F8: ECC, ECDLP, and ECDSA

Many slides are from Rong-Jaye Chen@NCTU

Discrete Logarithm

- Fix a prime p. Let a, b be nonzero integers (mod p)
- The problem of finding x such that $a^x \equiv b \pmod{p}$ is called the discrete logarithm problem (DLP)
- We denote $x=L_a(b)$, and call it the discrete log of b w.r.t. a (mod p)
- Ex: $p=11$, $a=2$, $b=9$, then $x=L_2(9)=6$

In many references, $x=Log_a(b)$

Discrete Logarithms

- In the Diffie–Hellman and ElGamal methods, the difficulty of solving the discrete logarithm problem yields good cryptosystems
- Given p, a, b, solve $a^x \equiv b \pmod{p}$
- If $\{a^x : 0 \le x \le p-2\} = \{1,2,3,\ldots p-1\}$, a is called a primitive root mod p
  - a is aka a generator

Discrete Logarithms

Set of elements which are generated by the exponentiation of α

- Discrete log problem
  - Given $Z_p^* = \langle\alpha\rangle$
  - $\mathrm{Log}_\alpha(y) = x$, if $y = \alpha^x$.

- Example
  - $Z_{13}^* = \langle 2\rangle$; $2^1=2$, $2^2=4$, $2^3=8$, $2^4=3$, $2^5=6$, $2^6=12$, $2^7=11$, $2^8=9$, $2^9=5$, $2^{10}=10$, $2^{11}=7$, $2^{12}=1$
  - $\mathrm{Log}_2(5) = L_2(5) = 9$.

Algorithms that solve DLP

- Some are of sub-exponential complexity

That's why we need ECC

- Elliptic Curve Cryptography (ECC)
  - Similar to DLP
    - Called ECDLP (Elliptic Curve DLP)
  - No efficient algorithms yet

Elliptic curves over Real (R)

- Definition

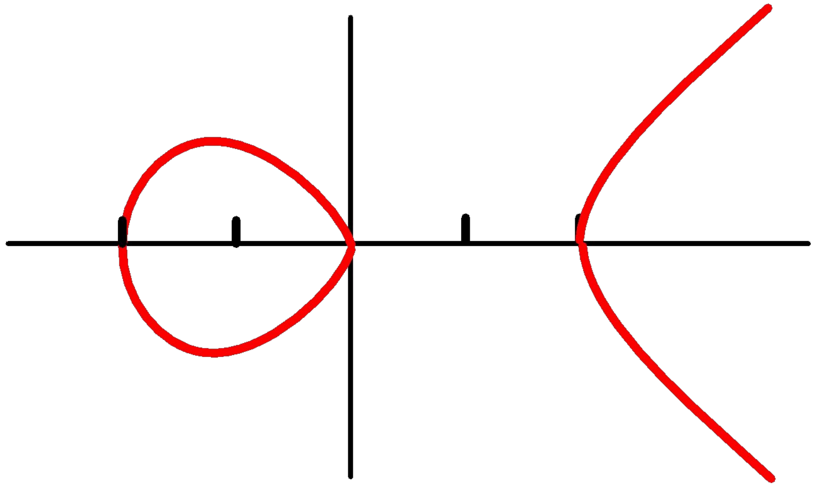
$$a, b \in \mathrm{R},\ 4a^3 + 27b^2 \neq 0$$

Let

$$E = \{ (x, y) \in \mathrm{R} \times \mathrm{R} | y^2 = x^3 + ax + b \} \cup \{ O \}$$

- Example:

$$E : y^2 = x^3 - 4x$$



"Adding" two points in an elliptic curve

- Group operation +

Given $P, Q \in E, P = (x_1, y_1), Q = (x_2, y_2)$

Compute $R = P + Q = (x_3, y_3)$

- Addition
  $(P \neq Q)$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$
$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = (x_1 - x_3)\lambda - y_1$$

- Doubling
  $(P = Q)$

$$\lambda = \frac{3{x_1}^2 + a}{2y_1}$$
$$x_3 = \lambda^2 - 2x_1$$
$$y_3 = (x_1 - x_3)\lambda - y_1$$

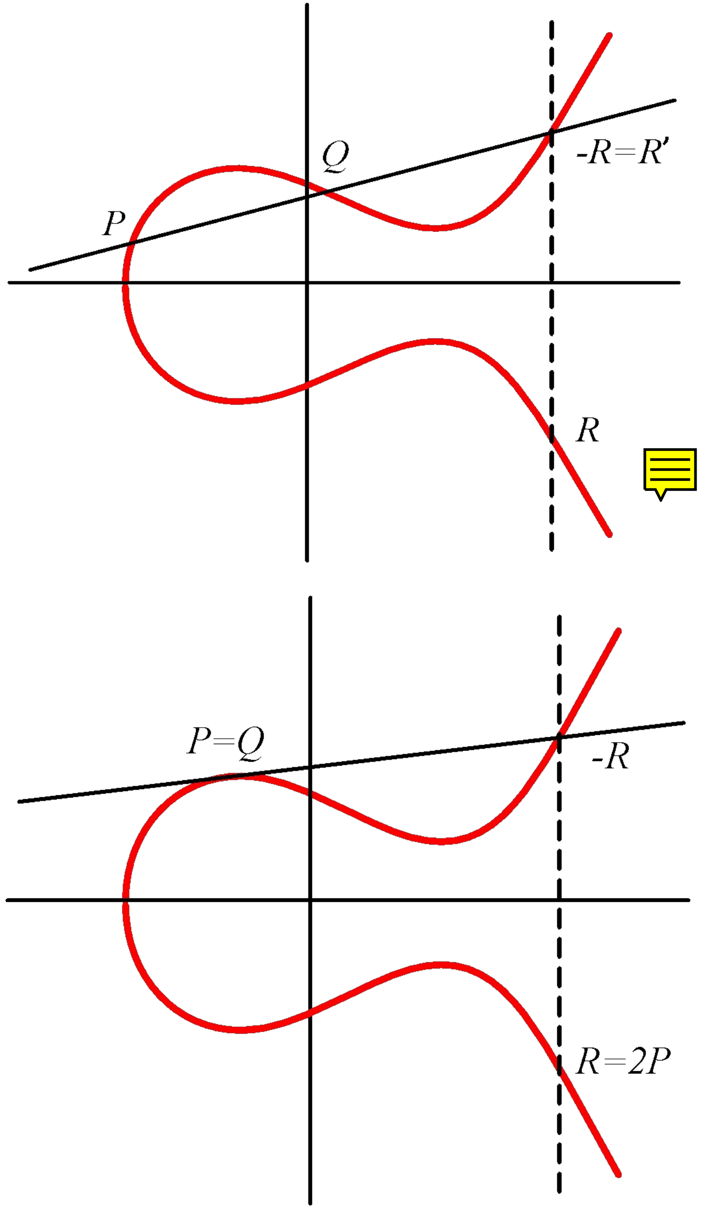
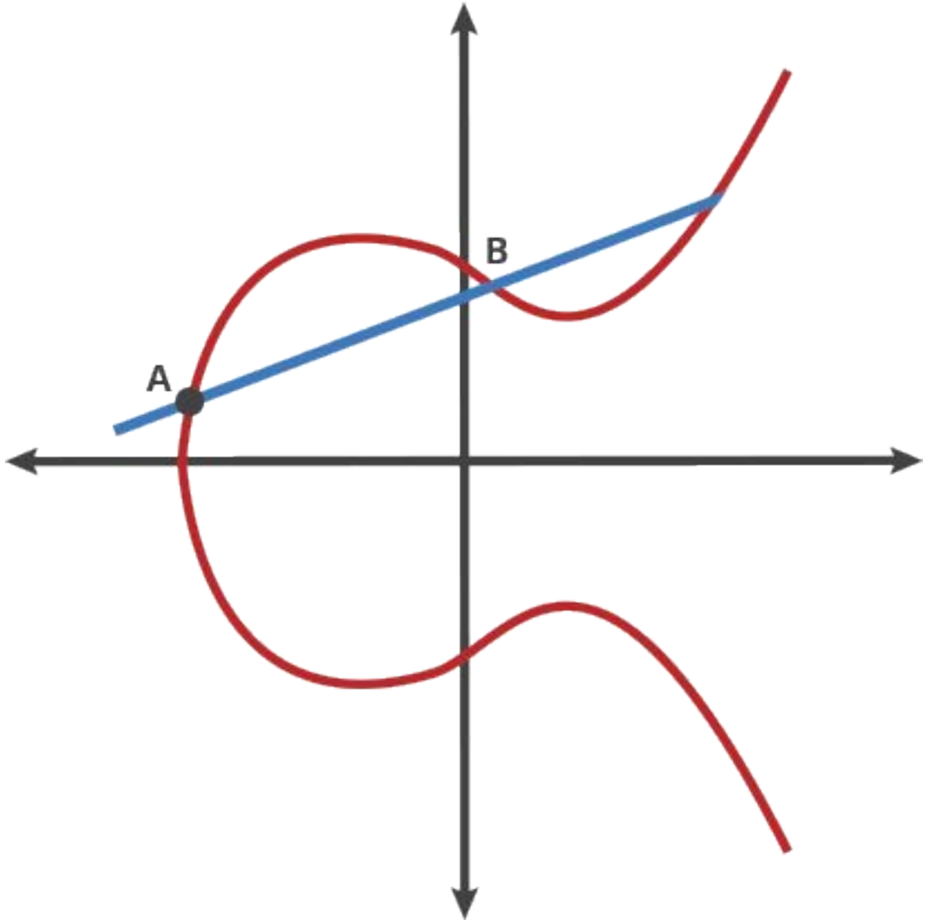


Illustration of Addition in an elliptic curve



A + B = C
A + C = D
A + D = E

Elliptic Curves over GF(p)

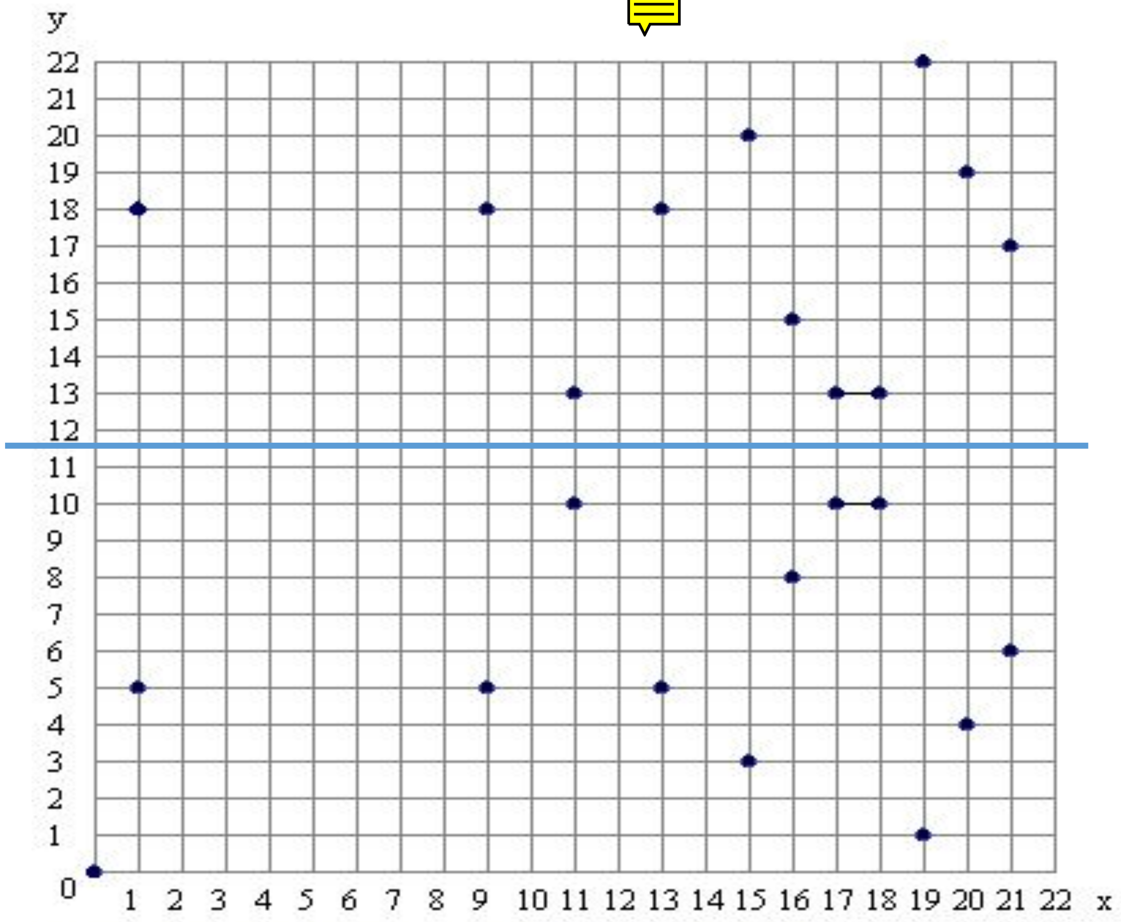
- Definition

$$p > 3, a, b \in Z_p,\ 4a^3 + 27b^2 \neq 0 \pmod{p}$$

Let $E = \{(x, y) \in Z_p \times Z_p | y^2 \equiv x^3 + ax + b \pmod{p}\} \cup \{O\}$

Example:

$$E : y^2 = x^3 + x \text{ over } Z_{23}$$



Elliptic curve equation: $y^2 = x^3 + x$ over F_{23}

a finite field or Galois field (GF) is a field that contains a finite number of elements. a field is a set on which the operations of multiplication, addition, subtraction and division are defined and satisfy certain basic rules.

An illustration of an ECDLP

- Example $E: y^2 = x^3 + x + 6 \text{ over } Z_{11}$

Find all (x, y) and **O**:

- Fix *x* and determine *y*
- **O** is an artificial point

12 (x, y) pairs plus **O**, and have *#E=13*

Cardinality, q in ECDSA table

x	x^3+x+6	quad res?	y
0	6	*no*	
1	8	*no*	
2	5	*yes*	4,7
3	3	*yes*	5,6
4	8	*no*	
5	4	*yes*	2,9
6	8	*no*	
7	4	*yes*	2,9
8	9	*yes*	3,8
9	7	*no*	
10	4	*yes*	2,9

Integer v is called a quadratic residue modulo n if there exists y s.t. $y^2 = v \bmod n$
Otherwise v is a quadratic non-residue

doubling illustration: ECDLP

- Example (continue):

There are 13 points on the group $E(Z_{11})$ and so any non-identity point (i.e. not the point at infinity, noted as **O**) is a generator of $E(Z_{11})$.

$$\alpha = (2,7)$$

Choose a generator

$$2\alpha = (x_2, y_2)$$

Compute

$$\lambda = \frac{3{x_1}^2 + 1}{2y_1} = \frac{3(2)^2 + 1}{2 \times 7} = \frac{13}{14} = 2 \times 3^{-1} = 2 \times 4 = 8 \mod 11$$
$$x_2 = \lambda^2 - 2x_1 = (8)^2 - 2 \times (2) = 5 \mod 11$$
$$y_2 = (x_1 - x_2)\lambda - y_1 = (2 - 5) \times 8 - 7 = 2 \mod 11$$

addition illustration: ECDLP

- Example (continue):

Compute $3\alpha = (x_3, y_3)$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{2-7}{5-2} = 2 \mod 11$$

$$x_3 = \lambda^2 - x_1 - x_2 = 2^2 - 2 - 5 = 8 \mod 11$$

$$y_3 = (x_1 - x_3)\lambda - y_1 = (2-8) \times 2 - 7 = 3 \mod 11$$

along this line, we can compute

$$\alpha = (2,7) \qquad 2\alpha = (5,2) \qquad 3\alpha = (8,3)$$

$$4\alpha = (10,2) \qquad 5\alpha = (3,6) \qquad 6\alpha = (7,9)$$

$$7\alpha = (7,2) \qquad 8\alpha = (3,5) \qquad 9\alpha = (10,9)$$

$$10\alpha = (8,8) \qquad 11\alpha = (5,9) \qquad 12\alpha = (2,4)$$

Calculate public key from private key: elliptic curve cryptography (ECC)

- Example:

Compute 3895P

$$3895P = \underbrace{P + P + \cdots + P}_{3894 \text{ additions needed}}$$

$$\begin{aligned}
&= (111100110111)_2 P \\
&= 2(2(2(2(2(2(2(2(2(2(2P + P) + P) + P))) + P) + P)) + P) + P) + P \\
&\rightarrow \quad \text{11 doublings and 8 additions needed}
\end{aligned}$$

$$\begin{aligned}
&= (1000(-1)0100(-1)00(-1))_2 P \\
&= 2(2(2(2(2(2(2(2(2(2(2(2P))) - P)) + P))) - P))) - P \\
&\rightarrow \quad \text{12 doublings and 4 (additions or subtractions) needed}
\end{aligned}$$

Elliptic Curve Discrete Logarithm Problem (ECDLP)

- Basic computation of ECC
  - $Q = kP = \underbrace{P + P + \dots + P}_{k\ times}$
    where P is a curve point, k is an integer
- Strength of ECC
  - Given curve, the point P, and kP
    It is hard to recover k
    – Elliptic Curve Discrete Logarithm Problem (ECDLP)

We will first see ElGamal DLP
Then ECC version of ElGamal, which is ECDLP

Setting up ElGamal

- Let p be a large prime
  - By “large” we mean here a prime of thousands bits
- Select a special number g
  - The number g must be a primitive element modulo p.
- Choose a private key x
  - This can be any number bigger than 1 and smaller than $p-1$
- Compute the public key y (from x, p and g)
  - The public key y is g raised to the power of the private key x modulo p.

$$y = g^x \bmod p$$

- Publicize p, g, y

ElGamal encryption

The first job is to represent the plaintext M as a series of numbers modulo p. Then:

1. Generate a random number k (ephemeral key)
   - Only the sender knows k
2. Compute two values C_1 and C_2, where

   $\mathbf{C_1 = g^k \bmod p}$ and $\mathbf{C_2 = My^k \bmod p}$
3. Send the ciphertext C, which consists of the two separate values C_1 and C_2.

ElGamal decryption

$$C_1 = g^k \bmod p \quad C_2 = My^k \bmod p$$

1 - The receiver begins by using their private key **x** to transform $\mathbf{C_1}$ into something more useful:

$$C_1^x = (g^k)^x \bmod p$$

NOTE: $C_1^x = (g^k)^x = (g^x)^k = (y)^k = y^k \bmod p$

2 - This is a very useful quantity because if you divide $\mathbf{C_2}$ by it, you get **M**. In other words:

$$C_2 / C_1^x = C_2 / y^k = (My^k) / y^k = M \bmod p$$

Come back to ECDLP

x is the message (x_1,x_2) and k is the ephemeral key

- Example (continue):
  Let's modify ElGamal encryption by using the elliptic curve $E(Z_{11})$.
  Suppose that generator $\alpha = (2,7)$ and Bob's private key is 7, so
  $$\beta = 7\alpha = (7,2)$$
  Thus the encryption operation for msg x is $e_K(x,k) = (k(2,7), x + k(7,2)),$
  where $x \in E$ and $0 \le k \le 12$, and the decryption operation is
  $$d_K(y_1,y_2) = y_2 - 7y_1 = x + k(7,2) - 7k(2,7)$$
  $$= x + k(7,2) - k(7,2)$$

ECDLP: encryption/decryption

- Example (continue):

  Suppose that Alice wishes to encrypt the plaintext x = (10,9) ($x \in E$).

  If she chooses the random value k = 3, then

  $$y_1 = 3(2,7) = (8,3) \text{ and}$$

  $$y_2 = (10,9) + 3(7,2) = (10,9) + (3,5) = (10,2$$

- Hence

  $$y = ((8,3),(10,2))$$

- Now, if Bob receives the ciphertext y, he decrypts it as follows:

  $$x = (10,2) - 7(8,3) = (10,2) - (3,$$

  $$= (10,2) + (3,6) = (10,9)$$

Security of ECC versus RSA/ElGamal

- Elliptic curve cryptosystems give the most security per bit of any known public-key scheme
- The ECDLP problem appears to be much more difficult than the integer factorization problem and the discrete logarithm problem of Z_p
  - no efficient algorithm like index calculus algorithm
- The strength of elliptic curve cryptosystems grows much faster with the key size increases than does the strength of RSA

Security level of Elliptic Curve Cryptography (ECC)

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

NIST Recommended Key Sizes

Before going into ECDSA,
let’s review DSA briefly

Digital Signature Algorithm (DSA)

$L=0 \bmod 64,\ 512 \le L \le 1024$

- Let p be a L–bit prime such that the DLP in Z_p* is intractable
- Let q be a 160–bit prime that divides $p-1$.
- Let α be a q_{th} root of 1 modulo p

$p = qr+1$
$h^r = \alpha$

Define $K=\{\ (p,q,\alpha,a,\beta):\ \beta=\alpha^a \bmod p\ \}$

p,q,α,β are the public key, a is private key

DSA revisited

- For a (secret) random number k, define
  $sig_K(x,k)=(\gamma,\delta)$, where
  $\gamma=(\alpha^k \bmod p) \bmod q$ and
  $\delta=(H(x)+a\gamma)k^{-1} \bmod q$

x is the message to be signed

- For a message $(x,(\gamma,\delta))$, verification is done by performing the following computations:

  $e_1=H(x)*\delta^{-1} \bmod q$
  $e_2=\gamma*\delta^{-1} \bmod q$

$verify(x,(\gamma,\delta))=true$ iff $(\alpha^{e1}\beta^{e2} \bmod p) \bmod q=\gamma$

Elliptic Curve DSA (ECDSA)

- Let p be a prime, and let E be an elliptic curve defined over F_p.

- Let A be a point on E having prime order q, such that DLP in $\langle A \rangle$ is infeasible.

Define $K=\{ (p,q,E,A,m,B): B=mA \}$

p,q,E,A,B are the public key, m is private key

ECDSA

- For a (secret) random number k, define $sig_K(x,k)=(r,s)$, where $kA=(u,v)$, $r=u \bmod q$ and $s=k^{-1}(H(x)+mr) \bmod q$

- For a message $\{x,(r,s)\}$, verification is done by performing the following computations:

  $i=H(x)*s^{-1} \bmod q$

  $j=r*s^{-1} \bmod q$

  $(u,v)=iA+jB$

  $verify(x,(r,s))=true$ iff $u \bmod q=r$

DSA issues

- DSA vulnerability
  - old scheme
  - Pseudo random number generation has poor implementation
- ECDSA is favored over DSA

Bitcoin uses ECDSA

- Elliptic Curve Digital Signature Algorithm
- curve used is `secp256k1`
- set of points $(x,y) \in \{F_p \times F_p \mid y^2 = x^3 + 7 \pmod{p}\}$
- $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$
- Forms a group E, $|E| = q \approx p \approx 2^{256}$

	range	format	size (bits)
sk	Z_q	random	256
pk	E	$sk \cdot G$	512/257*
m	Z_q	H(message)	256
sig	$Z_q \times Z_q$	(r, s)	512

ECDSA problem

(r, −s) is also a valid ECDSA signature $K=\{ (p,q,E,A,m,B): B=mA \}$

where m is the private key, A is the generator

For a (secret) random number k, define $sig_K(x,k)=(r,s)$,

where $kA=(u,v)$, $B=mA$, $r=u \bmod q$ and

$s=k^{-1}(H(x)+mr) \bmod q$

For a message $\{x,(r,s)\}$, verification is done as follows:

$i=H(x)*s^{-1} \bmod q$

$j=r*s^{-1} \bmod q$

$(u,v)=iA+jB=C$

verify(x,(r,s))=true iff u mod

x−axis coordinate of C is equal to that of −C

For a message $\{x,(r,-s)\}$, verification is done as follows:

$i=H(x)*(-s)^{-1} \bmod q$

$j=r*(-s)^{-1} \bmod q$

$iA+jB = -C = (u,-v)$

Transaction malleability attack

- Alice sends 1 bitcoin to Bob with a Tx id ***A***.
  - With her signature (r, s)
- However, before the transaction is confirmed, Bob alters the signature of the transaction to produce a new Tx id ***B***.
  - With the signature (r, –s)
- Having received the 1 bitcoin but with Tx id ***B***, Bob then informs Alice that he has not received the bitcoin.
- When Alice searches a block explorer using Tx id ***A*** to confirm Bob's claim, she cannot find the transaction.
  - Assuming a failed transaction and that the 1 bitcoin was never sent, Alice sends Bob another bitcoin, resulting in a total of 2 bitcoins being sent to Bob

Segregated Witness (SegWit)

- Signature and PubKey are not used in making a TXID
  - Called witness
  - moved outside the block
- Also increases the effective block size
- A soft fork in Bitcoin