T-79.159 Cryptography and Data Security

Lecture 10: Electronic Cash and Oblivious Transfer

Helger Lipmaa
Helsinki University of Technology
helger@tcs.hut.fi

Overview of the Lecture

- Quick & Dirty Intro to Electronic Cash
- Motivation
- Simple protocols, their weaknesses
- More advanced protocols
- Briefly on oblivious transfer

Short lecture! (Enjoy the spring)

Conventional Payments

- Cash
  - ⋆ Cheap to operate
  - ⋆ Anonymous
  - ⋆ Reusable
- Cheque
- . . .

Electronic Payments: Current Situation

- Payment with Credit Cards
  - ⋆ Credit card frauds — add x%+y cents to the price
  - ⋆ Also high costs of transaction
  - ⋆ Thus: High cost, can't allow small payments
  - ⋆ Security — accidentally published credit card numbers
- Open an account at the seller
- Both are non-anonymous

Example: Account-Based System

- During opening an account, the bank of payer issues a corresponding signing key to the payer, together with certificate (his own signature on the key, account number, . . .)

- If the payer wants to buy something, he just signs a message ”Pay X euros to Y”, and gives it to the seller

- The seller forwards this signature to her bank, who will obtain X euros from payer’s bank and transfers it to the seller’s accout

- Standard: SET (includes additional features)

Faults of Account-Based System

- One big fault: non-anonymity
  - ⋆ Your bank will basically know what did you buy when and where...
  - ⋆ Similar to credits cards etc
  - ⋆ Do you want your bank to know what exactly you buy?
- Another fault: a coin can be reused

Desiderata from Electronic Cash

- Emulate real cash, possibly even improving upon it
- Anonymity: the seller does not know your identity, your bank does not know what you buy
- Transferability: same coin can be reused
- Cheap processing (computationally, communicationally)
  - ⋆ Since cash is "prepaid", it usually involves small units of money. Processing such units should be easy!
  - ⋆ Clearly, an anonymous system is more costly than a nonanonymous one

More About Anonymity

- Untraceability: Given the coin and a view of a protocol between the payer and the seller, one should not be able to guess payer's identity
- Unlinkability: Given several coins of the same user, and corresponding views together, one should not be able to determine whether or not the coins were paid by the same person
  - ⋆ Prepaid phone cards provide untraceability but not unlinkability
- Privacy can be computational, statistical or information-theoretical

An Anonymous E-Cash Protocol

- Basic idea: use blind signatures

- Conventionally:
  - ⋆ User writes “100 euros” on a paper, and puts the paper in envelope
  - ⋆ The bank signs the envelope (by using a special pen) so that the signature will also be seen on the paper
  - ⋆ The user takes paper out from the envelope and uses it later for payments
  - ⋆ The bank does not know what was written on the paper!

Recall: RSA Signatures

- RSA modulus: $n = pq$, p and q are two secret primes
- Secret exponent d, public exponent e, st $de \equiv 1 \mod \varphi(n)$
- H is a hash function
- RSA signing of message m: $s \leftarrow H(m)^d \mod n$
- RSA verification: $s^e \equiv^? H(m) \mod n$
- Correct, since $s^e \equiv H(m)^{de} \equiv H(m)$

Blind RSA Signatures

- User generates a random $r \leftarrow_R \mathbb{Z}_n$ and sends $m' \leftarrow r^e H(m)$ to Bank
- Bank signs m': $s' \leftarrow (m')^d \mod n$
- User verifies that s' is a signature on m'
- After that, she computes $s \leftarrow s'/r \mod n$
- $s \equiv \frac{s'}{r} \equiv \frac{(m')^d}{r} \equiv \frac{(r^e H(m))^d}{r} \equiv \frac{r^{ed} H(m)^d}{/} r \equiv H(m)^d \mod n$
- Thus s is a signature on m, and bank does not know m!

An Anonymous E-Cash Protocol, Cont

- Protocol:
  - ⋆ Coin withdrawal: User generates a new random coin m, and gets his bank's blind signature s on it, $s = H(m)^d \mod n$
  - ⋆ When buying something, user shows the coin to the seller, who verifies the signature
  - ⋆ Seller's bank later shows the coin to the user's bank, who transfers 100 euros to her

An Anonymous E-Cash Protocol, Problems

- We want to use coins of different size. However, due to blind signing, the seller does not know what is the amount that m signifies

- Solution: bank uses a different signing key for every amount

- Second solution: cut-and-choose
  - ⋆ The user generates 1000 coins of form $1000||r_i$, where r_i is random, and sends them in a blinded form to the bank
  - ⋆ The bank asks the user to unblind 999 randomly chosen coins
  - ⋆ If all them are correct, the bank blindly signs the 1000th coin

An Anonymous E-Cash Protocol, Problems

- This protocol does not protect against double spending

- On-line solution:

  ⋆ The bank maintains a database of used coins

  ⋆ The seller contacts the bank after the payment, and asks the bank whether this coin has been used before

- Problem: bank's database grows large, impractical

- Problem: can't guarantee online connection (at least sometimes); contacting bank takes resources, and slows down the sales

Off-line E-Cash

- Basic idea:
  - ⋆ Instead of preventing double-spending, enables to detect it
- Anonymity: if user does not double-spend, his identity is protected
- Double-spending: if user pays twice with the same coin, his identity can be computed
- High-value payments are (in ideal) done on-line, for low value payments, traceability after the fact might discourage double-spending

Chaum-Fiat-Naor Protocol. Coin Withdrawal

- User generates $2k$ messages of the form $H(m_i)||H(m_i \oplus Id)$, where Id is his unique identifier, and m_i is a random coin. He sends all of them blinded to the bank.

- Bank asks the user to unblind random k coins, and receives the corresponding values m_i and r_i (r_i is the blinding factor)

- If all k coins are correct, bank knows that "most" of the remaining coins are correct, and signs them

- The user obtains thus blind signatures on k messages of the form $H(m_i)||H(m_i \oplus Id)$

Chaum-Fiat-Naor Protocol. Payment

- The seller sends k bits $(c_1, \ldots, c_k)$ (a challenge) to the payer

- For $i \in [1, k]$:

  - ⋆ If $c_1 = 0$, the payer sends $m_i || H(m_i \oplus Id)$ to the seller. If $c_1 = 1$, the payer sends $H(m_i) || m_i \oplus Id$ to the seller.

  - ⋆ The seller can compute in both cases the value $H(m_i) || H(m_i \oplus Id)$, and verify the correctness of bank's signature on it

- The seller accepts the payment if all verifications succeed

Chaum-Fiat-Naor Protocol. Deposit

- The seller sends the challenge $(c_1, \ldots, c_k)$ and the k received messages to the payer's bank

- Now, if the same coin has been double-spent, with high probability the corresponding challenges differ at least in one coefficient, say ith

- Since $c_i \neq c_i'$, the bank has both $m_i || H(m_i \oplus Id)$ and $H(m_i) || m_i \oplus Id$. From m_i and $m_i \oplus Id$ he can compute the Id of the double-spender

Micropayments

- In above payment schemes, the seller must verify at least one signature per payment
- This is often too much (imagine a pay TV, when you have to pay 0.01 cents per second)
- Idea: compute a secret A_0, and issue $A_n = H^n(A_0) = H^{n-1}(H(A_0))$ as a coin
- After a second, release $A_{n-1} = H^{n-1}(A_0)$, then $A_{n-2} = H^{n-2}(A_0)$, etc

Micropayments

- Release of A_{n-i} means the payment of i coins
- The seller only has to remember the last A_{n-i}
- No anonymity

Final Remarks

- E-cash with untraceability is clearly less efficient than one without it
- Efficient on-line e-cash systems (that prevent double-spending) exist
- Similar off-line systems can be built by using secure hardware
- Otherwise, in off-line systems one can only detect double-spending

Advanced Properties

- Revocability
  - ⋆ Blackmailing, money laundering — it is desirable to be able to revoke the anonymity if some number of authorities collaborate
- Divisibility
  - ⋆ You receive a 100 euro coin from the bank, but want to use it for buying a coffee, disposable camera, some books and beer from different sellers
  - ⋆ Need protection against double-spending and unlinkability!
- Both objectives can be achieved

Oblivious Transfer

- Assume Bob has a database of N elements
- Alice pays to Bob \$1 to access one item thus Alice should not get to know more
- Bob should not get to know which item Alice retrieved
- Many applications: e.g., medical databases

AIR/HOT Protocols

- Assume we have a homomorphic public key cryptosystem
- Additional assumption: the order ω of plaintext space is either prime or hard to factor ($\mathbb{Z}_p$, $\mathbb{Z}_n$)
- The latter assumption can be weakened to "the smallest prime divisor of ω should be larger than N"

[Aiello/Ishai/Reingold 2001, Lipmaa 2003]

AIR/HOT Protocols

Bob has $\mu = (\mu_1, \ldots, \mu_N)$, Alice has σ. Alice wants to retrieve μ_σ

- Alice creates a new key pair and sends the public key to Bob
- Alice generates a random r and sends $c \leftarrow E_K(\sigma; r)$ to Bob
- For all $i \in [1, N]$ Bob does: Set $c_i \leftarrow (c \cdot E_K(-i; r_i))^{s_i} \cdot E_K(\mu_i; t_i)$, where r_i, s_i, t_i are newly generated random values
- Bob sends $(c_1, \ldots, c_N)$ to Alice
- Alice decrypts $\mu_\sigma = D_K(c_\sigma)$

AIR/HOT Protocols: Correctness

Bob has $\mu = (\mu_1, \dots, \mu_N)$, Alice has σ. Alice wants to retrieve μ_σ

- Recall $c \leftarrow E_K(\sigma; r)$ and $c_i \leftarrow (c \cdot E_K(-i; r_i))^{s_i} \cdot E_K(\mu_i; t_i)$
- Thus, $c_i = E_K(\mu_i + s_i(\sigma - i); s_i r_i + t_i)$
- When $i = \sigma$ then $c_i = E_K(\mu_i; s_i r_i + t_i)$
- When $i \neq \sigma$ then $c_i = E_K(*; s_i r_i + t_i)$ for a random $*$, since $i \neq \sigma$ and s_i is random

OT: some applications

- Coin tossing: Bob creates two ciphetexts, one of them encrypts 1, another one encrypts 1. Bob proves to Alice that he encrypted correctly. Alice picks a random one.

- Yao's circuit evaluation: a garbled input goes to the circuit. Alice should not get to know the garbled value of another input. Bob should not know which input is used.

- Private Equality Test: Alice has private input a, Bob has private input b. Alice must get to know whether $a = b$ and nothing more. Exercise: how to do it?