Micropayment systems

Chapter 7

Contents

Smallest Coin(e.g., a Rial)

- Obtaining a quotation of the current price of a share on the stock market
- Making a single query of a database service
- To consult an on-line encyclopedia
- Purchasing a single song from an album
- Ordering just the business pages from a selection of daily newspapers

Conventional Solution

- Subscription
  - the buyer pays in advance and can avail of the product or service for a fixed period.
- Drawbacks
  - People who may only wish to use a service very occasionally
  - It also restricts the ability of people to try out a service

Micropayment Design Goal

- Minimizing Per-transaction overhead

A system in which the costs of conveying the payment are greater than the payment itself is unlikely to succeed.

  - Communications traffic
  - High rate transactions process.
    - must not involve computationally expensive cryptographic techniques

Millicent

Chapter 7.1

Characteristics

- Developed at DEC (now Compaq)
- Payments as low as one-tenth of a cent ($0.001)
- Distributed approach
  - Validation at a vendor’s site without the need to contact a third party
- Scaleable
  - Without any additional communication, expensive public key encryption, or off-line processing

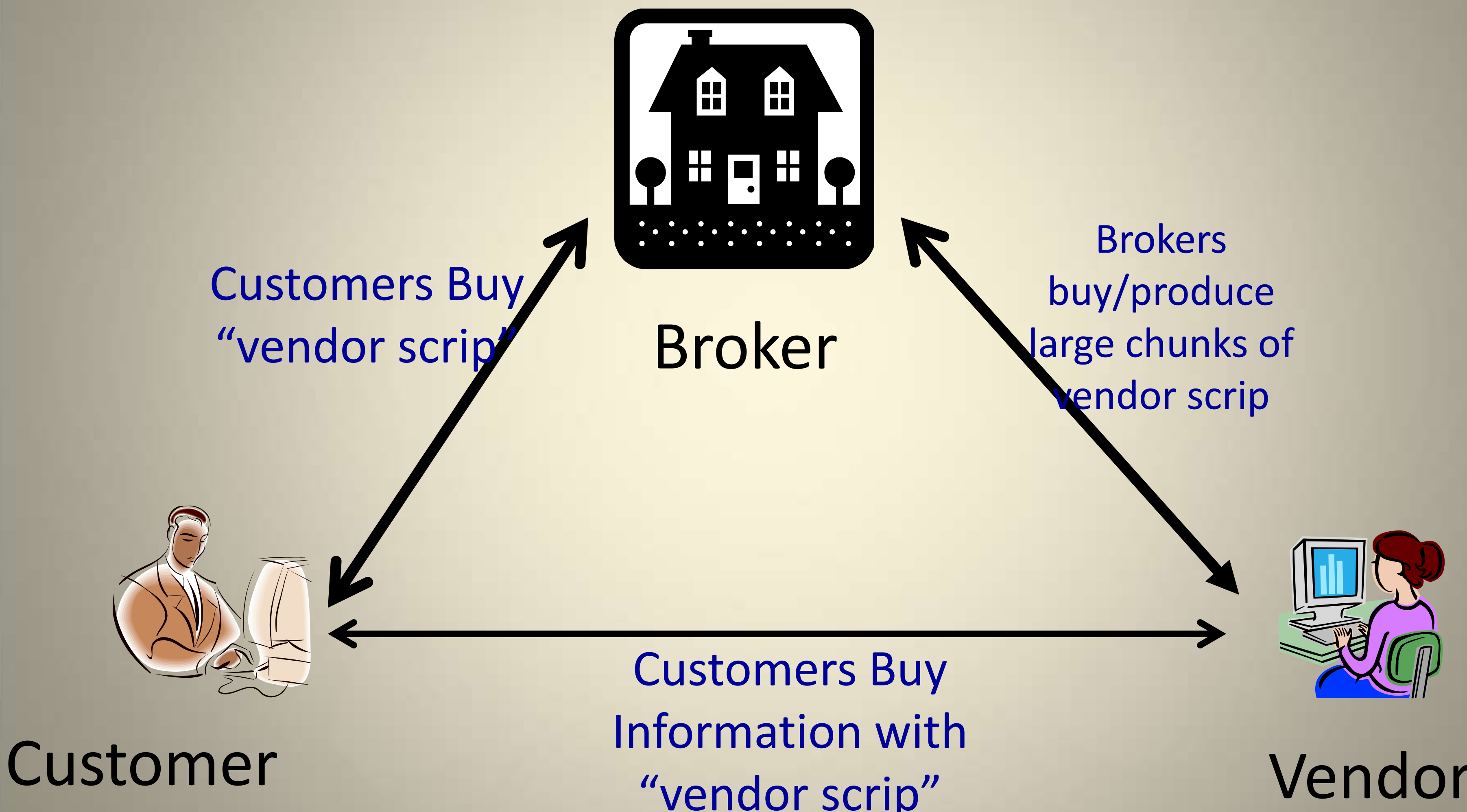
Scrip

- A form of electronic currency
- vendor-specific
- Security
  - The cost of committing a fraud more than the value of a purchase
  - Using symmetric encryption

The Millicent model

- Main entities in the Millicent system
  - Brokers
  - Vendors
  - Customers

The Millicent model



Millicent broker

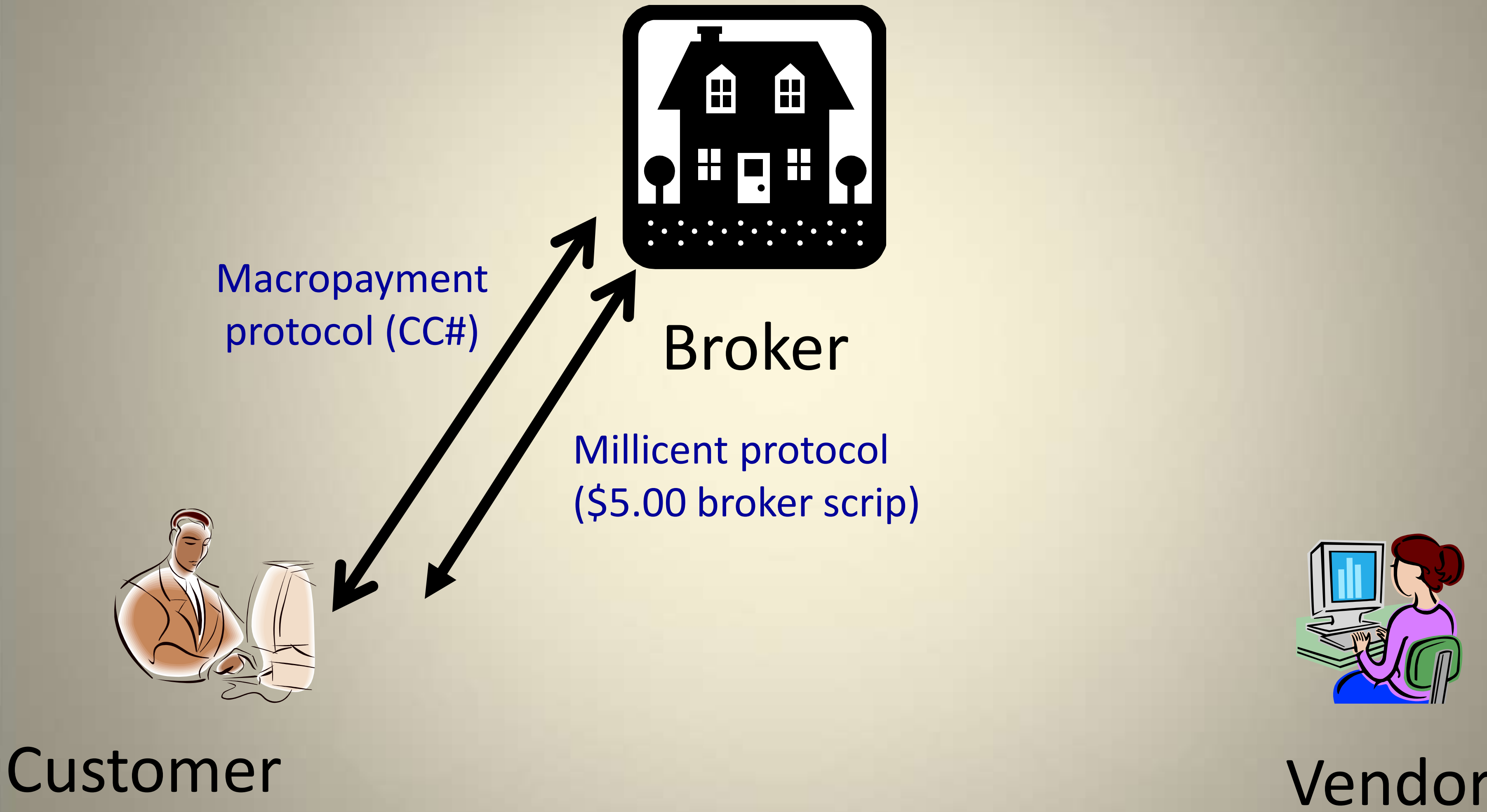
- Aggregating Micropayments
- Replacing a subscription service
- Selling Vendor Scrip
  - Scrip warehouse
  - Licensed scrip production

Better Solution

Purchasing with Millicent

1. customer buys some broker scrip using one of the macropayment systems,

Buying broker scrip



Purchasing from a vendor

- The customer buying from the same vendor again
- There is no need to contact the broker

Purchasing from a vendor

Broker

Customer

1-vendor script + request

2-vendor script
chamge+purchased
info/service

Vendor

Scrip

- Scrip is a piece of data used to represent micro-currency within the Millicent system

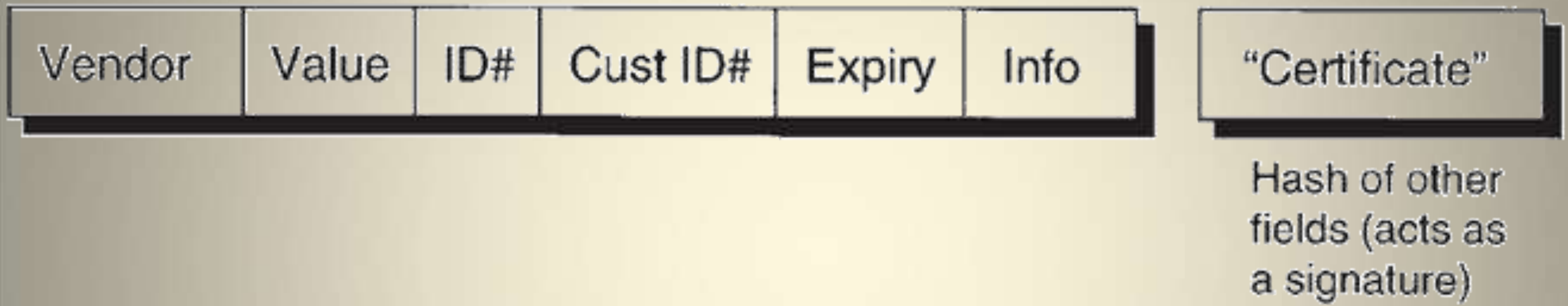
Scrip Properties

- Represents a prepaid value
  - like prepaid phone cards
- Can represent any denomination of currency
  - one-tenth of a cent
- Security for small amounts of money
- Vendor-specific
- Double spending will be detected locally by the vendor

Scrip Properties

- Spent only by its owner
- Cannot be tampered with or its value changed
- Computationally expensive to counterfeit
- No use of public-key
- Cannot provide full anonymity
  - Suggestion: buying broker scrip using an anonymous macro-payment system

Scrip structure (Data fields)



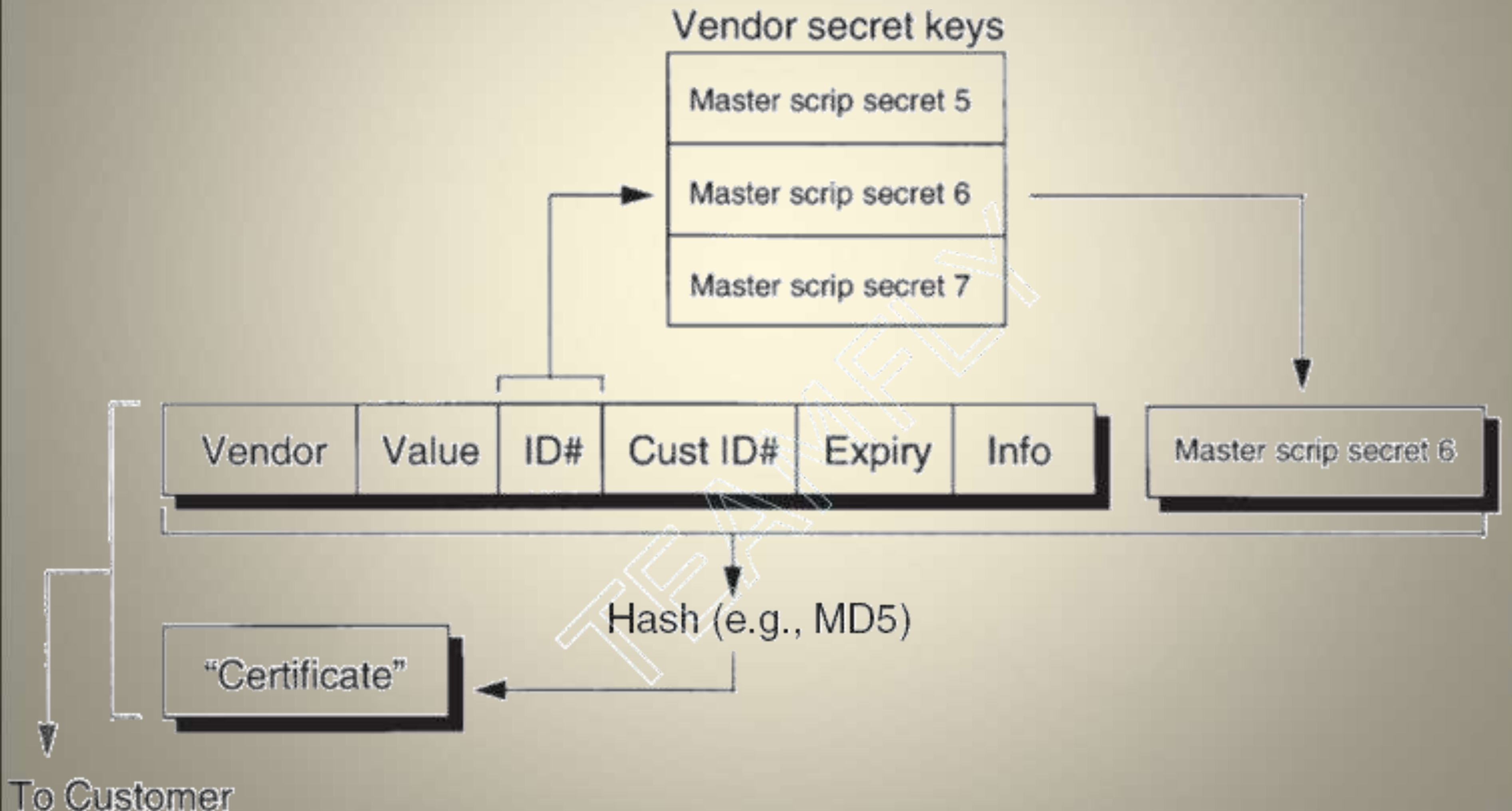
Scrip structure (Data fields)

- Value
  - Specifies how much the scrip is worth
- Cust_ID#
  - Used to calculate customer_secret
  - Unique to every customer
- Info
  - Optional details
- Certificate
  - as a digital signature

Generation of Certificate field

- Hashing the other fields of the scrip with a secret
  - prevents the scrip’s fields from being altered
- Master scrip secret
  - Only the vendor (or trusted broker) who mints the scrip will know this secret
  - Which master scrip secret is used with a particular piece of scrip depends on some part of the scrip’s ID#.

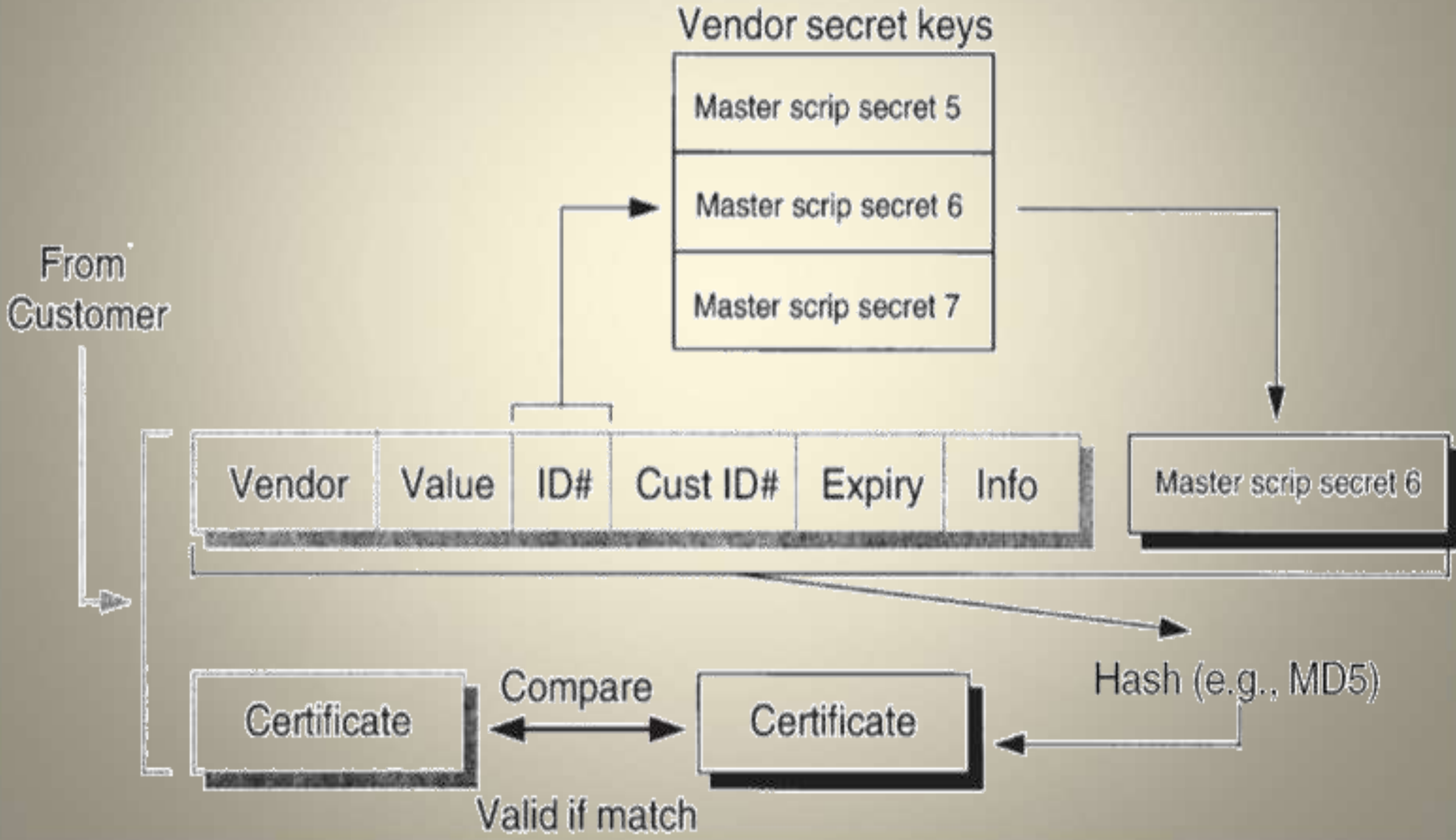
Scrip certificate generation



Scrip validation

- At the time of purchase, vendor validate customer’s scrip
  - Recalculates the certificate and compares it with the scrip certificate from the customer
    - Authentic scrip
    - Not already spent (double spending)

Validating scrip at the time of purchase



Preventing double spending

- The vendor checks that the ID# has not
- already been spent
- bit vectors
  - Covering ranges that have been fully spent or expired

Computation costs

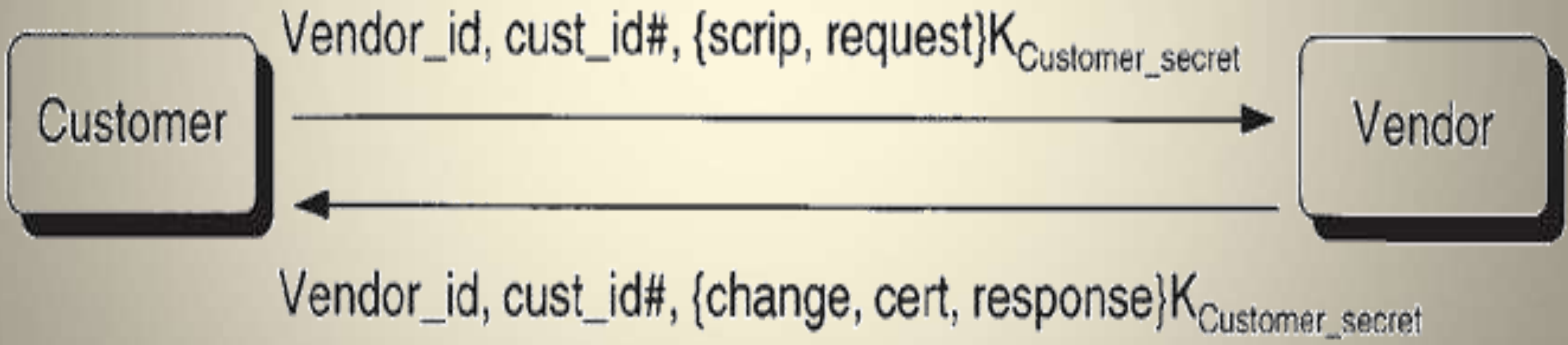
Action	Cost
Recalculate certificate	One hash function
Prevent double spending	One local ID# database lookup (in memory)
Making purchase across network	One network connection

Sending scrip over a network: the Millicent protocols

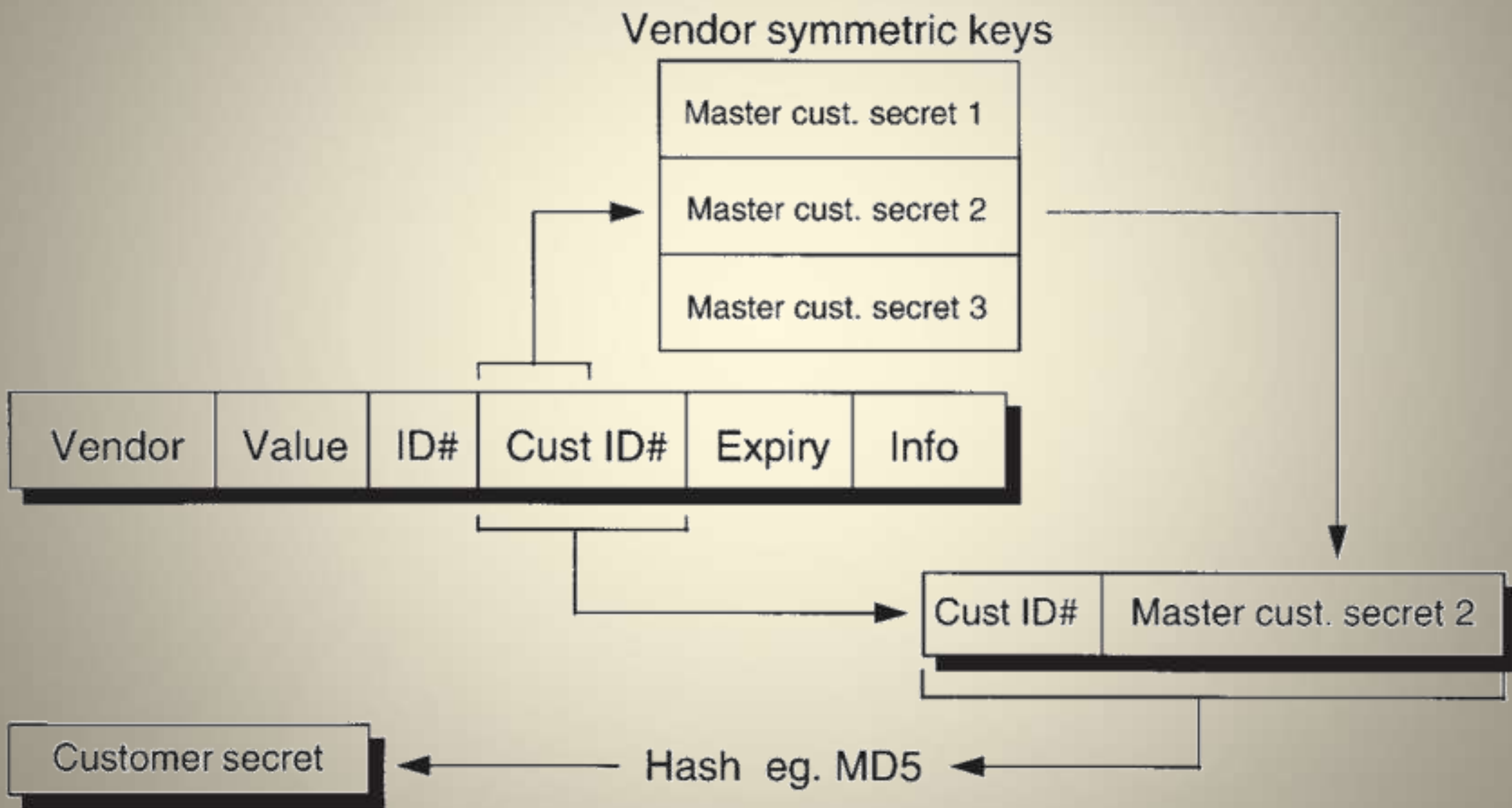
Millicent Protocol	Efficiency Ranking	Secure	Private
Scrip in the clear	1	No	No
Encrypted connection	3	Yes	Yes
Request signatures	2	Yes	No

Encrypted Network Connection

- Symmetric encryption using a shared symmetric key, called the customer_secret,



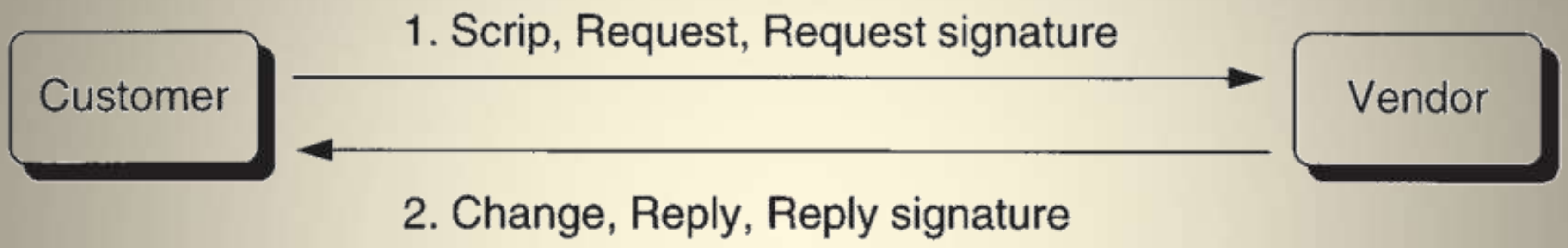
‘customer_secret’ Generation



Secrets Used in Producing, Validating, and Spending Scrip

Secret	Shared by	Purpose
Master_scrip_secret	Vendor, minting broker	Prevents tampering and counterfeiting of scrip. Used to authenticate scrip.
Customer_secret	Customer, vendor, minting broker	Proves ownership of the scrip. May be required to spend the scrip.
Master_customer_secret	Vendor, minting broker	Derives the customer_secret from customer information in the scrip.

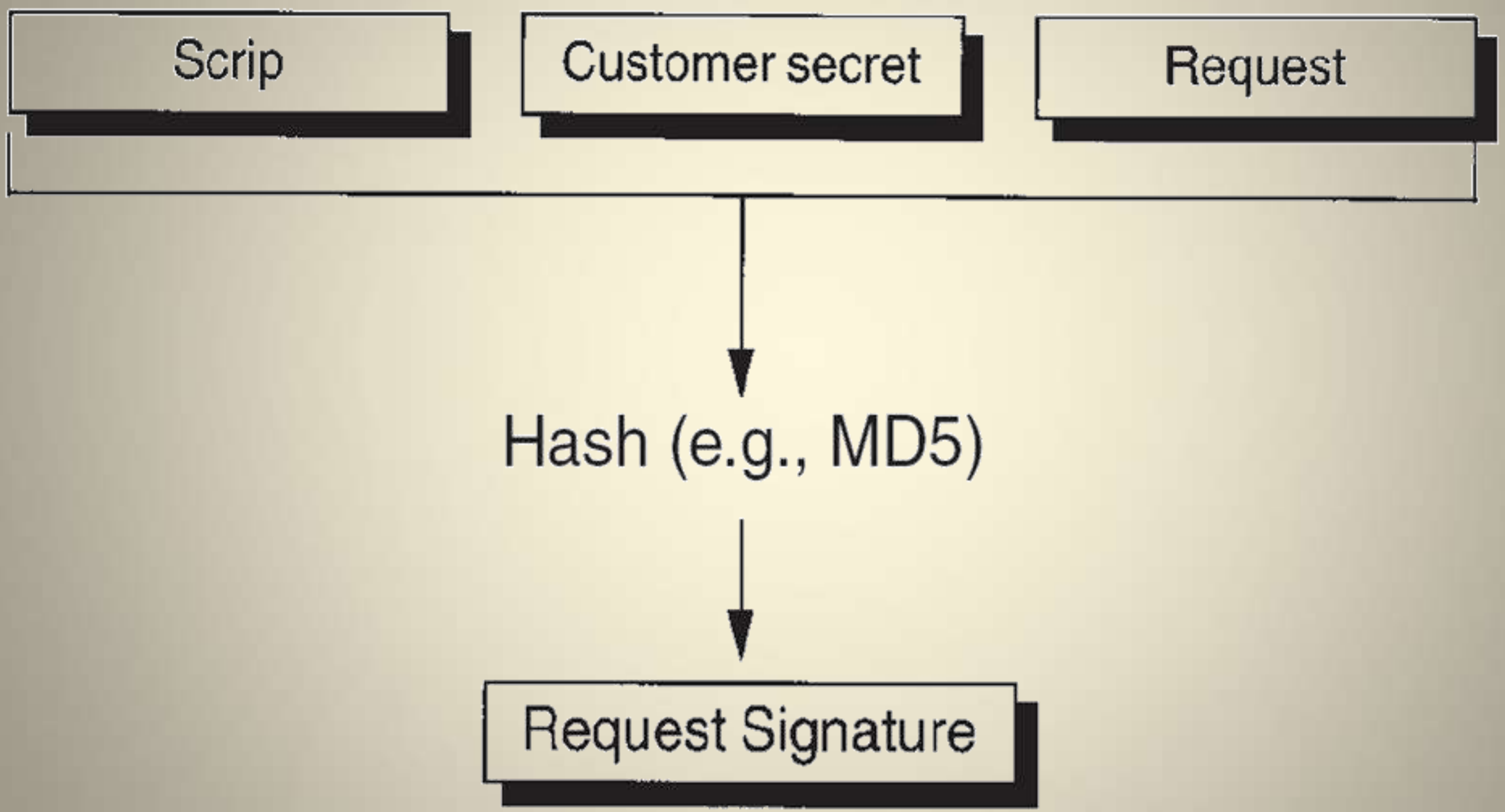
Purchase using a request signature



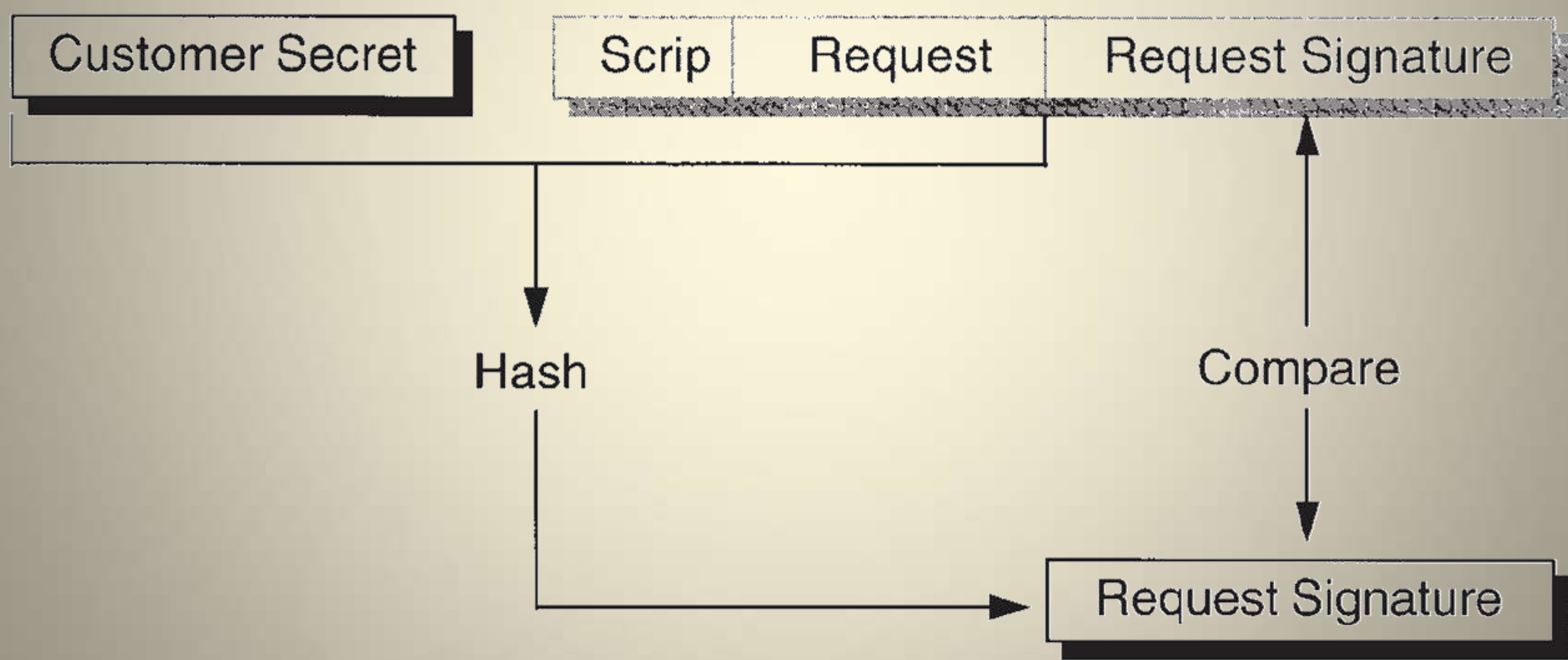
Request signatures

- The customer_secret is used to generate a request signature instead of being used for encryption
- Hashing the scrip, customer_secret, and request together

Generating a request signature



Vendor verifies the request signature



Performance

- Tests of a Millicent implementation on a Digital AlphaStation 4004/233
  - 14,000 pieces of scrip can be produced per second
  - 8,000 payments can be validated per second, with change scrip being produced
  - 1,000 Millicent requests per second can be received from the network and validated

bottleneck

Millicent with the Web

- The Millicent protocol as an extension to HTTP
- Software implementation
  - Wallet
  - Vendor server
  - Broker server

Extensions

- Authentication to distributed services
  - Kerberos-like authentication
  - To buy scrip for access to particular network services
- Accounting and metering applications inside private networks
- Per-connection charges for such services as e-mail, file transfer, Internet telephony

Extensions

- Discount coupons
  - Further fields could be added to scrip to provide discounts for certain content
- Preventing subscription sharing
  - Trying to gain access with an already used piece of scrip (such as shared scrip would be) will fail

scrip change giving
access the next time

Micropayment Systems

Chapter 7

Contents

SubScrip

Chapter 7.2

Introduction

- *pay-per-view payments* on the Internet
- Creating temporary prepaid accounts for users at a specific vendor

Level of Security

- No encryption
- The designers aimed to make the expense necessary for a successful attack much higher than the financial gain possible.

Transaction Cost

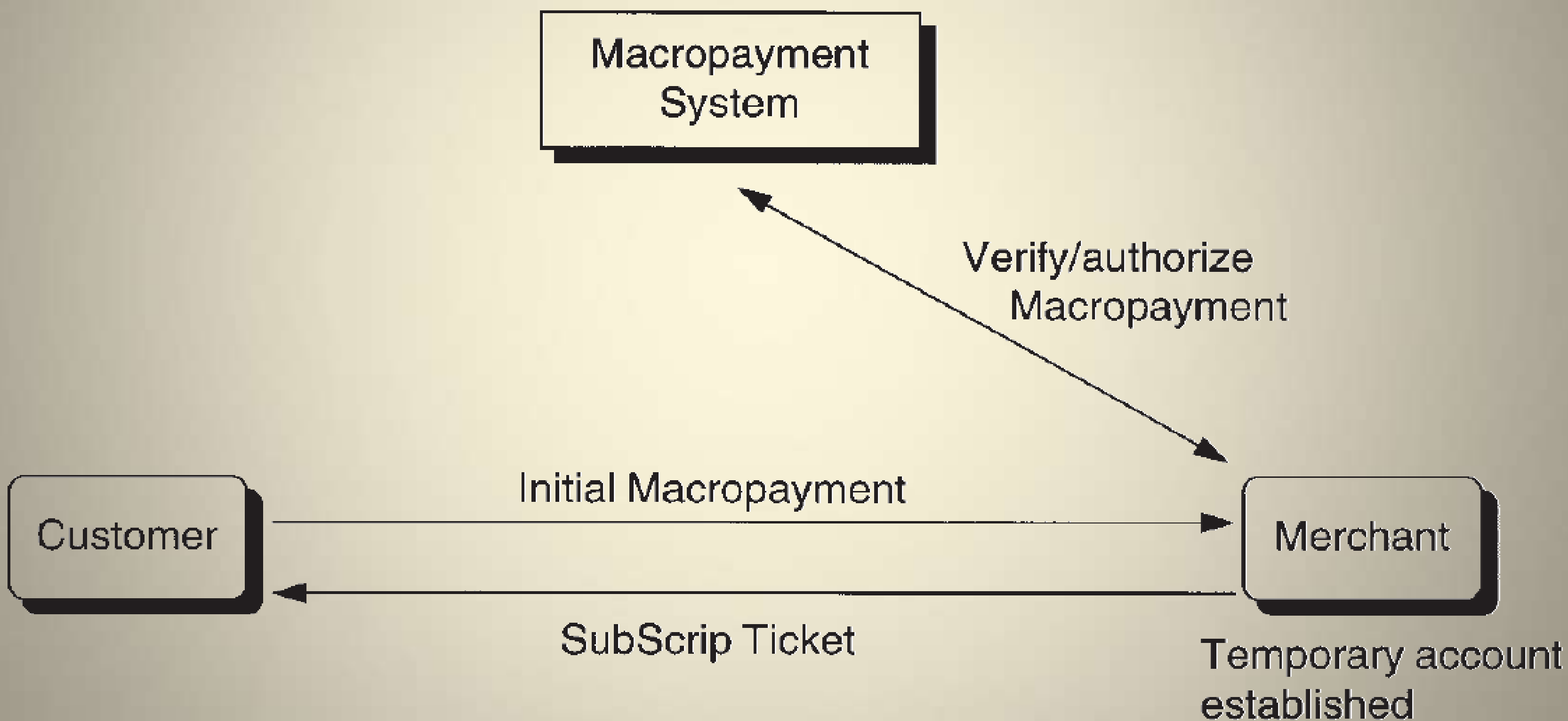
- Initial overhead for making a payment to a new vendor
- A micropayment can be verified locally by a vendor without the need for any on-line clearance with a third party

Temporary Account

- Millicent→ broker
- SubScrip→ a macropayment system

- Temporary Account
  - A payment large enough to cover the macropayment transaction costs to that vendor

Establishing a SubScrip Account with a Vendor



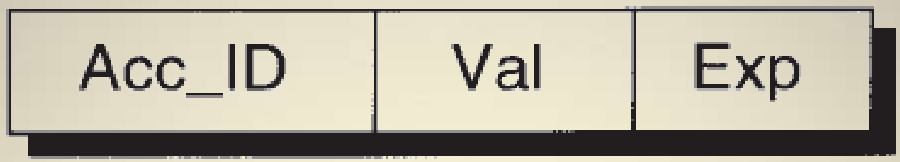
SubScrip Ticket

- Some type of account identifier for user
- The merchant returns a SubScrip ticket to the user to access the new account.

Anonymity

- Depend on the anonymity of the macropayment system used to initially pay a vendor.
- The merchant will only know the customer's network address

SubScrip Ticket Fields



- Acc_ID
  - A random number as account identifier
- Val
  - Remaining money in the account at the vendor
- Exp (expire date)
  - To limits the number of accounts that must be maintained by a vendor

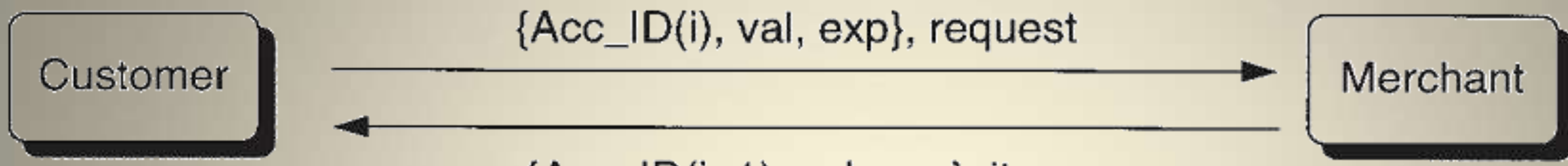
Notes

- The SubScrip ticket is not an electronic coin
  - Only for accessing prepaid value
- SubScrip value is transferable to another user.
  - By giving that user the valid ticket

Merchant ‘s Database Of Valid Accounts

- An account
  - IDs
  - amount
  - expiry date

SubScrip purchase



Verify ticket:

Acc_ID	Val	Exp
.	.	.
.	.	.
Acc_ID(i)	60	10-6
.	.	.
.	.	.

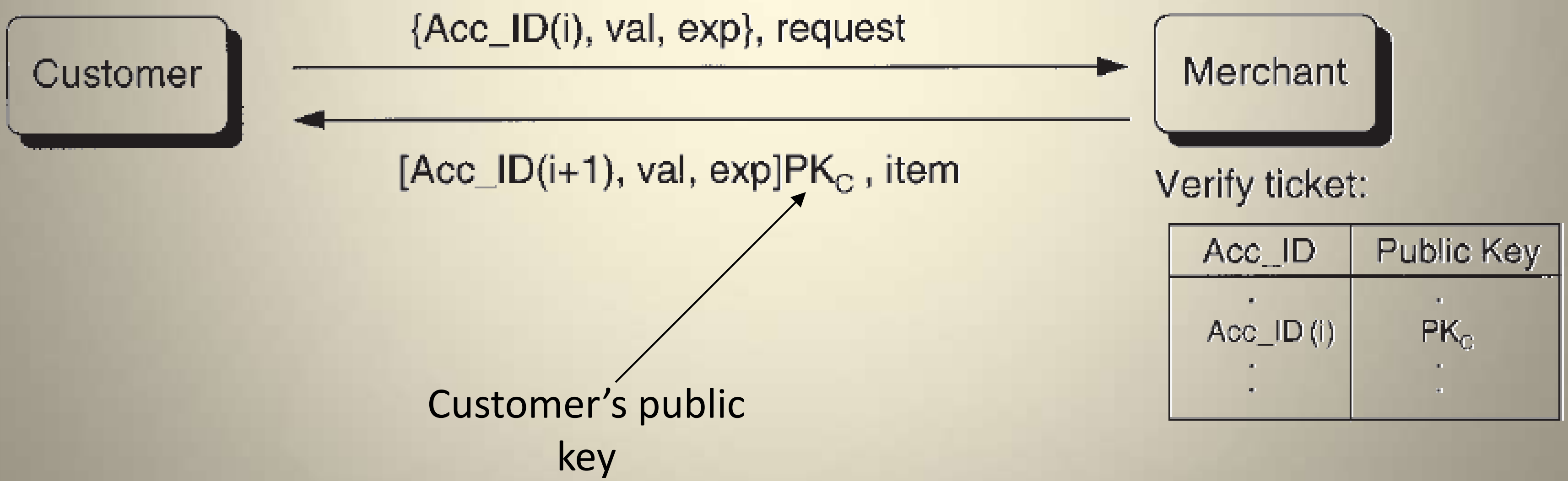
- vendor verifies a SubScrip ticket by checking SubScrip ticket against his database.

Security And Privacy

- Eavesdropper
  - Obtains a valid account ID and spent by an attacker
- Active attacker
  - Prevent the ticket to reaching its destination to retransmit it (stolen ticket) later

Protected SubScrip

- Using public-key cryptography



Refunding SubScrip

- To allow customers to convert unspent tickets back to real money.
- Vendor pay the remaining account balance to the user by existing macropayment system.

Lost Tickets

- Unsuccessful transmission
- Software failure

- Lost account ids recovery
  - Sending the delivery address approximate time of last access to regain the account

Micropayment Systems

Chapter 7

Contents

PayWord

Chapter 7.3

PayWord

- Designed by R.Rivest and A.Shamir
- Performance
  - Using hash functions
  - Reducing the number of public-key operations
- Each hash value, called a PayWord

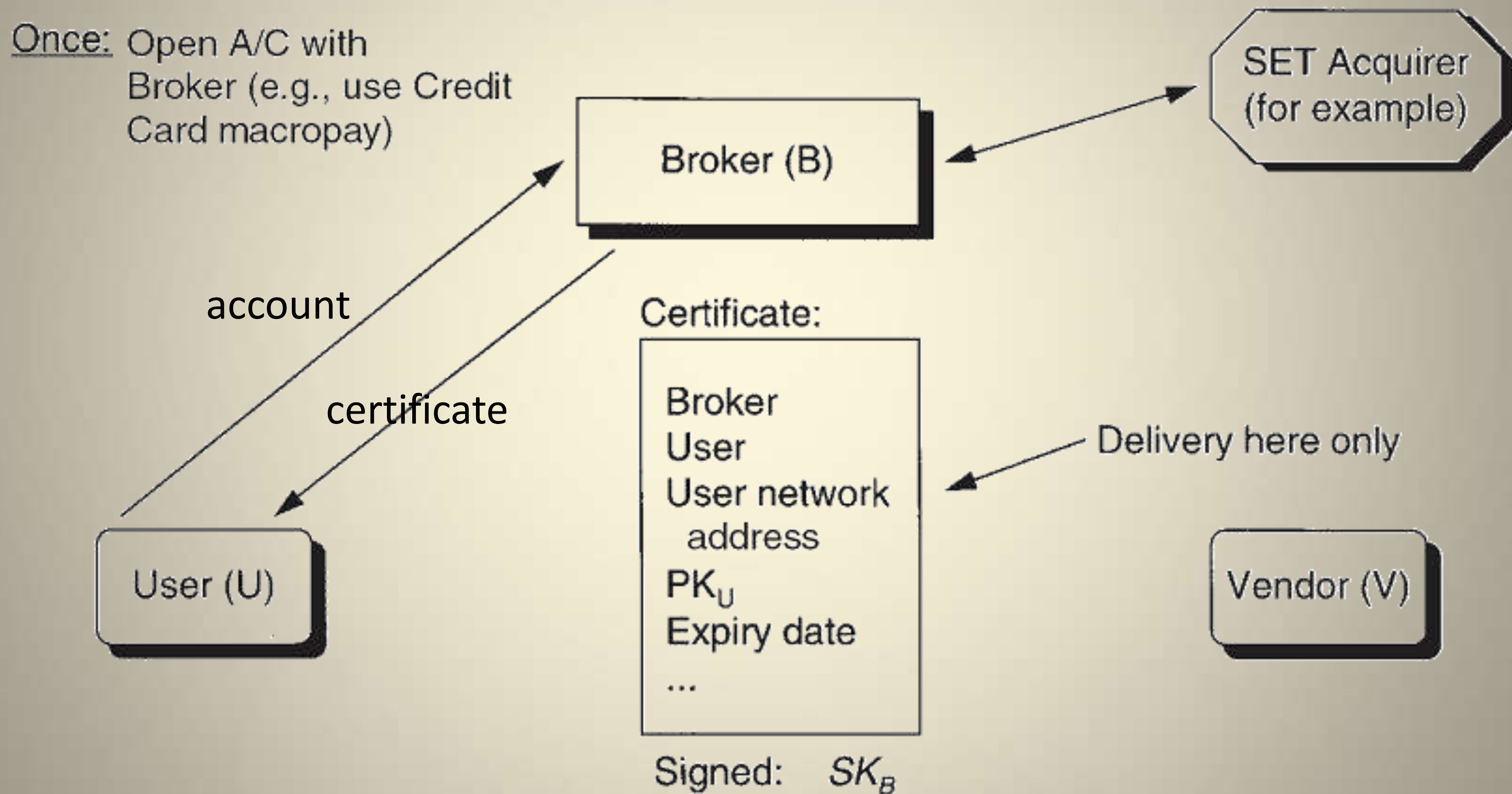
PayWord

- A PayWord chain is vendor-specific and the user digitally signs a commitment to honor payments for that chain.
- PayWord certificate by a broker

PayWord User Certificates

- Authorizes a user to generate PayWord chains
- Guarantees that a specific broker will redeem them

Obtaining a PayWord User Certificate



Certificate

Broker
User
User network address
PK_U
Expiry date
...

Signed: SK_B

No anonymity

$$C_U = \{B, U, A_U, PK_U, E, I_U\}\ SK_B$$

The user.s public key

- Optional information
  - Credit limits per vendor
  - User-specific details
  - Broker details

Verifying a Broker's Signature

- vendor must securely obtain that broker's public key, PK_B, in some way.
- It is not discussed in the PayWord scheme.
  - Implementation specific.

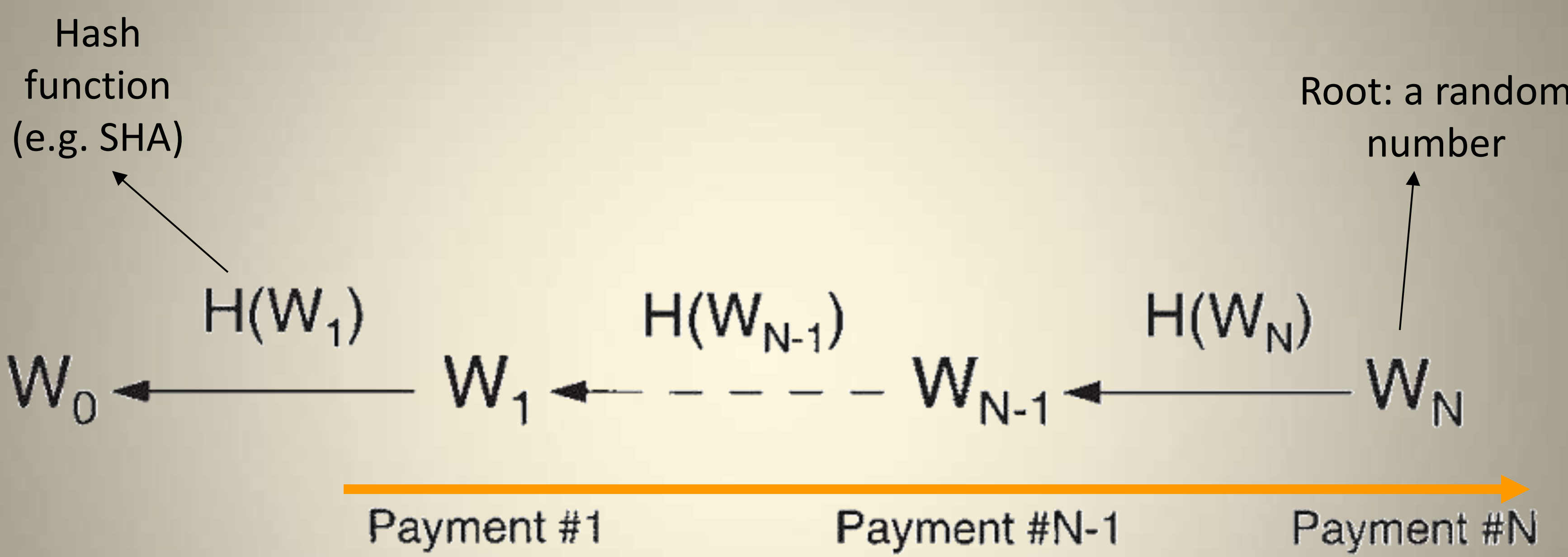
Revoked Certificates

- Secret key of user was lost or stolen
- Broker maintains blacklists of (revoked) certificates
- Vendor must obtain revoked certificate lists from a broker

PayWord Chains

- Represents user credit at a specific vendor
- Each PayWord (hash value) in the chain has the same value, normally 1 cent.

User Generating A PayWord Chain



Commitment To A PayWord Chain

- The user signs commitment
- The commitment will authorize the broker to redeem any PayWords from the committed chain

$$Comm = \{V, C_U, W_0, E, I_{Comm}\}\ SK_U$$

vendor

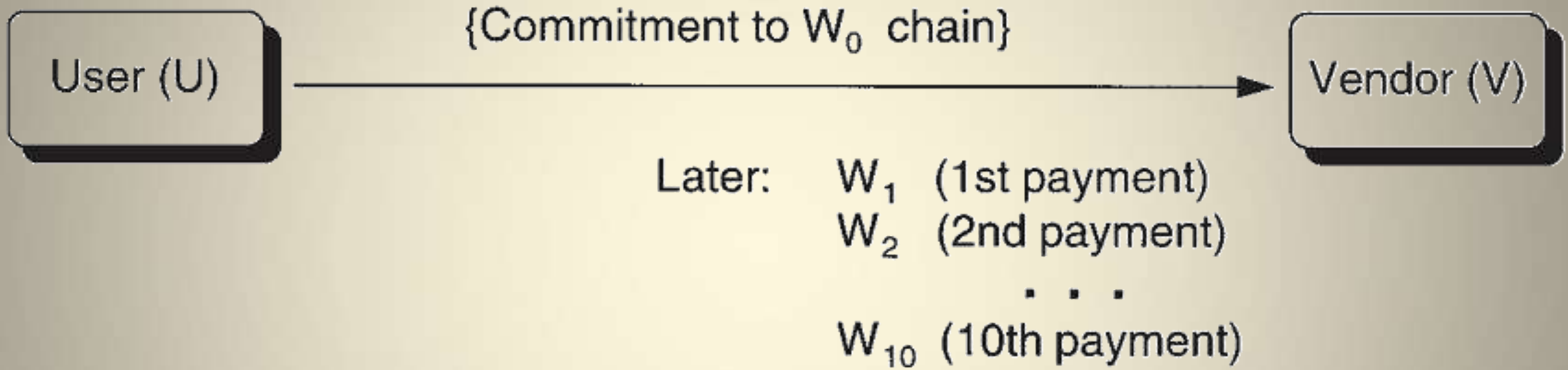
User’s PayWord certificate

The final hash

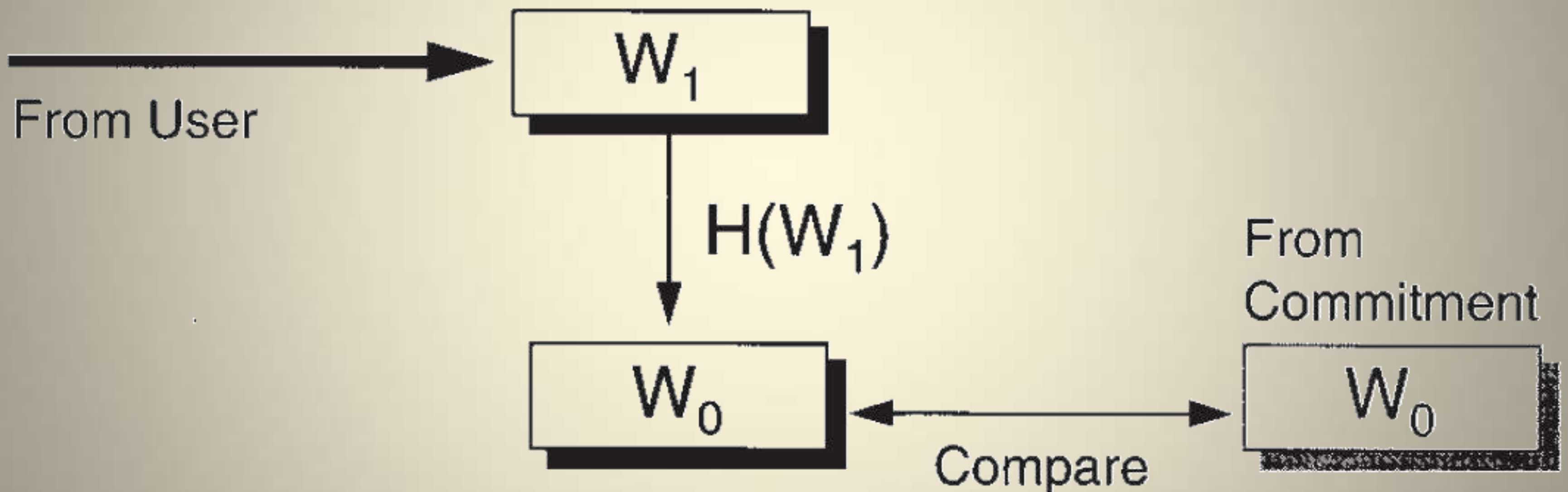
Expire date

use’s secret key

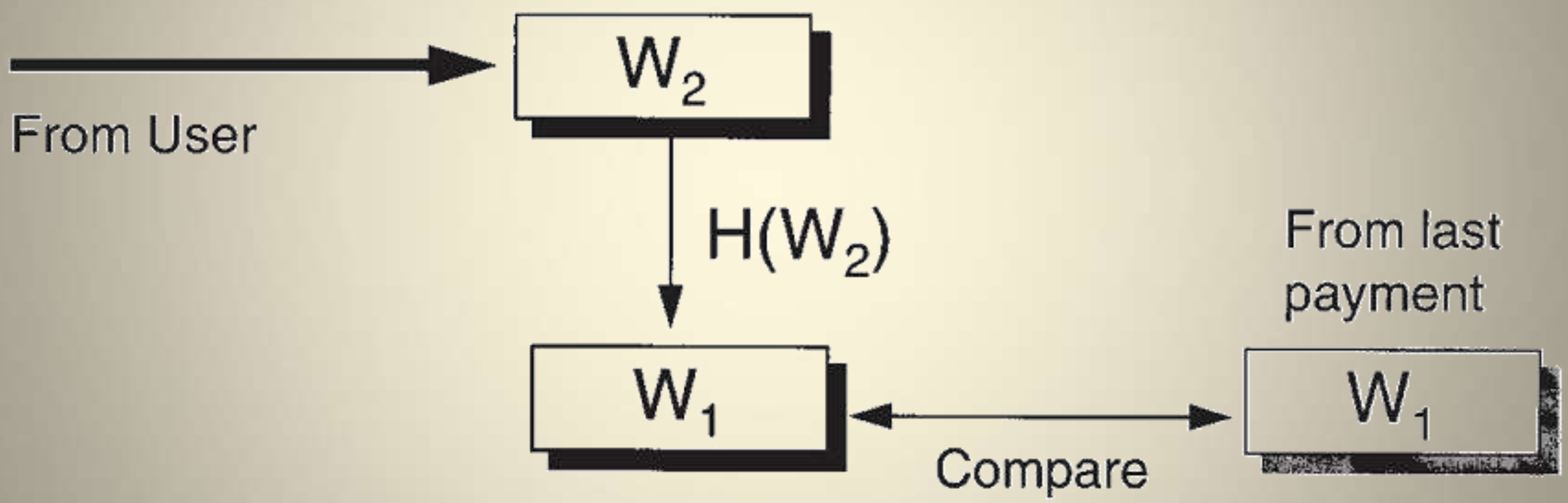
Spending PayWords



Verifying The First PayWord



Verifying Further Payments

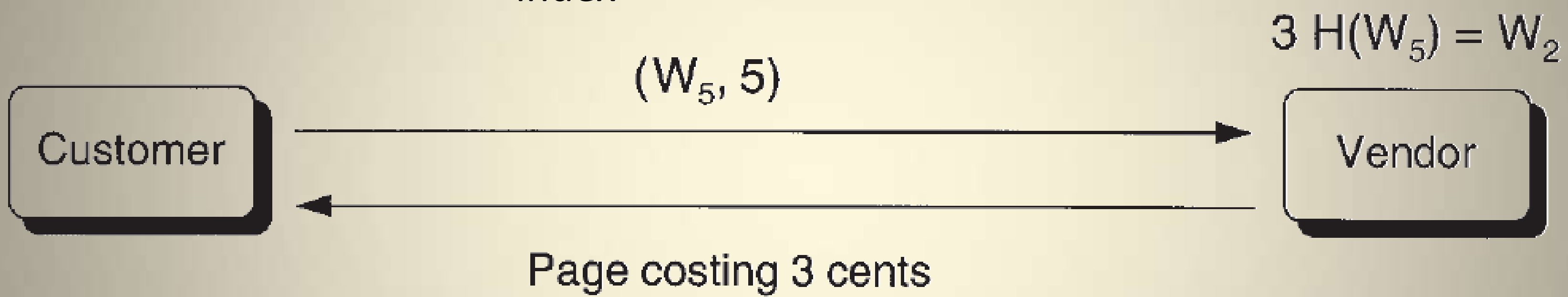


To make a further 1-cent payment, the user will send W2. The vendor then compares the value obtained by taking the hash of W2, H(W2) to the previous valid PayWord (W1) received. If W2 is valid, then the values will match

Variable-size Payments

$$P = (W_i, i)$$

index

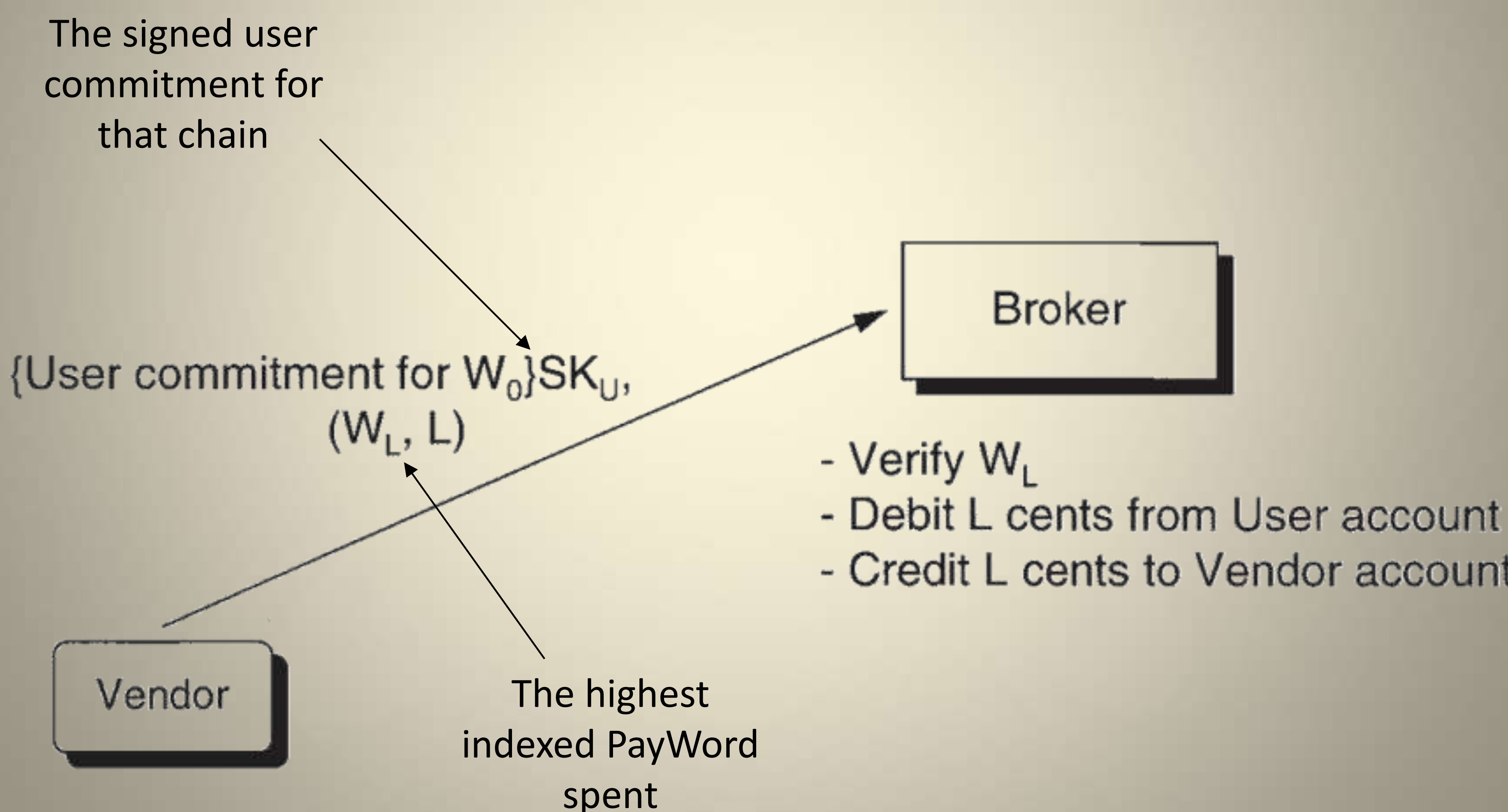


To make a 3-cent payment after having spent W2, the fifth PayWord, W5, can be sent.

Redeeming Spent PayWords

- To receive payment a vendor redeems PayWord chains with the appropriate broker, perhaps at the end of each day.

Reclaiming PayWords with a broker



Computational Costs

- Broker
  - One signature/user/month (C_U);
  - One signature verification/user/vendor/day (Commitment);
  - One hash per PayWord spent.
- Vendor
  - Two signature verifications/user/day (Commitment and CU);
  - One hash per PayWord spent.
- User
  - One signature/vendor/day (Commitment);
  - One hash per PayWord constructed.

Extensions

- PayWords of different values at the same merchant.
- A commitment could be used as a simple electronic check to make a macropayment.

Micropayment Systems

Chapter 7

Contents

MicroMint

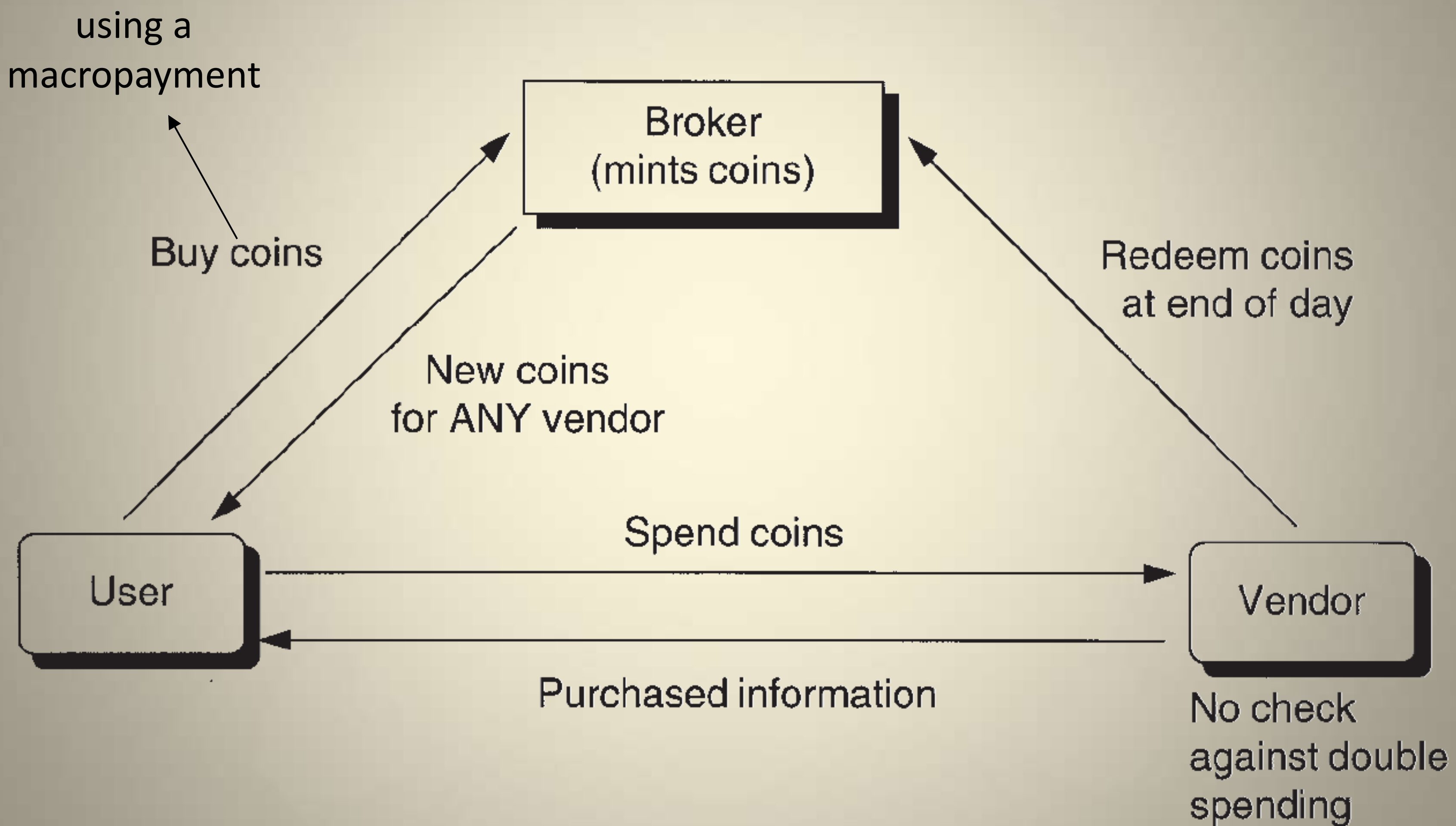
Chapter 7.6

Introduction

Rivest, R., and A. Shamir, "PayWord and MicroMint: Two Simple Micropayment Schemes," *Proc. 4th Security Protocols International Workshop (Security Protocols)*, Lecture Notes in Computer Science, Vol. 1189, Berlin: Springer-Verlag, 1996, pp. 69–87.

- It is very computationally difficult for anyone except the broker to mint valid coins.
- It is quick and efficient for anyone to verify a coin.

The MicroMint model



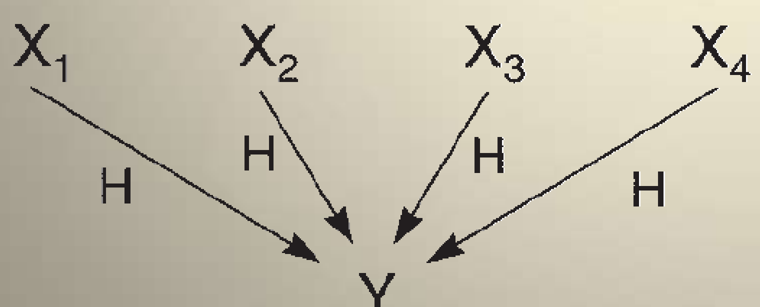
MicroMint

- k-way hash function collision

$$H(x_1) = H(x_2) = H(x_3) \ldots . = H(x_k) = y$$

- Four-way hash function collision

$$H(x_1) = H(x_2) = H(x_3) = H(x_4) = y$$



$$C = \{x_1, x_2, x_3, x_4\}$$

A coin with worth 1 cent has four input values

Verifying a Coin

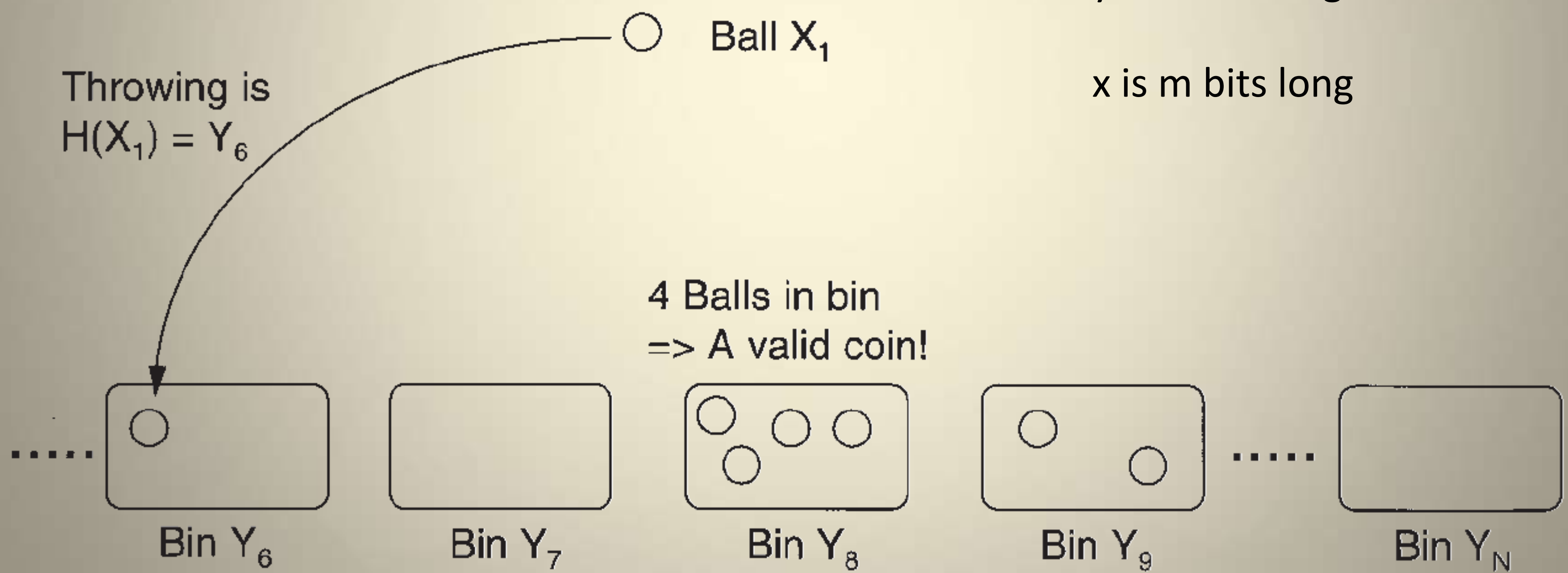
- Performing four hashes on each x_i to obtain the same y value
- Ensuring that each x is different
- To detect double spending
  - The broker maintains a copy of each coin already spent to check against

Minting Coins

- Throwing a ball (x) into one of 2^n bins (y values)

y is n bits long

x is m bits long



Computational Costs

- Balls are thrown at random and cannot be aimed at a specific bin
- The first coin (k-way collision)

$$T = 2^{n(k-1)/k}$$

Computational Costs

- Minting more coins becomes progressively cheaper
  - $c.T$ hash operations produces c^k coins

Linear — Exponential

- If we produce the large number of coins (c^k) *then* the required number of effort to create a valid coins decreases to c.T.

Multiple Coins Per Bin

- If more than k balls fall into the same bin, several coins could be made from subsets of the values in the bin.

$$C_1 = \{x_1, x_2, x_3, x_4\}$$

$$C_2 = \{x_1, x_2, x_3, x_5\}$$

$$C_3 = \{x_1, x_2, x_4, x_5\} \text{ and so on}$$

Controlling the Number of Thrown Balls

- Substantial storage space
  - For remembering the value of each ball and the bin it landed in

$$y = a, b$$

The coin's hash value

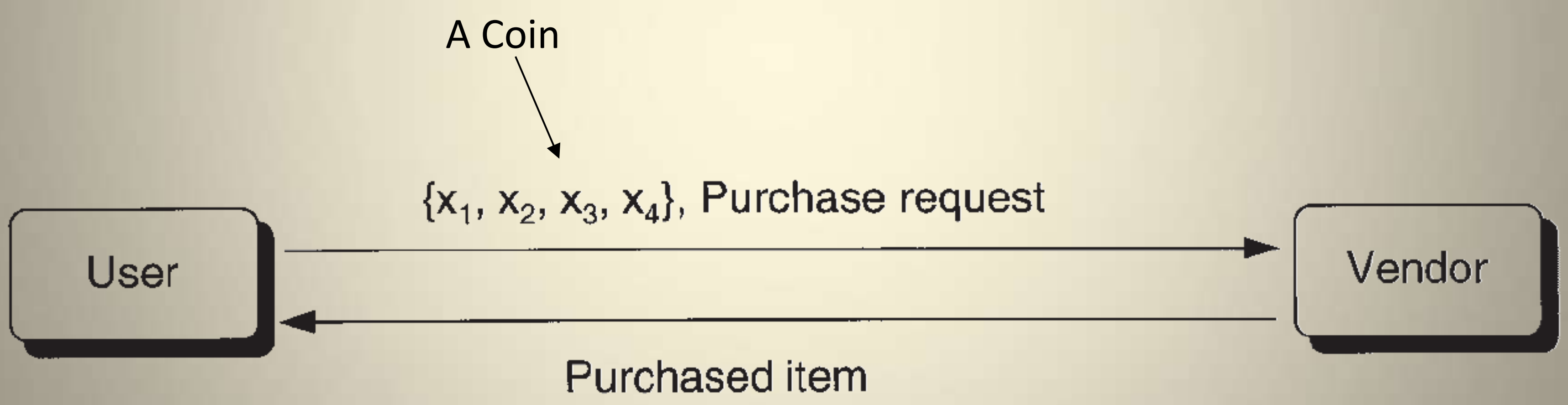
specific pattern

Broker only consider and saves x with prefix of ***a*** in its hash value.

Preventing Forgery

- Special hardware
- Short coin validity period
- Early minting
- Coin validity criterion
  - The same hash prefix
- Different bins (y values)
- $k > 2$ e.g. $k=4$

A MicroMint purchase



Double Spending

- No check is performed at the vendor against double spending
- Vendors might try to redeem coins already spent at other vendors
- No anonymity offers detecting doubly spent coins
  - Repeat offenders are blacklisted and denied further access to the system.

Hidden Predicates

- The x value must have certain properties initially known only to the broker
- A vendor can verify that a coin is valid by checking that it obeys the predicate published by the broker.

User-specific Coins

- This ensures that stolen coins cannot be spent by most users
- Example
  - All coins have this property:

$$h2\ (\mathrm{Coin}) = h2(U) = \mathrm{GID}$$

A Short
Length hash
e.g. 16 bit

Group Identity

Vendor-specific Coins

- They may only be redeemed by a small group of vendors.

Coins for multiple months

- To mint some of the coins for several different months at the same time
- Since the broker can effectively mint coins faster, the process can be slowed down by making them harder to mint

Different-valued Coins

- Coins could be worth different values, according to predicates on the x values.
- These predicates might be announced at the start of the month and could be verified by anyone.