picks $u_i \in_R \{0, \ldots, 2^{(1+\delta)H}\}$, and sends $U_i = g^{u_i}$ to $\mathcal{V}$.

(2) $\mathcal{V}$ sends $e_i \in_R \{0, \ldots, 2^{\epsilon}\}$.

(3) $\mathcal{P}$ responds with $u_i' = e_i x + u_i$.

(4) $\mathcal{V}$ verifies $g^{u_i'} = X^{e_i} U_i$ and $0 \leq u_i' \leq 2^{(1+\delta)H}$.

The protocol can be collapsed to one move if $e = e_1 \ldots e_j = \mathcal{H}_0(X, U_1, \ldots, U_j)$, where j is the number of iterations. In this case, it must be executed $j = \lceil S/\epsilon \rceil$ times, to guarantee that $|e| \geq S$.

Note that in each iteration the prover has a probability of $1/2^{\epsilon} \leq 1/2^k$ of incorrectly convincing the verifier that x is in the specified range, by selecting $u_i = -e_i x + u_i'$ for some $u_i' \in_R \{0, \ldots, 2^{(1+\delta)H}\}$.

The probability that a legitimate $\mathcal{P}$ (i.e., for which $|x| < H$) fails to convince $\mathcal{V}$ is $1 - (1 - 2^{\epsilon - \delta H})^j$ (the probability that for some i, $u_i > 2^{(1+\delta)H} - e_i x$), i.e. $< 1 - (1 - 2^{-2k})^j$, from the selection of ϵ, δ, H. According to [FGY96], whose second protocol is a generalized case of ours, the probability that $\mathcal{V}$ can extract some information regarding x is $j 2^{-\delta H + 1 + \epsilon}$ $(< j 2^{-2k+1})$.

Formally, we can prove the following theorem:

Theorem 2.1 *Assuming $g^x \pmod Q$ is a secure bit commitment on x, the above protocol is a range-bounded commitment.*

Remark: Security can be proven based solely on the DLA as long as $X \equiv g^x \pmod Q$ hides all information about x; in reality, however, even assuming the DLA some minimal information about x is revealed given X. For applicability in our e-cash scheme this leak of information is not important, since X is already known to $\mathcal{V}$; hence the range-bounded commitment protocol can be simplified by introducing this stronger assumption.

3 The Okamoto scheme

In Okamoto's divisible off-line e-cash scheme [Oka95] each user $\mathcal{U}$ generates a composite number $N = PQ$, such that N is a Williams integer[1] associated with $\mathcal{U}$.

In **the account establishment protocol** the bank $\mathcal{B}$ publishes its RSA public keys (n_1, K) and (n_2, K) and also (a_1, a_2) as public keys. It also publishes prime $\mathcal{P}$ and generator $g \in \mathcal{P}$. A users license is $(N, L_1 = (N + a_1)^{1/K} \bmod n_1, L_2 =$

[1] $N = PQ$ is a Williams integer iff P, Q are prime, and $P \equiv 3 \mod 8, Q \equiv 7 \mod 8$.

$(N + a_2)^{1/K} \bmod n_2)$ where L_1 and L_2 are blindly signed by the bank after the user proves that input is of the correct form. In [Oka95] Okamoto shows that this protocol takes approximately 4000 "multi-exponentiations"[2] modulo $\mathcal{P}$, assuming 256 bit primes P and Q (i.e. an RSA modulus of 512 bits) and a security parameter of $k = 20$ (i.e. the probability of misbehaving undetected is $1/2^{20}$). Furthermore the number of exponentiations depends both on the security parameter (k) and the length of the RSA modulus.

Withdrawal of the coin is nothing more than an RSA blind signature [Cha83] on $H(N||b)$, where H is a one-way function, b is a random value and the bank's public RSA key is dependent on the value of the coin.

The coin (i.e., the value N) defines a tree such that the following rules are satisfied:

- **Root route rule:** Once spent, a node's ancestors and descendants can not be used.
- **Same node rule:** A node can not be used more than once.

(See [Oka95, EO94, OO92, Pai93] for details, and appendix B for a short description).

The payment protocol consists of two parts:

- **(Coin Authentication)** $\mathcal{U}$ convinces $\mathcal{R}$ that the coin is a legitimate coin (i.e. it is signed by $\mathcal{B}$, and if N is factored then $\mathcal{U}$ is identified).
- **(Denomination Revelation)** $\mathcal{U}$ presents some data that are specific to the node(s) of the tree that is/are being spent, in such a way that $\mathcal{R}$ is guaranteed that (a) N is a Williams integer and (b) if $\mathcal{U}$ violates the root route or the same node rule then N can be factored.

The reader should note that the same N is revealed for each coin with the same license. Hence, coins can be linked. Our system does not have this property.

4 The basic idea

Okamoto's scheme is quite efficient. In fact it is only inefficient during the account establishment protocol. To emulate the functionality of this protocol, all that is needed is a method to provide a receiver $\mathcal{R}$ with an N, such that (1) N is a composite of two numbers, (2) N is signed by the bank, and (3) $\mathcal{R}$ (and subsequently the bank, at deposit time) is guaranteed that if N is factored, the owner of the coin will be identified. Condition (1) is satisfied by the denomination revelation protocol of [Oka95] which determines if N is a Williams integer and generates the tree as discussed in Section 3. What we suggest is a new approach for withdrawal (i.e. signing N) and coin authentication (i.e. proving the correctness of N to $\mathcal{R}$).

Our idea is to use a modified Brands [Bra93b] protocol for withdrawal and coin authentication. This can be done if an efficient range-bounded commitment is provided. $\mathcal{U}$ has (at account establishment) associated his identity with $I' = g_3^u$. At withdrawal, he randomly generates $N = pq$ and identifies the particular withdrawal (hence himself) with $I_W = g_1^p$. During withdrawal, $\mathcal{U}$ will end up with a message $(A = g_1^{pq} g_2^q g_3^{uq}, B = [N])$ and a signature on A, B: $sign(A, B)$. Hence (2) above is

[2] Each multi-exponent is equivalent to xxx modular exponents.

guaranteed. The correctness of A and the unforgeability of the signature are guaranteed by the protocol in [Bra93b].

To guarantee condition (3), we observe that during payment N is revealed and, if the coin is over-spent, N can be factored in the denomination revelation phase, based on the result in [Oka95], and as corrected in [?]. At coin authentication $\mathcal{U}$ proves that $A = g_1^N X$ for some $X = g_2^\beta g_3^\gamma$. Since $A = g_1^{pq} g_2^q g_3^{uq}$, this indirectly guarantees that $N = pq$, i.e. the factorization of N reveals I'. Notice that this only holds if we guarantee that p, q are small enough so there is no wrap-around in the modulus used (i.e. in $g_1^{pq} \pmod{Q}$, $pq < Q - 1$); our range-bounded commitment is used for this purpose.

5 The scheme

Remark: There are actually two viable variants of our scheme. In one variant, $u = 0$, i.e. only three generators (g, g_1, g_2) are used and the identifying information of the user $\mathcal{U}$ is g_1^p, created by the user at each withdrawal and stored by the bank $\mathcal{B}$. This variant, apart from requiring less communication/computation from both $\mathcal{U}$ and the bank $\mathcal{B}$, also allows us to prove the untraceability of our scheme with respect to an assumption similar to one appearing in [Oka95].

If $u \neq 0$, on the other hand, the bank, upon tracing a double-spender, only has to perform a search in its account database (in order to locate $I' = g_3^u$). In contrast, in the first variant, it would have to store all identification values $I_W = g_1^p$ appearing in withdrawal protocols, and then perform a search among them (note that this does not increase the order of computation or storage needed by $\mathcal{B}$, since $\mathcal{B}$ has to store transcripts, and perform searches—at *each* deposit—of deposit protocols too). The proof of this variant depends on a slightly more complex assumption, which we nevertheless believe is an interesting number theoretic problem to be analyzed.

We present our scheme when $u \neq 0$. It is then easy to derive the first variant (when $u = 0$). When we discuss our scheme's security (Section 6) we present the assumptions needed for both variants.

We use a generalization of the discrete logarithm problem *(DLP)*, the representation problem in groups of prime order; this is equivalent to the DLP [Bra93a]:

Definition 5.1 (The representation problem in groups of prime order)
Instance: *A group G_Q, a generator-tuple $(g_1, \ldots, g_k), h \in G_Q$.*
Problem: *Find a representation of h with respect to $(g_1, \ldots, g_k)$.*

5.1 Initialization

Bank Initialization (setup) procedure:
The bank $\mathcal{B}$ chooses the security parameters k, n, S, $H = |p| = |q| = |N|/2$, $\delta > (3k + 2)/H$, $n \leq k \leq \epsilon < \delta H - 2k$, and prime Q, with $|Q| = 2(1 + \delta)H + 6$. All arithmetic is performed in G_Q, except for the operations involving exponents, which are performed in Z_Q. $\mathcal{B}$ chooses:

- Four generators g, g_1, g_2, g_3 of G_Q,
- $\mathcal{H}, \mathcal{H}_0, \mathcal{H}_1, \ldots$, from a family of collision intractable hash functions,

- A private key $x \in_R Z_Q$ (a different key is used for every denomination).

$\mathcal{B}$ publicizes the description of G_Q (i.e. Q), the generator-tuple (g, g_1, g_2, g_3), the description of $\mathcal{H}, \mathcal{H}_0, \mathcal{H}_1, \ldots$, and its public keys $h = g^x, h_i = g_i^x,\ i = (1,2,3)$.
User Initialization (account establishment) procedure:
The user $\mathcal{U}$ shows (by physical or other means) his identity to the bank $\mathcal{B}$ and then associates himself[3] with $I' = g_3^u$ to $\mathcal{B}$. The bank verifies that $I' \neq \{1, g_3\}$.

5.2 Withdrawal

The signature that is used by the bank to sign a coin is a variation of the Schnorr signature [Sch91] and is also used in [Bra93b]. The signature $sign(A, B)$ on the pair $(A, B) \in G_Q \times G_Q$, consists of a tuple $(z, a, b, r) \in G_Q \times G_Q \times G_Q \times Z_Q$, such that:

$$g^r = h^{\mathcal{H}(A,B,z,a,b)} a \quad \text{and} \quad A^r = z^{\mathcal{H}(A,B,z,a,b)} b \tag{1}$$

The withdrawal protocol

At the beginning of the withdrawal protocol, the user creates an authenticated channel with the bank. This is needed in all e-cash (and physical cash!) protocols to guarantee that only the owner of an account withdraws money from it and that the user is communicating with the real bank. If $u \neq 0$ (i.e. the second variant is used), this functionality is included in our withdrawal protocol[4].

- $\mathcal{U}$: (This step can be pre-computed.)

  Select primes $p \equiv 3 \bmod 8, q \equiv 7 \bmod 8,\ |p| = |q| \leq H = (|Q| - 6)/[2(1+\delta)]$ at random, and calculate $N = pq$.

  Send $I_W = g_1^p$ and $I' = g_3^u$ to $\mathcal{B}$.

- $\mathcal{U}, \mathcal{B}$: Perform a Schnorr proof of knowledge [Sch91] that $\mathcal{U}$ knows the representation of I' w.r.t. g_3.

- $\mathcal{U}, \mathcal{B}$: Use *the range-bounded commitment* (base g_1, with security parameters $H, S, k, \delta, \epsilon$) with I_W to prove—in an interactive way and with just one iteration—that $|p| \leq (1+\delta)H$.

- $\mathcal{B}$: Set $I = I_W I' (= g_1^p g_3^u)$, and check that $I_W \neq \{1, g_1\}$, $Ig_2 \neq 1$.

  Pick $w \in_R Z_Q$, and send $a' = g^w,\ b' = (Ig_2)^w$ to $\mathcal{U}$.

- $\mathcal{U}$: Compute $z' = (h_1)^p h_2 (h_3)^u$ $[= (Ig_2)^x$ since $h_1 = g_1^x,\ h_2 = g_2^x,\ h_3 = g_3^x]$. This step can be pre-computed (or z' can be supplied by $\mathcal{B}$).

  Let $A = (Ig_2)^q = g_1^N g_2^q g_3^{uq},\ B = [N, Y = g_2^q]$, and $z = z'^q$.

  Pick $v_1 \in_R Z_Q^*,\ v_2 \in_R Z_Q$ and compute $a = a'^{v_1} g^{v_2}$ and $b = b'^{q v_1} A^{v_2}$.

  Compute the challenge $c = \mathcal{H}(A, B, z, a, b)$, and send the blinded challenge $c' = c/v_1 \pmod{Q}$ to $\mathcal{B}$.

[3] In the first variant $\mathcal{U}$ sends to $\mathcal{B}$ his public key –using any public key cryptosystem– so that an authenticated channel can be created at withdrawal ($\mathcal{U}$ already has $\mathcal{B}$'s keys from the bank initialization protocol).

[4] The user $\mathcal{U}$ proves to the bank $\mathcal{B}$ that he knows the representation of I' w.r.t. g_3, and $\mathcal{U}$ is convinced of $\mathcal{B}$'s identity by checking the correctness of $sig(A, B)$.

- $\mathcal{B}$: Send the response $r' = c'x + w \pmod{Q}$ to $\mathcal{U}$, and debit $\mathcal{U}$'s account.
- $\mathcal{U}$: Accept iff $g^{r'} = h^{c'}a'$ and $(Ig_2)^{r'} = z'^{c'}b'$. Compute $r = r'v_1 + v_2 \pmod{Q}$, to get the signature (z, a, b, r) on (A, B).

See appendix D for a graphical presentation of the protocol.

5.3 Payment & Deposit

Coin Authentication:

- $\mathcal{U}$: Pick $x_3 \in_R Z_Q$. Compute $Y = g_2^q$, $Y_3 = g_3^{uq}$, $y_3 = g_3^{x_3}$, $d = \mathcal{H}_1(A, B, Y_3, y_3, \text{date/time}, ID_{\mathcal{R}})$.

  This is a non-interactive approach but one could add a random challenge from $\mathcal{R}$ into the hash ($\mathcal{H}_1$) if desired[5]. The non-interactive case allows for the payment protocol to be conducted in one move, from $\mathcal{U}$ to $\mathcal{B}$.

  This step (except for the challenge, if an interactive approach is used) can be pre-processed.
- **$\mathcal{U}$ sends the coin to $\mathcal{R}$:** Send $A, B = [N, Y], sign(A, B), Y_3, y_3$, and respond to challenge d with $r_3 = duq + x_3$.
- **$\mathcal{R}$ verifies that the coin is legitimate:**
  1. **Verify the signature** $sign(A, B)$, and that $Y \neq g_2$, $Y \neq g_2^N$, $A \neq 1$, $Y_3 \neq g_3$, $(-1/N) = 1$, $(2/N) = -1$.
  2. **Verify that $\mathcal{U}$ knows a representation of Y_3 with respect to g_3 using [Sch91]:** $g_3^{r_3} \stackrel{?}{=} y_3 Y_3^d$.
  3. **Prove that q is chosen correctly:** Use *the range-bounded commitment* (base g_2) with Y, to prove that $|q| \leq (1+\delta)H$ (the challenge e is computed based on a hash function—as d above—so that even a collaboration of $\mathcal{U}$ and $\mathcal{R}$ cannot forge the proof). $\lceil S/\epsilon \rceil$ iterations are performed as discussed in Section 2.
  4. **Verify that A is correctly constructed:** $g_1^N Y Y_3 \stackrel{?}{=} A$.
  5. **Limit the way $\mathcal{U}$ can misbehave:** Check whether N is divided by the first $|N|$ primes that are congruent to 3 mod 8 or 7 mod 8. This addition is necessary due to a flaw in [Oka95]'s denomination revelation protocol [?], and simplifies identification of double-spenders, as shown in [?]. [?] also describes the tracing protocol used by $\mathcal{B}$ in this case. We adopt this protocol, but omit its description due to space considerations.

Denomination Revelation: We use [Oka95]'s protocol, with the only modification being the substitution of the coin (C, N) in the hash functions of Okamoto with our coin, (A, B). This protocol guarantees that if one of the node rules (see Section 3) is violated, then $\mathcal{U}$ has released enough information to allow $\mathcal{B}$ to factor N. Note that if $k' < k/4$ nodes are spent, then $2 \cdot (k/4 - k)'$ additional square roots of randomly

[5] In any case, A, B, Y_3, y_3, date/time, $ID_{\mathcal{R}}$ must be included in the hash (as in the self-challenging Schnorr proof of knowledge), so that even if $\mathcal{R}$ and $\mathcal{U}$ collaborate the subsequent proofs of knowledge are still valid.

chosen numbers must be shown by the user; these are described in [Oka95]'s coin authentication and are also performed here.

Deposit: $\mathcal{R}$ sends the payment transcript to $\mathcal{B}$.

6 Security

We now present our security model and give an overview (due to space limitations) of the proofs.

As with [Oka95], our security model has been based on [FY93] and is modified to work for divisible, unlinkable coins. To our knowledge, this is the first formal model in the literature covering *unlinkability*. We will model the security of our scheme by requiring that it satisfies four requirements, which are slightly stronger than the respective [Oka95] properties (included in brackets) that do not include unforgeability: *unreusability* [No overspending], *untraceability* [No tracing], *unexpandability* [No forging and No swindling[6]], and *unforgeability*. The use of a *non-uniform*, probabilistic polynomial time machine *(p.p.t. TM)* in our model simulates user collaboration as views to the TM. Thus, in establishing the security we prove that even a collaboration of users (and/or shops) cannot break the scheme.

Our proofs of security are based on the following **assumptions:**

- (when $u = 0$) **(Factoring and Diffie-Hellman II)** Let Q, p_0, q_0, p_1, q_1 be primes, $N_0 = p_0q_0$, $N_1 = p_1q_1$, and $H = |p_0| = |q_0| = |p_1| = |q_1| \leq (|Q| - 6)/(2(1+\delta))$ for some sufficiently large $\delta > 0$. Let the order of g in the multiplicative group Z_Q^* be Q. Then, no p.p.t. TM $\mathcal{M}$ can, given $Q, \delta, g, [Y_0(\equiv g^{q_0} \bmod Q), N_0(= p_0q_0)], [Y_1(\equiv g^{q_1} \bmod Q), N_1(= p_1q_1)]$ and $[I_r(\equiv g^{p_r} \bmod Q), I_{1-r}(\equiv g^{p_{1-r}} \bmod Q)]$ $(r \in_R \{0,1\})$, compute r with probability better than $1/2 + 1/H^c$, for all constants c and sufficiently large H (i.e. $\mathcal{M}$ cannot compute r non-negligibly better (in H) than random guessing).
- **(Withdrawal protocol)**[7] If random hash functions exist, then our withdrawal protocol is a *restrictive blind signature protocol*: the message $m = Ig_2 = g_1^{u_1}g_3^{u_3}g_2$ is signed by the string $A = g_1^{\alpha}g_3^{\gamma}g_2^{\beta}$, in such a way that $\alpha/\beta = u_1, \gamma/\beta = u_3$.
- **(Hash functions)** Hash functions $(\mathcal{H}, \mathcal{H}_0, \mathcal{H}_1, \ldots)$ behave like truly random functions.

Remarks: Remarks on these assumptions are provided in appendix A.
For our proofs we use the following lemma, which has been proven by [?], based on our *hash functions* assumption:

[6] "No swindling" is guaranteed from unreusability and unexpandability: even a collaboration of users/shops cannot over-deposit the withdrawn/paid coins.

[7] We can reduce this to Brands' original assumption from [Bra93b]. Assume it does not hold. Then let $g_2' = g_1^p g_2, g_2'' = g_3^u g_2$. Then, the original assumption, with either $(g_1, g_2 = g_2')$, or $(g_1, g_2 = g_2'')$, does not hold. The other direction is trivial.

Lemma 6.1 (Schnorr signatures) *Schnorr signatures [Sch91] are existentially unforgeable, even when the prover in the Schnorr identification protocol is queried polynomially many times.*

Theorem 6.2 Unreusability:
Let n be the security parameter. If the successfully deposited nodes of a coin violate the route node rule *or the* same node rule*, then the identity of the coin's owner can be efficiently (i.e. by a p.p.t. TM) computed (and subsequently proven) from the transcripts of the withdrawal and the deposit protocols with overwhelming probability (in n).*

Proof. The *Withdrawal protocol* assumption and lemma 6.1 (which themselves require assumption *Hash functions* above), together with the verification done at the coin authentication stage, guarantee that the element A (of the coin (A, B)) is of the form $g_1^{pq} g_2^q g_3^{uq}$ for some p, q, u (p, q not necessarily prime) and where $I_W \equiv g_1^p$, $I' \equiv g_3^u \pmod{Q}$. We will also show that p is a prime factor of the N included in B, and how this leads to identification of $\mathcal{U}$, i.e. to I_W and I'.

Steps 2 and 3 in payment guarantee that $\mathcal{U}$ knows a representation of Y, Y_3 w.r.t. g_2, g_3, respectively: $Y = g_2^t, Y_3 = g_3^{t_3}$, for $t, t_3 \in Z_Q^*$, $|t| \leq (1+\delta)H < |Q|$. With step 4, this proves that $\mathcal{U}$ knows $(N, t, t_3), (u, p, q)$ such that: $A \equiv g_1^N g_2^t g_3^{t_3} \equiv g_1^{pq} g_2^q g_3^{uq} \pmod{Q}$.

If $N \not\equiv pq \pmod{Q-1}$, $t \not\equiv q \pmod{Q-1}$ or $t_3 \not\equiv uq \pmod{Q-1}$, then $\mathcal{U}$ would know more than one representation of A with respect to (g_1, g_2, g_3), which contradicts the representation problem in groups of prime order and hence our *Withdrawal Protocol* assumption. Therefore, $t \equiv q \pmod{Q-1}$, $t_3 \equiv uq \pmod{Q-1}$, $N \equiv pq \equiv pt \pmod{Q-1}$, and $Y \equiv g_2^q, Y_3 \equiv g_3^{uq} \pmod{Q}$.

Assume that $N = p'^i q'^j$, with $p' \equiv 3 \bmod 8$, $q' \equiv 7 \bmod 8$ primes, i, j odd integers, and that N (and therefore p', q', i, j) is unique. Then $p'^i q'^j = N \equiv pt \pmod{Q-1}$. But $|N| < |Q|$ (since $|p|, |t| \leq (1+\delta)H, |Q| = 2(1+\delta)H + 6$), hence $N < Q - 1$ and $p'^i q'^j = N = pt$. Since $t|N$, t is either $N, 1$ or $p'^{i'} q'^{j'}$, $i' \leq i, j' \leq j$. But at coin authentication $\mathcal{R}$ has verified that $g_2^t \equiv Y \notin \{g_2, g_2^N\} \pmod{Q}$, i.e. $t \notin \{1, N\}$. Thus, $t = p'^{i'} q'^{j'}$ and, consequently, $p = N/t = p'^{i-i'} q'^{j-j'}$.

[Oka95] proves, under the assumption that factoring is difficult (which, in turn, is included in assumption *Factoring and Diffie-Hellman II* above), and assumption *Hash functions* above, that N is guaranteed to be of the form $p'^i q'^j$ with special p', q', and if the route node rule or the same node rule are violated, then $\mathcal{B}$ obtains p'^i, q'^j. Also, N is blindly signed by $\mathcal{B}$ at withdrawal (it is included in $\mathcal{H}(A, B, z, a, b)$), hence it is unique.

The ammendment to [Oka95]'s coin authentication protocol proposed by [?] guarantees that $\mathcal{B}$ can, with an acceptable overhead, find i, i', j, j' such that $g_2^t \equiv g_2^{p'^{i'} q'^{j'}} \pmod{Q} \iff q \equiv t \equiv p'^{i'} q'^{j'} \pmod{Q-1}$, $p \equiv p'^{i-i'} q'^{j-j'} \pmod{Q-1}$, $I_W \equiv g_1^p \pmod{Q}$ and $I' \equiv A^{(q)^{-1}} / (g_1^p g_2) \equiv A^{(t)^{-1}} / (g_1^p g_2) \pmod{Q}$, and the fact that $\mathcal{B}$ now knows a representation of I_W w.r.t. g_1 constitutes proof of double spending[8]. Therefore, if $\mathcal{U}$ over-spends he is identified. □

[8] Since the representation problem is equivalent to the DLA; but the DLA is implicit in [Bra93b] and hence our *withdrawal protocol* assumption.

Theorem 6.3 *Let n be the security parameter. WLOG we treat the collection of the portions of a coin as being a single, indivisible, coin.*
Unforgeability: *No p.p.t. TM can, from the views of users of arbitrarily many withdrawal and payment protocols, compute a single coin that does not embed the identity of at least one of these users and that will lead to two successful purchase (or deposit) protocols, except with negligible probability (in n).*

Unexpandability: *The probability that from the views of users and shops of N withdrawal and of N payment protocols, a p.p.t. TM can compute an additional coin that will lead to a successful purchase (or deposit), is negligible (in n).*

We discuss the validity of the above theorem. In [Bra93b] it is proven, under assumption *Hash functions* and lemma 6.1 above, that it is infeasible to existentially forge a coin (*unexpandability*), even when performing the withdrawal protocol polynomially many times and with respect to different account numbers. It is also proven, under assumption *Withdrawal protocol*, that every coin embeds the identity of its owner (*unforgeability*). [Bra93a] shows that these proofs are valid even if $\mathcal{U}$'s identity is represented in more than one generator (as in our case, (g_1, g_3)). Our scheme restricts $\mathcal{U}$'s power, in comparison to [Bra93b]: $\mathcal{U}$'s secret numbers (p, q) have to be factors of N, where $|N| < |Q|$. Our scheme also proves (at payment time) that $\mathcal{U}$ knows the representation of A w.r.t. (g_1, g_2, g_3) as in [Bra93b, Bra93a]. Hence, the proofs of [Bra93b, Bra93a] carry along, and guarantee the unforgeability and unexpandability of our scheme.

Theorem 6.4 Untraceability:
Let n be the security parameter. WLOG we treat the collection of the portions of a coin as being a single, indivisible, coin. Let $\mathcal{W}_i$ be the set of all withdrawals (W) of user $\mathcal{U}_i$. Then no p.p.t. TM $\mathcal{M}$ that can access all $\mathcal{B}$'s views, and that possesses two coins C_i, C_j, two users' withdrawal sets $\mathcal{W}_1, \mathcal{W}_2$ $(\mathcal{U}_1 \neq \mathcal{U}_2)$ and withdrawals W_i, W_1, W_2, such that C_i is the coin originating from $W_i \in \mathcal{W}_1$, and C_j is a coin originating from either $W_1 \in \mathcal{W}_1$ or $W_2 \in \mathcal{W}_2$, can distinguish (non-negligibly better (in n) than random guessing) whether C_j came from W_1 or W_2. (This theorem also guarantees unlinkability among coins of the same user).

For simplicity, we argue the theorem's validity for $u = 0$ (first variant).

Since all random numbers of $\mathcal{U}_1, \mathcal{U}_2$ (that determine $\mathcal{B}$'s views) are chosen independently of the users' identities, the fact $W_i \in \mathcal{W}_1$ cannot help $\mathcal{M}$. Hence the above problem is equivalent to saying that $\mathcal{M}$ cannot distinguish whether C_j came from W_1 or W_2, for any coin C_j and withdrawals W_1, W_2.

We concentrate on the information revealed in the coin authentication protocol, since the denomination revelation remains unchanged from [Oka95]. Assume that $\mathcal{B}$ has access to an oracle that allows it to distinguish which withdrawal protocol the coin C_j came from. We show that $\mathcal{B}$ can use this oracle to break the *Factoring and Diffie-Hellman II* assumption.

Given $[Y_0'(\equiv g_1^{q_0} \bmod Q), N_0(= p_0 q_0)], [Y_1'(\equiv g_1^{q_1} \bmod Q), N_1(= p_1 q_1)]$ and $[I_r(\equiv g_1^{p_r} \bmod Q), I_{1-r}(\equiv g_1^{p_{1-r}} \bmod Q)]$ $(r \in_R \{0, 1\})$, the bank calculates the views:

- W_i (withdrawal view for $(I_W)_i = I_i,\ i \in \{r, 1-r\}$):

Pick $w_i, c_i' \in_R Z_Q$.
Compute $a_i' = g^{w_i}$, $b_i' = (I_i g_2)^{w_i}$, $r_i' = c_i' x + w_i$.
The range-bounded commitment can be simulated by setting $U_l = g_1^{u_l}(I_W)_i^{-e_l}$.

- P_j (payment view for Y_j', N_j, $j \in \{0,1\}$):

  Compute $Y_j = (Y_j')^{x'} (\equiv g_2^{q_j} \bmod Q)$, where $g_2 = g_1^{x'}$. The range-bounded commitment can again be simulated in the same way as above; in this case the simulator also modifies the output of the hash function $\mathcal{H}_0$, but since the latter is a random oracle the simulation is computationally indistinguishable from the real protocol (a similar technique is used in, e.g., [?]).

  Compute $A_j = g_1^{N_j} Y_j, z_j = A_j^x$.
  Pick $f_j \in_R Z_Q$, and compute $a_j = g^{f_j}, b_j = A_j^{f_j}$.
  Compute $c_j = \mathcal{H}(A_j, N_j, z_j, a_j, b_j)$ and $r_j = c_j x + f_j$.

It is easy to see that $\mathcal{B}$'s views of the above protocols are valid. Furthermore, if $\mathcal{B}$ can use the oracle to see which of W_i corresponds to, e.g., C_0, then it can break the Factoring and Diffie-Hellman II assumption.

All we need to show now is that the views $\mathcal{B}$ constructed are valid views of coins, i.e. there exists a set of choices that any user $\mathcal{U}$ could have made in order to obtain coin C_j (of payment P_j) after engaging in W_i (for some $i \in \{r, 1-r\}$). Then the oracle does provide a valid response.

But for any pair (W_i, P_j), $\mathcal{U}$ could choose v_1, v_2 such that $v_1 = c_j/c_i'$ and $f_j = w_i v_1 + v_2$. Then it is easy to verify that $\mathcal{U}$ would end up with the same coin $\mathcal{B}$ simulated in P_j, after engaging in W_i.

7 Efficiency

The efficiency of our scheme and comparison to [Oka95], partially discussed in the introduction and abstract, is fully discussed in Appendix C.

It is apparent that the storage, computation and communication requirements of our scheme are well within the power of current smart cards, resulting in the first untraceable divisible off-line electronic cash scheme that can implemented in practice.

The most important open problem (excluding extensions to tamper-resistant devices and escrowing) is to find a way to break the linkability between portions of the same coin.

References

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. *First ACM journal on Computer and Communications security*, 1993.

[Bra93a] S. Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS–R9323, CWI (Centre for Mathematics and Computer Science), Amsterdam, 1993. anonymous ftp: ftp.cwi.nl:/pub/CWIreports/AA/CS-R9323.ps.zip.

[Bra93b] S. Brands. Untraceable off-line cash in wallets with observers. In *Advances in Cryptology — Crypto '93, Proceedings (Lecture Notes in Computer Science 773)*, pages 302–318. Springer-Verlag, 1993.

[CFN90] D. Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *Advances in Cryptology —Crypto '88 (Lecture Notes in Computer Science)*, pages 319–327. Springer-Verlag, 1990.

[Cha83] D. Chaum. Blind signatures for untraceable payments. In D. Chaum, R.L. Rivest, and A. T. Sherman, editors, *Advances in Cryptology. Proc. Crypto'82*, pages 199–203, Santa Barbara, 1983. Plenum Press N. Y.

[DC94] S. D'amingo and G. Di Crescenzo. Methodology for digital money based on general cryptographic tools. In *Advances in Cryptology, Proc. of Eurocrypt '94*, pages 157–170. Springer-Verlag, 1994. Italy, 1994.

[EO94] T. Eng and T. Okamoto. Single-term divisible electronic coins. In *Advances in Cryptology — Eurocrypt '94, Proceedings*, pages 306 – 319, New York, 1994. Springer-Verlag.

[FGY96] Y. Frankel, P. Gemmell, and M. Yung. Witness-based cryptographic program checking and robust function sharing. In *Proceedings of the twenty eighth annual ACM Symp. in Theory of Computing, STOC*, 1996.

[FY93] M. Franklin and M. Yung. Secure and efficient off-line digital money. In *Proceedings of the twentieth International Colloquium on Automata, Languages and Programming (ICALP 1993), (Lecture Notes in Computer Science 700)*, pages 265–276. Springer-Verlag, 1993. Lund, Sweden, July 1993.

[Oka95] T. Okamoto. An efficient divisible electronic cash scheme. In Don Coppersmith, editor, *Advances in Cryptology, Proc. of Crypto '95 (Lecture Notes in Computer Science 963)*, pages 438–451. Springer-Verlag, 1995. Santa Barbara, California, U.S.A., August 27–31.

[OO92] T. Okamoto and K. Ohta. Universal electronic cash. In *Advances in Cryptology — Crypto '91 (Lecture Notes in Computer Science)*, pages 324–337. Springer-Verlag, 1992.

[Pai93] J. C. Pailles. New protocols for electronic money. In *Proceedings of Ausicrypt '92*, pages 263–274, 1993.

[PW92] B. Pfitzmann and M. Waidner. How to break and repair a 'provably secure' untraceable payment system. In J. Feigenbaum, editor, *Advances in Cryptology, Proc. of Crypto '91 (Lecture Notes in Computer Science 576)*, pages 338–350. Springer-Verlag, 1992.

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

figure=tree.ps

Figure 1: Tree for a \$1,000 coin.

A Remarks on Assumptions

1. An assumption similar to *Factoring and Diffie-Hellman II* appears in [Oka95]. It implies that the Discrete Logarithm Assumption (DLA) holds and that factoring is difficult, since if either can be solved the assumption doesn't hold. If $u \neq 0$, i.e. the second variant of our scheme is used, then this assumption needs to be modified as follows:
   (Multiple Factoring and Diffie-Hellman) (Let $Q, H, \delta, p_0, q_0, p_1, q_1, N_0, N_1, Y_0, Y_1, I_r, I_{1-r}$ be defined as previously, and $u_0, u_1 \in_R Z_q$). No p.p.t. TM can, given $Q, \delta, g, [Y_0, N_0, Y_0'(\equiv g^{u_0 q_0} \bmod Q)], [Y_1, N_1, Y_1'(\equiv g^{u_1 q_1} \bmod Q)]$ and $[I_r, I_r'(\equiv g^{u_r} \bmod Q)], [I_{1-r}, I_{1-r}'(\equiv g^{u_{1-r}} \bmod Q)]$ ($r \in_R \{0,1\}$), compute r with probability better than $1/2+1/H^c$, for all constants c and sufficiently large H.

   We believe that this assumption represents an interesting number theoretic problem to be studied.

2. An assumption equivalent (see footnote on this assumption) to *Withdrawal protocol* appears in [Bra93b]. Although it is stronger than the DLA, there are convincing arguments that suggest that breaking it requires breaking either the Schnorr signature scheme or the DLA.

3. The *Hash functions* assumption is difficult to guarantee. As suggested by [Oka95] it requires tamper-free devices. [BR93] suggest an implementation using MD5 in a special manner. We use it because it clarifies our scheme, and, for all practical purposes, commonly available one way hash functions can be used.

B Binary tree approach

In all anonymous off-line truly divisible e-cash schemes, a binary tree approach is used. Each node of the tree represents an amount; the root represents the whole amount, its children half of the amount, its "grand-children" a quarter of the amount, etc. This approach limits the size of the coin, by requiring that the bank authenticates only the root of the tree.

For example, in figure 1 we show a tree for the coin n. In this tree if, say, $n_0 = \$1,000$, then $n_{00} = n_{01} = \$500$ and $n_{000} = n_{001} = n_{010} = n_{011} = \$250$.

For a tree construction to work properly, we must ensure that the *same node rule* and the *root route rule* (see Section 3) are satisfied. These two rules guarantee that a user cannot spend more than the total value of the coin (i.e. the denomination of the tree's root). In the context of off-line electronic cash, "cannot spend" means that if these nodes are spent, then the user's identity is revealed.

C Efficiency

We examine the efficiency when $H = |p| = |q| = 256$, $n = k = \epsilon = 40$, $S = 160$, $|N| = 512$[9], $|Q| = 640$ (i.e. $\delta = 0.24$), and the binary tree has 18 levels, i.e. the divisibility precision is 2^{17}, hence sufficient to divide a \$1,000 coin down to 1 cent. We assume the existence of fast, random hash functions. No pre-processing is assumed (unless explicitly stated). In practice several of the steps can be pre-computed. We calculate for $u = 0$. *A * marks the numbers that have been used in the abstract and introduction.*

Storage requirements: The information $\mathcal{U}$ needs to store for one coin $(p, q, (a, b, r))$ is 304* Bytes (up to 464 Bytes if $\mathcal{U}$ stores, rather than recalculating before each payment, A and/or z, and plus 80 Bytes if $u \neq 0$). In comparison, the coins in [Oka95] are 264 Bytes and in [Bra93b] 384 Bytes, when the same parameters are used.

Computation and communication: Our exponentiations are 5 times less costly than [Oka95]. The exponentiations in our range-bounded commitment are 10 times less expensive. At withdrawal $\mathcal{U}$ performs the equivalent of less than 2* exponentiations of [Oka95] (and $\mathcal{B}$ less than 1*). $\mathcal{U}$ sends 440 Bytes, and $\mathcal{B}$ 306 Bytes. $\mathcal{U}$ also needs to calculate one Williams integer, but he can pre-compute one any time before withdrawal. In contrast, in [Oka95] $\mathcal{U}$ needs to perform more than 4,000* "multi-exponentiations" for the same functionality.

In the coin authentication phase, $\mathcal{U}$ transmits 664 Bytes to $\mathcal{R}$. $\mathcal{U}$ needs to perform the equivalent of less than 1* exponentiation of [Oka95] (if he re-computes A and z) and $\mathcal{R}$ 1* (the $|N|$ divisions are as costly as 3 exponentiations, or 3/5 exps of [Oka95]).

In the denomination revelation phase[10], 9 nodes (on average) are paid. For each node, two 512 bit values are sent to $\mathcal{R}$, for a total of 1,152 Bytes. In addition, about 320 Bytes (on average) are sent for verifying that N is a Williams integer. Both $\mathcal{U}$ and $\mathcal{R}$ compute approximately 19* roots (mod N) (we include the Williams integer verification), where each root computation is similar to an exponentiation (mod N).

[9] Although we believe that 512 bits are not sufficient for an RSA modulus, we use this value for comparison with [Oka95]. However our coin remains small if, e.g. $|N| = 1024$ (+300 Bytes).

[10] We adopt the calculations of [Oka95].

D The withdrawal protocol

The user $\mathcal{U}$		The bank $\mathcal{B}$
$p \equiv 3 \bmod 8, q \equiv 7 \bmod 8$		
$B = [N = pq, Y = g_2^q]$ $I_W = g_1^p$	$\xrightarrow{I', I_W}$	$I = I_W I'$
		$I_W \overset{?}{\neq} \{1, g_1\}, I g_2 \overset{?}{\neq} 1$
Proves knowledge of representation of I' w.r.t. g_3	———	Verifies knowledge
Proves I_W is a bounded commitment	———	Verifies proof
		$w \in_R Z_Q$
$A = (Ig_2)^q$	$\xleftarrow{a', b'}$	$a' = g^w, b' = (Ig_2)^w$
$z = z'^q = h_1^{pq} h_2^q h_3^{uq}$ $[= (Ig_2)^{xq}]$		
$v_1, v_2 \in_R Z_Q$		
$a = a'^{v_1} g^{v_2}$		
$b = b'^{q v_1} A^{v_2}$		
$c = \mathcal{H}(A, B, z, a, b)$		
$c' = c / v_1$	$\xrightarrow{c'}$	
$CHECK?$	$\xleftarrow{r'}$	$r' = c'x + w$
$g^{r'} = h^{c'} a'$		
$(Ig_2)^{r'} = z'^{c'} b'$		
Compute: $r = r' v_1 + v_2$		