Practical Aspects of Modern Cryptography

Winter 2011

Josh Benaloh
Brian LaMacchia

Public-Key History

- 1976 *New Directions in Cryptography*

  Whit Diffie and Marty Hellman

  - One-Way functions
  - Diffie-Hellman Key Exchange

- 1978 RSA paper

  Ron Rivest, Adi Shamir, and Len Adleman

  - RSA Encryption System
  - RSA Digital Signature Mechanism

The Fundamental Equation

$$Z = Y^X \bmod N$$

Diffie-Hellman

$$Z = Y^X \bmod N$$

When X is unknown, the problem is known as the *discrete logarithm* and is generally believed to be hard to solve.

Diffie-Hellman Key Exchange

Alice

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

Bob

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

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

One-Way Trap-Door Functions

$$Z=Y^X \bmod N$$

Recall that this equation is solvable for Y if the factorization of N is known, but is *believed* to be hard otherwise.

RSA Public-Key Cryptosystem

Alice

- Select two large random primes P & Q.
- Publish the product $N=PQ$.

- Use knowledge of P & Q to compute Y.

Anyone

- To send message Y to Alice, compute

$$Z=Y^X \bmod N.$$

- Send Z and X to Alice.

Why Does RSA Work?

Fact

When $N = PQ$ is the product of distinct primes,

$$Y^X \bmod N = Y$$

whenever

$X \bmod (P-1)(Q-1) = 1$ and $0 \leq Y < N$.

Fermat's Little Theorem

If p is prime,

then $x^{p-1} \text{mod } p = 1$ for all $0 < x < p$.

Equivalently …

If p is prime, then

$$x^p \text{ mod } p \;=\; x \text{ mod } p$$

for all integers x.

Proof of Fermat's Little Theorem

<u>The Binomial Theorem</u>

$(x+y)^p = \sum_{i=0}^{p} \binom{n}{i} x^i y^{p-i}$ where $\binom{p}{i} = \frac{p!}{i!(p-i)!}$

Proof of Fermat's Little Theorem

<u>The Binomial Theorem</u>

$(x+y)^p = \sum_{i=0}^{p} \binom{n}{i} x^i y^{p-i}$ where $\binom{p}{i} = \frac{p!}{i!(p-i)!}$

If p is prime, then $\binom{p}{i} = 0$ for $0 < i < p$.

Proof of Fermat's Little Theorem

<u>The Binomial Theorem</u>

$(x+y)^p = \sum_{i=0}^{p}\binom{n}{i}x^i y^{p-i}$ where $\binom{p}{i} = \frac{p!}{i!(p-i)!}$

If p is prime, then $\binom{p}{i} = 0$ for $0 < i < p$.

Thus, $(x+y)^p \bmod p = (xp + yp) \bmod p$.

Proof of Fermat's Little Theorem

$$x^p \bmod p = x \bmod p$$

Proof of Fermat's Little Theorem

$$x^p \bmod p = x \bmod p$$

By induction on x…

Proof of Fermat's Little Theorem

$$x^p \bmod p = x \bmod p$$

By induction on x…

Basis

Proof of Fermat's Little Theorem

$$x^p \bmod p = x \bmod p$$

By induction on x…

<u>Basis</u>

If $x = 0$, then $x^p \bmod p = 0 = x \bmod p$.

Proof of Fermat's Little Theorem

$$x^p \bmod p = x \bmod p$$

By induction on x…

<u>Basis</u>

If $x = 0$, then $x^p \bmod p = 0 = x \bmod p$.

If $x = 1$, then $x^p \bmod p = 1 = x \bmod p$.

Proof of Fermat’s Little Theorem

<u>Inductive Step</u>

Proof of Fermat's Little Theorem

Inductive Step

Assume that $x^p \bmod p = x \bmod p$.

Proof of Fermat's Little Theorem

Inductive Step

Assume that $x^p \bmod p \; = \; x \bmod p$.

Then $(x+1)^p \bmod p = (x^p + 1^p) \bmod p$

(by the binomial theorem)

Proof of Fermat's Little Theorem

<u>Inductive Step</u>

Assume that $x^p \bmod p = x \bmod p$.

Then $(x+1)^p \bmod p = (x^p + 1^p) \bmod p$

$= (x+1) \bmod p$ (by inductive hypothesis).

Proof of Fermat's Little Theorem

<u>Inductive Step</u>

Assume that $x^p \bmod p \ = \ x \bmod p$.

Then $(x+1)^p \bmod p = (x^p + 1^p) \bmod p$

$= (x+1) \bmod p$ (by inductive hypothesis).

Hence, $x^p \bmod p = x \bmod p$ for integers $x \geq 0$.

Proof of Fermat's Little Theorem

<u>Inductive Step</u>

Assume that $x^p \bmod p = x \bmod p$.

Then $(x+1)^p \bmod p = (x^p + 1^p) \bmod p$

$= (x+1) \bmod p$ (by inductive hypothesis).

Hence, $x^p \bmod p = x \bmod p$ for integers $x \geq 0$.

Also true for negative x, since $(-x)^p = (-1)^p x^p$.

Proof of RSA

Proof of RSA

We have shown …

$Y^P \bmod P = Y$ whenever $0 \leq Y < P$

and P is *prime*.

Proof of RSA

We have shown …

$Y^P \bmod P = Y$ whenever $0 \le Y < P$

and P is *prime*.

You will show …

$Y^{K(P-1)(Q-1)+1} \bmod PQ = Y$ when $0 \le Y < PQ$

P and Q are distinct primes and $K \ge 0$.

Corollary of Fermat

$$x^p \bmod p \;=\; x \bmod p$$

$$\Downarrow$$

$$x^{k(p-1)+1} \bmod p \;=\; x \bmod p$$

For all prime p and $k \geq 0$.

Finding Primes

Finding Primes

Euclid's proof of the infinity of primes

Finding Primes

Euclid's proof of the infinity of primes

- Suppose that the set of all primes were finite.

Finding Primes

Euclid's proof of the infinity of primes

- Suppose that the set of all primes were finite.
- Let N be the product of all of the primes.

Finding Primes

Euclid's proof of the infinity of primes

- Suppose that the set of all primes were finite.
- Let N be the product of all of the primes.
- Consider $N + 1$.

Finding Primes

Euclid's proof of the infinity of primes

- Suppose that the set of all primes were finite.
- Let N be the product of all of the primes.
- Consider $N + 1$. Is $N + 1$ prime or composite?

Finding Primes

Euclid's proof of the infinity of primes

- Suppose that the set of all primes were finite.
- Let N be the product of all of the primes.
- Consider $N + 1$. Is $N + 1$ prime or composite?
- The prime factors of $N + 1$ are not among the finite set of primes multiplied to form N.

Finding Primes

Euclid's proof of the infinity of primes

- Suppose that the set of all primes were finite.
- Let N be the product of all of the primes.
- Consider $N + 1$. Is $N + 1$ prime or composite?
- The prime factors of $N + 1$ are not among the finite set of primes multiplied to form N.
- So N must be a prime not in the set.

Finding Primes

Euclid's proof of the infinity of primes

- Suppose that the set of all primes were finite.
- Let N be the product of all of the primes.
- Consider $N + 1$. Is $N + 1$ prime or composite?
- The prime factors of $N + 1$ are not among the finite set of primes multiplied to form N.
- So N must be a prime not in the set.
- This contradicts the assumption that the set of all primes is finite.

The Prime Number Theorem

The Prime Number Theorem

The number of primes less than N is approximately $N/(\ln N)$.

The Prime Number Theorem

The number of primes less than N is approximately $N/(\ln N)$.

Thus, approximately 1 out of every n randomly selected n-bit integers will be prime.

But How Do We Find Primes?

Testing Primality

Testing Primality

Recall Fermat's Little Theorem

If p is prime, then $a^{p-1} \bmod p = 1$ for all a in the range $0 < a < p$.

Testing Primality

Recall Fermat's Little Theorem

If p is prime, then $a^{p-1} \bmod p = 1$ for all a in the range $0 < a < p$.

Fact

For almost all composite p and $a > 1$, $a^{p-1} \bmod p \neq 1$.

The Miller-Rabin Primality Test

The Miller-Rabin Primality Test

To test an integer N for primality, write $N - 1$ as
$N - 1 = m2^k$ where m is odd.

The Miller-Rabin Primality Test

To test an integer N for primality, write $N - 1$ as
$N - 1 = m2^k$ where m is odd.

Repeat several (many) times

The Miller-Rabin Primality Test

To test an integer N for primality, write $N-1$ as
$N-1 = m2^k$ where m is odd.

Repeat several (many) times

- Select a random a in $1 < a < N-1$

The Miller-Rabin Primality Test

To test an integer N for primality, write $N-1$ as

$N-1 = m2^k$ where m is odd.

Repeat several (many) times

- Select a random a in $1 < a < N-1$
- Compute $a^m, a^{2m}, a^{4m}, \ldots, a^{(N-1)/2}$ all mod N.

The Miller-Rabin Primality Test

To test an integer N for primality, write $N-1$ as
$N-1 = m2^k$ where m is odd.

Repeat several (many) times

- Select a random a in $1 < a < N-1$
- Compute $a^m, a^{2m}, a^{4m}, \ldots, a^{(N-1)/2}$ all mod N.
- If $a^m = \pm 1$ or if some $a^{2^i m} = -1$, then N is probably prime – continue.

The Miller-Rabin Primality Test

To test an integer N for primality, write $N-1$ as
$N-1 = m2^k$ where m is odd.

Repeat several (many) times

- Select a random a in $1 < a < N-1$
- Compute $a^m, a^{2m}, a^{4m}, \ldots, a^{(N-1)/2}$ all mod N.
- If $a^m = \pm 1$ or if some $a^{2^i m} = -1$, then N is probably prime – continue.
- Otherwise, N is composite – stop.

Sieving for Primes

Pick a random starting point N.

N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10	N+11

Sieving out multiples of 2

Sieving for Primes

Pick a random starting point N.

N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10	N+11
	X										

Sieving out multiples of 2

Sieving for Primes

Pick a random starting point N.

N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10	N+11
	X		X								

Sieving out multiples of 2

Sieving for Primes

Pick a random starting point N.

N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10	N+11
	X		X		X						

Sieving out multiples of 2

Sieving for Primes

Pick a random starting point N.

N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10	N+11
	X		X		X		X				

Sieving out multiples of 2

Sieving for Primes

Pick a random starting point N.

N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10	N+11
	X		X		X		X		X		

Sieving out multiples of 2

Sieving for Primes

Pick a random starting point N.

N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10	N+11
	X		X		X		X		X		X

Sieving out multiples of 2

Sieving for Primes

Pick a random starting point N.

N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10	N+11
	X		X		X		X		X		X

Sieving out multiples of 3

Sieving for Primes

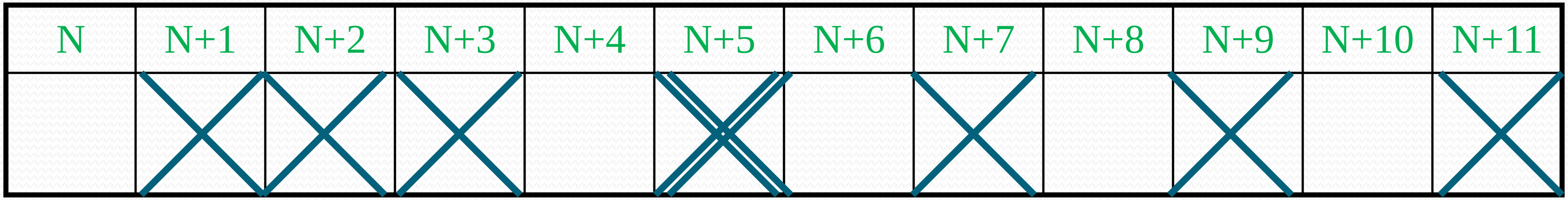
Pick a random starting point N.

N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10	N+11
	X	X	X		X		X		X		X

Sieving out multiples of 3

Sieving for Primes

Pick a random starting point N.



Sieving out multiples of 3

Sieving for Primes

Pick a random starting point N.

N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10	N+11
	X	X	X		XX		X	X	X		X

Sieving out multiples of 3

Sieving for Primes

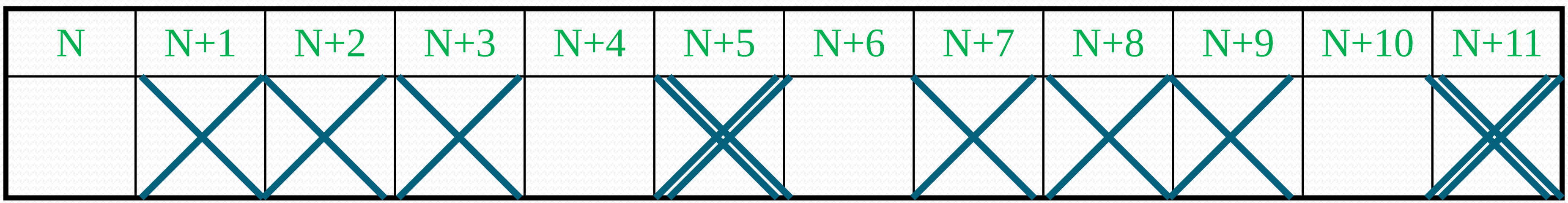
Pick a random starting point N.

N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10	N+11
	X	X	X		XX		X	X	X		XX

Sieving out multiples of 3

Sieving for Primes

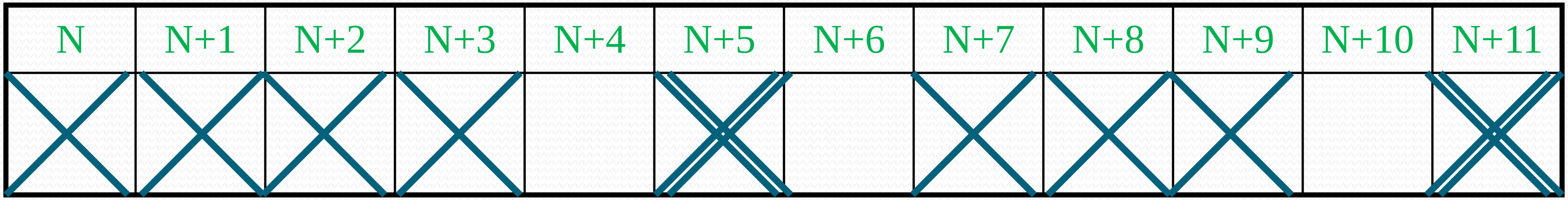
Pick a random starting point N.



Sieving out multiples of 5

Sieving for Primes

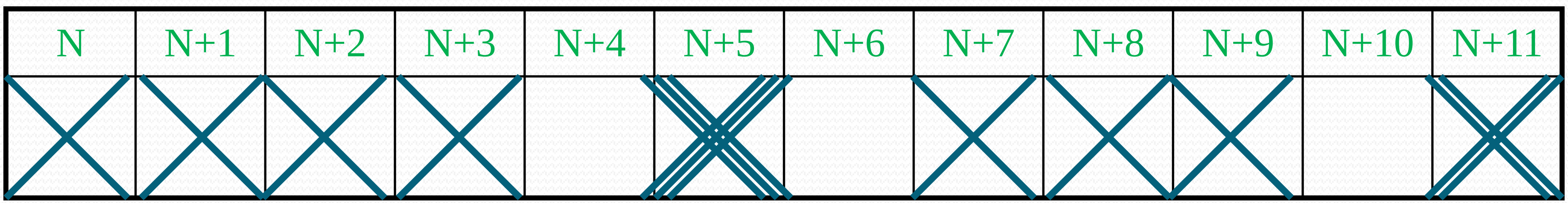
Pick a random starting point N.



Sieving out multiples of 5

Sieving for Primes

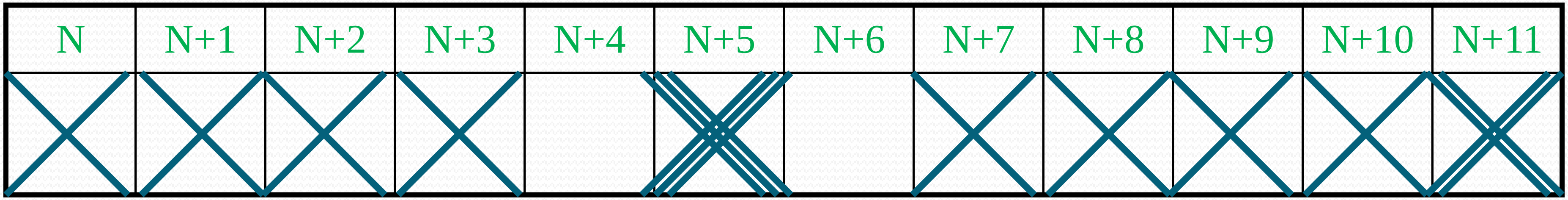
Pick a random starting point N.



Sieving out multiples of 5

Sieving for Primes

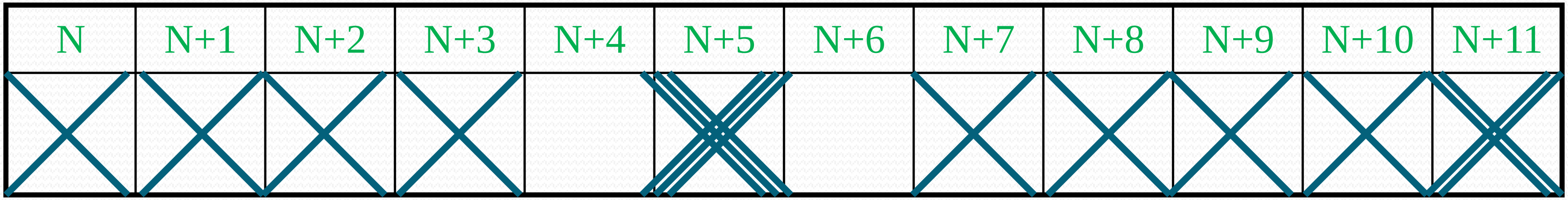
Pick a random starting point N.



Sieving out multiples of 5

Sieving for Primes

Pick a random starting point N.



Sieving out multiples of 5

Only a few "good" candidate primes will survive.

Reprise of RSA Set-Up

Reprise of RSA Set-Up

- Use sieving to find large candidate primes.

Reprise of RSA Set-Up

- Use sieving to find large candidate primes.
- Use Miller-Rabin on candidate primes to find two *almost* certainly prime integers P and Q.

Reprise of RSA Set-Up

- Use sieving to find large candidate primes.
- Use Miller-Rabin on candidate primes to find two *almost* certainly prime integers P and Q.
- Form public modulus $N = PQ$.

Reprise of RSA Set-Up

- Use sieving to find large candidate primes.
- Use Miller-Rabin on candidate primes to find two *almost* certainly prime integers P and Q.
- Form public modulus $N = PQ$.
- Select public exponent e (usually $e = 65537$).

Reprise of RSA Set-Up

- Use sieving to find large candidate primes.
- Use Miller-Rabin on candidate primes to find two *almost* certainly prime integers P and Q.
- Form public modulus $N = PQ$.
- Select public exponent e (usually $e = 65537$).
- Use extended Euclidean algorithm to compute private exponent $d = e^{-1} \bmod (P-1)(Q-1)$.

Reprise of RSA Set-Up

- Use sieving to find large candidate primes.
- Use Miller-Rabin on candidate primes to find two *almost* certainly prime integers P and Q.
- Form public modulus $N = PQ$.
- Select public exponent e (usually $e = 65537$).
- Use extended Euclidean algorithm to compute private exponent $d = e^{-1} \bmod (P-1)(Q-1)$.
- Publish public key N (and e).

Reprise of RSA Encryption

Reprise of RSA Encryption

- Use public key to encrypt message $0 \leq m < N$ as

Reprise of RSA Encryption

- Use public key to encrypt message $0 \leq m < N$ as
$$E(m) = m^e \bmod N.$$

Reprise of RSA Encryption

- Use public key to encrypt message $0 \leq m < N$ as

$$E(m) = m^e \bmod N.$$

- Use private decryption exponent d to decrypt

Reprise of RSA Encryption

- Use public key to encrypt message $0 \leq m < N$ as
$$E(m) = m^e \bmod N.$$

- Use private decryption exponent d to decrypt
$$D\big(E(m)\big) = (m^e \bmod N)^d \bmod N$$

Reprise of RSA Encryption

- Use public key to encrypt message $0 \leq m < N$ as

$$E(m) = m^e \bmod N.$$

- Use private decryption exponent d to decrypt

$$\begin{aligned} D\big(E(m)\big) &= (m^e \bmod N)^d \bmod N \\ &= m^{ed} \bmod N \end{aligned}$$

Reprise of RSA Encryption

- Use public key to encrypt message $0 \leq m < N$ as

$$E(m) = m^e \bmod N.$$

- Use private decryption exponent d to decrypt

$$\begin{aligned} D\big(E(m)\big) &= (m^e \bmod N)^d \bmod N \\ &= m^{ed} \bmod N \\ &= m \end{aligned}$$

Reprise of RSA Signatures

Reprise of RSA Signatures

- Use private decryption exponent d to sign message $0 \leq m < N$ as

Reprise of RSA Signatures

- Use private decryption exponent d to sign message $0 \leq m < N$ as

$$D(m) = m^d \bmod N.$$

Reprise of RSA Signatures

- Use private decryption exponent d to sign message $0 \leq m < N$ as
$$D(m) = m^d \bmod N.$$
- Verify signature by using public key to compute

Reprise of RSA Signatures

- Use private decryption exponent d to sign message $0 \leq m < N$ as
$$D(m) = m^d \bmod N.$$

- Verify signature by using public key to compute
$$E\big(D(m)\big) = (m^d \bmod N)^e \bmod N$$

Reprise of RSA Signatures

- Use private decryption exponent d to sign message $0 \leq m < N$ as

$$D(m) = m^d \bmod N.$$

- Verify signature by using public key to compute

$$E\big(D(m)\big) = (m^d \bmod N)^e \bmod N$$
$$= m^{de} \bmod N$$

Reprise of RSA Signatures

- Use private decryption exponent d to sign message $0 \leq m < N$ as

$$D(m) = m^d \bmod N.$$

- Verify signature by using public key to compute

$$\begin{aligned} E\big(D(m)\big) &= (m^d \bmod N)^e \bmod N \\ &= m^{de} \bmod N \\ &= m \end{aligned}$$

The Digital Signature Algorithm

In 1991, the National Institute of Standards and Technology published a Digital Signature Standard that was intended as an option free of intellectual property constraints.

The Digital Signature Algorithm

DSA uses the following parameters

- Prime p – anywhere from 512 to 1024 bits
- Prime q – 160 bits such that q divides $p - 1$
- Integer h in the range $1 < h < p - 1$
- Integer $g = h^{(p-1)/q} \bmod p$
- Secret integer x in the range $1 < x < q$
- Integer $y = g^x \bmod p$

The Digital Signature Algorithm

The Digital Signature Algorithm

To sign a 160-bit message M,

The Digital Signature Algorithm

To sign a 160-bit message M,

- Generate a random integer k with $0 < k < q$,

The Digital Signature Algorithm

To sign a 160-bit message M,

- Generate a random integer k with $0 < k < q$,
- Compute $r = (g^k \bmod p) \bmod q$,

The Digital Signature Algorithm

To sign a 160-bit message M,

- Generate a random integer k with $0 < k < q$,
- Compute $r = (g^k \bmod p) \bmod q$,
- Compute $s = ((M + xr)/k) \bmod q$.

The Digital Signature Algorithm

To sign a 160-bit message M,

- Generate a random integer k with $0 < k < q$,
- Compute $r = (g^k \bmod p) \bmod q$,
- Compute $s = ((M + xr)/k) \bmod q$.

The pair (r, s) is the signature on M.

The Digital Signature Algorithm

The Digital Signature Algorithm

A signature (r, s) on M is verified as follows:

The Digital Signature Algorithm

A signature (r, s) on M is verified as follows:

- Compute $w = 1/s \bmod q$,

The Digital Signature Algorithm

A signature (r, s) on M is verified as follows:

- Compute $w = 1/s \; mod \; q$,
- Compute $a = wM \; mod \; q$,

The Digital Signature Algorithm

A signature (r, s) on M is verified as follows:

- Compute $w = 1/s \bmod q$,
- Compute $a = wM \bmod q$,
- Compute $b = wr \bmod q$,

The Digital Signature Algorithm

A signature (r, s) on M is verified as follows:

- Compute $w = 1/s \; mod \; q$,
- Compute $a = wM \; mod \; q$,
- Compute $b = wr \; mod \; q$,
- Compute $v = (g^a y^b \bmod p) \bmod q$.

The Digital Signature Algorithm

A signature (r, s) on M is verified as follows:

- Compute $w = 1/s \ mod \ q$,
- Compute $a = wM \ mod \ q$,
- Compute $b = wr \ mod \ q$,
- Compute $v = (g^a y^b \bmod p) \bmod q$.

Accept the signature only if $v = r$.

Elliptic Curve Cryptosystems

Elliptic Curve Cryptosystems

An elliptic curve

Elliptic Curve Cryptosystems

An elliptic curve

$$y^2 = x^3 + Ax + B$$

Elliptic Curves

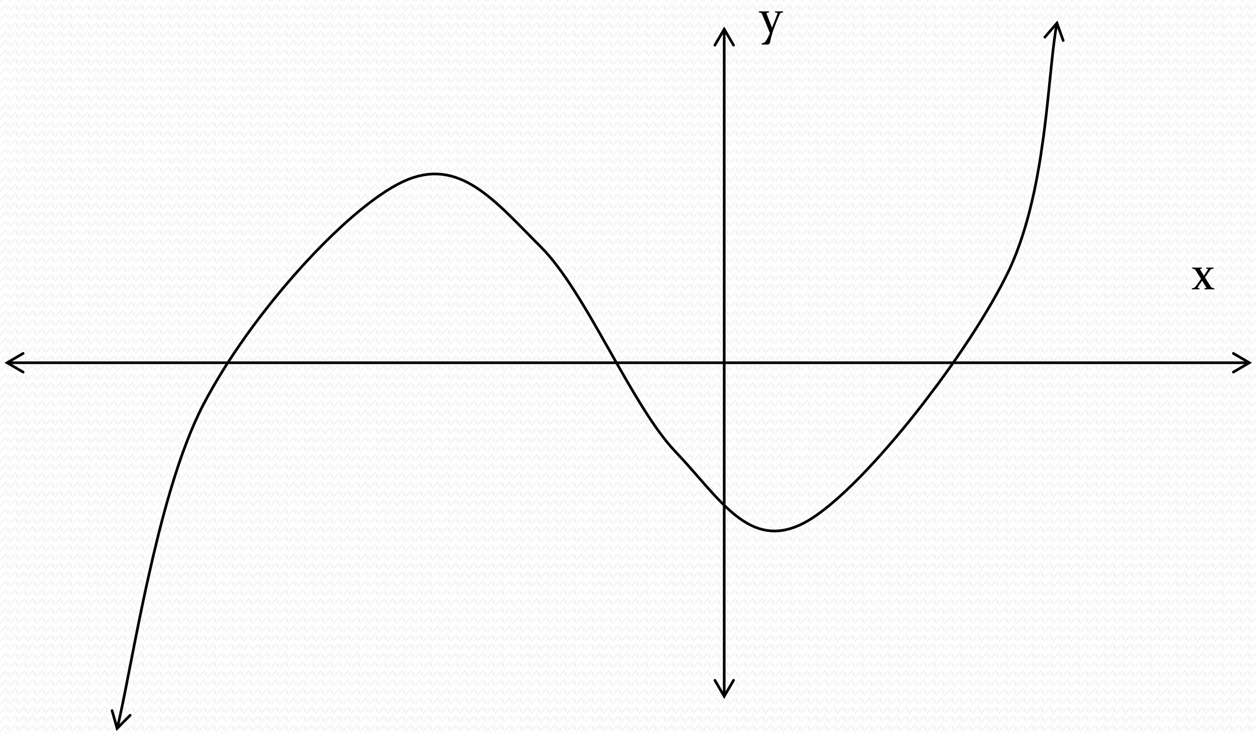
$$y^2 = x^3 + Ax + B$$

Elliptic Curves

$$y = x^3 + Ax + B$$

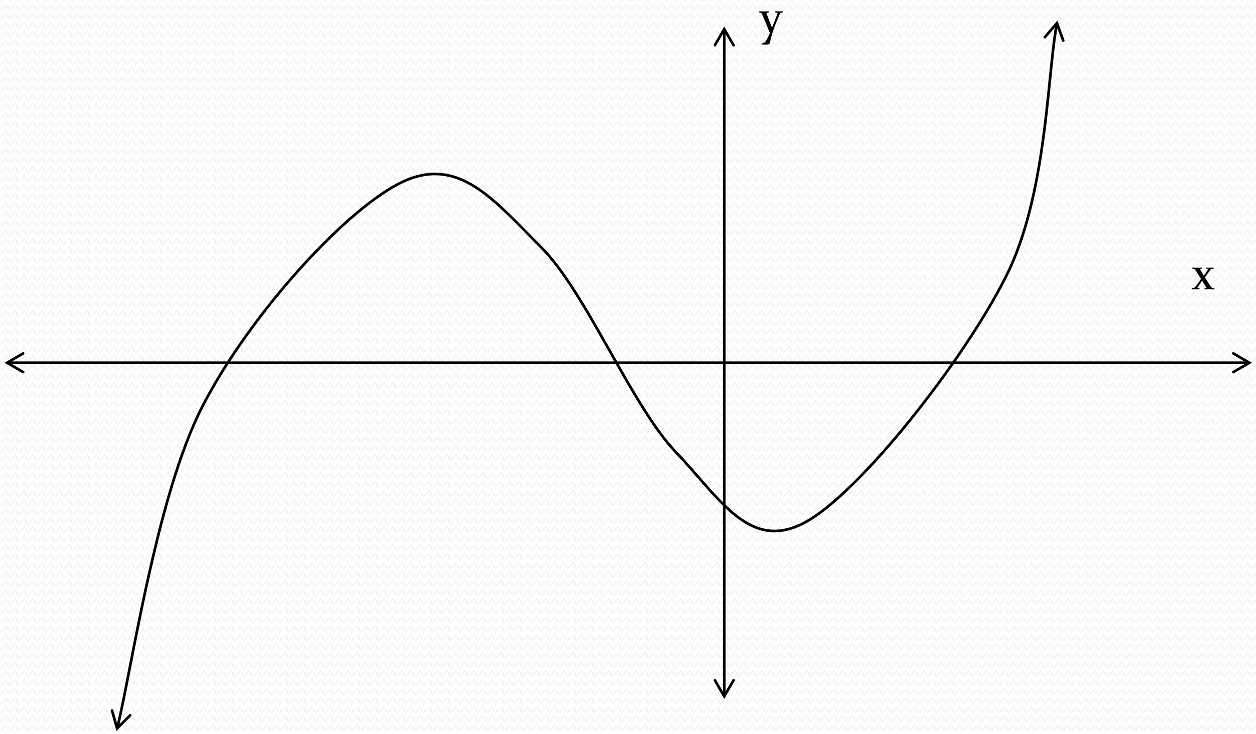
Elliptic Curves

$$y \ \ = x^3 + Ax + B$$



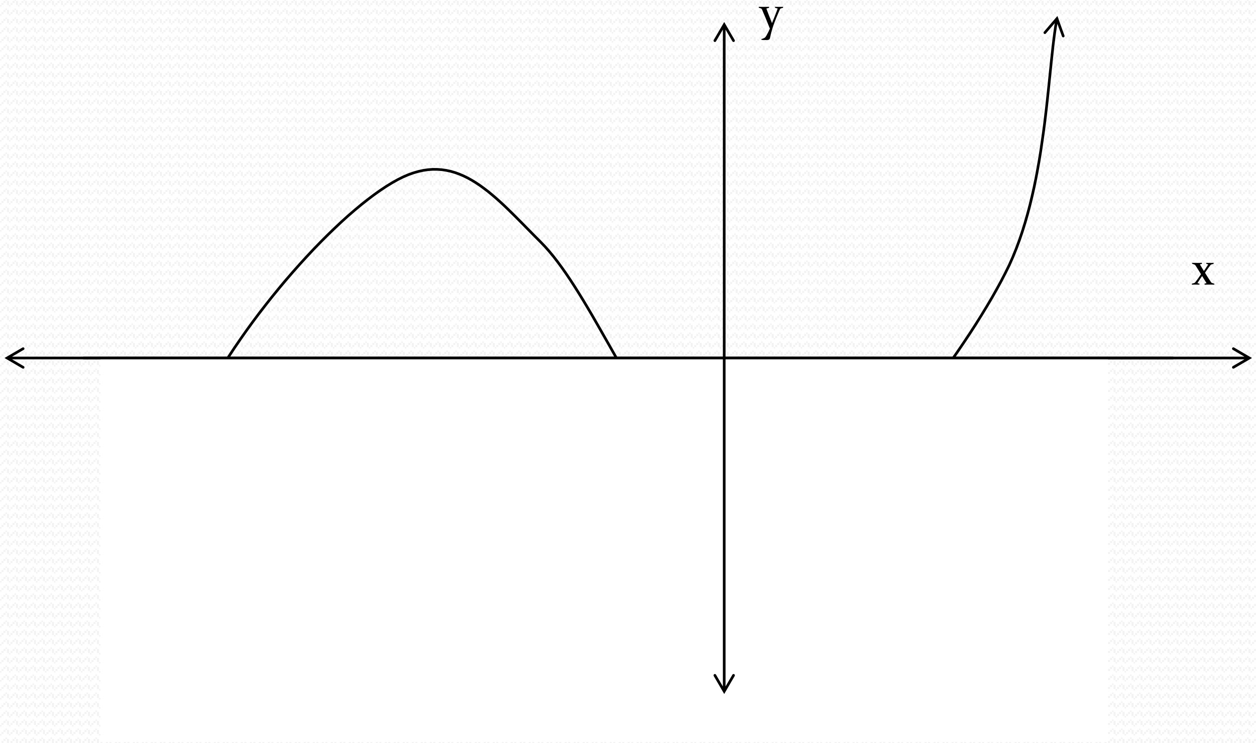
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



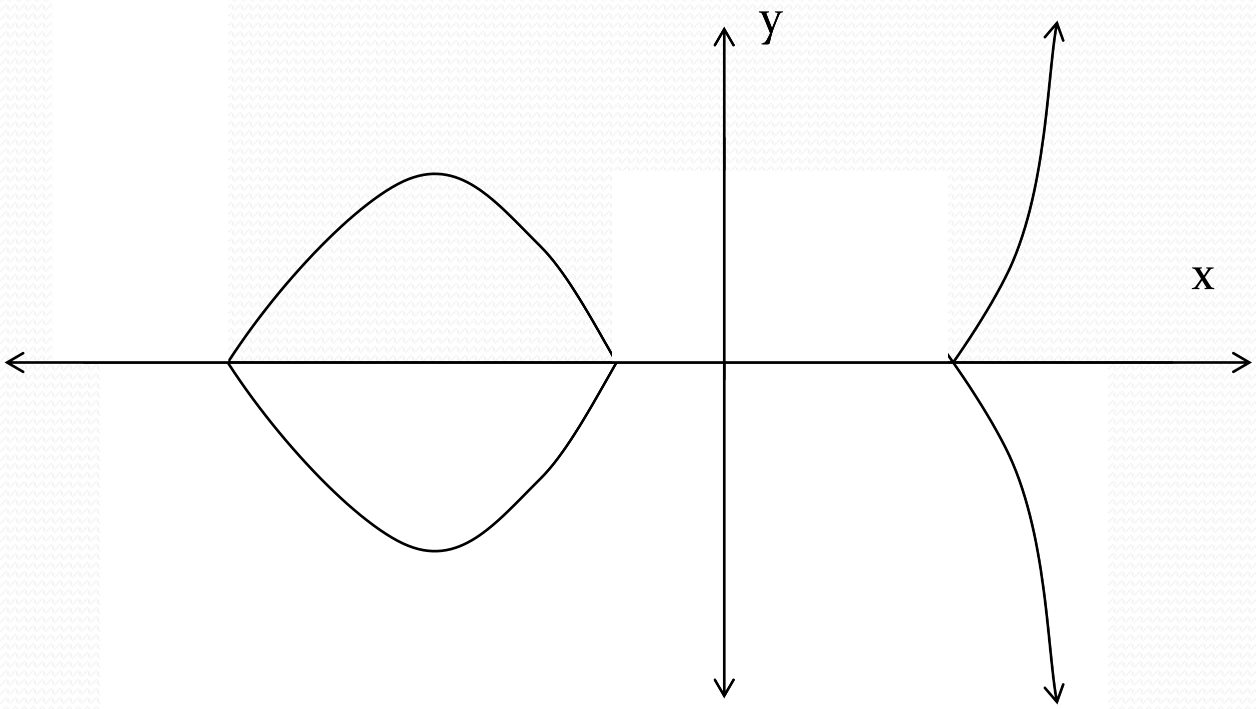
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



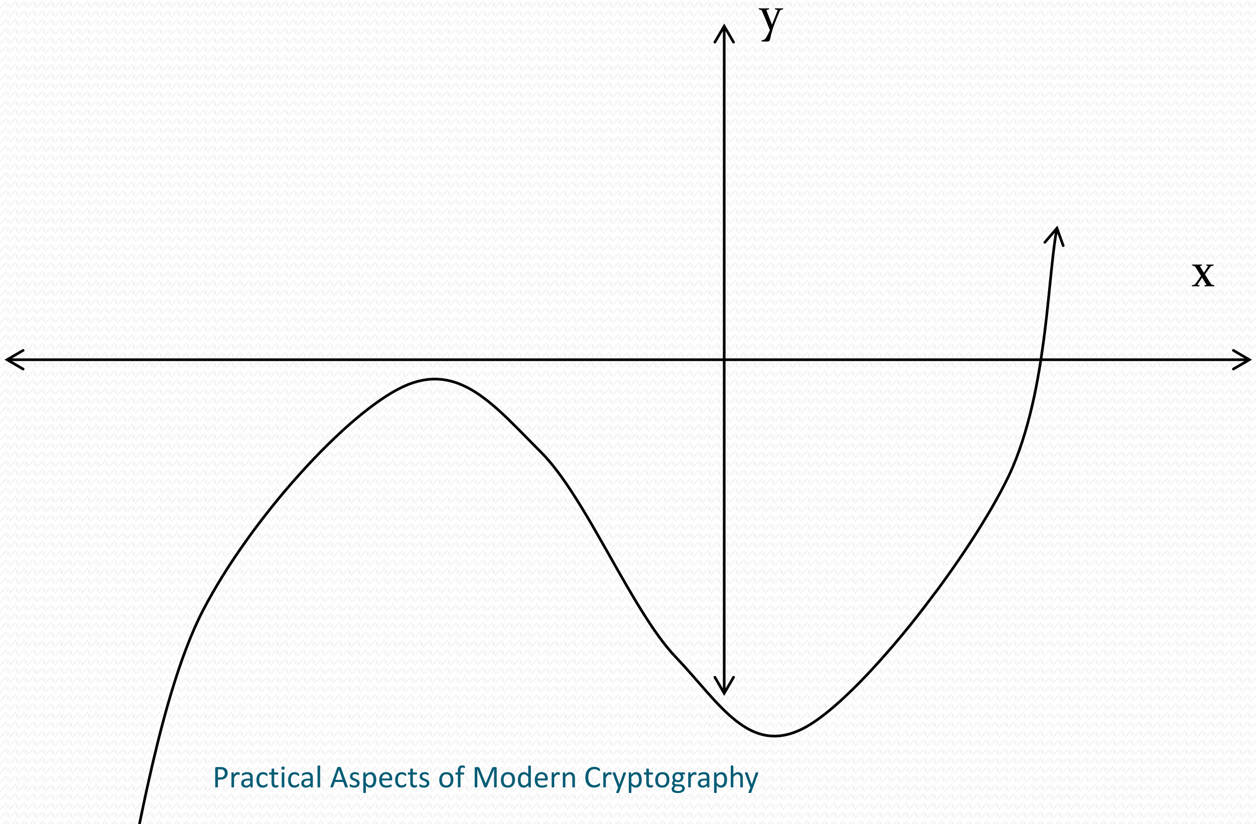
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



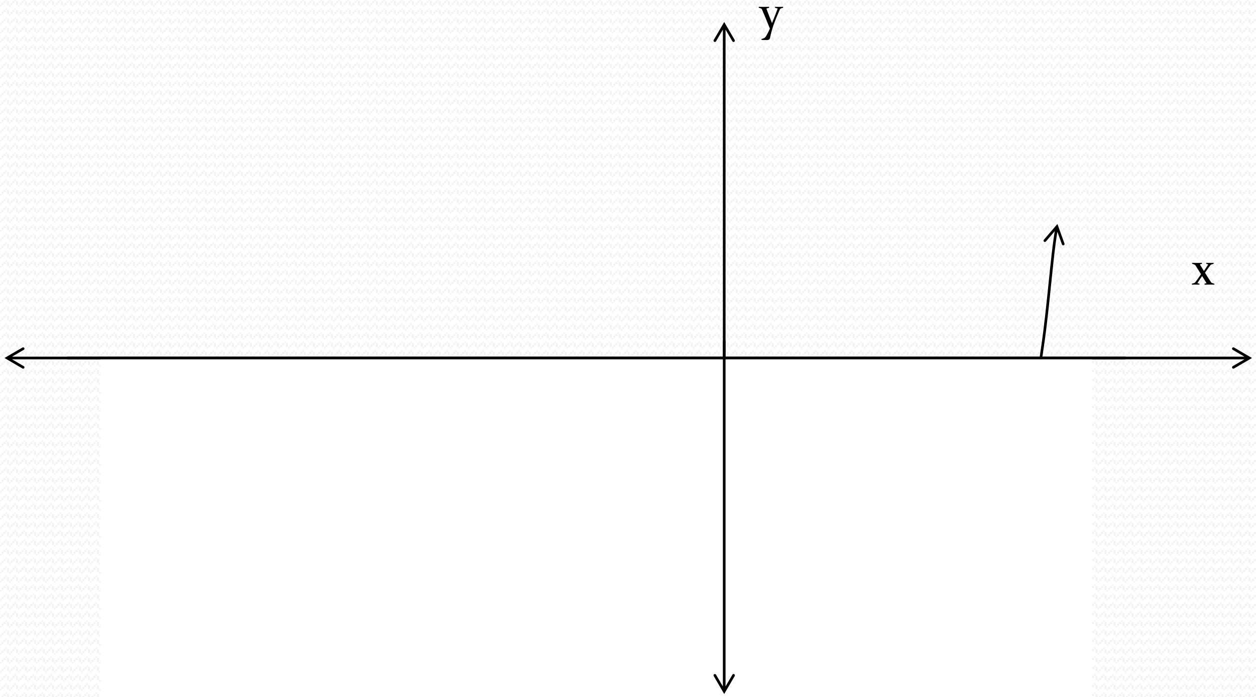
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



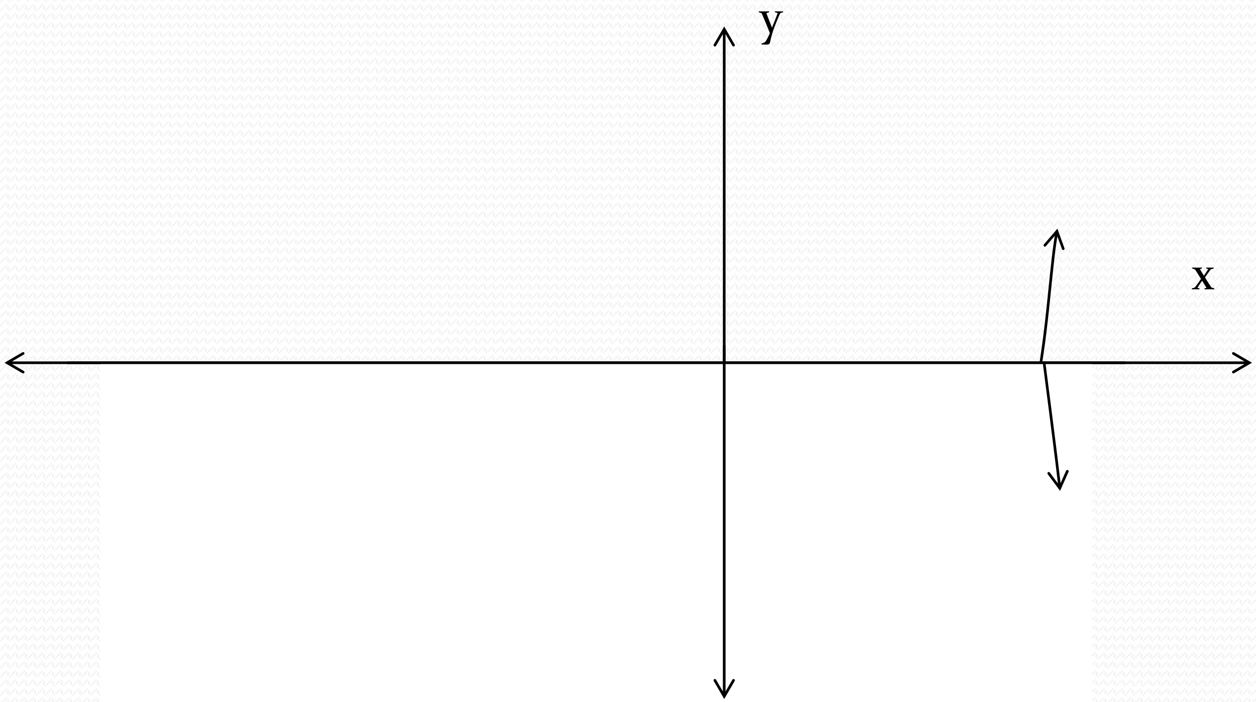
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



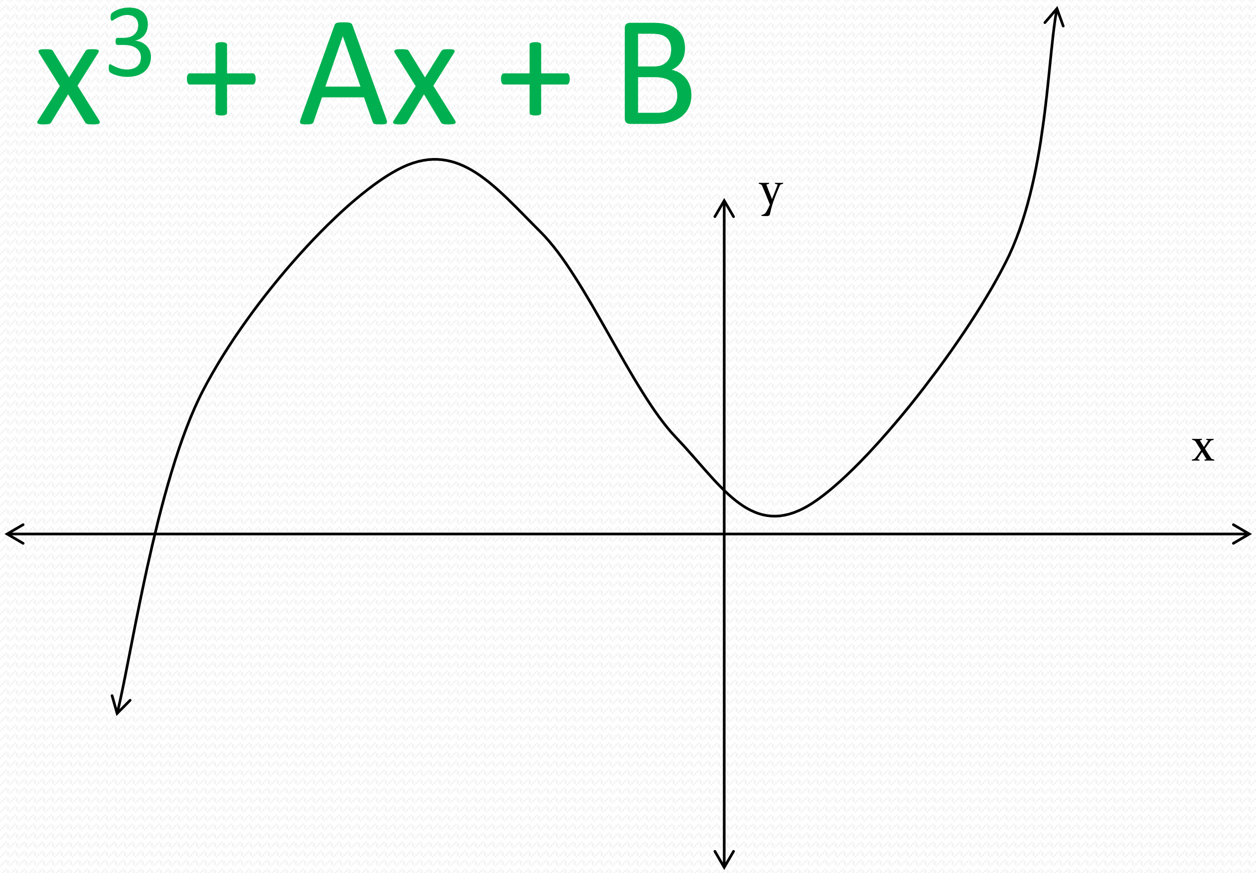
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



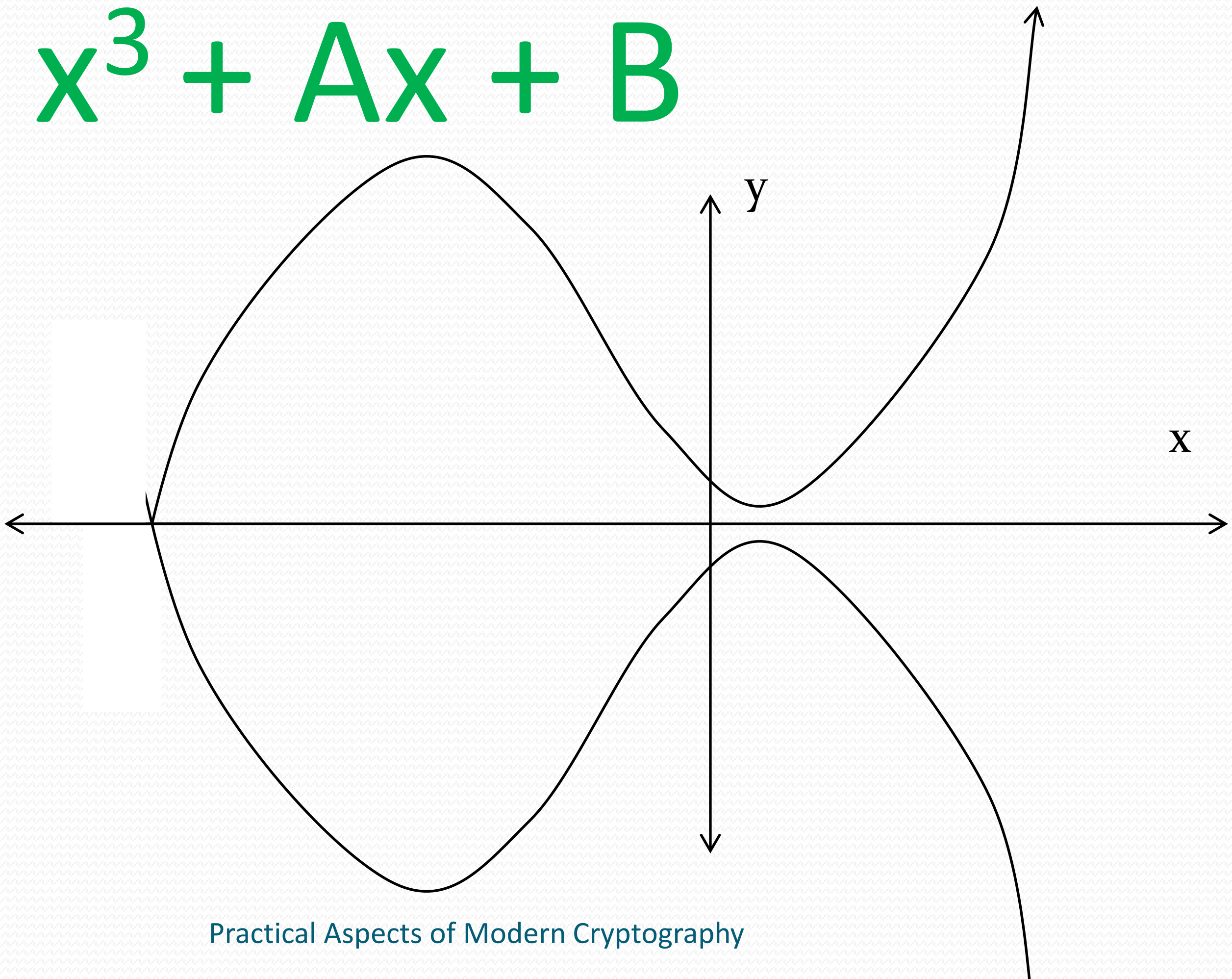
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



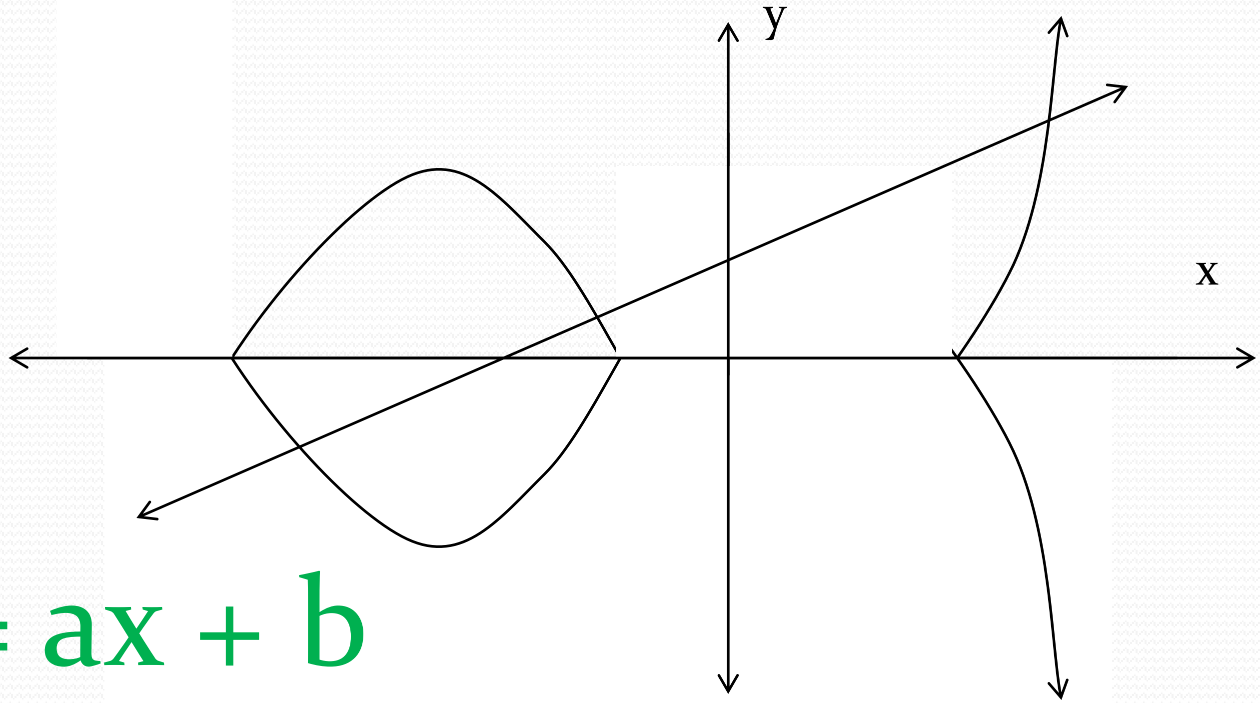
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



Elliptic Curves Intersecting Lines

$$y^2 = x^3 + Ax + B$$



$$y = ax + b$$

Elliptic Curves Intersecting Lines

Non-vertical Lines

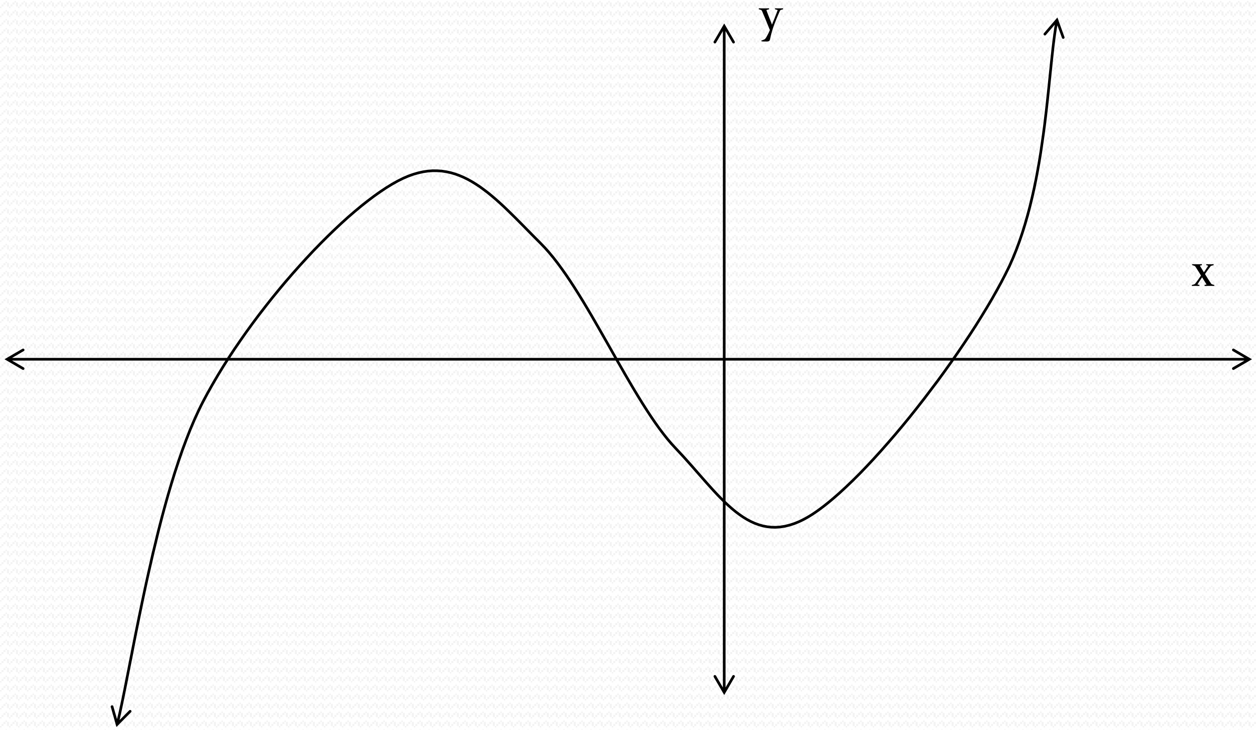
$$\begin{cases} y^2 = x^3 + Ax + B \\ y = ax + b \end{cases}$$

$$(ax + b)^2 = x^3 + Ax + B$$

$$x^3 + A'x^2 + B'x + C' = 0$$

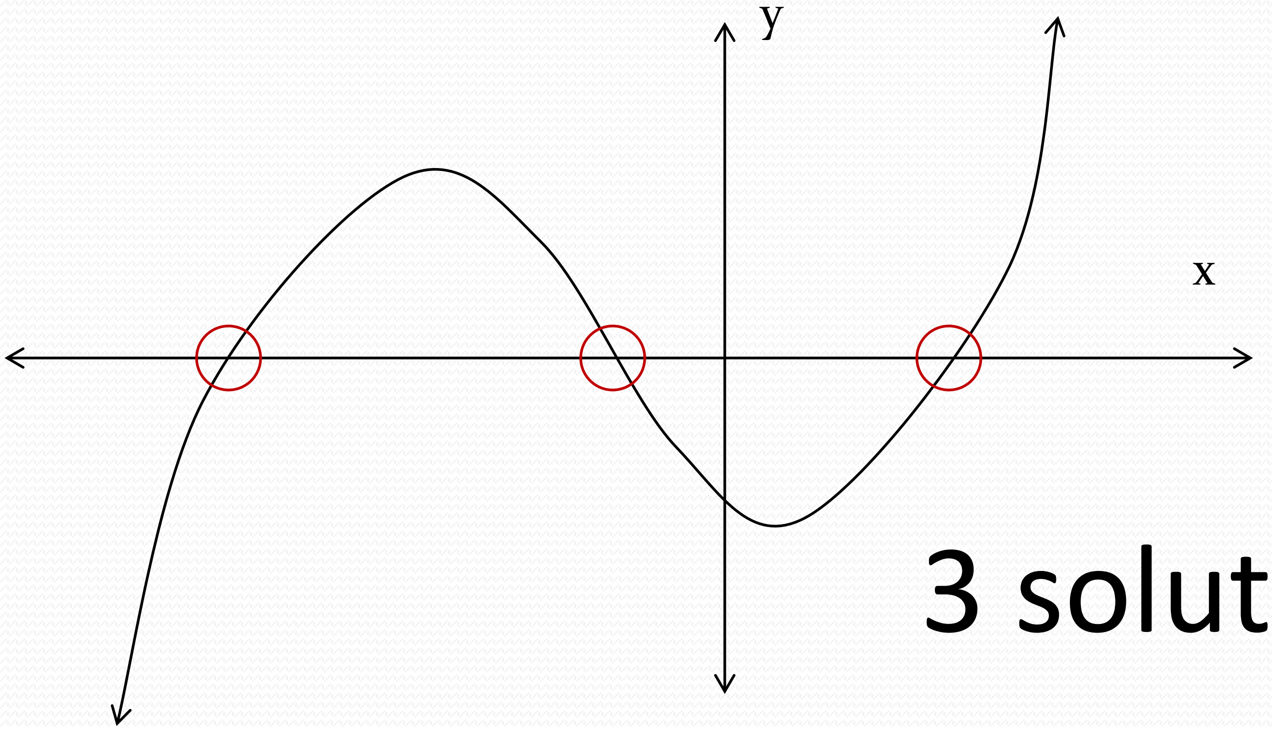
Elliptic Curves Intersecting Lines

$$x^3 + A'x^2 + B'x + C' = 0$$



Elliptic Curves Intersecting Lines

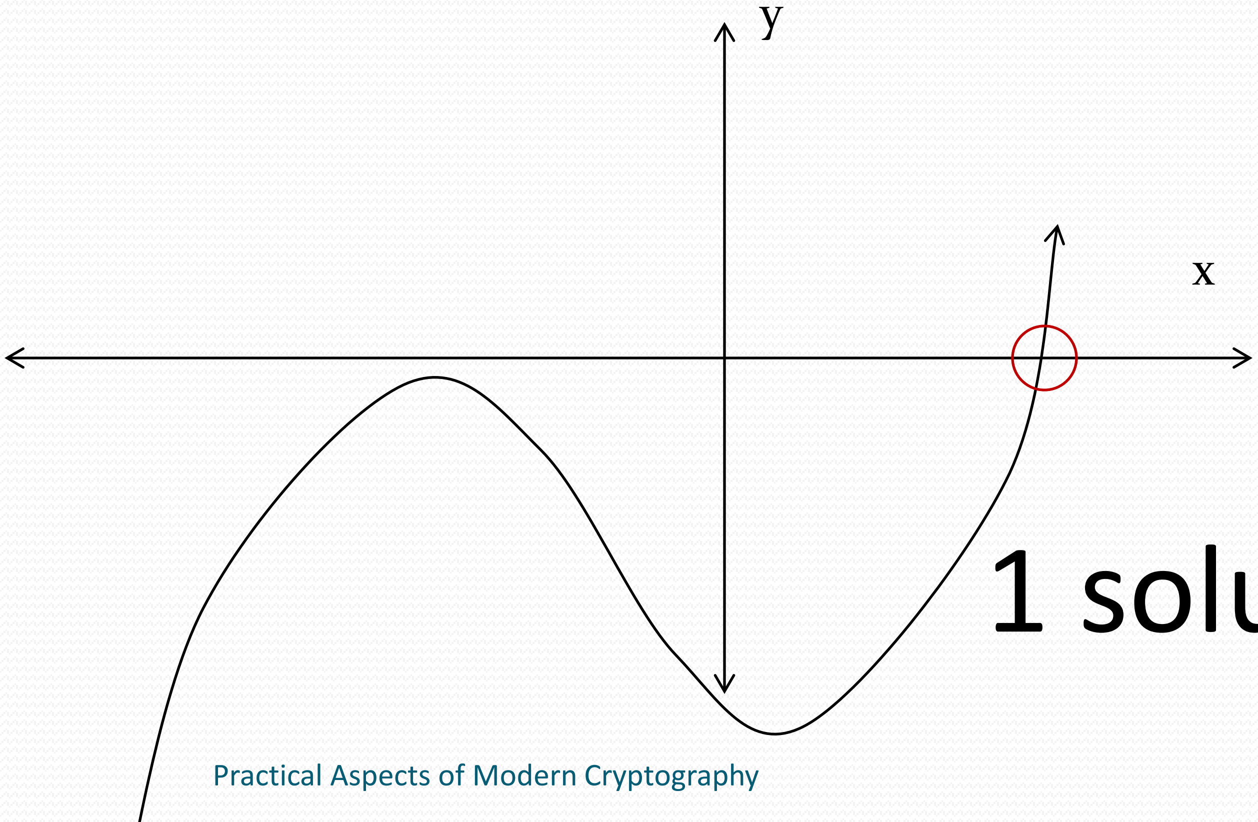
$$x^3 + A'x^2 + B'x + C' = 0$$



3 solutions

Elliptic Curves Intersecting Lines

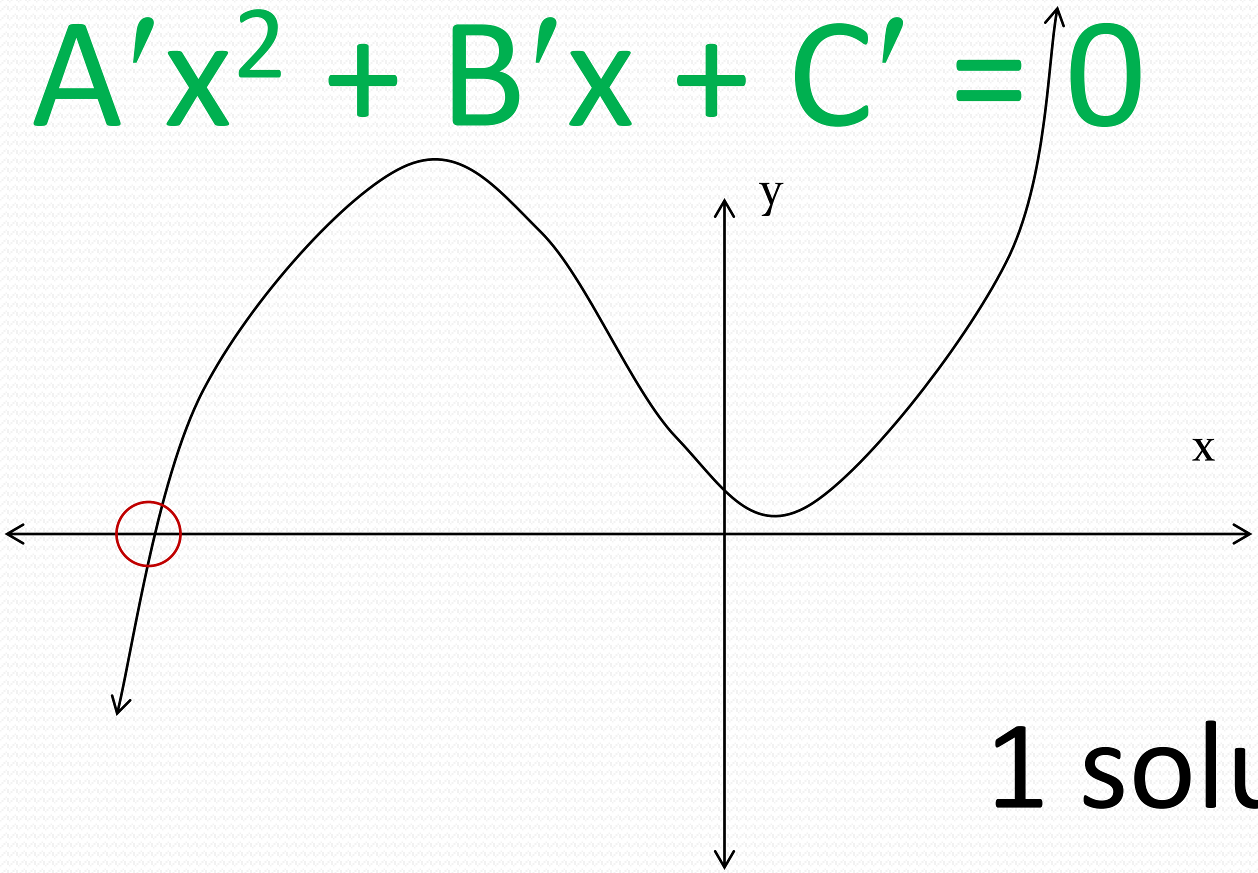
$$x^3 + A'x^2 + B'x + C' = 0$$



1 solution

Elliptic Curves Intersecting Lines

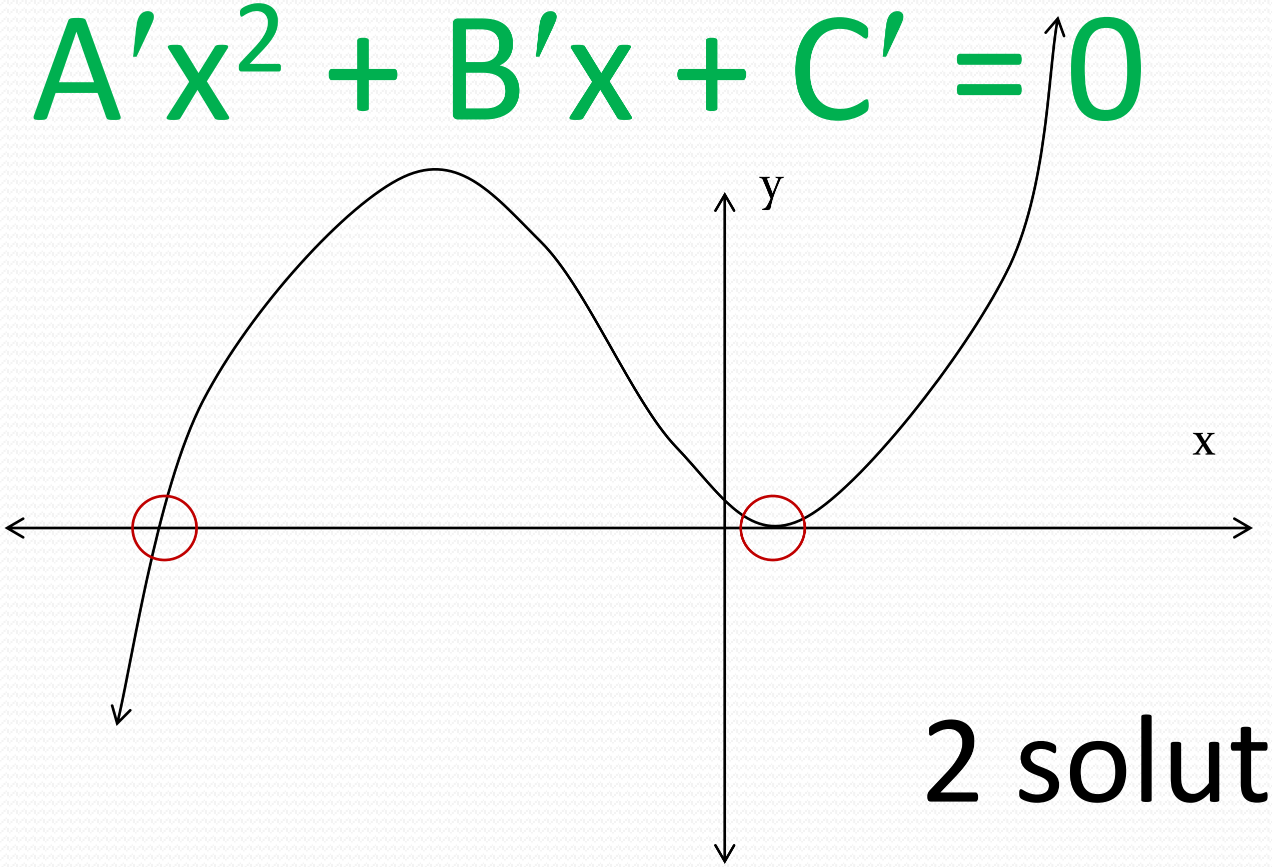
$$x^3 + A'x^2 + B'x + C' = 0$$



1 solution

Elliptic Curves Intersecting Lines

$$x^3 + A'x^2 + B'x + C' = 0$$



2 solutions

Elliptic Curves Intersecting Lines

Non-vertical Lines

- 1 intersection point (typical case)
- 2 intersection points (tangent case)
- 3 intersection points (typical case)

Elliptic Curves Intersecting Lines

Vertical Lines

$$\begin{cases} y^2 = x^3 + Ax + B \\ x = c \end{cases}$$

$$y^2 = c^3 + Ac + B$$

$$y^2 = C'$$

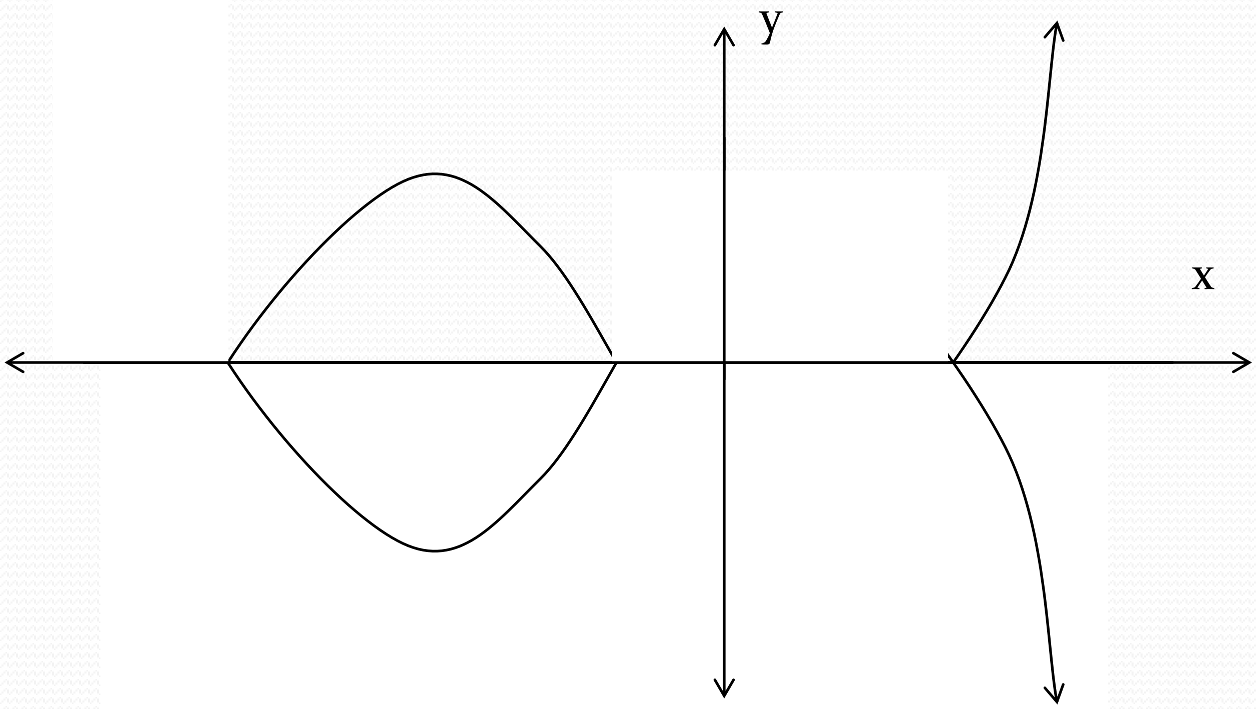
Elliptic Curves Intersecting Lines

Vertical Lines

- 0 intersection point (typical case)
- 1 intersection points (tangent case)
- 2 intersection points (typical case)

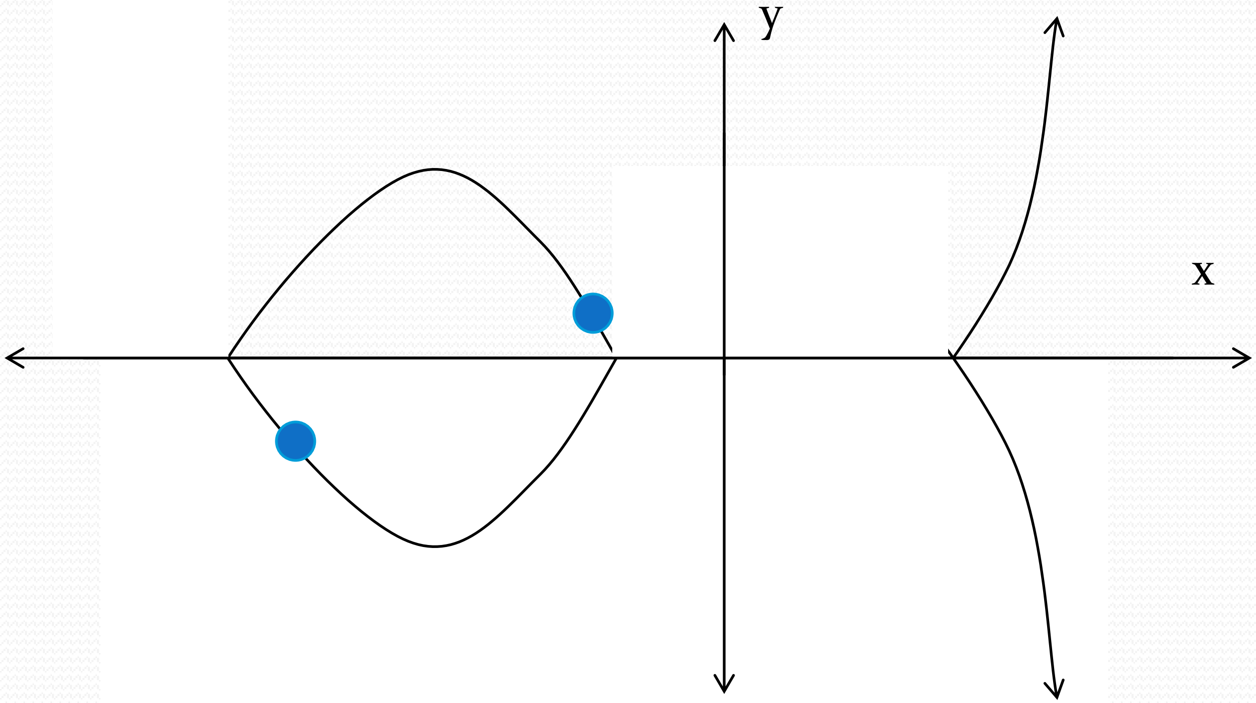
Elliptic Groups

$$y^2 = x^3 + Ax + B$$



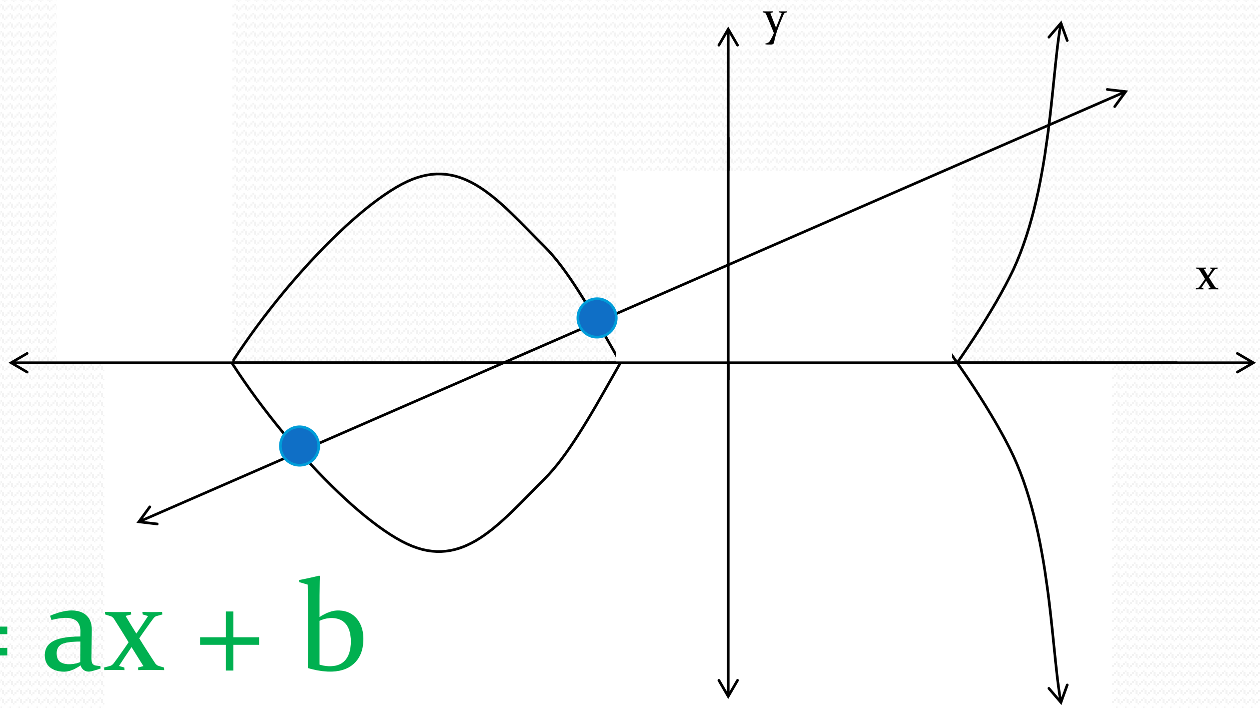
Elliptic Groups

$$y^2 = x^3 + Ax + B$$



Elliptic Groups

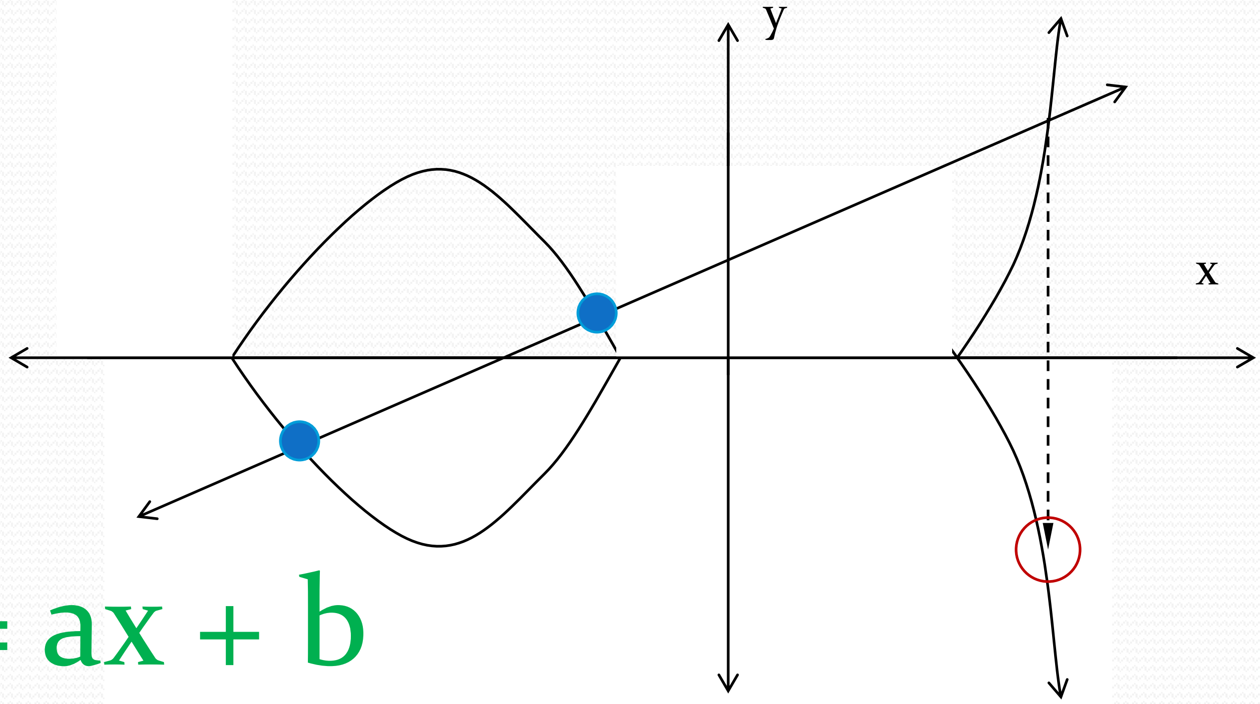
$$y^2 = x^3 + Ax + B$$



$$y = ax + b$$

Elliptic Groups

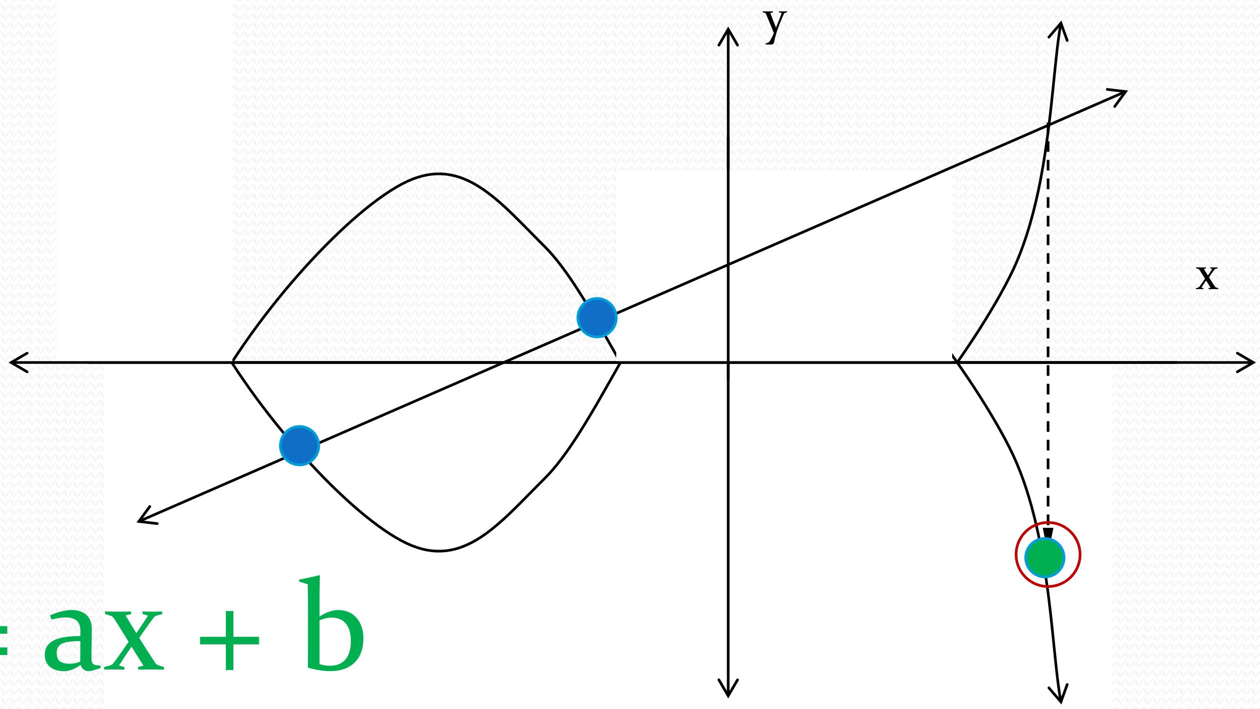
$$y^2 = x^3 + Ax + B$$



$$y = ax + b$$

Elliptic Groups

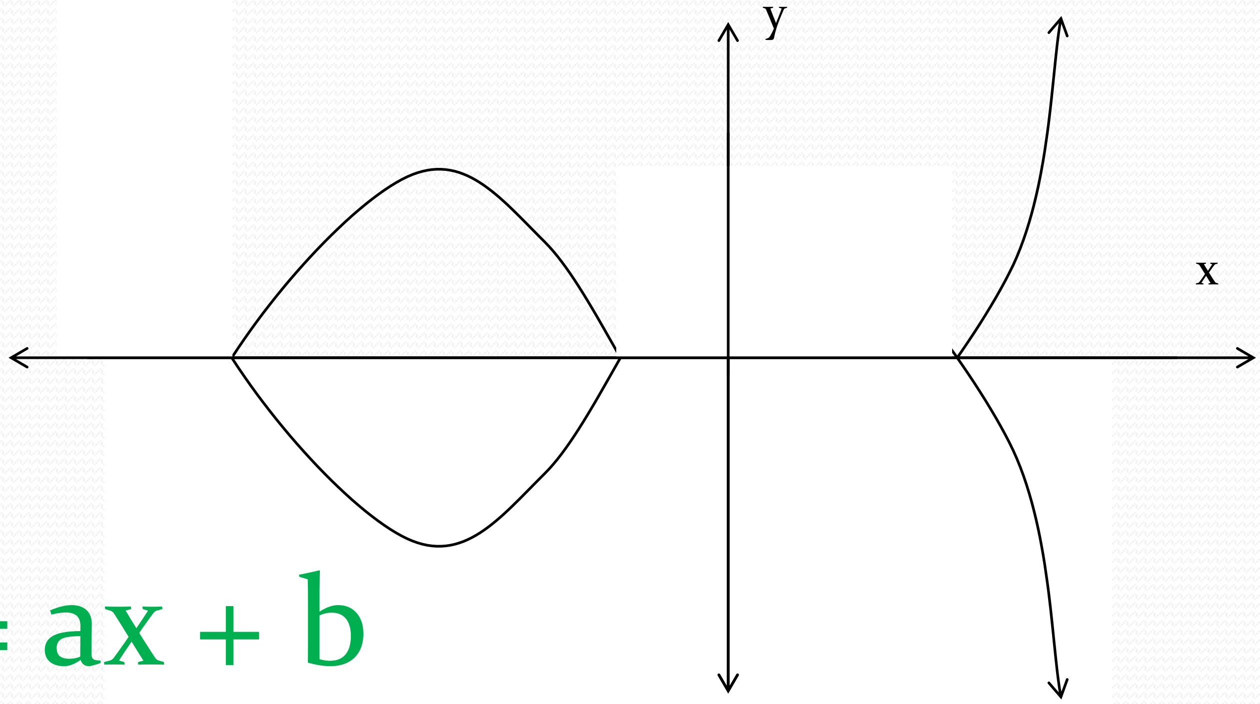
$$y^2 = x^3 + Ax + B$$



$$y = ax + b$$

Elliptic Groups

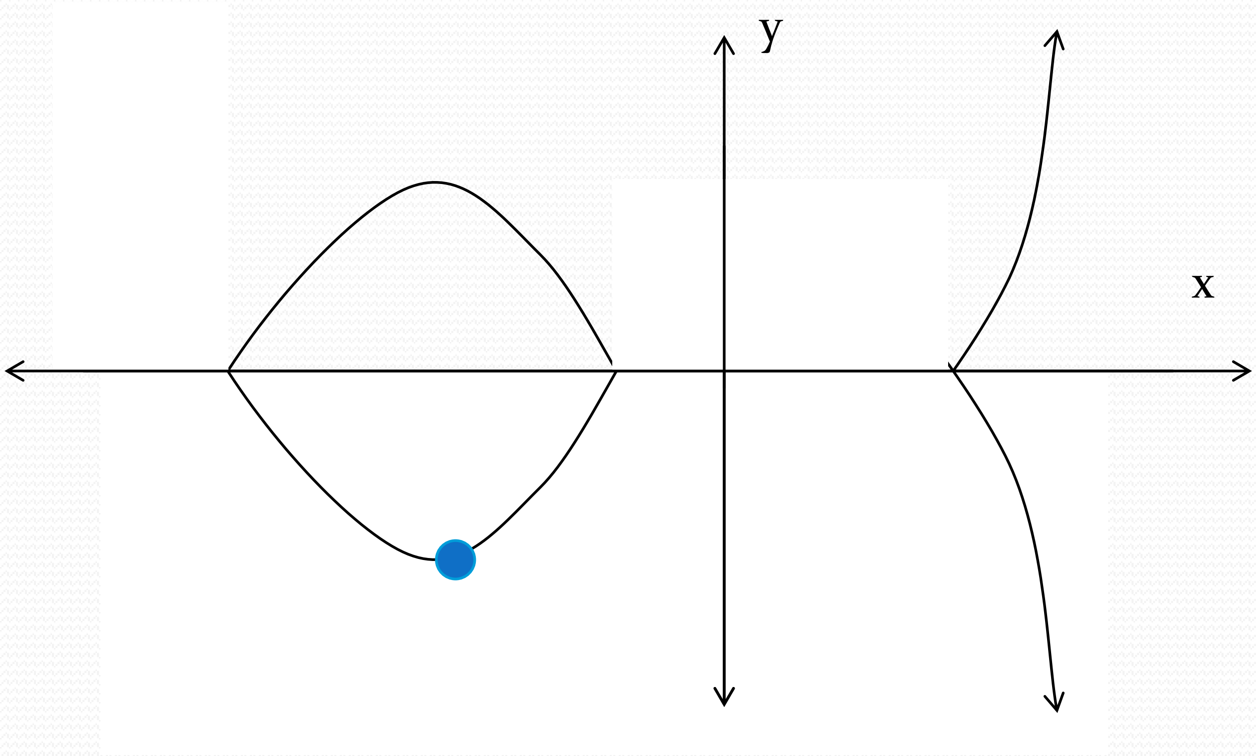
$$y^2 = x^3 + Ax + B$$



$$y = ax + b$$

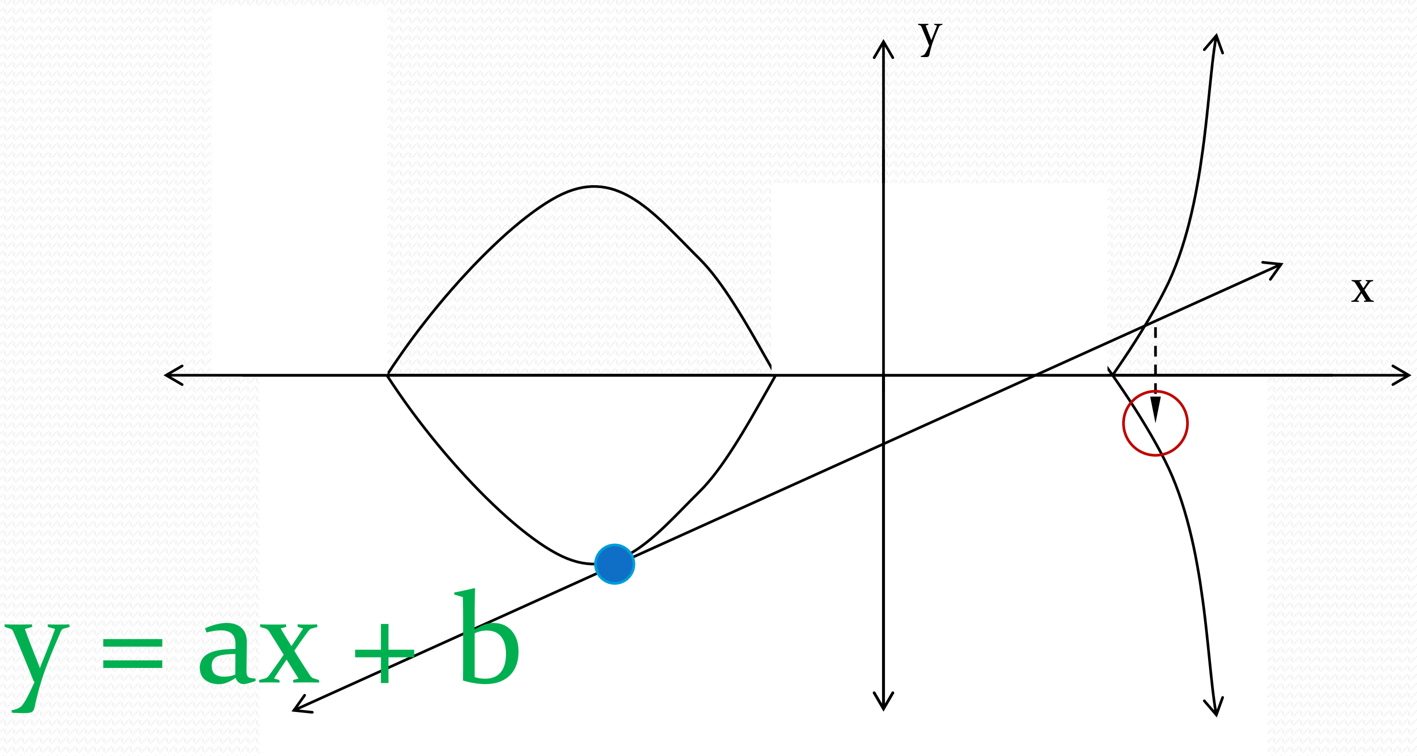
Elliptic Groups

$y^2 = x^3 + Ax + B$



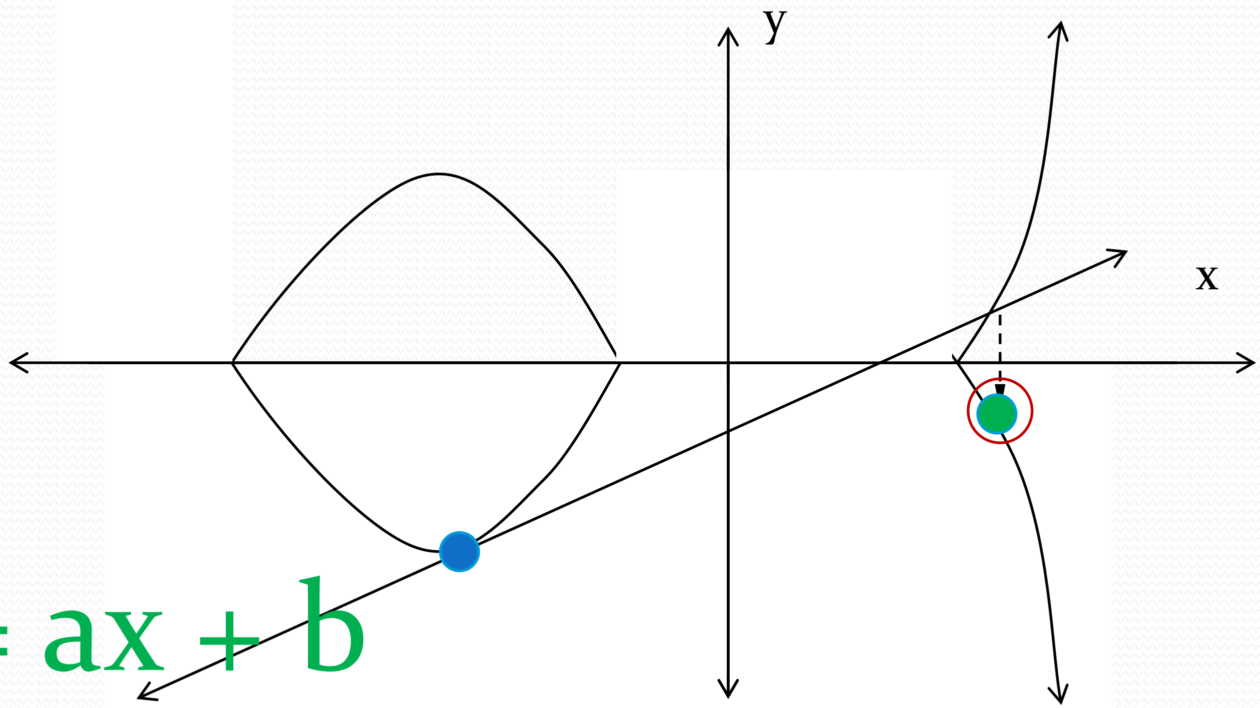
Elliptic Groups

$$y^2 = x^3 + Ax + B$$



Elliptic Groups

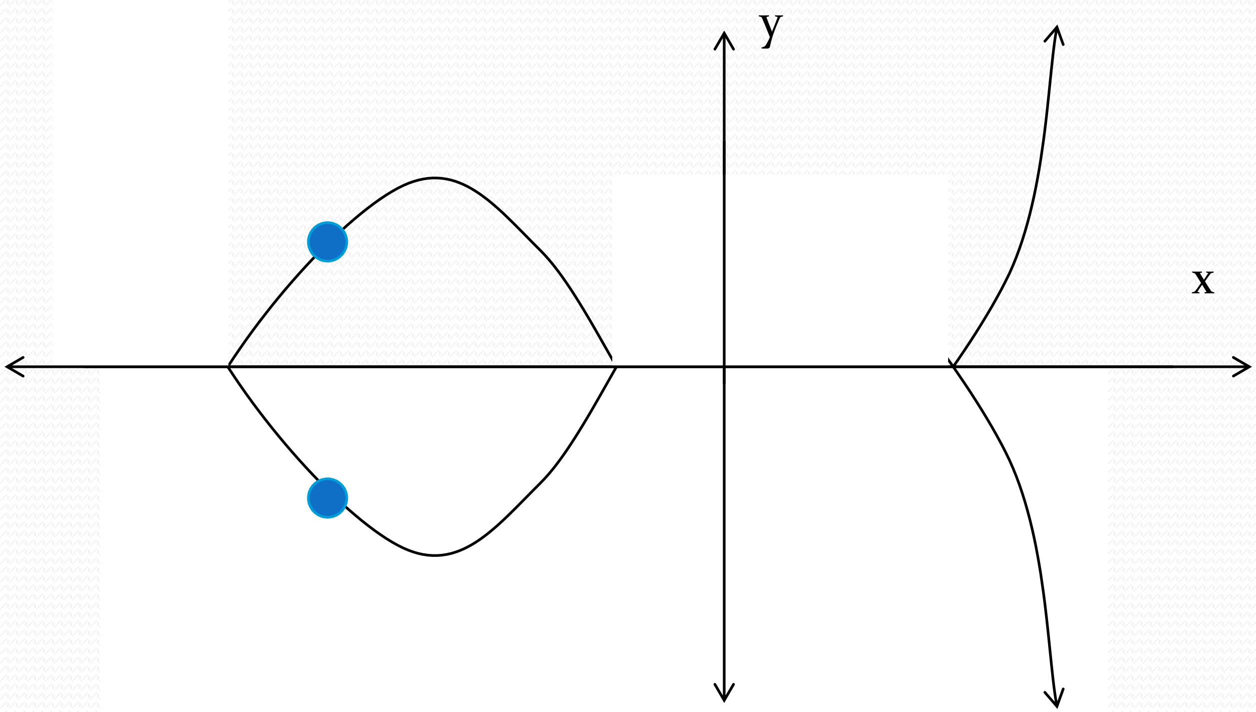
$$y^2 = x^3 + Ax + B$$



$$y = ax + b$$

Elliptic Groups

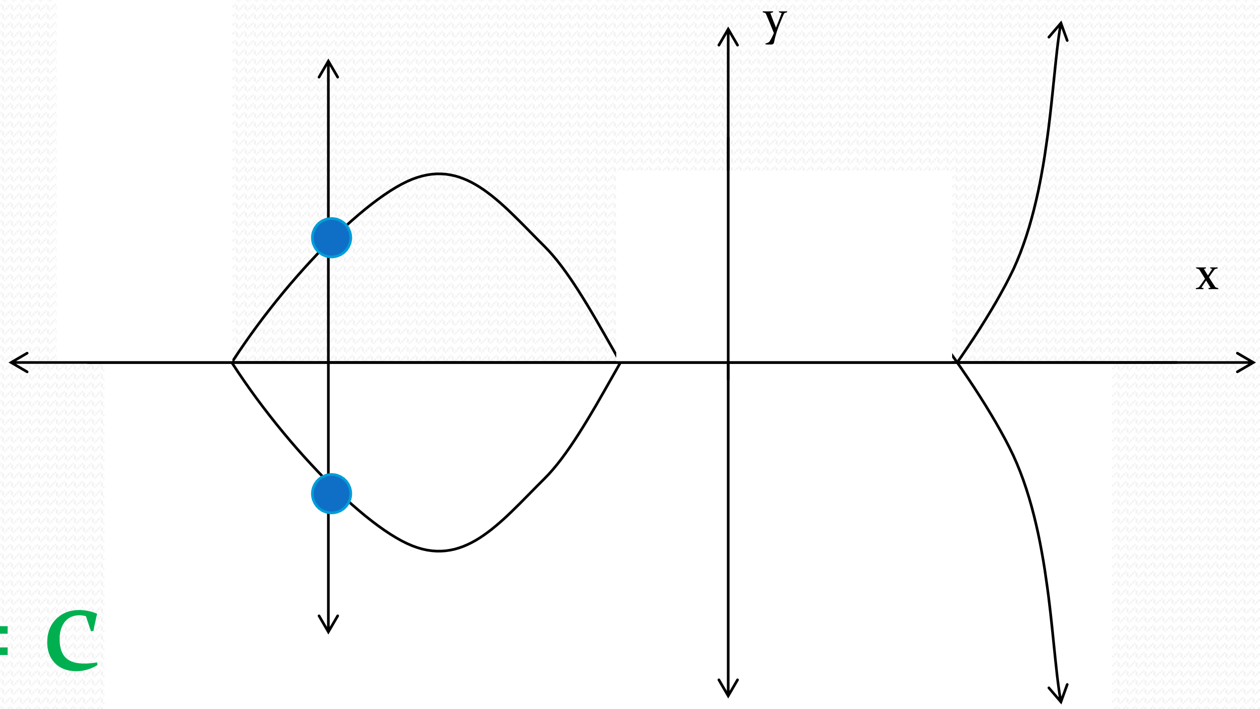
$$y^2 = x^3 + Ax + B$$



Elliptic Groups

$$y^2 = x^3 + Ax + B$$

$$x = c$$



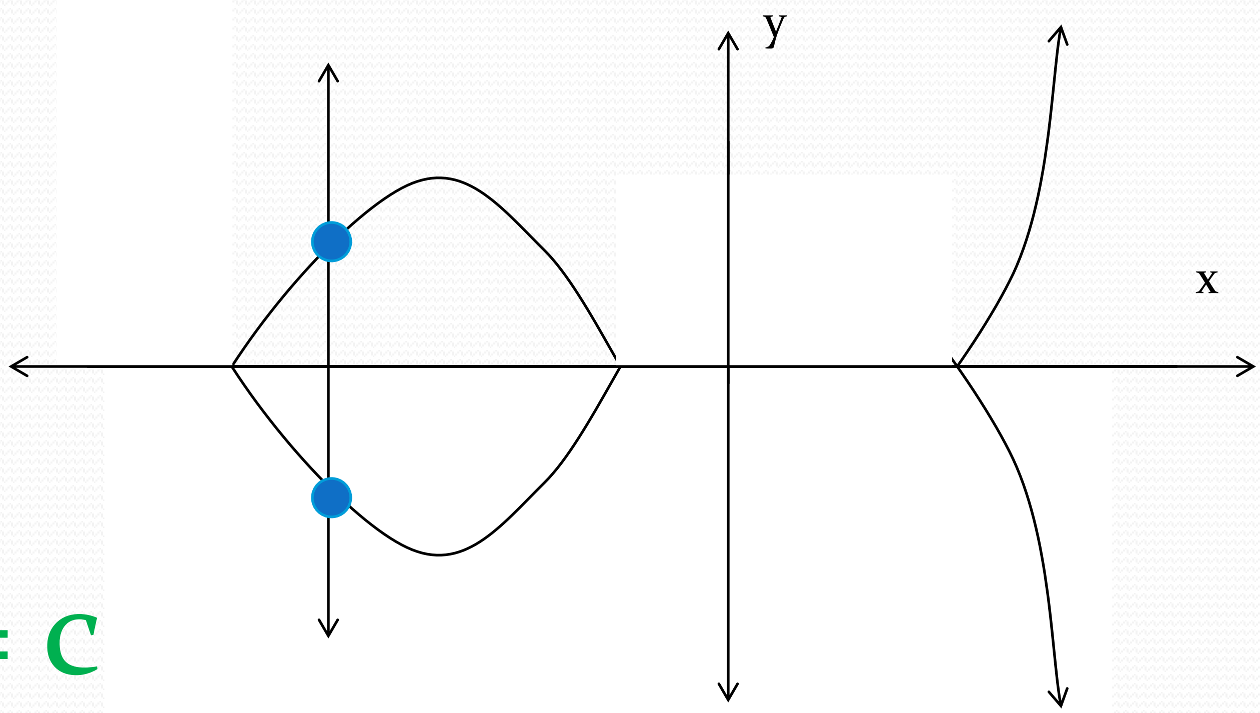
Elliptic Groups

- Add an “artificial” point I to handle the vertical line case.
- This point I also serves as the group identity value.

Elliptic Groups

$$y^2 = x^3 + Ax + B$$

$$x = c$$



Elliptic Groups

$$(x_1,y_1) \times (x_2,y_2) = (x_3,y_3)$$

$$x_3 = ((y_2-y_1)/(x_2-x_1))^2 - x_1 - x_2$$
$$y_3 = -y_1 + ((y_2-y_1)/(x_2-x_1))\,(x_1-x_3)$$

when $x_1 \neq x_2$

Elliptic Groups

$$(x_1,y_1) \times (x_2,y_2) = (x_3,y_3)$$

$$x_3 = ((3x_1{}^2+A)/(2y_1))^2 - 2x_1$$

$$y_3 = -y_1 + ((3x_1{}^2+A)/(2y_1))\ (x_1 - x_3)$$

when $x_1 = x_2$ and $y_1 = y_2 \neq 0$

Elliptic Groups

$$(x_1,y_1) \times (x_2,y_2) = I$$

when $x_1 = x_2$ but $y_1 \neq y_2$ or $y_1 = y_2 = 0$

$$(x_1,y_1) \times I = (x_1,y_1) = I \times (x_1,y_1)$$

$$I \times I = I$$

Finite Elliptic Groups

Finite Elliptic Groups

- The equations use basic arithmetic operations (addition, subtraction, multiplication, and division) on *real* values.

Finite Elliptic Groups

- The equations use basic arithmetic operations (addition, subtraction, multiplication, and division) on *real* values.
- But we know how to do modular operations, so we can do the same computations modulo a prime p.

The Elliptic Group $E_p(A, B)$

$$(x_1,y_1) \times (x_2,y_2) = (x_3,y_3)$$

$$x_3 = ((y_2 - y_1)/(x_2 - x_1))^2 - x_1 - x_2$$

$$y_3 = -y_1 + ((y_2 - y_1)/(x_2 - x_1))\,(x_1 - x_3)$$

when $x_1 \neq x_2$

The Elliptic Group $E_p(A, B)$

$$(x_1,y_1) \times (x_2,y_2) = (x_3,y_3)$$

$$x_3 = ((y_2 - y_1)/(x_2 - x_1))^2 - x_1 - x_2 \bmod p$$
$$y_3 = -y_1 + ((y_2 - y_1)/(x_2 - x_1))\,(x_1 - x_3) \bmod p$$

when $x_1 \neq x_2$

The Elliptic Group $E_p(A, B)$

$$(x_1,y_1) \times (x_2,y_2) = (x_3,y_3)$$

$$x_3 = ((3x_1^2+A)/(2y_1))^2 - 2x_1$$
$$y_3 = -y_1 + ((3x_1^2+A)/(2y_1))\ (x_1 - x_3)$$

when $x_1 = x_2$ and $y_1 = y_2 \neq 0$

The Elliptic Group $E_p(A, B)$

$$(x_1,y_1) \times (x_2,y_2) = (x_3,y_3)$$

$$x_3 = ((3x_1^2+A)/(2y_1))^2 - 2x_1 \bmod p$$
$$y_3 = -y_1 + ((3x_1^2+A)/(2y_1))\,(x_1 - x_3) \bmod p$$

when $x_1 = x_2$ and $y_1 = y_2 \neq 0$

The Elliptic Group $E_p(A, B)$

$$(x_1,y_1) \times (x_2,y_2) = I$$

when $x_1 = x_2$ but $y_1 \neq y_2$ or $y_1 = y_2 = 0$

$$(x_1,y_1) \times I = (x_1,y_1) = I \times (x_1,y_1)$$

$$I \times I = I$$

The Fundamental Equation

$$Z=Y^X \bmod N$$

The Fundamental Equation

$$Z = Y^X \text{ in } E_p(A,B)$$

The Fundamental Equation

$$Z = Y^X \text{ in } E_p(A,B)$$

When Z is unknown, it can be efficiently computed by repeated squaring.

The Fundamental Equation

$$Z = Y^X \text{ in } E_p(A,B)$$

When X is unknown, this version of the discrete logarithm is believed to be quite hard to solve.

The Fundamental Equation

$$Z=Y^X \text{ in } E_p(A,B)$$

When Y is unknown, it *can* be efficiently computed by “sophisticated” means.

Diffie-Hellman Key Exchange

Alice

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

Bob

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

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

Diffie-Hellman Key Exchange

Alice

- Randomly select a large integer a and send $A = Y^a$ in E_p.
- Compute the key $K = B^a$ in E_p.

Bob

- Randomly select a large integer b and send $B = Y^b$ in E_p.
- Compute the key $K = A^b$ in E_p.

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

DSA on Elliptic Curves

DSA on Elliptic Curves

- Almost identical to DSA over the integers.

DSA on Elliptic Curves

- Almost identical to DSA over the integers.
- Replace operations mod p and q with operations in E_p and E_q.

Why use Elliptic Curves?

Why use Elliptic Curves?

- The best *currently known* algorithm for EC discrete logarithms would take about as long to find a 160-bit EC discrete log as the best *currently known* algorithm for integer discrete logarithms would take to find a 1024-bit discrete log.

Why use Elliptic Curves?

- The best *currently known* algorithm for EC discrete logarithms would take about as long to find a 160-bit EC discrete log as the best *currently known* algorithm for integer discrete logarithms would take to find a 1024-bit discrete log.
- 160-bit EC algorithms are somewhat faster and use shorter keys than 1024-bit "traditional" algorithms.

Why *not* use Elliptic Curves?

Why *not* use Elliptic Curves?

- EC discrete logarithms have been studied far less than integer discrete logarithms.

Why *not* use Elliptic Curves?

- EC discrete logarithms have been studied far less than integer discrete logarithms.
- Results have shown that a fundamental break in integer discrete logs would also yield a fundamental break in EC discrete logs, although the reverse may not be true.

Why *not* use Elliptic Curves?

- EC discrete logarithms have been studied far less than integer discrete logarithms.
- Results have shown that a fundamental break in integer discrete logs would also yield a fundamental break in EC discrete logs, although the reverse may not be true.
- Basic EC operations are more cumbersome than integer operations, so EC is only faster if the keys are *much* smaller.

Symmetric Cryptography

The Practical Side

For efficiency, one generally uses RSA (or another public-key algorithm) to transmit a private (symmetric) key.

The private *session* key is used to encrypt and authenticate any subsequent data.

Digital signatures are only used to sign a *digest* of the message.

One-Way Hash Functions

Generally, a *one-way hash function* is a function $H : \{0,1\}^* \rightarrow \{0,1\}^k$ (typically k is 128, 160, 256, 384, or 512) such that given an input value x, one cannot find a value $x' \neq x$ such $H(x) = H(x')$.

One-Way Hash Functions

There are many measures for one-way hashes.

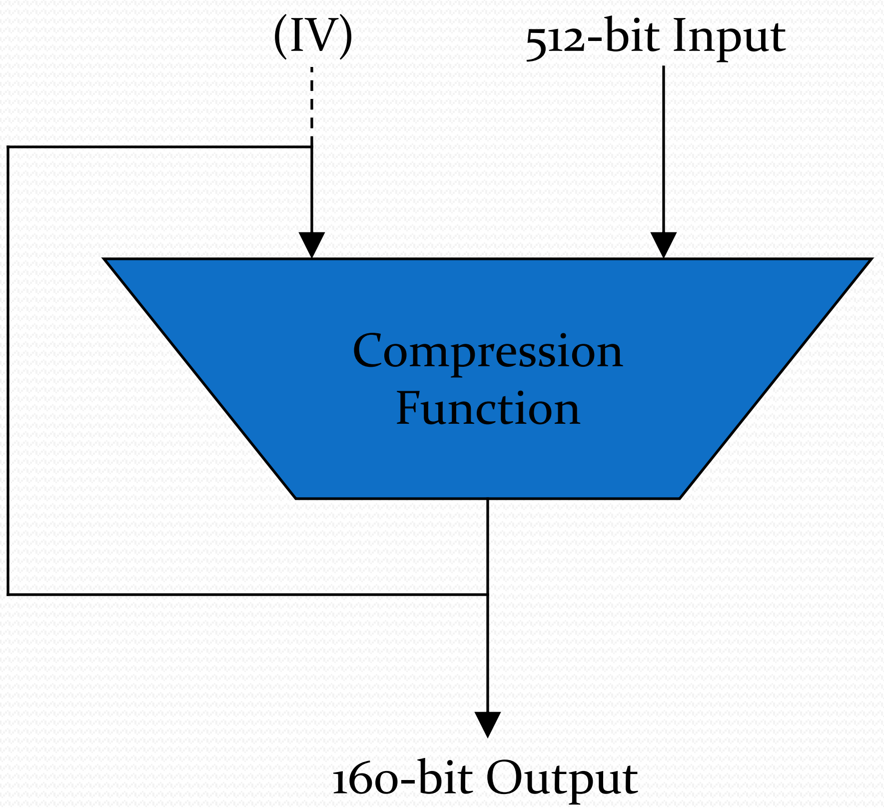
- Non-invertability: given y, it's difficult to find any x such that H(x) = y.
- Collision-intractability: one cannot find a pair of values $x' \neq x$ such that $H(x) = H(x')$.

One-Way Hash Functions

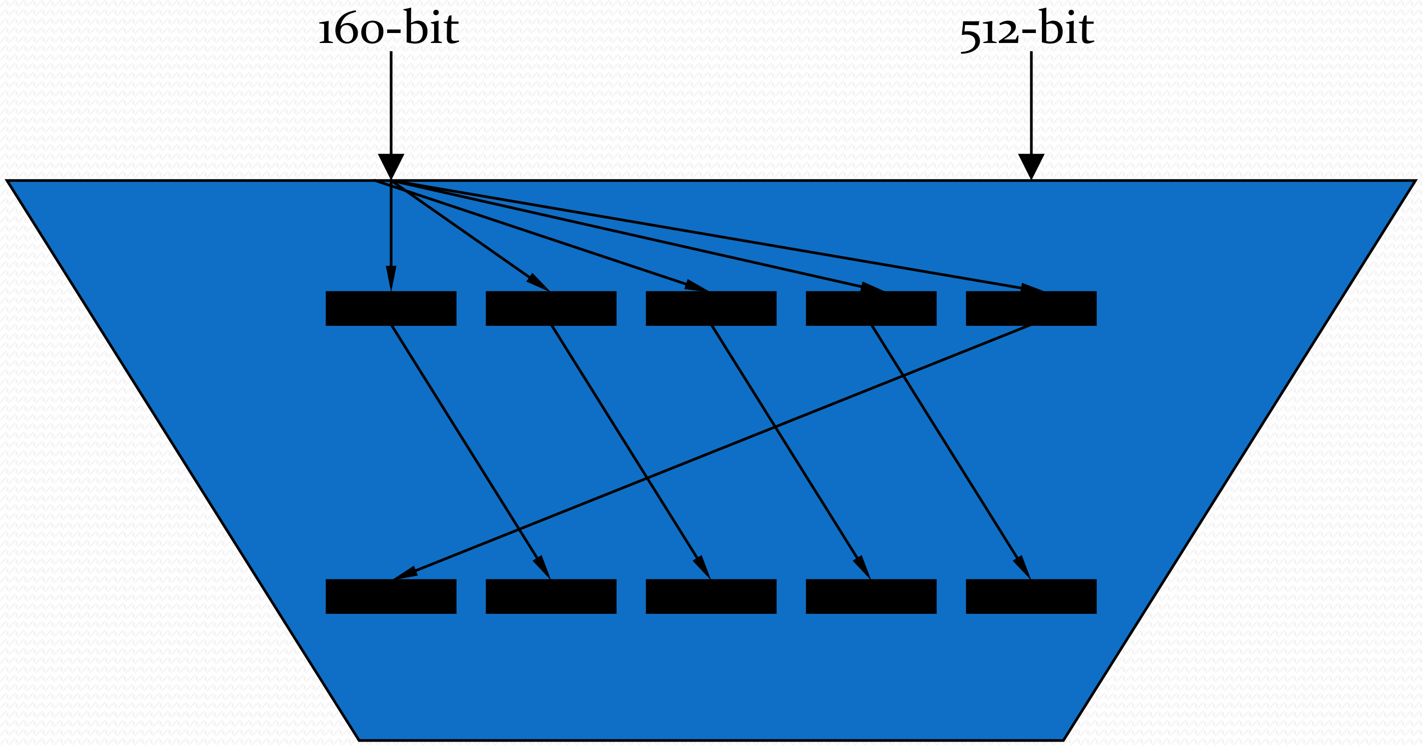
- When using a stream cipher, a hash of the message can be appended to ensure integrity. [Message Authentication Code]

- When forming a digital signature, the signature need only be applied to a hash of the message. [Message Digest]

A Cryptographic Hash: SHA-1

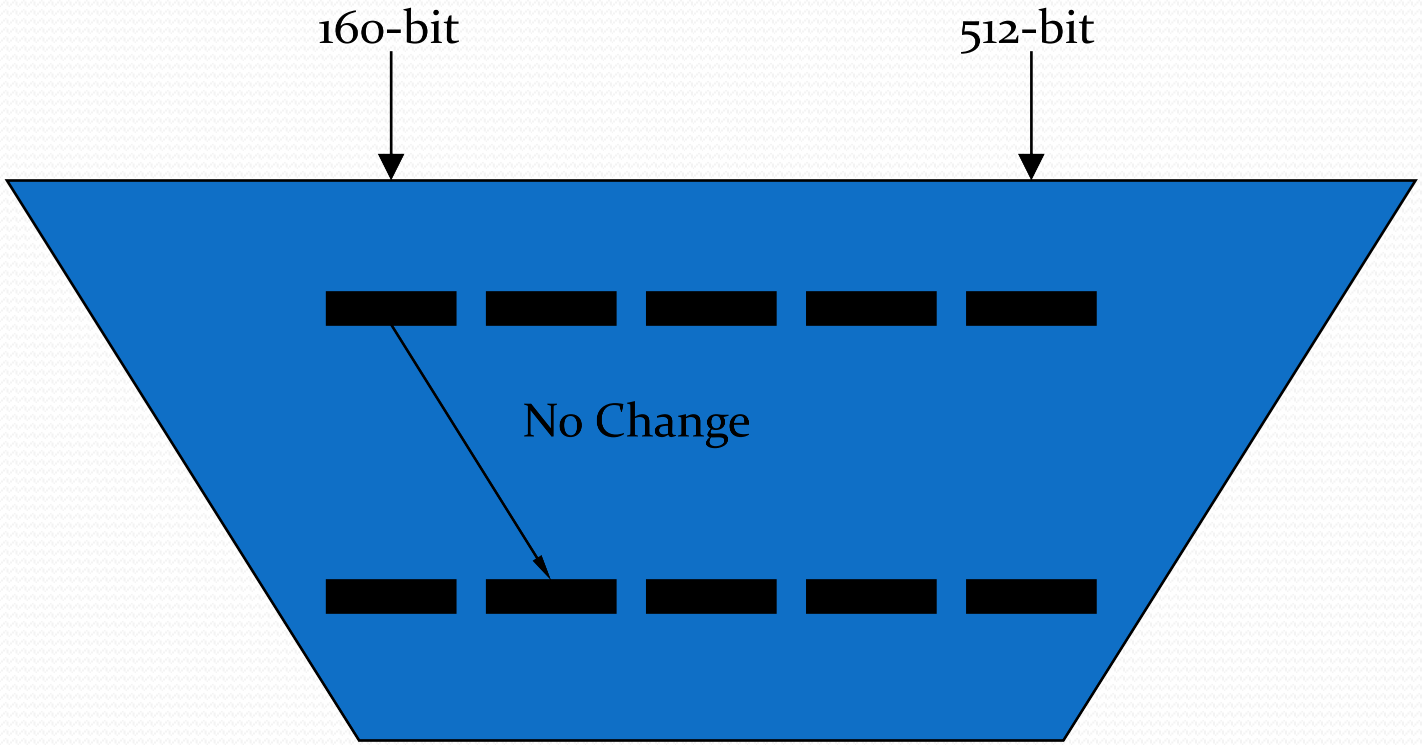


A Cryptographic Hash: SHA-1



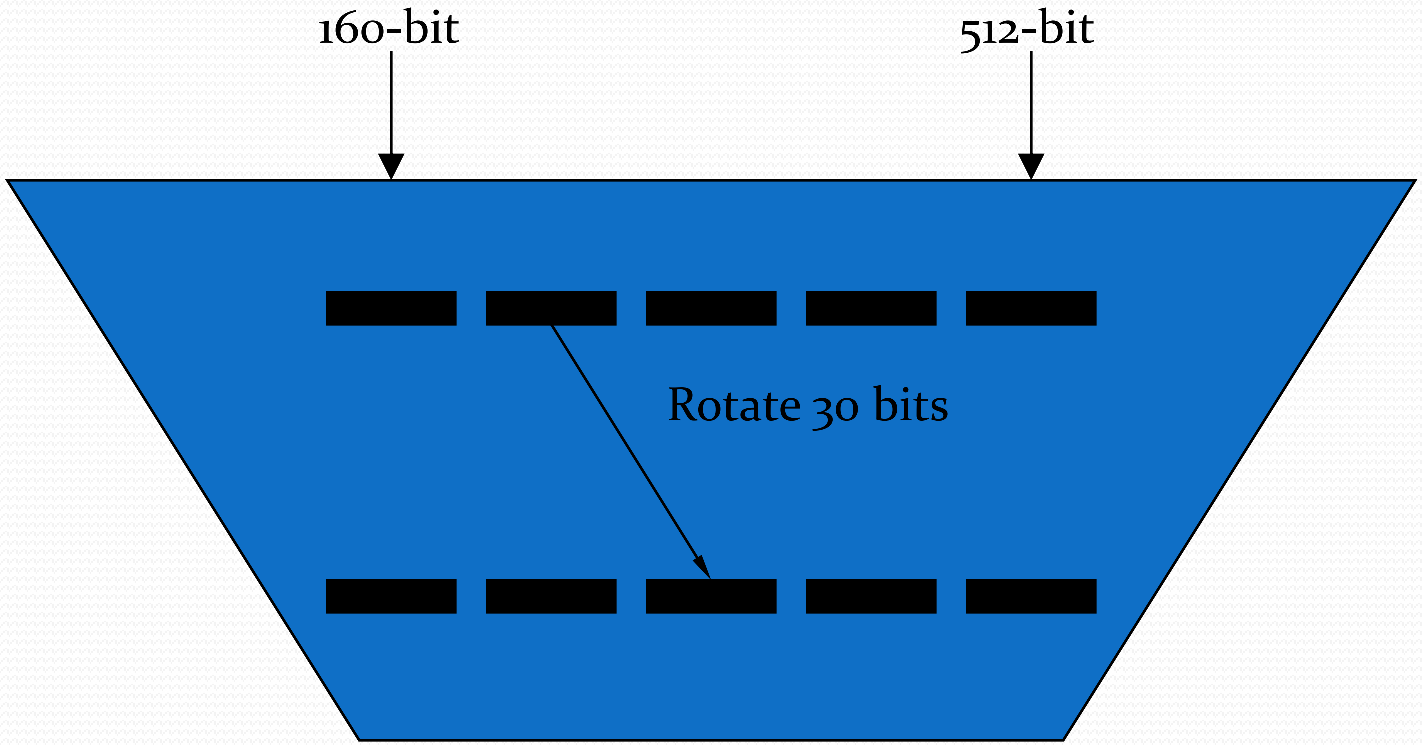
One of 80 rounds

A Cryptographic Hash: SHA-1



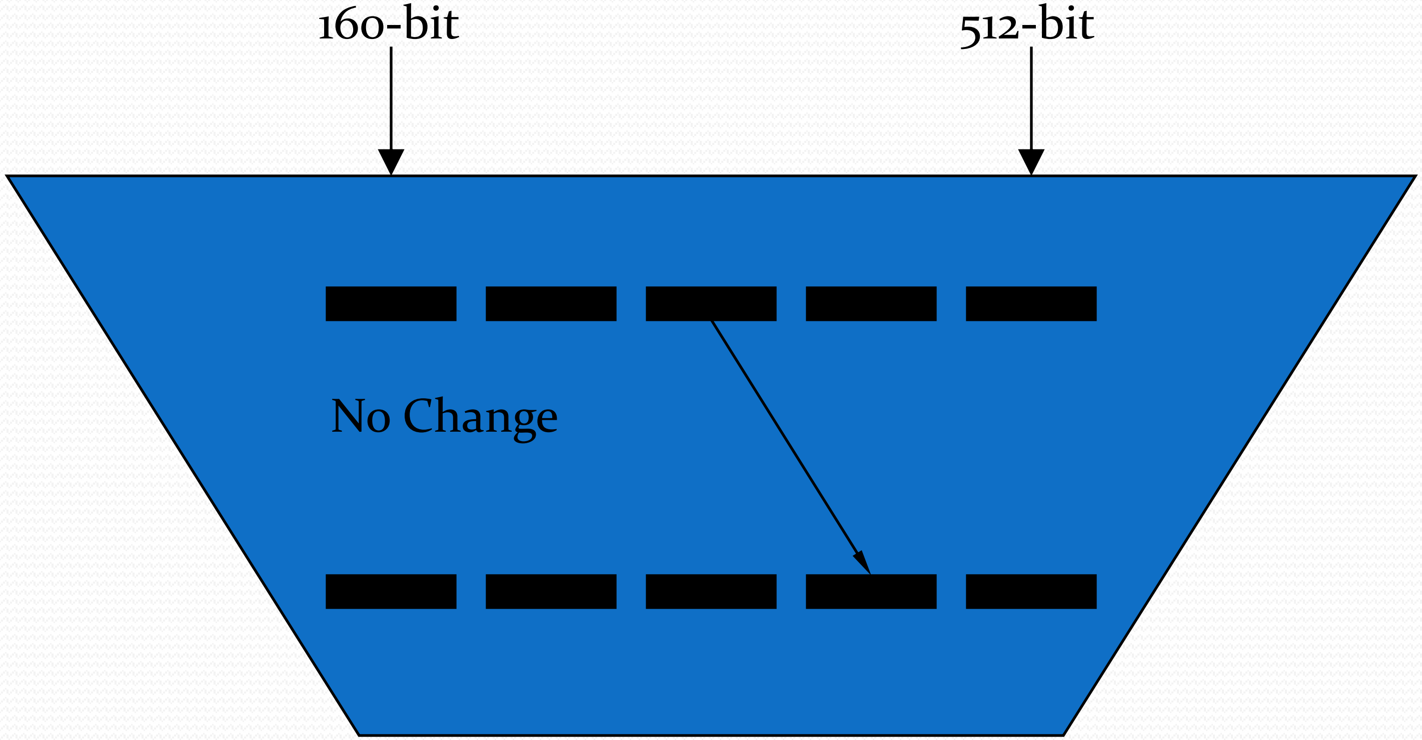
One of 80 rounds

A Cryptographic Hash: SHA-1



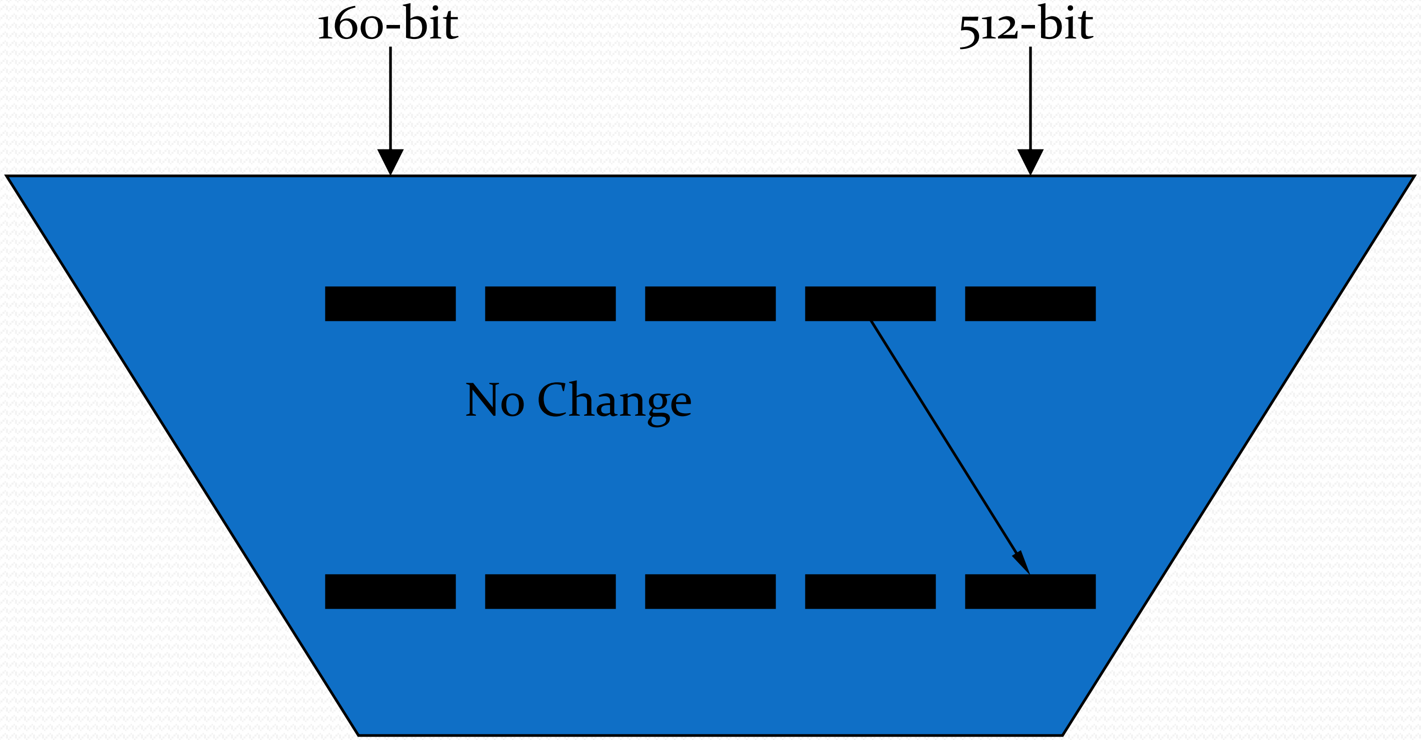
One of 80 rounds

A Cryptographic Hash: SHA-1



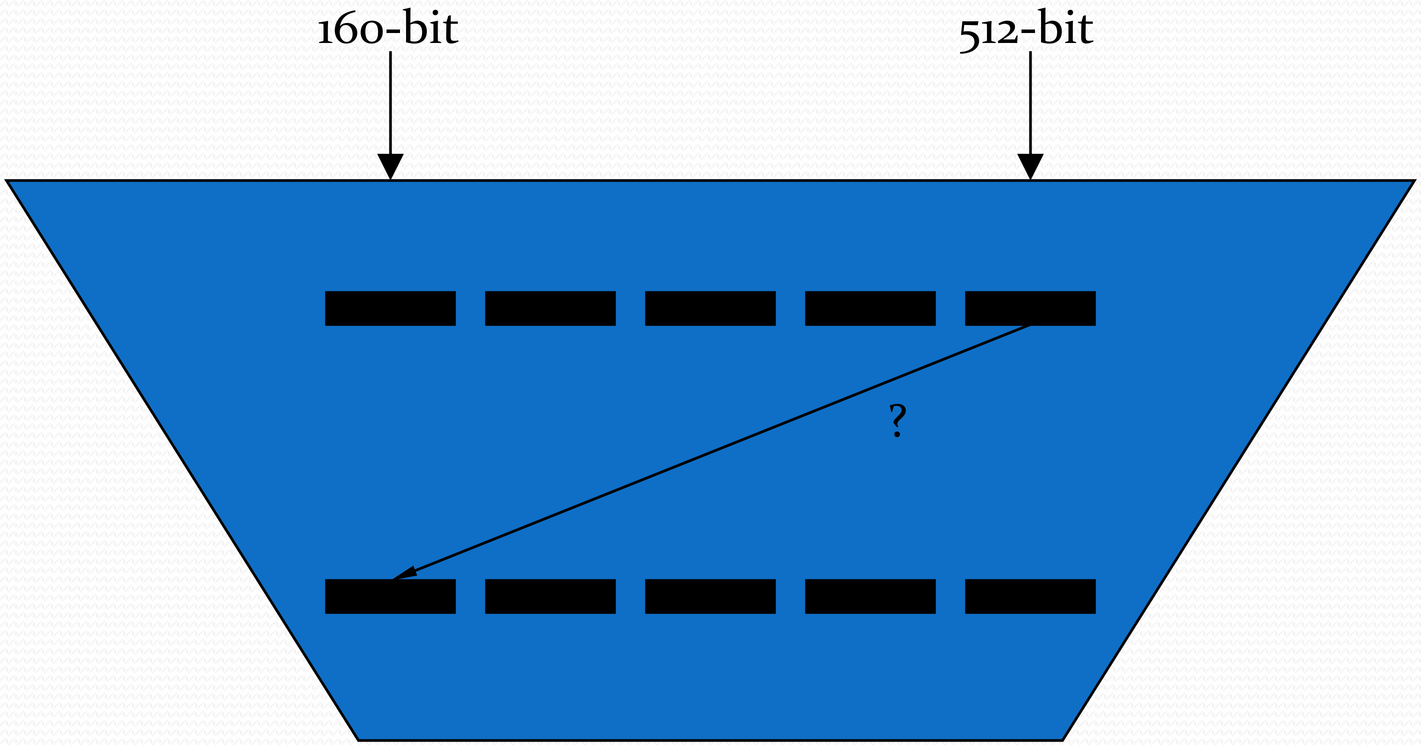
One of 80 rounds

A Cryptographic Hash: SHA-1



One of 80 rounds

A Cryptographic Hash: SHA-1



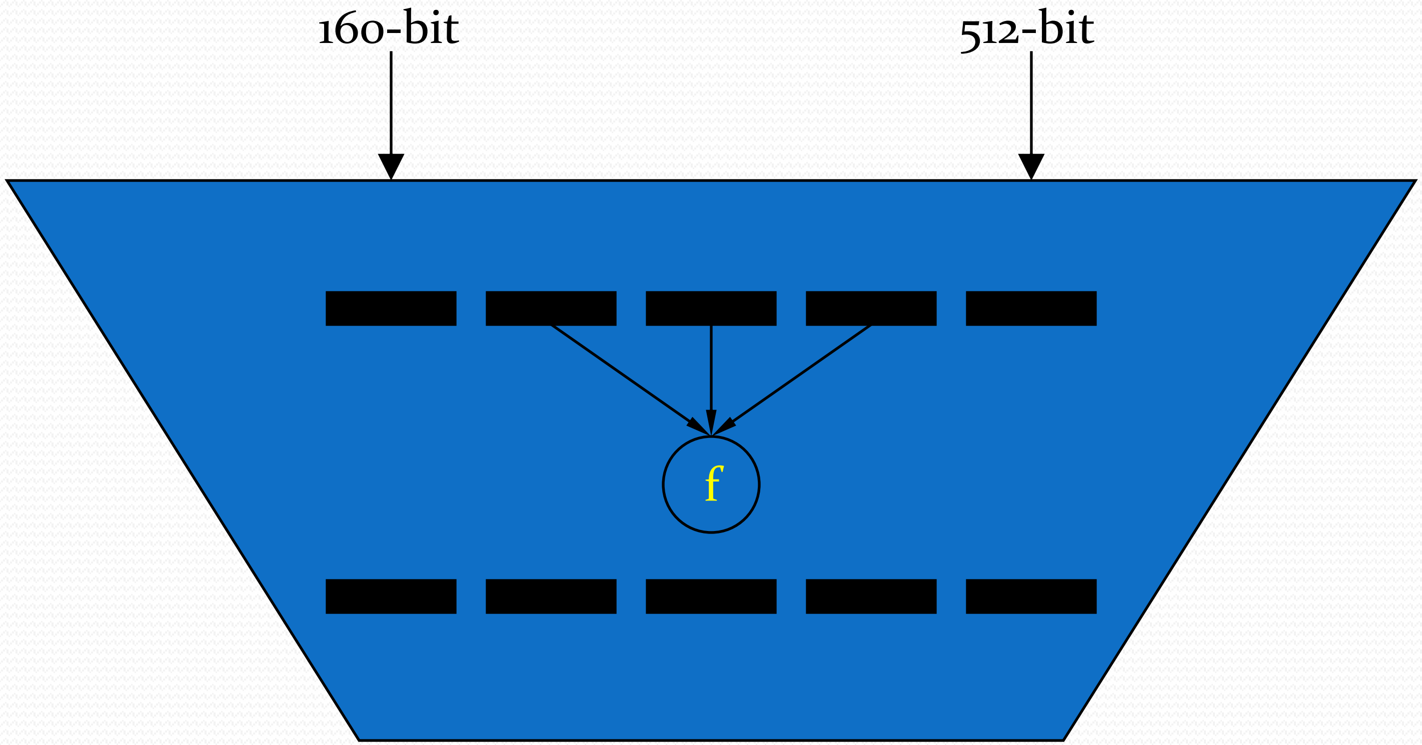
One of 80 rounds

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

- Take the rightmost word.
- Add in the leftmost word rotated 5 bits.
- Add in a round-dependent function f of the middle three words.

A Cryptographic Hash: SHA-1



One of 80 rounds

A Cryptographic Hash: SHA-1

Depending on the round, the "non-linear" function f is one of the following.

$$f(X,Y,Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$$

$$f(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$$

$$f(X,Y,Z) = X \oplus Y \oplus Z$$

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

- Take the rightmost word.
- Add in the leftmost word rotated 5 bits.
- Add in a round-dependent function f of the middle three words.

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

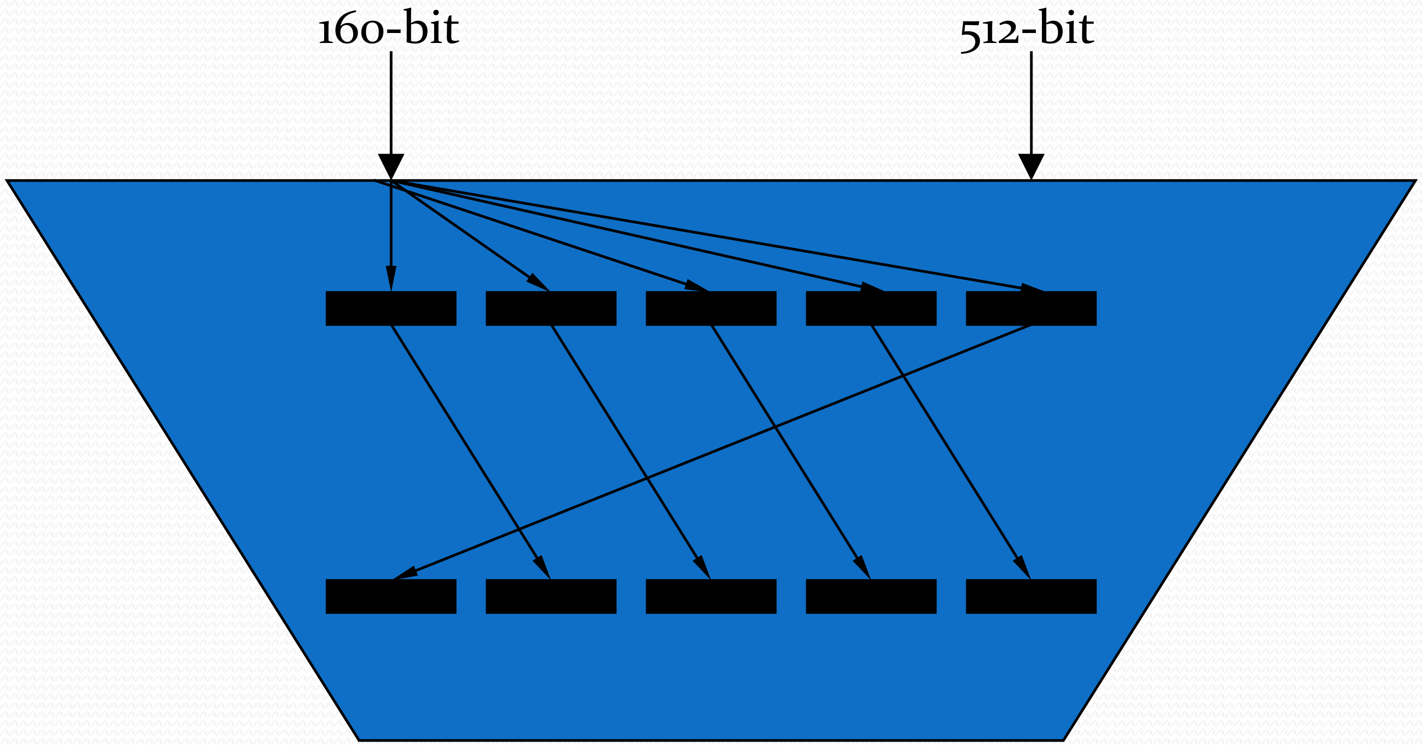
- Take the rightmost word.
- Add in the leftmost word rotated 5 bits.
- Add in a round-dependent function f of the middle three words.
- Add in a round-dependent constant.

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

- Take the rightmost word.
- Add in the leftmost word rotated 5 bits.
- Add in a round-dependent function f of the middle three words.
- Add in a round-dependent constant.
- Add in a portion of the 512-bit message.

A Cryptographic Hash: SHA-1



One of 80 rounds

Symmetric Ciphers

Private-key (symmetric) ciphers are usually divided into two classes.

- Stream ciphers
- Block ciphers

Symmetric Ciphers

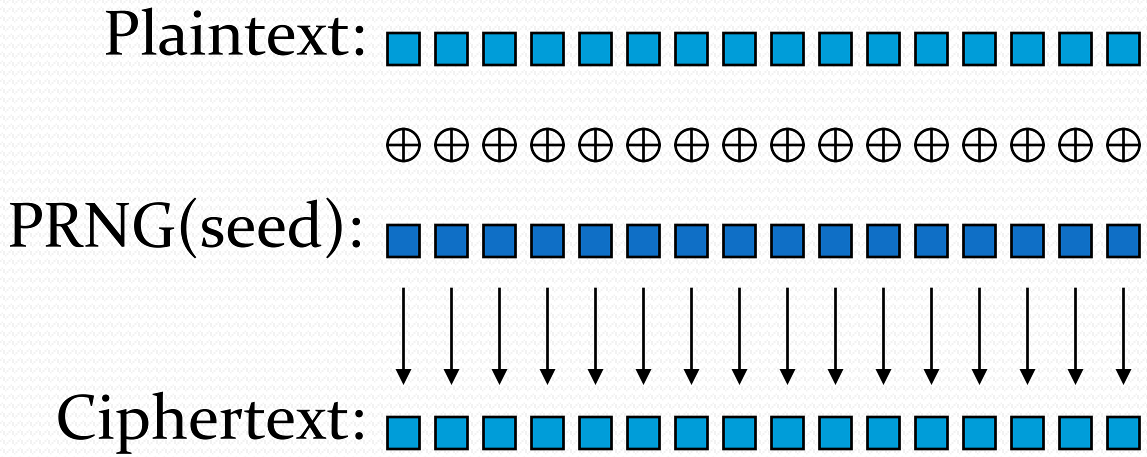
Private-key (symmetric) ciphers are usually divided into two classes.

- Stream ciphers
- Block ciphers

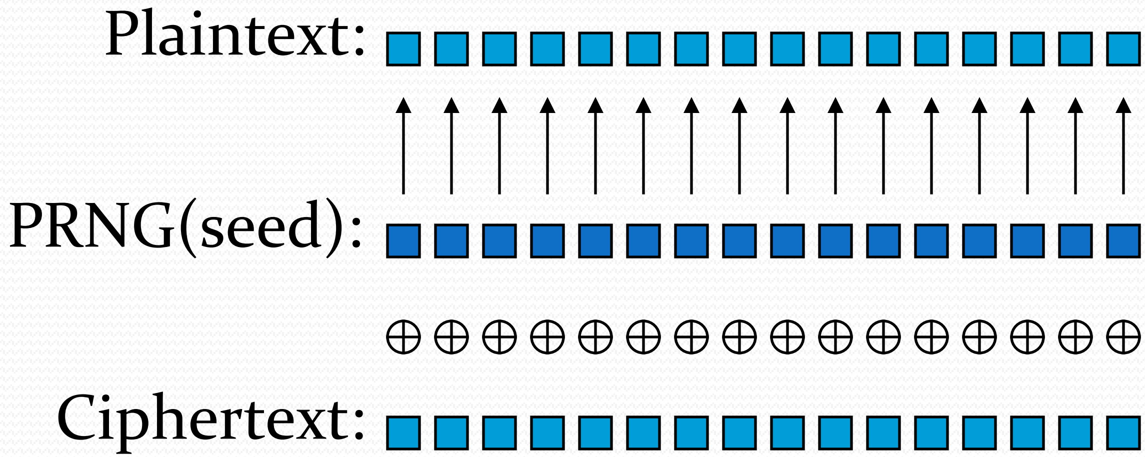
Stream Ciphers

- Use the key as a seed to a pseudo-random number-generator.
- Take the stream of output bits from the PRNG and XOR it with the plaintext to form the ciphertext.

Stream Cipher Encryption



Stream Cipher Decryption



A PRNG: Alleged RC4

<u>Initialization</u>

```
S[0..255] = 0,1,...,255
K[0..255] = Key,Key,Key,...
for i = 0 to 255
    j = (j + S[i] + K[i]) mod 256
    swap S[i] and S[j]
```

A PRNG: Alleged RC4

Iteration

i = (i + 1) mod 256

j = (j + S[i]) mod 256

swap S[i] and S[j]

t = (S[i] + S[j]) mod 256

Output S[t]

Stream Cipher Integrity

- It is easy for an adversary (even one who can't decrypt the ciphertext) to alter the plaintext in a known way.

Bob to Bob's Bank:

Please transfer $0,000,002.00 to the account of my good friend Alice.

Stream Cipher Integrity

- It is easy for an adversary (even one who can't decrypt the ciphertext) to alter the plaintext in a known way.

Bob to Bob's Bank:

Please transfer $1,000,002.00 to the account of my good friend Alice.

Stream Cipher Integrity

- It is easy for an adversary (even one who can't decrypt the ciphertext) to alter the plaintext in a known way.

Bob to Bob's Bank:

Please transfer $1,000,002.00 to the account of my good friend Alice.

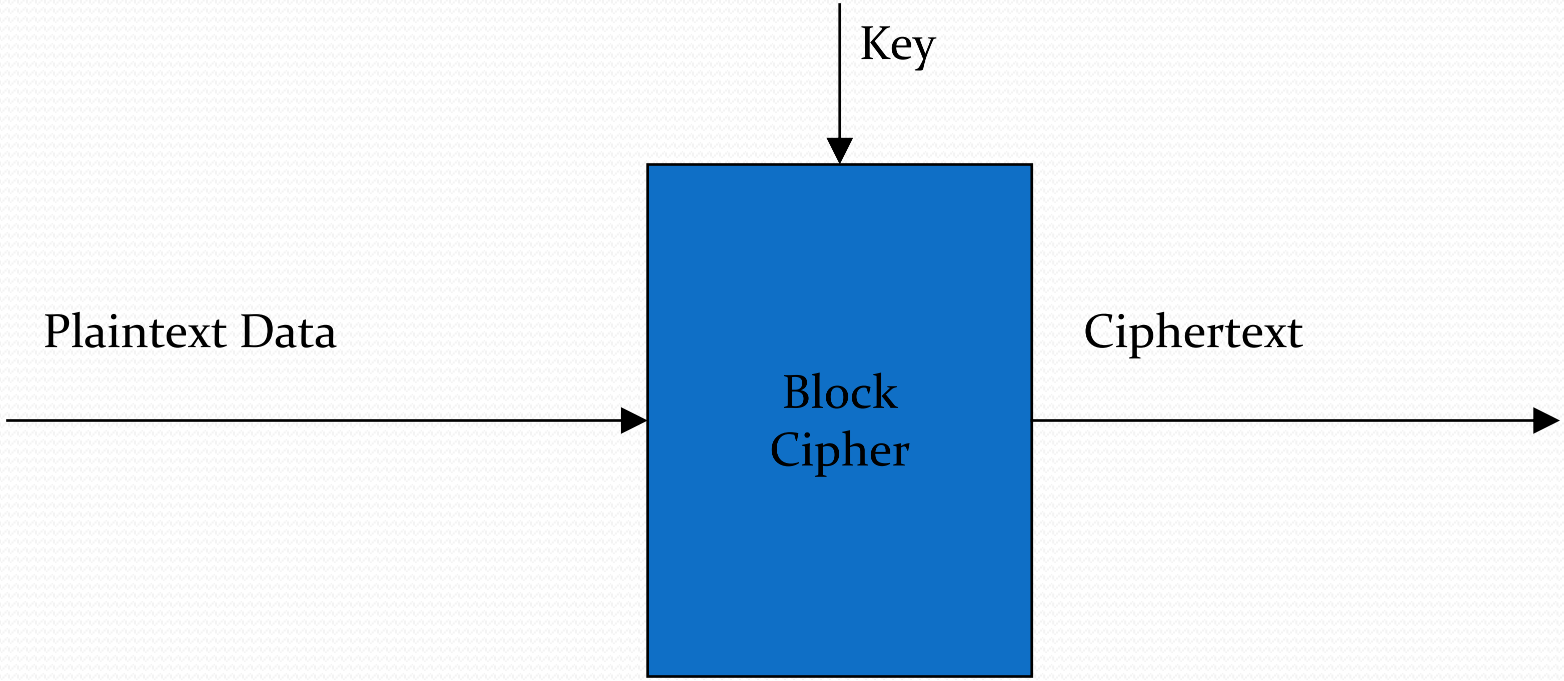
- This can be protected against by the careful addition of appropriate redundancy.

Symmetric Ciphers

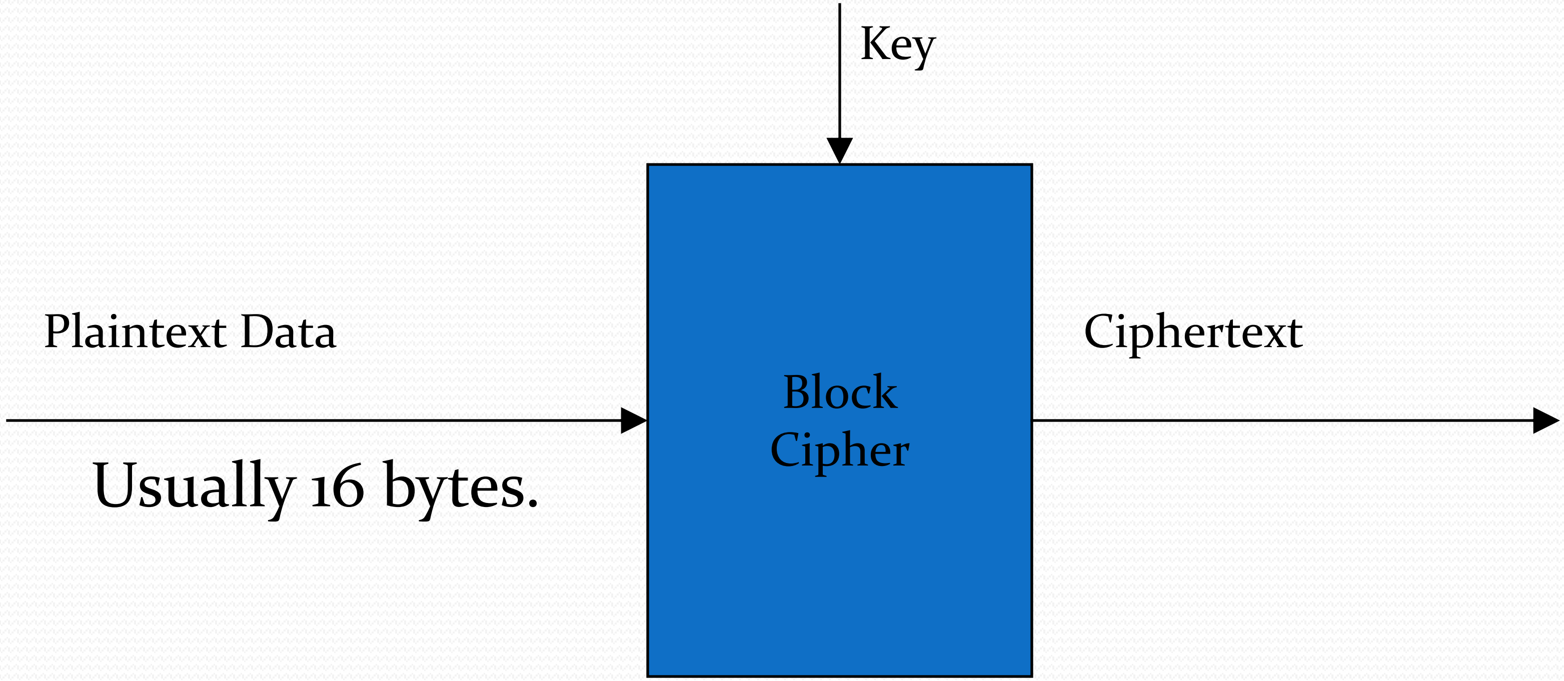
Private-key (symmetric) ciphers are usually divided into two classes.

- Stream ciphers
- Block ciphers

Block Ciphers

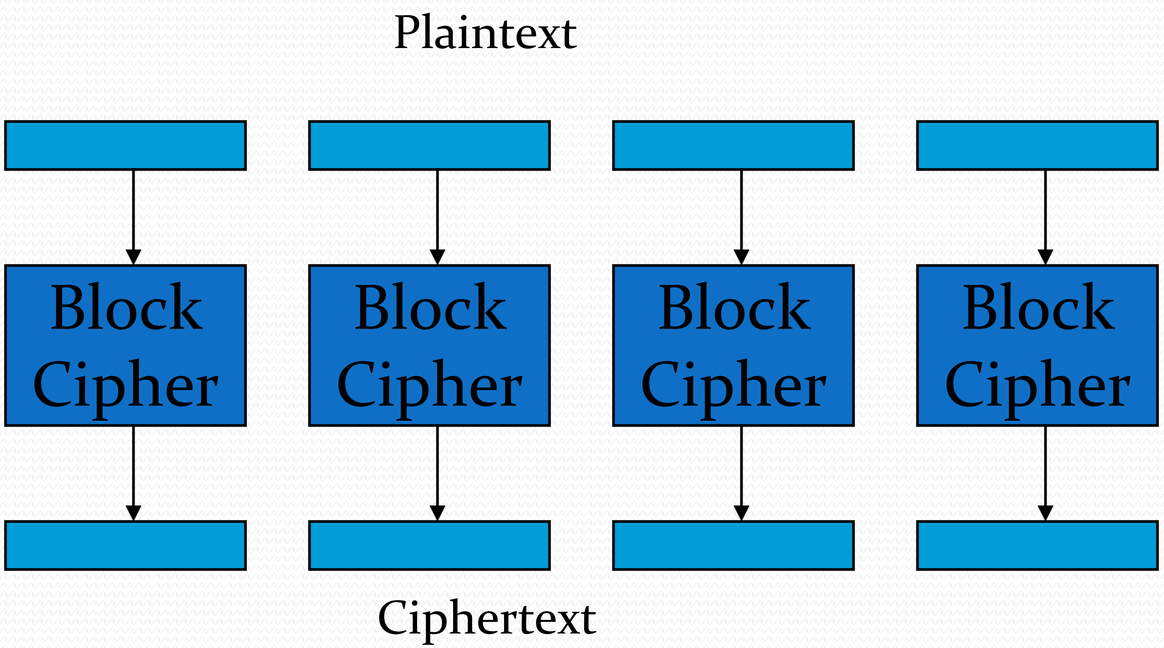


Block Ciphers



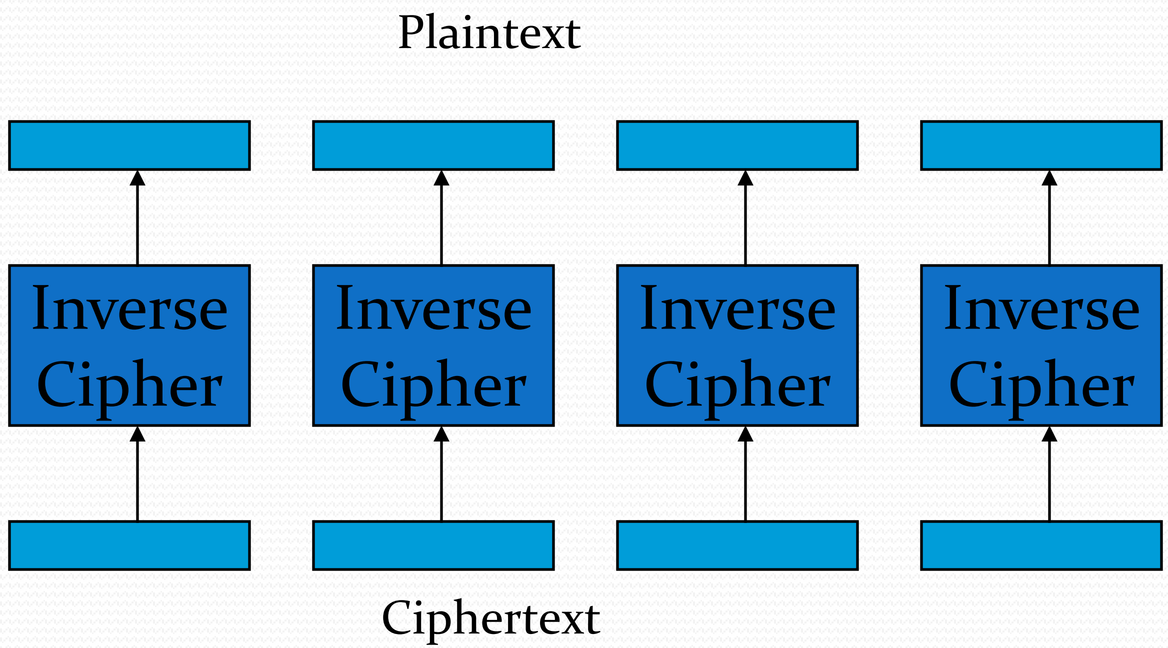
Block Cipher Modes

Electronic Code Book (ECB) Encryption:



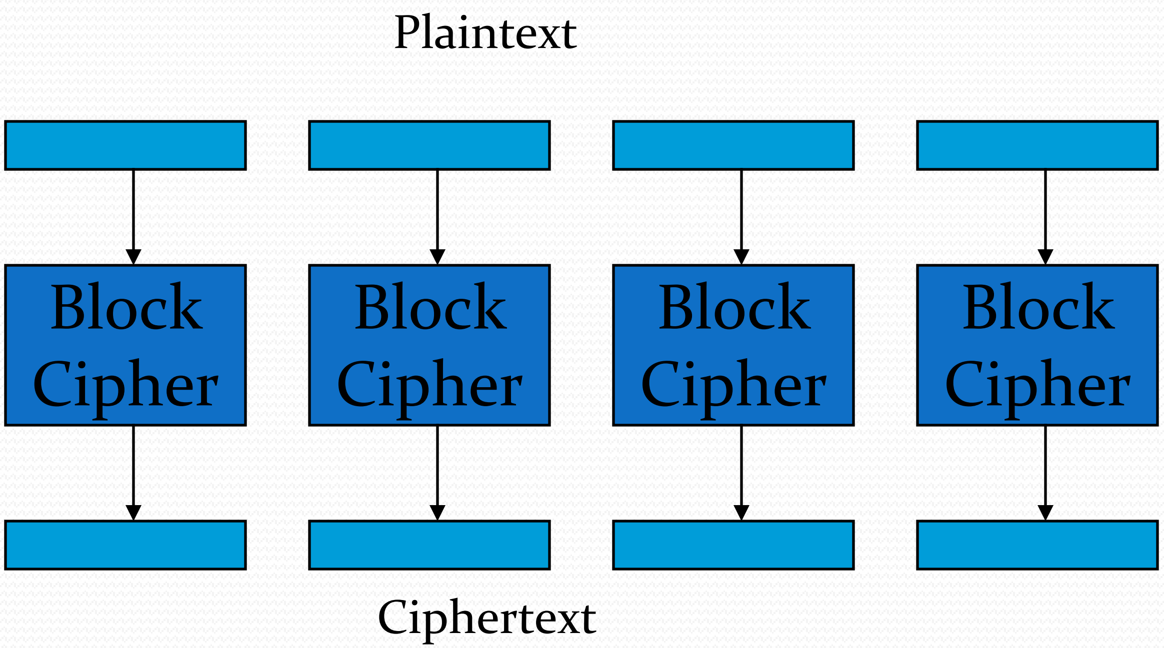
Block Cipher Modes

Electronic Code Book (ECB) Decryption:



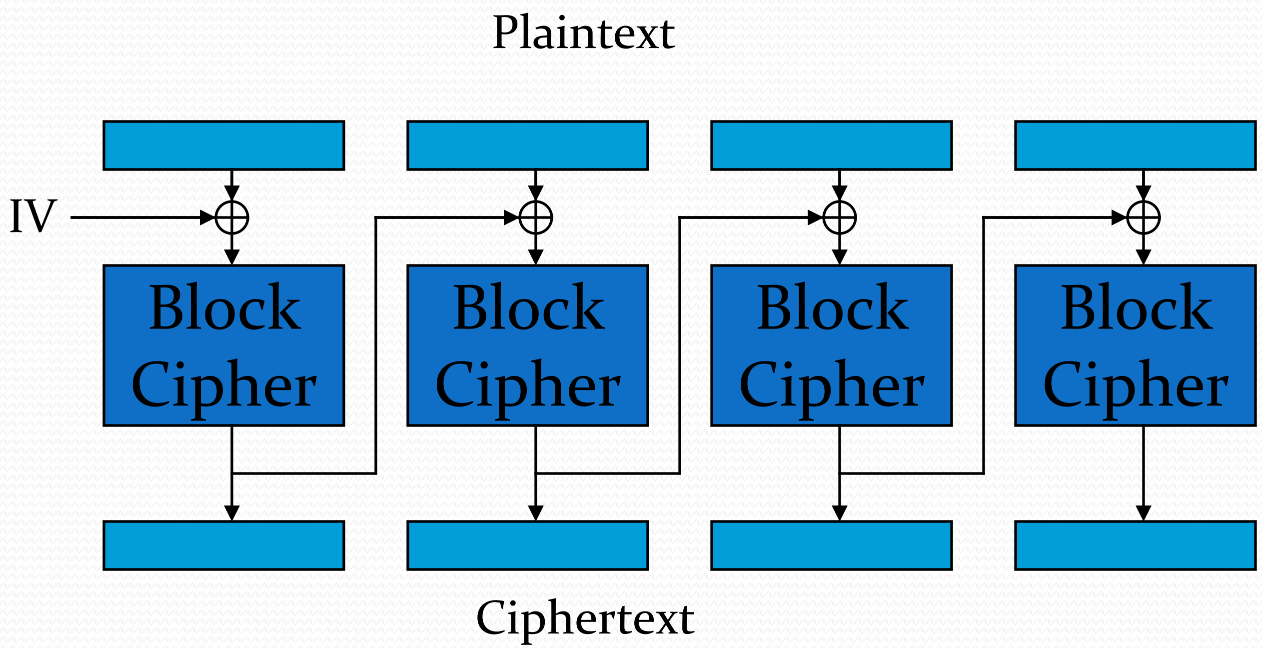
Block Cipher Modes

Electronic Code Book (ECB) Encryption:



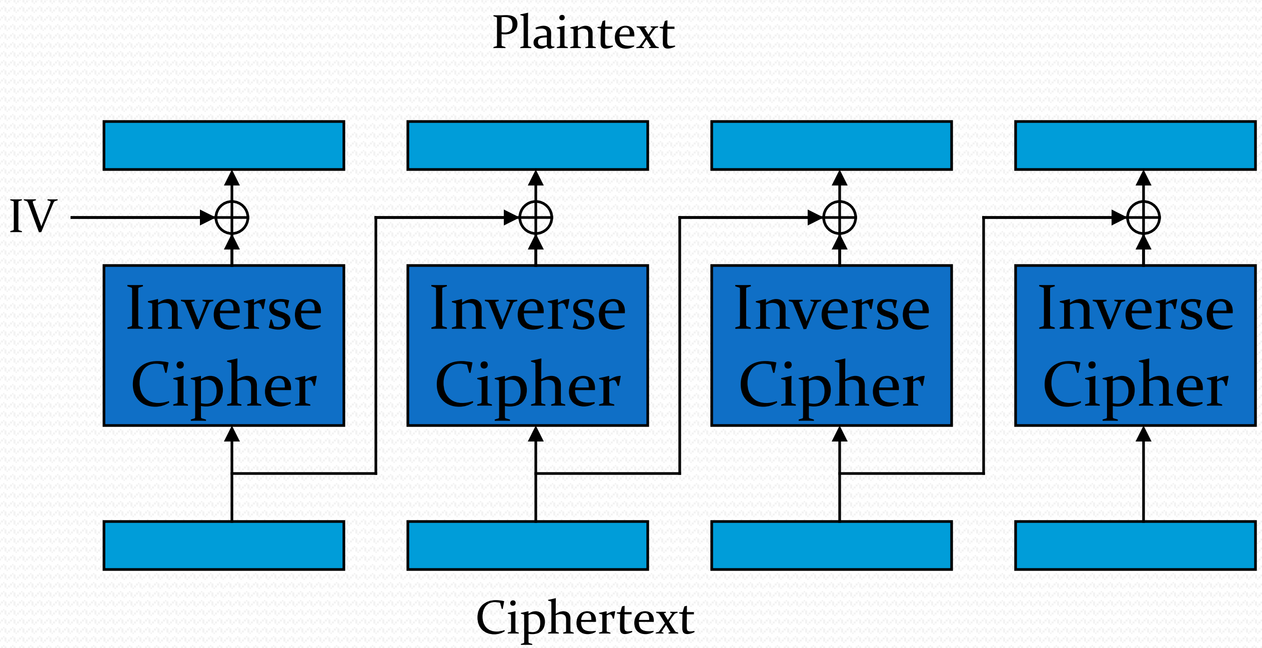
Block Cipher Modes

Cipher Block Chaining (CBC) Encryption:



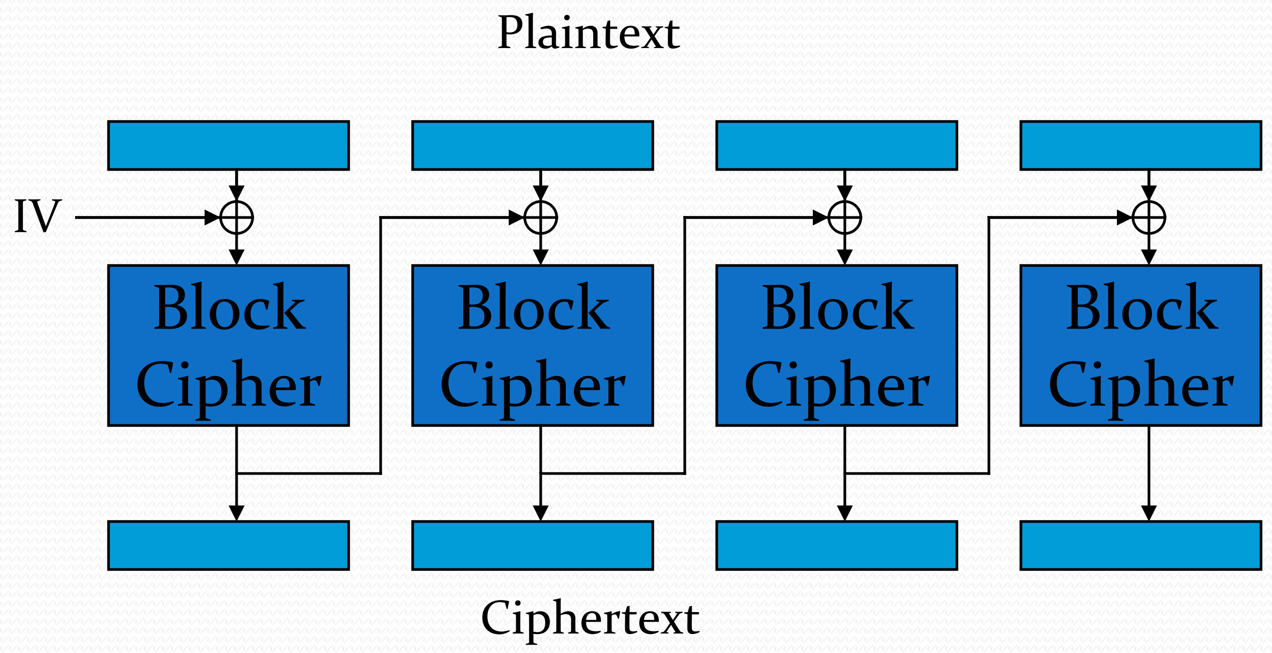
Block Cipher Modes

Cipher Block Chaining (CBC) Decryption:

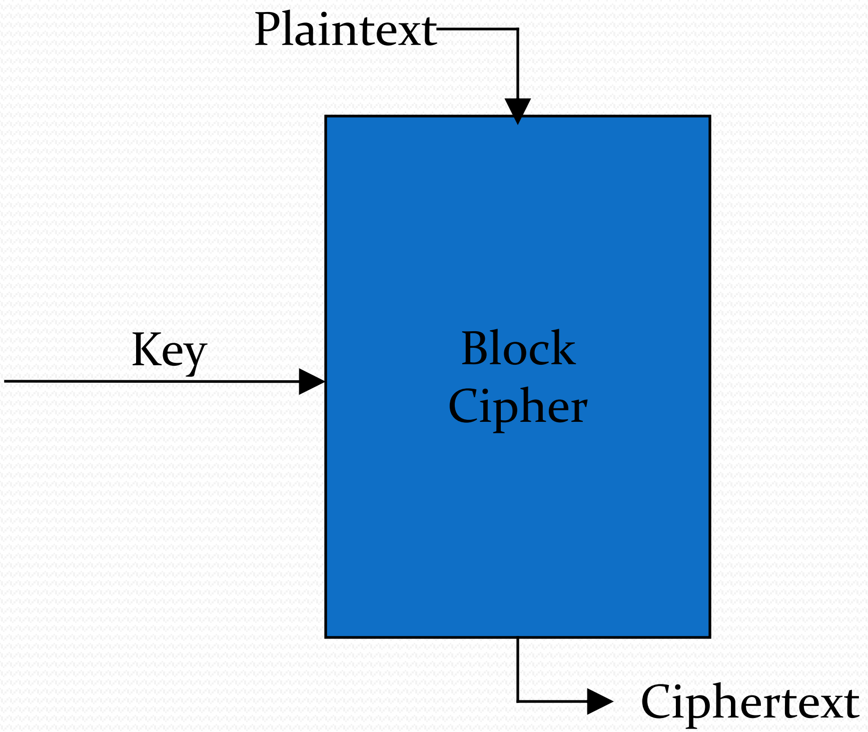


Block Cipher Modes

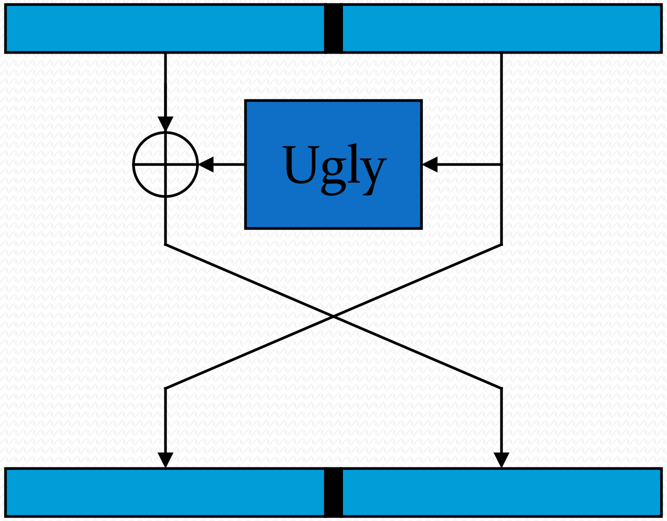
Cipher Block Chaining (CBC) Encryption:



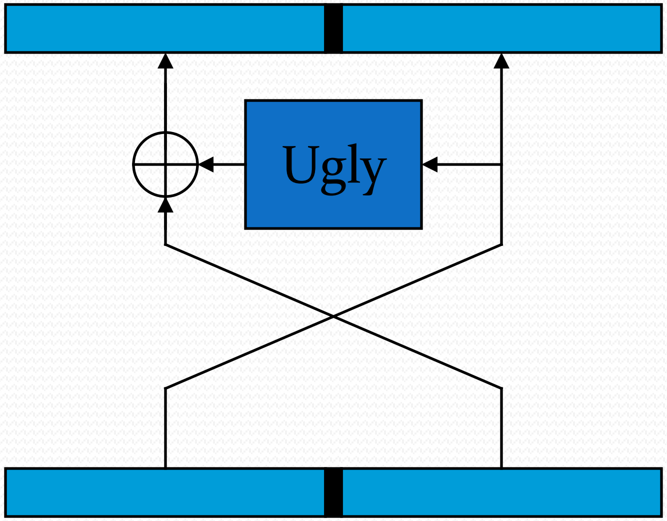
How to Build a Block Cipher



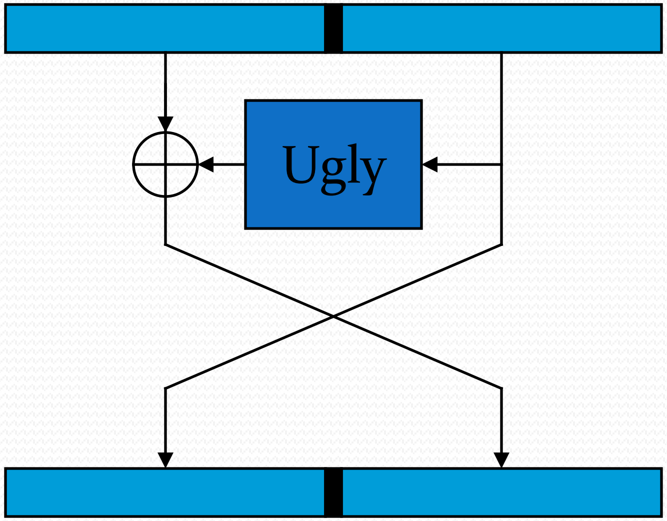
Feistel Ciphers



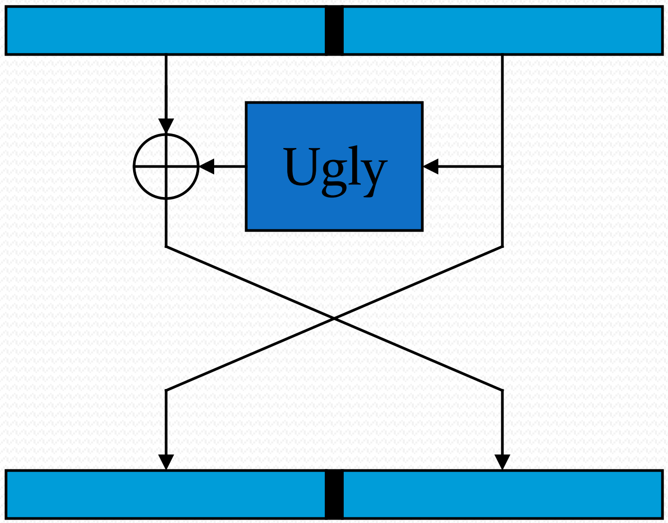
Feistel Ciphers



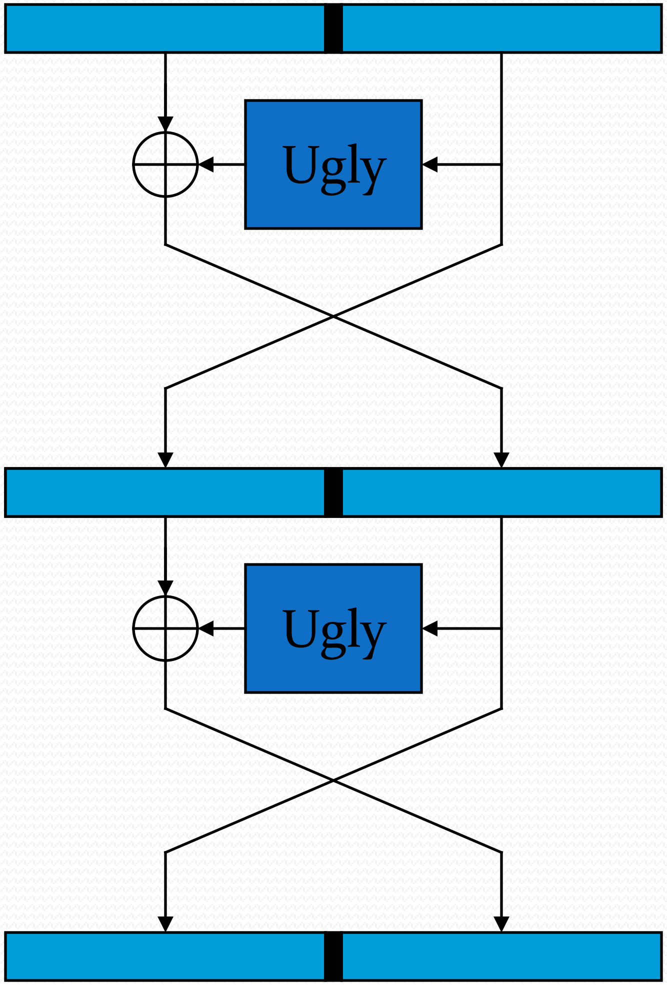
Feistel Ciphers



Feistel Ciphers



Feistel Ciphers

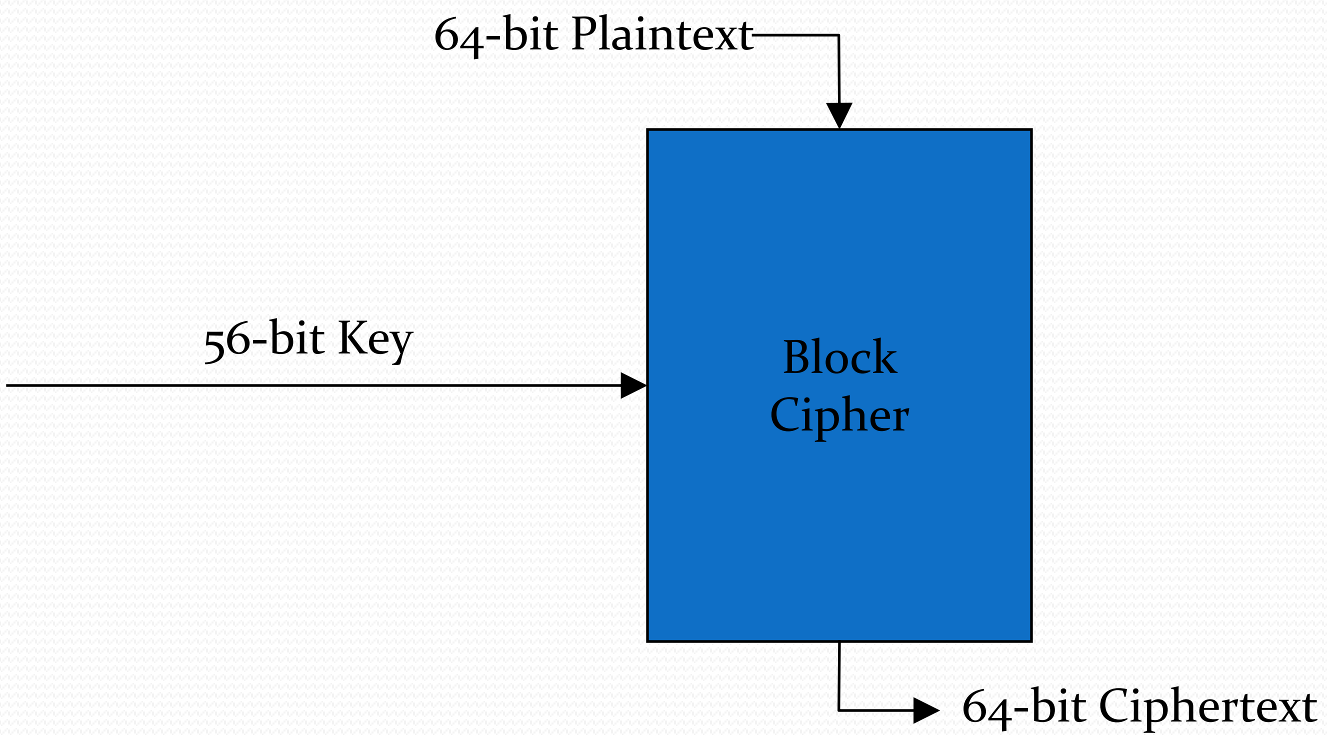


Feistel Ciphers

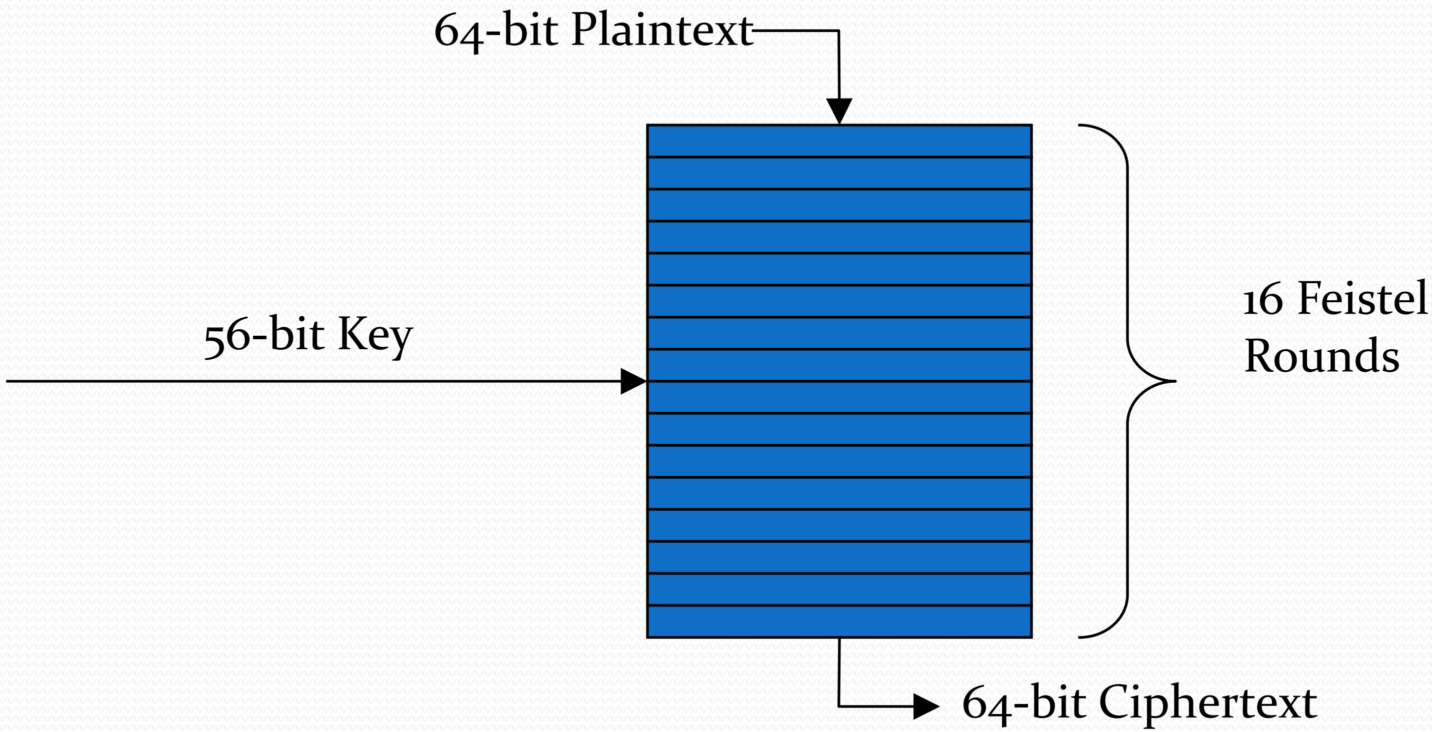
- Typically, most Feistel ciphers are iterated for about 16 rounds.
- Different "sub-keys" are used for each round.

- Even a weak round function can yield a strong Feistel cipher if iterated sufficiently.

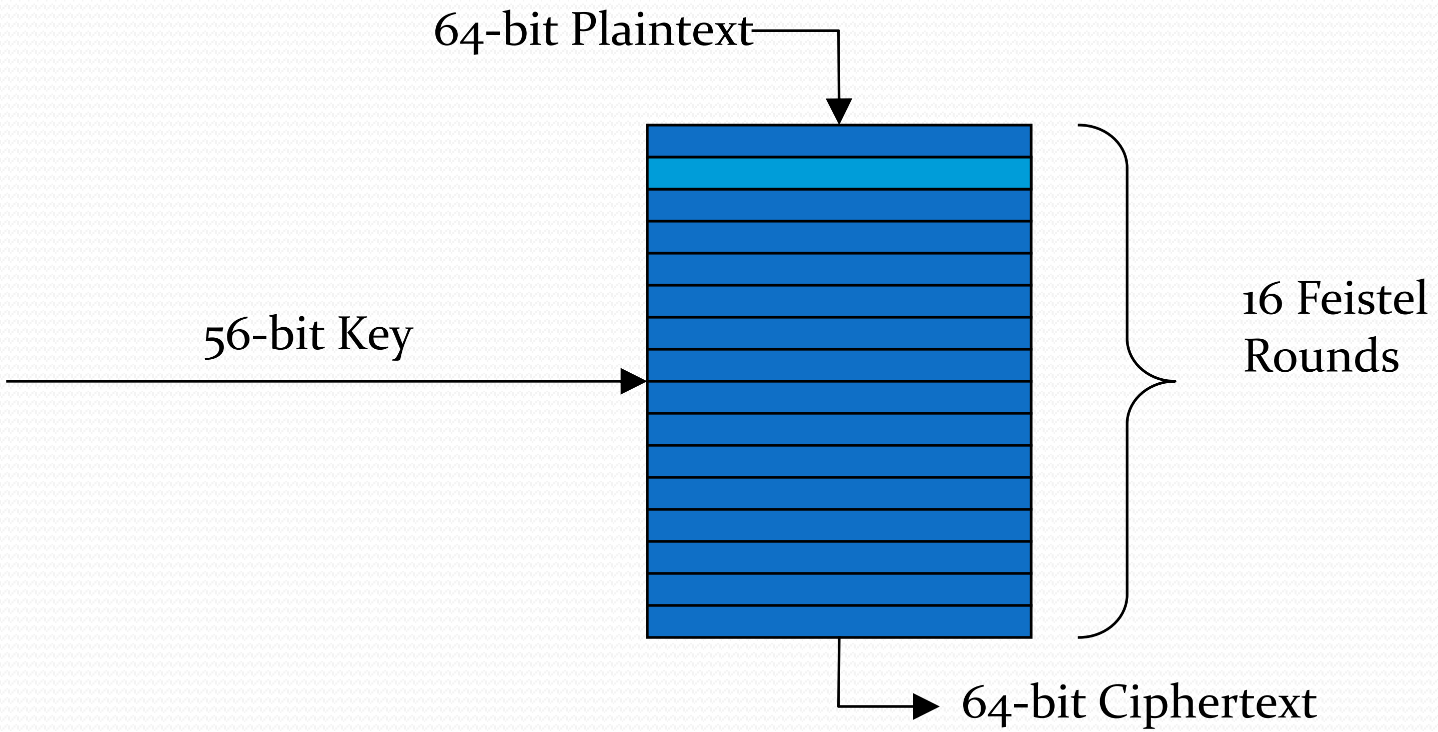
Data Encryption Standard (DES)



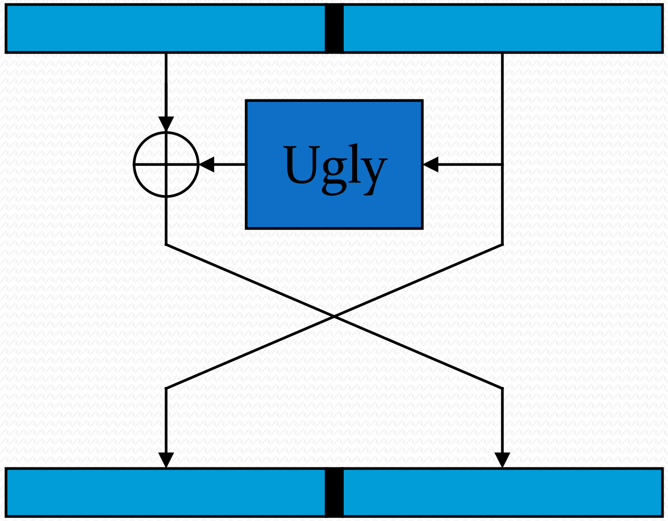
Data Encryption Standard (DES)



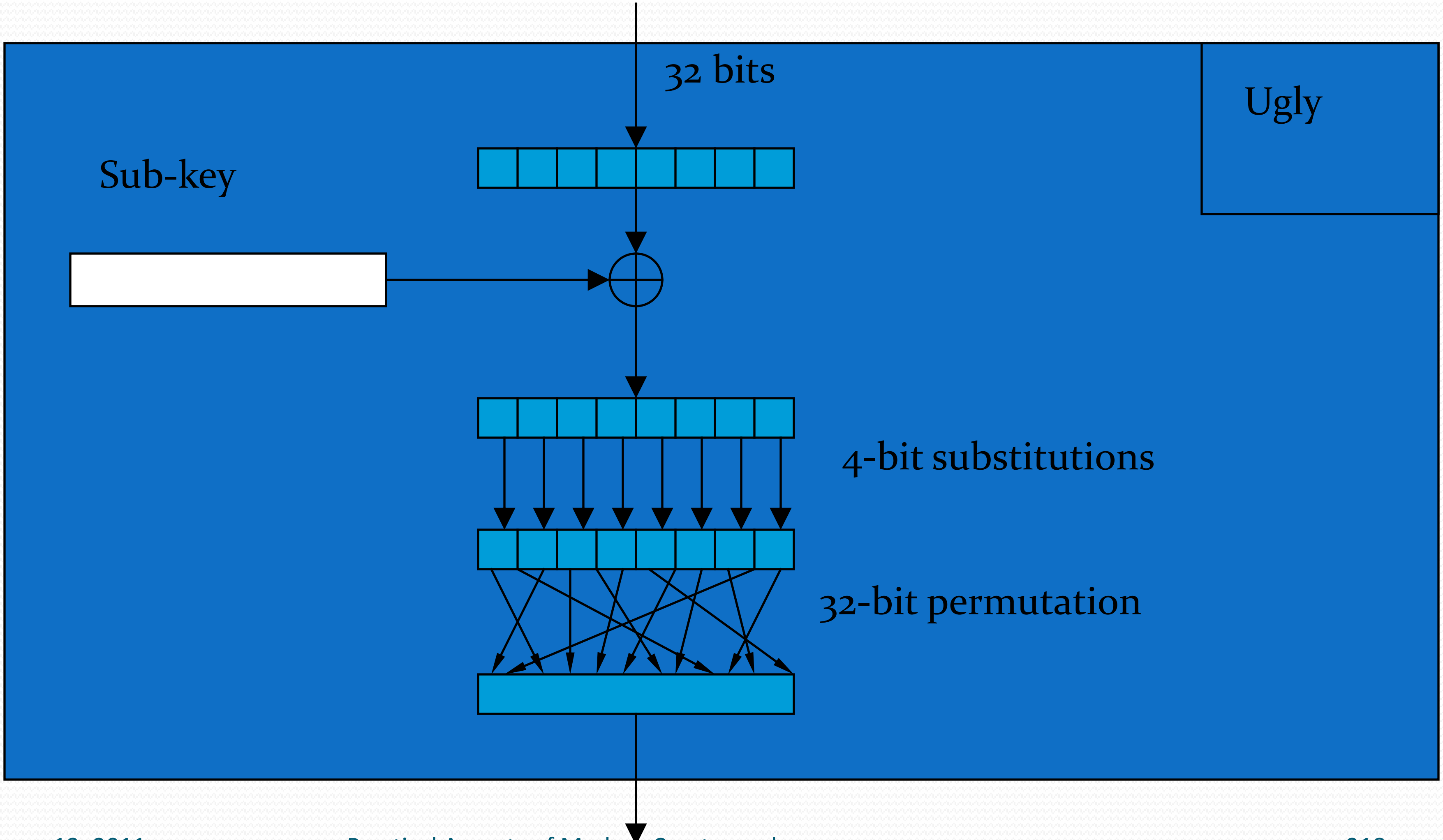
Data Encryption Standard (DES)



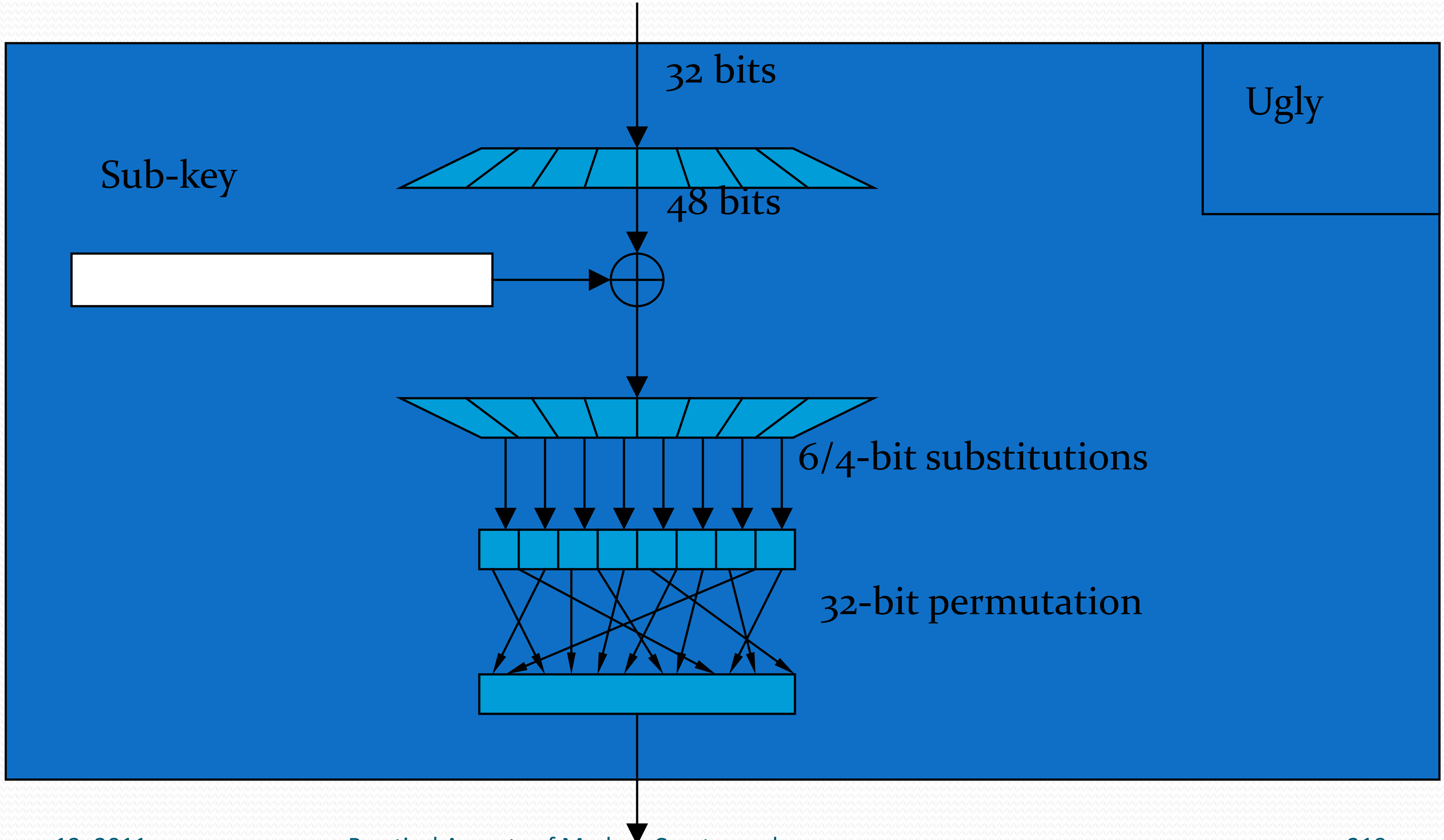
DES Round



Simplified DES Round Function



Actual DES Round Function



Cryptographic Tools

One-Way Trapdoor Functions

Public-Key Encryption Schemes

One-Way Functions

One-Way Hash Functions

Pseudo-Random Number-Generators

Secret-Key Encryption Schemes

Digital Signature Schemes