Università degli Studi di "Roma Tre"

Tesina per il corso Elementi di crittografia
della prof Rosaria Rota

Contanti Elettronici

Marco Quartullo
Luca Proietto
Claudio Neri
Leonardo Puleggi

Indice

1 Introduzione

Con il termine contanti elettronici (o moneta digitale, o anche digital cash) ci si riferisce a delle tecnologie atte ad implementare pagamenti elettronici anonimi, andando così a creare un corrispondente elettronico del denaro comunemente utilizzato.

I sistemi di pagamento odierni possono essere visti come appartenenti a due categorie: quelli basati su acconto (come carte di credito, ordini di pagamento e conti bancari) e quelli basati su oggetti (come denaro, carte prepagate o francobolli). Alcune caratteristiche che possiamo notare per i due sistemi sono la necessità per i primi di identificare gli utenti partecipanti alle transazioni, e la facilità d'uso dei secondi, unita però ad un più elevato rischio di perdite dovute allo smarrimento degli oggetti su cui i pagamenti si basano. Un sistema di pagamento elettronico si propone di ottenere le caratteristiche di entrambi i sistemi, risolvendone i problemi e colmandone le lacune. Con il termine moneta digitale definiamo dunque un sistema di pagamento anonimo e basato su oggetti, che permette di avere sicurezza verso la perdita di denaro e contemporaneamente di proteggere la privacy degli utenti, mantenendo la facilità d'uso tipica del denaro tradizionale.

Nel prosieguo, vedremo dapprima alcune caratteristiche generali di un sistema di digital cash; successivamente presenteremo le teorie matematiche alla base degli algoritmi impiegati nei sistemi in oggetto, per poi scendere nel dettaglio delle tecnologie e degli algoritmi d'interesse.

Caratteristiche del denaro digitale:

Un sistema di digital cash è progettato prendendo a modello il sistema di pagamento in denaro, dunque deve possedere alcune caratteristiche tipiche di esso, e rendere possibili le stesse operazioni.

Per quanto concerne le operazioni che coinvolgono il denaro digitale, ve ne sono tre: il prelievo, il pagamento, e il deposito; queste definiscono tre tipi di attori del sistema: una rete finanziaria, un utente pagante e un utente che riceve il pagamento. Tutti i sistemi di digital cash dovranno confrontarsi con questa struttura generale, implementando i vari tipi di transazioni e considerando le interazioni tra i diversi tipi di utenti

Le caratteristiche e proprietà utili per il denaro digitale sono modellate su quelle già appartenenti al denaro tradizionale. Le vediamo nel dettaglio:

Sicurezza

Il denaro digitale non può essere copiato o riutilizzato; si dovrà dunque minimizzare il rischio di falsificazioni e instaurare un sistema di autenticazione.

Contraffazione:

Analogamente al denaro tradizionale, si hanno due principali tipi di contraffazione:

- Riproduzione della moneta: si crea una "moneta" valida per il sistema senza effettuare un corrispondente prelievo bancario
- Spesa multipla: si utilizza la stessa moneta più volte.

Per proteggersi contro la riproduzione delle monete si fa affidamento sull'autenticazione degli utenti e sull'integrità dei messaggi. Per evitare la spesa multipla, invece, la banca (o comunque, la generica rete finanziaria) mantiene una base di dati delle monete spese, e rifiuta di depositare monete già spese; tuttavia una simile soluzione è effettivamente utile solamente se i pagamenti sono in linea (vedremo tra poco cosa questo comporti), mentre se sono fuori linea si può al massimo risalire a quando è accaduta la spesa multipla.

Autenticità:

Per far fronte ai problemi di sicurezza è necessario stabilire varie contromisure:

- Identificazione degli utenti: durante una transazione, gli utenti devono sapere con chi stanno trattando.
- Integrità dei messaggi: si deve avere la certezza che il messaggio arrivato sia proprio il messaggio originale, senza modifiche.
- Non ripudiabilità: per evitare che un utente possa negare di aver partecipato ad una transazione anche se questa è effettivamente avvenuta.

L'autenticità si ottiene attraverso la gestione di chiavi, con l'appoggio di autorità di certificazione (CA, Certification Authority) fidate, senza le quali la sicurezza di un sistema come il digital cash sarebbe praticamente impossibile da ottenere con un mezzo di trasmissione non sicuro come Internet.

Privacy

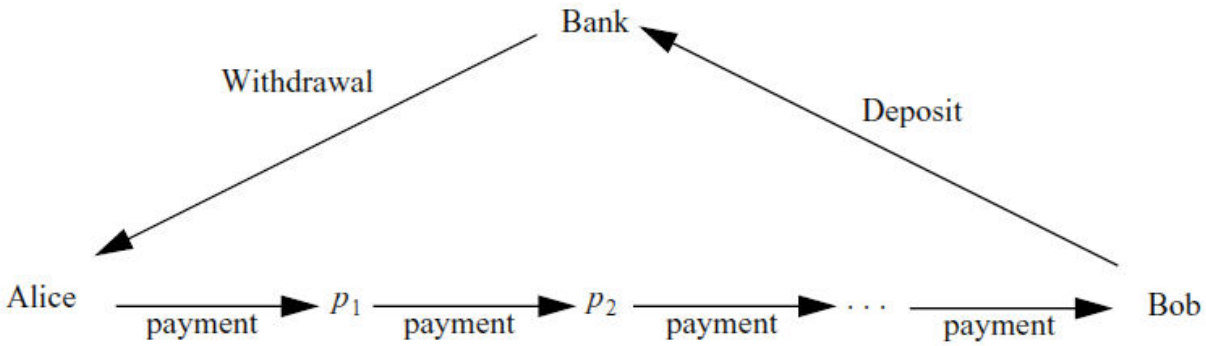
Come avviene per il denaro tradizionale, anche quello digitale si vuole che sia anonimo, nel senso che non deve essere possibile risalire, a partire da esso, agli utenti che lo hanno utilizzato. Ciò comunque non inficia la possibilità di autenticare il denaro, sebbene l'anonimato incrementi il rischio di attività illegali come il riciclaggio, la contraffazione e il ricatto. Maggiore privacy significa dunque minor sicurezza, e viceversa: sta ai diversi sistemi di digital cash scegliere il livello di compromesso tra le due opposte esigenze.

Portabilità

Ovviamente le caratteristiche di utilizzo e sicurezza del denaro digitale non devono essere dipendenti da alcuna locazione fisica.

Trasferibilità

La proprietà del denaro di essere trasferibile consente agli utenti di spenderlo senza dover contattare la banca. Un pagamento si dice essere un trasferimento se chi lo riceve è in grado di utilizzare il denaro ricevuto in un altro pagamento. Un sistema di pagamenti si dice trasferibile se consente almeno un trasferimento per ogni moneta. Il ciclo di vita di una moneta elettronica può essere esemplificato dunque dalla seguente figura:



I problemi che possono occorrere nei sistemi trasferibili sono i seguenti:

- Qualsiasi sistema di contanti elettronici ha la proprietà che il denaro deve crescere in dimensioni ogni volta che viene speso, a causa delle informazioni che deve contenere. Queste riguardano le persone che hanno speso il denaro, utili alla banca per essere in grado di identificare chi abbia effettuato spese multiple. Ciò limita il massimo numero di trasferimenti permessi in base alla massima dimensione che la moneta può avere.
- Il riciclaggio e l'evasione di tasse sono difficili da identificare, dal momento che le transazioni non vengono registrate.
- Ogni trasferimento ritarda l'identificazione di spese multiple, dal momento che esse non verranno identificate finché due copie della stessa moneta non verranno depositate.
- Gli utenti possono riconoscere le proprie monete se le vedono in un altro pagamento.

Divisibilità

Per divisibilità si intende l'abilità di effettuare dei cambi per le monete, ad esempio utilizzando solo frazioni di esse. Ciò consente una maggior flessibilità e praticità nelle transazioni.

Pagamenti fuori linea

I protocolli di gestione del denaro digitale possono essere implementati in due modi: in linea o fuori linea, sebbene il sistema ideale sia quello in grado di consentire anche pagamenti fuori linea.

Per pagamento fuori linea intendiamo una transazione per la quale chi riceve denaro ne può effettuare il deposito e la relativa verifica di autenticità dopo diverso tempo dal pagamento stesso, che dunque può avvenire senza che necessariamente la banca faccia da intermediario. I sistemi fuori linea sono ovviamente preferibili da un punto di vista pratico, ma sono anche maggiormente vulnerabili al problema della spesa multipla rispetto a quelli in linea.

Nei sistemi in linea, in ogni pagamento la banca fa da intermediario, verificando la validità del denaro utilizzato (un po' come avviene per le odierne carte di credito). Essi garantiscono così una maggior sicurezza, ma richiedono la comunicazione con la banca (che può rivelarsi onerosa); inoltre non proteggono l'anonimato degli utenti, e non rispettano la proprietà del denaro di essere trasferibile.

2 Basi Matematiche e Algoritmiche

In questa seconda parte della tesina illustreremo le teorie matematiche e gli algoritmi alla base dei sistemi di digital cash; alcuni concetti, comunque, verranno solo citati superficialmente, in quanto già ampiamente trattati in altri lavori.

Fondamenta matematiche

Forniamo qui un elenco di alcuni dei concetti matematici che risulteranno essere importanti nei nostri algoritmi, mentre ne tralasceremo diversi altri (come quelli di gruppo, di modulo o di generatore) già approfonditi in altre tesine.

Definizione

$a \in Z_n^*$ è un *quadrato residuo* modulo *n* se esiste una $x \in Z_n^*$ tale che risulti $x^2 \equiv a \bmod n$. L'insieme di tutti i residui quadratici modulo *n* è indicato con Q_n.

Fatto

Sia *n* il prodotto di due numeri dispari distinti e primi *p* e *q*: *n* = *pq*. Allora $a \in Z_n^*$ è un *quadrato residuo* modulo *n* se e solo se $a \in Q_p^*$ e $a \in Q_q^*$. Ne consegue che $|Q_n| = |Q_q||Q_p| = ((p-1)(q-1))/4$.

Definizione

Sia $a \in Q_n$. Se $x \in Z_n^*$ soddisfa $x^2 \equiv a \bmod n$, allora *x* è una *radice quadrata* di modulo *n*.

Il predicato della residuità quadratica definisce una relazione di equivalenza in Z^*_n (i quadrati residui modulo *n* formano una classe di equivalenza $\sim_x$)

Esiste un algoritmo efficiente, basato sul teorema di Eulero, per decidere la residuità quadratica modulo interi con fattorizzazione conosciuta.

Nel caso in cui invece la fattorizzazione non sia conosciuta, qualche informazione circa la residuità quadratica può essere data dal simbolo di Jacobi.

Definizione

Il *simbolo di Jacobi* (y / x) è definito come

$$(y/x) = \prod_{i=1}^{k} (y/p_i)^{h_i}$$

dove i p_i sono primi distinti e gli h_i sono interi positivi.

Il simbolo di Jacobi può essere computato in tempo polinomiale anche se la fattorizzazione di *n* non è data, e fornisce qualche informazione circa la residuità quadratica di *y*. Se (*y* / *x*) = -1 allora sicuramente *y* è un quadrato non residuo modulo *x*. Non si conosce alcun algoritmo efficiente per decidere la residuità quadratica se il simbolo di Jacobi vale +1: in questo caso la via più "veloce" consiste nel fattorizzare *n*.

Definizione

Se $p = 3 \bmod 4$ e $q = 3 \bmod 4$, con *p* e *q* interi e primi, allora $n = pq$ è un *intero di Blum*. Più in generale, un intero *x* è un intero di Blum se e solo se

$$x = p^{k_1} q^{k_2}$$

dove *p* e *q* sono primi differenti ed entrambi $\equiv 3 \bmod 4$, e k_1 e k_2 sono interi positivi dispari. Con *BL* denotiamo l'insieme degli interi di Blum, e con $BL(n)$ il sottoinsieme degli interi di Blum prodotto di due primi (ossia $k_1 = k_2 = 1$) congrui 3 mod 4 entrambi di lunghezza *n*. Esiste un algoritmo efficiente in grado di fornire la fattorizzazione di un intero random $x \in BL(n)$ in tempo polinomiale rispetto ad *n*. Inoltre, è computazionalmente facile calcolare la radice quadrata di un residuo quadrato modulo un intero di Blum.

Definizione

$N = pq$ è un *intero di Williams* se $p = 3 \bmod 8$ e $q = 3 \bmod 8$. Un intero di Williams è un tipo specifico di intero di Blum, e ne conserva tutte le proprietà.

Fatto (*assunzione della residuità quadratica*)

La classe degli interi di Blum prodotto di due primi della stessa lunghezza costituisce il più difficile input per ogni conosciuto algoritmo efficiente di fattorizzazione; più precisamente, nessun algoritmo efficiente può dare il valore del predicato della residuità quadratica con probabilità maggiore della scelta casuale.

RSA

RSA è uno degli algoritmi utilizzati nell'ambito dei sistemi per i contanti elettronici. Senza voler entrare troppo nel merito dell'argomento, in quanto già approfondito in altri lavori, ricordiamo solamente che RSA è un algoritmo a chiave pubblica (in cui cioè ogni utente possiede una chiave privata ed una pubblica con cui cifrare e decifrare messaggi) e risulta adatto per essere usato sia nell'ambito della crittografia dei dati che in quello della firma digitale. Si basa sull'aritmetica modulare, e fonda la sua sicurezza sulla difficoltà di fattorizzare grandi numeri primi. La sua affidabilità lo ha reso, al giorno d'oggi, uno degli standard de facto nella crittografia.

Firma Digitale

La firma digitale, proposta per la prima volta nel 1976 da Whitfield Diffie, è l'equivalente elettronico della firma autografa; si basa sulla crittografia a chiave pubblica, in un sistema per cui un messaggio viene firmato da un utente cifrandolo con la sua chiave privata, e riconosciuto come valido dagli altri attraverso la decifrazione del messaggio stesso con la chiave pubblica del

firmatario. Sono stati proposti differenti schemi per l'implementazione della firma digitale, tra cui uno basato su RSA. Nell'uso comune, data la lentezza della cifratura a chiave pubblica, si utilizzano funzioni di message digest per generare "impronte" di messaggi sulle quali applicare, più velocemente, la firma digitale. È facile capire come un meccanismo di firma elettronica sia di fondamentale importanza in un sistema che si propone di gestire denaro elettronico: basti pensare che attraverso esso siamo in grado di soddisfare una proprietà, quella della non ripudiabilità, essenziale nell'ambito delle transazioni finanziarie.

Firma cieca

Una firma cieca è un tipo speciale di firma digitale. La differenza non sta nella firma in sé, in realtà, ma nel documento a cui essa viene apposta. Quando qualcuno firma elettronicamente un documento, normalmente è a conoscenza dei suoi contenuti; chi vi appone una firma cieca, invece, non sa cosa è presente su di esso. Le firme cieche garantiscono lo stesso tipo di autenticazione delle normali firme, ma lo fanno in modo non identificabile: chi riceve il messaggio è sicuro che la trasmissione è autentica ed affidabile, ma non ne conosce il mittente.

Vediamo il funzionamento della firma cieca attraverso un esempio: un utente porta un documento ad un notaio, e vuole che nessuno (incluso il notaio) venga a conoscenza dei contenuti di esso. L'utente allora sigilla il documento in un involucro. Una porzione del documento è visibile attraverso l'involucro, e il notaio appone un timbro su di essa, come prova di autenticità. Quando la firma cieca viene utilizzata, le tecniche crittografiche sostituiscono l'involucro e il timbro: l'utente cifra il documento digitale (atto analogo. nell'esempio, ad inserirlo in un involucro), e il notaio appone una firma digitale (il timbro) sul documento nell'involucro; quando il documento verrà esaminato per controllarne la validità, verrà convalidata la firma del notaio.

Vediamo come è possibile implementare la firma cieca utilizzando RSA, utilizzando questa volta come esempio quello a noi più vicino della realtà finanziaria:

- Un utente A sceglie un "fattore di accecamento" casuale r tale che risulti gcd(r, n) = 1 (con gcd massimo comun denominatore); la banca ha chiave pubblica (n, e) e chiave privata (n, d). L'utente A presenta alla banca non già il messaggio originale m, ma un messaggio modificato m' codificato come

  $m' = mr^e \bmod n$

- La banca firma il messaggio:

  $s' = (m')^d \bmod n = (mr^e)^d \bmod n$

- L'utente A divide da s' il fattore di accecamento:

  $s = s' / r \bmod n$

- Ora l'utente A può utilizzare

  $s = m^d$

  per le sue operazioni.

Dal momento che r è un numero casuale, la banca non è in grado di determinare m, e dunque non può collegare la firma con le operazioni dell'utente stesso: in tal modo viene garantita la proprietà che ci interessava di avere autenticazione senza identificazione dell'utente.

La sicurezza di questo schema di firma è garantita dalla difficoltà della fattorizzazione e dell'estrazione di radici; in ogni caso, dal momento che il numero r è casuale, la firma rimane "cieca" indipendentemente da questi problemi.

Metodo cut-and-choose

Il protocollo cut-and-choose venne utilizzato per la prima volta da Michael Rabin nel 1978. Brevemente, il suo funzionamento si basa su tre passi:

1. L'utente A taglia l'oggetto di interesse in due metà
2. L'utente B sceglie una delle due
3. L'utente A prende la metà rimanente

Anche se molto semplice, l'algoritmo riesce comunque ad assicurare, in caso di necessità, che l'oggetto da dividere sia effettivamente partizionato equamente dall'utente A, in quanto egli non sa quale delle due parti sceglierà l'utente B.

Secret Sharing

Il secret sharing è una tecnica che risulta utile quando si vuole dividere un messaggio segreto tra differenti persone. Solamente se *m* parti del messaggio sono disponibili esso può essere ricostruito. Ci sono diversi algoritmi per ottenere il secret sharing; il più semplice di essi è detto della *soglia* (*m, n*), e consiste nel dividere il messaggio in *n* parti (con $n > m$), in modo tale che una qualsiasi combinazione di *m* di essi possa portare a ricostruire il messaggio.

Protocollo di Bit Commitment

I protocolli di bit commitment sono stati sviluppati per evitare la possibilità di modificare risposte, e questo problema risulta avere una certa importanza nel contesto del digital cash. Illustriamo meglio la questione attraverso un esempio: supponiamo che una persona voglia far credere ad un amico di sapere in anticipo quale partito, tra due partecipanti alle elezioni, sarà il vincitore. Potrebbe allora scrivere la risposta (consistente in un bit) su di un file, cifrarlo e consegnarlo all'amico; al termine delle elezioni darebbe poi all'amico una chiave con cui decifrare il file per conoscere la risposta. Il problema è che si potrebbe barare utilizzando due differenti chiavi k_1 e k_2 tali che ognuna di esse, applicate al file, fornisca una risposta diversa (proprio quella corretta). I protocolli di bit commitment sono progettati proprio per evitare questo tipo di inganni: ne vediamo alcune possibili implementazioni:

Bit commitment con crittografia simmetrica

- L'utente B genera una stringa casuale, *R*, e la manda all'utente A
- L'utente A manda il messaggio *b* cifrato con la chiave *K* e contenente anche la stringa *R* dell'utente B: E_K (*R*, *b*)
- L'utente A invia poi all'utente B la chiave *K* quando è il momento di rivelare il messaggio *b*
- L'utente B decifra il messaggio, che giudicherà valido se conterrà proprio la stringa *R* da lui originariamente scelta.

Bit commitment con funzioni one-way

- L'utente A genera due stringhe casuali di bit R_1 e R_2

- L'utente A crea un messaggio contenente R_1, R_2 e il bit d'interesse b: (R_1, R_2, b)
- L'utente A calcola la funzione one-way scelta sul messaggio ed invia il risultato, insieme ad una delle due stringhe casuali, all'utente B: $H(R_1, R_2, b)$, R_1
- L'utente A invia poi all'utente B l'intero messaggio, una volta venuto il momento: (R_1, R_2, b)
- L'utente B calcola la funzione one-way del messaggio e la confronta, insieme alle stringhe casuali, a quanto aveva ricevuto in precedenza: il bit è dunque valido se i messaggi coincidono.

Bit commitment con generatori di sequenze pseudo casuali

- L'utente B genera una stringa casuale di bit e la manda all'utente A
- L'utente A genera un seme casuale per un generatore di bit pseudo casuale. Poi, per ogni bit della stringa appena ricevuta, invia all'utente B l'output del generatore se il bit considerato è 0, e lo XOR tra l'output del generatore e il bit b da trasmettere se invece il bit considerato è 1.
- L'utente A invia, al momento opportuno, il seme casuale all'utente B
- L'utente B calcola infine la stringa casuale derivata dal passo 2 e verifica che sia quella corretta, assicurandosi così della validità del bit b.

Problema del logaritmo discreto e schema di firma basato su esso

Questo problema è alla base del sistema di digital cash progettato da Brands.

Siano p e q primi, e tali che $q \mid (p - 1)$ (cioè q è un fattore primo di $(p - 1)$. Sia g un generatore. Allora, dati p, q, g e y, il problema consiste nel trovare l'unico intero a, con $0 \leq a \leq q - 1$, tale che

$$g^a \equiv y \bmod p$$

Ciò che garantisce sicurezza ai sistemi che si basano sul problema del logaritmo discreto è che non esiste alcun algoritmo in grado di risolvere tale problema in tempo polinomiale.

Schema di firma basato sul logaritmo discreto

L'idea per un simile sistema di firma si basa sul fatto che se qualcuno riceve

$$g^a \bmod p$$

dove p è un numero primo grande, g è un generatore e a è un intero, allora ritrovare a è molto difficile. La chiave privata è proprio a, mentre la chiave pubblica è

$$y = g^a \bmod p$$

Il protocollo funziona in questo modo:

- L'utente A vuole firmare un documento m, calcola dunque

  $m^a \bmod p$

  La sua firma è una lista di $m^a \bmod p$, g, e $g^a \bmod p$
- L'utente B vuole verificare la firma; l'utente A deve allora creare un numero casuale w per calcolare

  $g^w \bmod p$

  e

  $m^w \bmod p$

ed inviarli all'utente B.

- L'utente B genera a sua volta un numero casuale c, come sfida, e lo invia all'utente A.
- L'utente A invia come risposta alla sfida:

  $r = ca + w$
- L'utente B calcola poi

  $g^r \bmod p$

  e lo confronta con

  $(g^a)^c \times g^w \bmod p$

  Dovrebbero risultare uguali, e lo stesso dicasi per

  $m^r \bmod p$ e $(m^a)^c \times m^w \bmod p$

  La firma è corretta e non vi sono state manomissioni se

  $(g^a)^c \times g^w \equiv g^{ca+w} \equiv g^r$

  e

  $(m^a)^c \times m^w \equiv m^{ca+w} \equiv m^r$

Schema di firma di Schnorr

Lo schema di firma di Schnorr è un protocollo di tipo zero-knowledge che si basa sulla difficoltà di calcolare logaritmi discreti. Analogamente a quanto visto nella trattazione di questi, la chiave privata è a, con $0 \leq a \leq q - 1$, se p e q sono primi e tali che $q \mid (p - 1)$; viene inoltre scelto un elemento g diverso da 1 tale che $g\text{^}q = 1 \bmod p$. La chiave pubblica è

$$y = g^a \bmod p$$

Si noti che p, q, g, e y sono pubblici, e solo a è segreta

Il protocollo di firma di un messaggio M è il seguente:

- L'utente A sceglie un numero casuale w, con $1 \leq w \leq q - 1$, calcola

  $x = g^w \bmod p$
- L'utente A concatena poi il messaggio M ed x, e calcola una funzione hash (una funzione one-way, descritta brevemente nel seguito) del risultato

  $e = H(M,x)$
- L'utente A calcola ancora

  $v = (w + ae) \bmod q$

  ed invia la firma (v, e) all'utente B, insieme ad x ed M
- L'utente B verifica allora che

  $g^v = x * y^e \bmod p$

  poi controlla che la funzione hash calcolata sulla concatenazione di M e x sia proprio pari ad e

  $e = H(M,x)$

  Se ciò accade, la firma è valida.

La prova che la verifica della firma funziona sta nel fatto che, se la firma è effettivamente quella

dell'utente A, allora

$$g^v \equiv g^w g^{ae} \equiv xy^e$$

per come sono state definite le variabili.

Problema della rappresentazione in gruppi del primo ordine

Questo problema estende la difficoltà computazionale del problema del logaritmo discreto al calcolo di un insieme di numeri collegati a logaritmi discreti (e in tal modo fornendo anche maggior flessibilità d'uso). Il problema consiste nel trovare, a partire da un gruppo primo G, una sequenza di generatori (g_1, g_2, ..., g_k) con $k \geq 2$, $g_i \in G$ e un elemento fissato $h \in G$, una rappresentazione di h tale che

$$h \bmod p = \prod_{i=1}^{k} g_i^{a_i}$$

dove a_i è un intero.

Trovare una rappresentazione di h è difficile, tranne nel caso in cui prima scegliamo (g_1, g_2, ..., g_k) e (a_1, a_2, ..., a_k) e poi calcoliamo h: questa assunzione è utilizzata nel sistema di contanti elettronici di Brands.

Funzioni Hash

Citiamo solamente per completezza, dacché già descritte in altri lavori, le funzioni hash, in quanto giocano un ruolo fondamentale negli algoritmi di digital cash, in particolare nei sistemi di Brands e di Okamoto. Il loro ruolo è di comprimere un messaggio in input in una stringa di lunghezza fissata, in modo tale che non si possa (o comunque, che sia altamente improbabile) generare lo stesso output a partire da un input diverso dal messaggio stesso, né che dall'output si possa risalire all'input. Calcolano quindi una "impronta" di un messaggio, rendendolo univocamente identificabile.

Crittografia a curva ellittica

In ultimo, descriviamo i principi della crittografia a curva ellittica, una avanzata teoria in grado di migliorare le prestazioni dei sistemi a noi d'interesse. L'utilizzo di curve ellittiche nei sistemi a chiave pubblica è stato inizialmente proposto nel 1985 da Victor Miller, e indipendentemente da Neal Koblitz. L'idea di base è quella di utilizzare gruppi di punti su una curva ellittica nei sistemi esistenti basati sul problema del logaritmo discreto.

I Crittosistemi a Curva Ellittica offrono elevata efficienza ed un basso overhead per la cifratura, la firma digitale e la gestione delle chiavi, proponendosi come la scelta ideale per sistemi di contanti elettronici basati su smart card.

I vantaggi dei Crittosistemi a Curva Ellittica includono:

- La più alta forza crittografica per bit tra tutti i sistemi a chiave pubblica conosciuti.
- Enorme risparmio di calcoli, occupazione di banda e spazio su disco rispetto agli altri i sistemi a

chiave pubblica

- Ulteriore risparmio di banda permesso dalle ridotte dimensioni di firme e certificati
- Processi di cifratura e firma veloci sia in hardware che in software
- Ideale per implementazioni hardware molto piccole, come le smart card
- Permette di separare gli stadi di cifratura e firma per maggior semplicità

Principi della crittografia ellittica

Una curva ellittica è definita da un'equazione della forma

$$y^2 [+xy] = x^3 + ax^2 + b$$

dove x e y sono variabili, a e b sono costanti, e le parentesi quadrate indicano l'opzionalità del termine $+xy$.

La proprietà cruciale di una curva ellittica è che possiamo definire una regola per "aggiungere" due punti appartenenti alla curva per ottenere un terzo punto, anch'esso appartenente alla curva; tale regola soddisfa le normali proprietà dell'addizione.

Le equazioni per la regola dell'addizione sono le seguenti:

- Se il campo è GF(p), dove p è un numero primo grande, la regola dell'addizione ha la forma

  $$(x_1, y_1) + (x_2, y_2) => (x_3, y_3)$$

  dove

  $$x_3 = L^2 - x_1 - x_2$$
  $$y_3 = L(x_1 - x_3) - y_1$$
  $$L = (y_1 - y_2) / (x_2 - x_1)$$

  Se $x_1 = x_2$ e $y_1 = y_2$ dobbiamo utilizzare invece

  $$x_3 = L^2 - 2x_1$$
  $$y_3 = L(x_1 - x_3) - y_1$$
  $$L = (3x_1^2 + a) / (2y_1)$$

- Se il campo è GF(2^m), la regola dell'addizione ha la forma

  $$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

  dove

  $$x_3 = L^2 + L + x_1 + x_2 + a$$
  $$y_3 = L(x_1 + x_3) + x_3 + y_1$$
  $$L = (y_1 + y_2) / (x_1 + x_2)$$

  Se $x_1 = x_2$ e $y_1 = y_2$ dobbiamo utilizzare invece

  $$x_3 = L^2 + L + a$$
  $$y_3 = x_1^2 + (L + 1)x_3$$
  $$L = x_1 + y_1 / x_1$$

Affinché l'addizione sia ben definita per ogni coppia di punti, è necessario includere un punto zero "extra", che non soddisfi l'equazione della curva ellittica. Questo punto zero è considerato un punto

della curva a tutti gli effetti. L'ordine della curva è il numero di punti distinti della curva, incluso il punto zero.

Avendo definito l'addizione tra due punti, possiamo anche definire la moltiplicazione tra un punto P e un intero positivo k come la somma del punto con se stesso ripetuta k volte: $k*P$

La sicurezza del sistema a curva ellittica è basata sulla difficoltà di calcolare k dato F e $k*F$.

Utilizzo della crittografia ellittica

Un insieme di utenti si accordano su una curva ellittica comune e su un punto segreto fissato F. Ogni utente sceglie poi un numero intero segreto, ad esempio l'utente A potrà scegliere un numero segreto A_k, e pubblicherà poi una chiave pubblica pari al prodotto tra la sua chiave segreta e il punto F, ad esempio l'utente A avrà la chiave pubblica $A_p = A_k*F$.

Ora, supponiamo che l'utente A voglia inviare un messaggio all'utente B. Un possibile procedimento consiste nel far sì che l'utente A calcoli semplicemente $A_k * B_p$ e utilizzi il risultato come chiave segreta per un convenzionale sistema di cifratura a chiave simmetrica a blocchi, ad esempio DES. L'utente B può poi calcolare lo stesso numero calcolando $B_k * A_p$, dal momento che risulta

$$B_k * A_p = B_k * (A_k * F) = (B_k * A_k) * F = A_k * (B_k * F) = A_k * B_p$$

Viene detto *punto fisso c* un punto sulla curva tale che

$$c*F = 0$$

dove

$$c = b - a$$

$$a*F = b*F, \qquad b > a$$

L'ultimo c per cui vale ciò è chiamato l'ordine del punto.

Per maggior sicurezza, la curva e il punto fisso sono scelti in modo tale che l'ordine del punto fissato F sia un numero primo grande.

Con le specifiche suddette, se l'ordine del punto fissato F è un numero primo di n bit, allora per calcolare k da $k*F$ e F sono necessarie pressappoco $2^{n/2}$ operazioni.

Questo significa, ad esempio, che se l'ordine di F è un numero primo di 240 bit, allora un attacco avrebbe bisogno, per avere successo, di circa 2^{120} operazioni. Questo è il motivo per cui l'utilizzo delle curve ellittiche è interessante: permette di avere chiavi pubbliche e firme digitali molto più piccole di quelle previste ad esempio per RSA, fornendo il medesimo livello di sicurezza, nonché una maggior velocità nelle operazioni.

3 MONETA ELETTRONICA

3.1 Sistemi di pagamento con moneta elettronica

- Approccio innovativo e di ultima generazione che permette la compravendita direttamente durante la navigazione in Internet.
- La Moneta Elettronica risulta essere a tutti gli effetti un titolo di credito digitale opportunamente firmato da un soggetto bancario.
- La Moneta Elettronica è immateriale ed essendo una sequenza di bit può essere utilizzata per qualsiasi attività di compravendita tramite reti telematiche.
- La memorizzazione di moneta elettronica avviene su supporto fisico (hard disk).
- Esempi di Sistemi che supportano tale tecnologia sono E-Cash e DigiCash.

3.2 Firma cieca

- Le tecniche di firma cieca (blind signature) permettono di far firmare un documento senza rivelare il contenuto.
- Metaforicamente l'effetto della firma cieca corrisponde a mettere un documento all'interno di una busta con un foglio di carta carbone. Se qualcuno firma la busta avrà contemporaneamente firmato anche il documento all'interno della busta senza esserne necessariamente al corrente del contenuto.
- Va sottolineato che la firma rimane sul documento anche quando quest'ultimo viene estratto dalla busta.
- La tecnologia della firma cieca trova largo impiego oltre che nei sistemi di pagamento elettronico con monete virtuali anche nei sistemi che gestiscono votazioni elettorali elettroniche.

3.3 Tipico scenario di prelievo

- Il PC dell'utente su cui è installato il sistema calcola quante monete sono necessarie per ottenere la somma richiesta.

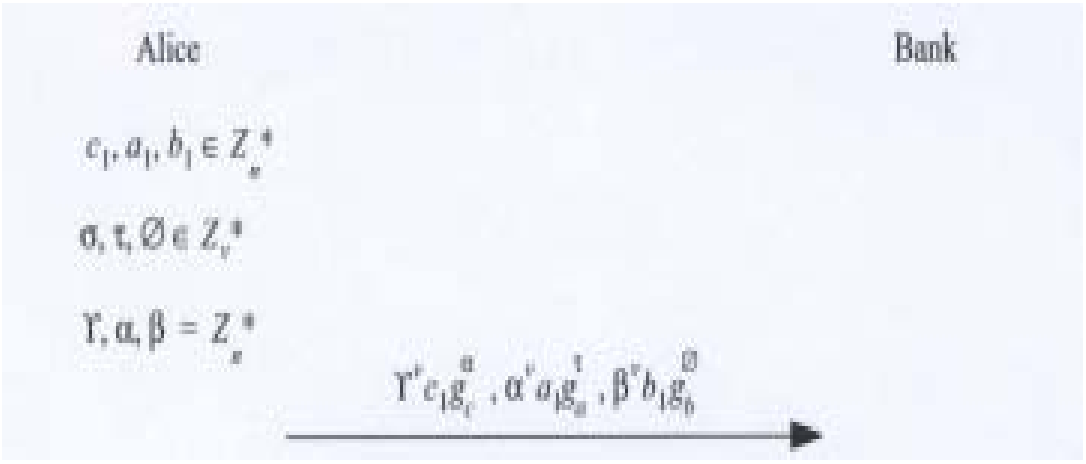
- Lo stesso utente tramite la sua applicazione genera un numero di monete pari alla quantità sopra calcolata, associando ad ognuna di esse un numero di serie casuale. Quindi spedisce queste monete una ad una inserite in una speciale busta. (rappresenta il fattore "cecità").

- La banca codifica i numeri ciechi con la propria chiave segreta (firma digitale), grazie alla proprietà della firma cieca che consente di applicare tale firma attraverso la busta, contemporaneamente la banca addebita sul conto dell'utente la stessa somma.

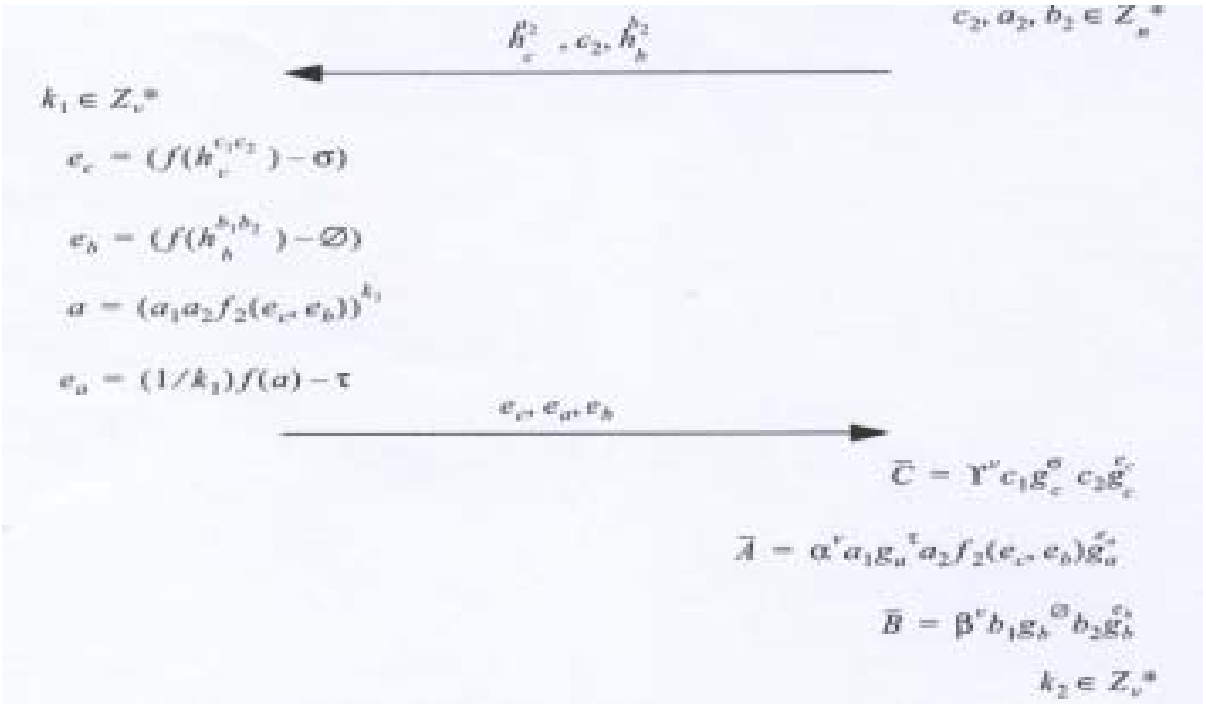
- Le monete autenticate sono restituite all'utente che potrà togliere loro il fattore di cecità introdotto in precedenza, senza alterare la firma della banca. I numeri di serie con le loro relative firme costituiscono la moneta digitale garantita dalla banca.

- Quando l'utente spenderà tali monete, la banca le accetterà poiché da lei autenticate ma non potrà risalire all'identità dell'utente in quanto i numeri di serie delle monete erano nascosti (nella busta) al momento dell'autenticazione da parte della banca stessa.

4 FERGUSON SCHEME



L'utente (nel caso specifico Alice) sceglie i tre suoi contributi casuali che andranno a formare i tre numeri base relativi alla moneta. Inoltre i tre valori numerici vengono opportunamente nascosti da tre fattori di cecità scelti sempre dall'utente stesso.



La banca spedisce all'utente un suo contributo ad un solo numero base ed altri due numeri che sono così calcolati: la base della potenza è nota e i due esponenti sono gli altri due contributi ai numeri base. Quindi in questa fase solo un contributo viene inviato in modo esplicito dalla banca.

In questa fase della comunicazione la banca invia all'utente gli altri due contributi per i numeri base della moneta ed una parte essenziale nel calcolo della firma che permetterà alla moneta di circolare in modo legalmente riconosciuto.

5 MICROMINT

5.1 Introduzione

Uno schema di pagamento Micromint usa completamente operazioni a chiave non pubblica e realizza una sicurezza ragionevole a costi molto bassi. Anche in questo schema troviamo gli stessi partecipante del protocollo Payword. In linea sommaria un broker produce monete che distribuisce ai vari utenti. Questi ultimi le utilizzano per effettuare i pagamenti dai venditori che a loro volta le restituiscono al broker che rimborsa il tutto utilizzando altri mezzi. Una moneta è caratterizzata da una stringa di bit e la sua validità può essere constatata facilmente da qualsiasi partecipante. Risulta particolarmente vantaggioso coniare più monete possibili, in particolare l'investimento economico più consistente lo si effettua per coniare la prima moneta. Le altre sono producibili a costi relativamente bassi. Un broker provvede a coniare un certo numero di monete, tipicamente ogni inizio mese, che distribuisce ai vari utenti; la loro validità scadrà alla fine del mese stesso. Le monete che un utente non consuma possono essere rispedite al broker che le sostituisce con delle nuove. Lo schema di Micromint possiede molte varianti. Qui di seguito descriveremo inizialmente la sua versione base ed infine alcune delle sue varianti.

5.2 Monete come collisioni di funzioni hash

Le monete vengono rappresentate come collisioni di funzioni hash, per alcune specifiche funzioni hash del tipo "one-way" che provvedono a trasformare stringhe x di *m* bit in stringhe y di *n* bit.
Diamo ora alcune definizioni che risulteranno utili.
Una stringa x è preimmagine di una stringa y se h(x)=y.
Una coppia di stringhe distinte (x_1,x_2) ad *m* bit e' una collisione a 2-vie se $h(x_1)=h(x_2)=y$ per qualche y ad *n* bit.
Un modo per produrre una collisione a 2-vie in modo accettabile potrebbe essere quello di effettuare l'hash di $2^{n/2}$ valori di x. Se effettuiamo l'hash per *c* volte su tanti valori di x, quanti sono necessari per produrre la prima collisione, si generano approssimativamente in c^2, con $1<= c <=2^{n/2}$, altrettante collisioni. Ciò vuol dire che una volta trovata la prima collisione produrre le altre non richiede eccessivi tempi di calcolo.

5.3 Monete come collisioni a k-vie

Per rendere facile il lavoro del broker nel generare collisioni a 2-vie si sceglie generalmente un valore *n* abbastanza piccolo. In questo modo però si rende altrettanto facile il lavoro per un intruso che vuole falsificare le monete. La sicurezza si raggiunge scegliendo le collisioni a k-vie. Una collisione a k-vie è un'insieme di k valori distinti $x_1,..,x_k$ che hanno lo stesso valore hash y. In questo modo per trovare una collisione a k-vie, approssimativamente, si dovrebbero esaminare $2^{n*(k-1)/k}$ valori di x. Esaminando *c* volte questi valori, con $1<=c<=2^{n/k}$, ci si aspetta di vedere circa c^k collisioni a k-vie.
La scelta di k influenza il grado di sicurezza contro eventuali falsificazioni.
Inoltre la validità di k monete può essere facilmente verificata dalla relazione $h(x_1)=h(x_2)=..=h(x_k)=y$.

5.4 La coniatura delle monete

Il processo di calcolo h(x)=y equivale a lanciare una biglia in uno tra 2^n recipienti. Per rendere il discorso più semplice supponiamo che una moneta sia proprio una serie di k biglie che lanciate sono entrate nello stesso contenitore. Far si che k biglie lanciate entrino nello stesso contenitore risulta molto difficile, per cui per ottenere ciò se ne dovrebbero lanciare una quantità abbastanza

grossa.
Per coniare una moneta il broker crea 2^n recipienti e lancia circa $k*2^n$ biglie. Alla fine ogni recipiente che contiene almeno k biglie diventerà una moneta. Quindi k biglie scelte a caso formeranno una moneta mentre le altre in genere non vengono utilizzate perché possono risultare utili nel momento in cui qualcuno intende effettuare delle falsificazioni. Inoltre in questo modo si semplifica il lavoro del broker che può tener traccia di ogni monete coniata utilizzando un semplice bit.
In questa descrizione di base esiste però un problema: la memorizzazione dei dati è di gran lunga più onerosa della computazione. Il numero di biglie che possono essere lanciate supera abbondantemente il numero di quelle che possono essere memorizzate su un hard-disk ed il numero di quelle di cui il broker ha realmente bisogno. Per trovare un giusto equilibrio possiamo pensare di rendere molte biglie inutili allo scopo di coniare monete.
Ciò può essere fatto supponendo che una biglia sia "buona" se i bit di maggior peso del valore hash y hanno un valore *z* specificato dal broker. Per essere più precisi siano *t* ed *u* per cui $n=t+u$. Allora se i *t* bit più significativi di h hanno valore *z* allora il valore y è buono e gli *u* bit di y determinano il valore del recipiente nel quale la biglia x è stata lanciata. Utilizzando questo processo il broker lancia $k*2^n$ e ne memorizza circa $k*2^u$ generando circa $1/2*2^u$ monete valide.
Nel caso in cui il numero di bit in output della funzione hash scelta (ex. : DES, MD5) superi il valore si possono scegliere soltanto *n* bit, ad esempio gli *n* bit di peso minore.

5.5 Uno scenario dettagliato

In questa sezione vedremo un esempio specifico di come un broker fissa la scelta dei parametri per coniare monete valide per un dato mese. I calcoli sono approssimati alla più vicina potenza di due ma restano comunque molto significativi. Il broker investirà in un hardware solido, per avere vantaggi computazionali su eventuali contraffattori, che farà lavorare continuamente in un mese per produrre monete per il mese successivo. E' preferibile che l'hardware contenga chip specifici per calcolare i valori della h.
Supponiamo che il broker abbia, mensilmente, un profitto di un milione di dollari al mese (circa 2^{27} cent).
Carica la tariffa di mediazione pari al 10%, quindi, per ogni moneta inviata, pagherà al venditore 0.9 cent quando la stessa sarà riscattata. In questo modo deve distribuire un bilione di monete al mese (circa 2^{30}) per raccogliere il suo milione di dollari. Se ogni cliente medio acquista 2.500 monete al mese egli necessita di una clientela di 500.000 unità. Il valore k scelto per la collisione è 4 mentre il valore di *u* è 31. In questo modo deve creare un array di 2^{31} recipienti ognuno dei quali può contenere dai 4 valori di x in su.
Con queste assunzioni numeriche il broker deve lanciare una media di 4 biglie per ognuno dei 2^{31} recipienti generando quindi $4*2^{31}=2^{33}$ valori di x che producono i valori y buoni. Operando in questo modo la probabilità che un recipiente contenga almeno 4 valori di x e' circa 1/2, quindi ogni recipiente produce una moneta con probabilità pari ad 1/2. Ciò vuol dire che il numero di monete generate e' $1/2*2^{31}= 2^{30}$ cioè il numero che si desiderava. Se il broker ha a disposizione una macchina efficiente per il calcolo dei valori hash può scegliere valori di *t* molto grandi in modo tale che i valori buoni risultino pochissimi.
La funzione hash viene scelta come gli *n* bit di peso minore della codifica di qualche fissato valore v_0 con chiave x sotto DES:

$$h(x)=[DES_{x(v0)}]_{1..n}$$

Il broker acquista un numero di chip (FPGA), ognuno dei quali è capace di applicare una funzione hash approssimativamente su 2^{25} (circa 30 milioni) valori di x al secondo. Comprandone 256 può calcolare 2^{33} valori hash al secondo. Poiché in un mese ci sono 2^{21} secondi i chip possono calcolare circa 2^{54} valori al mese. Sulla base di queste stime il broker può scegliere i valori $n=52$ e $t=21$ e conia monete per un mese.
Dei $k*2^n$ valori calcolati solo 1 su 2^{21} sarà buono e quindi complessivamente ci saranno 2^{33} valori di

x che sono buoni, cioè quelli necessari per produrre 2^{30} monete.
La memorizzazione di una coppia (x,h(x)) richiede meno di 16 byte . In totale lo spazio richiesto è di circa 2^{37} cioè 128 Gbyte.

5.6 Distribuzione delle monete

Il broker inizia la distribuzione delle monete ai suoi clienti verso la fine di ogni mese. Queste monete verranno poi utilizzate nel mese successivo e soltanto all'inizio del mese rende pubblico il criterio per ritenerle valide. I clienti che comprano monete caricano questo acquisto sulla propria carta di credito. Il broker da parte sua tiene traccia delle monete distribuite ai singoli utenti. Quelle monete non utilizzate gli verranno restituite a favore di altre valide per il mese successivo.

5.7 Pagamenti

Ogni volta che un cliente deve pagare un acquisto ad un venditore gli spedisce la serie $x=x_1,x_2,..,x_k$ che forma la moneta. Quest'ultimo controlla che sia una moneta valida calcolando l'hash su ogni valore della serie e verificando che è uguale per tutti (collisione a k-vie). Questo calcolo è particolarmente rapido.

5.8 Riscatto delle monete

Il venditore, ogni giorno, restituisce al broker le monete che ha accumulato. Il broker controlla le varie monete tentando di individuare eventuali monete che sono state già riscattate. Per le monete valide paga al venditore la somma stabilita. Per quelle che gli sono state inviate più volte sceglie di pagare uno solo dei venditori, finendo inevitabilmente per penalizzare gli altri.

5.9 Proprietà di sicurezza

Gli attacchi possibili ad uno schema Micromint sono a larga e piccola scala. Con questi termini si intendono rispettivamente attacchi che portano a consistenti guadagni o a piccoli guadagni per eventuali contraffattori. Visto che gli attacchi a piccola scala non portano a veri profitti i meccanismi di sicurezza, per lo schema, sono costruiti per opporsi agli attacchi a larga scala. Comunque qui di seguito verranno descritti tre tipi di attacchi che si possono effettuare.

Contraffazione: Una contraffazione a piccola scala è inapplicabile se si pensa che una normale workstation può effettuare solo 2^{14} operazioni hash al secondo mentre per coniare una moneta falsa ci vogliono 2^{45} operazioni hash. Ciò vuol dire che il contraffattore impiega 2^{31} secondi, cioè 80 anni, solo per coniare una moneta. E' necessario quindi contrastare la contraffazione a larga scala. Ciò può essere fatto nel seguente modo:

- tutte le monete false vengono invalidate automaticamente alla fine del mese;
- le monete false non possono essere generate sino a quando il broker non annuncia il nuovo criterio di validità delle monete per il prossimo mese;
- utilizzare dei predicati nascosti che forniscono un intervallo di tempo più adeguato per respingere monete false senza andare ad intaccare la validità delle monete legali che sono in circolazione;

- il broker può intercettare un falsario verificando tra le monete ricevute quelle che lui non ha mai generato. Ciò è possibile nel nostro esempio di prima perché solo la metà dei recipienti utilizzati producevano monete;
- in qualsiasi momento del mese il broker può dichiarare la validità delle monete, ritirando tutte le monete ed immettendone delle nuove comunicando il nuovo criterio di validità;
- il broker può simultaneamente generare monete per più mesi utilizzando una grande computazione; ciò rende il compito del falsario più difficile a meno che questi non abbia a disposizione gli stessi mezzi del broker.

Furto di monete: Il furto di monete è relativo alla fase di distribuzione delle monete agli utenti e alla fase di riscatto di monete dal venditore. Ciò è risolvibile eseguendo in queste fasi delle cifrature e quindi utilizzando una chiave di crittografia nelle relazioni broker-utente e broker-venditore. Per quanto riguarda l'altra relazione, utente-venditore, la protezione durante l'invio di monete può essere fatta utilizzando una chiave pubblica di crittografia oppure fornendo ad ogni utente delle monete personalizzate in modo tale che questi sia l'unico a poterle utilizzare. Un'altra situazione da considerare è che due venditori possono riscattare le stesse monete. Ciò è evitabile creando monete specifiche anche per i venditori.

Riutilizzo di monete: Lo schema Micromint non garantisce l'anonomato e quindi il broker può individuare, per moneta riutilizzata, i venditori che gli hanno fornito le diverse copie. Conoscendo a chi egli ha fornito la monete, può individuare attraverso l'aiuto di venditori onesti, il cliente che ha speso diverse copie della stessa moneta. Provare però legalmente che un cliente sia reo di utilizzo di monete duplicate risulterà molto difficile poiché non si utilizzano schemi di firme digitali.

Varianti

Come detto in precedenza esistono in commercio alcune varianti dello schema Micromint. Vedremo ora, brevemente, alcune di queste varianti.

Monete personalizzate per utenti: descriveremo due proposte fatte per questa variante.
Nella prima proposta il broker divide i suoi utenti in gruppi e distribuisce ad ogni utente monete la cui validità dipende dall'identità di un gruppo. Ad esempio il broker può dare all'utente U monete che soddisfino l'identità $h'(x_1,x_2,..,x_k)=h'(U)$ dove h' e' una funzione hash che produce un valore di output corto e che indica il gruppo di U. Per il venditore è facile verificare questa condizione e quindi rifiutare una moneta presentata da un membro del gruppo non corretto. Il problema con questo approccio è che se i gruppi contengono molti utenti un ladro può facilmente vendere monete rubate a qualche utente. Al contrario se i gruppi sono composti da poche persone il calcolo computazionale del broker diventa più grosso.
La seconda proposta la validità di una moneta per un utente U è accertata se sussiste la relazione :

$$y_{i+1} - y_i = d_i \bmod 2^U$$

per i=1,2,..,k-1 e dove $y_1=h(x_1),y_2=h(x_2),..,y_k=h(x_k)$ ed inoltre $(d_1,d_2,..,d_{k-1})=h'(U)$ utilizzando un'adeguata funzione hash ausiliaria h'. Se $d_1=d_2=..=d_{k-1}=0$ ritorniamo al caso generale.
Per coniare monete di questo tipo il broker riempie la maggior parte dei suoi contenitori in maniera casuale.
Dopo questa pre-computazione la moneta per U sarà coniata rapidamente seguendo queste fasi:

- si calcola $(d_1,d_2,..,d_{k-1})=h'(U)$;
- si sceglie a caso un indice y_1 di un recipiente (questo recipiente non dovrebbe essere stato già utilizzato per far si che y_1 sia usata come identità della moneta quando il broker usa un array di bit per determinare le monete già riscattate);

- si calcola $y_{i+1}=y_i+d_i \bmod 2^U$, per i=1,2,..,k-1;
- si estrae una biglia x_1 dal recipiente y_1 ed una copia di una biglia da ogni recipiente $y_2,..,y_k$ (se tutti i recipienti sono vuoti si riparte da una nuova scelta di y_1);
- si producono le k tuple $(x_1,..,x_k)$ come monete in output.

Il broker non necessita di conoscere a priori quante monete serviranno ad ogni utente ed inoltre ogni biglia "buona" può essere usata per creare diverse monete.

Monete personalizzate per i venditori: per diminuire il furto di monete, l'utente può darle al venditore, facendo in modo che ogni moneta possa essere riscattata solo da un piccolo gruppo di venditori. Questa tecnica è alla base dello schema Millicent.
La difficoltà principale sta nel fatto che a priori non si possono stabilire quali solo i venditori con cui un utente avrà dei rapporti.

Produzione anticipata per due mesi: questa è sicuramente la tecnica di difesa più valida contro la falsificazione a larga scala.
Il broker produce le monete con anticipo mentre un eventuale falsario può iniziare la sua opera soltanto quando il broker rende pubblici i criteri di validità delle monete per il mese corrente. Una tecnica con la quale calcolare le biglie per le monete di un singolo mese, richiede circa otto mesi di tempo, ma il broker non viene danneggiato, perché egli può contemporaneamente generare biglie per otto mesi consecutivi.
Da parte sua, il falsario, dovrà partire in ritardo e sarà comunque più lento anche se ha a disposizione le stesse macchine del broker. La validità delle monete si ha in base al valore *z* che il broker sceglie mensilmente quando i *t* bit di peso maggiore di h(x) assumono valore *z*.
Il broker sceglie in anticipo questi valori di *z* usati per i mesi successivi.
La differenza sostanziale tra il broker ed un falsario sta nel fatto che qualsiasi lancio della biglia x per il broker è potenzialmente "buona" mentre il falsario partendo da *z* deve decidere o meno se mantenere i valori delle biglie da lui lanciate.
Chiaramente questa è la fase per lui più delicata visto che non può memorizzare tutti i valori delle biglie che ha lanciato.

Predicati nascosti: questa tecnica funziona nel seguente modo. Sia $m>n$, richiediamo che ogni preimmagine di un bit soddisfi un certo numero di predicati nascosti. I predicati dovrebbero essere costruiti in modo tale che generare le preimmagini soddisfacenti i predicati deve risultare facile ed inoltre devono essere difficili da capire utilizzando dei campioni casuali.
Ad esempio un broker può avere un predicato nascosto per ogni giorno del mese. A questo punto può rilevare un predicato al giorno ai venditori imponendo quindi loro, di accettare solo quelle monete che verificano quel predicato.
Con questa tecnica l'opera di falsificazione risulta decisamente complicata.

6 MILLICENT

6.1 Introduzione

Millicent è un protocollo progettato per acquisti a basso costo basato su una validazione di moneta elettronica sul server del commerciante senza comunicazioni addizionali, onerose operazioni di crittografia, processing fuori linea. Il protocollo si basa su scrips e su brokers.
Quando un cliente effettua un acquisto con lo scrip, il costo dei beni è defalcato dal bilancio dello

scrip stesso. Il cliente stabilisce un account con il broker mentre quest'ultimo lo stabilisce con il commerciante. A questo punto l'account tra cliente e commerciante può essere visto come suddiviso in due parti:

- uno tra cliente e broker;
- l'altro tra broker e commerciante.

Un cliente ha un solo account con un broker; ciascun venditore ha un ridotto numero di account con i broker. Il cliente gestisce il bilancio dell'account senza rischi per il venditore, il quale è tutelato da modifiche del bilancio da una firma digitale. Allora il commerciante al più verifica la correttezza del valore dello scrip senza dover memorizzare i movimenti effettuati dal cliente.

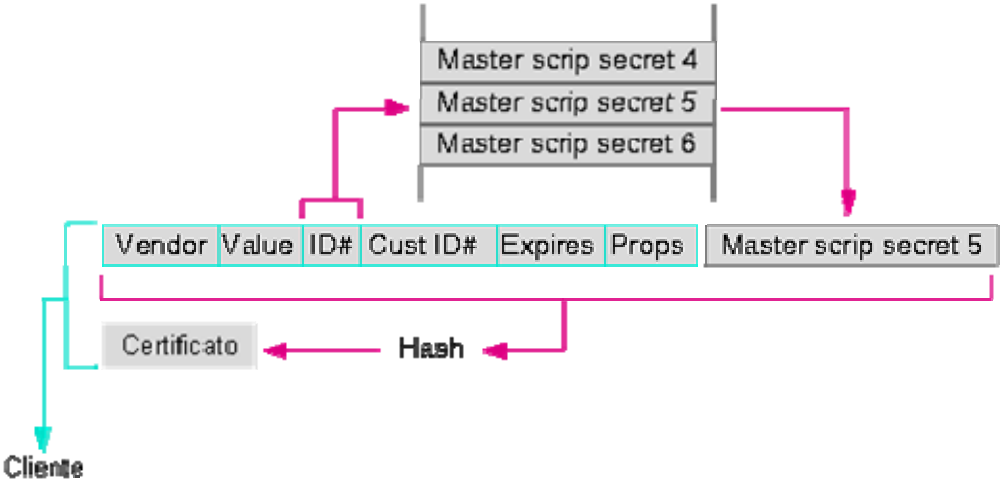
6.2 Sicurezza

Esistono tre fattori che rendono una frode non attrattiva:

- il cliente ed il venditore possono controllare indipendentemente il bilancio;
- i clienti non mantengono di volta in volta molti scrip, per cui un broker dovrebbe confermare molte transazioni fraudolente prima di ottenere un guadagno sensibile;
- nel caso in cui il commerciante non consegni la merce acquistata regolarmente (unico caso in cui il venditore può commettere una frode), il cliente può notificarlo al broker che provvederà ad eliminare dalla rete un commerciante nel caso in cui risulti più volte contestato.

6.3 Scrip

Gli scrip sono difficili da contraffare, possono essere spesi solo dai legittimi proprietari, è difficile spenderli più volte, possono essere agevolmente creati e validati.



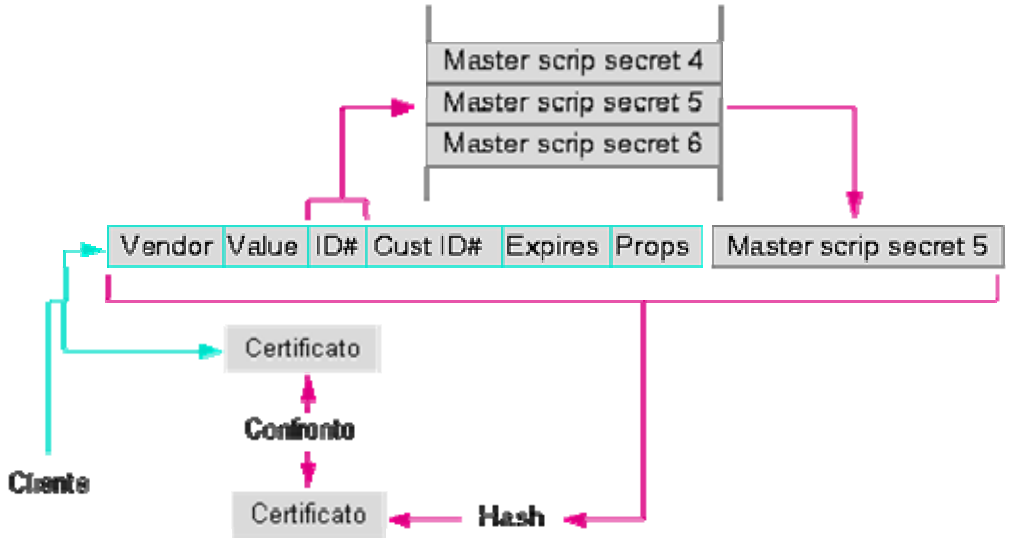
Lo scrip ha i seguenti campi:

- Vendor: identifica il commerciante;
- Values: il valore dello scrip;

- ID#: identificatore dello scrip (parte di esso contiene il master_scrip_secret);
- Cust_ID: produce il customer_secret (da cui è possibile selezionare anche il master_customer_secret);
- Expires: indica la scadenza dello scrip;
- Props: descrive informazioni aggiuntive del cliente al commerciante;
- Certificate: è la firma dello scrip.

Validazione e scadenza sono ottenute in due passi :

- il certificato è ricalcolato e controllato con il certificato inviato. Se lo scrip è stato falsificato, allora non c'è match (*VEDI FIGURA 2*);
- c'è un unico ID incluso nel corpo dello scrip ed il commerciante può verificare se la moneta è stata già spesa. Generazione e validazione dello scrip richiede una manipolazione del testo e una operazione hash.



Il commerciante può caricare una piccola tassa in modo tale da scoraggiare la richiesta di scrip superiore alle esigenze.

6.4 Protocolli

Descriveremo tre protocolli della famiglia Millicent.
Il più semplice ed efficiente, ma meno sicuro è lo "scrip in the clear".
Il secondo, "private and secure", offre una buona privacy, ma è più costoso.
Infine il "secure without encryption", è altrettanto sicuro, ma più efficiente.

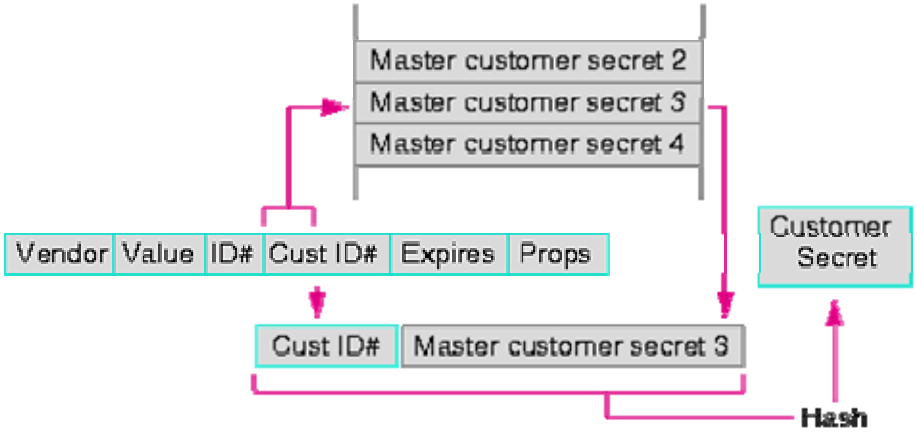
Scrip in the clear

In questo protocollo la comunicazione tra cliente e commerciante non è protetta, ossia avviene completamente in chiaro. Il cliente richiede una moneta, che gli viene offerta dal commerciante con

un identificativo ed in seguito può essere spesa. Una terza persona che si intromette sulla rete e cattura la moneta può spenderla.

Private and secure

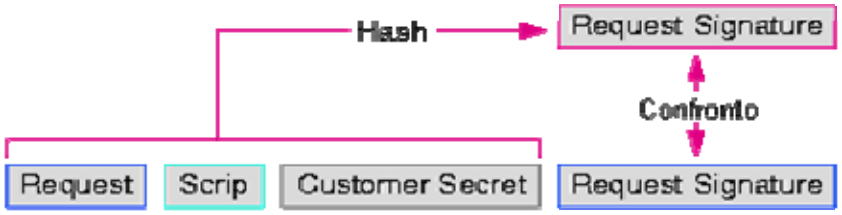
Per aggiungere sicurezza si stabilisce una chiave condivisa tra le due parti che si utilizza per cifratura simmetrica (DES,RC6), assicurando nello stesso tempo l'efficienza. Lo scrip può essere utilizzato per stabilire questa chiave condivisa. Quando un cliente acquista una parte dello scrip, si genera una chiave che si basa essenzialmente sull'ID del cliente stesso, che viene restituita tramite lo scrip.



Questa transazione deve essere necessariamente sicura.
La chiave non viene memorizzata direttamente, visto che può essere calcolata dal campo Cust_ID dello scrip.

Secure without encryption

La cifratura appesantisce il protocollo rendendolo meno economico. Per aumentare l'efficienza si usa il terzo protocollo.
Come nel caso precedente il cliente riceve un pezzo di scrip ed il customer secret.
Per acquistare il cliente invia la richiesta (scrip) ed una firma al venditore.
La firma è prodotta nello stesso modo in cui è prodotto il certificato dello scrip.
Lo scrip e la richiesta sono concatenate al customer secret.
Il cliente esegue una funzione crittografica hash efficiente su tale stringa ed invia il risultato come firma.
Quando il commerciante riceve la richiesta deriva il customer secret dallo scrip e rigenera la firma per la richiesta. In caso di alterazioni non c'è match tra le firme.

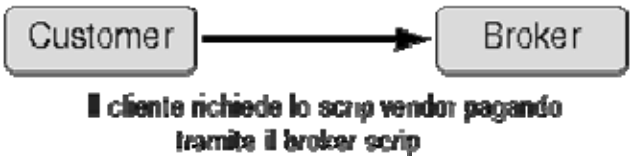


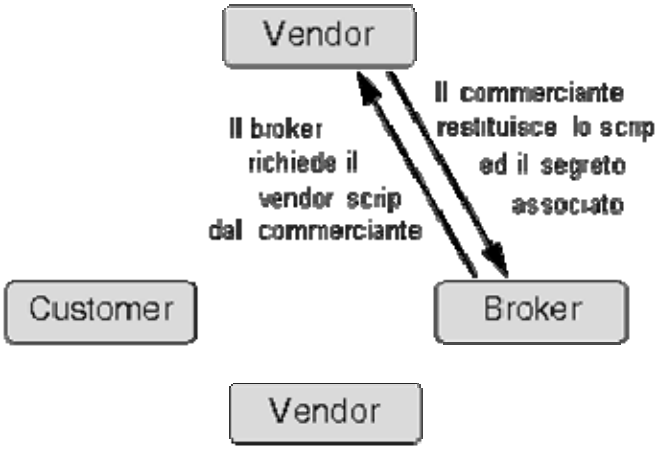
Brokers

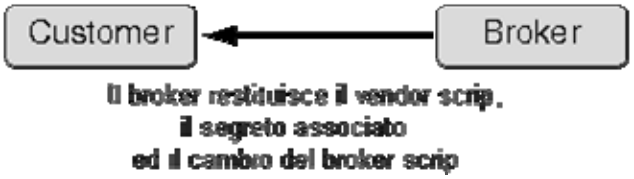
Il broker si pone tra il cliente ed il venditore e manipolano tutte le transazioni in moneta reale. Il cliente stabilisce un account con un broker utilizzando un sistema sicuro per acquistare un broker scrip. Il cliente usa lo scrip broker per acquistare un vendor scrip. Il commerciante ed il broker hanno relazioni a lungo termine. Il broker invia il vendor scrip al customer e paga il commerciante. Il cliente usa il suo broker scrip per pagare il vendor scrip, utilizzando il protocollo Millicent.

Interazioni tra Customer, Broker e Vendor

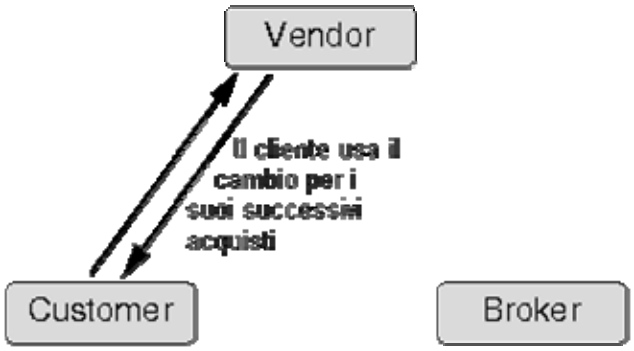
Le figure 5a-f mostrano un'intera sessione Millicent (incluso l'acquisto di broker scrip dal commerciante).




Vendor
Il commerciante restituisce lo scrip ed il segreto associato
Il broker richiede il vendor scrip dal commerciante
Customer
Broker
Vendor


Customer
Broker
Il broker restituisce il vendor scrip, il segreto associato ed il cambio del broker scrip


Vendor
Il cliente acquista servizi con il vendor scrip
Il commerciante restituisce risultato ed il cambio
Customer
Broker


Vendor
Il cliente usa il cambio per i suoi successivi acquisti
Customer
Broker

Il passo iniziale si ha solo una volta a sessione.
Il secondo succede ogni volta che il cliente non ha memorizzato lo scrip per un commerciante.
Il terzo passo si ha solo se il brokwer deve contattare il commerciante per acquistare uno scrip.
Il quarto passo mostra il broker che restituisce il vendor scrip al cliente.
Il quinto passo mostra il cliente che utilizza lo scrip per acquistare.
Infine l'ultimo passo mostra una tipica transazione Millicent. Il cliente ha gia lo scrip del commerciante e lo utilizza per effettuare acquisti.

7 Simple Anonymous Cash (Chaum-Fiat-Naor) - *I contanti anonimi semplici*

Questo schema è basato sull'idea che di ricevere dalla vostra banca moneta digitale da prelevare come se fosse carta moneta, il proprio computer partecipa effettivamente alla creazione di moneta e sceglie effettivamente i numeri di serie tramite una procedura che simula una distribuzione casuale. Essi saranno tenuti segreti sul proprio computer in un *layer of blinding*, del quale solo il computer la chiave. Quindi, sottoporrete quei numeri crittati alla banca per la vidimazione; la banca li vidimerà riconoscendo loro un valore equivalente a quello da essa prelevato dal vostro conto; quando riceverete di ritorno i numeri crittati potrete rimuovere il layer of blinding. Perciò, quando spenderete quelle monete digitali, ognuno potrà vedere che sono state vidimate dalla banca e che la banca deve risponderne (almeno la prima volta in cui ognuna è spesa), ma nessuno, nemmeno la banca, potrà sapere da quale conto la moneta è stata prelevata. In questo modo verrà preservata la privacy.

Ma poiché il computer conosce i numeri di serie della moneta che ha creato, potrà sempre cooperare con la banca per ricostruire il tragitto della moneta versata sul conto di un destinatario. Questo significa che chi paga può sempre, retroattivamente e irrefutabilmente, rivelare il destinatario dei fondi. Con un simile contante elettronico, i vari usi criminali del denaro - estorsione, mercato nero, corruzione non saranno più probabili di quanto lo siano oggi con gli assegni.
Non solo chi paga è sempre in grado di risalire al destinatario, ma la moneta deve essere depositata in un conto bancario, al fine di verificarne la validità. Perciò, le entrate totali ricevute da una qualsiasi società potrebbero essere conosciute quasi in tempo reale, prevenendo così la possibilità di occultare il reddito per evadere il fisco e molte altre forme di riciclaggio.

7.1 Come funziona in pratica?

Per spiegare in pratica il funzionamento di questo protocollo vediamo prima quali sono le parti coinvolte in una transazione economica:

1. Alice, la titolare del conto, la persona generica che vuole effettuare l'acquisto;
2. Bob, la persona generica che vende il bene/servizio;
3. La banca, l'ente finanziario cui Alice si appoggia.

La banca pubblica una chiave RSA (e,n) e regola un parametro di sicurezza k. La banca inoltre pubblica due funzioni, f e g, dove f produce un valore casuale mentre g è una funzione di valore univoco.
Alice ha un numero di conto bancario u e la banca mantiene un contatore v associato.

7.1.1 Il protocollo di prelevamento

1. Alice crea k unità campioni, U_i. Ogni unità è data da alcuni numeri di serie casuali, a_i, c_i, d_i, per $1 \leq i \leq k$, scelti da un' insieme abbastanza grande per accertarsi che nessuna altra unità ottenga lo stesso valore.

$$U_i = f(x_i, y_i)\ 1 \leq i \leq k$$

Dove $x_i = g(a_i,c_i)$, $y_i = g(a_i \oplus (u \wedge (v + i)), d_i)$

Indicando con $\oplus$ l'or esclusivo e ^ la concatenazione

2. Alice maschera questa k unità con fattori casuali $\{r_1 \ldots r_k\}$ e li trasmette alla banca. I fattori impediscono alla banca il controllo del contenuto

$$B_i = r_i^{bi} U_i \mod n$$

3. La banca sceglie casualmente *k/2* unità per eseguire il test.

4. Alice fornisce alla banca r_i, a_i, c_i, d_i, per $1 \leq i \leq k/2$, (qui abbiamo supposto che la banca ha scelto la *i* tali che $1 \leq i \leq k/2$).

5. La banca *unblinds k/2* unità e controlla che Alice non abbia provato a truffare. Se non ci sono errori, la banca firma le unità a sinistra con la relativa chiave riservata e fornisce ad Alice

$$\prod_{k/2 \leq j \leq k} (B_j)^d \mod n$$

ed addebita il suo cliente.

6. Alice *unbliends* le unità firmate moltiplicando per l'inverso r_j^{-1}. Dopo questo punto Alice ha una moneta elettronica. Lei incrementa la copia del suo conto personale *v* con *K*.

$$C = \prod_{k/2 \leq j \leq k} f(x_j, y_j)^d \mod n$$

7.1.2 Il protocollo di pagamento.

1. Alice spende la moneta digitale ottenuta dalla banca, la C, per pagare Bob.

2. Bob sceglie una stringa binaria casuale $z_1, z_2, \ldots, z_{k/2}$ e la trasmette ad Alice.

3. Alice risponde come segue, per ogni $1 \leq i \leq k/2$
   a- se $z_i = 1$ allora Alice invia a Bob a_i, c_i e y_i
   b- se $z_i = 0$ allora Alice invia a Bob x_i, $a_i \oplus (u \Lambda (v+i))\ d_i$.

4. Bob verifica che la C sia valida prima di accettare il pagamento da parte di Alice.

7.1.3 Il protocollo di deposito.

1. Bob trasmette i pagamenti alla banca.

2. La banca verifica che sulla moneta sia apposta la propria firma digitale.

3. Inoltre la banca verifica che la moneta non sia già stata spesa.

4. La banca registra la moneta nella base di dati della moneta spesa ed inoltre immagazzina la

stringa binaria e la risposta corrispondente di Alice.(Questa sarà successivamente usata per interferire il *double-spender*).

5. La banca accredita l'importo sul conto personale di Bob.

7.1.4 Caratteristiche di questo protocollo

Sicurezza*:* In questo protocollo la probabilità per la frode ed il rischio di avere interferenze sono determinati da *k/2*, anche se *K* è piccolo ed uguale a 2, quindi le probabilità sono ancora 1 a 1.

Uno degli atti fraudolenti più semplici qui è double-spending. Per minimizzare questo atto, quando la banca riceve una moneta per il pagamento esso può registrare il numero della moneta in una base di dati, quindi se lo stesso numero compare due volte, la banca sa che è stato commesso un atto fraudolento.

Segretezza: Alice deve scrivere la sua identità su ogni unità che genera, ma solo metà di queste unità vengono rese note. Le monete non possono essere spese più volte. Devono essere restituite alla banca dopo ogni transazione a causa del protocollo rivelante di identità. Allora la banca conosce chi ha ritirato i soldi e chi li ha depositati, ma non conosce l'acquisto svolto.

Portabilità: Essendo queste monete costituite da numeri digitale e casuali sono facili da trasportare.

Trasferibilità: Questo schema è non trasferibile prorpio perché queste monete non possono essere spese parecchie volte. Devono essere restituite alla banca dopo ogni transazione.

Divisibilità: Questo sistema non dà alcuna alternativa ad una moneta divisibile.

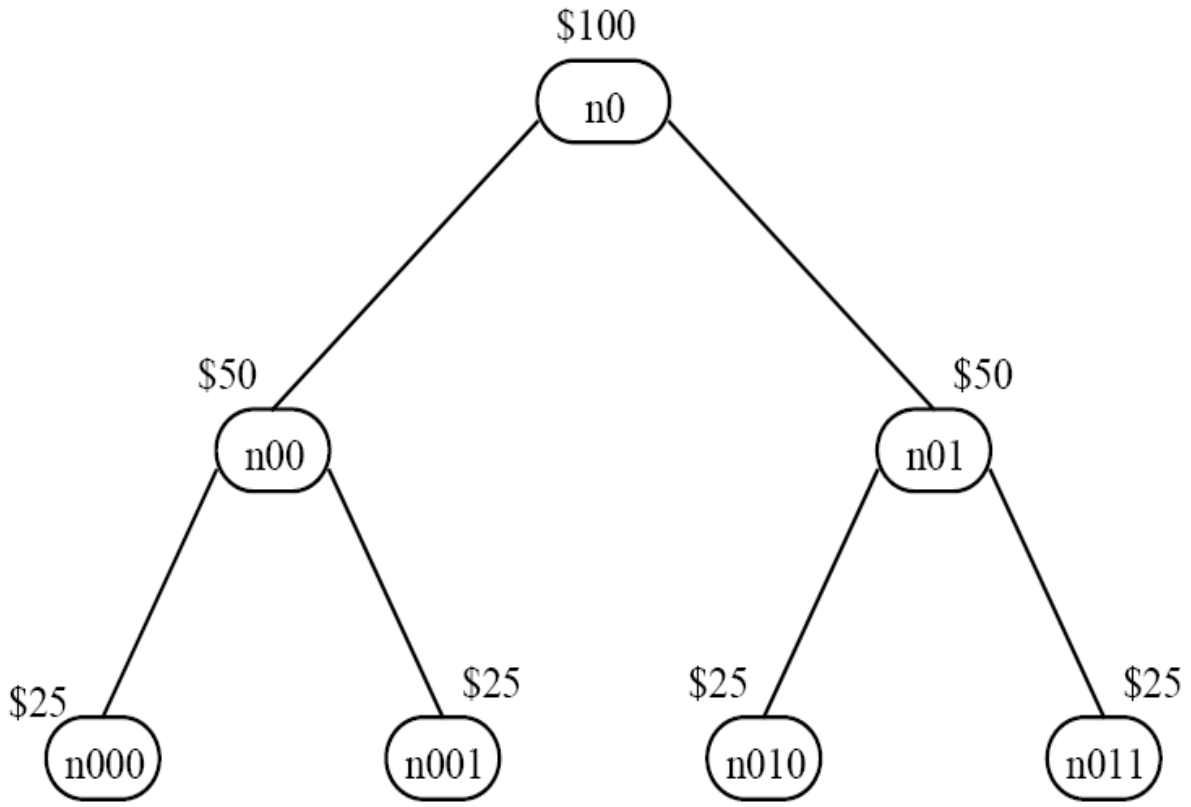
7.2 Gli attacchi

Questo sistema, come detto precedentemente, è basato su RSA quindi è vulnerabile a tutti i possibili attacchi del RSA
Inoltre, un attacco possibile contro questo sistema è un attacco di cooperazione fra Alice ed una eventuale terza persona. Se Alice dopo una transazione di pagamento con Bob, trasmette la sua moneta spesa ad una terza persona con la stringa binaria scelta da Bob e la risposta a questa stringa, questa terza persona avrà una storia di pagamento esatta. Bob e la banca non potranno determinare quale di loro sta truffando.

8 Divisible Electronic Cash

I sistemi di pagamento digitale descritti finora non hanno la possibilità di passare il denaro a più persone né di dividerlo in parti più piccole.
Un sistema per superare questo problema è stato proposto da **Okamoto** ed è basato sullo schema di bit commitment già visto, sulla radice quadrata modulo N e una rappresentazione dell'albero binario.

La sicurezza di questo sistema si basa sulla fattorizzazione (essendo basato sul RSA) e su funzioni hash.
La rappresentazione dell'albero binario gioca un ruolo importante, poiché permette di dividere la moneta elettronica in molti pezzi, in modo che poi questi possano essere usati separatamente.



Inoltre la struttura dell'albero è concepita in modo tale che sia possibile risalire all'identità del cliente se dovessero verificarsi tentativi di attacco.

Il metodo proposto da Okamoto si basa sull'uso di 2 regole :

1. **Nodi intermedi** - quando un nodo è usato non è possibile usare nessuno dei suoi figli né un suo genitore.
2. **Nodo uguale** – un nodo non può essere usato più di una volta.

Possiamo fare a questo punto due considerazioni:

1. Le regole garantiscono che un utente non possa usare una moneta di un valore maggiore di quello richiesto alla banca

2. La profondità dell'albero dipende dal valore iniziale della moneta e dalla precisione desiderata (ad esempio una moneta da \$1000 da cui si vogliano generare i cent richiederà un albero di $\log_2(1000*100)$ livelli (che corrisponde a 16,5, quindi abbiamo bisogno di 17 livelli)

8.1 Protocollo di Apertura conto

Questo protocollo serve per fornire al cliente una licenza elettronica, che successivamente può essere usata per usare le monete della banca.

1. La banca pubblica le sue chiavi pubbliche (sia **RSA** che delle firma cieca)
2. Il cliente controlla che siano valide e genera due numeri primi **p** e **q** tali che

$$p = 3 \bmod 8 \ , \ |p| < k \ , \ p > (1/4)\ \mathrm{n}_1^{\frac{1}{2}}$$

$$q = 7 \bmod 8 \ , \ |q| < k \ , \ q > (1/2)\ \mathrm{n}_1^{\frac{1}{2}}$$

3. Invia poi x e y alla banca(questi rappresenteranno l'identità del cliente)

$$x = g^p \bmod \mathrm{p}$$

$$y = g^q \bmod \mathrm{p}$$

ed inoltre manda alla banca anche S1 e S2 (dove N=p*q è un intero di Williams):

$$S_i = (N + a_i) r_i^{\,k} \bmod n_i \ , \ (i = 1/2)$$

4. La banca a questo punto manda indietro al cliente

$$\delta_i = s_i^{\,1/k} \bmod n_i \quad (i = 1/2)$$

5. Il cliente calcola

$$L_i = (N + a_i)^{1/k} \mod n_i = \frac{\delta_i}{s_i} \mod n_i \ \ (i = 1/2)$$

6. La licenza elettronica del cliente è formata quindi da (L1,L2,N)

$$N = pq \ , \ L_1 = (N + a_1)^{1/k} \bmod n_1 \ , \ L_2 = (N + a_2)^{1/k} \bmod n_2$$

8.2 Protocollo di prelievo

La transizione di prelievo è molto semplice perché consiste in una firma sul numero N concatenato ad un numero casuale b(la moneta è quindi unica).
Assumiamo che il cliente voglia ritirare un moneta da $w=2^l$(la banca avrà diverse chiavi pubbliche per i vari tagli)

I passi del protocollo sono quindi:

1. ll cliente sceglie un numero casuale b e lo concatena con N
2. Questo valore viene passato per una funzione hash unidirezionale per ottenere una moneta C

$$z = r^{e_w} \text{ H(N,b) } \bmod n_w$$

(dove r è un intero generico)

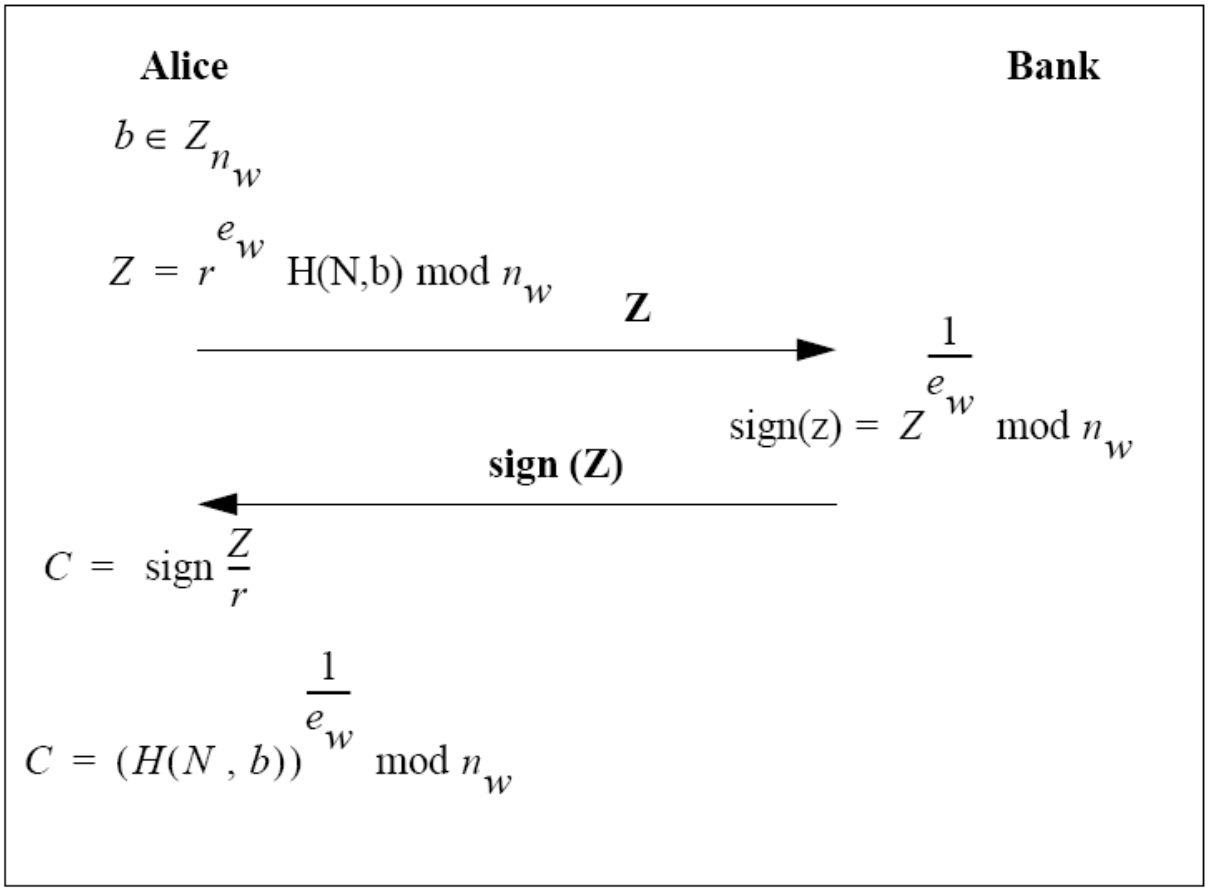
3. La banca firma la moneta che è stato inviato e toglie il valore corrispondente a w dal conto del cliente

$$\text{sign(Z)} = z^{1/e_w} \bmod n_w$$

4. Il cliente decifra la moneta e lo usa come moneta C

$$C = (H(N\ ,\ b))^{1/e_w} \bmod n_w$$

Nella figura seguente possiamo vedere tutto il processo schematizzato



8.3 Protocollo di pagamento

Questo protocollo si divide in due parti: **autenticazione della moneta** e **rivelazione del taglio.**

8.3.1 Autenticazione della moneta

Durante questa fase il venditore verifica che la moneta sia legittima.
Per compiere questa azione le operazioni da compiere sono le seguenti :

1. Il cliente fornisce la sua licenza,cioè (L1,L2,N), la moneta e il numero random per generarla (C,b) e infine il taglio del moneta w
2. Il venditore deve semplicemente verificare la licenza

$$L_i^{\,k} \equiv N + a_i \mod n_i \,,\ (i = 1\ ,\ 2)$$

3. Il venditore deve inoltre verificare la firma della banca sulla moneta

$$C^{e_w} \equiv \mathrm{H}(\mathrm{N,b}) \mod n_w$$

8.3.2 Rivelazione del taglio

Il cliente in questa fase rivela delle informazioni sui nodi dell'albero che sono stati già usati; infatti egli deve selezionare un nodo (il cui valore associato sia almeno pari a w) tra quelli che non violino le due regole di cui abbiamo parlato in precedenza.
Nel caso che lui fosse un malintenzionato e voglia non rispettare le regole il protocollo tramite le informazioni che il cliente deve inviare, la banca è in grado di verificare immeditamente se la transizione è corretta ed eventualmente bloccarla.

8.4 Protocollo di deposito

In questa fase il commerciante deposita la somma che ha guadagnato; per compiere questa azionale azioni da effettuare sono semplicemente :

- Il commerciante manda una copia della transizione d'acquisto alla banca.
- La banca deve ovviamente verificare che la stessa moneta non sia già stata usata

Vediamo come sia possibile compiere le due azioni precedenti :

1. Per verificare cerca nel suo database una moneta con gli stessi valori **C** e **b** e con la licenza elettronica (**L1**,**L2**,**N**)

2. Verifica poi per quella moneta i nodi usati (grazie all'albero)
3. Se una moneta è stata usata due volte può isolare N da cui risalire al cliente

8.5 Caratteristiche dell'e-cash

In questa sezione vedremo come si pone questo sistema rispetto alle caratteristiche che abbiamo individuato nella parte iniziale di questa tesina.

Sicurezza
La caratteristica comune ai vari protocolli è la K che viene presentata all'inizio e rappresenta il numero di bit richiesto per la chiave; poiché tutto il sistema si basa su RSA possiamo consideraci al sicuro fintanto che non sia possibile realizzare un attacco polinomiale

Privacy
Ogni cliente riceve una licenza elettronica da mandare al negozio durante la fase del pagamento, legando quindi ad ogni pagamento la licenza e una particolare moneta; questo rende facile seguire i vari pagamenti e quindi consiste in una specie di identificazione dell'utente

Portabilità
Ogni volta che il cliente vuole usare la moneta deve calcolare la rappresentazione di un albero binario; vediamo con un esempio se tale sistema possa essere utilizzabile nella pratica.
Se prendiamo k=257 e una moneta da $1000 (con precisione fino ai centesimi) possiamo calcolare che lo spazio richiesto dalla moneta più la licenza consiste di 264 byte; inoltre i dati totali per tutti i protocolli occupano 3.2 KB(entro i limiti delle attuali smart card)

Cedibilità
Supponiamo che il cliente A voglia pagare a B ad esempio $25 da una moneta di $100 e che B voglia poi passare i $25 ad un'altra persona C

1. Il protocollo di pagamento procede come specificato precedentemente
2. Al passo di rilevazione del taglio A invia una certificazione che denota il trasferimento della moneta a B, che poi potrà usare la certificazione con C
3. A e B continuano come da protocollo
4. B manda succesivamente tutta la trascrizione del pagamento tra lui e A a C
5. C controlla la validità della trascrizione inviatagli da B e poi seguono il protocollo

Divisibilità
Ovviamente questa è una delle caratteristiche più importanti che hanno guidato lo sviluppo; l'idea stessa di albero binario per rappresentare la moneta nasce dall'esigenza di dividire la stessa in formati diversi

9 Confronto

Nei capitoli precedenti è stato discusso che cosa i contanti digitali ed alcuni schemi differenti per i sistemi di contanti digitali. Il seguente capitolo dopo un confronto dei quattro schemi presentati esplorerà i problemi ed i punti di vista dell'utente di contanti digitali.

Confrontando i quattro sistemi di contanti digitali ha presentato nei risultati di questo rapporto la seguente conclusione:

Riguardo alla sicurezza: Lo schema di Chaum-Fiat-Naor, lo schema di Ferguson e lo schema di Okamoto sono sistemi basati su RSA quindi vulnerabili ai possibili attacchi del RSA. La loro sicurezza è basata aulla difficoltà di scomporre i grandi numeri in fattori, mezzi di elaborazione veloci potrebbero violare la sicurezza, anche se ci vuole del tempo quindi dobbiamo aumentare la lunghezza della chiave di questi sistemi.
La sicurezza dello schema di marche è basata sulla difficoltà a computare i logaritmi discreti piuttosto che sul RSA, quindi l'abilità di scomporre i grandi numeri in fattori non interessa la sicurezza di questo sistema. Allora il sistema di contanti della marca sta offrendo una sicurezza migliore.

Riguardo ai costi di comunicazione: Il sistema di contanti digitali di Chaum-Fiat-Naor consiste di otto movimenti; quattro movimenti per il protocollo di ritiro, tre movimenti per il protocollo di pagamento ed un movimento per il protocollo del deposito. Il sistema di Ferguson di otto movimenti; quattro movimenti per il protocollo di ritiro, tre movimenti per il protocollo di pagamento ed un movimento per il protocollo del deposito. Il sistema della marca di dieci movimenti; sei movimenti per il protocollo di ritiro, tre movimenti per il protocollo di pagamento ed un movimento per il protocollo del deposito. Il sistema di Okamoto di sei movimenti; due movimenti per il protocollo di ritiro, tre movimenti per il protocollo di pagamento ed un movimento per il protocollo del deposito.
Ogni movimento ha chiaramente un costo di tempo per la trasmissioe e di sicurezza in base al mezzo utilizzato. Il sistema di marche è quello che utilizza più movimenti mentre quello di Chaum-Fiat-Naor lo stesso numero e lo schema di Okamoto è quello più efficiente in termini di numero di movimenti.

Riguardo a complessità di esecuzione: Tra i quattro schemi presentati lo schema di Chaum-Fiat-Naor è il più facile in termini di esecuzione. Lo schema di Okamoto è più complesso poiché usa una combinazione dell'albero binario, il numero intero di William e la radice quadrata che provoca una procedura complessa da capire ed effettuare.

Riguardo al problema di double-spending: Tutti i sistemi presentati hanno il problema del *double-spending*.

Il risultato del confronto di questi quattro schemi mostra che lo schema delle marche risulta essere il migliore poiché offre un elevato livello di sicurezza mantenendo comunque l'anonimato del loro utente. L'unico svantaggio del sistema di marche è l'alto costo di comunicazione che può tuttavia essere abbassato chiedendo all'utente di ritirare parecchie monete nello stesso momento. Tuttavia, persino il sistema di contanti di marche non è né divisibile nè trasferibile anche se offre un' alternativa di divisibile al costo dell' anonimato.
Quindi possiamo dire che neanche questo sistema soddisfa tutte le condizioni di un sistema di contanti digitali ideale. Prima che questi sistemi possano essere realmente utilizzatati da parte dell'utente devono essere risolti vari problemi:

- se una moneta viene persa o rubata chi ci rimette?

- che genere di valuta deve essere attribuita alle monete? I dollari degli STATI UNITI, contrassegni tedeschi o dovremo avere un nuovo valore di contanti del cyber?
- Come dovremmo cambiare i soldi?
- Come dovrebbero essere calcolate le tasse? Su una transazione o sulla creazione di una moneta?

9.1 Descrizione del prodotto

Attualmente molte aziende sono impegnate nello sviluppo e nella implementazione di moneta virtuale valida solo su Internet come l' e-cash, cioè "contante elettronico"; il cybercoin, la "moneta cibernetica", il "digicash", o "denaro digitale" e altri ancora.
Anche se fino ad oggi nessun ha potuto sviluppare un sistema di contanti ideale.

Successivamente verranno presentati tre dei sistemi principale:

- Il DigiCash
- Il CyberCash
- Il First Virtual

9.2 Il DigiCash

Il DigiCash (denaro digitale) è stata fondata da David Chaum nel 1990 ed è una azienda situata ad Amsterdam. Principalmente si occupa di sviluppare i prodotti relativi al pagamento elettronico e ai sistemi di autorizzazione.

Uno dei prodotti di questa è l'Ecash, un prototipo di contanti digitali il cui progetto è iniziato nel 1995. Questo sistema è progettato per i pagamenti sicuri da effettuare tramite Internet o posta elettronica. Ecash è realizzato quasi completamente sul sistema digital cash di Chaum, l'unica differenza è che è un sistema in linea.
L'ecash non è un sistema basato su transizioni con carta di credito ma usa un proprio denaro virtuale, rappresentato da monete fornite da banche associate. Tali banche sono responsabili della certificazione e dell'autenticazione delle monete virtuali di ecash.
Per poter utilizzare ecash è necessario aprire un conto con una delle banche partecipante. Sarà possibile memorizzare sul proprio computer monete elettroniche; tali monete, al momento di un acquisto, verranno trasferite al venditore sfruttando tecniche di crittografia a chiave pubblica e firma cieca.

Operare con Ecash è piuttosto semplice: è sufficiente procurarsi il software *Ecash client* e aprire un conto con una delle banche partecipanti.

Il client Ecash, reperibile gratuitamente in rete, è in grado di operare (o ricevere) pagamenti con qualsiasi altro utente Internet che stia anch'egli utilizzando tale client.

Ad utenti e negozi non è richiesto alcun hardware particolare. Le banche avranno uno speciale hardware di codifica che assicuri velocità e affidabilità delle operazioni.

Ecash è un sistema basato su *moneta*, il che significa che si è creato denaro digitale usando una firma elettronica: tale firma rappresenta una quantità fissa di denaro chiamata *moneta*.

Sfruttando il client, il compratore (ad esempio Alice) ritira **ecash** (una particolare forma di denaro digitale) da una banca e lo memorizza sul proprio computer.

Il compratore è adesso in grado di spendere tale denaro presso qualsiasi negozio che accetti ecash (sia Bob il proprietario di uno di tali negozi), senza dover aprire un conto con tale negozio e senza dover trasmettere un numero di carta di credito..

Il **negozio** è rappresentato da un documento *html* contenente una serie di indirizzi (*URL*) indicanti le merci in vendita.

Ecash consente anche di effettuare pagamenti da persona a persona.

9.2.1 Ecash client software

Ecash lavora con tutte le maggiori piattaforme (MS Windows, Macintosh, UNIX). Esiste una versione con interfaccia grafica, descritta più avanti, e una versione solo testuale.
Per la versione corrente di Ecash è necessaria una connessione, ma una versione che sfrutterà l'**e-mail** è annunciata per un prossimo futuro.

9.3 Sicurezza e riservatezza

Per garantire sicurezza e riservatezza Ecash sfrutta tecniche di firma digitale a chiave pubblica. I prelievi di ecash dal conto di ogni utente sono inoltre protetti da una password nota esclusivamente all'utente stesso.

Quando è utilizzato per la prima volta, il software Ecash genera automaticamente una coppia di chiavi per codifica RSA. Ogni persona che utilizza Ecash possiede un'unica coppia di chiavi. Con queste è possibile garantire la **sicurezza** di ogni transazione e messaggio. Quando il mittente vuole inviare un messaggio pubblico lo autentica cifrandolo con la propria segreta e chiunque volesse verificarne l'autenticità lo decodifica con la chiave pubblica del mittente. Nel caso in cui il mittente volesse mandare un messaggio privato ad una sola persona può cifrare il messaggio con la chiave pubblica del destinatario il quale può decifrare il messaggio con la sua stessa chiave privata.

Ecash garantisce l'**anonimato** solo di chi paga. Durante un pagamento, colui che lo effettua può rendere nota la propria identità , ma solo se decide di farlo.

Chi riceve il pagamento, invece, non gode di anonimato: durante la fase di compensazione il beneficiario di una transazione è identificato dalla banca.

Per quanto riguarda la **riservatezza** delle transazioni, Ecash sfrutta la tecnica della **firma cieca**.
Quando il compratore ha necessità di effettuare un pagamento, deve avere ecash sul proprio computer. Il prelievo dalla banca è in realtà, per ragioni di riservatezza, qualcosa di più complesso che un semplice trasferimento di ecash dalla banca al PC del compratore.

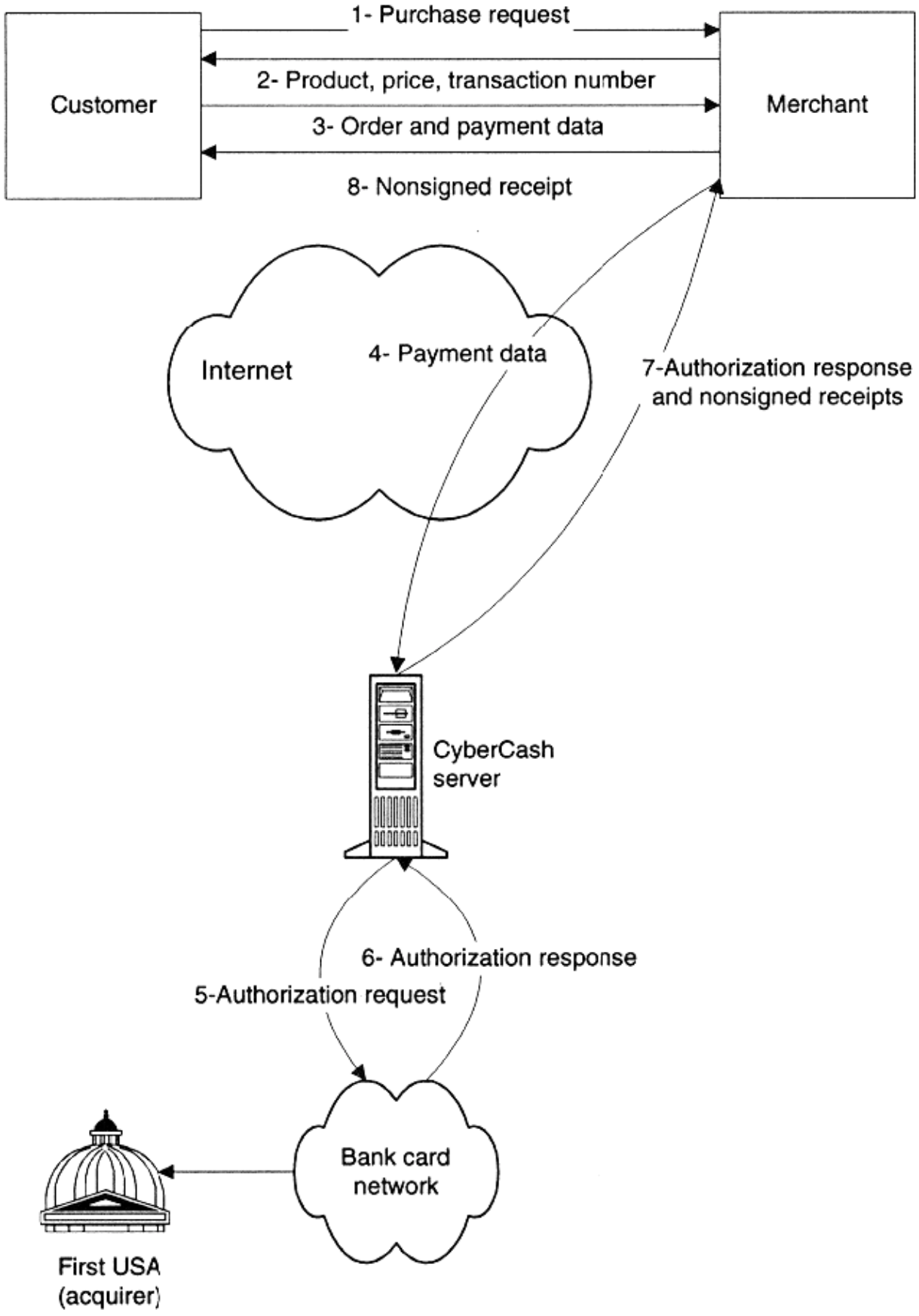
Il PC dell'utente (Alice) calcola quante monete sono necessarie per ottenere la somma richiesta. Successivamente è il computer di Alice stesso che crea monete assegnando ad ognuna di loro un numero di serie casuale. Quindi spedisce alla banca queste monete, una ad una inserite in una speciale *busta*: questa rappresenta il fattore "cecità ".

La banca codifica i numeri ciechi con la propria chiave segreta (firma digitale), grazie alla proprietà della firma cieca che consente di applicare tale firma *attraverso la busta*; allo stesso tempo, la banca addebita sul conto di Alice la stessa somma. Le monete autenticate sono restituite ad Alice, che potrà togliere loro il fattore di cecità introdotto in precedenza, senza alterare la firma della banca. I numeri di serie con le loro firma rappresentano adesso moneta digitale; il valore delle monete è garantito dalla banca.

Quando Alice spenderà tali monete, la banca le accetterà in quanto da lei firmate. Tuttavia, poichè non sarà in grado di riconoscere le monete (che erano nascoste nella busta al momento di essere firmate), la banca non potrà dire chi ha effettuato il pagamento.

9.4 Il CyberCash

L'azienda CyberCash è stata fondato nel 1994 da William Melton e Daniel Lynch; è nota come una fra i leader mondiali nella fornitura di supporti al CE. L'azienda offre un prodotto denominato Wallet ai relativi clienti. Il consumatore, cliente dell'aziensa, che volesse utilizzare tale sistema, basato su firme digitali e con una licenza di un algoritmo di crittografia di tipo RSA con una chiave a 1024 bit, deve scaricare sul proprio sistema il software necessario per il collegamento con la Cybercash. Il venditore dovrebbe in primo luogo aprire un conto presso una banca che supporti i pagamenti elettronici con questo prodotto e poi modificare il software del proprio server in modo da poter utilizzare Cybercash.
Nel momento in cui il consumatore effettua il suo acquisto presso il venditore inviandogli l'ordine, questi gli risponderà con il riepilogo degli articoli scelti indicando per ognuno il singolo prezzo, aggiungendo il numero della transazione ed altre informazioni. A questo punto il cliente potrà scegliere l'opzione "PAY" lanciando il *Cybercash wallet* in cui dovrà inserire il numero della sua carta di credito.
Queste informazioni verranno trasferite in forma cifrata al commerciante, il quale provvederà ad estrarvi l'ordine e ad inoltrare il resto del messaggio alla Cybercash, avendolo prima cifrato con la propria chiave privata. Si noti che il venditore non è in grado di decifrare la parte del messaggio contenente il numero della carta di credito. Ciò garantisce un'ulteriore protezione. La Cybercash provvederà a decifrare e a trasferire alla banca del merchant attraverso linee dedicate non collegate ad Internet. La banca del venditore inoltrerà la richiesta di autorizzazione alla banca del cliente.
L'esito dell'operazione di verifica della validità degli estremi comunicati sarà notificato alla Cybercash che ne informerà di conseguenza il venditore che poi provvederà a girarla all'acquirente.



Il sistema di cyberCash offre un'più alta sicurezza che le carte di credito. In questo sistema però né

la segretezza dell'acquirente né quella del consumatore sono protette. Questo sistema è in linea e Bob non può trasmettere l'articolo a Alice prima che ottenga la risposta alla banca.

9.5 First Virtual

L'azienda First Virtual è una delle prime aziende in grado di offrire un sistema di trasferimento digitale dei soldi generato per Internet. Il First Virtual è un sistema di pagamento elettronico basato sulla sicuerezza senza crittografia. la caratteristica di questo sistema è che tutti i rischi connessi all'operazione ricadono sul venditore.

Come funziona il protocollo di pagamento?

Vediamo come funziona il protocollo di pagamento nel First Virtual:

Registrazione del potenziale acquirente

1. Prima di tutto il cliente deve registrarsi presso la First Virtual.propri dati personali. In seguito gli viene consegnato dalla First Virtual un numero telefonico che gli permetterà di comunicare anche il proprio numero di conto corrente. In questo caso l'acquirente evita di inviare il numero di conto direttamente su Internet.
2. Conclusa la fase di registrazione, viene attribuito al richiedente un *codice identificativo personale* (*PIN*) che servirà per identificarsi al momento del pagamento della merce acquistata sulla Rete.

Registrazione del commerciante

2. Anche il commerciante deve registrarsi presso la First Virtual, fornendo gli estremi del proprio conto bancario.

Procedura di acquisto.

1. L'acquirente accede al sito di commercio elettronico del venditore tramite una password inviatagli in chiaro dalla First Virtual. Seleziona i prodotti desiderati, per i quali gli viene comunicato il relativo importo.
2. Il cliente invierà poi al venditore il PIN rilasciatogli dalla First Virtual, in modo che questi abbia modo di verificare la validità presso l'istituto stesso. Il messaggio in questione può essere spedito secondo diverse modalità: una delle più semplici è l'e-mail.

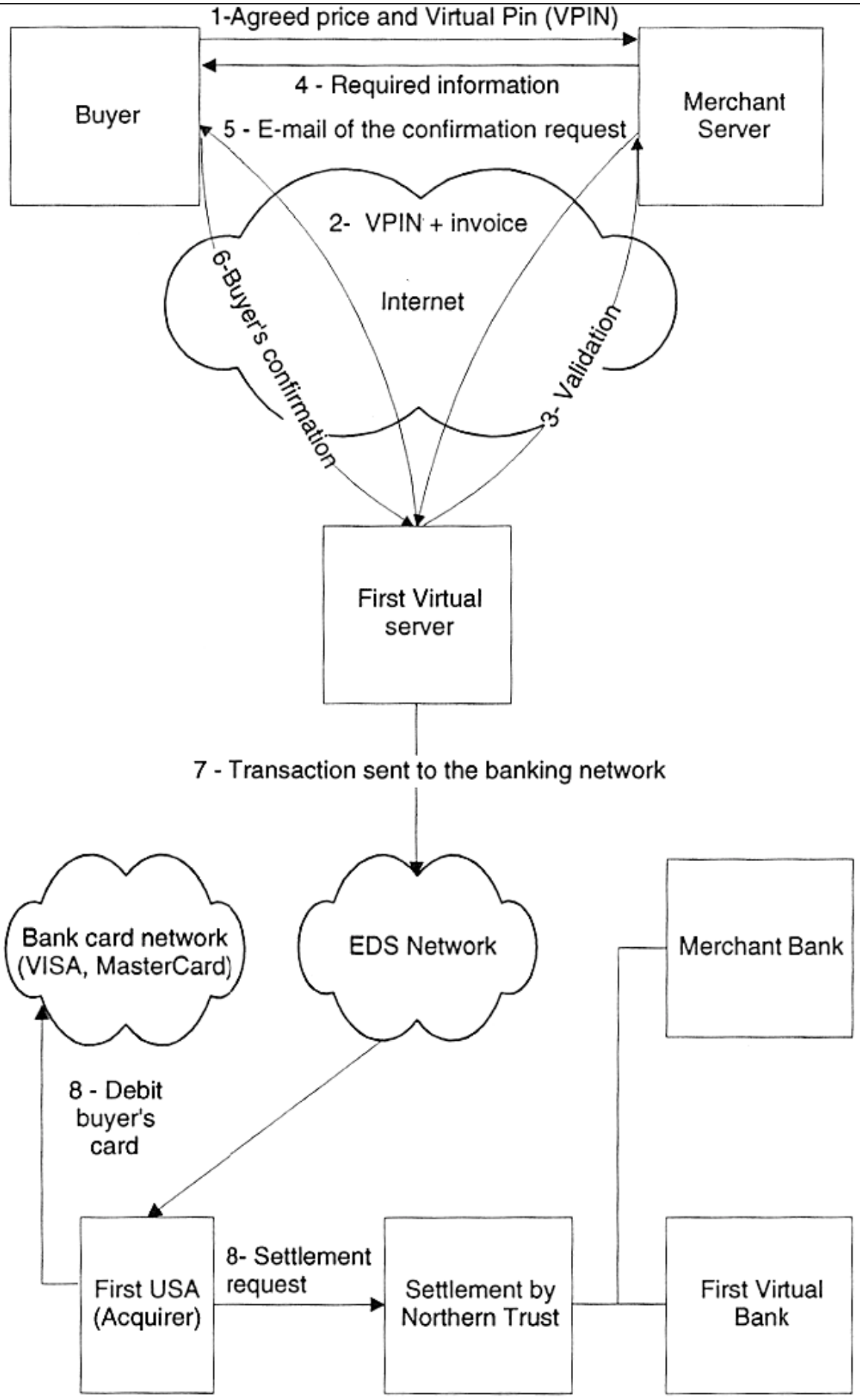
Seguendo questo approccio verrà indicato oltre all'e-mail del venditore anche quello dell'acquirente in modo che questo sia reperibile al momento della richiesta di pagamento.

3. First Virtual trasmette una richiesta al cliente per una conforma.
4. Il cliente può rispondergli con due risposte differenti:
   SÌ Il cliente autorizza.
   NO Il cliente sta rifiutando di pagare.

Se le risposte sono affermative condurranno alla detrazione dell'ammontare corrispondente all'acquisto dal conto corrente, altrimenti nessuna transazione finanziaria andrà in porto.
Qualora invece l'utente comunichi alla First Virtual di non aver mai autorizzato la transazione, questa provvederà ad eseguire una serie di controlli per intercettare l'autore della frode.
Quindi la First Virtual partendo dal presupposto che in una struttura come Internet è difficile evitare

in assoluto le frodi, ha cercato di realizzare un sistema che fosse in ogni caso in grado di ridurre il più possibile gli effetti di un furto. Nel caso che fosse rubato il PIN l'intruso non avrebbe modo di concludere la transazione in quanto non riceverebbe lui la richiesta di conferma dell'ordine, ma il reale proprietario del codice. La maggiore attrazione di First Virtual è la sua estrema semplicità ed il fatto che, diversamente da altri approcci utilizzati per il commercio elettronico, non si basa su protocolli crittografici, ma sull'invio di messaggi di conferma al cliente tramite tradizionali messaggi di posta elettronica.

Nella pagina successiva la figua mostra gli scambi di messaggi nel sistema First Virtual.


1-Agreed price and Virtual Pin (VPIN)
Buyer
4 - Required information
5 - E-mail of the confirmation request
Merchant Server
2- VPIN + invoice
Internet
6-Buyer's confirmation
3- Validation
First Virtual server
7 - Transaction sent to the banking network
Bank card network (VISA, MasterCard)
EDS Network
Merchant Bank
8 - Debit buyer's card
First USA (Acquirer)
8- Settlement request
Settlement by Northern Trust
First Virtual Bank

Bibliografia:

Carmine Ventre − Oblivious transfer per la generazione distribuita di chiavi RSA

Mandana Jahanian Farsi − Digital cash

Rosaria Rota, Rita Procesi − Elementi di algebra e matematica discreta
Xuhua Ding − Digital signature (presentazione)

J. Orlin Grabbe - Cryptography and number theory for digital cash