Cryptography Lecture 11

Bit commitment, zero knowledge, secret sharing, games over the phone

LINKÖPING UNIVERSITY

Bit commitment

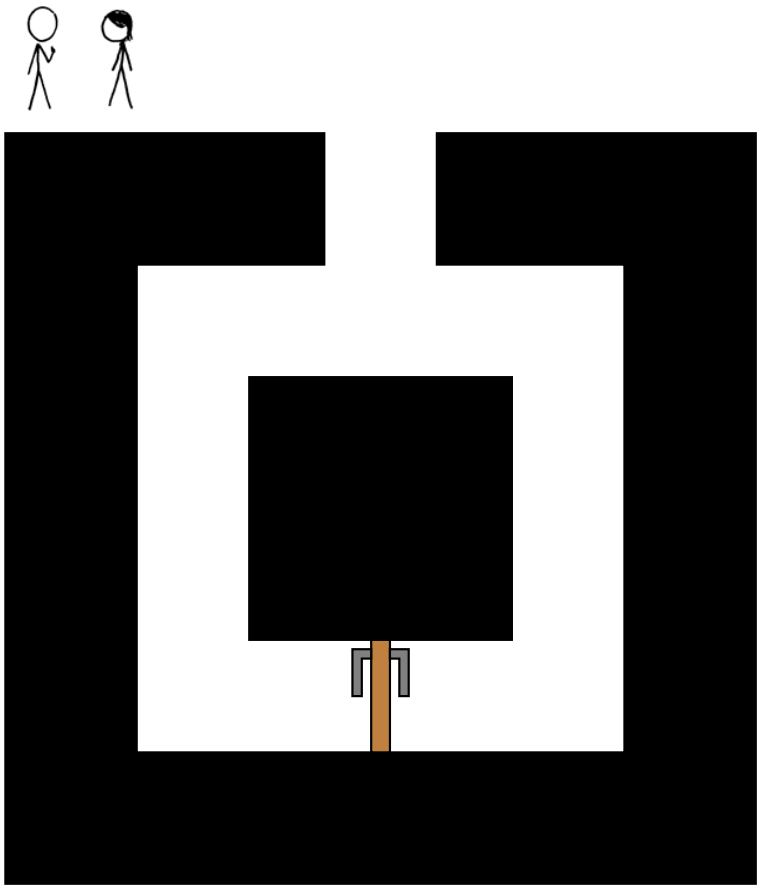
- Alice wants to commit to a value of a bit b, such as the outcome of a Hockey game the next weekend
- Her intent is to show Bob she can predict the outcome, but she does not want to tell Bob how, or what the result is (so that he won't reduce Alice's winnings)
- But nonetheless she wants to give Bob something, that he can use to verify that she knew, after the game has been played

Bit commitment, using discrete log

- Alice will base her commitment scheme on discrete logs, so sets up like ElGamal: a prime p and primitive root α and makes them public
- She now chooses a random x whose second bit is b, and sends $\beta = \alpha^x$ to Bob
  - It is easy to compute discrete log mod 2
  - but hard to compute discrete log mod 4 (see the book)
- When Alice reveals the value x to Bob, he can check that α^x and β are the same
- This will work with any one-way function (typically you will want to use a strongly collision-resistant hash function)

Zero knowledge

- A “zero-knowledge” scheme is intended to convince someone that you have certain knowledge, without giving the knowledge to this someone

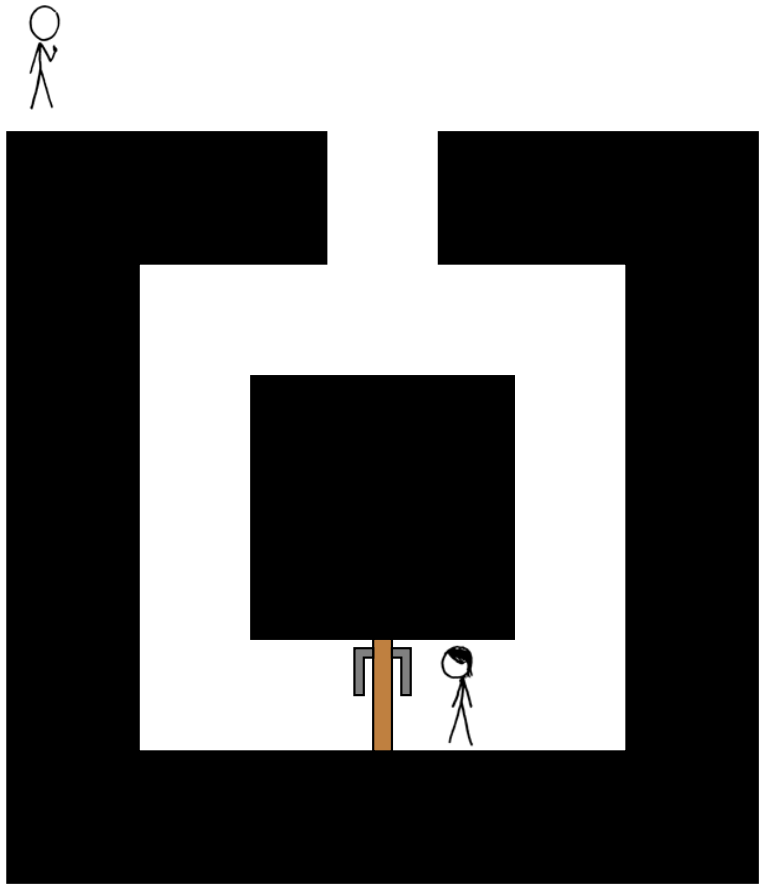


- The standard example is that Peggy claims to know how to open a door inside a tunnel system
- She wants to convince Victor that she can do this, without telling Victor how to open the door

Zero knowledge

- A “zero-knowledge” scheme is intended to convince someone that you have certain knowledge, without giving the knowledge to this someone

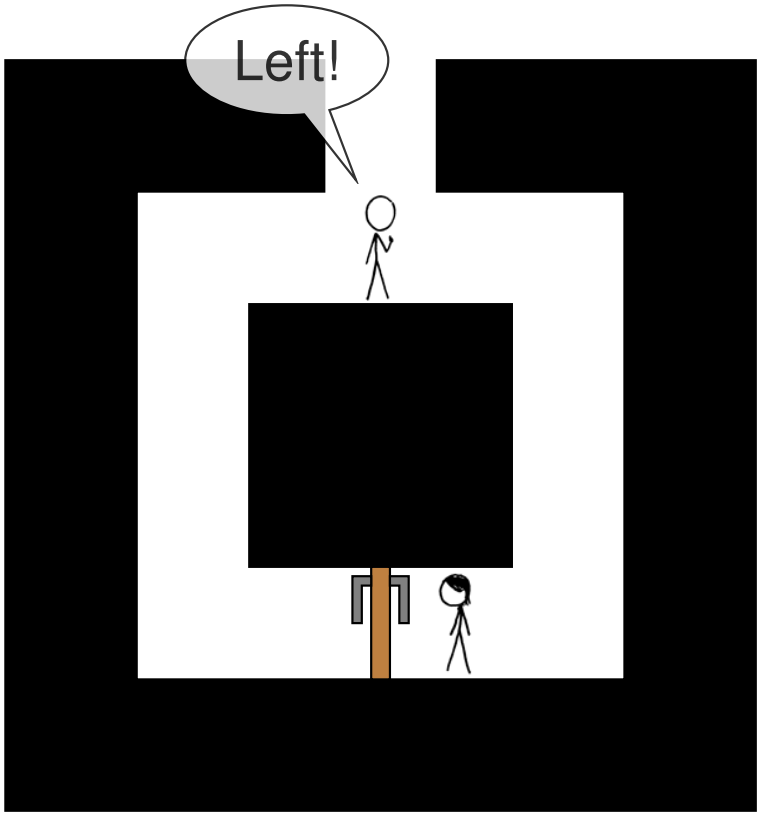
1. Peggy enters, while Victor waits outside



Zero knowledge

- A “zero-knowledge” scheme is intended to convince someone that you have certain knowledge, without giving the knowledge to this someone

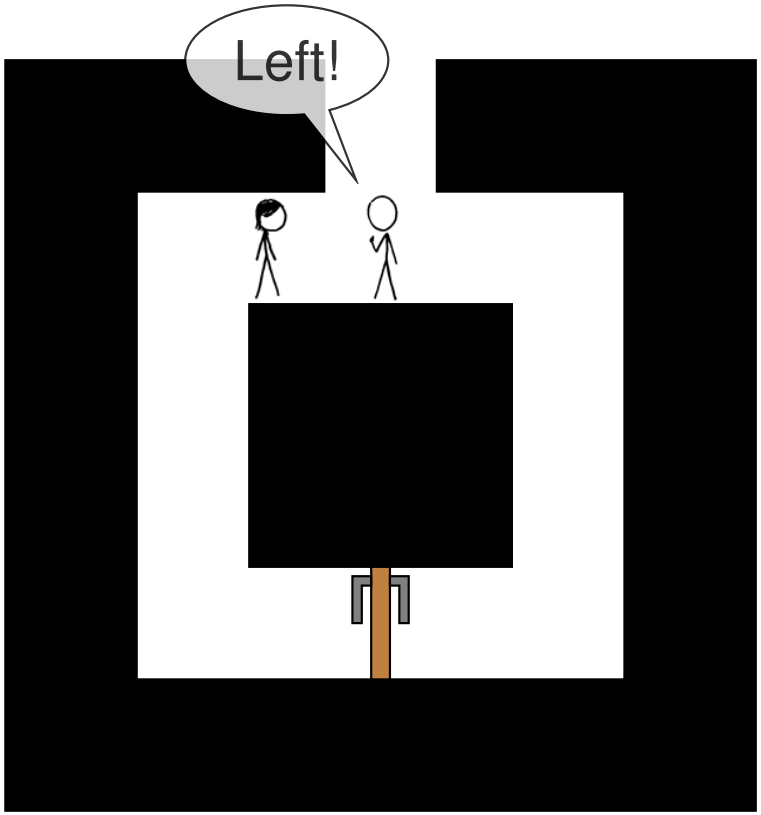
1. Peggy enters, while Victor waits outside
2. Victor goes to the intersection and calls out “left” or “right”



Zero knowledge

- A “zero-knowledge” scheme is intended to convince someone that you have certain knowledge, without giving the knowledge to this someone

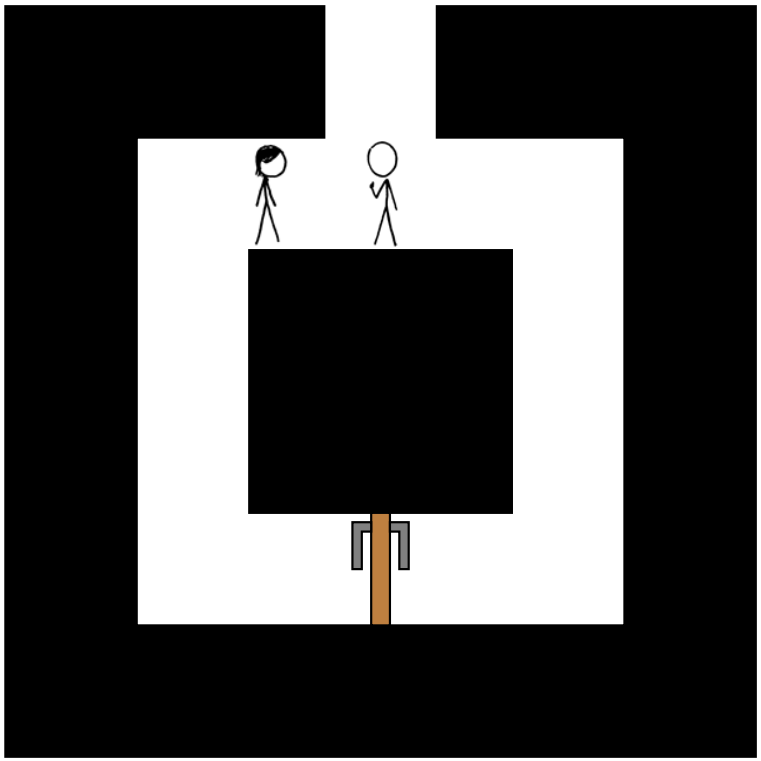
1. Peggy enters, while Victor waits outside
2. Victor goes to the intersection and calls out “left” or “right”
3. Peggy exits from the correct path



Zero knowledge

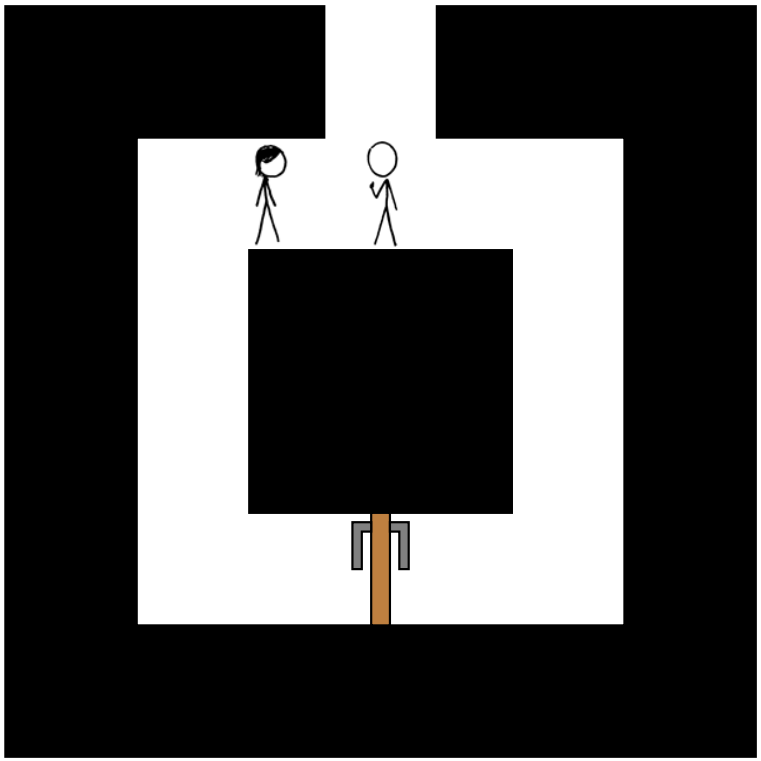
- A “zero-knowledge” scheme is intended to convince someone that you have certain knowledge, without giving the knowledge to this someone

1. Peggy enters, while Victor waits outside
2. Victor goes to the intersection and calls out “left” or “right”
3. Peggy exits from the correct path
4. If this happens enough times, Victor is convinced that Peggy can open the door



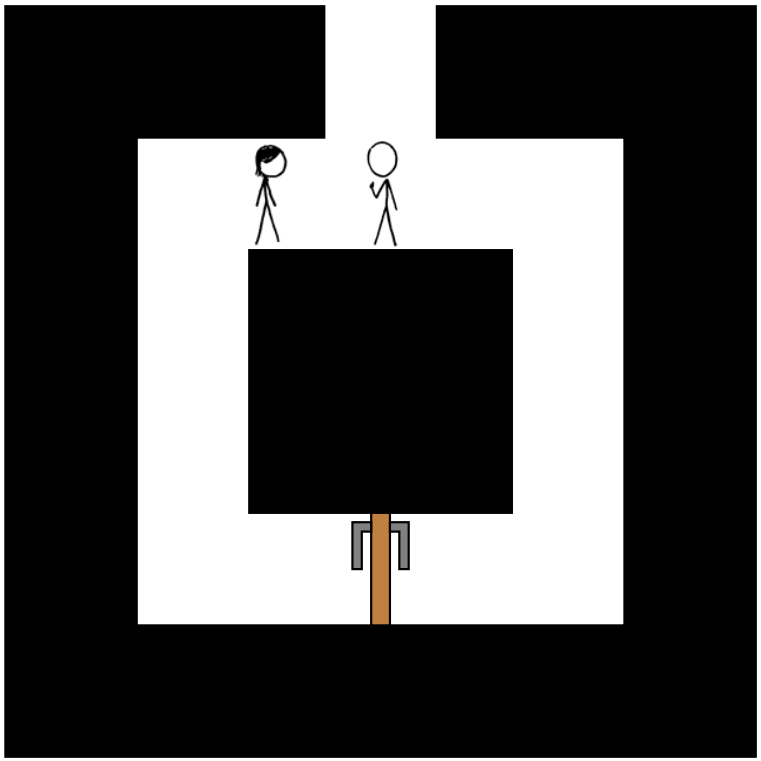
Zero knowledge

- A “zero-knowledge” scheme is intended to convince someone that you have certain knowledge, without giving the knowledge to this someone
- If Peggy exits from the correct path enough times, Victor is convinced that Peggy can open the door
- But Victor has not learnt how to open the door



Zero knowledge

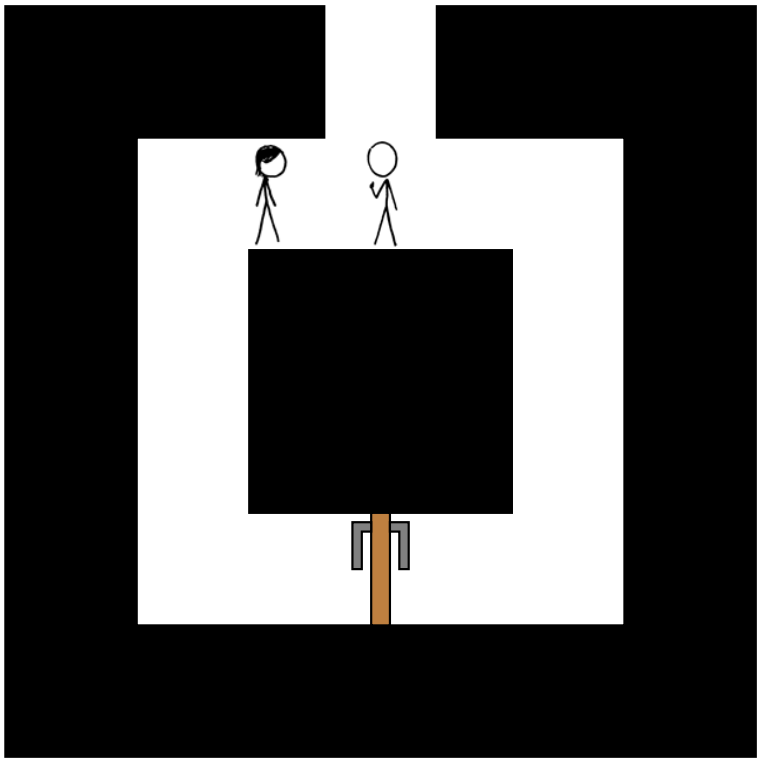
- A “zero-knowledge” scheme is intended to convince someone that you have certain knowledge, without giving the knowledge to this someone
- If Peggy exits from the correct path enough times, Victor is convinced that Peggy can open the door
- But Victor has not learnt how to open the door
- And a third person cannot be convinced that Peggy can open the door



Zero knowledge

- A “zero-knowledge” scheme is intended to convince someone that you have certain knowledge, without giving the knowledge to this someone

- A third person cannot be convinced that Peggy can open the door
- It doesn’t even help to have a video of everything that Victor sees
- This is known as an interactive protocol



Zero-knowledge cut-and-choose

- The classic cut-and-choose
  1. Peggy cuts the cake in half
  2. Victor chooses one of the halfs for himself
  3. Peggy takes the remainders

- The cave&door protocol works because Peggy cannot guess which path Victor will ask for, so she only has 50% chance of fooling him

- Repeating 10 times reduces Peggy's chance to fool Victor to $2^{-10} \approx 1/1000$

Zero-knowledge example, $\sqrt{t}$ mod n

- Peggy claims to know a square root s of t mod $n = pq$, where p and q are large primes (efficiently solving for s is equivalent to efficiently factoring n)
    1. Peggy uses a random number r, computes $x = r^2$ and sends x to Victor
    2. Victor asks either for the square root of x, or the square root of xt
    3. If Victor asked for the square root of x, Peggy lets $y = r$.
       If he asked for the square root of xt, Peggy lets $y = rs$.
       She now returns y to Victor
    4. Victor checks that $y^2 = x$ or $y^2 = xt$ depending on what he asked for
    5. If Peggy can do this enough times (she must use different random numbers r each time), Victor will conclude that Peggy knows s, the square root of t

Zero-knowledge example, $\sqrt{t}$ mod n

- Peggy claims to know a square root s of t mod $n = pq$, where p and q are large primes (efficiently solving for s is equivalent to efficiently factoring n)
    1. Peggy uses a random number r, computes $x = r^2$ and sends x to Victor
    2. Victor asks either for the square root of x, or the square root of xt
    3. If Victor asked for the square root of x, Peggy lets $y = r$.
       If he asked for the square root of xt, Peggy lets $y = rs$.
       She now returns y to Victor
    4. Victor checks that $y^2 = x$ or $y^2 = xt$ depending on what he asked for
    5. If Peggy can do this enough times (she must use different random numbers r each time), Victor will conclude that Peggy knows s, the square root of t

Zero-knowledge example, $\sqrt{t}$ mod n, cheating

- Peggy claims to know a square root s of t mod $n = pq$, where p and q are large primes (efficiently solving for s is equivalent to efficiently factoring n)
  1. Peggy uses a random number r, computes $x = r^2$ and sends x to Victor
  2. Victor asks either for the square root of x, or the square root of xt
  3. If Victor asked for the square root of x, Peggy lets $y = r$. If he asked for the square root of xt, Peggy doesn't know s and must guess $y = rs$. She now returns y to Victor
  4. Victor checks that $y^2 = x$ or $y^2 = xt$ depending on what he asked for
  5. If Peggy can do this enough times (she must use different random numbers r each time), Victor will conclude that Peggy knows s, the square root of t

Zero-knowledge example, $\sqrt{t}$ mod n, cheating

- Peggy claims to know a square root s of t mod $n = pq$, where p and q are large primes (efficiently solving for s is equivalent to efficiently factoring n)
  1. Peggy uses a random number r, computes $x = r^2 t^{-1}$ and sends x to Victor
  2. Victor asks either for the square root of x, or the square root of xt
  3. If Victor asked for the square root of x, Peggy doesn't know s and must guess $y = rs^{-1}$. If he asked for the square root of xt, Peggy lets $y = r$. She now returns y to Victor
  4. Victor checks that $y^2 = x$ or $y^2 = r^2 = xt$ depending on what he asked for
  5. If Peggy can do this enough times (she must use different random numbers r each time), Victor will conclude that Peggy knows s, the square root of t

The Feige-Fiat-Shamir identification scheme

- Peggy claims to know square roots s_i of t_i mod $n = pq$, where p and q are large primes
  1. Peggy uses a random number r, computes $x = r^2$ and sends x to Victor
  2. Victor asks for the square root of $xt_1^{b_1} t_2^{b_2} \ldots t_n^{b_n}$, for b_i in $\{0, 1\}$
  3. Peggy computes $y = rs_1^{b_1} s_2^{b_2} \ldots s_n^{b_n}$ and sends y to Victor
  4. Victor checks that $y^2 = xt_1^{b_1} t_2^{b_2} \ldots t_n^{b_n}$
  5. If Peggy can do this enough times (she must use different random numbers r each time), Victor will conclude that Peggy knows s_i, the square roots of t_i

Numbers for Feige-Fiat-Shamir identification

- Peggy is a customer at Victor's bank. She wants to identify herself electronically
- To generate numbers to be used in the scheme, Peggy and Victor agree on the following
  1. Peggy uses her name and personnummer, and a publicly known hash function, she also chooses (secret) p and q and gives Victor $n = pq$ (as in RSA)
  2. She concatenates a counter to the end of the data, and hashes the resulting number, for a few values of the counter. Victor does the same
  3. Roughly half the numbers have square roots mod p, and half the numbers mod q. This means that rougly a quarter of the numbers have square roots mod n. Knowing p and q, Peggy can efficiently calculate these
- They can now use Feige-Fiat-Shamir identification
- An eavesdropper on a card transaction will learn nothing about the square roots that Peggy uses

Zero-knowledge example, $\sqrt{t}$ mod n

- Peggy claims to know a square root s of t mod $n = pq$, where p and q are large primes
  1. Peggy uses a random number r, computes $x = r^2$ and sends x to Victor
  2. Victor asks either for the square root of x, or the square root of xt
  3. If Victor asked for the square root of x, Peggy lets $y = r$.
     If he asked for the square root of xt, Peggy lets $y = rs$.
     She now returns y to Victor
  4. Victor checks that $y^2 = x$ or $y^2 = xt$ depending on what he asked for
  5. If Peggy can do this enough times (she must use different random numbers r each time), Victor will conclude that Peggy knows s, the square root of t

Zero-knowledge, formal setup

- Peggy knows the solution to a hard mathematical problem, such as the pre-image to a one-way function output
    1. Peggy uses a random number to transform the problem into a different, but equally hard mathematical problem. She commits to the solution of the new instance (using bit commitment) and reveals it to Victor
    2. Victor asks Peggy to either a) prove that the two problems are isomorphic, or b) provide the solution of the second problem
    3. If Peggy can do this enough times, Victor will conclude that Peggy can solve the original problem

Zero-knowledge, formal setup

- Peggy knows the solution to a hard mathematical problem, such as the pre-image to a one-way function output
  1. Peggy uses a random number to transform the problem into a different, but equally hard mathematical problem. She commits to the solution of the new instance (using bit commitment) and reveals it to Victor
  2. Victor asks Peggy to either a) prove that the two problems are isomorphic, or b) provide the solution of the second problem
  3. If Peggy can do this enough times, Victor will conclude that Peggy can solve the original problem

- Peggy is known as the prover, while Victor is known as the verifier

Zero-knowledge, formal setup

- Peggy knows the solution to a hard mathematical problem, such as the pre-image to a one-way function output
  1. Peggy uses a random number to transform the problem into a different, but equally hard mathematical problem. She commits to the solution of the new instance (using bit commitment) and reveals it to Victor
  2. Victor asks Peggy to either a) prove that the two problems are isomorphic, or b) provide the solution of the second problem
  3. If Peggy can do this enough times, Victor will conclude that Peggy can solve the original problem

- Peggy is known as the prover, while Victor is known as the verifier
- It is important that Peggy's and Victor's choices is random (to the other participant), otherwise cheating is possible

Digital signatures are not Zero-knowledge

- Either Alice chooses the message, and gives Bob message and signature. In this case, there is no (random) challenge from Bob.
    - Bob is not convinced. Messages have no structure here, so how does he know Alice did not select the signature first and used the public key to generate the message?
- Or Bob chooses the message and asks Alice to sign, but then, there is no (random) commitment by Alice
    - Here the danger is that Bob could carefully select messages to extract information on the secret

- In proper zero knowledge protocols, new randomness is added at each instance, so that no information on the secret is revealed to the verifier

Zero-knowledge, formal setup

- Peggy knows the solution to a hard mathematical problem, such as the pre-image to a one-way function output
  1. Peggy uses a random number to transform the problem into a different, but equally hard mathematical problem. She commits to the solution of the new instance (using bit commitment) and reveals it to Victor
  2. Victor asks Peggy to either a) prove that the two problems are isomorphic, or b) provide the solution of the second problem
  3. If Peggy can do this enough times, Victor will conclude that Peggy can solve the original problem

- Peggy is known as the prover, while Victor is known as the verifier
- It is important that Peggy's and Victor's choices is random (to the other participant), otherwise cheating is possible

Secret sharing

- Imagine you make a lot of money on the latest Iphone app you wrote

- You live happily, but make plans for the future: at some point your (less trustworthy) children are going to inherit
- You want to force them to cooperate to open the safe
- This is easily done by giving them each one number in the three-number combination
- If you happen to have, say, seven children, or want to give them less information (not making exhaustive search easier), you need to be more sophisticated

Secret splitting

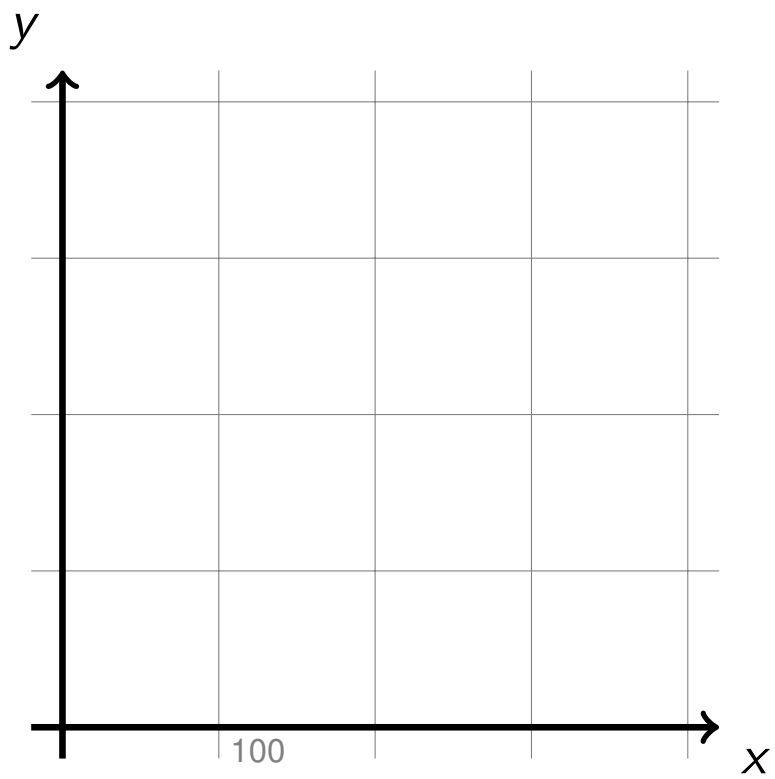
- You want to split a secret message m between Alice and Bob so that they need to cooperate to read it
- The simplest way to do this is to take a random r, give r to Alice and $m - r$ to Bob
- They must now add their numbers to find m

Secret splitting

- You want to split a secret message m between Alice and Bob so that they need to cooperate to read it
- The simplest way to do this is to take a random r, give r to Alice and $m - r$ to Bob
- If you want to do this among n people, give $n - 1$ of them random numbers, and the last one $m - r_1 - r_2 - ... - r_{n-1}$
- They must add all their shares to find m

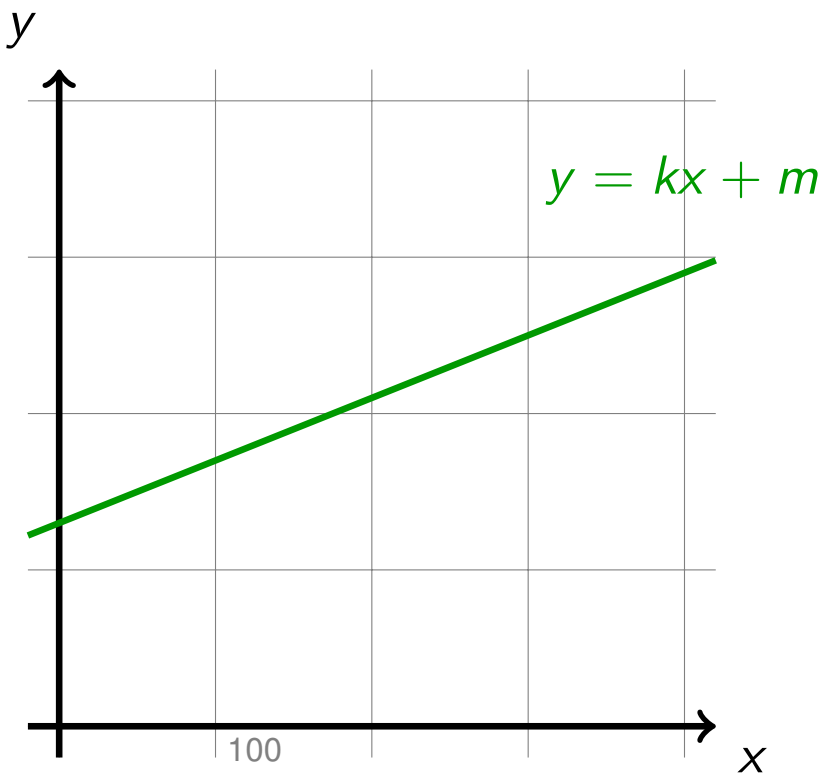
More sophisticated secret sharing

- We want to share the secret among n persons so that any pair of them can retrieve m



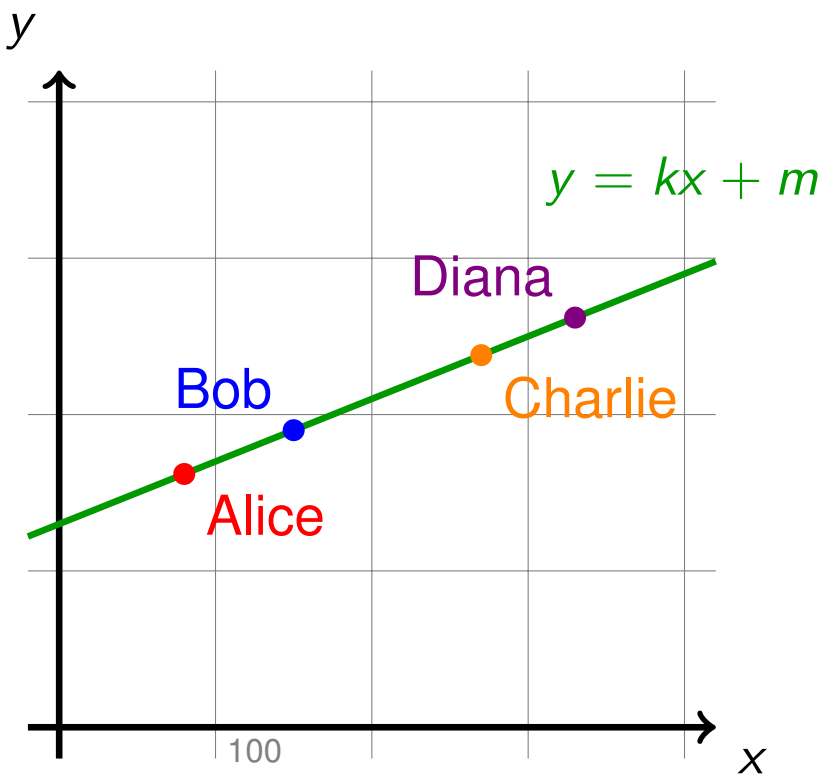
More sophisticated secret sharing

- We want to share the secret among n persons so that any pair of them can retrieve m
- Create the line $(x, kx + m)$, that passes the point $(0, m)$



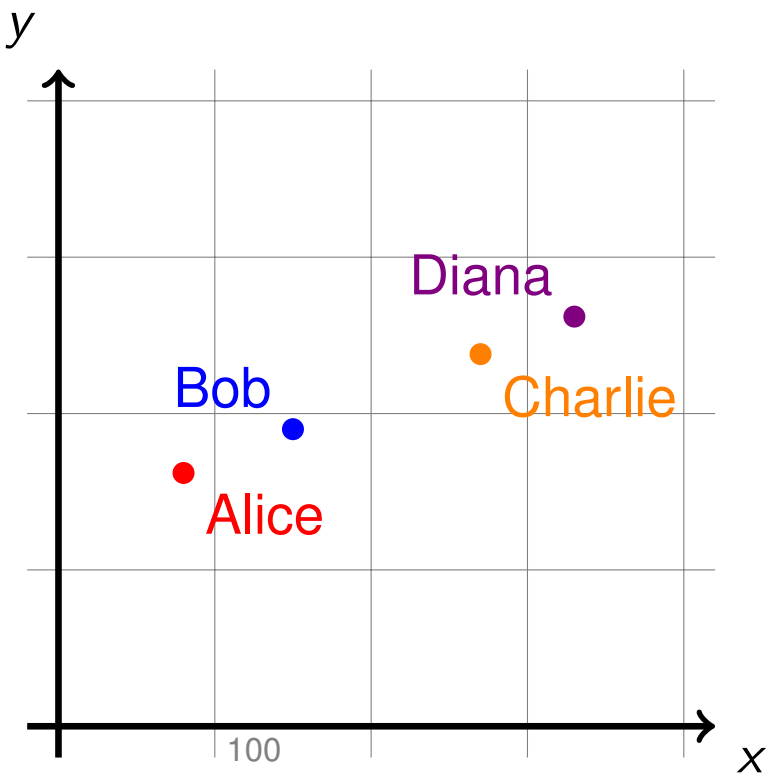
More sophisticated secret sharing

- We want to share the secret among n persons so that any pair of them can retrieve m
- Create the line $(x, kx + m)$, that passes the point $(0, m)$
- Give each of the n persons one point on the line



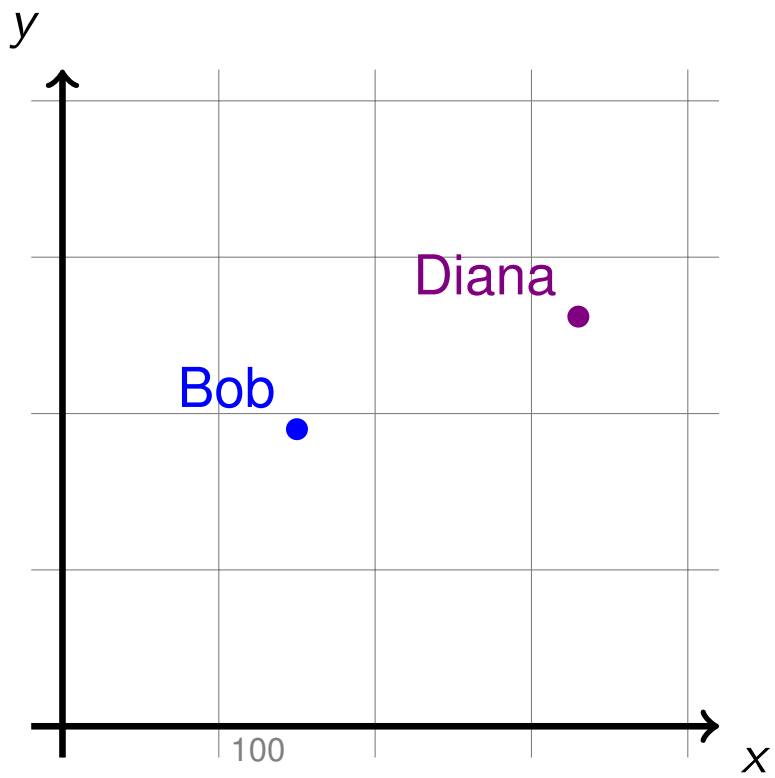
More sophisticated secret sharing

- We want to share the secret among n persons so that any pair of them can retrieve m
- Create the line $(x, kx + m)$, that passes the point $(0, m)$
- Give each of the n persons one point on the line
- They need two points to reconstruct the line, to find $(0, m)$



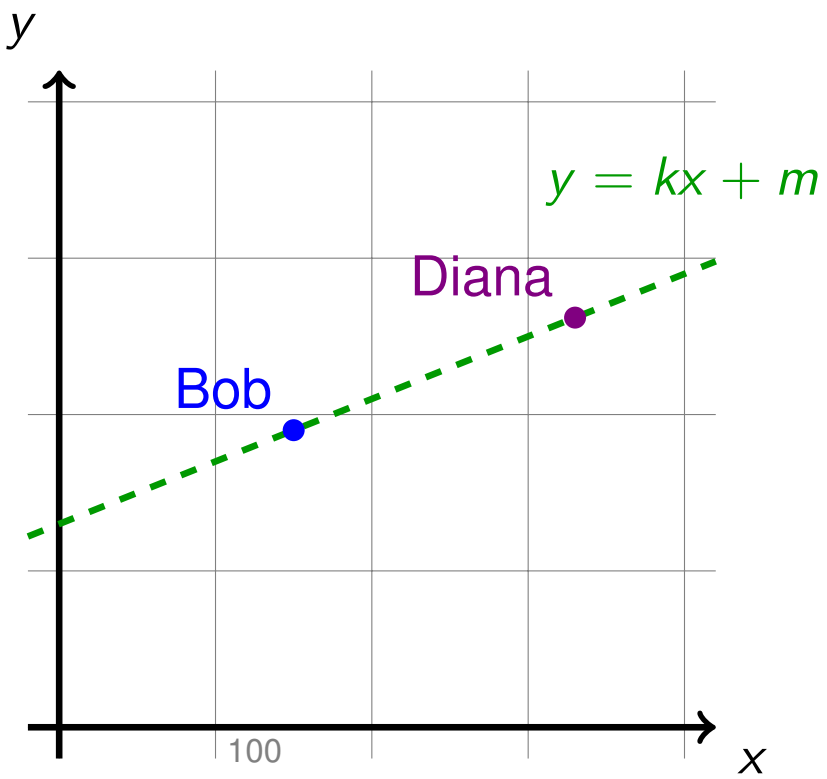
More sophisticated secret sharing

- We want to share the secret among n persons so that any pair of them can retrieve m
- Create the line $(x, kx + m)$, that passes the point $(0, m)$
- Give each of the n persons one point on the line
- They need two points to reconstruct the line, to find $(0, m)$



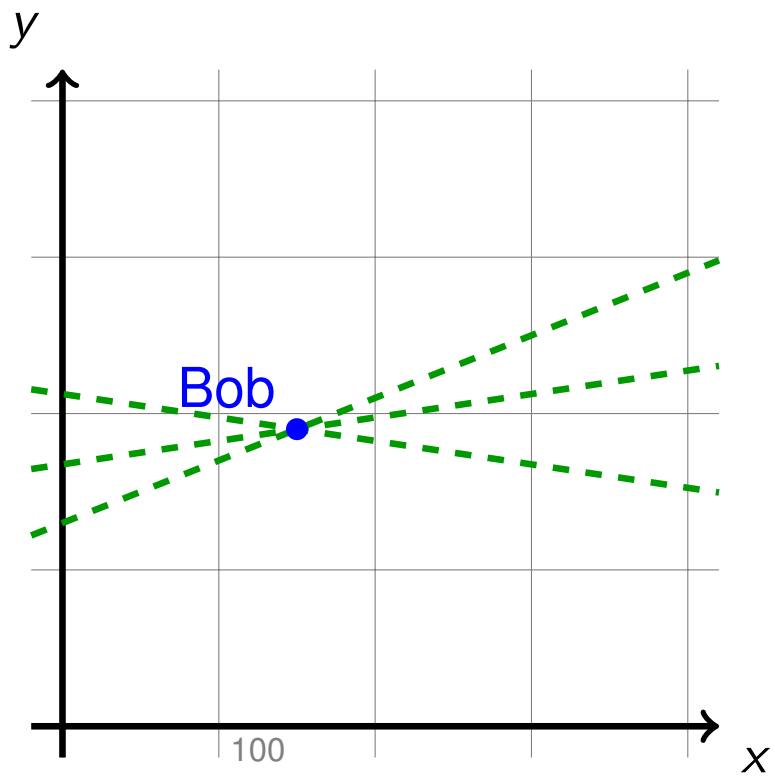
More sophisticated secret sharing

- We want to share the secret among n persons so that any pair of them can retrieve m
- Create the line $(x, kx + m)$, that passes the point $(0, m)$
- Give each of the n persons one point on the line
- They need two points to reconstruct the line, to find $(0, m)$



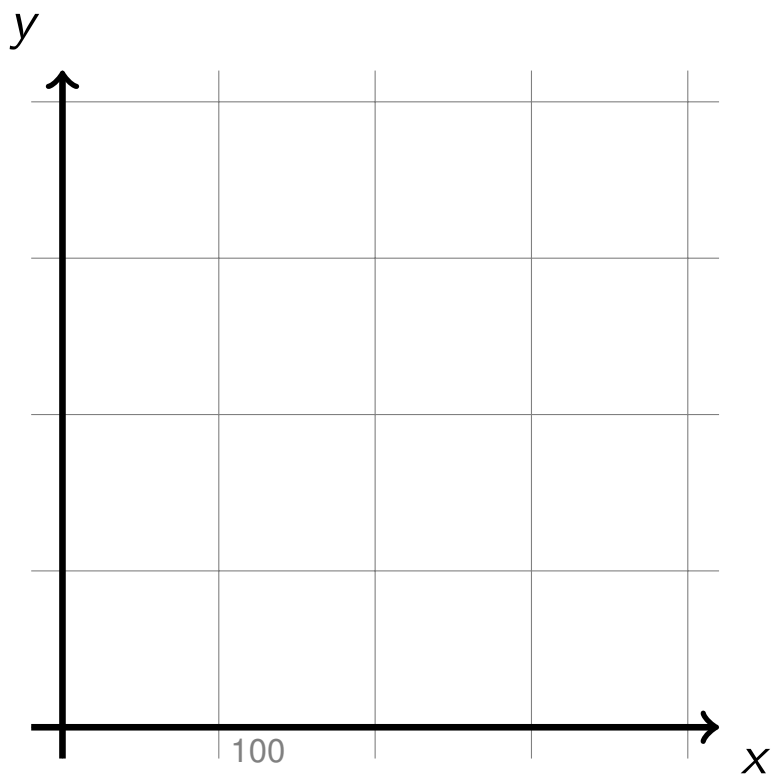
More sophisticated secret sharing

- We want to share the secret among n persons so that any pair of them can retrieve m
- Create the line $(x, kx + m)$, that passes the point $(0, m)$
- Give each of the n persons one point on the line
- They need two points to reconstruct the line, to find $(0, m)$
- One point is not enough



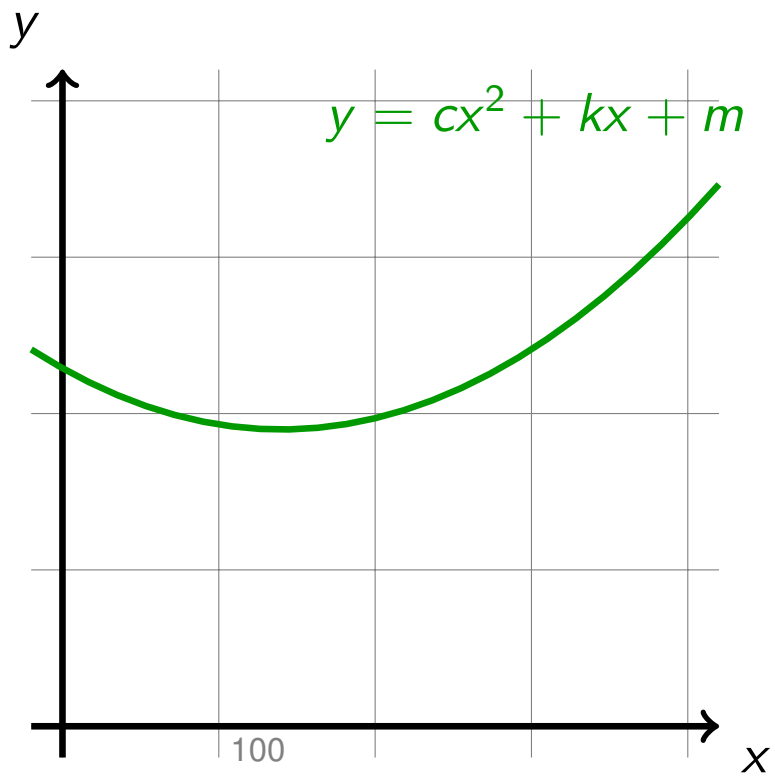
More sophisticated secret sharing

- We want to share the secret among n persons so that any three of them can retrieve m



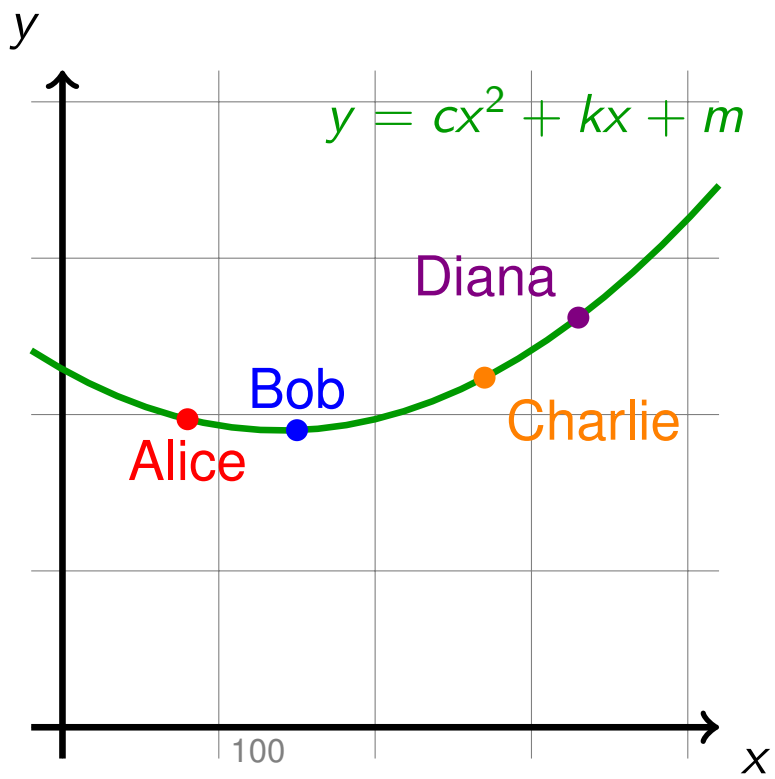
More sophisticated secret sharing

- We want to share the secret among n persons so that any three of them can retrieve m
- Create a second-degree polynomial that passes the point $(0, m)$



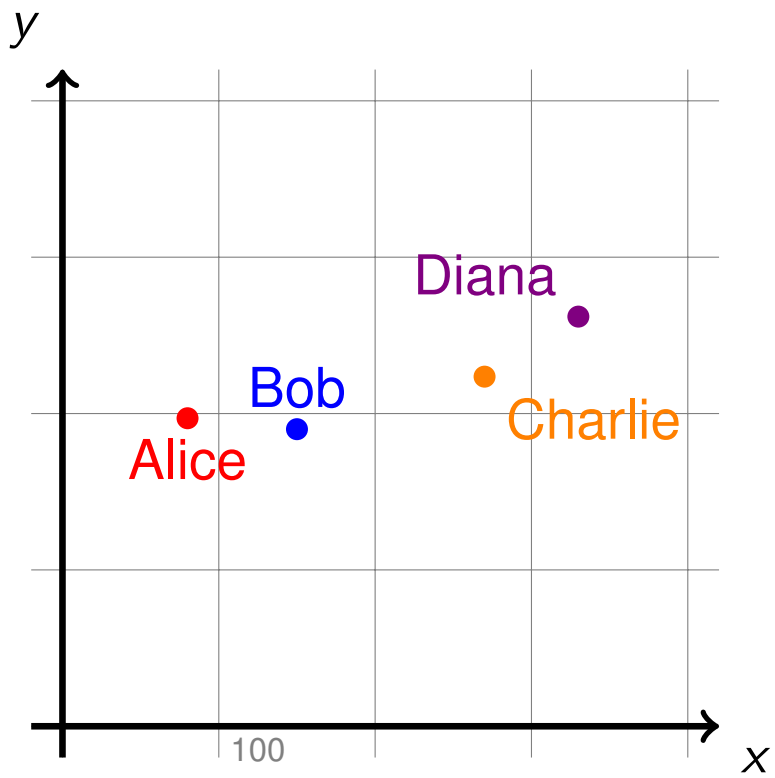
More sophisticated secret sharing

- We want to share the secret among n persons so that any <u>three</u> of them can retrieve m
- Create a second-degree polynomial that passes the point $(0, m)$
- Give each of the n persons one point on the line



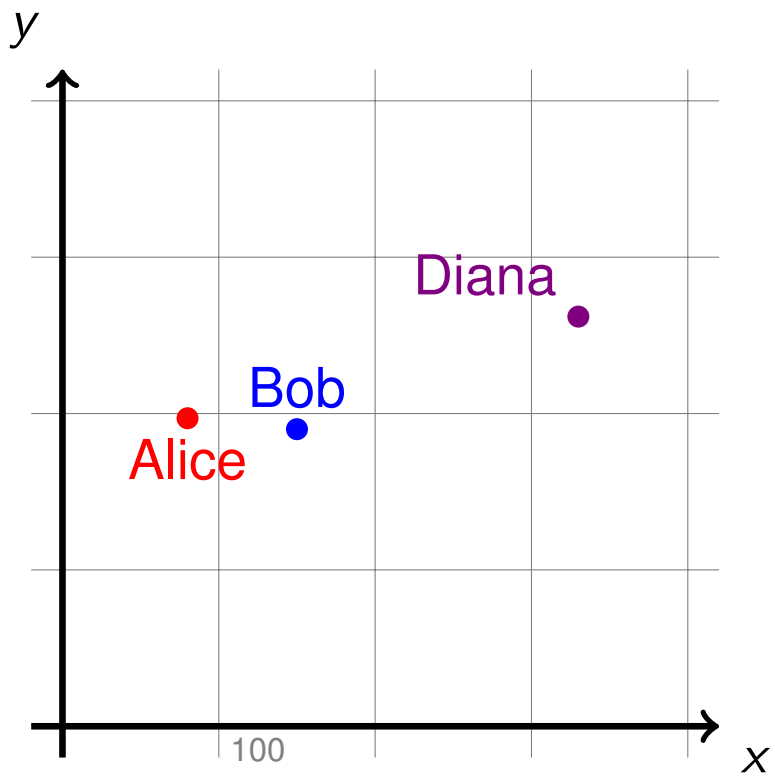
More sophisticated secret sharing

- We want to share the secret among n persons so that any three of them can retrieve m
- Create a second-degree polynomial that passes the point $(0, m)$
- Give each of the n persons one point on the line
- They need three points to reconstruct the line, to find $(0, m)$



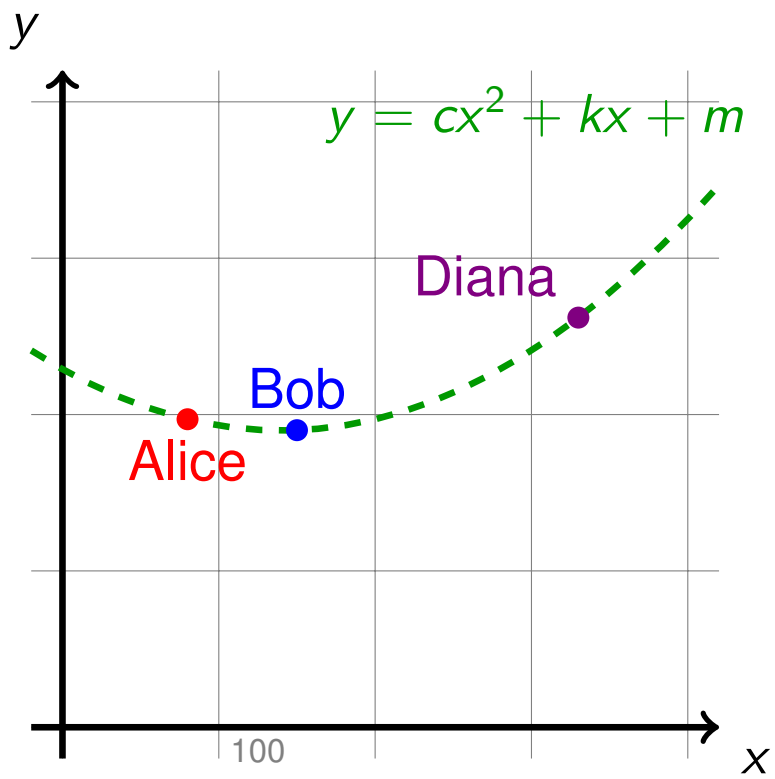
More sophisticated secret sharing

- We want to share the secret among n persons so that any three of them can retrieve m
- Create a second-degree polynomial that passes the point $(0, m)$
- Give each of the n persons one point on the line
- They need three points to reconstruct the line, to find $(0, m)$



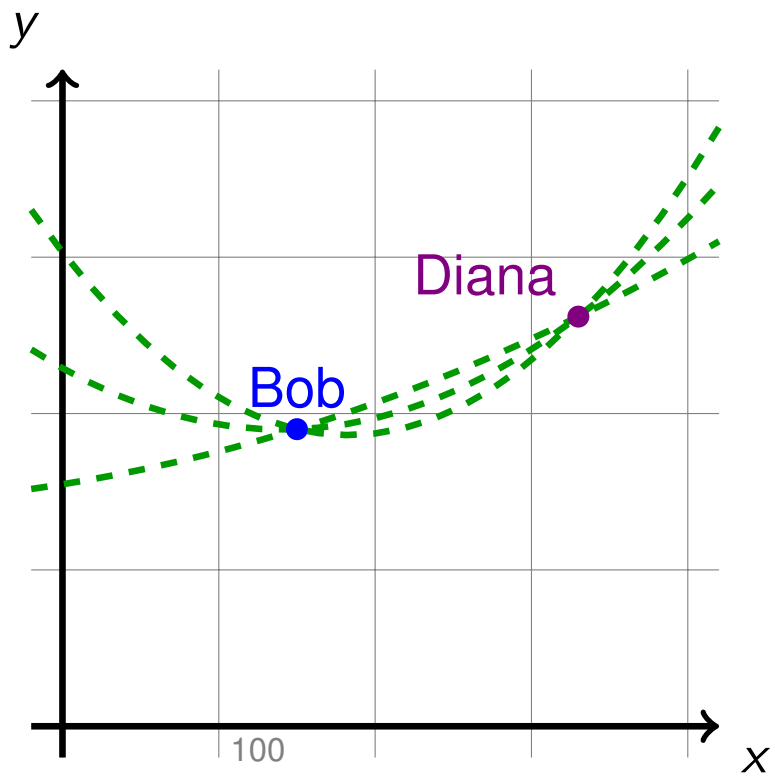
More sophisticated secret sharing

- We want to share the secret among n persons so that any three of them can retrieve m
- Create a second-degree polynomial that passes the point $(0, m)$
- Give each of the n persons one point on the line
- They need three points to reconstruct the line, to find $(0, m)$



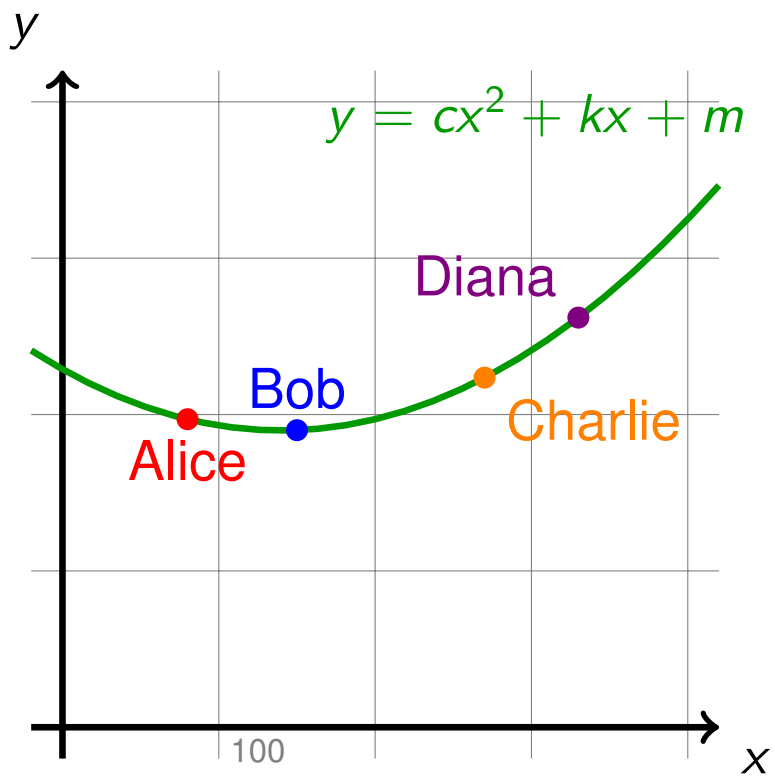
More sophisticated secret sharing

- We want to share the secret among n persons so that any three of them can retrieve m
- Create a second-degree polynomial that passes the point $(0, m)$
- Give each of the n persons one point on the line
- They need three points to reconstruct the line, to find $(0, m)$
- Two points are not enough



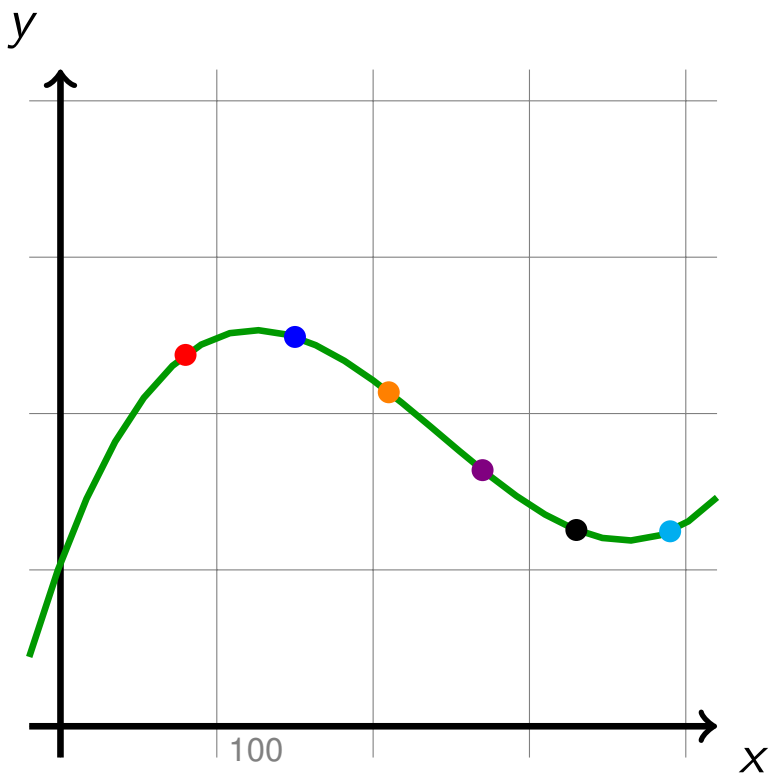
Treshold scheme (a $(3, n)$ treshold scheme)

- We want to share the secret among n persons so that any three of them can retrieve m
- Create a second-degree polynomial that passes the point $(0, m)$
- Give each of the n persons one point on the line
- They need three points to reconstruct the line, to find $(0, m)$
- Two points are not enough



A (t, n) treshold scheme

- We want to share the secret among n persons so that any t of them can retrieve m
- Create a $t-1$'st-degree polynomial that passes the point $(0, m)$
- Give each of the n persons one point on the line
- They need t points to reconstruct the line, to find $(0, m)$
- $t-1$ points are not enough



Shamir (t, n) treshold scheme

- We want to share the secret among n persons so that any t of them can retrieve m
- Choose a prime p larger than n and larger than m, and create a (random) polynomial

$$s(x) = m + s_1 x + s_2 x^2 + ... + s_{t-1} x^{t-1} \bmod p$$

- Give each of the n persons one unique point on the line
- They now need t points to reconstruct the curve, because this will give t equations for the t unknowns m, s_1, s_2, ..., s_{t-1}

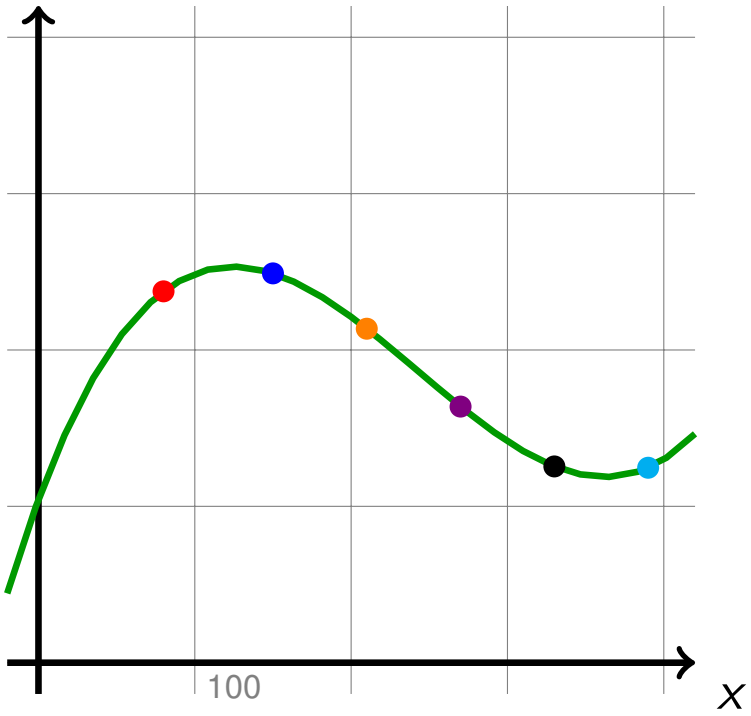
Lagrange interpolation

- Given t points, let

$$l_k(x) = \prod_{\substack{i=1 \\ i \neq k}}^{t} \frac{x - x_i}{x_k - x_i} \bmod p$$

- The Lagrange interpolation polynomial is

$$p(x) = \sum_{k=1}^{t} y_k l_k(x)$$



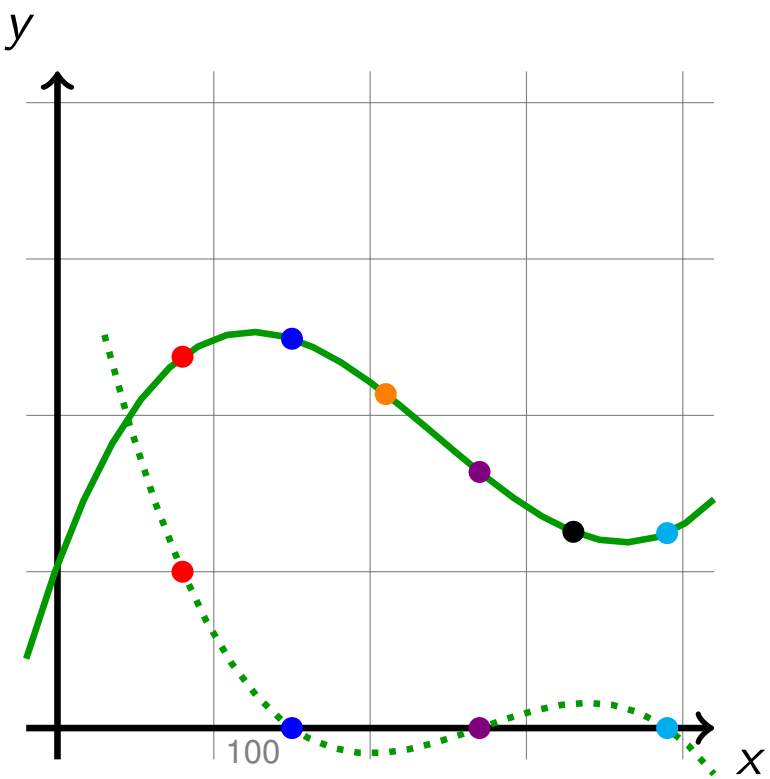
Lagrange interpolation

- Given $x_1 = 80$, $x_2 = 150$, $x_3 = 270$, and $x_4 = 390$, let

$$l_1(x) = \frac{x - x_2}{x_1 - x_2} \cdot \frac{x - x_3}{x_1 - x_3} \cdot \frac{x - x_4}{x_1 - x_4} \bmod p$$

- The Lagrange interpolation polynomial is

$$p(x) = \sum_{k=1}^{t} y_k l_k(x)$$



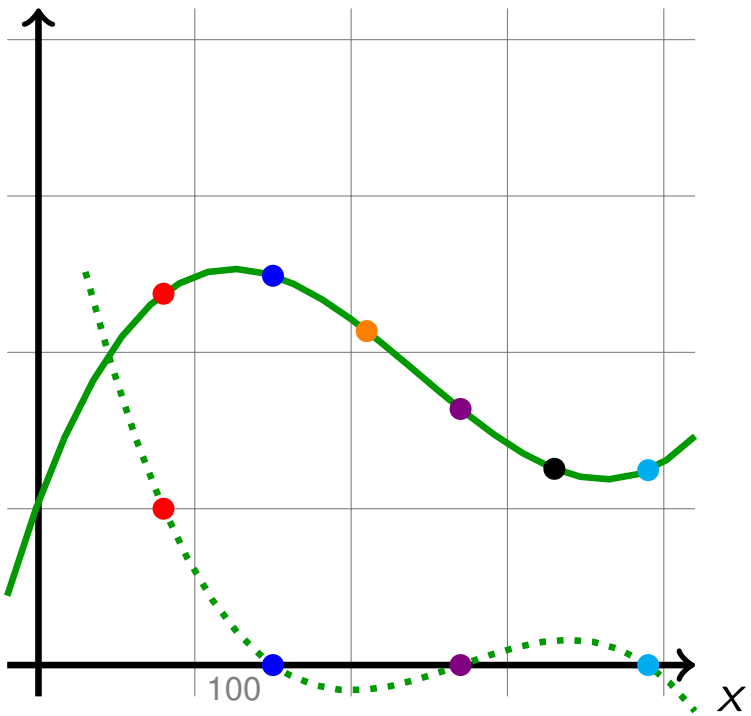
Lagrange interpolation

- Given $x_1 = 80$, $x_2 = 150$, $x_3 = 270$, and $x_4 = 390$, let

$$l_k(x) = \prod_{\substack{i=1 \\ i \neq k}}^{t} \frac{x - x_i}{x_k - x_i} \bmod p$$

- The Lagrange interpolation polynomial is

$$p(x) = \sum_{k=1}^{t} y_k l_k(x)$$



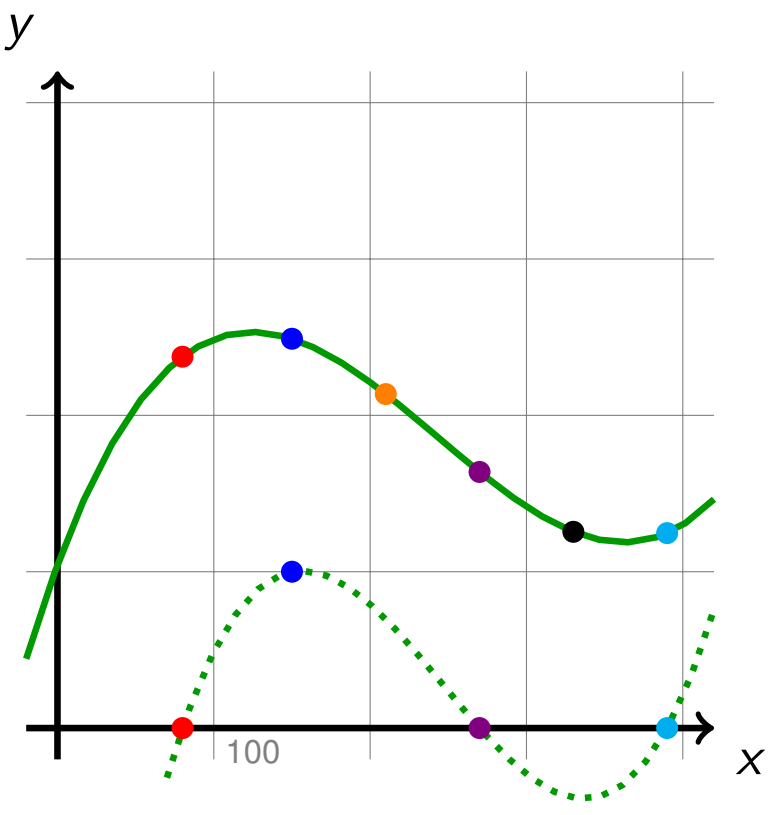
Lagrange interpolation

- Given $x_1 = 80$, $x_2 = 150$, $x_3 = 270$, and $x_4 = 390$, let

$$l_2(x) = \frac{x - x_1}{x_2 - x_1} \cdot \frac{x - x_3}{x_2 - x_3} \cdot \frac{x - x_4}{x_2 - x_4} \bmod p$$

- The Lagrange interpolation polynomial is

$$p(x) = \sum_{k=1}^{t} y_k l_k(x)$$



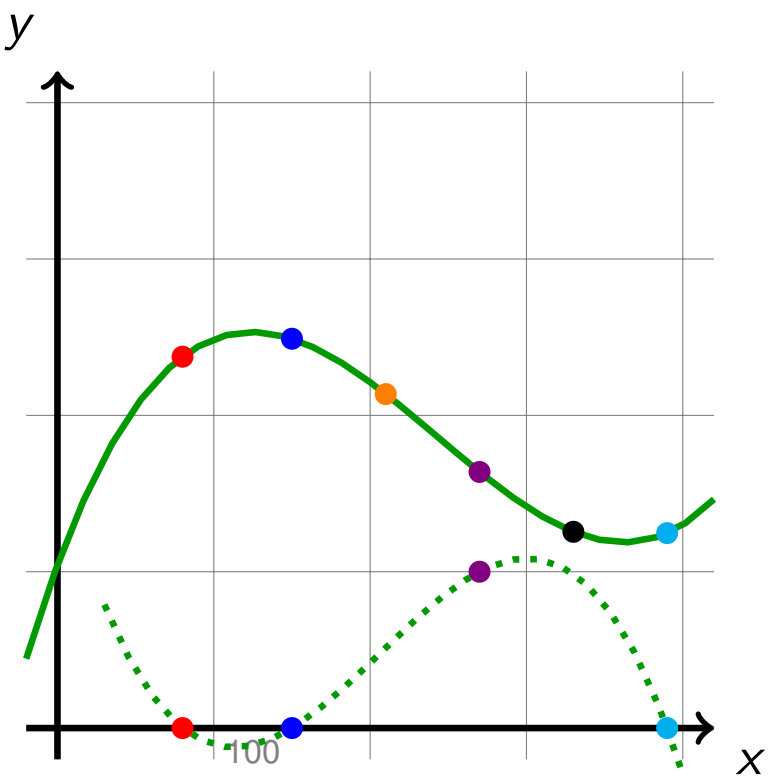
Lagrange interpolation

- Given $x_1 = 80$, $x_2 = 150$, $x_3 = 270$, and $x_4 = 390$, let

$$l_3(x) = \frac{x - x_1}{x_3 - x_1} \cdot \frac{x - x_2}{x_3 - x_2} \cdot \frac{x - x_4}{x_3 - x_4} \bmod p$$

- The Lagrange interpolation polynomial is

$$p(x) = \sum_{k=1}^{t} y_k l_k(x)$$



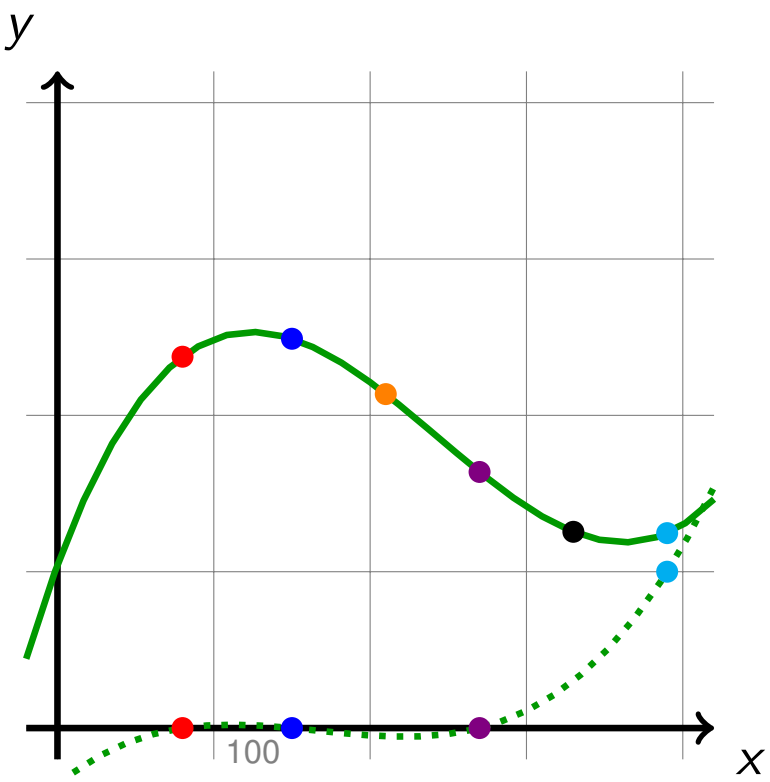
Lagrange interpolation

- Given $x_1 = 80$, $x_2 = 150$, $x_3 = 270$, and $x_4 = 390$, let

$$l_4(x) = \frac{x - x_1}{x_4 - x_1} \cdot \frac{x - x_2}{x_4 - x_2} \cdot \frac{x - x_3}{x_4 - x_3} \bmod p$$

- The Lagrange interpolation polynomial is

$$p(x) = \sum_{k=1}^{t} y_k l_k(x)$$



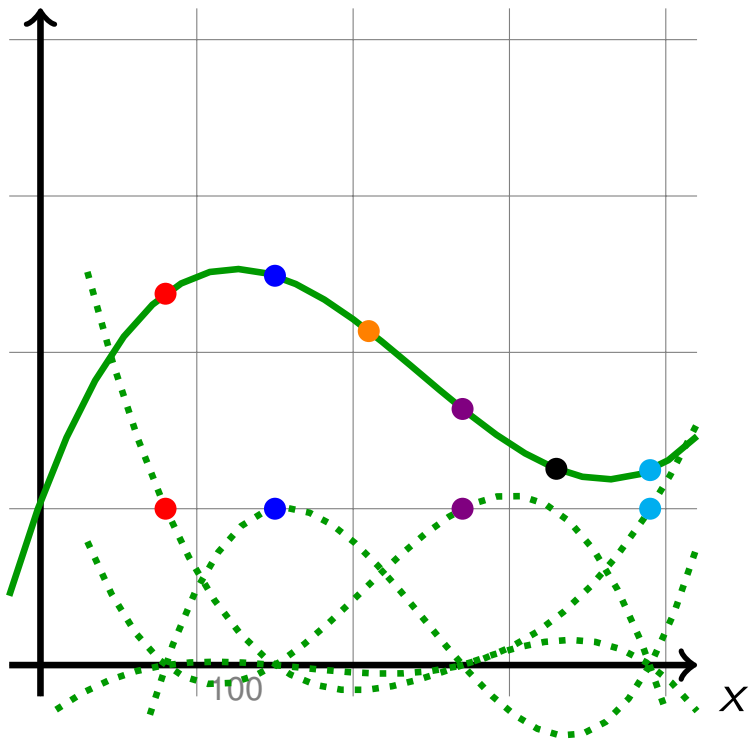
Lagrange interpolation

- Given $x_1 = 80$, $x_2 = 150$, $x_3 = 270$, and $x_4 = 390$, let

$$l_k(x) = \prod_{\substack{i=1 \\ i \neq k}}^{t} \frac{x - x_i}{x_k - x_i} \bmod p$$

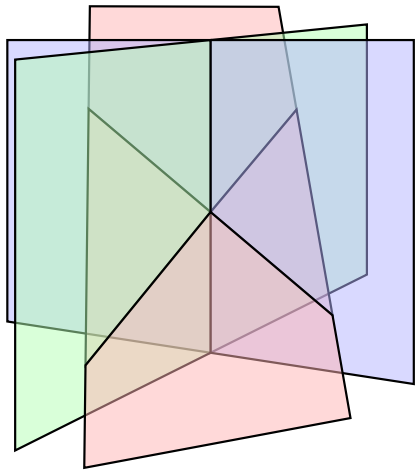
- The Lagrange interpolation polynomial is

$$p(x) = \sum_{k=1}^{t} y_k l_k(x)$$



Blakley secret sharing

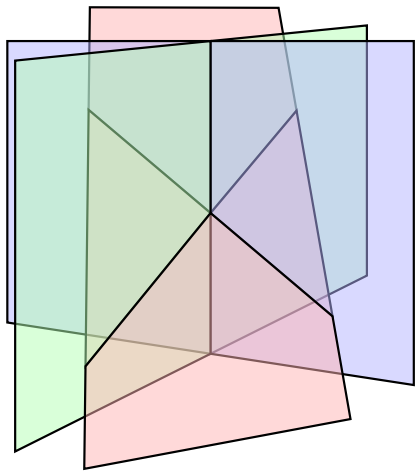
- Another way to generalize the linear function we started with is to go to higher dimension
- Use intersecting hyperplanes in t dimensions instead
- Almost the same as Shamir's polynomial secret sharing



Blakley secret sharing

- If x_0 is the secret, choose random y_0, z_0 (mod p)
- For each share, choose random a_i and b_i and let $c_i = a_i x_0 + b_i y_0 - z_0$
- Each share is a Hyperplane

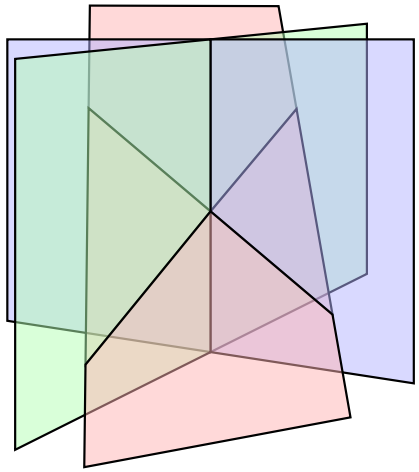
$$\begin{cases} a_1 x + b_1 y - z = c_1 & \leftarrow \\ a_2 x + b_2 y - z = c_2 & \leftarrow \\ a_3 x + b_3 y - z = c_3 & \leftarrow \end{cases}$$



Blakley secret sharing

- Each share is a Hyperplane

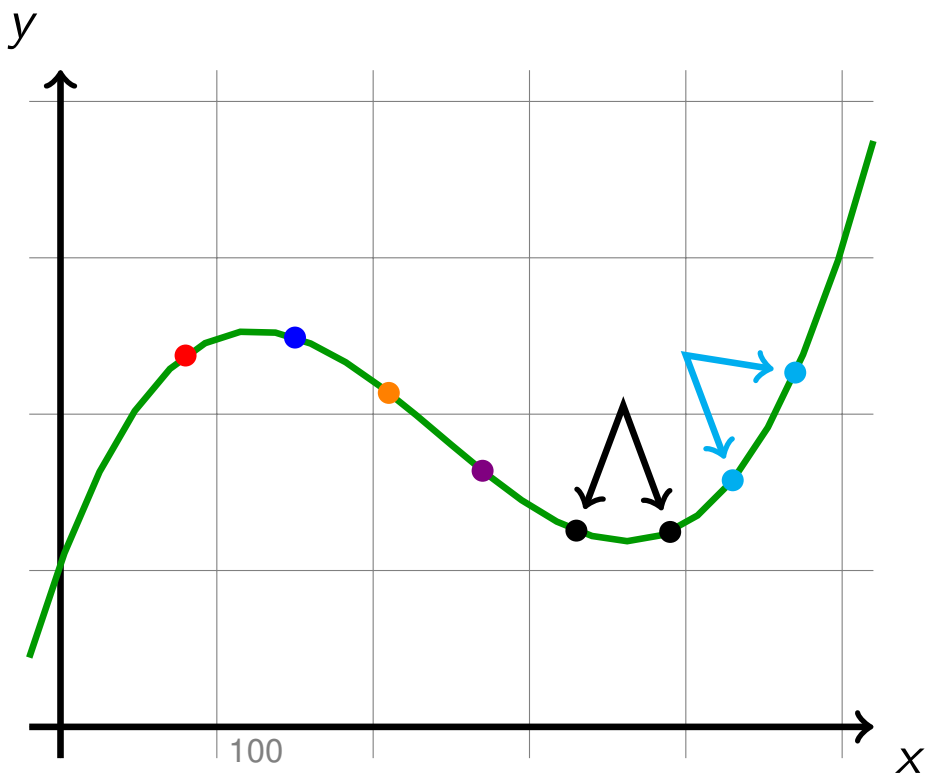
$$\begin{cases} a_1x + b_1y - z = c_1 & \leftarrow \\ a_2x + b_2y - z = c_2 & \leftarrow \\ a_3x + b_3y - z = c_3 & \leftarrow \end{cases}$$



- Three unknowns: need three equations to solve for the unknown point (x_0, y_0, z_0)
- To solve the system, the equations need to be linearly independent

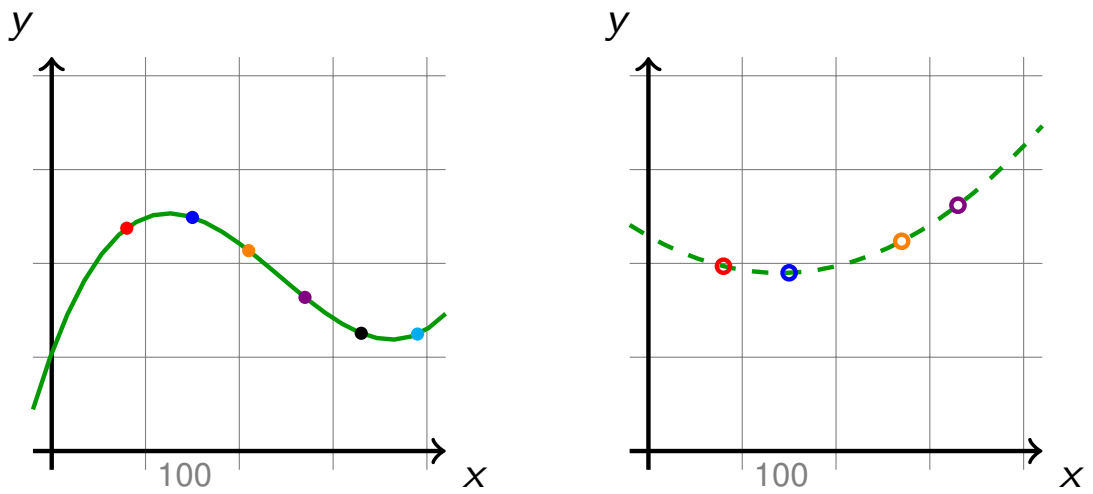
Uneven sharing

- Suppose you need two chartered accountants or four ordinary clerks from the economy department or one chartered accountant and two clerks to sign a document
  - Create a $(4, n)$ threshold scheme
  - Give each chartered accountant two shares



Uneven sharing

- Suppose you need at least four employees from company A and three from company B to retrieve a secret
  - A $(7, n)$ threshold scheme won't work since 7 people from company A could cooperate
  - Create a $(4, n)$ threshold scheme for company A, and a $(3, n)$ scheme for company B
  - Use the initial $m = m_A + m_B \bmod p$ scheme we started with



Coin tossing over the phone

- Alice and Bob want to toss coins over telephone to decide who gets something they both want. To set up the system, Alice chooses p and q prime (congruent to 3 mod 4) and sends Bob $n = pq$
- Bob chooses a random $x < n/2$ and sends Alice $u = x^2 \bmod n$
- Alice computes the four roots (easy if you know p and q, hard if not), at random chooses v as one of the two $< n/2$ and sends v to Bob
- If $v = \pm x \bmod n$, Bob sends "OK, you win" to Alice. If $v \neq x$, Bob sends "I win" and x to Alice, to prove he knew it (he can now factor n; Alice should check that $x^2 = u \bmod n$)

Poker over the telephone, discrete log

- Alice and Bob want to play Poker over the telephone; they agree on a large prime p, represent the cards using 52 numbers, and choose secrets α and β so that $\gcd(\alpha, p-1) = 1 = \gcd(\beta, p-1)$
  - Bob encrypts each card by raising it to the power β and sends the set to Alice
  - Alice chooses five cards for herself and encrypts them with α, and five cards for Bob. She sends all ten cards to Bob
  - Bob decrypts the cards by raising it to the power β^{-1} mod $p-1$, and sends back Alice's cards
  - Alice decrypts her cards
- Both keep the encrypted version of the other's cards
- Discarded cards are encrypted and sent to the other
- To end the game, they claim their hands and prove them by revealing their keys

Poker over the telephone, discrete log

- Alice and Bob want to play Poker over the telephone; they agree on a large prime p, represent the cards using 52 numbers, and choose secrets α and β so that $\gcd(\alpha, p-1) = 1 = \gcd(\beta, p-1)$
- Encryption as $c = m^k \bmod p$ and decryption as $m = c^{1/k} \bmod p$
- This is a system with private keys only
- Security depends on discrete log complexity

Poker over the telephone, discrete log

- Alice and Bob want to play Poker over the telephone; they agree on a large prime p, represent the cards using 52 numbers, and choose secrets α and β so that $\gcd(\alpha, p-1) = 1 = \gcd(\beta, p-1)$
- Encryption as $c = m^k \bmod p$ and decryption as $m = c^{1/k} \bmod p$
- This is a system with private keys only
- Security depends on discrete log complexity
- But in this system, you can cheat

Discrete log poker, cheating

- A quadratic residue mod p is a number that is a square mod p

$$r^{(p-1)/2} = \begin{cases} +1 \bmod p & \text{if } r \text{ is a quadratic residue} \\ -1 \bmod p & \text{otherwise} \end{cases}$$

- And since α and β are odd (no nontrivial factors common with $p-1$), this does not change under encryption

$$c^{(p-1)/2} = (m^{\alpha})^{(p-1)/2} = (m^{(p-1)/2})^{\alpha} = m^{(p-1)/2} \mod p$$

- There is information still available in the ciphertext

Poker over the telephone, with RSA

- Alice and Bob want to play Poker over the telephone; they both set up for RSA
  - Alice encrypts each card with her public key and sends the set to Bob
  - Bob chooses five cards for himself and encrypts them with his public key, and five cards for Alice. He sends all ten cards to Alice
  - Alice decrypts the cards using her private key, and sends back Bob's cards
  - Bob decrypts his cards
- Both keep the encrypted version of the other's cards
- Discarded cards are encrypted and sent to the other
- To end the game, they claim their hands and prove them by revealing their private keys

RSA poker, cheating

- Alice and Bob want to play Poker over the telephone; they both set up for RSA
- Alice's encryption as $c = m^{e_a} \bmod n_a$ and decryption as $m = c^{d_a} \bmod n_a$
- This is a system with public keys, or?
- Security depends on complexity of factoring

RSA poker, cheating

- Alice and Bob want to play Poker over the telephone; they both set up for RSA
- Alice's encryption as $c = m^{e_a} \bmod n_a$ and decryption as $m = c^{d_a} \bmod n_a$
- This is a system with public keys, or?
- Security depends on complexity of factoring
- Can you cheat in this system?

Bit commitment, zero knowledge, secret sharing, games over the phone

- Bit commitment is used to convince someone that you have certain knowledge, without giving the knowledge to this someone — uses one-way functions
- A zero-knowledge scheme is also used to convince someone that you have certain knowledge, without giving the knowledge to this someone — uses a random commitment and a random challenge, not one-way functions
- Secret sharing is used to divide a secret into pieces so that all (or at least t) pieces are needed to read the secret
- Coin tossing or poker over the phone uses bit (or card) commitment to convince participants that it is a fair game