Practical Aspects of Modern Cryptography

Winter 2011

Josh Benaloh

Brian LaMacchia

Fun with Public-Key

Tonight we'll …

- Introduce some basic tools of public-key crypto
- Combine the tools to create more powerful tools
- Lay the ground work for substantial applications

Challenge-Response Protocols

Challenge-Response Protocols

One party often wants to convince another party that something is true …

Challenge-Response Protocols

One party often wants to convince another party that something is true …

… *without* giving everything away.

Proof of Knowledge

"I know the secret key k."

PoK: Method 1

PoK: Method 1

Here is k.

PoK: Method 2

PoK: Method 2

Here is a nonce c.

←

PoK: Method 2

Here is a nonce c.

Here is the hash $h(c, k)$.

Traditional Proofs

Traditional Proofs

I want to convince you that something is true.

Traditional Proofs

I want to convince you that something is true.

I write down a proof and give it to you.

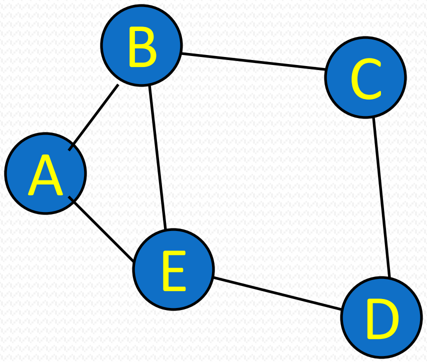
Interactive Proofs

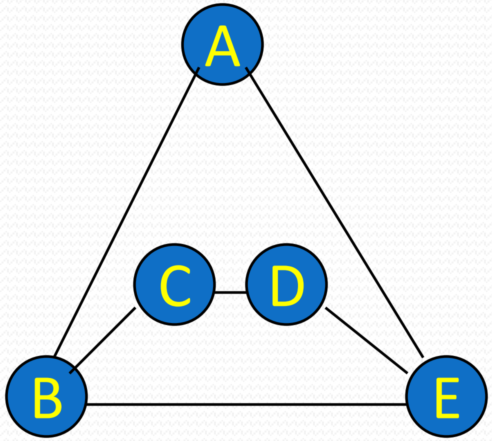
We engage in a dialogue at the conclusion of which you are convinced that my claim is true.

Graph Isomorphism

Graph Isomorphism

Graph Isomorphism





IP of Graph Isomorphism

G_1 G_2

IP of Graph Isomorphism

Generate, say, 100 additional graphs isomorphic to G_1 (and therefore also isomorphic to G_2).

IP of Graph Isomorphism

H_1

H_2

H_3

G_1 G_2

H_{100}

IP of Graph Isomorphism

IP of Graph Isomorphism

Accept a single bit challenge “L/R” for each of the 100 additional graphs.

IP of Graph Isomorphism

Accept a single bit challenge “L/R” for each of the 100 additional graphs.

Display the indicated isomorphism for each of the additional graphs.

IP of Graph Isomorphism

H_1

H_2

H_3

G_1 G_2

H_{100}

IP of Graph Isomorphism

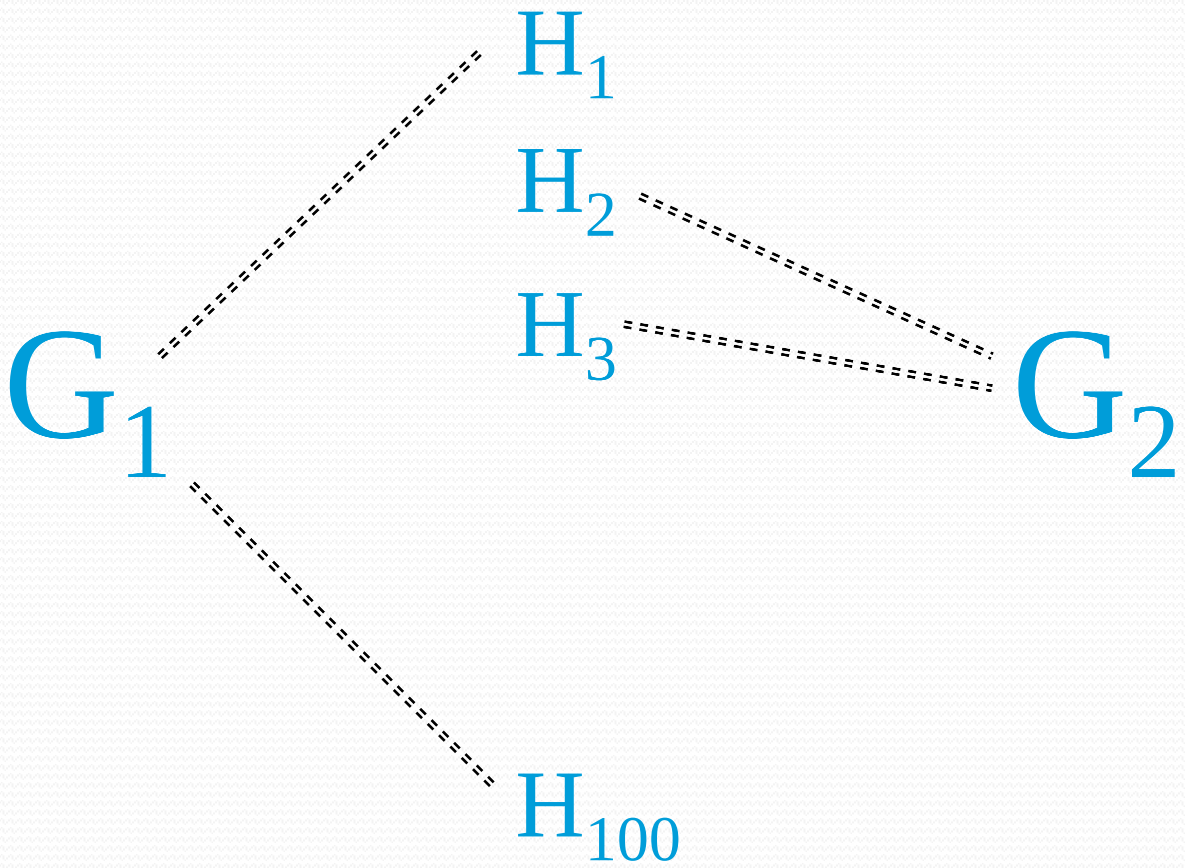
L H_1

H_2 R

G_1 H_3 R G_2

L H_{100}

IP of Graph Isomorphism



IP of Graph Isomorphism

IP of Graph Isomorphism

If graphs G_1 and G_2 were *not* isomorphic, then the “prover” would not be able to show any additional graph to be isomorphic to *both* G_1 and G_2.

IP of Graph Isomorphism

If graphs G_1 and G_2 were *not* isomorphic, then the "prover" would not be able to show any additional graph to be isomorphic to *both* G_1 and G_2.

A successful false proof would require the prover to guess all 100 challenges in advance: probability 1 in 2^{100}.

Fiat-Shamir Heuristic

Fiat-Shamir Heuristic

Instead of challenge bits being externally generated, they can be produced by applying a one-way hash function to the full set of additional graphs.

Fiat-Shamir Heuristic

Instead of challenge bits being externally generated, they can be produced by applying a one-way hash function to the full set of additional graphs.

This allows an interactive proof to be "published" without need for interaction.

IP of Graph Non-Isomorphism

IP of Graph Non-Isomorphism

G_1 G_2

IP of Graph Non-Isomorphism

A verifier can generate 100 additional graphs, each isomorphic to one of G_1 and G_2 , and present them to the prover.

IP of Graph Non-Isomorphism

A verifier can generate 100 additional graphs, each isomorphic to one of G_1 and G_2 , and present them to the prover.

The prover can then demonstrate that the graphs are not isomorphic by identifying which of G_1 and G_2 each additional graph is isomorphic to.

IP of Graph Non-Isomorphism

G_1 G_2

IP of Graph Non-Isomorphism

H_1

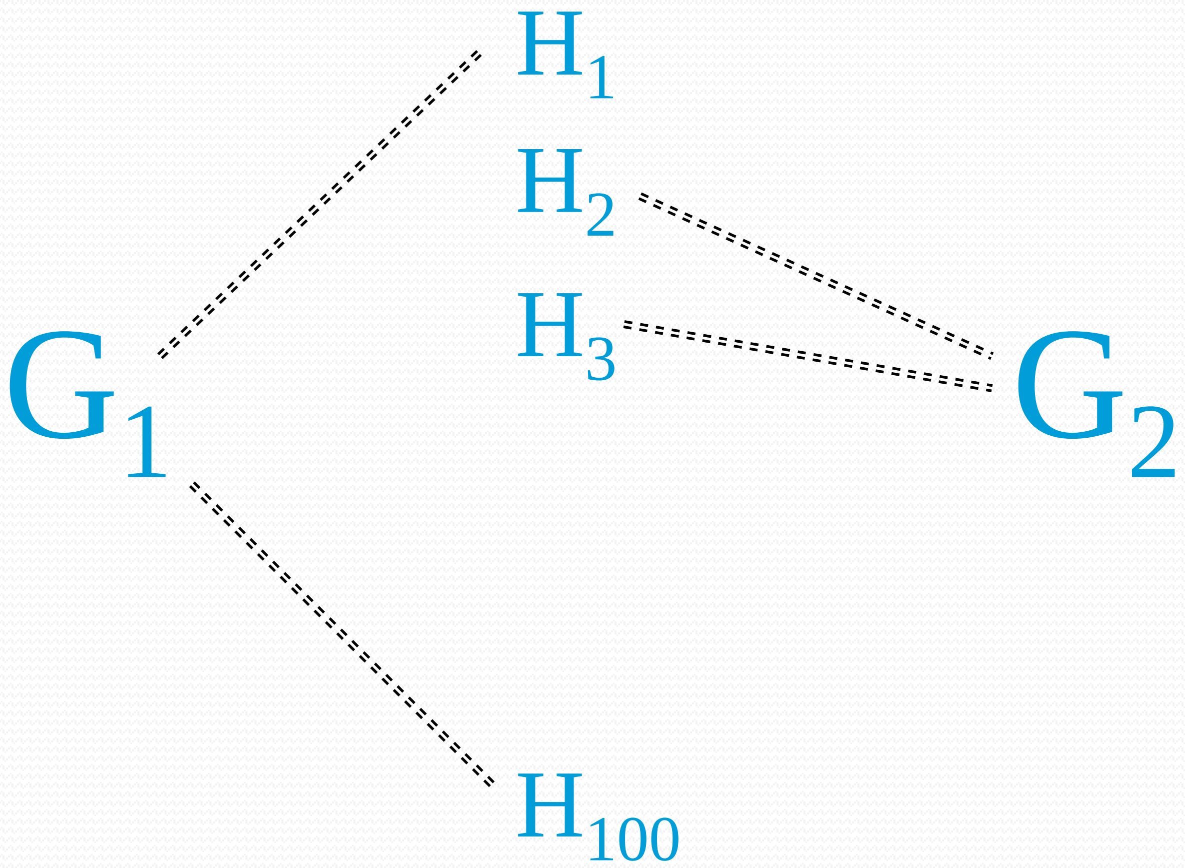
H_2

H_3

G_1 G_2

H_{100}

IP of Graph Non-Isomorphism



Proving Something is a Square

Proving Something is a Square

Suppose I want to convince you that Y is a square modulo N.

[There exists an X such that $Y = X^2 \mod N$.]

Proving Something is a Square

Suppose I want to convince you that Y is a square modulo N.

[There exists an X such that $Y = X^2 \mod N$.]

First approach: I give you X.

An Interactive Proof

$$Y$$

$$Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad Y_5 \quad \cdots\cdots\cdots\cdots\cdots \quad Y_{100}$$

An Interactive Proof

Y

$Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad Y_5 \quad \cdots\cdots\cdots\cdots\cdots \quad Y_{100}$

$0 \quad 1 \quad 0 \quad 0 \quad 1 \quad \cdots\cdots\cdots\cdots\cdots \quad 1$

An Interactive Proof

Y

Y_1 Y_2 Y_3 Y_4 Y_5 ……………… Y_{100}

0 1 0 0 1 ……………… 1

$\sqrt{Y_1}$ $\sqrt{Y_3}$ $\sqrt{Y_4}$

An Interactive Proof

Y

$Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad Y_5 \quad \cdots\cdots\cdots \quad Y_{100}$

$0 \quad 1 \quad 0 \quad 0 \quad 1 \quad \cdots\cdots\cdots \quad 1$

$\sqrt{Y_1} \quad \sqrt{Y_3} \quad \sqrt{Y_4}$

$\sqrt{(Y_2 \bullet Y)} \quad \sqrt{(Y_5 \bullet Y)} \quad \sqrt{(Y_{100} \bullet Y)}$

An Interactive Proof

An Interactive Proof

In order for me to "fool" you, I would have to guess your exact challenge sequence.

An Interactive Proof

In order for me to "fool" you, I would have to guess your exact challenge sequence.

The probability of my successfully convincing you that Y is a square when it is not is 2^{-100}.

An Interactive Proof

In order for me to "fool" you, I would have to guess your exact challenge sequence.

The probability of my successfully convincing you that Y is a square when it is not is 2^{-100}.

This interactive proof is said to be "*zero-knowledge*" because the challenger received no information (beyond the proof of the claim) that it couldn't compute itself.

Applying Fiat-Shamir

Once again, the verifier challenges can be simulated by the use of a one-way function to generate the challenge bits.

An Non-Interactive ZK Proof

Y

$Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad Y_5 \quad \cdots\cdots\cdots\cdots \quad Y_{100}$

An Non-Interactive ZK Proof

Y

Y_1 Y_2 Y_3 Y_4 Y_5 ……………… Y_{100}

0 1 0 0 1 ……………… 1

where the bit string is computed as
xxx = SHA-1(Y_1, Y_2,…, Y_{100})

An Non-Interactive ZK Proof

$$Y$$

Y_1 Y_2 Y_3 Y_4 Y_5 ··············· Y_{100}

0 1 0 0 1 ··············· 1

$\sqrt{Y_1}$ $\sqrt{Y_3}$ $\sqrt{Y_4}$

An Non-Interactive ZK Proof

$$Y$$

$$Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad Y_5 \quad \cdots\cdots\cdots \quad Y_{100}$$

$$0 \quad 1 \quad 0 \quad 0 \quad 1 \quad \cdots\cdots\cdots \quad 1$$

$$\sqrt{Y_1} \quad \sqrt{(Y_2 \bullet Y)} \quad \sqrt{Y_3} \quad \sqrt{Y_4} \quad \sqrt{(Y_5 \bullet Y)} \quad \sqrt{(Y_{100} \bullet Y)}$$

Proving Knowledge

Suppose that we share a public key consisting of a modulus N and an encryption exponent E and that I want to convince you that I have the corresponding decryption exponent D.

How can I do this?

Proving Knowledge

Proving Knowledge

- I can give you my private key D.

Proving Knowledge

- I can give you my private key D.

- You can encrypt something for me and I decrypt it for you.

Proving Knowledge

- I can give you my private key D.
- You can encrypt something for me and I decrypt it for you.
- You can encrypt something for me and I can engage in an interactive proof with you to show that I *can* decrypt it.

A Proof of Knowledge

Y

A Proof of Knowledge

Y

$Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad Y_5 \quad \cdots\cdots\cdots\cdots\cdots \quad Y_{100}$

A Proof of Knowledge

Y

$Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad Y_5 \quad \cdots\cdots\cdots\cdots\cdots \quad Y_{100}$

$0 \quad 1 \quad 0 \quad 0 \quad 1 \quad \cdots\cdots\cdots\cdots\cdots \quad 1$

A Proof of Knowledge

Y

Y_1 Y_2 Y_3 Y_4 Y_5 ··············· Y_{100}

0 1 0 0 1 ··············· 1

${Y_1}^D$ ${Y_3}^D$ ${Y_4}^D$

A Proof of Knowledge

Y

$Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad Y_5 \quad \cdots\cdots\cdots\cdots \quad Y_{100}$

$0 \quad 1 \quad 0 \quad 0 \quad 1 \quad \cdots\cdots\cdots\cdots \quad 1$

${Y_1}^D \quad {Y_3}^D \quad {Y_4}^D$

$(Y_2 \bullet Y)^D \quad (Y_5 \bullet Y)^D \quad (Y_{100} \bullet Y)^D$

A Proof of Knowledge

By engaging in this proof, the prover has demonstrated its knowledge of Y^D – without revealing this value.

If Y is generated by a challenger, this is compelling evidence that the prover possesses D.

Facts About Interactive Proofs

- Anything in PSPACE can be proven with a polynomial-time interactive proof.
- Anything in NP can be proven with a zero-knowledge interactive proof.

Secret Sharing

Secret Sharing

Suppose that I have some data that I want to share amongst three people such that

Secret Sharing

Suppose that I have some data that I want to share amongst three people such that

- any two can uniquely determine the data

Secret Sharing

Suppose that I have some data that I want to share amongst three people such that

- any two can uniquely determine the data
- but any one alone has *no information whatsoever* about the data.

Secret Sharing

Some simple cases: "AND"

I have a secret value z that I would like to share with Alice and Bob such that both Alice *and* Bob can together determine the secret at any time, but such that neither has any information individually.

Secret Sharing – AND

Let $z \in \mathbb{Z}_m = \{0,1,\ldots,m-1\}$ be a secret value to be shared with Alice and Bob.

Randomly and uniformly select values x and y from $\mathbb{Z}_m$ subject to the constraint that

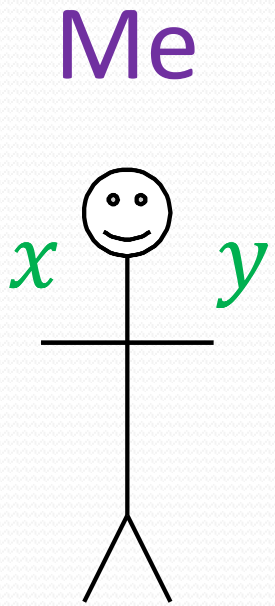
$$(x + y) \bmod m = z.$$

Secret Sharing – AND

The secret value is $z = (x + y) \bmod m$.

Secret Sharing – AND

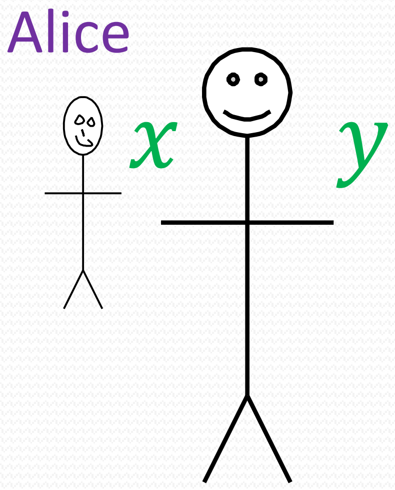
The secret value is $z = (x + y) \bmod m$.



Secret Sharing – AND

The secret value is $z = (x + y) \bmod m$.

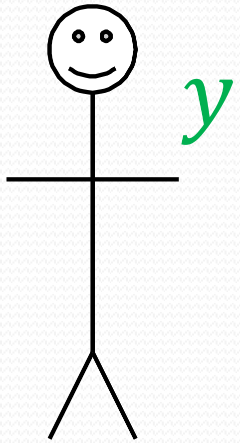
Me



Secret Sharing – AND

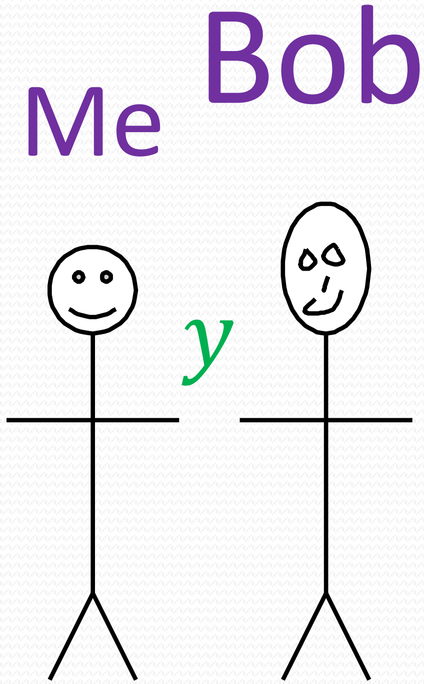
The secret value is $z = (x + y) \bmod m$.

Me



Secret Sharing – AND

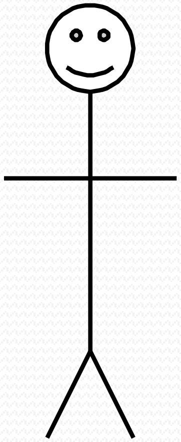
The secret value is $z = (x + y) \bmod m$.



Secret Sharing – AND

The secret value is $z = (x + y) \bmod m$.

Me

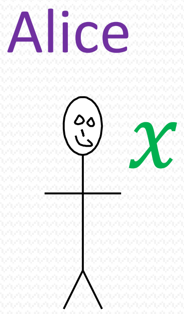


Secret Sharing – AND

The secret value is $z = (x + y) \bmod m$.

Secret Sharing – AND

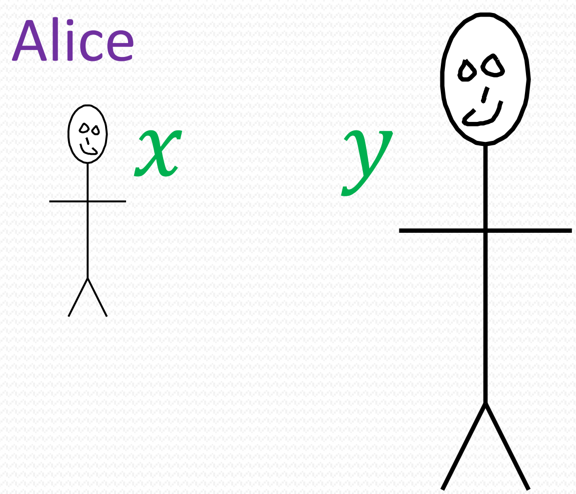
The secret value is $z = (x + y) \bmod m$.



Secret Sharing – AND

The secret value is $z = (x + y) \bmod m$.

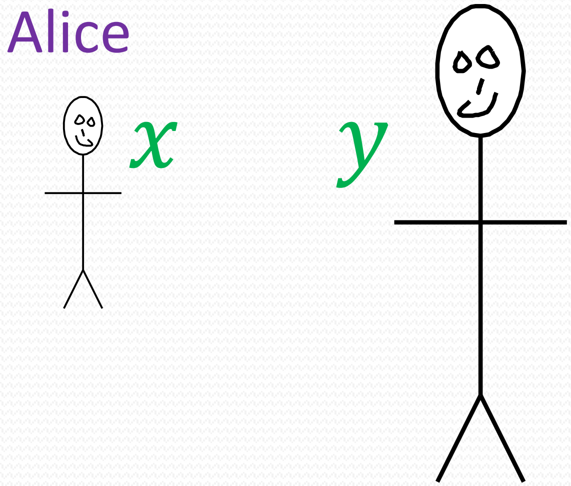
Bob



Secret Sharing – AND

The secret value is $z = (x + y) \bmod m$.

Bob



Secret Sharing – AND

This trick easily generalizes to more than two shareholders.

Secret Sharing – AND

This trick easily generalizes to more than two shareholders.

A secret S can be written as

$$S = (s_1 + s_2 + \cdots + s_n) \bmod m$$

for any randomly chosen integer values $s_1, s_2, \ldots, s_n$ in the range $0 \leq s_i < m$.

Secret Sharing

Some simple cases: "OR"

I have a secret value z that I would like to share with Alice and Bob such that either Alice *or* Bob can determine the secret at any time.

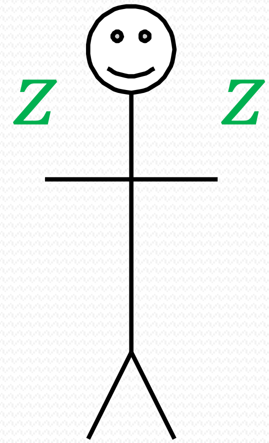
Secret Sharing – OR

The secret value is z.

Secret Sharing – OR

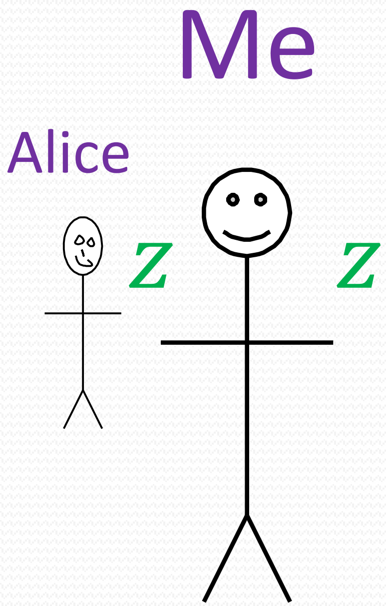
The secret value is z.

Me



Secret Sharing – OR

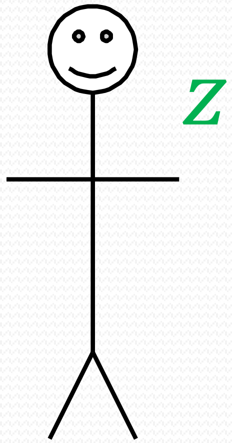
The secret value is z.



Secret Sharing – OR

The secret value is z.

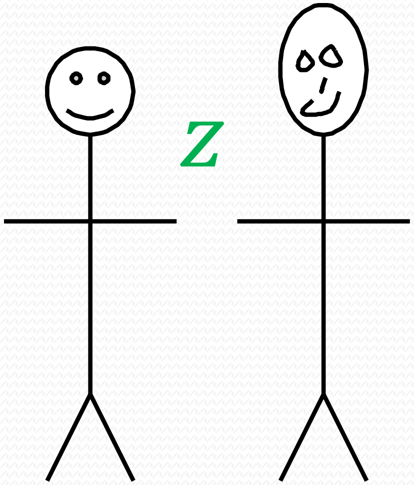
Me



Secret Sharing – OR

The secret value is z.

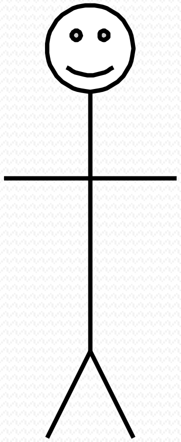
Me Bob



Secret Sharing – OR

The secret value is z.

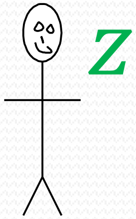
Me



Secret Sharing – OR

The secret value is z.

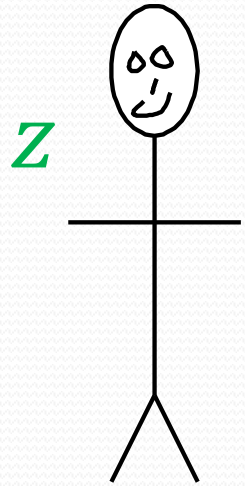
Alice



Secret Sharing – OR

The secret value is z.

Bob



Secret Sharing – OR

This case also generalizes easily to more than two shareholders.

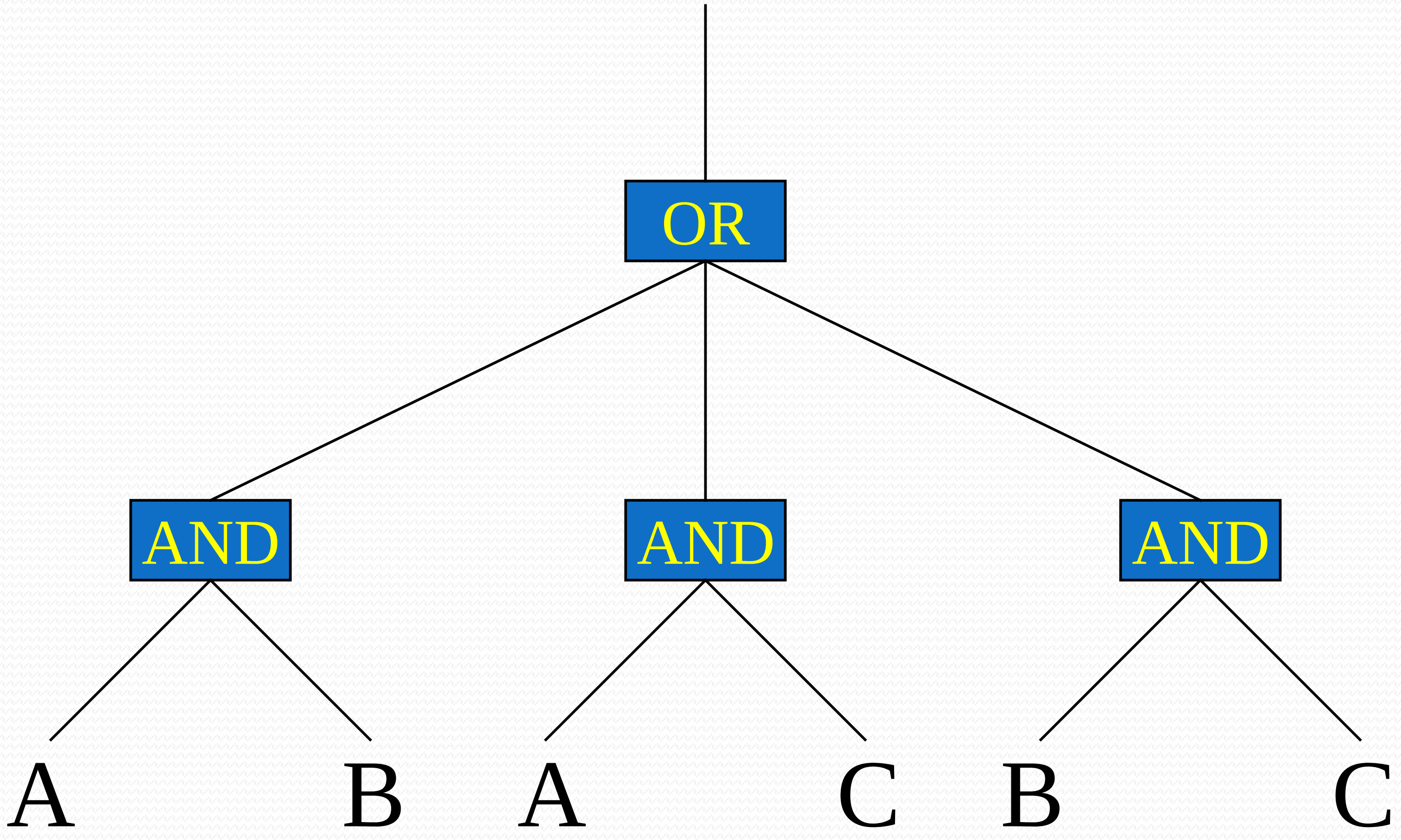
Secret Sharing

More complex *access structures* …

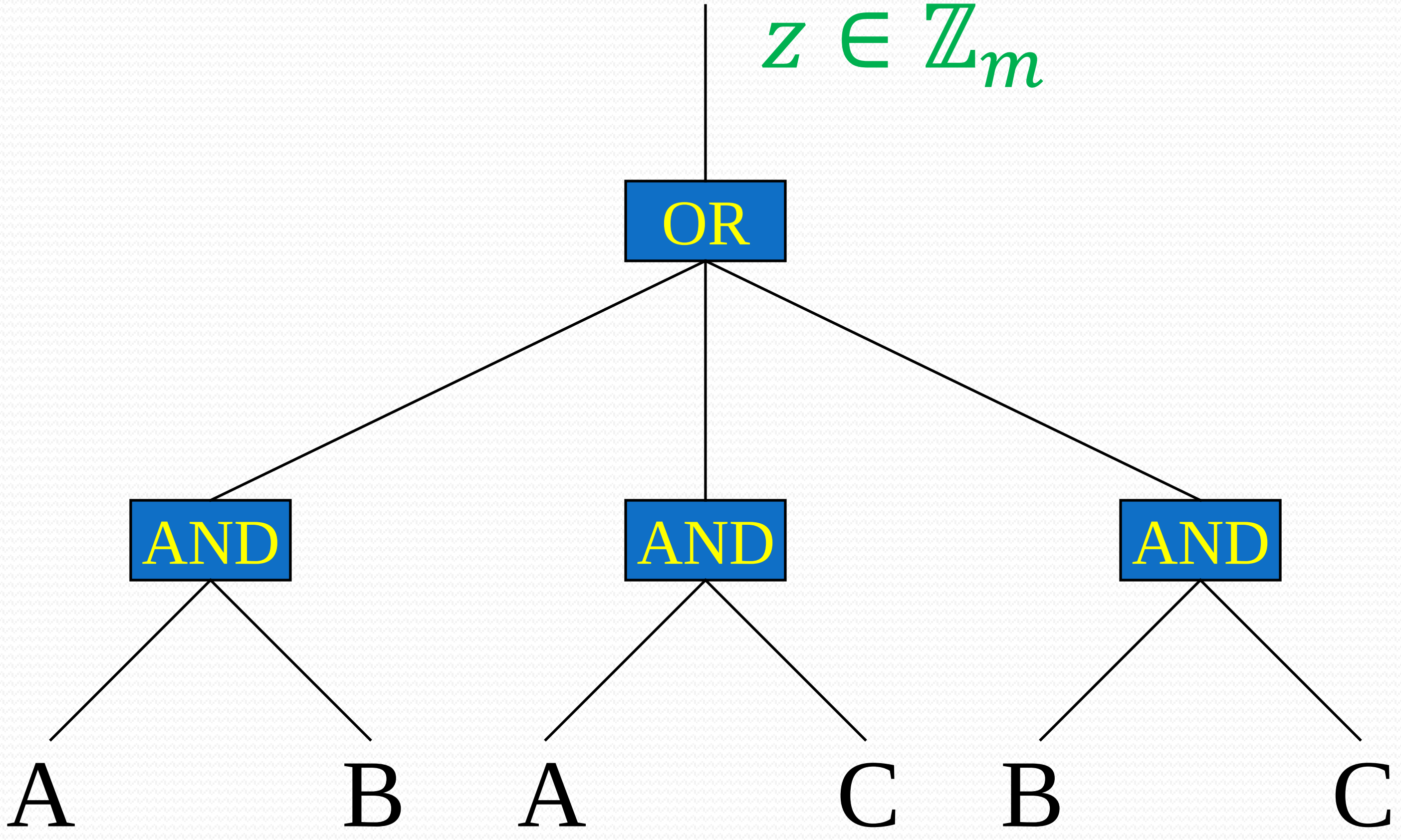
I want to share secret value z amongst Alice, Bob, and Carol such that any two of the three can reconstruct z.

$$S = (A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$$

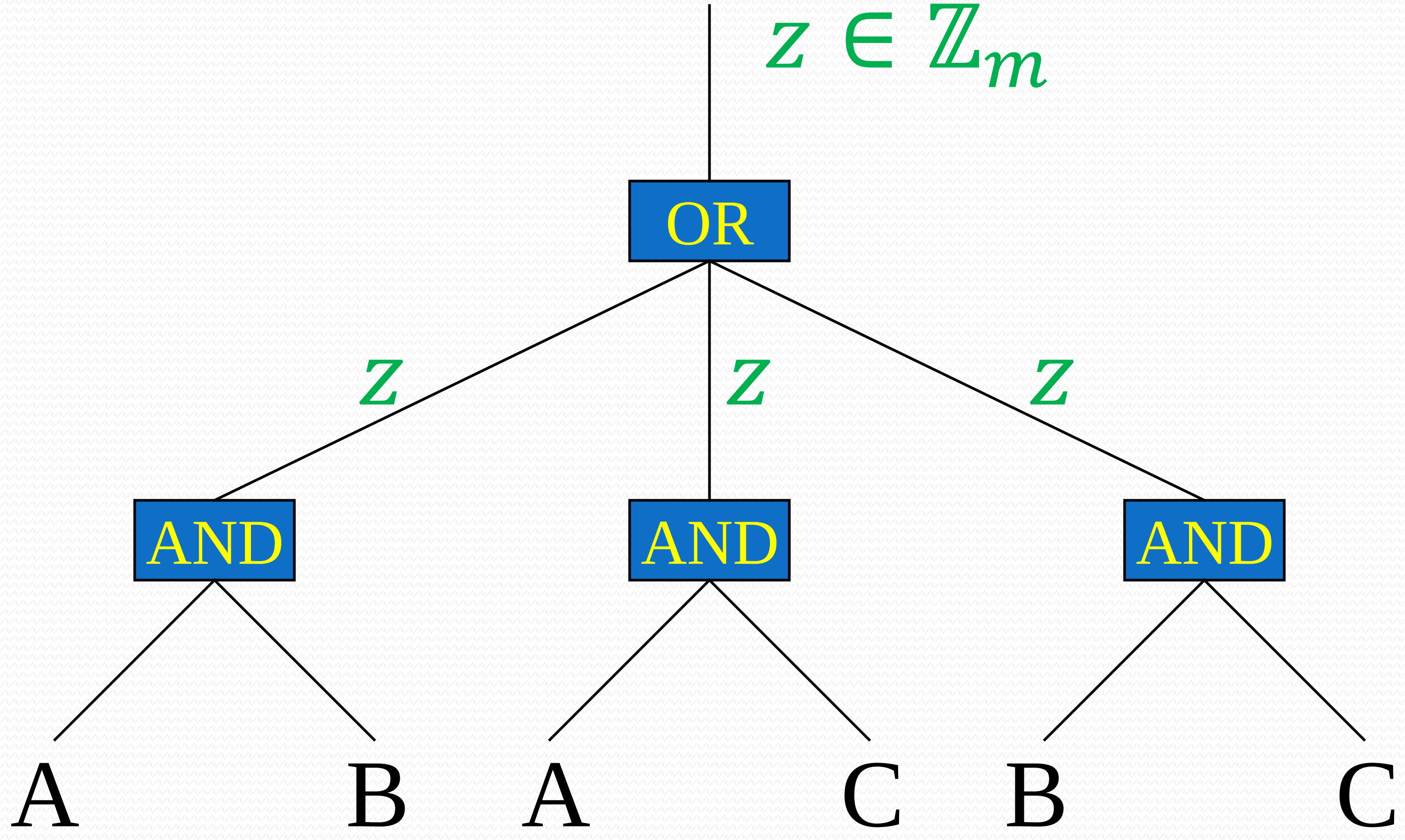
Secret Sharing



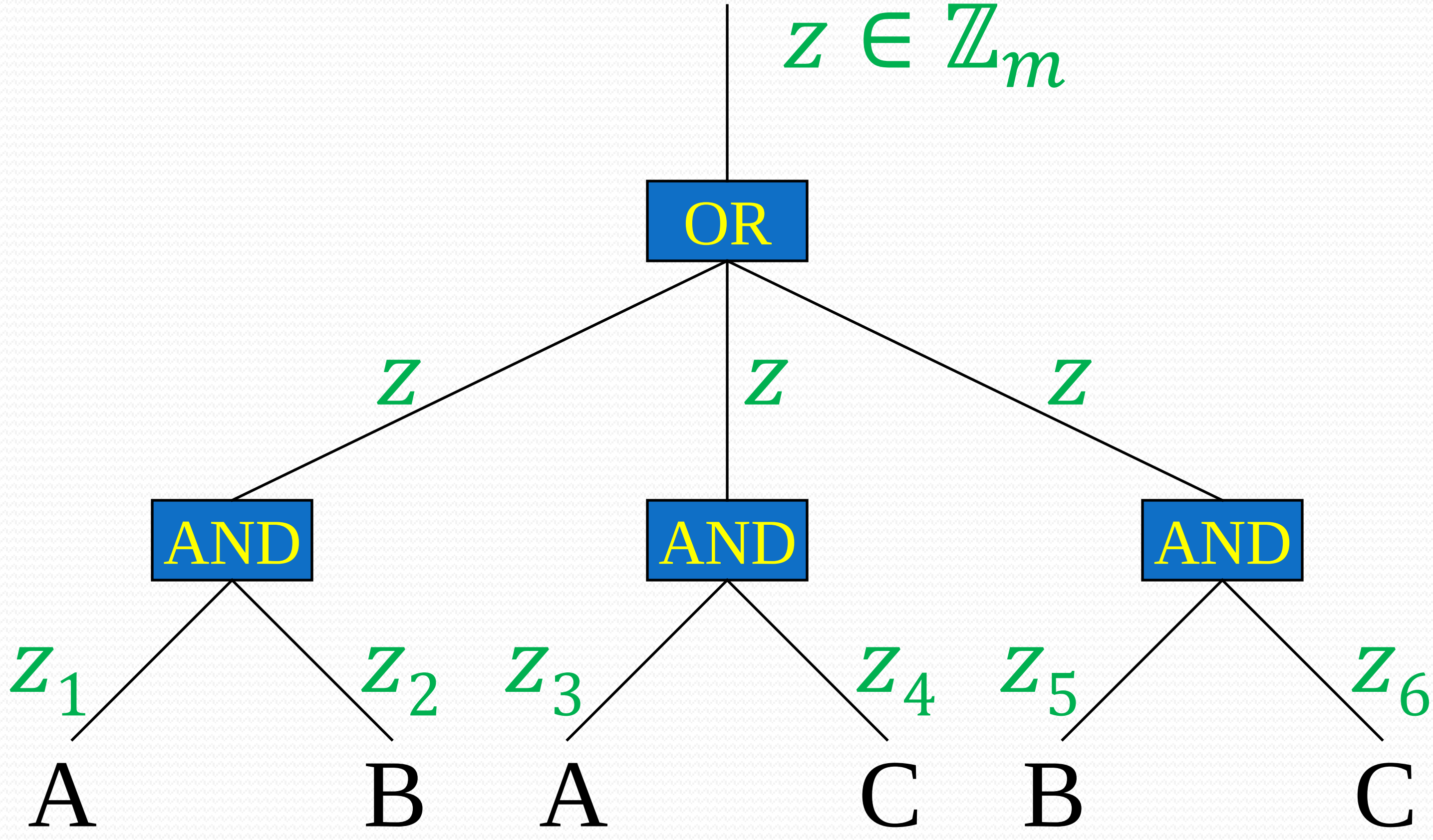
Secret Sharing



Secret Sharing



Secret Sharing



Threshold Schemes

Threshold Schemes

I want to distribute a secret datum amongst n trustees such that

Threshold Schemes

I want to distribute a secret datum amongst n trustees such that

- any k of the n trustees can uniquely determine the secret datum,

Threshold Schemes

I want to distribute a secret datum amongst n trustees such that

- any k of the n trustees can uniquely determine the secret datum,
- but any set of fewer than k trustees has *no information whatsoever* about the secret datum.

Threshold Schemes

OR	≡	1 out of n
AND	≡	n out of n

Shamir's Threshold Scheme

Any k points s_1, s_2, …, s_k in a field *uniquely* determine a polynomial P of degree at most $k-1$ with $P(i) = s_i$ for $i = 1, 2, \ldots, k$.

This not only works of the reals, rationals, and other infinite fields, but also over the finite field

$$\mathbb{Z}_p = \{0,1, \ldots, p-1\}$$

where p is a prime.

Shamir's Threshold Scheme

To distribute a secret value $s \in \mathbb{Z}_p$ amongst a set of n Trustees $\{T_1, T_2, \ldots, T_n\}$ such that any k can determine the secret

Shamir's Threshold Scheme

To distribute a secret value $s \in \mathbb{Z}_p$ amongst a set of n Trustees $\{T_1, T_2, \dots, T_n\}$ such that any k can determine the secret

- pick random *coefficients* $a_1, a_2, \dots, a_{k-1} \in \mathbb{Z}_p$

Shamir's Threshold Scheme

To distribute a secret value $s \in \mathbb{Z}_p$ amongst a set of n Trustees $\{T_1, T_2, \dots, T_n\}$ such that any k can determine the secret

- pick random *coefficients* $a_1, a_2, \dots, a_{k-1} \in \mathbb{Z}_p$
- let $P(x) = a_{k-1}x^{k-1} + \cdots + a_2x^2 + a_1x + s$

Shamir's Threshold Scheme

To distribute a secret value $s \in \mathbb{Z}_p$ amongst a set of n Trustees $\{T_1, T_2, \ldots, T_n\}$ such that any k can determine the secret

- pick random *coefficients* $a_1, a_2, \ldots, a_{k-1} \in \mathbb{Z}_p$
- let $P(x) = a_{k-1}x^{k-1} + \cdots + a_2x^2 + a_1x + s$
- give $P(i)$ to trustee T_i.

Shamir's Threshold Scheme

To distribute a secret value $s \in \mathbb{Z}_p$ amongst a set of n Trustees $\{T_1, T_2, \ldots, T_n\}$ such that any k can determine the secret

- pick random *coefficients* $a_1, a_2, \ldots, a_{k-1} \in \mathbb{Z}_p$
- let $P(x) = a_{k-1}x^{k-1} + \cdots + a_2x^2 + a_1x + s$
- give $P(i)$ to trustee T_i.

The secret value is $s = P(0)$.

Shamir's Threshold Scheme

The threshold 2 case:

Example: Range = $\mathbb{Z}_{11} = \{0,1,\ldots,10\}$, Secret = 9

Shamir’s Threshold Scheme

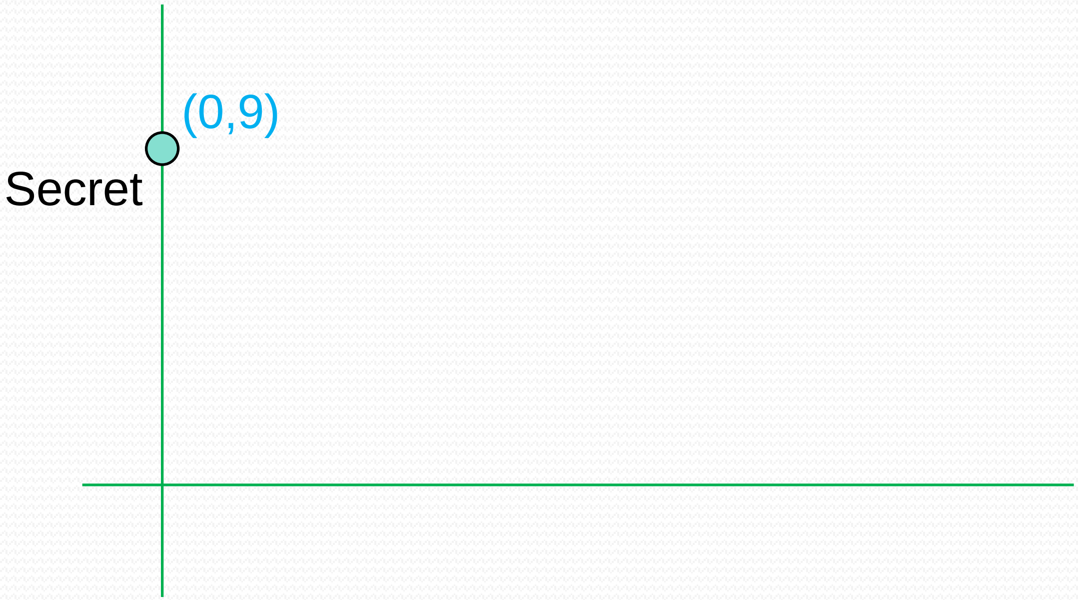
The threshold 2 case:

Example: Range = $\mathbb{Z}_{11} = \{0,1,\ldots,10\}$, Secret = 9

Shamir's Threshold Scheme

The threshold 2 case:

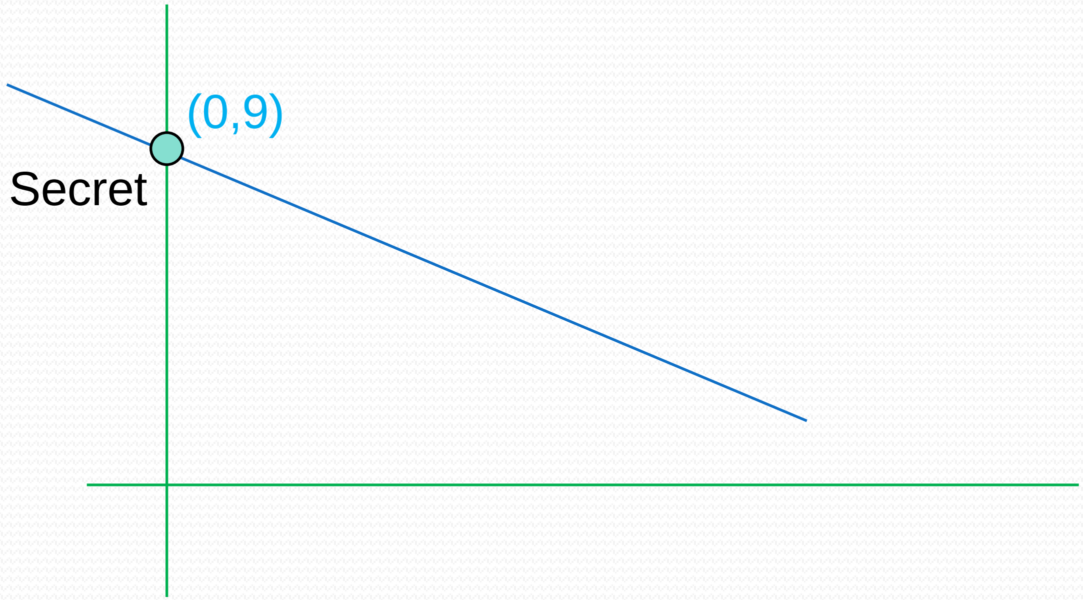
Example: Range = $\mathbb{Z}_{11} = \{0,1,\ldots,10\}$, Secret = 9



Shamir's Threshold Scheme

The threshold 2 case:

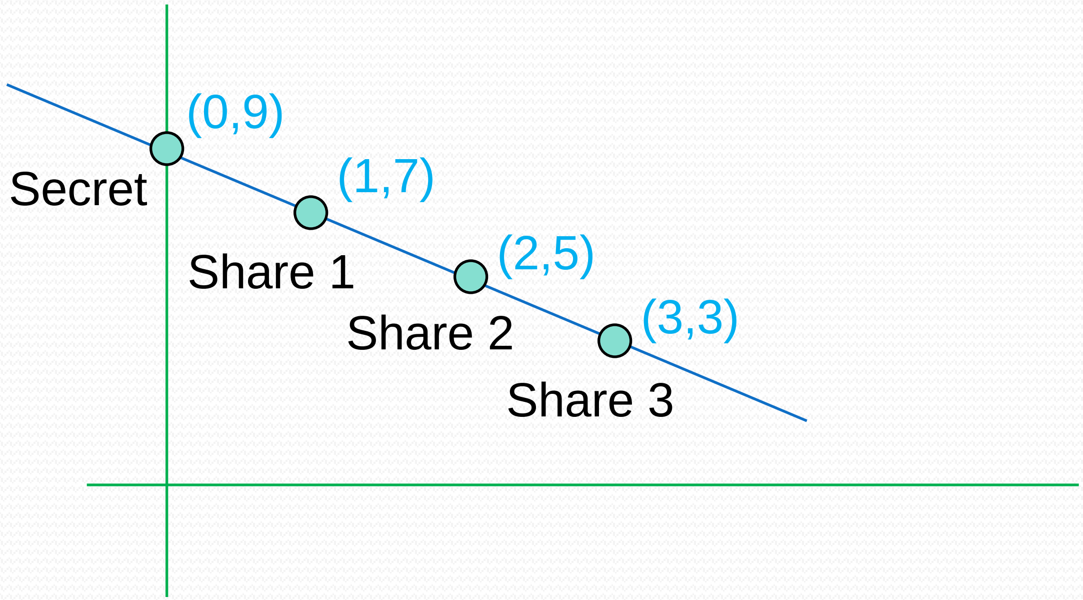
Example: Range = $\mathbb{Z}_{11} = \{0,1,\ldots,10\}$, Secret = 9



Shamir's Threshold Scheme

The threshold 2 case:

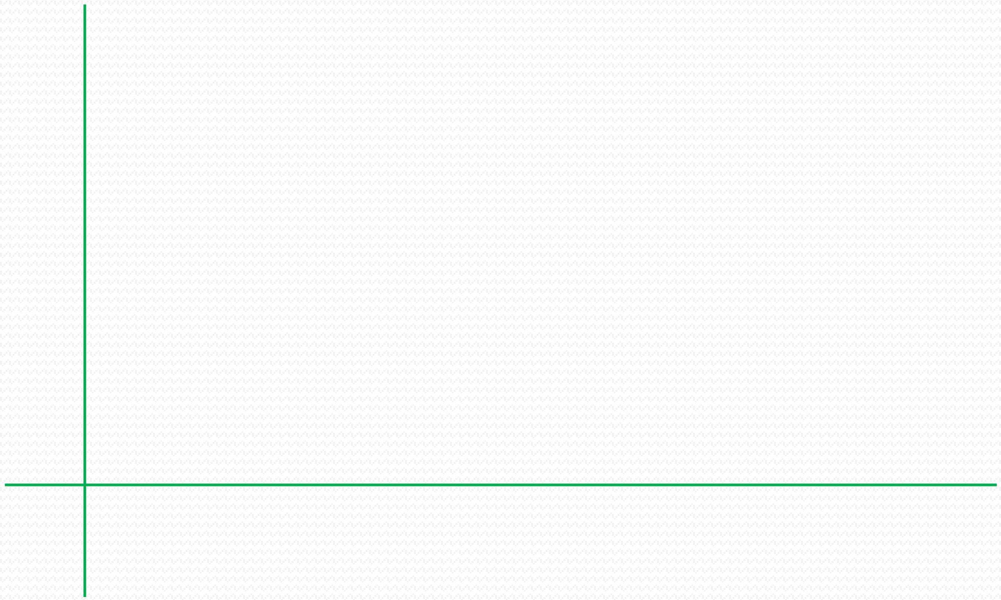
Example: Range = $\mathbb{Z}_{11} = \{0,1,\ldots,10\}$, Secret = 9



Shamir's Threshold Scheme

The threshold 2 case:

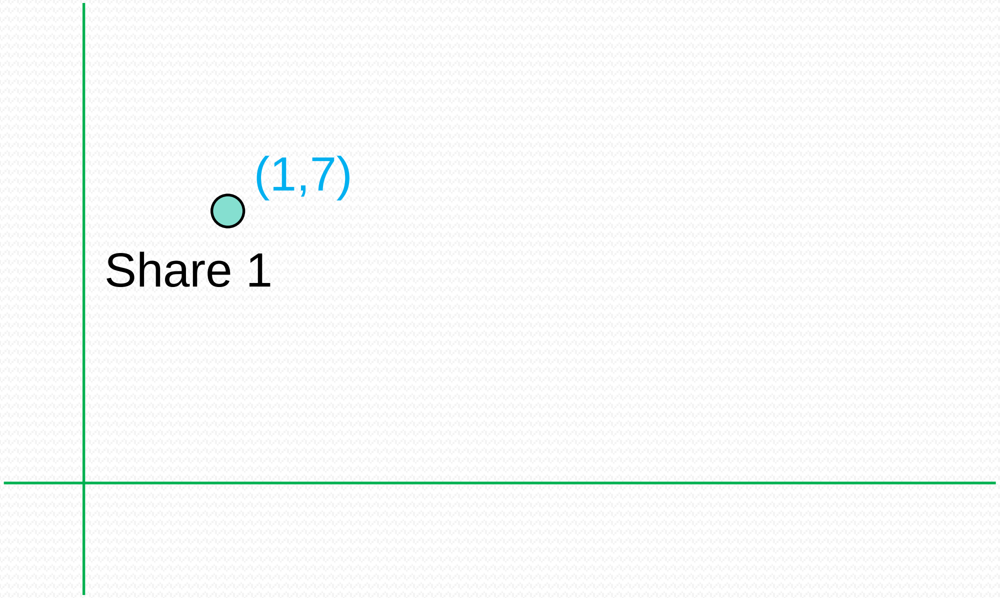
Example: Range = $\mathbb{Z}_{11} = \{0,1,\ldots,10\}$, Secret = 9



Shamir's Threshold Scheme

The threshold 2 case:

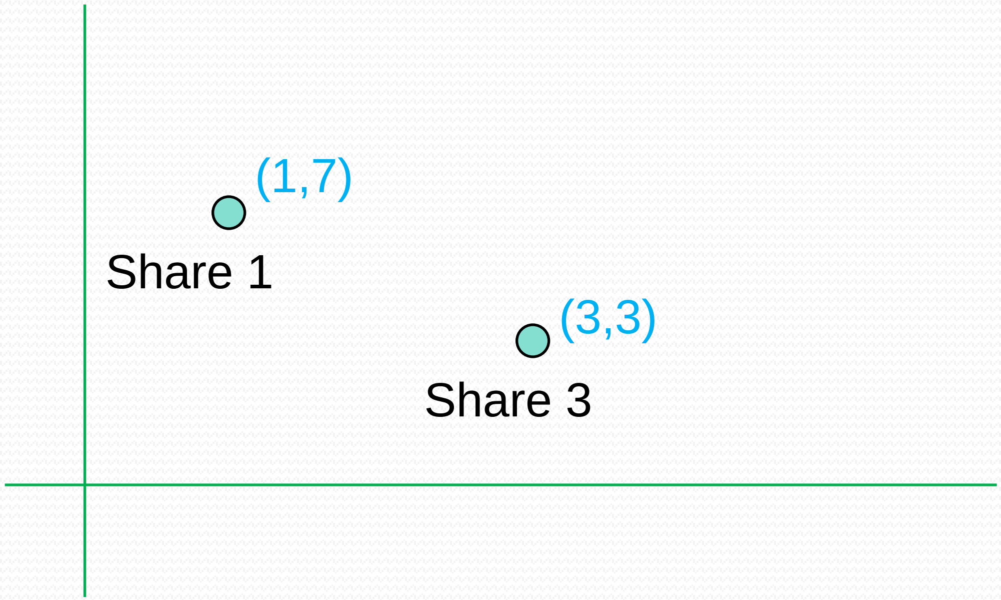
Example: Range = $\mathbb{Z}_{11} = \{0,1,\ldots,10\}$, Secret = 9



Shamir's Threshold Scheme

The threshold 2 case:

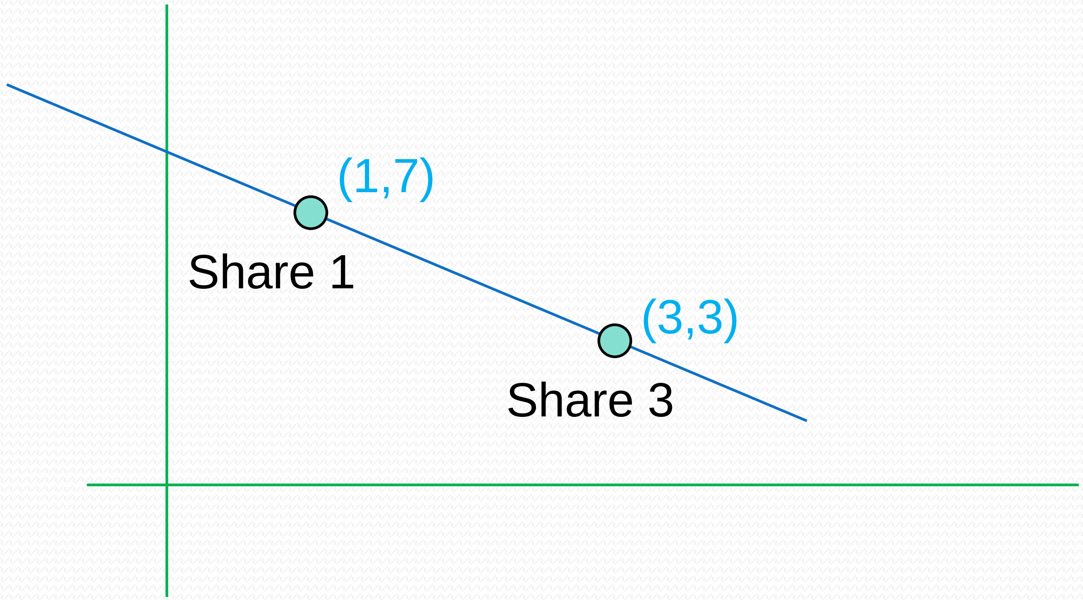
Example: Range = $\mathbb{Z}_{11} = \{0,1,\ldots,10\}$, Secret = 9



Shamir's Threshold Scheme

The threshold 2 case:

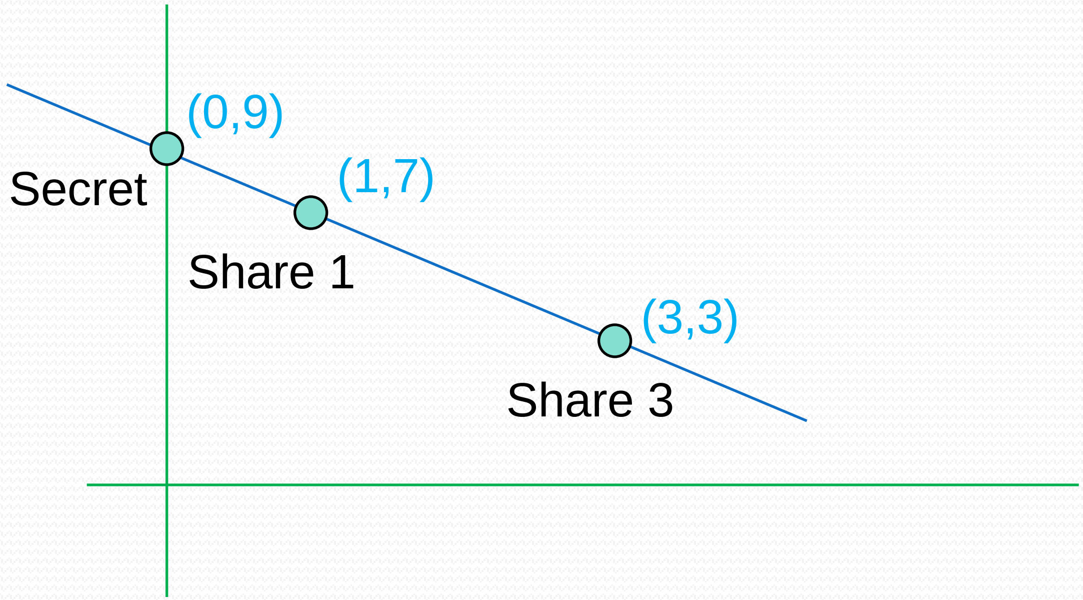
Example: Range = $\mathbb{Z}_{11} = \{0,1,\ldots,10\}$, Secret = 9



Shamir's Threshold Scheme

The threshold 2 case:

Example: Range = $\mathbb{Z}_{11} = \{0,1,\dots,10\}$, Secret = 9



Shamir's Threshold Scheme

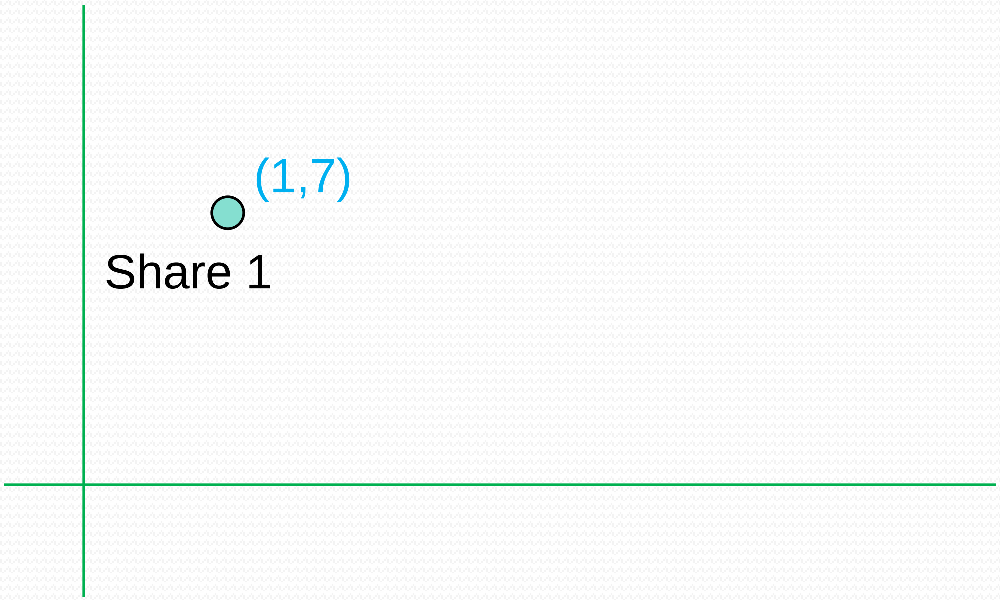
The threshold 2 case:

Example: Range = $\mathbb{Z}_{11} = \{0,1,\ldots,10\}$

Shamir's Threshold Scheme

The threshold 2 case:

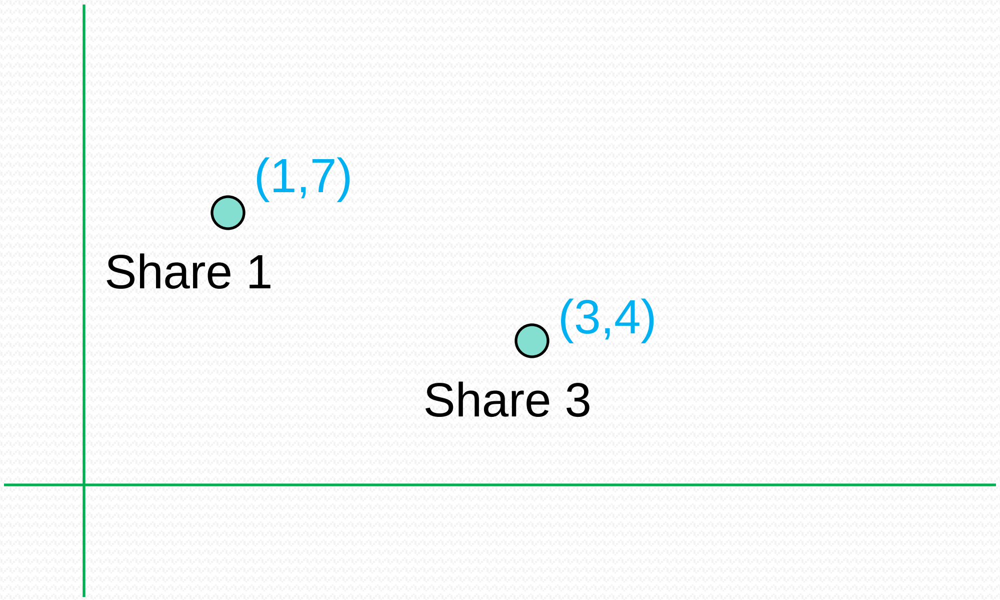
Example: Range = $\mathbb{Z}_{11} = \{0,1,\ldots,10\}$



Shamir's Threshold Scheme

The threshold 2 case:

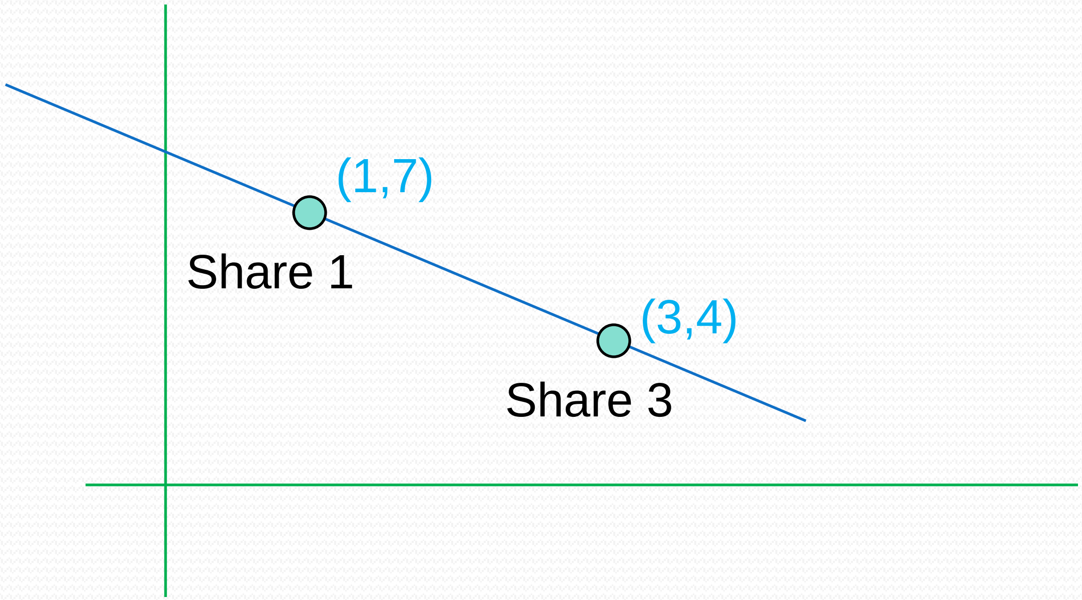
Example: Range = $\mathbb{Z}_{11} = \{0,1,\ldots,10\}$



Shamir's Threshold Scheme

The threshold 2 case:

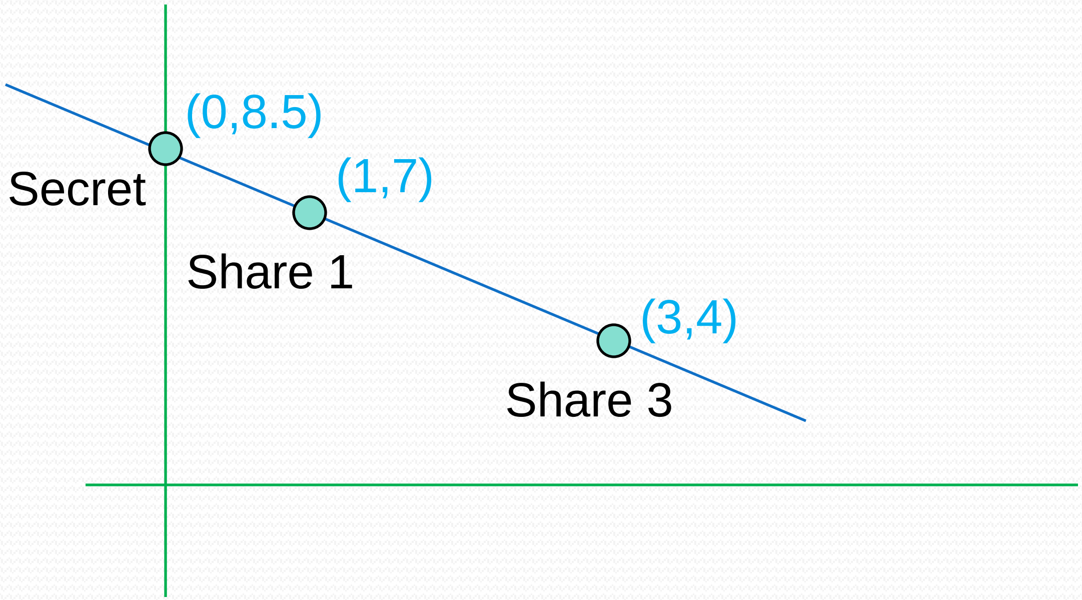
Example: Range = $\mathbb{Z}_{11} = \{0,1, \ldots, 10\}$



Shamir's Threshold Scheme

The threshold 2 case:

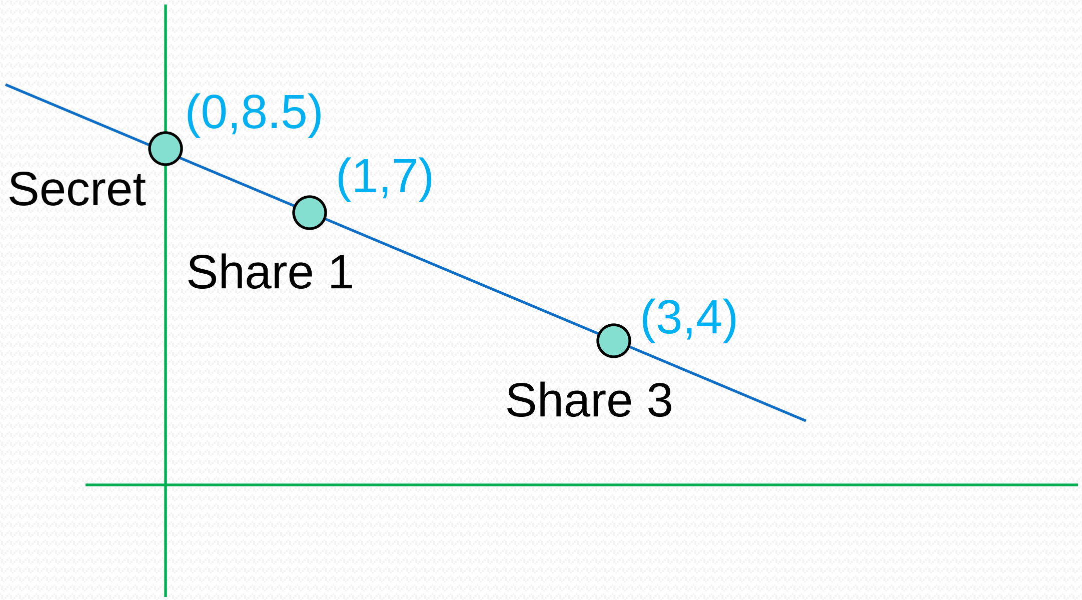
Example: Range = $\mathbb{Z}_{11} = \{0,1,\ldots,10\}$



Shamir's Threshold Scheme

The threshold 2 case:

Example: Range = $\mathbb{Z}_{11} = \{0,1,\ldots,10\}$



In $\mathbb{Z}_{11}$, 8.5

$$\equiv 17 \div 2$$
$$\equiv 6 \times 6$$
$$\equiv 36$$
$$\equiv 3$$

Shamir's Threshold Scheme

Two methods are commonly used to interpolate a polynomial given a set of points.

Shamir's Threshold Scheme

Two methods are commonly used to interpolate a polynomial given a set of points.

- Lagrange interpolation

Shamir's Threshold Scheme

Two methods are commonly used to interpolate a polynomial given a set of points.

- Lagrange interpolation
- Solving a system of linear equations

Lagrange Interpolation

Lagrange Interpolation

For each point (i, s_i), construct a polynomial P_i with the correct value at i and a value of zero at the other given points.

Lagrange Interpolation

For each point (i, s_i), construct a polynomial P_i with the correct value at i and a value of zero at the other given points.

$$P_i(x) = s_i \times \prod_{j \neq i} (x - j) \div \prod_{j \neq i} (i - j)$$

Lagrange Interpolation

For each point (i, s_i), construct a polynomial P_i with the correct value at i and a value of zero at the other given points.

$$P_i(x) = s_i \times \prod_{j \neq i} (x - j) \div \prod_{j \neq i} (i - j)$$

Then sum the $P_i(x)$ to compute $P(x)$.

Lagrange Interpolation

For each point (i, s_i), construct a polynomial P_i with the correct value at i and a value of zero at the other given points.

$$P_i(x) = s_i \times \prod_{j \neq i} (x - j) \div \prod_{j \neq i} (i - j)$$

Then sum the $P_i(x)$ to compute $P(x)$.

$$P(x) = \sum_i P_i(x)$$

Solving a Linear System

Solving a Linear System

- Regard the polynomial coefficients as unknowns.

Solving a Linear System

- Regard the polynomial coefficients as unknowns.
- Plug in each known point to get a *linear* equation in terms of the unknown coefficients.

Solving a Linear System

- Regard the polynomial coefficients as unknowns.
- Plug in each known point to get a *linear* equation in terms of the unknown coefficients.
- Once there are as many equations as unknowns, use linear algebra to solve the system of equations.

Verifiable Secret Sharing

Secret sharing is very useful when the “dealer” of a secret is honest, but what bad things can happen if the dealer is potentially dishonest?

Can measures be taken to eliminate or mitigate the damages?

Homomorphic Encryption

Recall that with RSA, there is a multiplicative *homomorphism*.

$$E(x)E(y) \equiv E(xy)$$

Can we find an encryption function with an additive homomorphism?

An Additive Homomorphism

Can we find an encryption function for which the sum (or product) of two encrypted messages is the (an) encryption of the sum of the two original messages?

$$E(x) \circ E(y) \equiv E(x + y)$$

An Additive Homomorphism

Recall the one-way function given by

$$f(x) = g^x \bmod m.$$

For this function,

$$f(x)f(y) \bmod m = g^x g^y \bmod m = g^{x+y} \bmod m = f(x+y) \bmod m.$$

Verifiable Secret Sharing

Verifiable Secret Sharing

- Select a polynomial with secret a_0 as

$$P(x) = a_{k-1}x^{k-1} + \cdots + a_2x^2 + a_1x + a_0.$$

Verifiable Secret Sharing

- Select a polynomial with secret a_0 as

$$P(x) = a_{k-1}x^{k-1} + \cdots + a_2x^2 + a_1x + a_0.$$

- Commit to the coefficients by publishing

$$g^{a_0}, g^{a_1}, g^{a_2}, \dots, g^{a_{k-1}}.$$

Verifiable Secret Sharing

- Select a polynomial with secret a_0 as

$$P(x) = a_{k-1}x^{k-1} + \cdots + a_2x^2 + a_1x + a_0.$$

- Commit to the coefficients by publishing

$$g^{a_0}, g^{a_1}, g^{a_2}, \ldots, g^{a_{k-1}}.$$

- Compute a commitment to $P(i)$ from public values as

$$g^{P(i)} = g^{a_0 i^0} g^{a_1 i^1} g^{a_2 i^2} \ldots g^{a_{k-1} i^{k-1}}.$$

Verifiable Secret Sharing

<u>An important detail</u>

Randomness must be included to prevent small spaces of possible secrets and shares from being exhaustively searched.

Secret Sharing Homomorphisms

All of these secret sharing methods have an additional useful feature:

If two secrets are separately shared amongst the same set of people in the same way, then the sum of the individual shares constitute shares of the sum of the secrets.

Secret Sharing Homomorphisms

OR

Secret: a – Shares: $a, a, \ldots, a$

Secret: b – Shares: $b, b, \ldots, b$

Secret sum: $a + b$

Share sums: $a + b, a + b, \ldots, a + b$

Secret Sharing Homomorphisms

AND

Secret: a – Shares: $a_1, a_2, \ldots, a_n$

Secret: b – Shares: $b_1, b_2, \ldots, b_n$

Secret sum: $a + b$

Share sums: $a_1 + b_1, a_2 + b_2, \ldots, a_n + bn$

Secret Sharing Homomorphisms

THRESHOLD

Secret: $P_1(0)$ – Shares: $P_1(1), P_1(2), \ldots, P_1(n)$

Secret: $P_2(0)$ – Shares: $P_2(1), P_2(2), \ldots, P_2(n)$

Secret sum: $P_1(0) + P_2(0)$

Share sums: $P_1(1) + P_2(1), P_1(2) + P_2(2), \ldots, P_1(n) + P_2(n)$

Threshold Encryption

I want to encrypt a secret message M for a set of n recipients such that

- any k of the n recipients can uniquely decrypt the secret message M,
- but any set of fewer than k recipients has *no information whatsoever* about the secret message M.

Recall Diffie-Hellman

Alice

- Randomly select a large integer a and send $A = g^a \bmod p$.
- Compute the key $K = B^a \bmod p$.

Bob

- Randomly select a large integer b and send $B = g^b \bmod p$.
- Compute the key $K = A^b \bmod p$.

$$B^a = g^{ba} = g^{ab} = A^b$$

ElGamal Encryption

ElGamal Encryption

- Alice selects a large random private key a and computes an associated public key $A = g^a \bmod p$.

ElGamal Encryption

- Alice selects a large random private key a and computes an associated public key $A = g^a \bmod p$.
- To send a message M to Alice, Bob selects a random value r and computes the pair

$$(X, Y) = (A^r M \bmod p, g^r \bmod p).$$

ElGamal Encryption

- Alice selects a large random private key a and computes an associated public key $A = g^a \bmod p$.
- To send a message M to Alice, Bob selects a random value r and computes the pair
  $$(X, Y) = (A^r M \bmod p, g^r \bmod p).$$
- To decrypt, Alice computes
  $$X/Y^a \bmod p = A^r M/g^{ra} \bmod p = M.$$

ElGamal Re-Encryption

If $A = g^a \bmod p$ is a public key and the pair

$$(X, Y) = (A^r M \bmod p, g^r \bmod p)$$

is an encryption of message M, then for any value c, the pair

$$(A^c X, g^c Y) = (A^{c+r} M \bmod p, g^{c+r} \bmod p)$$

is an encryption of the same message M, for any value c.

Group ElGamal Encryption

Group ElGamal Encryption

- Each recipient selects a large random private key a_i and computes an associated public key $A_i = g^{a_i} \bmod p$.

Group ElGamal Encryption

- Each recipient selects a large random private key a_i and computes an associated public key $A_i = g^{a_i} \bmod p$.
- The group key is $A = \prod A_i \bmod p = g^{\sum a_i} \bmod p$.

Group ElGamal Encryption

- Each recipient selects a large random private key a_i and computes an associated public key $A_i = g^{a_i} \bmod p$.
- The group key is $A = \prod A_i \bmod p = g^{\sum a_i} \bmod p$.
- To send a message M to the group, Bob selects a random value r and computes the pair $(X, Y) = (A^r M \bmod p, g^r \bmod p)$.

Group ElGamal Encryption

- Each recipient selects a large random private key a_i and computes an associated public key $A_i = g^{a_i} \bmod p$.
- The group key is $A = \prod A_i \bmod p = g^{\sum a_i} \bmod p$.
- To send a message M to the group, Bob selects a random value r and computes the pair $(X, Y) = (A^r M \bmod p, g^r \bmod p)$.
- To decrypt, each group member computes $Y_i = Y^{a_i} \bmod p$. The message $M = X / \prod Y_i \bmod p$.

Threshold Encryption (ElGamal)

Threshold Encryption (ElGamal)

- Each recipient selects k large random secret coefficients $a_{i,0}$, $a_{i,1}$, …, $a_{i,k-2}$, $a_{i,k-1}$ and forms the polynomial
  $P_i(x) = a_{i,k-1}x^{k-1} + a_{i,k-2}x^{k-2} + \cdots + a_{i,1}x + a_{i,0}$

Threshold Encryption (ElGamal)

- Each recipient selects k large random secret coefficients $a_{i,0}, a_{i,1}, \ldots, a_{i,k-2}, a_{i,k-1}$ and forms the polynomial
$$P_i(x) = a_{i,k-1}x^{k-1} + a_{i,k-2}x^{k-2} + \cdots + a_{i,1}x + a_{i,0}$$
- Each polynomial $P_i(x)$ is then verifiably shared with the other recipients by distributing each $g^{a_{i,j}}$.

Threshold Encryption (ElGamal)

- Each recipient selects k large random secret coefficients $a_{i,0}, a_{i,1}, \ldots, a_{i,k-2}, a_{i,k-1}$ and forms the polynomial
  $$P_i(x) = a_{i,k-1}x^{k-1} + a_{i,k-2}x^{k-2} + \cdots + a_{i,1}x + a_{i,0}$$
- Each polynomial $P_i(x)$ is then verifiably shared with the other recipients by distributing each $g^{a_{i,j}}$.
- The joint (threshold) public key is $\prod g^{a_{i,0}}$.

Threshold Encryption (ElGamal)

- Each recipient selects k large random secret coefficients $a_{i,0}, a_{i,1}, \ldots, a_{i,k-2}, a_{i,k-1}$ and forms the polynomial
  $$P_i(x) = a_{i,k-1}x^{k-1} + a_{i,k-2}x^{k-2} + \cdots + a_{i,1}x + a_{i,0}$$
- Each polynomial $P_i(x)$ is then verifiably shared with the other recipients by distributing each $g^{a_{i,j}}$.
- The joint (threshold) public key is $\prod g^{a_{i,0}}$.
- Any set of k recipients can form the secret key $\sum a_{i,0}$ to decrypt.