PayWord and MicroMint: Two Simple MicroPayment Schemes

Ronald L. Rivest (MIT)

Adi Shamir (Weizmann)

Outline

- Micropayments: Framework and Motivation
- PayWord: a credit-based scheme using chains of hash values (or *paywords*):

  $$w_0 \leftarrow w_1 \leftarrow w_2 \leftarrow w_3 \leftarrow \ldots$$

- MicroMint: digital coins as *k*-way hash function collisions: $x_1 \quad x_2 \quad x_3 \quad x_4 \rightarrow y$
- Conclusions

Micropayments

- Payment scheme for *low-value* transactions, such as 1¢ per web page access
- Too small for credit-card “macropayments” (which may incur fee of 29 ¢ + 2%)
- Public-key crypto relatively expensive:

RSA sign (private key)	2 / sec
RSA verify (public key)	200 / sec
Hash function	20000 / sec

RSA DATA SECURITY, INC.

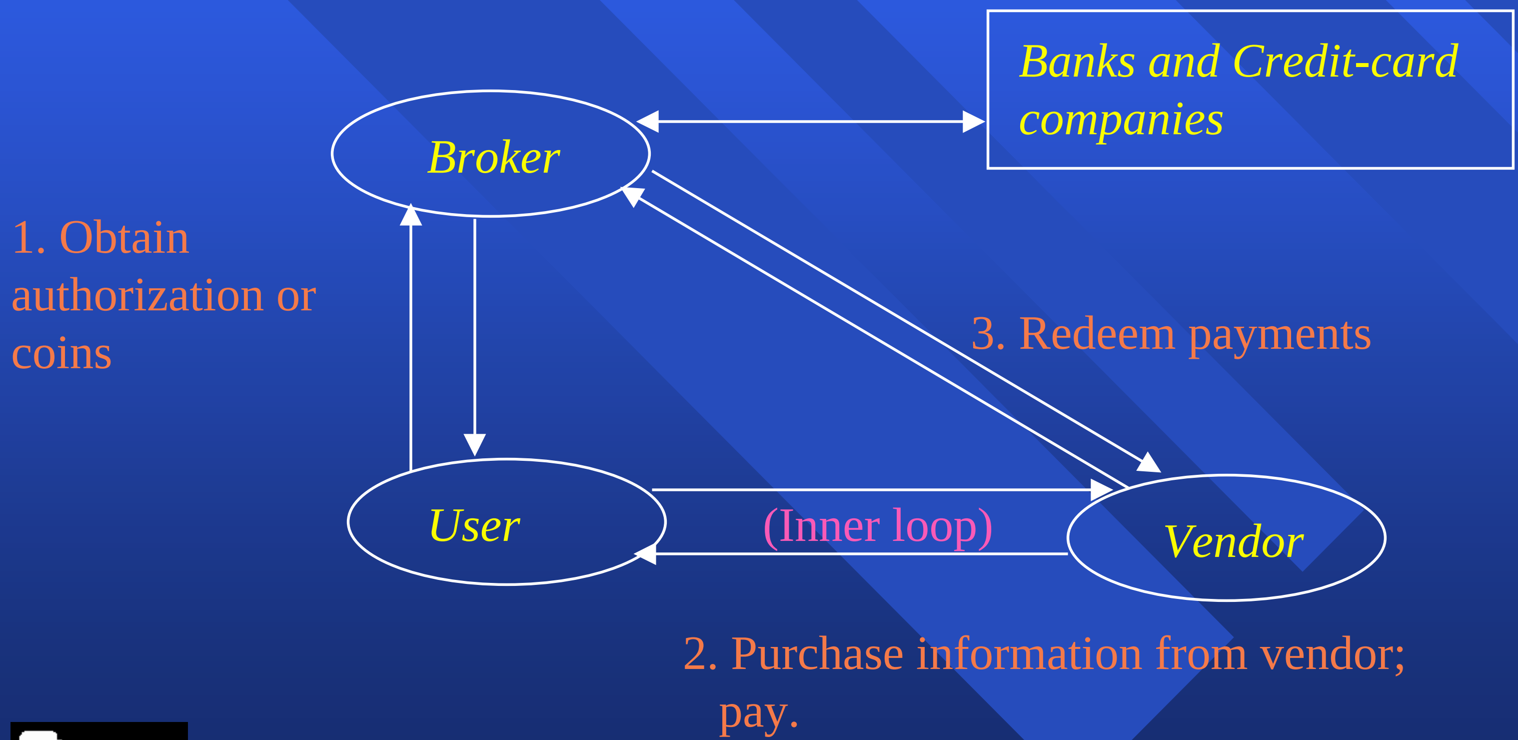
Micropayments

- Some advanced features, such as *anonymity*, are probably just too expensive to implement in a micropayment scheme.
- With light-weight schemes, one must be pragmatic about fraud and abuse: the goal should be effective *risk management*, rather than total prevention. "Bad apples" can be detected and eliminated from the system.

RSA
DATA SECURITY, INC.

Micropayments

- Introduce ***Broker*** to intermediate and aggregate:



Efficiency Goals

- Try to minimize use of public-key operations.
- Try to keep Broker “off-line” as much as possible.
- Make inner loop (purchase/payment) efficient, especially for repeated small purchases.

PayWord

PayWord Chains

- $w_0 \xleftarrow{h} w_1 \xleftarrow{h} w_2 \xleftarrow{h} w_3 \xleftarrow{h} \ldots \xleftarrow{h} w_n$
- **Easy for User to create a chain of length, say, $n = 1000$ for a vendor using h = MD5, by starting with w_n.**
- **User commits (signs with RSA) "root" w_0 over to Vendor.**
- **User makes successive 1¢ payments by revealing "paywords" w_1, w_2, ... in turn.**
- **Vendor redeems commitment, and last payword received, with Broker.**

PayWord Certificate

- **Broker gives User a signed "certificate" C_U good for one month authorizing User to make PayWord chains. Broker is extending credit to User.**
- C_U = *{Broker, User, User's IP Address,*

  PK_U*, expiration-date, limits, etc.*$\}_{SKB}$

  where

  PK_U *= User's Public RSA Key.*
- **Certificate authorizes delivery of goods only to specified Internet address.**

RSA DATA SECURITY, INC.

PayWord Commitment

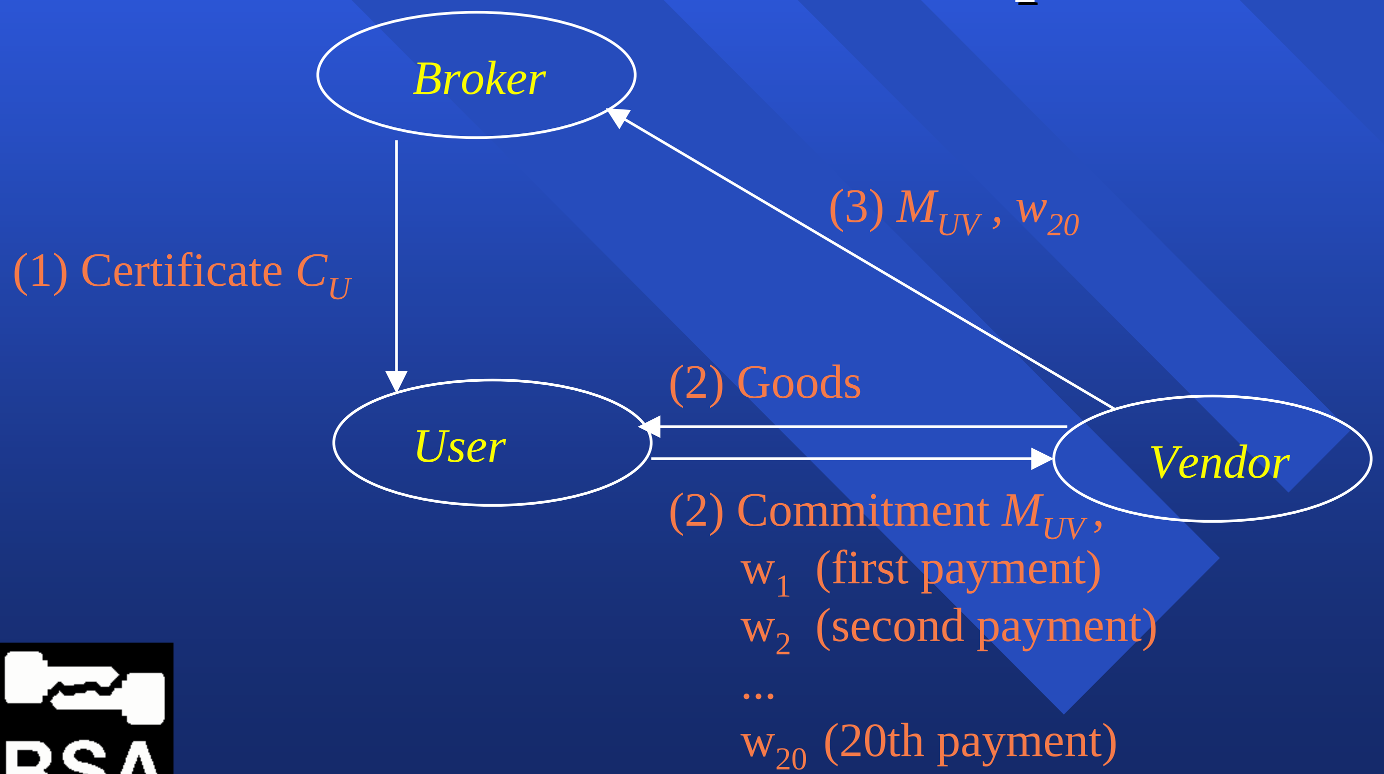
- User commits root w_0 to Vendor by signing a commitment message M_{UV}:

$$M_{UV} = \{User, Vendor, w_0, C_U, \; expiration\text{-}date\}_{SK_U}$$

- Commitment contains User's certificate C_U.
- User commits to PayWord chain for, say, one day.
- Note that Broker is *not* directly involved.

PayWord

- **Basic PayWord information flow. Note that Broker is *off-line* except for issuing monthly certificate and final redemption.**



PayWord Costs

- One signature by Broker / user / month (C_U)
- One signature by User / vendor / day (M_{UV})
- Two verifications by Vendor / user / day (C_U and M_{UV})
- One verification by Broker / user / vendor /day (for M_{UV})
- One hash function computation by each of User, Vendor, and Broker for each 1¢ payment.

RSA DATA SECURITY, INC.

PayWord Extensions

- Can pay for a 5¢ item by revealing w_{10} after w_5 (like revealing five paywords at once).
- Can have several payword chains per commitment, with different values per payword in each chain: e.g. a chain of 1¢ paywords, a chain of 25¢ paywords, and a chain of 1$ paywords.

RSA DATA SECURITY, INC.

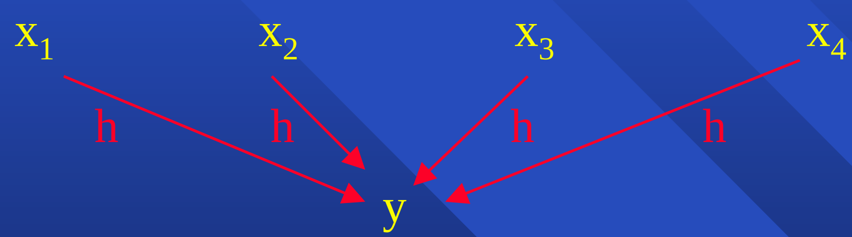
MicroMint

MicroMint

- **A digital coin should be:**
  - Hard to produce [except by Broker]
  - Easy to verify [by anyone]
- **Digital signatures "work," but are relatively expensive.**
- **MicroMint uses hash functions *only* (<u>no</u> public-key crypto).**
- **Broker utilizes *economy of scale* to produce MicroMint coins cheaply (as with a regular mint).**

MicroMint Coins

- **Suppose hash function *h* : $\{0,1\}^{48} \longrightarrow \{0,1\}^{36}$ maps *m = 48*-bit strings to *n = 36*-bit strings.**
- **A *k-way collision* is a *k*-tuple $(x_1, x_2, \ldots, x_k)$ of values such that $h(x_1) = h(x_2) = \ldots = h(x_k)$:**



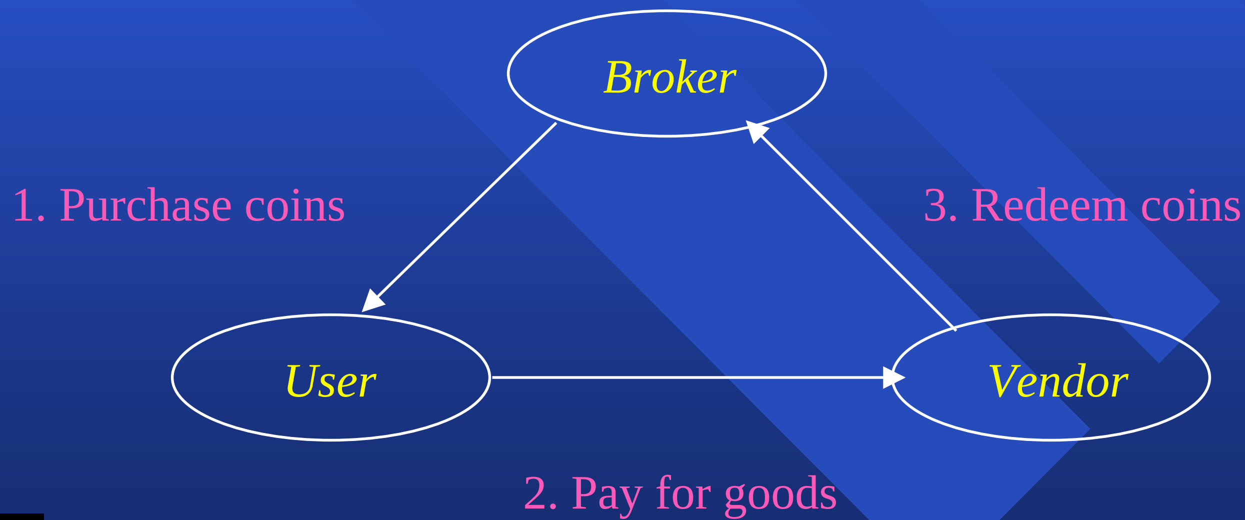
- **<u>A MicroMint coin is a *k*-way collision</u> (*k=4*). Verifying a coin is easy.**

Minting Coins

- **Producing coins is like tossing balls into 2^n bins; k balls in a bin makes one coin.**
- **Producing *first* 2-way collision requires time $2^{n/2}$; this is the "birthday paradox."**
- **Producing *first* k-way collision requires time $N_k = 2^{n(k-1)/k}$. (e.g. 2^{27} for $k=4$, $n = 36$.) (It's hard to forge even one coin.)**
- **Time cN_k yields c^k coins; once threshold of N_k is passed, coins are produced rapidly. (Mint has economy of scale).**

Flow of MicroMint Coins

1. **Broker mints coins and sells them to User.**
2. **User spends coins with Vendor.**
3. **Vendor deposits coins back to Broker.**



Security Concerns

- **Forgery**: Can an adversary forge MicroMint coins? (Economically?)
- **Double-spending**: What if a user "double-spends" his MicroMint coins?
- **Vendor fraud**: What if a vendor gives copies of coins received to an accomplice?
- **Framing**: Can vendor "frame" user for double-spending, or user "frame" vendor for fraud?

RSA
DATA SECURITY, INC.

Protections against forgery

- Computational difficulty of minting coins.
- Small-scale forgery not really a concern; large-scale forgers will get caught.
- Coins "expire" monthly. New hash function revealed each month, and old coins exchanged for newly minted ones. (Broker works during May to make coins good for June; forger only learns h_{June} at beginning of June, and so starts out way behind.)

Protection against double-spending

- There is no “anonymity” in MicroMint: the Broker keeps track of whom each coin was sold to, and notes when it is returned by vendor.
- Small-scale double-spending not a concern.
- A user whose coins are consistently double-spent (with many vendors) will be caught and black-listed; he will not be sold any more MicroMint coins.

RSA
DATA SECURITY, INC.

Protection against vendor fraud

- Vendors who consistently redeem coins that are also redeemed by other vendors will be black-listed and refused further redemption service by the Broker.
- Users can cooperate with Broker to identify bad vendors by identifying where coin was first spent.

Protection against framing

- It may be difficult for Broker to distinguish user double-spending from vendor fraud.
- Small-scale double-spending or fraud not a concern. Large-scale cheaters should be distinguishable by weight of evidence against them.

RSA DATA SECURITY, INC.

Additional protection against forgery

- Coins may satisfy "hidden predicates" which are only announced if forgery is detected by Broker.
- For example, legitimate coins may all satisfy condition that low-order bit of x_i is equal to some complicated function of other bits.
- Forger's coins will typically not pass this additional "verification condition".
- Broker can announce several such conditions (or even one each day of month).

Related Micropayment Schemes

- **Millicent (Manasse et al. / DEC)**
  - “Scrip” for each vendor, broker on-line.
- **NetBill (Tygar / CMU)**
  - Heavy use of public-key crypto.
- **NetCard (Anderson / Cambridge)**
  - Similar to PayWord, but bank signs commitments.
- **CAFE (Pederson and IBM Zurich)**
  - Similar to PayWord, but not credit-based.

RSA DATA SECURITY, INC.

Conclusions

- **We have presented two new micropayment schemes, PayWord and MicroMint, that minimize or eliminate public-key operations.**
- **PayWord/ MicroMint paper available from:** `http://theory.lcs.mit.edu/~rivest`