Introduction to Modern Cryptography
Lecture/Recitation 9[2]

December 12, 2013

Instructor: Benny Chor

Teaching Assistant: Nir Bitansky

School of Computer Science
Tel-Aviv University

Fall Semester, 2013–14, 15:00–18:00
Dan David 201

Course site: http://tau-crypto-f13.wikidot.com/

[2]some slides courtesy of Benny Appelbaum

Lecture 8: Reminder

- Random self reducibility of RSA and QR/QNR: Code.

- Digital Signatures.
- Diffie Hellman signatures paradigm and its shortcomings.
- The hash and decrypt signature paradigm.
- Elgamal probabilistic signature, based on discrete logarithm.
- Other signatures paradigm: Sketch.

- Integer factoring algorithms: Pollard's rho.

Lecture/Recitation 9 Plan: Secret sharing

Let us start with a well known motivation: the dying pirate:

Lecture/Recitation 9 Plan: Secret sharing

Let us start with a well known motivation: the dying pirate:

An old pirate is going to die.
He wants to reveal the coordinates of the secret treasure to his three pirate sons so that they can share the treasure.
However, if he tells them the coordinates then only one of them (the one who gets to the treasure first) will get the treasure.

What should he do?

Motivating Example I: Splitting Trust

- Suppose that we have a secret key for some sensitive operation

Is there a way to split the secret among the participants?

Motivating Example I: Splitting Trust

- Suppose that we have a secret key for some sensitive operation
  - Missile launch codes

Is there a way to split the secret among the participants?

Motivating Example I: Splitting Trust

- Suppose that we have a secret key for some sensitive operation
  - Missile launch codes
  - Codeword to a secret numbered bank account

Is there a way to split the secret among the participants?

Motivating Example I: Splitting Trust

- Suppose that we have a secret key for some sensitive operation
  - Missile launch codes
  - Codeword to a secret numbered bank account
- We do not want to put all responsibility on one person

Is there a way to split the secret among the participants?

Motivating Example I: Splitting Trust

- Suppose that we have a secret key for some sensitive operation
  - Missile launch codes
  - Codeword to a secret numbered bank account
- We do not want to put all responsibility on one person
- Idea: split this secret among a group of people

Is there a way to split the secret among the participants?

Motivating Example I: Splitting Trust

- Suppose that we have a secret key for some sensitive operation
  - Missile launch codes
  - Codeword to a secret numbered bank account
- We do not want to put all responsibility on one person
- Idea: split this secret among a group of people
- Only together they will be able to recover the secret info

Is there a way to split the secret among the participants?

Motivating Example I: Splitting Trust

- Suppose that we have a secret key for some sensitive operation
  - Missile launch codes
  - Codeword to a secret numbered bank account
- We do not want to put all responsibility on one person
- Idea: split this secret among a group of people
- Only together they will be able to recover the secret info
  - At least two out of President, VP, Minister of defense should agree on launching a missile

Is there a way to split the secret among the participants?

Motivating Example I: Splitting Trust

- Suppose that we have a secret key for some sensitive operation
  - Missile launch codes
  - Codeword to a secret numbered bank account
- We do not want to put all responsibility on one person
- Idea: split this secret among a group of people
- Only together they will be able to recover the secret info
  - At least two out of President, VP, Minister of defense should agree on launching a missile
  - All board members should collaborate to access the account

Is there a way to split the secret among the participants?

Motivating Example II: Key protection

- In cryptography, security crucially depend on the adversary not knowing the secret key

Is there a way to split the secret among several servers ?

Motivating Example II: Key protection

- In cryptography, security crucially depend on the adversary not knowing the secret key
- But the key is stored somewhere (e.g., your laptop)

Is there a way to split the secret among several servers ?

Motivating Example II: Key protection

- In cryptography, security crucially depend on the adversary not knowing the secret key
- But the key is stored somewhere (e.g., your laptop)
- What if the adversary somehow breaks into the machine on which the key is stored?

Is there a way to split the secret among several servers ?

Motivating Example II: Key protection

- In cryptography, security crucially depend on the adversary not knowing the secret key
- But the key is stored somewhere (e.g., your laptop)
- What if the adversary somehow breaks into the machine on which the key is stored?
- This is especially crucial for long term keys (e.g., signature keys), that may be used for many years.

Is there a way to split the secret among several servers ?

Secret Sharing (Shamir 79')

We would like to split a secret s into n pieces (called shares), such that for $t \leq n$ (e.g., $t = n$):

- If an adversary has only $t - 1$ out of the n shares, then he has absolutely no information about the secret s.

Secret Sharing (Shamir 79')

We would like to split a secret s into n pieces (called shares), such that for $t \leq n$ (e.g., $t = n$):

- If an adversary has only $t - 1$ out of the n shares, then he has absolutely no information about the secret s.
- t shares suffice to fully reconstruct the secret s.

Naive Attempt

- Say that s is a 128-bit key of AES, and we have 4 parties.

Naive Attempt

- Say that s is a 128-bit key of AES, and we have 4 parties.
- Let each party get 32 bits of the key.

Naive Attempt

- Say that s is a 128-bit key of AES, and we have 4 parties.
- Let each party get 32 bits of the key.

Naive Attempt

- Say that s is a 128-bit key of AES, and we have 4 parties.
- Let each party get 32 bits of the key.

Is this a good idea?
What do 3 parties know about the secret?

Better Idea

- To share s among 4 players: choose at random 3 values $r_1, r_2, r_3 \xleftarrow{R} \{0,1\}^{128}$.
- The shares s_1, s_2, s_3 of players $1, 2, 3$ are r_1, r_2, r_3, correspondingly.
- Share s_4 of player 4 is $s - (r_1 + r_2 + r_3) \mod 2^{128}$.

Better Idea

- To share s among 4 players: choose at random 3 values $r_1, r_2, r_3 \xleftarrow{R} \{0,1\}^{128}$.
- The shares s_1, s_2, s_3 of players $1, 2, 3$ are r_1, r_2, r_3, correspondingly.
- Share s_4 of player 4 is $s - (r_1 + r_2 + r_3) \mod 2^{128}$.

Q1: What do 3 parties know about the secret?

Better Idea

- To share s among 4 players: choose at random 3 values $r_1, r_2, r_3 \xleftarrow{R} \{0,1\}^{128}$.
- The shares s_1, s_2, s_3 of players $1, 2, 3$ are r_1, r_2, r_3, correspondingly.
- Share s_4 of player 4 is $s - (r_1 + r_2 + r_3) \mod 2^{128}$.

Q1: What do 3 parties know about the secret?
Q2: Can the parties (altogether) reconstruct the secret?

Better Idea

- To share s among 4 players: choose at random 3 values $r_1, r_2, r_3 \xleftarrow{R} \{0,1\}^{128}$.
- The shares s_1, s_2, s_3 of players $1, 2, 3$ are r_1, r_2, r_3, correspondingly.
- Share s_4 of player 4 is $s - (r_1 + r_2 + r_3) \mod 2^{128}$.

Q1: What do 3 parties know about the secret?
Q2: Can the parties (altogether) reconstruct the secret?
A2: Reconstruction is easy:

Better Idea

- To share s among 4 players: choose at random 3 values $r_1, r_2, r_3 \xleftarrow{R} \{0,1\}^{128}$.
- The shares s_1, s_2, s_3 of players $1, 2, 3$ are r_1, r_2, r_3, correspondingly.
- Share s_4 of player 4 is $s - (r_1 + r_2 + r_3) \mod 2^{128}$.

Q1: What do 3 parties know about the secret?
Q2: Can the parties (altogether) reconstruct the secret?
A2: Reconstruction is easy: $s_1 + s_2 + s_3 + s_4 = \sum r_i + s - \sum r_i = s$.

Better Idea

- To share s among 4 players: choose at random 3 random values $r_1, r_2, r_3 \xleftarrow{R} \{0,1\}^{128}$ uniformly and independently.
- The shares s_1, s_2, s_3 of players $1, 2, 3$ are r_1, r_2, r_3, correspondingly.
- Share s_4 of player 4 is $s - (r_1 + r_2 + r_3) \mod 2^{128}$.

Better Idea

- To share s among 4 players: choose at random 3 random values $r_1, r_2, r_3 \xleftarrow{R} \{0,1\}^{128}$ uniformly and independently.
- The shares s_1, s_2, s_3 of players $1, 2, 3$ are r_1, r_2, r_3, correspondingly.
- Share s_4 of player 4 is $s - (r_1 + r_2 + r_3) \mod 2^{128}$.

Q1: What do 3 parties know about the secret?

Better Idea

- To share s among 4 players: choose at random 3 random values $r_1, r_2, r_3 \xleftarrow{R} \{0,1\}^{128}$ uniformly and independently.
- The shares s_1, s_2, s_3 of players $1, 2, 3$ are r_1, r_2, r_3, correspondingly.
- Share s_4 of player 4 is $s - (r_1 + r_2 + r_3) \mod 2^{128}$.

Q1: What do 3 parties know about the secret?

- Clearly, the first 3 parties see just random elements.

Better Idea

- To share s among 4 players: choose at random 3 random values $r_1, r_2, r_3 \xleftarrow{R} \{0,1\}^{128}$ uniformly and independently.
- The shares s_1, s_2, s_3 of players $1, 2, 3$ are r_1, r_2, r_3, correspondingly.
- Share s_4 of player 4 is $s - (r_1 + r_2 + r_3) \mod 2^{128}$.

Q1: What do 3 parties know about the secret?

- Clearly, the first 3 parties see just random elements.
- So they learn nothing on the secret.

Better Idea

- To share s among 4 players: choose at random 3 random values $r_1, r_2, r_3 \xleftarrow{R} \{0,1\}^{128}$ uniformly and independently.
- The shares s_1, s_2, s_3 of players $1, 2, 3$ are r_1, r_2, r_3, correspondingly.
- Share s_4 of player 4 is $s - (r_1 + r_2 + r_3) \mod 2^{128}$.

Q1: What do 3 parties know about the secret?

- Clearly, the first 3 parties see just random elements.
- So they learn nothing on the secret.
- Is this true for every coalition of 3 parties?

Security

Claim: For every secret s, and every set of 3 players, the random variables $s_{i_1}, s_{i_2}, s_{i_3}$ are uniformly distributed.

Proof:

- Will demonstrate for the coalition $\{2, 3, 4\}$.

Security

Claim: For every secret s, and every set of 3 players, the random variables $s_{i_1}, s_{i_2}, s_{i_3}$ are uniformly distributed.

Proof:

- Will demonstrate for the coalition $\{2, 3, 4\}$.
- Fix some secret a.

Security

Claim: For every secret s, and every set of 3 players, the random variables $s_{i_1}, s_{i_2}, s_{i_3}$ are uniformly distributed.

Proof:

- Will demonstrate for the coalition $\{2, 3, 4\}$.
- Fix some secret a.
- Fix c_2, c_3, c_4 to be some 3 values in Z_m.

Security

Claim: For every secret s, and every set of 3 players, the random variables $s_{i_1}, s_{i_2}, s_{i_3}$ are uniformly distributed.

Proof:

- Will demonstrate for the coalition $\{2, 3, 4\}$.
- Fix some secret a.
- Fix c_2, c_3, c_4 to be some 3 values in Z_m.
- It suffices to show that the shares (s_2, s_3, s_4) takes the value (c_2, c_3, c_4) with probability exactly $1/m^3$, where $m = 2^{128}$. (Why?)

Security

Claim: For every secret s, and every set of 3 players, the random variables $s_{i_1}, s_{i_2}, s_{i_3}$ are uniformly distributed.

Proof:

- Will demonstrate for the coalition $\{2, 3, 4\}$.
- Fix some secret a.
- Fix c_2, c_3, c_4 to be some 3 values in Z_m.
- It suffices to show that the shares (s_2, s_3, s_4) takes the value (c_2, c_3, c_4) with probability exactly $1/m^3$, where $m = 2^{128}$. (Why?)
- Claim: There is exactly one choice of r_1, r_2, r_3 that yields these 3 shares. Specifically, $r_2 = c_2, r_3 = c_3$, and $r_1 = a - (c_2 + c_3 + c_4) \mod m$.

Security

Claim: For every secret s, and every set of 3 players, the random variables $s_{i_1}, s_{i_2}, s_{i_3}$ are uniformly distributed.

Proof:

- Will demonstrate for the coalition $\{2, 3, 4\}$.
- Fix some secret a.
- Fix c_2, c_3, c_4 to be some 3 values in Z_m.
- It suffices to show that the shares (s_2, s_3, s_4) takes the value (c_2, c_3, c_4) with probability exactly $1/m^3$, where $m = 2^{128}$. (Why?)
- Claim: There is exactly one choice of r_1, r_2, r_3 that yields these 3 shares. Specifically, $r_2 = c_2, r_3 = c_3$, and $r_1 = a - (c_2 + c_3 + c_4) \mod m$.

• Since r_1, r_2, r_3 are chosen uniformly and independently in Z_m, the probability of getting these 3 shares, given that the secret equals a, is exactly $1/m^{n-1}$. ♠

n-out-of-n Secret Sharing: Setting

- 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 all n parties to reconstruct S, but any subset of $n-1$ parties (or less) cannot get any partial information on S.
- Computational requirements: Reconstruction should be efficient (polynomial in the length of S and of number of parties n).
- Want impossibility of obtaining any partial information by any subset with $n-1$ parties (or less) should be information theoretical – not based on any complexity assumptions.

Formal Definition: n-out-of-n Secret Sharing

- The trusted dealer holds $S \in \mathcal{U}$, picks a random value r by some distribution over a finite space, and generates the shares $s_1, s_2, \ldots, s_n$ by applying a known function $F(S, r) = (s_1, s_2, \ldots, s_n)$.
- Note: F is deterministic. All randomness comes from r.
- Party i receives share s_i.
- Reconstruction: There is a reconstruction function H such that applying it to the n shares always gives back the secret. Formally, for all S, r, $H(F(S, r)) = S$.

Formal Definition (cont.): n-out-of-n Secret Sharing

- Secrecy against coalitions: For any two possible values a, b of the secret $S \in \mathcal{U}$, and for any subset (coalition) of $n-1$ players,

- The distribution of the shares of the coalition members, given that $S = a$ is exactly equal to the distribution of the shares of the coalition members, given that $S = b$.

n-out-of-n Secret Sharing: Construction

- Denote the size of the secrets universe, $|U|$, by m.
- Without loss of generality, the possible values of S are a subset of $Z_m = \{0, 1, \ldots, m-1\}$.
- The dealer choses at random $n-1$ values $r_1, \ldots, r_{n-1}$ uniformly and independently.
- Shares $s_1, s_2, \ldots, s_{n-1}$ of players $1, \ldots, n-1$ are $r_1, \ldots, r_{n-1}$, correspondingly.
- Share s_n of player n is $S - \left(\sum_{i=1}^{n-1} r_i\right) \mod m$.

n-out-of-n Secret Sharing: Validity

Claim: This is a valid n-out-of-n secret sharing scheme.

- Reconstruction by n players is easy:
  Simply express $S = \sum_{i=1}^{n} s_i \mod m$.
- Secrecy against coalitions: Will demonstrate for the coalition $\{2, \ldots, n-1, n\}$. Suppose the value of the secret is a. Let $c_2, \ldots, c_{n-1}, c_n$ be any $n-1$ values in Z_m.
- Claim: There is exactly one choice of $r_1, \ldots, r_{n-1}$ that yields these $n-1$ shares. Specifically, $r_2 = c_2, \ldots, r_{n-1} = c_{n-1}$, and $r_n = a - \left(\sum_{i=1}^{n-1} c_i\right) \mod m$.
- Since $r_1, \ldots, r_{n-1}$ are chosen uniformly and independently in Z_m, the probability of getting these $n-1$ shares, given the that the secret equals a, is exactly $1/m^{n-1}$.
- Obviously this probability (but not the values of the random r_i) is exactly the same if the secret equals b. ♠

t-out-of-n Secret Sharing: Setting

- 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, we want impossibility of achieving any partial information by $t-1$ parties to be information theoretical, not computational.

t-out-of-n Secret Sharing

A simple solution is to construct $\binom{n}{k}$ independent k-out-of-k secret sharing scheme, and share the secret S in each.

This works, but has a large overhead, which becomes exponential in n as k grows (think of $k = n/2$).

A far more efficient scheme, based on polynomial interpolation, was given by Adi Shamir.

Shamir's t-out-of-n Secret Sharing: Intuition

Preliminaries: Without loss of generality, the size of the secrets's universe satisfies $n+1 \leq |U|$, and furthermore $U = Z_p$ for some prime number p (we can always extend U, and simply not use some of the values in Z_p).

Intuition: Suppose we have a degree one univariate polynomial $f[x] = ax + b$ over Z_p, and we give each participant i ($i = 1, \ldots, n$) the value $f(i)$.

Question 1: What does a single participant, i, learn about $f(0) = b$?

Answer 1: Nothing!

Question 2: What can two participants, i, j ($i \neq j$) learn about $f(0) = b$?

Answer 2: Everything! Having $f(i) = ai + b$ and $f(j) = aj + b$, they can solve two linear equations in two variables (a and b) and recover both a, b. In fact what they do is polynomial interpolation, in this case of a degree one polynomial.

A (Slight) Detour: Lagrange Polynomial Interpolation

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

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

Lagrange Polynomial Interpolation: Uniqueness

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

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

Note that there is a unique $f[x]$ with these properties. For suppose $g[x]$ also satisfies these properties. Then $f[x] - g[x]$ is a degree $t-1$ polynomial with t different roots. So it must be the zero polynomial, thus $f[x] = g[x]$.

Lagrange Polynomial Interpolation in Sage

Good old Sage naturally supports Lagrange interpolation. We just got to be a bit careful so it understands the numbers we input are Z_p elements, rather than integers (which are the default).

```
F = GF(19)
R = PolynomialRing(F, 'x')
g=R.lagrange_polynomial([(F(0),F(4)),(F(2),F(12)),(F(6),F(6))])
       # F(b) is the conversion of integer b to a GF(19) element
       # so g(0)=4, g(2)=12, g(6)=6
g
```

```
>     7*x^2 + 9*x + 4
```

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

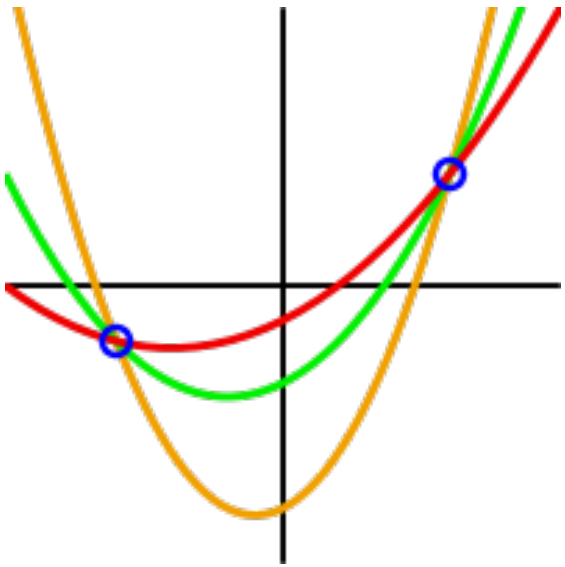
- Dealer defines a degree $t-1$ polynomial whose free term equals the secret: $f[x] = r_{t-1}x^{t-1} + \ldots + r_1 x + S$.
- Shares $s_1, s_2, \ldots, s_n$ of players $1, \ldots, n$ are values of $f[x]$ at n corresponding points,
  $s_1 = f(1), \ldots, s_{n-1} = f(n-1), s_n = f(n)$.

- 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).

Shamir n-out-of-n Secret Sharing: Secrecy

- 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,
  $s_1 = f(1), \ldots, s_{n-1} = f(n-1), s_n = f(n)$.

- Secrecy: Should show that any set of $t-1$ players (or less) learns nothing about the secret.



(image from Wikipedia)

t-out-of-n Secret Sharing: Proof of Secrecy

Claim: This is a valid t-out-of-n secret sharing scheme.

- Secrecy against coalitions: Will demonstrate for the coalition $\{1, \ldots, t-1\}$ (but all others are the same). Suppose the value of the secret is a. Let $c_1, \ldots, c_{t-1}$ be any $t-1$ values in Z_p.
- Claim: There is exactly one choice of $r_1, \ldots, r_{t-1}$ that yields these $t-1$ shares. The $t-1$ shares $s_1 = f(1), \ldots, s_{t-1} = f(t-1)$ plus the secret a uniquely determine a a degree $t-1$ polynomial $f[x] = r_{t-1}x^{t-1} + \ldots + r_1 x + a$.
- Thus is the secret is a, the probability of getting the shares $s_1 = f(1), \ldots, s_{t-1} = f(t-1)$ is exactly $1/p^{t-1}$.
- Likewise, if is the secret is b, the probability of getting the shares $s_1 = f(1), \ldots, s_{t-1} = f(t-1)$ is exactly $1/p^{t-1}$.
- So the shares are distributed exactly the same given that the secret equals is a or is b. ♠

t-out-of-n Secret Sharing: Some Remarks

- For the scheme to work, we need a finite field with at least $n+1$ elements (where n is the number of parties). We worked in Z_p, but might as well work inside $GF(p^k)$, including the case where $p = 2$, since Lagrange interpolation is applicable to any field.
- For the case where the domain of the secrets is larger than n, the size of secrets is the same as size of each share. Such scheme is termed ideal secret sharing scheme.
- The set of minimal subsets that can reconstruct the secret is termed the access structure of the scheme. We dealt with threshold access structure, but there are efficient schemes for other access structures as well.
- Most access structures are not known to possess efficient secret sharing schemes. Inefficient ones (large size of shares) are easy to come by. No "substantial" (exponential) lower bounds on shares sizes were shown so far.

One More Remark: Why prime?

- For the interpolation to work, we had to assume that p is prime (there are no "Zero divisors")

One More Remark: Why prime?

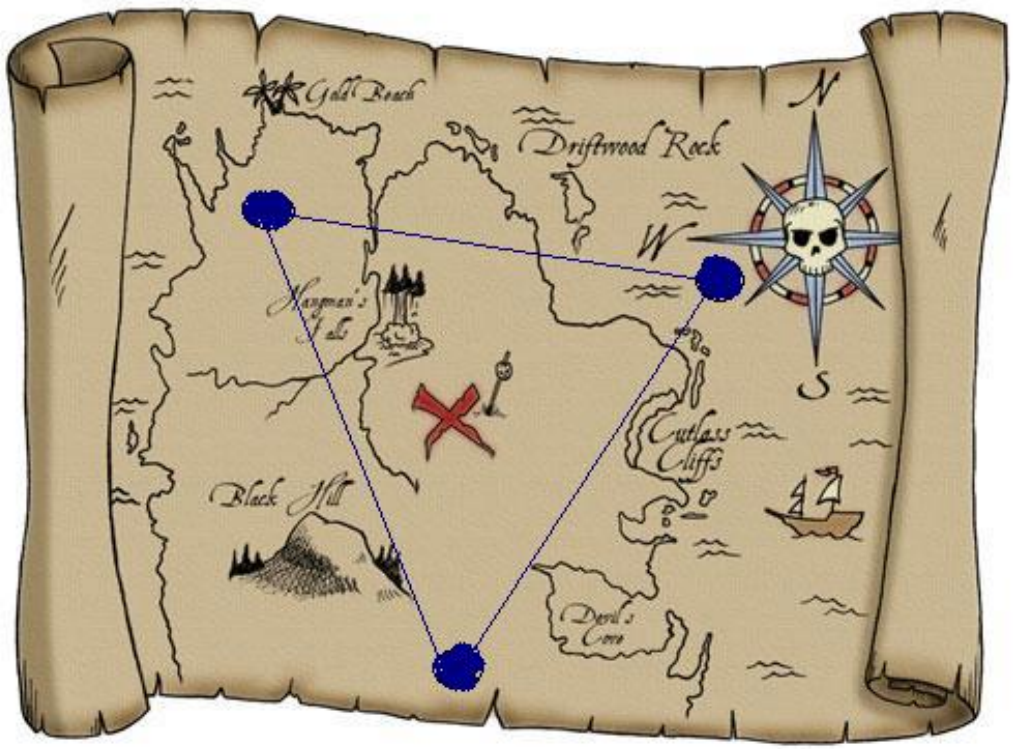
- For the interpolation to work, we had to assume that p is prime (there are no "Zero divisors")
- In general, we can work over any finite field as long as it has $n + 1$ elements (where n is the number of parties).

One More Remark: Why prime?

- For the interpolation to work, we had to assume that p is prime (there are no "Zero divisors")
- In general, we can work over any finite field as long as it has $n+1$ elements (where n is the number of parties).
- For example, we can work over an extension field $GF(p^k)$, including the case where $p = 2$, since Lagrange interpolation is applicable to any field.

The Dying Pirate: Solution

The pirate draws 3 random points whose center (as a triangle) hits the location of the treasure. Each son gets the coordinates of a single point.
The info of a single point, or even a pair of points, reveals no information about the true location.



Application 1: PGP key recovery mechanism

- Pretty Good Privacy[3] is a program to provide encrypted email, employing public key cryptography.

[3] also termed Pretty Good Piracy in the early days of commercial public key encryption

Application 1: PGP key recovery mechanism

- Pretty Good Privacy[3] is a program to provide encrypted email, employing public key cryptography.
- It is risky to store the key on the PC of the user

[3] also termed Pretty Good Piracy in the early days of commercial public key encryption

Application 1: PGP key recovery mechanism

- Pretty Good Privacy[3] is a program to provide encrypted email, employing public key cryptography.
- It is risky to store the key on the PC of the user
- But the user cannot remember the key k

[3] also termed Pretty Good Piracy in the early days of commercial public key encryption

Application 1: PGP key recovery mechanism

- Pretty Good Privacy[3] is a program to provide encrypted email, employing public key cryptography.
- It is risky to store the key on the PC of the user
- But the user cannot remember the key k
- So users remember a very long password p and store only $H(p) \oplus k$ where H is a hash function

[3] also termed Pretty Good Piracy in the early days of commercial public key encryption

Application 1: PGP key recovery mechanism

- Pretty Good Privacy[3] is a program to provide encrypted email, employing public key cryptography.
- It is risky to store the key on the PC of the user
- But the user cannot remember the key k
- So users remember a very long password p and store only $H(p) \oplus k$ where H is a hash function
- What if the user forgets the password?

[3] also termed Pretty Good Piracy in the early days of commercial public key encryption

Application 1: PGP key recovery mechanism

Solution of PGP: Secret Sharing

- The key is shared in a 3-out-of-5 scheme to 5 shares $s_1, \ldots, s_5$.

Application 1: PGP key recovery mechanism

Solution of PGP: Secret Sharing

- The key is shared in a 3-out-of-5 scheme to 5 shares $s_1, \ldots, s_5$.
- The user selects 5 personal questions to which he knows the answers $a_1, \ldots, a_5$.

Application 1: PGP key recovery mechanism

Solution of PGP: Secret Sharing

- The key is shared in a 3-out-of-5 scheme to 5 shares $s_1, \ldots, s_5$.
- The user selects 5 personal questions to which he knows the answers $a_1, \ldots, a_5$.
- The information $H(a_1) \oplus s_1$, $\ldots$, $H(a_5) \oplus s_5$ is stored in the computer (or actually, on a special purpose key reconstruction server)

Application 1: PGP key recovery mechanism

Solution of PGP: Secret Sharing

- The key is shared in a 3-out-of-5 scheme to 5 shares $s_1, \ldots, s_5$.
- The user selects 5 personal questions to which he knows the answers $a_1, \ldots, a_5$.
- The information $H(a_1) \oplus s_1$, $\ldots$, $H(a_5) \oplus s_5$ is stored in the computer (or actually, on a special purpose key reconstruction server)
- If the user remembers the answers to at least 3 of the questions, he can reconstruct the key.