Introduction to Modern Cryptography

Benny Chor

Threshold Cryptography, Revisited
t-out-of-n ElGamal Decryption

Zero Knowledge Proofs

Lecture 11

Tel-Aviv University

28 December 2009; revised January 16, 2010

t-out-of-n Secret Sharing, Revisited

Scenario:

- A value $S \in \mathcal{U}$ is the secret key allowing access or activation of an extremely critical and sensitive device.
- A "trusted dealer" holds S, but does not wish to activate the device right now.
- This dealer wants to delegate the secret to n parties.
- The parties can only be partially trusted: We seek a mechanism that will enable any t parties to reconstruct S, but any subset of $t-1$ parties (or less) cannot get any partial information on S.
- Computational requirements: Reconstruction should be efficient (as a function of length of S and of number of parties n).
- Like before, impossibility of achieving any partial information by $t-1$ parties should be information theoretical, not computational.

Lagrange Polynomial Interpolation: Reminder

We are given a set of t pairs $(x_1, y_1), \ldots, (x_t, y_t)$. Furthermore, we are told there is a univariate, degree $t-1$ polynomial, $f[x]$, satisfying $f(x_i) = y_i$ $(i = 1, \ldots, t)$.

How can we find this polynomial (namely find its t coefficients $f[x] = a_{t-1}x^{t-1} + \ldots + a_1 x + a_0$)?

Define

$$f_1[x] = y_1 \cdot \frac{x - x_2}{x_1 - x_2} \cdot \frac{x - x_3}{x_1 - x_3} \cdot \ldots \cdot \frac{x - x_t}{x_1 - x_t} \ .$$

Then $f_1[x]$ is a degree $t-1$ polynomial, satisfying $f(x_1) = y_1$, and for all other $i \neq 1$, $f(x_i) = 0$. (Note that all terms $x_1 - x_j$ in the denominator are non-zero.)

We can define $f_2[x], \ldots, f_t[x]$ analogously. Then the desired degree $t-1$ polynomial is

$$f[x] = f_1[x] + f_2[x] + \ldots + f_t[x] \ .$$

Shamir t-out-of-n Secret Sharing: Construction

- Let $S \in Z_p$ be the secret. We take p satisfying $p > n + 1$. If the domain of secrets is smaller, some values in Z_p will just not be used.
- The dealer choses at random $t - 1$ values $r_{t-1}, \ldots, r_1$ uniformly and independently in the domain Z_p.
- Dealer defines a degree $t - 1$ polynomial whose free term equals the secret: $f[x] = r_{t-1}x^{t-1} + \ldots + r_1 x + S$.
- Shares $s_1, s_2, \ldots, s_n$ of players $1, \ldots, n$ are values of $f[x]$ at n corresponding points,
  $s_1 = f(1), \ldots, s_{n-1} = f(n-1), s_n = f(n)$.

Shamir t-out-of-n Secret Sharing: Construction

- Dealer defines a degree $t-1$ polynomial whose free term equals the secret: $f[x] = r_{t-1}x^{t-1} + \ldots + r_1 x + S$.
- Shares $s_1, s_2, \ldots, s_n$ of players $1, \ldots, n$ are values of $f[x]$ at n corresponding points,
  $s_1 = f(1), \ldots, s_{n-1} = f(n-1), s_n = f(n)$.

- Reconstruction: Any set of t players (or more) apply Langrange interpolation formula to their shares, find the coefficients of the unique degree $t-1$ polynomial $f[x] = r_{t-1}x^{t-1} + \ldots + r_1 x + S$. The free term of this polynomial, S, is the desired secret.
- Note that no matter which t (or more) shares are used, reconstruction yields the same polynomial (and hence, secret).
- Secrecy: We saw that any set of $t-1$ players (or less) learns nothing about the secret.

Threshold Cryptography

Last time (*i.e.* lecture 10) there was some confusion in the underlying context of threshold crypto.

Today we will try to remove that confusion.

In addition, we'll describe in detail a solution to threshold El-Gamal t-out-of-n encryption.

This mixes threshold secret sharing (which provides info theoretic security) with El Gamal encryption. The result is secure only in a computational context (assuming inverting El Gamal is hard, and adversaries are limited to poly time computations).

Threshold Cryptography in Our Context

The dealer sets up and publishes the public key. It also distributes shares of the private (secret) key to the n participants. At thispoint, the dealer is no longer needed.

When Alice wishes to send an encrypted message, she encrypts "as usual" by the PKC.

Each of the n participants computes a "share of the decryption", based on its share of the secret key, together with the encrypted message.

Later, any t-out-of-n shares of the decryption can be used to reconstruct the decrypted message.

Notice that the shares of the secret key are not revealed in the process.

Elgamal PKC (Reminder)

- Public information: A large prime p, where $p-1$ has a known factorization and a large prime factor. Recommended to take $p = 2q+1$, q is prime too, and p is 756 or 1024 bits long.
  - A multiplicative generator g of Z_p^*
  - Bob publishes p, g.
  - Bob picks $a \in [0..p-2]$ at random.
  - Bob computes and publishes $\beta = g^a \pmod p$.
- Bob's private information: a.
- Encryption: of the message m:
  - Alice picks $k \in [0..p-2]$ at random.
  - Alice computes $g^k \pmod p$, $m\beta^k \pmod p$.
  - Alice sends $E(m) = (g^k, m \cdot \beta^k)$ to Bob.
    (β^k "masks" m; k obviously is not made public).
- Decryption of $(g^k, m \cdot \beta^k) = (c_1, c_2)$:
  - Bob computes ${c_1}^a = (g^k)^a = (g^a)^k = \beta^k \pmod p$.
  - This enables Bob to compute the multiplicative inverse of $\beta^k \bmod p$, β^{-k} (even though he does not know k).
  - Bob now computes $\beta^{-k} \cdot c_2 = m$. ♠

Elgamal PKC (Slightly Modified to Suit Our Needs)

- Public information: A large prime p, where $p = 2q + 1$, q is prime too.
  - A multiplicative generator g of Z_p^*
  - Bob publishes p, g.
  - Bob picks $a \in [0..q]$ at random.
  - Bob computes and publishes $\beta = g^{2a} \pmod p$ (β is a QR).
- Bob's private information: $2a \pmod{p-1}$.
- Encryption: of the message m:
  - Alice picks $k \in [0..p-2]$ at random.
  - Alice computes $g^{2k} \pmod p$, $m\beta^k \pmod p$.
  - Alice sends $E(m) = (g^{2k}, m \cdot \beta^k)$ to Bob.
    (β^k "masks" m; k obviously is not made public).
- Decryption of $(g^{2k}, m \cdot \beta^k) = (c_1, c_2)$:
  - Bob computes ${c_1}^a = (g^{2k})^a = (g^{2a})^k = \beta^k \pmod p$.
  - This enables Bob to compute the multiplicative inverse of $\beta^k \mod p$, β^{-k} (even though he does not know k).
  - Bob now computes $\beta^{-k} \cdot c_2 = m$. ♠
- Can show that the modified PKC is as strong as the original.

Threshold Cryptography: Elgamal PKC as Special Case

- Dealer publishes $p = 2q + 1$, g and public "encryption key" $\beta = g^{2a} \pmod{p}$. Dealer knows secret "decryption key" $2a$.
- Dealer distributes "shares of a" to parties so that given (c_1, c_2), an encryption of any future message m, each party can produce (on its own) a "share of a decryption".
- Later, any t of them can produce a decryption of m, but any $t - 1$ or fewer cannot.
- Notice that simply doing secret sharing for $a \pmod{p-1}$ does not work. (why?)
- An n-out-of-n solution is similar to the RSA signature case done last week (easy).
- Today, we'll describe the t-out-of-n solution.
- A crucial observation is the fact that the secret, $f(0)$, is a linear combination of the shares (t values, $f(i)$'s).

Threshold Cryptography: El Gamal PKC as Special Case

- The "shares of a" are simply shares in Shamir's t-out-of-n scheme, where the secret is a. Denote these shares by $a_1, a_2, \ldots, a_n$ (one per participant).
- First, let us describe what the "share of a decryption" (c_1, c_2) is going to be.
- In the regular (one recipient) case, the critical step in decryption was computing ${c_1}^a = (g^{2k})^a = (g^{2a})^k = \beta^k \pmod p$.
- Once we get $\beta^k \pmod p$, it is easy to extract m.
- Define "i-th share of a decryption" for (c_1, c_2) as $c_1^{a_i} \pmod p$. This equals $c_1^{a_i \pmod{p-1}} = (g^{2k})^{a_i \pmod{p-1}}$.
- The natural thing to do is "interpolate in the exponent", but $p-1$ is not a prime number.
- So over which finite field are we going to work?

Threshold El Gamal PKC: Where Do We Interpolate?

- So over which finite field are we going to work?

- We probably want to interpolate in the exponent. But
- Recall that $p = 2q + 1$, so $p - 1 = 2q$.
- $p - 1 = 2q$ is not prime.
- This is why we work with even powers of g.
- The quadratic residues in Z_p^* constitue a multiplicative group of order q, a prime.
- Since its order, q, is a prime, this group is cyclic.

Threshold El Gamal PKC: Construction

- The secret and the shares will be taken from the field Z_q.
- Given the secret a, the dealer picks $t-1$ values $r_{t-1}, \ldots, r_1$ from Z_q independently at random, and defines the polynomial $f[x] = r_{t-1}x^{t-1} + \ldots + r_1 x + a$ over Z_q.
- The n "shares of a" are simply $a_1 = f(1)$ through $a_n = f(n)$, where n is the number of participants. (Note: a is just an element in Z_q, not necessarily a QR.)
- This ends the role of the dealer. From now on participants are on their own.
- Given an encryption (c_1, c_2) of m, each party, i, computes $c_1^{a_i} \pmod p$. We call these "shares of m's encryption".
- These shares are kept secret until t parties decide to decrypt.
- The n shares $a_1, \ldots, a_n$ are going to be used just for such exponentiation.
- To recover m, t parties cooperate by working on their shares of m's encryption.

Threshold El Gamal PKC: Interpolation and Decryption

- Suppose, without loss of generality, that these are parties $1, \ldots, t$ that wish to decrypt c_1, c_2 and recover m.
- As pointed out earlier, the value $f(0) = a$ can be expressed as a linear combination of the shares $f(1) = a_1, \ldots, f(1) = a_1$.
- This means that we can express $a = \sum_{i=1}^{t} b_i a_i \pmod{q}$, where the $b_i \in Z_q$ are constants (not dependent upon the $f(i)$).
- We want to construct $c_1^a \pmod{p}$, based on $c_1^{a_1}, \ldots, c_1^{a_t} \pmod{p}$.
- The natural thing to do seems to compute

$$(c_1^{a_1})^{b_1} \cdot \ldots \cdot (c_1^{a_t})^{b_t} = c_1^{\sum_{i=1}^{t} b_i a_i} \pmod{p}$$

- Which, in turn, should equal c_1^{2a}.

Houston, We Have a Problem

- The natural thing to do seems to compute

$$(c_1^{a_1})^{b_1} \cdot \ldots \cdot (c_1^{a_t})^{b_t} = c_1^{\sum_{i=1}^{t} b_i a_i} \pmod{p}$$

- Which, in turn, should equal c_1^a.

- But $a = \sum_{i=1}^{t} b_i a_i \pmod{q}$, and not $\pmod{p-1}$. So it is not clear if we have equality at all.

In Fact, We do not Have a Problem

- In our modification, Bob computes and publishes $\beta = g^{2a} \pmod{p}$.
- Alice computes and sends $c_1 = g^{2k}$.
- Each party, i, computes $c_1^{a_i} \pmod{p}$, which we termed "shares of m's encryption".
- Since c_1 is a quadratic residue in Z_p, all its powers, including these shares of the encryption, are also quadratic residue in Z_p.
- The order of the quadratic residues group in Z_p is q, not $2q = p - 1$.
- Therefore, if two powers are equal modulo q, and we take any element from this group and raise it to these two powers, the outcome will be the same.

Wrapping Things Up

- Suppose $a = \sum_{i=1}^{t} b_i a_i \pmod{q}$.
- Then either
  $\sum_{i=1}^{t} b_i a_i = a \pmod{2q}$ or
  $\sum_{i=1}^{t} b_i a_i = a + q \pmod{2q}$.
- In both cases, when multiplying by 2, we get
  $2\sum_{i=1}^{t} b_i a_i = 2a \pmod{2q}$.
- Recall that the i-th share of a decryption was $c_1^{a_i} \pmod{p}$. This equals $c_1^{a_i \pmod{p-1}} = (g^{2k})^{a_i \pmod{p-1}}$.
- Thus $c_1^{\sum_{i=1}^{t} b_i a_i} = \left(g^k\right)^{\left(2\sum_{i=1}^{t} b_i a_i\right)} = \left(g^k\right)^{2a} \pmod{p}$
- So specifically, $a = \sum_{i=1}^{t} b_i a_i \pmod{q}$ does imply
  $c_1^{\sum_{i=1}^{t} b_i a_i} = \beta^k \pmod{p}$.
- $c_1^{\sum_{i=1}^{t} b_i a_i} \pmod{p}$ can be computed by t participants (different coefficients for different subsets) from their shares, as desired. ♠