Practical Aspects of Modern Cryptography

Josh Benaloh
Brian LaMacchia
John Manferdelli

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

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

Diffie-Hellman

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

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

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

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.

Some RSA Details

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

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

whenever

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

Alice can easily select integers E and D such that $E \bullet D \bmod (P-1)(Q-1) = 1$.

Remaining RSA Basics

- Why is $Y^X \bmod PQ = Y$ whenever

  $X \bmod (P-1)(Q-1) = 1$, $0 \leq Y < PQ$,

  and P and Q are distinct primes?

- How can Alice can select integers E and D such that $E \bullet D \bmod (P-1)(Q-1) = 1$?

Fermat's Little Theorem

If p is prime,

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

Equivalently …

If p is prime,

then $x^{p} \bmod p = x \bmod p$ for all integers x.

Proof of Fermat's Little Theorem

The Binomial Theorem

$$(x+y)^p = x^p + \binom{p}{1}x^{p-1}y + \ldots + \binom{p}{p-1}xy^{p-1} + y^p$$

where $\binom{p}{i} = \frac{p!}{i!(p-i)!}$

Proof of Fermat's Little Theorem

The Binomial Theorem

$$(x+y)^p = x^p + \binom{p}{1}x^{p-1}y + \ldots + \binom{p}{p-1}xy^{p-1} + y^p$$

where $\binom{p}{i} = \frac{p!}{i!(p-i)!}$

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

Proof of Fermat's Little Theorem

The Binomial Theorem

$$(x+y)^p = x^p + \binom{p}{1}x^{p-1}y + \ldots + \binom{p}{p-1}xy^{p-1} + y^p$$

where $\binom{p}{i} = \frac{p!}{i!(p-i)!}$

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

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

Proof of Fermat's Little Theorem

Proof of Fermat's Little Theorem

By induction on x…

Proof of Fermat's Little Theorem

By induction on x…

Basis

Proof of Fermat's Little Theorem

By induction on x…

Basis

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

Proof of Fermat's Little Theorem

By induction on x…

Basis

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

Proof of Fermat's Little Theorem

Inductive Step

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

Proof of Fermat's Little Theorem

Inductive Step

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

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

$= (x + 1) \bmod p$.

Proof of Fermat's Little Theorem

Inductive Step

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

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

$= (x + 1) \bmod p$.

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

Proof of Fermat's Little Theorem

Inductive Step

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

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

$= (x + 1) \bmod p$.

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 …

Proof of RSA

We have shown …

$$Y^P \bmod P = Y \text{ whenever } 0 \leq Y < P$$

Proof of RSA

We have shown …

$$Y^P \bmod P = Y \text{ whenever } 0 \leq Y < P$$

and P is *prime*!

Proof of RSA

We have shown …

$$Y^P \bmod P = Y \text{ whenever } 0 \leq Y < P$$

and P is *prime*!

You will show …

Proof of RSA

We have shown …

$$Y^P \bmod P = Y \text{ whenever } 0 \leq Y < P$$

and P is *prime*!

You will show …

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

Proof of RSA

We have shown …

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

and P is *prime*!

You will show …

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

P and Q are distinct primes 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.
- 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.
- The prime factors of N+1 are not among the finite set of primes multiplied to form N.
- 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.

Testing Primality

Recall Fermat's Little Theorem

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

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}, …, $a^{(N-1)/2}$ 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}, …, $a^{(N-1)/2}$ 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}, …, $a^{(N-1)/2}$ 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

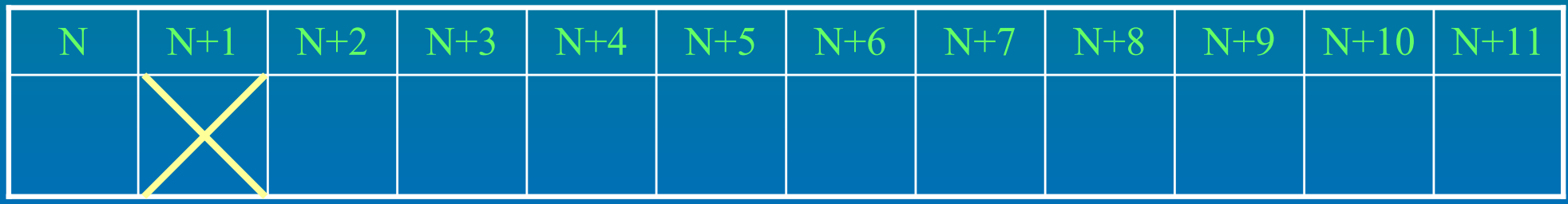
Pick a random starting point N.

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

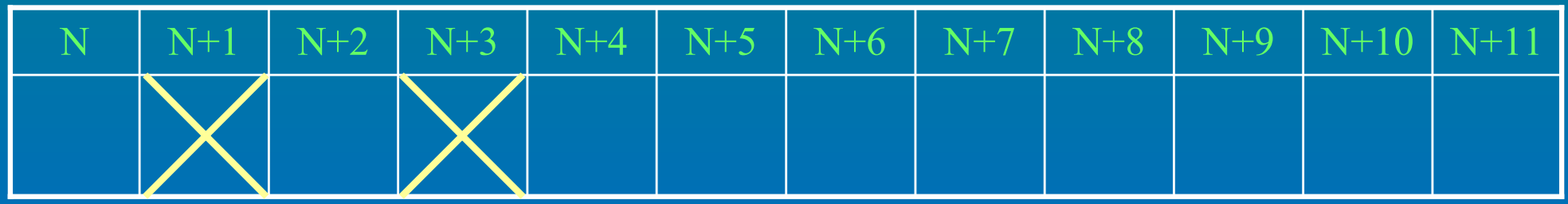
Pick a random starting point N.



Sieving out multiples of 2

Sieving for Primes

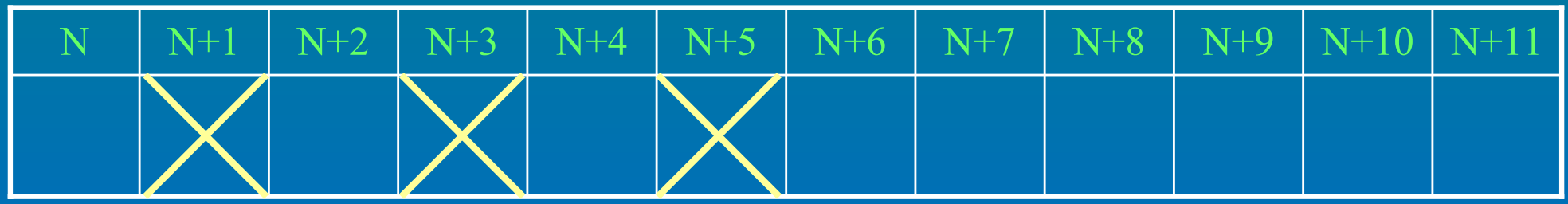
Pick a random starting point N.



Sieving out multiples of 2

Sieving for Primes

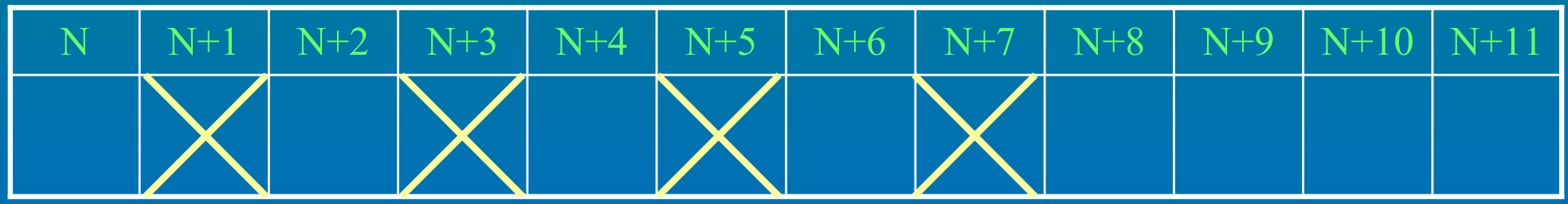
Pick a random starting point N.



Sieving out multiples of 2

Sieving for Primes

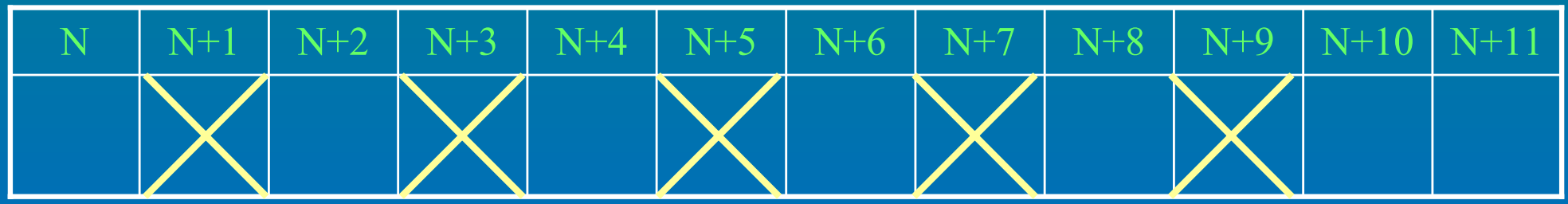
Pick a random starting point N.



Sieving out multiples of 2

Sieving for Primes

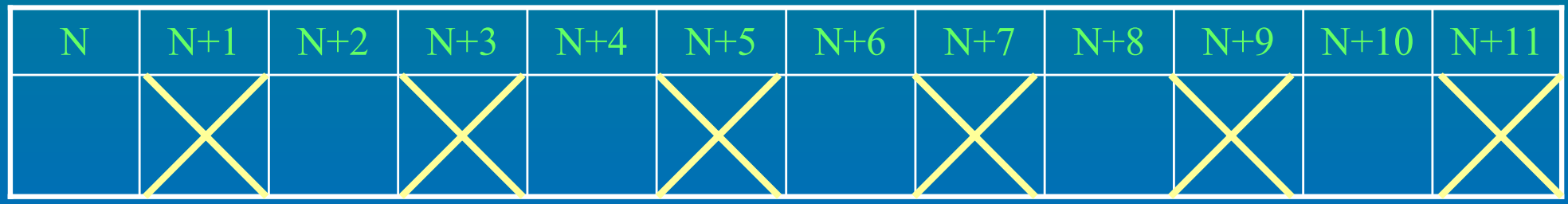
Pick a random starting point N.



Sieving out multiples of 2

Sieving for Primes

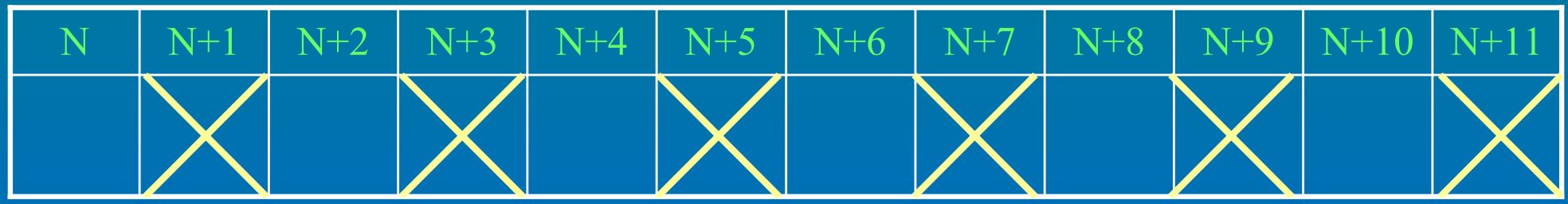
Pick a random starting point N.



Sieving out multiples of 2

Sieving for Primes

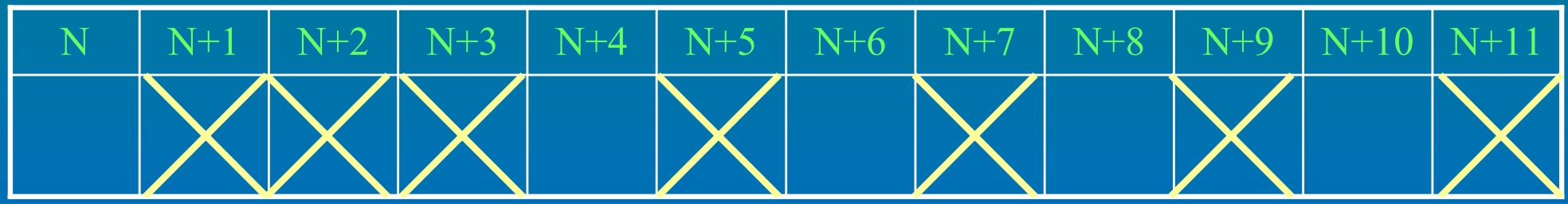
Pick a random starting point N.



Sieving out multiples of 3

Sieving for Primes

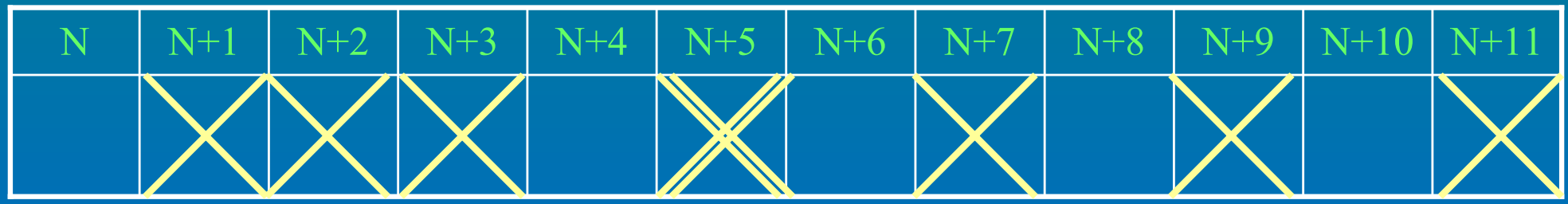
Pick a random starting point N.



Sieving out multiples of 3

Sieving for Primes

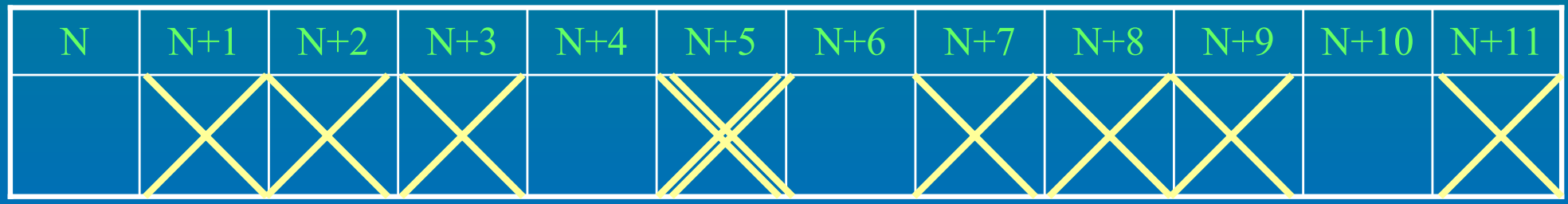
Pick a random starting point N.



Sieving out multiples of 3

Sieving for Primes

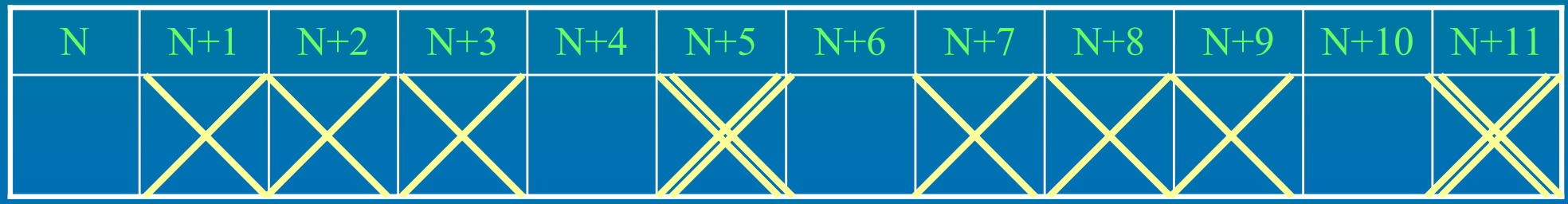
Pick a random starting point N.



Sieving out multiples of 3

Sieving for Primes

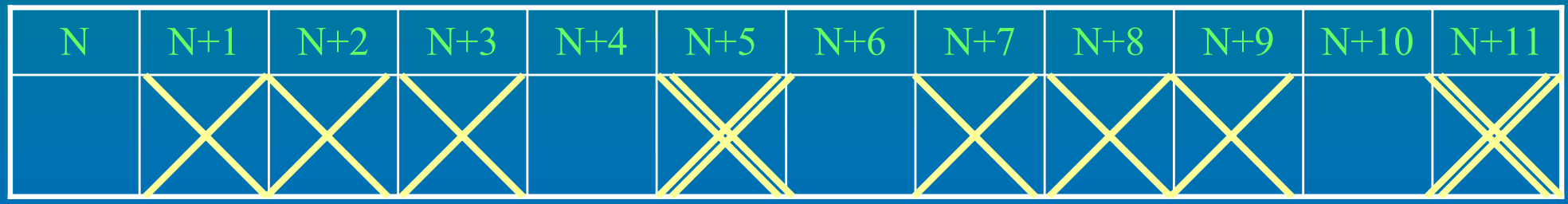
Pick a random starting point N.



Sieving out multiples of 3

Sieving for Primes

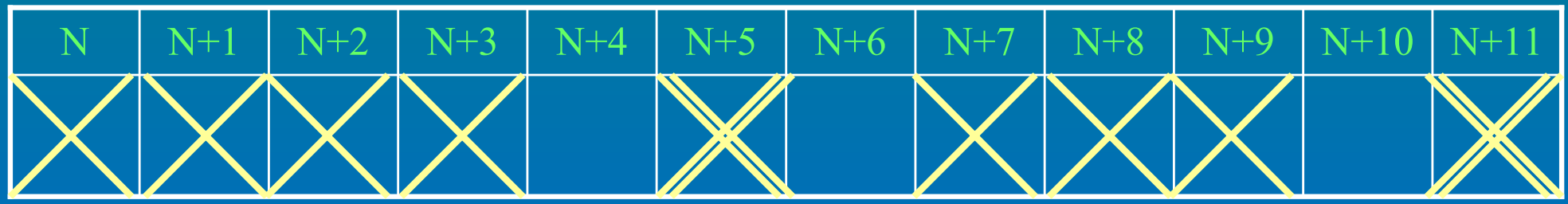
Pick a random starting point N.



Sieving out multiples of 5

Sieving for Primes

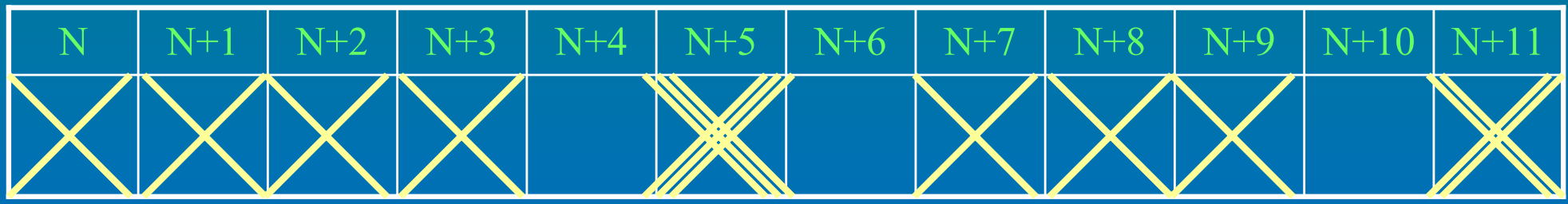
Pick a random starting point N.



Sieving out multiples of 5

Sieving for Primes

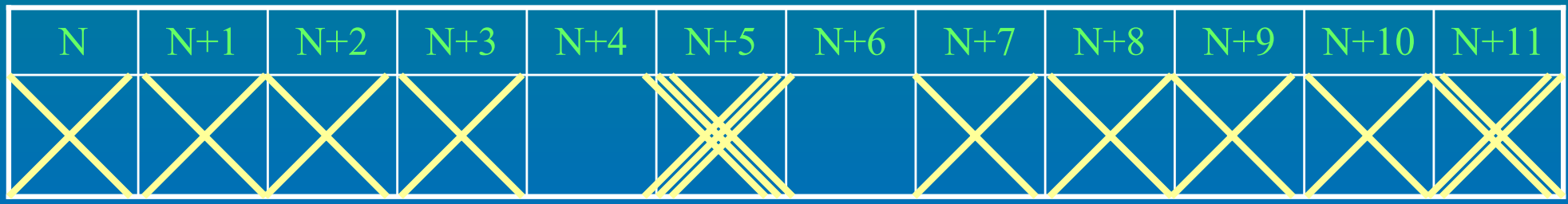
Pick a random starting point N.



Sieving out multiples of 5

Sieving for Primes

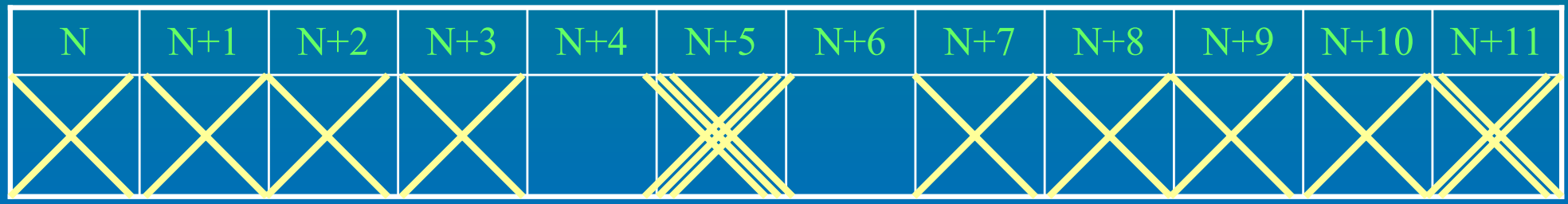
Pick a random starting point N.



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.

Remaining RSA Basics

Remaining RSA Basics

- Why is $Y^X \bmod PQ = Y$ whenever

  $X \bmod (P-1)(Q-1) = 1$, $0 \leq Y < PQ$,

  and P and Q are distinct primes?

Remaining RSA Basics

- Why is $Y^X \bmod PQ = Y$ whenever

  $X \bmod (P-1)(Q-1) = 1$, $0 \leq Y < PQ$,

  and P and Q are distinct primes?

- How can Alice can select integers E and D such that $E \bullet D \bmod (P-1)(Q-1) = 1$?

Modular Arithmetic

Modular Arithmetic

- To compute (A+B) mod N,
  compute (A+B) and take the result mod N.

Modular Arithmetic

- To compute (A+B) mod N,

  compute (A+B) and take the result mod N.
- To compute (A-B) mod N,

  compute (A-B) and take the result mod N.

Modular Arithmetic

- To compute $(A+B) \bmod N$,

  compute $(A+B)$ and take the result $\bmod N$.
- To compute $(A-B) \bmod N$,

  compute $(A-B)$ and take the result $\bmod N$.
- To compute $(A \times B) \bmod N$,

  compute $(A \times B)$ and take the result $\bmod N$.

Modular Arithmetic

- To compute $(A+B) \bmod N$,

  compute $(A+B)$ and take the result $\bmod N$.
- To compute $(A-B) \bmod N$,

  compute $(A-B)$ and take the result $\bmod N$.
- To compute $(A \times B) \bmod N$,

  compute $(A \times B)$ and take the result $\bmod N$.
- To compute $(A \div B) \bmod N$, …

Modular Division

Modular Division

What is the value of $(1 \div 2) \bmod 7$?

We need a solution to $2x \bmod 7 = 1$.

Modular Division

What is the value of $(1 \div 2) \bmod 7$?

We need a solution to $2x \bmod 7 = 1$.

Try $x = 4$.

Modular Division

What is the value of $(1 \div 2) \bmod 7$?

We need a solution to $2x \bmod 7 = 1$.

Try $x = 4$.

What is the value of $(7 \div 5) \bmod 11$?

We need a solution to $5x \bmod 11 = 7$.

Modular Division

What is the value of $(1 \div 2) \bmod 7$?

We need a solution to $2x \bmod 7 = 1$.

Try $x = 4$.

What is the value of $(7 \div 5) \bmod 11$?

We need a solution to $5x \bmod 11 = 7$.

Try $x = 8$.

Modular Division

Modular Division

Is modular division always well-defined?

Modular Division

Is modular division always well-defined?

$(1 \div 3) \bmod 6 = ?$

Modular Division

Is modular division always well-defined?

$(1 \div 3) \bmod 6 = ?$

$3x \bmod 6 = 1$ has no solution!

Modular Division

Is modular division always well-defined?

$(1 \div 3) \bmod 6 = ?$

$3x \bmod 6 = 1$ has no solution!

Fact

$(A \div B) \bmod N$ always has a solution when $\gcd(B,N) = 1$.

Modular Division

Fact

$(A \div B) \bmod N$ always has a solution when $\gcd(B,N) = 1$.*

Modular Division

Fact

$(A \div B) \bmod N$ always has a solution when $\gcd(B,N) = 1$.*

*There is no solution if $\gcd(A,B) = 1$ and $\gcd(B,N) \neq 1$.

Greatest Common Divisors

Greatest Common Divisors

$$\gcd(A , B) = \gcd(B , A - B)$$

Greatest Common Divisors

$\gcd(A , B) = \gcd(B , A - B)$

since any common factor of A and B is also a factor of $A - B$.

Greatest Common Divisors

gcd(A , B) = gcd(B , A – B)

since any common factor of A and B is also a factor of A – B.

gcd(21,12) = gcd(12,9) = gcd(9,3)

= gcd(6,3) = gcd(3,6) = gcd(3,3)

= gcd(3,0) = 3

Greatest Common Divisors

$$\gcd(A , B) = \gcd(B , A - B)$$

Greatest Common Divisors

$\gcd(A , B) = \gcd(B , A - B)$

$\gcd(A , B) = \gcd(B , A - kB)$ for any integer k.

Greatest Common Divisors

$\gcd(A , B) = \gcd(B , A - B)$

$\gcd(A , B) = \gcd(B , A - kB)$ for any integer k.

$\gcd(A , B) = \gcd(B , A \bmod B)$

Greatest Common Divisors

gcd(A , B) = gcd(B , A – B)

gcd(A , B) = gcd(B , A – *k*B) for any integer *k*.

gcd(A , B) = gcd(B , A mod B)

gcd(21,12) = gcd(12,9) = gcd(9,3)

= gcd(3,0) = 3

Extended Euclidean Algorithm

Given integers A and B, find integers X and Y such that $AX + BY = \gcd(A,B)$.

Extended Euclidean Algorithm

Given integers A and B, find integers X and Y such that $AX + BY = \gcd(A,B)$.

When $\gcd(A,B) = 1$, solve $AX \bmod B = 1$, by finding X and Y such that

$$AX + BY = \gcd(A,B) = 1.$$

Extended Euclidean Algorithm

Given integers A and B, find integers X and Y such that $AX + BY = \gcd(A,B)$.

When $\gcd(A,B) = 1$, solve $AX \bmod B = 1$, by finding X and Y such that

$$AX + BY = \gcd(A,B) = 1.$$

Compute $(C \div A) \bmod B$ as $C \times (1 \div A) \bmod B$.

Extended Euclidean Algorithm

gcd(35, 8) =

gcd(8, 35 mod 8) = gcd(8, 3) =

gcd(3, 8 mod 3) = gcd(3, 2) =

gcd(2, 3 mod 2) = gcd(2, 1) =

gcd(1, 2 mod 1) = gcd(1, 0) = 1

Extended Euclidean Algorithm

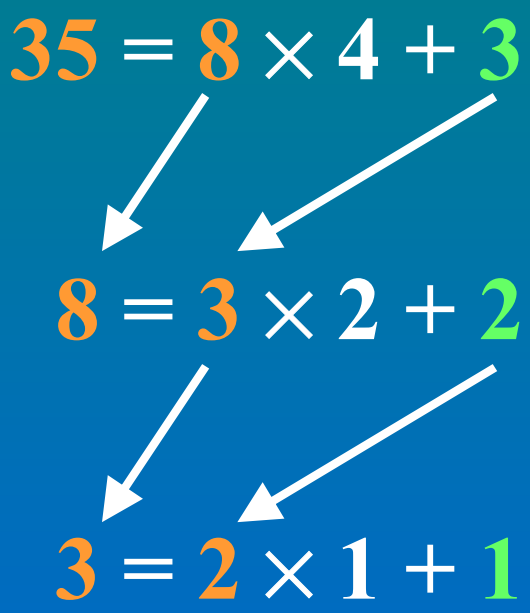
$$35 = 8 \times 4 + 3$$

Extended Euclidean Algorithm

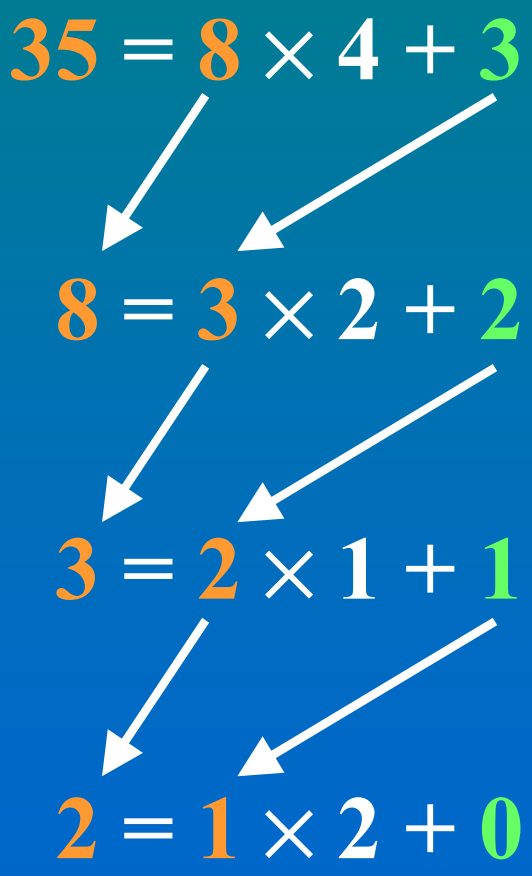
$$35 = 8 \times 4 + 3$$

$$8 = 3 \times 2 + 2$$

Extended Euclidean Algorithm



Extended Euclidean Algorithm



Extended Euclidean Algorithm

$35 = 8 \times 4 + 3$

$8 = 3 \times 2 + 2$

$3 = 2 \times 1 + 1$

$2 = 1 \times 2 + 0$

$3 = 35 - 8 \times 4$

$2 = 8 - 3 \times 2$

$1 = 3 - 2 \times 1$

Extended Euclidean Algorithm

$$3 = 35 - 8 \times 4$$

$$2 = 8 - 3 \times 2$$

$$1 = 3 - 2 \times 1$$

Extended Euclidean Algorithm

$$3 = 35 - 8 \times 4$$

$$2 = 8 - 3 \times 2$$

$$1 = 3 - 2 \times 1 = (35 - 8 \times 4) - (8 - 3 \times 2) \times 1$$

Extended Euclidean Algorithm

$$3 = 35 - 8 \times 4$$

$$2 = 8 - 3 \times 2$$

$$1 = 3 - 2 \times 1 = (35 - 8 \times 4) - (8 - 3 \times 2) \times 1$$
$$= (35 - 8 \times 4) - (8 - (35 - 8 \times 4) \times 2) \times 1$$

Extended Euclidean Algorithm

$$3 = 35 - 8 \times 4$$

$$2 = 8 - 3 \times 2$$

$$\begin{aligned} 1 &= 3 - 2 \times 1 = (35 - 8 \times 4) - (8 - 3 \times 2) \times 1 \\ &= (35 - 8 \times 4) - (8 - (35 - 8 \times 4) \times 2) \times 1 \\ &= 35 \times 3 - 8 \times 13 \end{aligned}$$

Extended Euclidean Algorithm

Given A,B > 0, set $x_1=1$, $x_2=0$, $y_1=0$, $y_2=1$,
$a_1=A$, $b_1=B$, $i=1$.

Repeat while $b_i>0$: {$i = i + 1$;
$q_i = a_{i-1}$ div b_{i-1}; $b_i = a_{i-1}-qb_{i-1}$; $a_i = b_{i-1}$;
$x_{i+1}=x_{i-1}-q_ix_i$; $y_{i+1}=y_{i-1}-q_iy_i$}.

For all i: $Ax_i + By_i = a_i$. Final $a_i = \gcd(A,B)$.

Digital Signatures

Recall that with RSA,

$D(E(Y)) = Y^{ED} \bmod N = Y$

$E(D(Y)) = Y^{DE} \bmod N = Y$

Only Alice (knowing the factorization of N) knows D. Hence only Alice can compute $D(Y) = Y^{D} \bmod N$.

This $D(Y)$ serves as Alice's signature on Y.

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 $1 < h < p\text{-}1$
- 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

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

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 $w = 1/s \bmod q$,
- Compute $a = w\mathrm{M} \bmod q$,

The Digital Signature Algorithm

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

- Compute $w = 1/s \bmod q$,
- Compute $a = w\mathrm{M} \bmod q$,
- Compute $b = wr \bmod q$,

The Digital Signature Algorithm

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

- Compute $w = 1/s \bmod q$,
- Compute $a = w\mathrm{M} \bmod q$,
- Compute $b = wr \bmod q$,
- 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 $w = 1/s \bmod q$,
- Compute $a = w\mathrm{M} \bmod q$,
- Compute $b = wr \bmod q$,
- 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

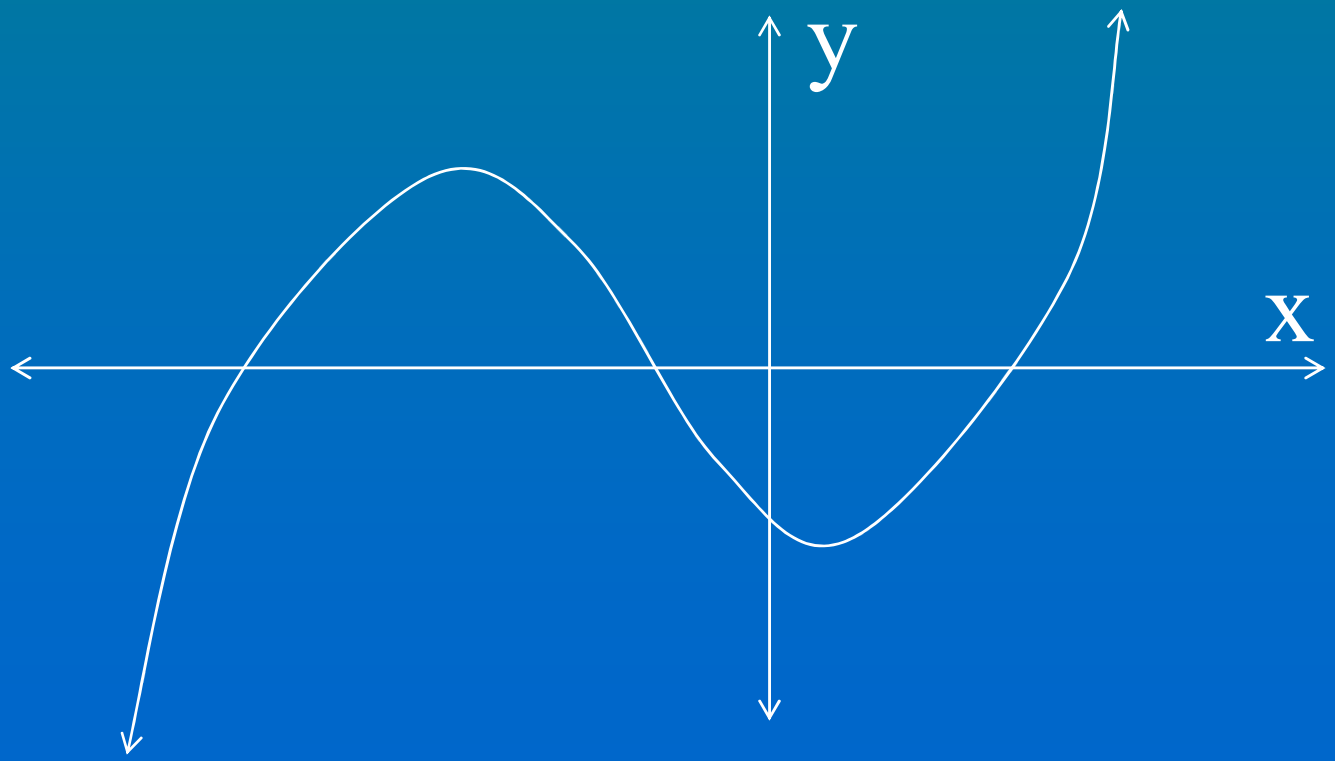
$$y^2 = x^3 + Ax + B$$

Elliptic Curves

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

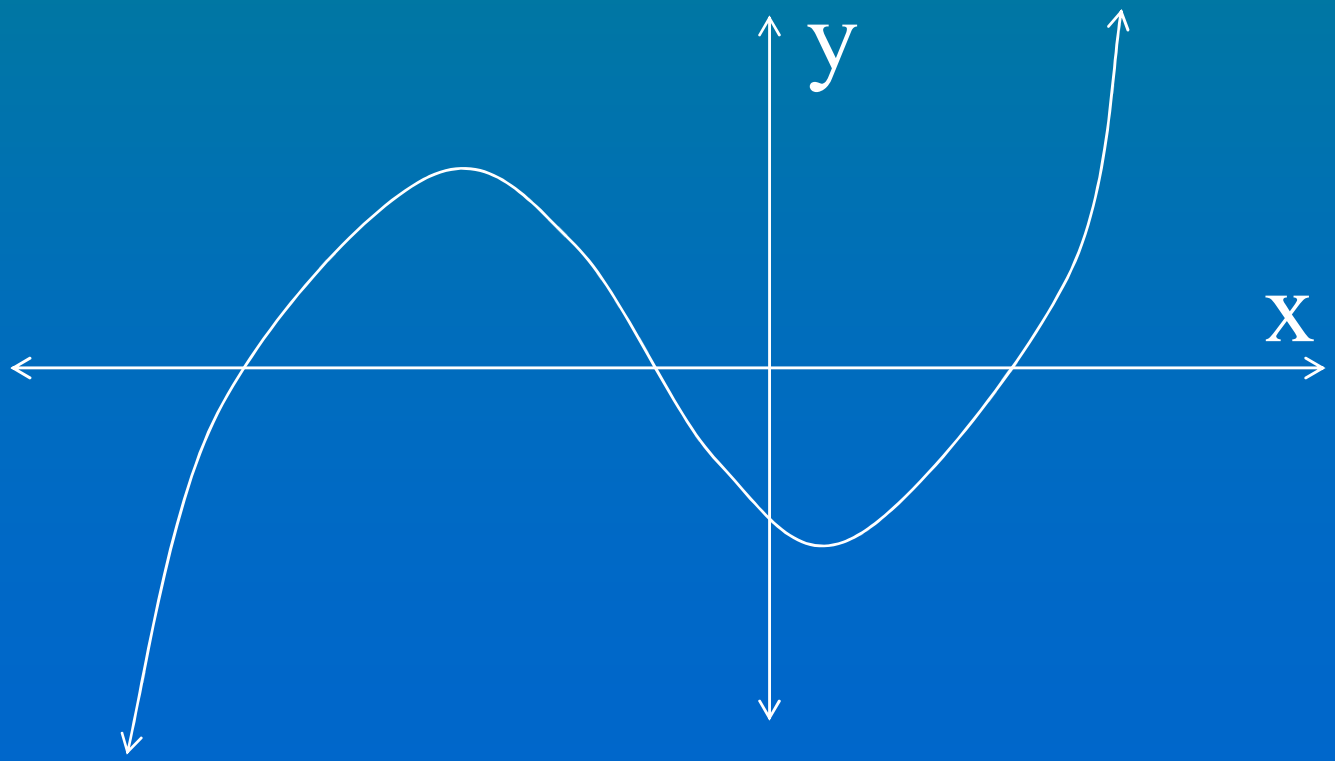
Elliptic Curves

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



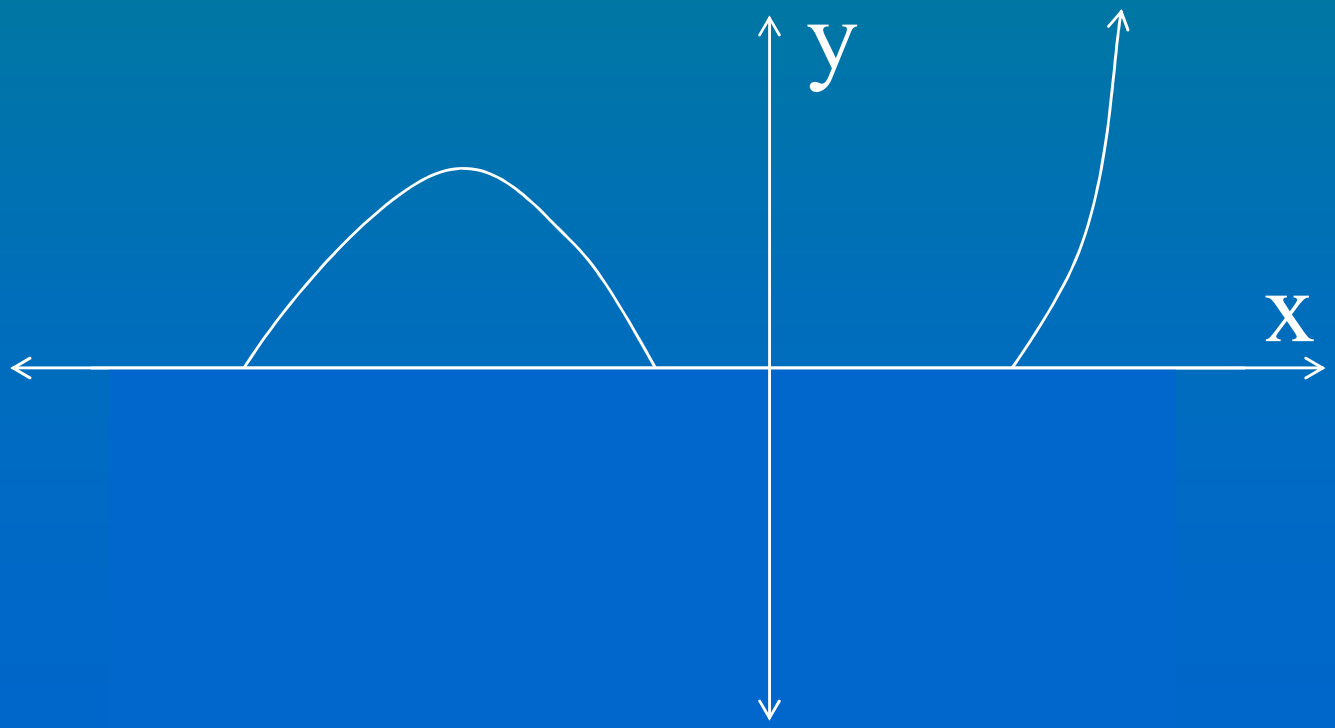
Elliptic Curves

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



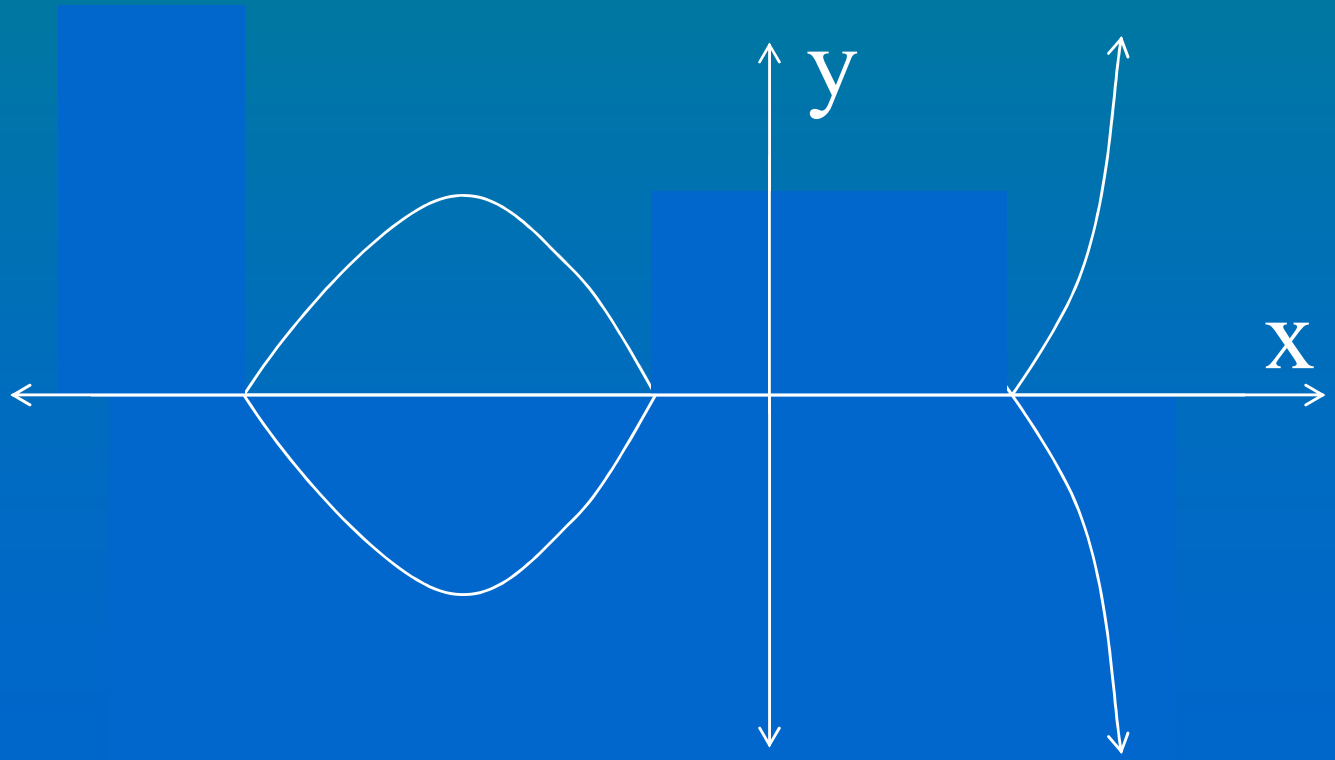
Elliptic Curves

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



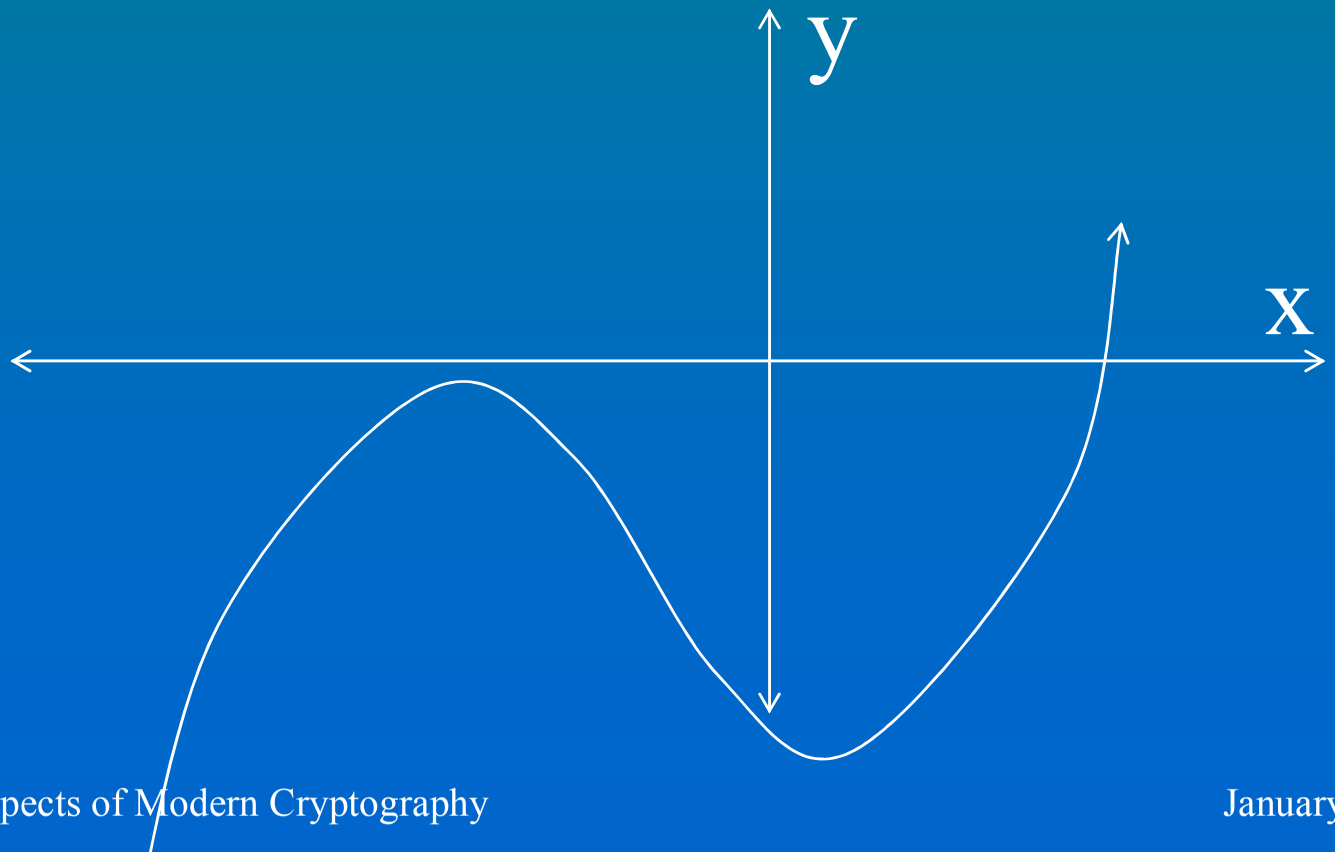
Elliptic Curves

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



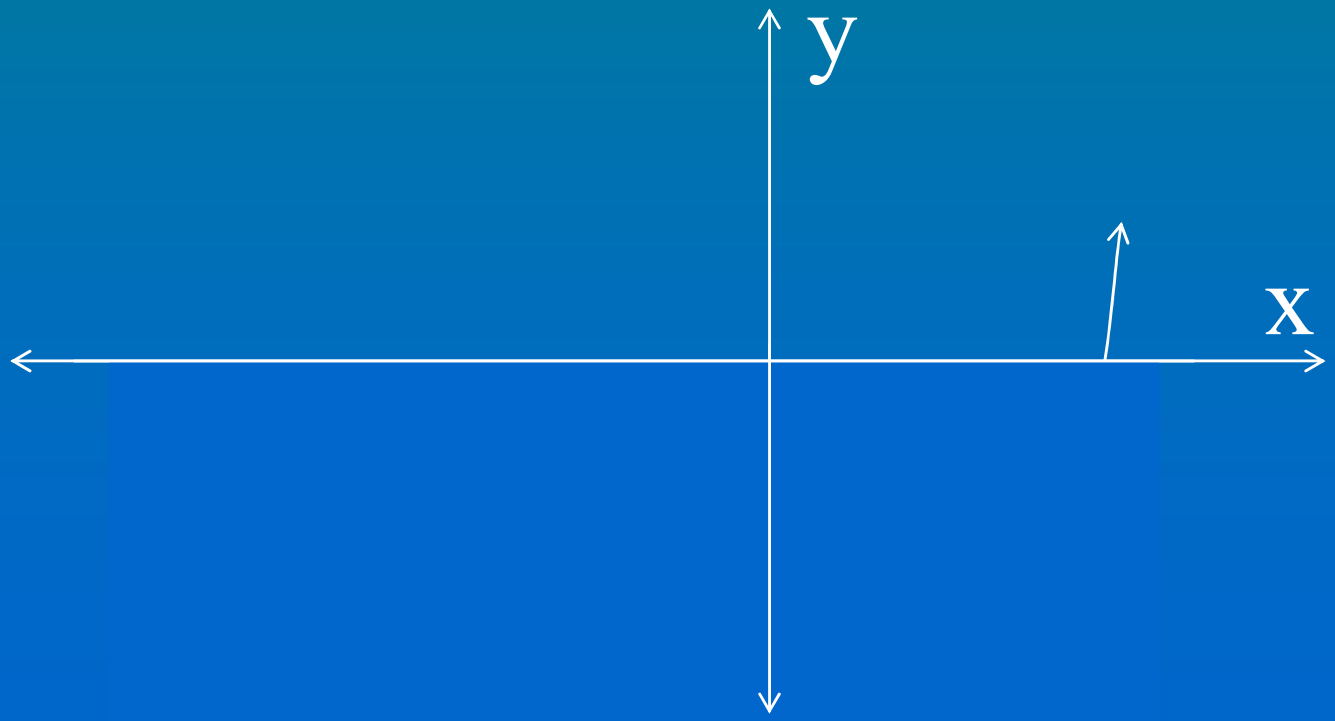
Elliptic Curves

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



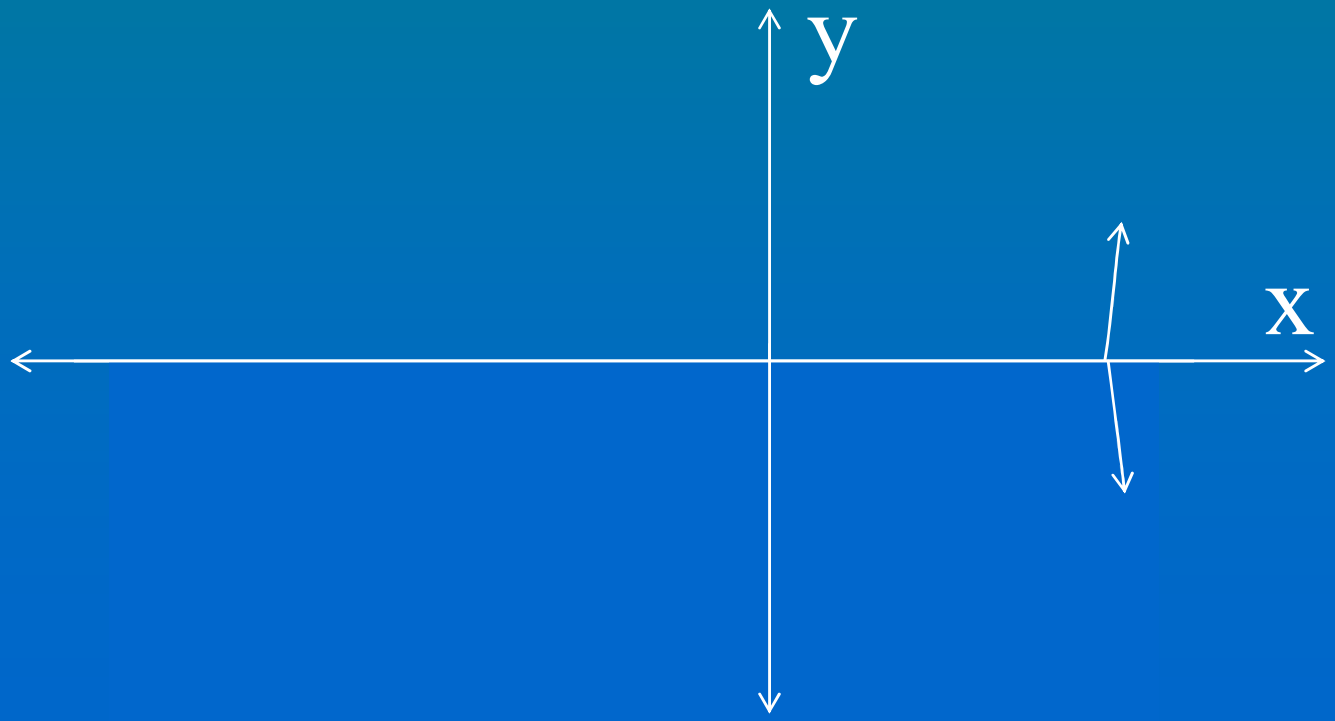
Elliptic Curves

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



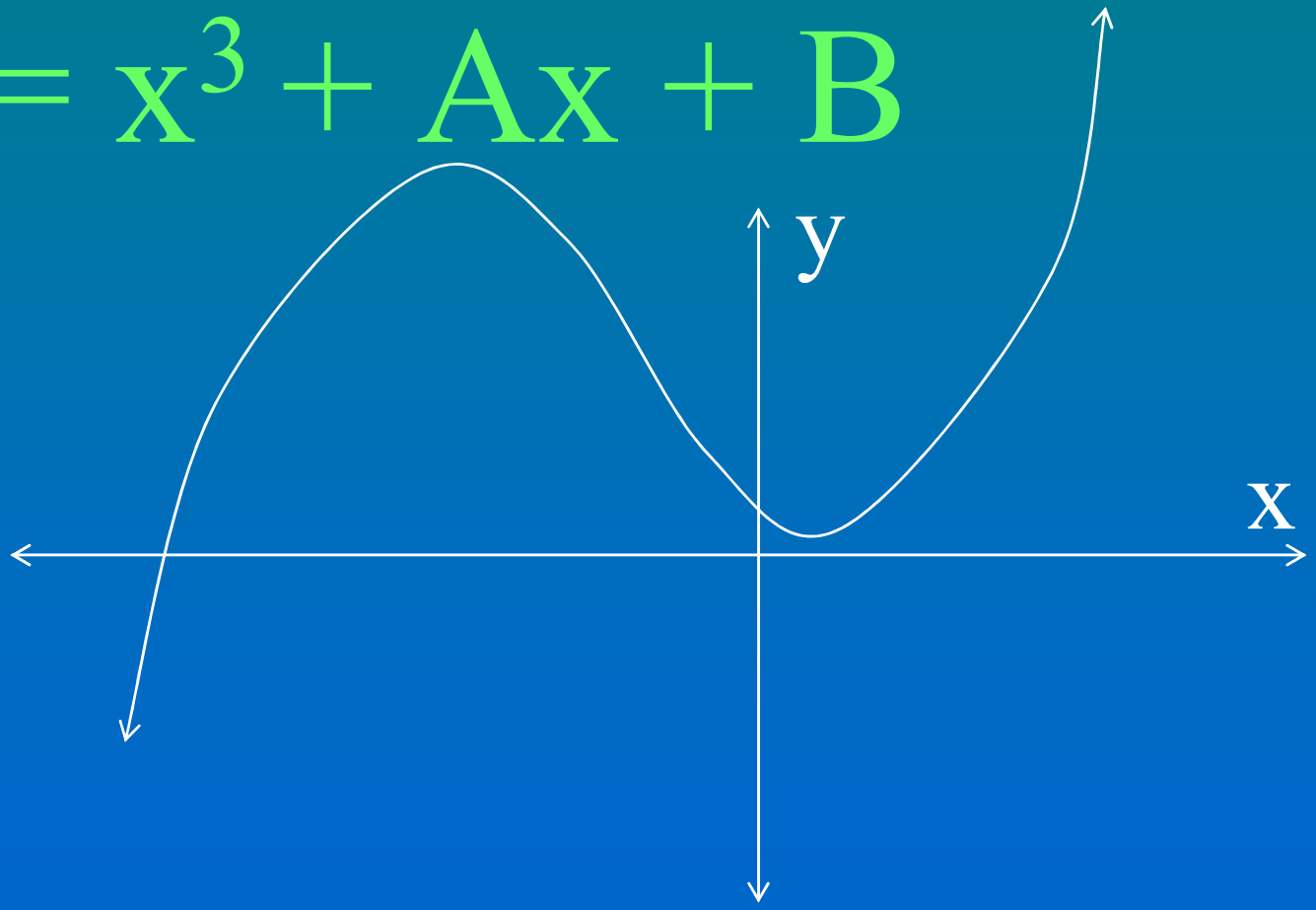
Elliptic Curves

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



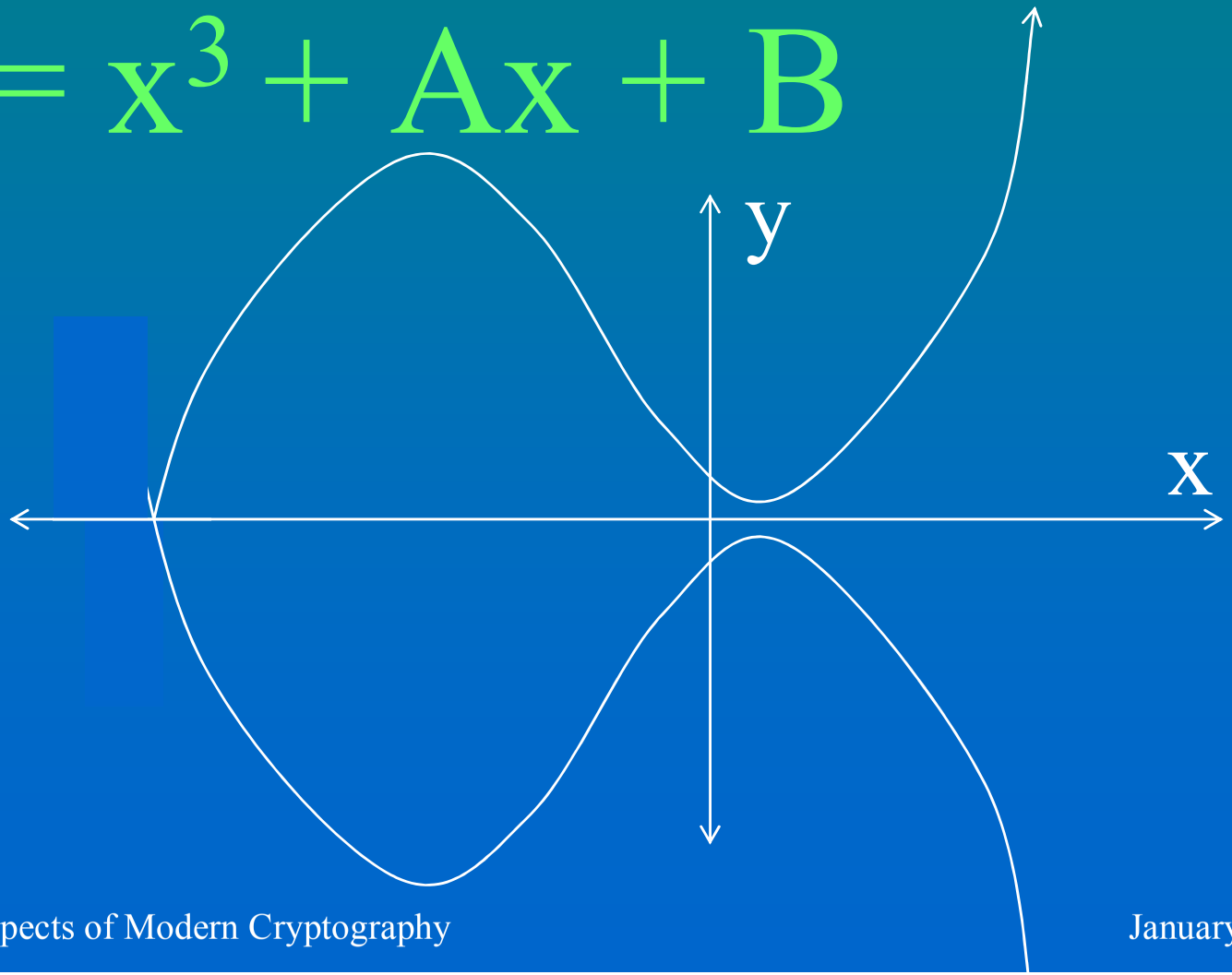
Elliptic Curves

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



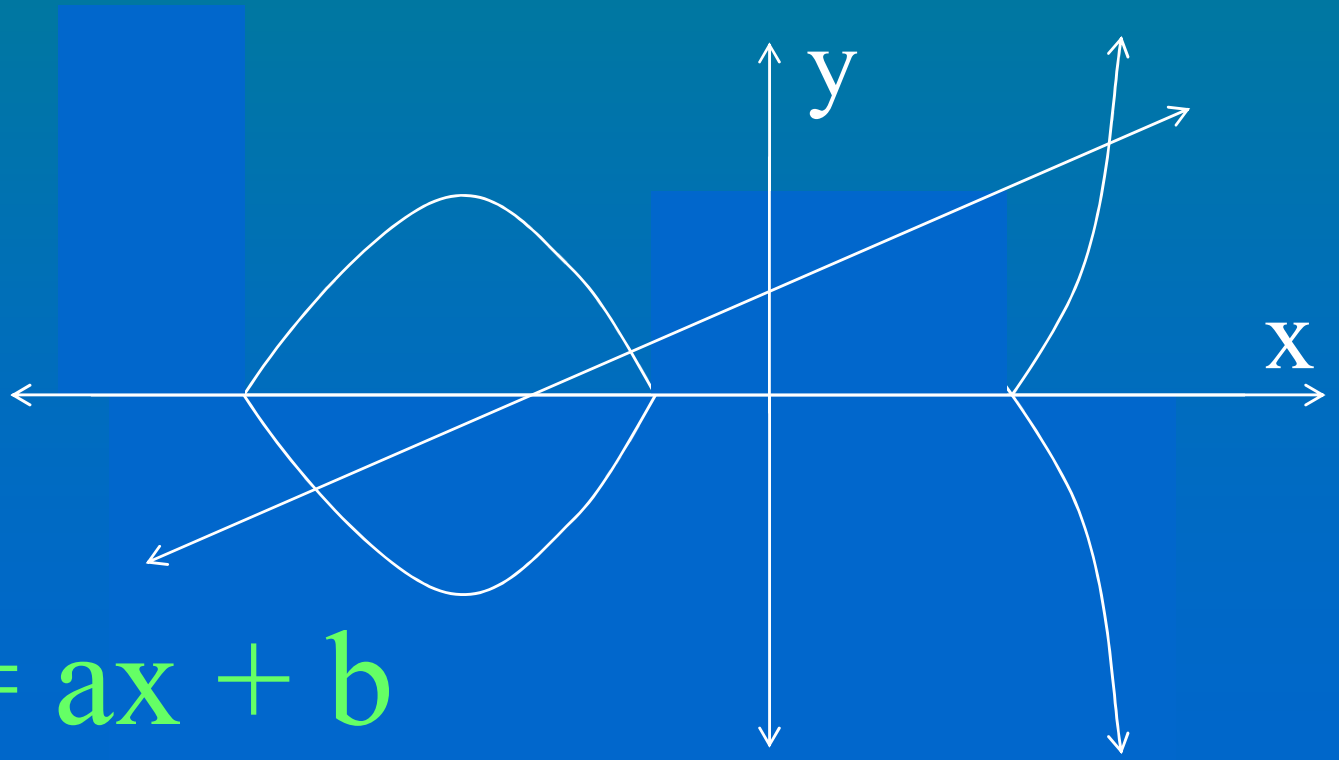
Elliptic Curves

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



Elliptic Curves Intersecting Lines

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



$$y = ax + b$$

Elliptic Curves Intersecting Lines

Non-vertical Lines

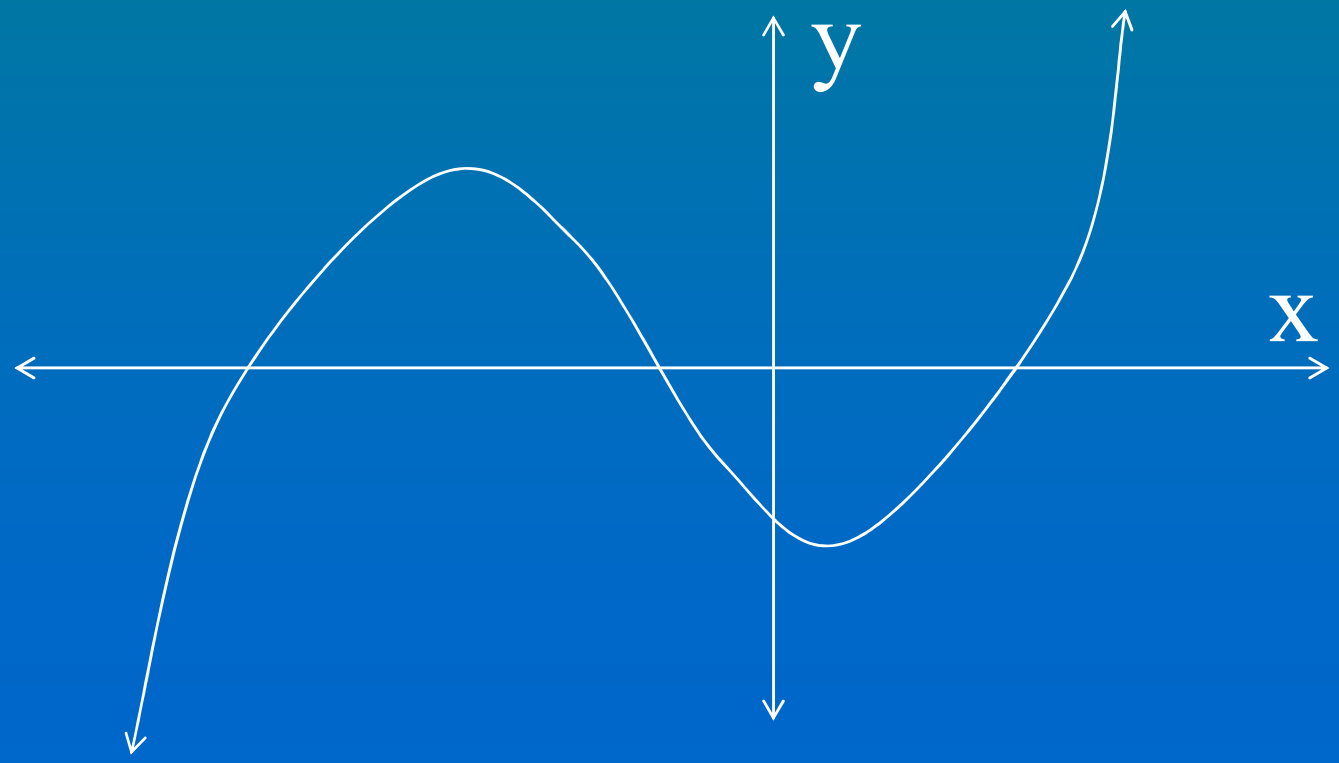
$$\begin{cases} y^2 = x^3 + Ax + B \\ y = ax + b \end{cases}$$

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

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

Elliptic Curves Intersecting Lines

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



Elliptic Curves Intersecting Lines

Non-vertical Lines

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

Elliptic Curves Intersecting Lines

Vertical Lines

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

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

$$y^2 = C$$

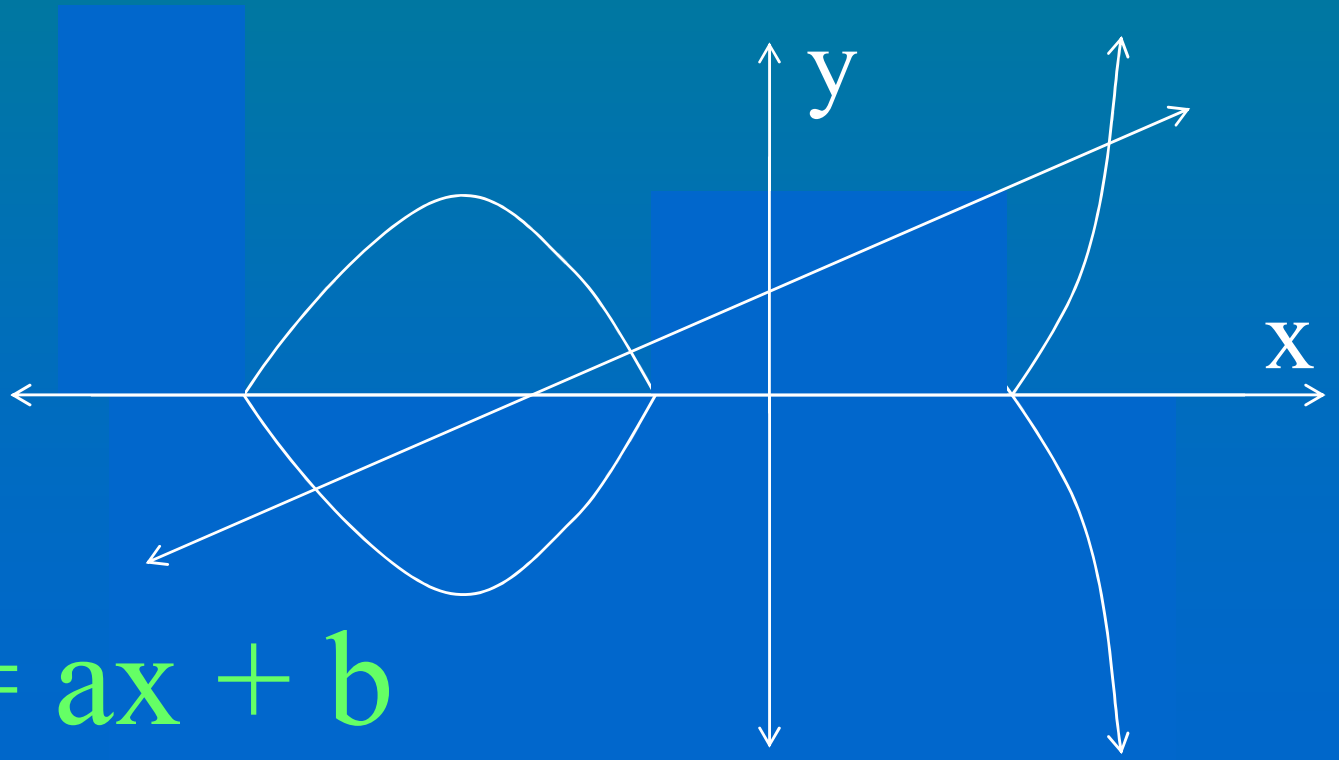
Elliptic Curves Intersecting Lines

Vertical Lines

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

Elliptic Groups

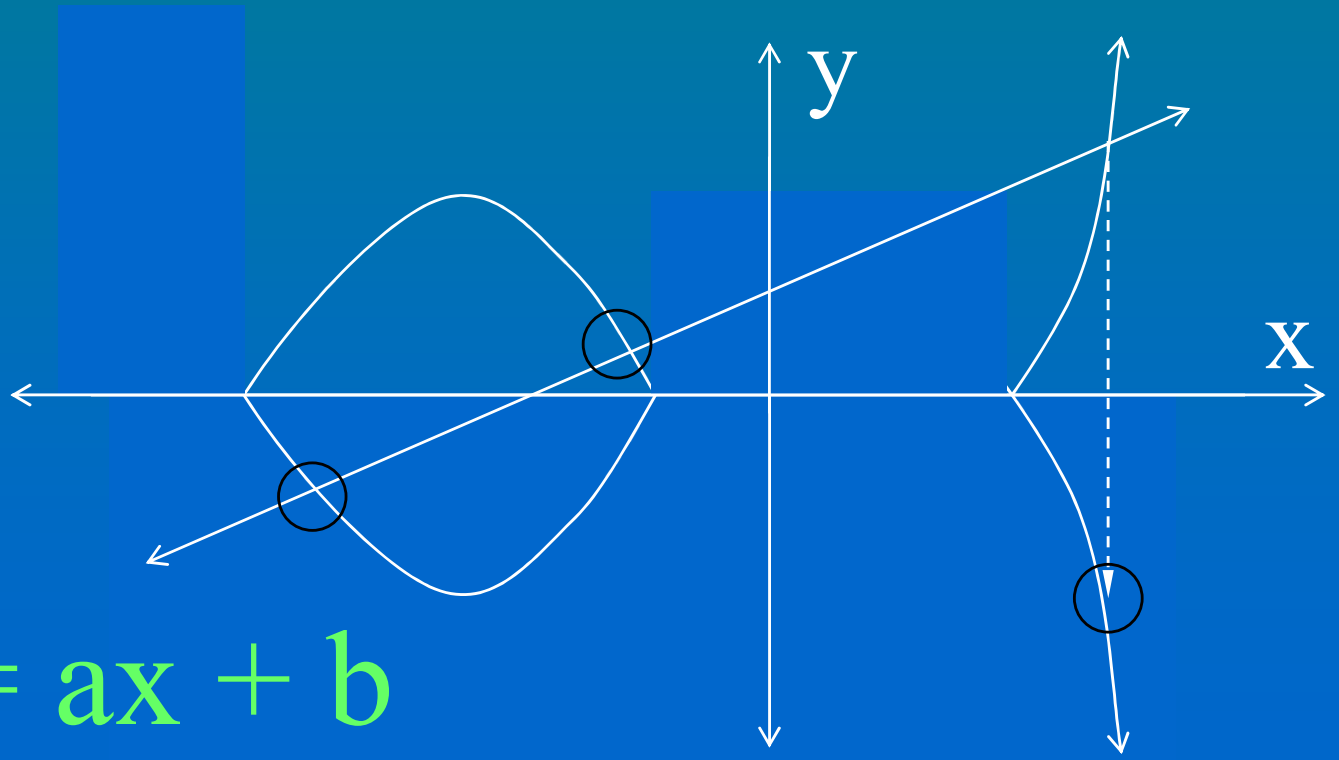
$$y^2 = x^3 + Ax + B$$



$$y = ax + b$$

Elliptic Groups

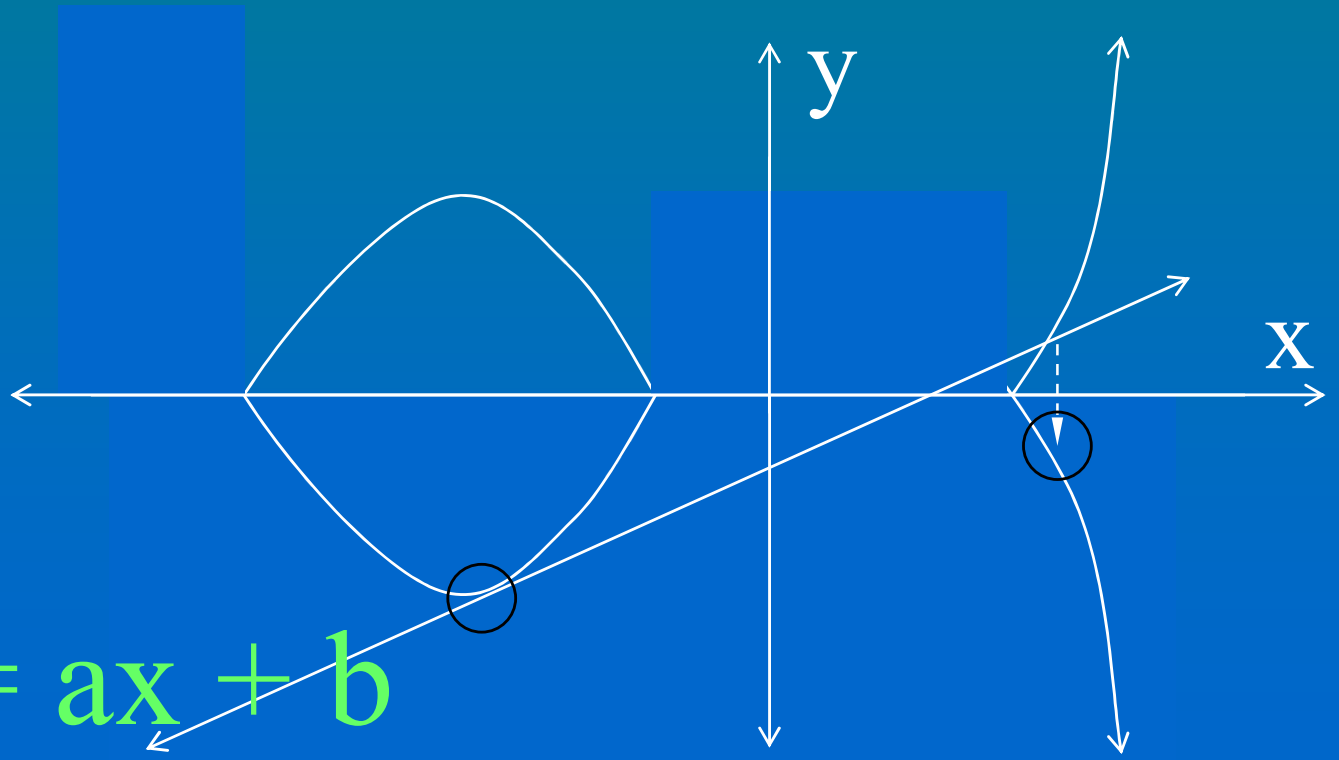
$$y^2 = x^3 + Ax + B$$



$$y = ax + b$$

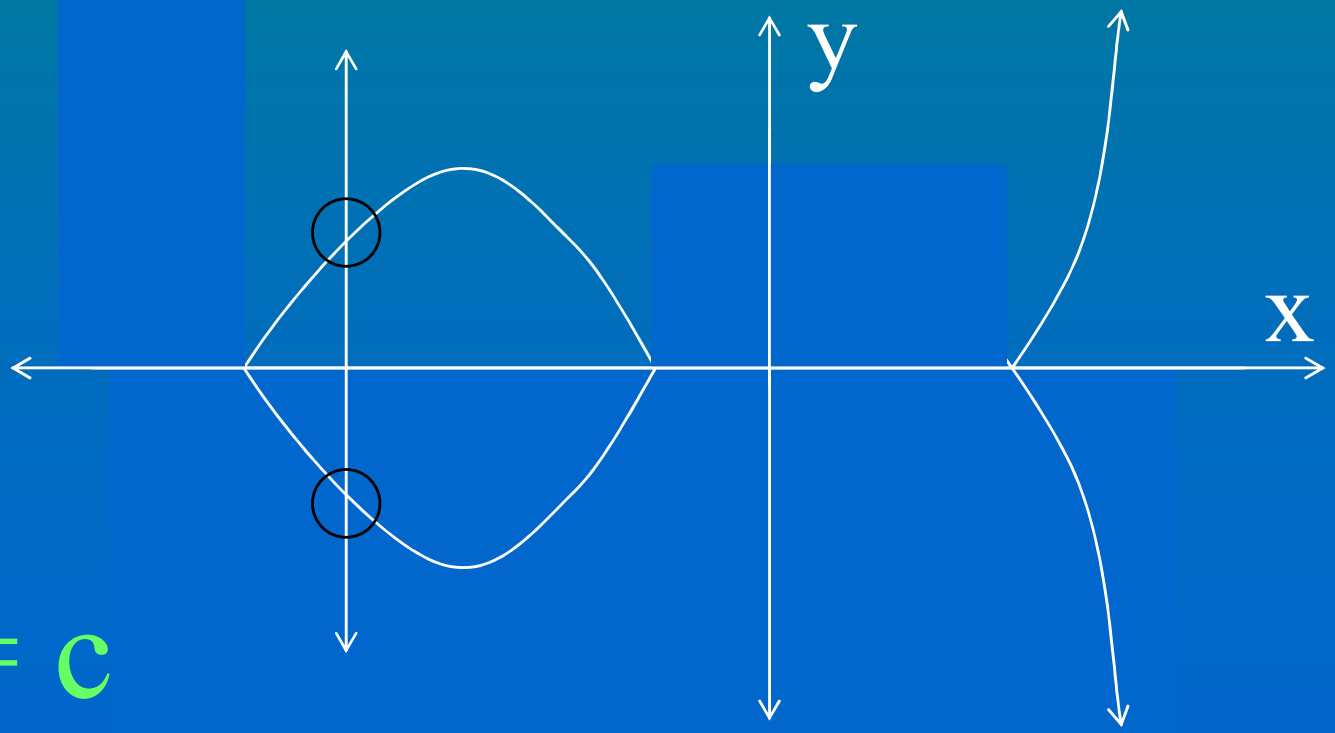
Elliptic Groups

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



Elliptic Groups

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



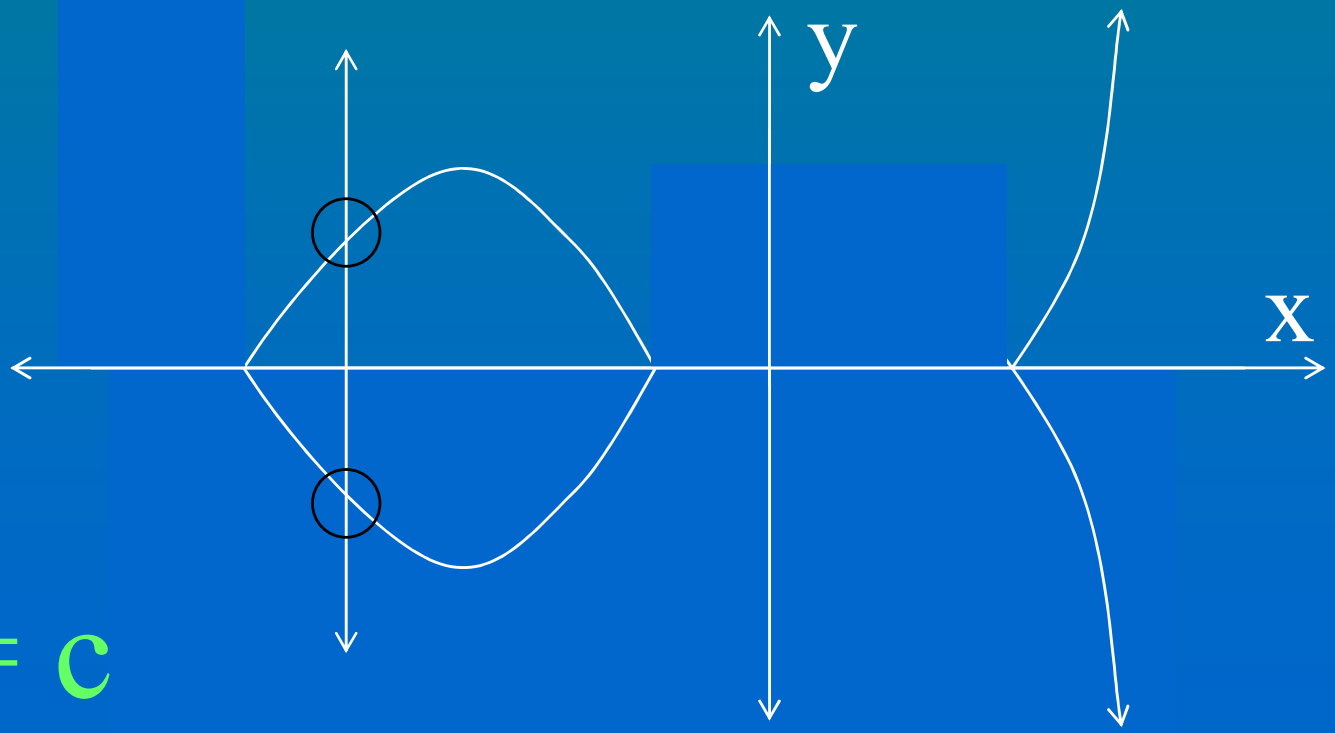
$$x = c$$

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 $y_1 = y_2 = 0$

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

$$I \times I = I$$

The Fundamental Equation

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

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

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

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