

Introduction to Cryptography Lecture 11

Verifiable secret sharing, electronic cash

Benny Pinkas

t -out-of- n secret sharing

- Shamir's secret sharing scheme:
 - Choose a large prime and work in the field Z_p .
 - The secret S is an element in the field.
 - Define a polynomial P of degree $t-1$ by choosing random coefficients a_1, \dots, a_{t-1} and defining
$$P(x) = a_{t-1}x^{t-1} + \dots + a_1x + S.$$
 - The share of party j is $(j, P(j))$.
- Reconstructing the secret:
 - Assume we have $P(x_1), \dots, P(x_t)$.
 - Use Lagrange interpolation to compute the unique polynomial of degree $\leq t-1$ which agrees with these points.
 - Output the free coefficient of this polynomial.

t -out-of- n secret sharing

- Lagrange interpolation
 - $P(x) = \sum_{i=1..t} P(x_i) \cdot L_i(x)$
 - where $L_i(x) = \prod_{j \neq i} (x - x_j) / \prod_{j \neq i} (x_i - x_j)$
 - (Note that $L_i(x_i) = 1$, $L_i(x_j) = 0$ for $j \neq i$.)
 - I.e., $S = \sum_{i=1..t} P(x_i) \cdot L_i$, where $L_i = \prod_{j \neq i} x_j / \prod_{j \neq i} (x_i - x_j)$.
- Malicious modification threats
 - Users might receive corrupt shares (perhaps intentionally by sender). This is undetectable until secret reconstruction (or not at all).
 - Malicious user might send a corrupt share. Real secret cannot be recovered, except by the malicious user.

Verifiable secret sharing [Feldman]

- Work in Z_p . Publish a generator g .
- $P(x) = a_{t-1}x^{t-1} + \dots + a_1x + S$.
- The sender also publishes
- User i receives $P(i)$, and verifies that

- The same verification is done when users publish their shares.

- This scheme is
 - Unconditionally binding. I.e., even an adversary with infinite power cannot use corrupt shares. (why?)
 - Computationally hiding. I.e. finding S (for a randomly chosen S), is equivalent to computing discrete logs. (why?)

Distributed security

- Disclosure of the private key is disastrous.
- Suppose we have n servers, and we wish that
 - Any subset of t of them is able to decrypt.
 - Breaking into $t-1$ servers does not reveal the key.
- Solution 1:
 - Construct a t -out-of- n secret sharing scheme, where the private key, k , is the secret. Give each server a share.
 - When t servers want to decrypt they reconstruct the secret key k and use it for decryption (say, by one of the servers).
 - Problem: After the secret key is reconstructed, breaking into a single server reveals it.

The El Gamal public key encryption system

- Recall the El Gamal public key encryption system.
 - Public information (can be common to different public keys):
 - A prime $p=2q+1$, and a generator g of $H \subset Z_p^*$ of order q .
 - Private key: $0 < k < q$.
 - Public key: $h=g^k \text{ mod } p$.
 - Encryption of message $m \in H \subset Z_p^*$
 - Pick a random $0 < r < q$.
 - The ciphertext is $(g^r, h^r \cdot m)$.
 - Decryption of (s,t)
 - Compute $t/s^k = (h^r \cdot m) / (g^r)^k$
- } Using public key alone
- } Using private key

Distributed El Gamal

- The secret key k is shared between the n servers using t -out-of- n secret sharing:
 - $P(0)=K$. Server j receives $P(j)$.
- Encryption is done as usual
 - Pick a random $0 < r < q$. The ciphertext is $(g^r, h^r \cdot m)$.
- Decryption:
 - Each server j computes $(g^r)^{P(j)-L_j}$
 - The servers compute $\prod (g^r)^{P(j)-L_j} = g^{r \sum P(j)-L_j} = g^{rP(0)} = g^{rK} = h^r$, use it for decryption.
 - The secret key k is never revealed. (Except for when it is generated. However, key generation can also be done in a distributed way.)

Electronic cash

Simple electronic checks

- A payment protocol:
 - Sign a document transferring money from your account to another account
 - This document goes to your bank
 - The bank verifies that this is not a copy of a previous check
 - The bank checks your balance
 - The bank transfers the sum
- Problems:
 - Requires online access to the bank (to prevent reusage)
 - Expensive.
 - The transaction is traceable (namely, the bank knows about the transaction between you and Alice).

First try at a payment protocol

- Withdrawal
 - User gets bank signature on {I am a \$100 bill, #1234}
 - Bank deducts \$100 from user's account
- Payment
 - User gives the signature to a merchant
 - Merchant verifies the signature, and checks online with the bank to verify that this is the first time that it is used.
- Problems:
 - As before, online access to the bank, and lack of anonymity.
- Advantage:
 - The bank doesn't have to check online whether there is money in the user's account.
 - In fact, there is no real need for the signature.

Anonymous cash via blind signatures

- The bank signs the bill without seeing it
 - (e.g. like signing on a carbon paper)
- RSA Blind signatures (Chaum)
- RSA signature: $H(m)^{1/e} \bmod n$
- Blind RSA signature:
 - Alice sends Bob $(r^e H(m)) \bmod n$, where r is a random value.
 - Bob computes $(r^e H(m))^{1/e} = r H(m)^{1/e} \bmod n$, and sends to Alice.
 - Alice divides by r and computes $H(m)^{1/e} \bmod n$
- Problem: Alice can get Bob to sign anything, Bob does not know what he is signing.

Enabling the bank to verify the signed value

- "cut and choose" protocol
- Suppose Alice wants to sign a \$20 bill.
 - She prepares 100 different \$20 bills for blind signature, and sends them to the Bank (Bob).
 - The bank chooses 99 of them and asks Alice unblind them (divide by the corresponding r values). It verifies that they are all \$20 bills.
 - The bank blindly signs the remaining bill and gives it to Alice.
- If Alice tries to cheat she is caught with probability 99/100.
- 100 can be replaced by any parameter m .
- We would have preferred an exponentially small cheating probability.

Exponentially small cheating probability

- Define that a \$20 bill is valid if it is the e^{th} root of the multiplication of 50 values of the form $H(x)$, (H is one-way) and the owner of the bill can present all 50 x values.
- The withdrawal protocol:
 - Alice sends to the Bank z_1, z_2, \dots, z_{100} (where $z_i = r_i^e \cdot H(x_i)$).
 - The Bank asks Alice to reveal $\frac{1}{2}$ of the values $z_i = r_i^e \cdot H(x_i)$.
 - The Bank verifies them and extracts the e^{th} root of the multiplication of all the other 50 values.
- Payment: Alice sends the signed bill and reveals the 50 preimage values. The merchant sends them to the bank which verifies that they haven't been used before.
- Alice can only cheat if she guesses the 50 locations in which she will be asked to unblind the z_i s, which happens with probability $\sim 2^{-100}$.

January 9, 2005

Introduction to Cryptography, Benny Pinkas

page 13

Online vs. offline digital cash

- We solved the anonymity problem, while verifying that Alice can only get signatures on bills of the right value
- The bills can still be duplicated
- Merchants must check with the bank whenever they get a new bill, to verify that it wasn't used before.
- A new idea:
 - During the payment protocol the user is forced to encode a random identity string (RIS) into the bill
 - If the bill is used twice, the RIS reveals the user's identity and she loses her anonymity.

January 9, 2005

Introduction to Cryptography, Benny Pinkas

page 14

Offline digital cash

Withdrawal protocol:

- Alice prepares 100 bills of the form
 - {I am a \$20 bill, #1234, $y_1, y'_1, y_2, y'_2, \dots, y_m, y'_m$ }
 - S.t. $\forall i y_i = H(x_i), y'_i = H(x'_i)$, and it holds that $x_i \oplus x'_i = \text{Alice's id}$, where $H()$ is a collision resistant function.
- Alice blinds these bills and sends to the bank.
- The bank asks her to unblind 99 bills and show their x_i, x'_i values, and checks their validity. (Alternatively, as in the previous example, Alice can do a check with fails with only an exponential probability.)
- The bank signs the remaining blinded bill.

January 9, 2005

Introduction to Cryptography, Benny Pinkas

page 15

Offline digital cash

Payment protocol:

- Alice gives a signed bill to the vendor
 - {I am a \$20 bill, #1234, $y_1, y'_1, y_2, y'_2, \dots, y_m, y'_m$ }
- The vendor verifies the signature, and if valid sends to Alice a random bit string $b = b_1 b_2 \dots b_m$ of length m .
- $\forall i$ if $b_i = 0$ Alice returns x_i , otherwise ($b_i = 1$) she returns x'_i
- The vendor checks that $y_i = H(x_i)$ or $y'_i = H(x'_i)$ (depending on b_i). If this check is successful it accepts the bill. (Note that Alice's identity is kept secret.)
- Note that the merchant does not need to contact the bank during the payment protocol.

January 9, 2005

Introduction to Cryptography, Benny Pinkas

page 16

Offline digital cash

- The merchant must deposit the bill in the bank. It cannot use the bill to pay someone else.
 - Because it can't answer challenges b^* different from the challenge b it sent to Alice.
- How can the bank detect double spenders?
 - Suppose two merchants M and M^* receive the same bill
 - With very high probability, they send *different* queries b, b^*
 - Suppose $b_i=0, b_i^*=1$. Then M receives x_i and M^* receives x_i'
 - When they deposit the bills the bank receives x_i and x_i' , and can compute $x_i \oplus x_i' = \text{Alice's id}$.