Course Business

- **Homework 3**
  - **Due:** Tuesday, October 31st at the *beginning* of class

- Professor Blocki is travelling, but will be back on Thursday

- Remaining Office Hours before Deadline:
  - Professor Blocki: Friday @ 10:30 AM
  - TA Duc Le: Wed @ 10AM
  - TA Duc Le: Mon @ 10AM

Cryptography
CS 555

Week 10:

- RSA
- Attacks on Plain RSA
- Discrete Log/DDH

Readings: Katz and Lindell Chapter 8.2-8.3,11.5.1

CS 555: Week 10: Topic 1
Finding Prime Numbers, RSA

Recap

- Number Theory Basics
- Polynomial Time Operations on Integers
  - Addition/Subtraction/Multiplication/Division
  - Exponentiation Modulo N
  - GCD
  - Find Modular Inverse of $x \in \mathbb{Z}_N^*$
- $\phi(N) = |\mathbb{Z}_N^*|$
- **Special Case:** $\phi(pq) = (p-1)(q-1)$ for distinct primes p and q

- **Key Property:** For each $g \in \mathbb{Z}_N^*$ and integer $x \in \mathbb{N}$ we have

$$[g^x \bmod N] = \left[g^{[x \bmod \boldsymbol{\phi}(\boldsymbol{N})]} \bmod N\right]$$

RSA Key-Generation

KeyGeneration(1^n)

Step 1: Pick two random n-bit primes p and q

Step 2: Let N=pq, $\phi(N) = (p-1)(q-1)$

Step 3: …

Question: How do we accomplish step one?

Bertrand's Postulate

Theorem 8.32. For any n > 1 the fraction of n-bit integers that are prime is at least $^1/_{3n}$.

GenerateRandomPrime(1^n)

For i=1 to $3n^2$:

  p'← $\{0,1\}^{n-1}$

  p← $1\|p'$

  if isPrime(p) **then**

    return p

return fail

Can we do this in polynomial time?

Bertrand's Postulate

Theorem 8.32. For any n > 1 the fraction of n-bit integers that are prime is at least ${}^{1}/_{3n}$.

GenerateRandomPrime(1^n)

For i=1 to $3n^2$:

  p' ← $\{0,1\}^{n-1}$

  p ← $1||p'$

  if isPrime(p) **then**

    return p

return fail

Assume for now that we can run isPrime(p). What are the odds that the algorithm fails?

On each iteration the probability that p is not a prime is $\left(1-\frac{1}{3n}\right)$

We fail if we pick a non-prime in all $3n^2$ iterations. The probability of failure is at most

$$\left(1-\frac{1}{3n}\right)^{3n^2} = \left(\left(1-\frac{1}{3n}\right)^{3n}\right)^{n} \leq e^{-n}$$

isPrime(p): Miller-Rabin Test

- We can check for primality of p in polynomial time in $||p||$.

Theory: Deterministic algorithm to test for primality.

- See breakthrough paper "Primes is in P"

Practice: Miller-Rabin Test (randomized algorithm)

- **Guarantee 1:** If p is prime then the test outputs YES
- **Guarantee 2:** If p is not prime then the test outputs NO except with negligible probability.

https://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf

The "Almost" Miller-Rabin Test

Input: Integer N and parameter 1^t

Output: "prime" or "composite"

for i=1 to t:

  a ← {1,…,N-1}

  if $a^{N-1} \neq 1 \bmod \mathrm{N}$ then return "composite"

Return "prime"

Claim: If N is prime then algorithm always outputs "prime"

Proof: For any a ∈ {1,…,N−1} we have $a^{N-1} = a^{\phi(N)} = 1 \; mod \; N$

The “Almost” Miller-Rabin Test

Input: Integer N and parameter 1^t

Output: “prime” or “composite”

for i=1 to t:

  a ← {1,…,N-1}

  if $a^{N-1} \neq 1 \bmod \mathrm{N}$ then return “composit

Return “prime”

Need a bit of extra work to handle Carmichael numbers (see textbook).

Fact: If N is composite and not a Carmichael number then the algorithm outputs “composite” with probability

$$1-2^{-t}$$

Back to RSA Key-Generation

KeyGeneration(1^n)

Step 1: Pick two random n-bit primes p and q

Step 2: Let N=pq, $\phi(N) = (p-1)(q-1)$

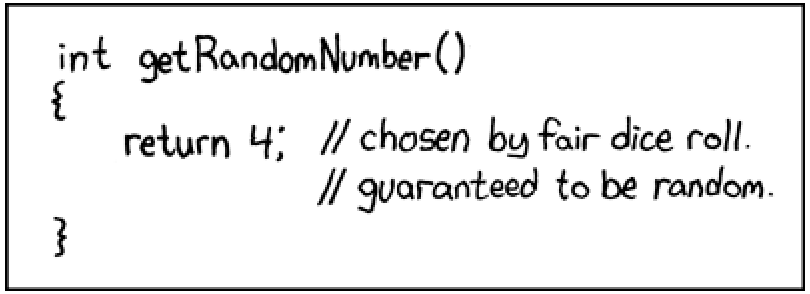
Step 3: Pick e > 1 such that gcd(e, $\phi(N)$)=1

Step 4: Set d=[e^{-1} mod $\phi(N)$] (secret key)

Return: N, e, d

- How do we find d?
- **Answer:** Use extended gcd algorithm to find e^{-1}mod $\phi(N)$.

Be Careful Where You Get Your "Random Bits!"



- RSA Keys Generated with weak PRG
  - Implementation Flaw
  - Unfortunately Commonplace
- Resulting Keys are Vulnerable
  - Sophisticated Attack
  - Coppersmith's Method


ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE FORUMS

COMPLETELY BROKEN —

Millions of high-security crypto keys crippled by newly discovered flaw

Factorization weakness lets attackers impersonate key holders and decrypt their data.

DAN GOODIN - 10/16/2017, 7:00 AM

Enlarge / 750,000 Estonian cards that look like this use a 2048-bit RSA key that can be factored in a matter of days.

The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli (CCS 2017)

(Plain) RSA Encryption

- Public Key: PK=(N,e)
- Message $\mathrm{m} \in \mathbb{Z}_{\mathrm{N}}$

$$\mathbf{Enc}_{\mathbf{PK}}(\mathrm{m}) = [m^e \bmod \mathrm{N}]$$

- **Remark:** Encryption is efficient if we use the power mod algorithm.

(Plain) RSA Decryption

- Secret Key: SK=(N,d)
- Ciphertext $c \in \mathbb{Z}_{\mathrm{N}}$

$$\mathbf{Dec_{SK}}(\mathrm{c}) = [c^d \bmod \mathrm{N}]$$

- **Remark 1:** Decryption is efficient if we use the power mod algorithm.
- **Remark 2:** Suppose that $\mathrm{m} \in \mathbb{Z}_{\mathrm{N}}^*$ and let c=**Enc$_{\mathbf{PK}}$**(m) $= [m^e \bmod \mathrm{N}]$

$$\begin{aligned} \mathbf{Dec_{SK}}(\mathrm{c}) &= [(m^e)^d \bmod \mathrm{N}] = [m^{ed} \bmod \mathrm{N}] \\ &= \left[m^{[ed \; mod \; \boldsymbol{\phi}(\boldsymbol{N})]} \bmod \mathrm{N}\right] \\ &= [m^1 \bmod \mathrm{N}] = m \end{aligned}$$

RSA Decryption

- Secret Key: SK=(N,d)
- Ciphertext $c \in \mathbb{Z}_N$

$$\mathbf{Dec_{SK}}(c) = [c^d \bmod N]$$

- **Remark 1:** Decryption is efficient if we use the power mod algorithm.
- **Remark 2:** Suppose that $m \in \mathbb{Z}_N^*$ and let c=**Enc$_{PK}$**(m) $= [m^e \bmod N]$ **then**

$$\mathbf{Dec_{SK}}(c) = m$$

- **Remark 3:** Even if $m \in \mathbb{Z}_N - \mathbb{Z}_N^*$ and let c=**Enc$_{PK}$**(m) $= [m^e \bmod N]$ **then**

$$\mathbf{Dec_{SK}}(c) = m$$

  - Use Chinese Remainder Theorem to show this

Plain RSA (Summary)

- Public Key (pk): N = pq, e such that $\mathrm{GCD}\left(\mathrm{e}, \phi(N)\right) = 1$
  - $\phi(N) = (p-1)(q-1)$ for distinct primes p and q
- Secret Key (sk): N, d such that ed=1 mod $\phi(N)$
- **Encrypt(pk=(N,e),m)** $= m^{\boldsymbol{e}}\ mod\ N$
- **Decrypt(sk=(N,d),c)** $= c^{\boldsymbol{d}}\ mod\ N$

- Decryption Works because
  $[c^d \text{mod N}] = [m^{ed} \text{mod N}] = \left[m^{[ed\ mod\ \boldsymbol{\phi}(\boldsymbol{N})]} \text{mod N}\right] = [m \text{ mod N}]$

Factoring Assumption

Let **GenModulus**(1^n) be a randomized algorithm that outputs ($N=pq,p,q$) where p and q are n-bit primes (except with negligible probability **negl**(n)).

Experiment $\text{FACTOR}_{A,n}$

1. ($N=pq,p,q$) $\leftarrow$ **GenModulus**(1^n)
2. Attacker A is given N as input
3. Attacker A outputs $p' > 1$ and $q' > 1$
4. Attacker A wins if $N=p'q'$.

Factoring Assumption

- Necessary for security of RSA.
- Not known to be sufficient.

Experiment $\mathrm{FACTOR}_{A,n}$

1. $(N=pq,p,q) \leftarrow$ **GenModulus**(1^n)
2. Attacker A is given N as input
3. Attacker A outputs p' > 1 and q' > 1
4. Attacker A wins ($\mathrm{FACTOR}_{A,n} = 1$) if and only if N=p'q'.

$$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}\ \Pr[\mathrm{FACTOR}_{A,n} = 1] \leq \mu(n)$$

RSA-Assumption

RSA-Experiment: RSA-INV$_{A,n}$

1. **Run KeyGeneration**(1^n) **to obtain (N,e,d)**
2. **Pick uniform** $\mathrm{y} \in \mathbb{Z}_{\mathrm{N}}^{*}$
3. Attacker A is given N, e, y and outputs $\mathrm{x} \in \mathbb{Z}_{\mathrm{N}}^{*}$
4. Attacker wins (RSA-INV$_{A,n}$=1) if $x^e = \mathrm{y} \bmod \mathrm{N}$

$$\forall PPT\ A\ \exists \mu\ \text{(negligible) s.t}\ \Pr[\text{RSA–INVA}_{,n} = 1] \leq \mu(n)$$

RSA-Assumption

RSA-Experiment: RSA-INV$_{A,n}$

1. **Run KeyGeneration**(1^n) **to obtain (N,e,d)**
2. **Pick uniform** $y \in \mathbb{Z}_N^*$
3. Attacker A is given N, e, y and outputs $x \in \mathbb{Z}_N^*$
4. Attacker wins (RSA-INV$_{A,n}$=1) if $x^e = \mathrm{y} \bmod \mathrm{N}$

$$\forall PPT\ A\ \exists \mu \text{ (negligible) s.t } \Pr[\text{RSA-INVA}_{,n} = 1] \leq \mu(n)$$

- Plain RSA Encryption behaves like a one-way function
- Attacker cannot invert encryption of random message

Discussion of RSA-Assumption

- Plain RSA Encryption behaves like a one-way-function

- Decryption key is a “trapdoor” which allows us to invert the OWF

- RSA-Assumption → OWFs exist

Recap

- Plain RSA
- Public Key (pk): $\mathrm{N} = \mathrm{pq}$, e such that $\mathrm{GCD}\big(\mathrm{e}, \phi(N)\big) = 1$
  - $\phi(N) = (p-1)(q-1)$ for distinct primes p and q
- Secret Key (sk): N, d such that ed=1 mod $\phi(N)$
- **Encrypt(pk=(N,e),m)** $= m^{\boldsymbol{e}}\ mod\ N$
- **Decrypt(sk=(N,d),c)** $= c^{\boldsymbol{d}}\ mod\ N$

- Decryption Works because
  $[c^d \text{mod N}] = [m^{ed} \text{mod N}] = \left[m^{[ed\ mod\ \boldsymbol{\phi}(\boldsymbol{N})]} \text{mod N}\right] = [m \text{ mod N}]$

Mathematica Demo

https://www.cs.purdue.edu/homes/jblocki/courses/555_Spring17/slides/Lecture24Demo.nb

Note: Online version of mathematica available at https://sandbox.open.wolframcloud.com (reduced functionality, but can be used to solve homework bonus problems)

(Toy) RSA Implementation in Mathematica

(* Random Seed 123456 is not secure, but it allows us to repeat the experiment *)

SeedRandom[123456]

(* Step 1: Generate primes for an RSA key *)

p = RandomPrime[{10^1000, 10^1050}];

q = RandomPrime[{10^1000, 10^1050}];

NN = p q; (*Symbol N is protected in mathematica *)

phi = (p - 1) (q - 1);

https://www.cs.purdue.edu/homes/jblocki/courses/555_Spring17/slides/Lecture24Demo.nb

(Toy) RSA Implementation in Mathematica

(* Step 1.A: Find e *)

GCD[phi,7]

Output: 7

(* GCD[phi,7] is not 1, so he have to try a different value of e *)

GCD[phi,3]

Output: 1

(* We can set e=3 *)

e=3;

https://www.cs.purdue.edu/homes/jblocki/courses/555_Spring17/slides/Lecture24Demo.nb

(Toy) RSA Implementation in Mathematica

(* Step 1.B find d s.t. ed = 1 mod N by using the extended GCD algorithm *)

(* Mathematica is clever enough to do this automatically *)

Solve[e x == 1, Modulus->phi]

Output:

{{x->3646968059066302830170062613288386727271872890520508 8...

....

3940694217786102094256244409800844813981 31}}

(* We can now set d = x *)

d=364696805…. 8131;

(Toy) RSA Implementation in Mathematica

(* Double Check 1 = [ed mod $\phi(N)$] *)

Mod [e d, (p-1)(q-1)]

Output: 1

(* Encrypt the message 200, c= m^e mod N *)

m = 200;

PowerMod[m,e,NN]

Output: 8 000 000

(Toy) RSA Implementation in Mathematica

(* Encrypt the message 200, c= m^e mod N *)

m = 200;

PowerMod[m,e,NN]

Output: 8 000 000

(* Hm...That doesn't seem too secure *)

CubeRoot[PowerMod[m,e,NN]]

Output: 200

(* Moral: if $m^e < N$ then Plain RSA does not hide the message m. *)

RSA Implementation in Mathematica

(* Encrypt a larger message, c= m^e mod N *)

```
SeedRandom[1234567];
m2= RandomInteger[{10^1500,10^1501}];
c=PowerMod[m2,e,NN]
```

Output: 405215834903772786……… 388068292685976133

(* Does it Decrypt Properly? *)

```
PowerMod[c,d, NN]-m2
```

Output: 0

(* Yes! *)

CS 555: Week 10: Topic 2
Attacks on Plain RSA

(Plain) RSA Discussion

- We have not introduced security models like CPA-Security or CCA-security for Public Key Cryptosystems

- However, notice that (Plain) RSA Encryption is stateless and deterministic.

→Plain RSA is not secure against chosen-plaintext attacks

- As we will see Plain RSA is also highly vulnerable to chosen-ciphertext attacks

(Plain) RSA Discussion

- However, notice that (Plain) RSA Encryption is stateless and deterministic.

→Plain RSA is not secure against chosen-plaintext attacks

- **Remark:** In a public key setting the attacker who knows the public key *always* has access to an encryption oracle

- Encrypted messages with low entropy are particularly vulnerable to brute-force attacks
  - **Example:** If $m < B$ then attacker can recover m from $\mathrm{c} = \mathrm{Enc}_{\mathrm{pk}}(m)$ after at most B queries to encryption oracle (using public key)

Chosen Ciphertext Attack on Plain RSA

1. Attacker intercepts ciphertext $c = [m^e \bmod \mathrm{N}]$
2. Attacker generates ciphertext c' for secret message 2m as follows
3. c' $= [(c2^e) \bmod \mathrm{N}]$
4. $= [(m^e 2^e) \bmod \mathrm{N}]$
5. $= [(2m)^e \bmod \mathrm{N}]$
6. Attacker asks for decryption of $[c2^e \bmod \mathrm{N}]$ and receives 2m.
7. Divide by two to recover message

Above Example: Shows plain RSA is highly vulnerable to ciphertext-tampering attacks

More Weaknesses: Plain RSA with small e

- (Small Messages) If $m^e < N$ then we can decrypt $c = m^e \bmod N$ directly e.g., $m=c^{(1/e)}$

- (Partially Known Messages) If an attacker knows first 1-(1/e) bits of secret message $m = m_1 \| ??$ then he can recover m given
$$\mathbf{Encrypt}(pk, m) = m^e \bmod \mathrm{N}$$

Theorem[Coppersmith]: If p(x) is a polynomial of degree e then in polynomial time (in log(N), e) we can find all m such that p(m) = 0 mod N and $|m|<N^{(1/e)}$

More Weaknesses: Plain RSA with small e

Theorem[Coppersmith]: If p(x) is a polynomial of degree e then in polynomial time (in log(N), e) we can find all m such that p(m) = 0 mod N and $|m| < N^{(1/e)}$

Example: e = 3, $m = m_1 \| m_2$ and attacker knows $m_1 (2k\ bits)$ and $\boldsymbol{c} = (m_1 \| m_2)^e \mathrm{mod\ N}$, but not $m_2 (k\ bits)$

$$p(x) = \left(2^k m_1 + x\right)^3 - c$$

Polynomial has a small root mod N at x= m_2 and coppersmith's method will find it!

D. Coppersmith (1996). "Finding a Small Root of a Univariate Modular Equation".

More Weaknesses: Plain RSA with small e

Theorem[Coppersmith]: Can also find small roots of bivariate polynomial $\mathrm{p}(\boldsymbol{x_1}, \boldsymbol{x_2})$

- **Similar Approach used to factor weak RSA secret keys N=$\mathbf{q_1 q_2}$**
- Weak PRG → Can guess many of the bits of prime factors
  - Obtain $\widetilde{q_1} \approx q_1$ and $\widetilde{q_2} \approx q_2$
- Coppersmith Attack: Define polynomial p(.,.) as follows
$$\mathrm{p}(\boldsymbol{x_1}, \boldsymbol{x_2}) = (\boldsymbol{x_1} + \widetilde{\boldsymbol{q_1}})(\boldsymbol{x_2} + \widetilde{\boldsymbol{q_2}}) - \boldsymbol{N}$$
- **Small Roots of** $\mathrm{p}(\boldsymbol{x_1}, \boldsymbol{x_2})$**:** $x_1 = q_1 - \widetilde{q_1}$ and $x_2 = q_2 - \widetilde{q_2}$

D. Coppersmith (1996). "Finding a Small Root of a Bivariate Integer Equation; Factoring with high bits known" ars TECHNICA | BIZ & IT | TECH | SCIENCE | POLICY | CARS | GAMING & CULTURE | FORUMS

COMPLETELY BROKEN —

Millions of high-security crypto keys crippled by newly discovered flaw

Factorization weakness lets attackers impersonate key holders and decrypt their data.

DAN GOODIN - 10/16/2017, 7:00 AM

Steve Jurvetson

Enlarge / 750,000 Estonian cards that look like this use a 2048-bit RSA key that can be factored in a matter of days.

The Return of ***Coppersmith's Attack****: Practical Factorization of Widely Used RSA Moduli (CCS 2017)*

Fixes for Plain RSA

- Approach 1: RSA-OAEP
  - Incorporates random nonce r
  - CCA-Secure (in random oracle model)

- Approach 2: Use RSA to exchange symmetric key for Authenticated Encryption scheme (e.g., AES)
  - Key Encapsulation Mechanism (KEM)

- More details in future lectures…stay tuned!
  - For now we will focus on attacks on Plain RSA

Chinese Remainder Theorem

Theorem: Let N = pq (where gcd(p,q)=1) be given and let $f: \mathbb{Z}_{\mathrm{N}} \to \mathbb{Z}_p \times \mathbb{Z}_q$ be defined as follows

$$f(x) = ([x \bmod p], [x \bmod q])$$

then

- f is a bijective mapping (invertible)
- f and its inverse $f^{-1}: \mathbb{Z}_p \times \mathbb{Z}_q \to \mathbb{Z}_{\mathrm{N}}$ can be computed efficiently
- $f(x+y) = f(x) + f(y)$
- The restriction of f to $\mathbb{Z}_N^*$ yields a bijective mapping to $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$
- For inputs $x, y \in \mathbb{Z}_N^*$ we have $f(x)f(y) = f(xy)$

Chinese Remainder Theorem

Application of CRT: Faster computation

Example: Compute [11^{53} mod 15]

f(11)=([-1 mod 3],[1 mod 5])

f(11^{53}) =([$(-1)^{53}$ mod 3],[1^{53} mod 5])= (-1,1)

f^{-1}(-1,1)=11

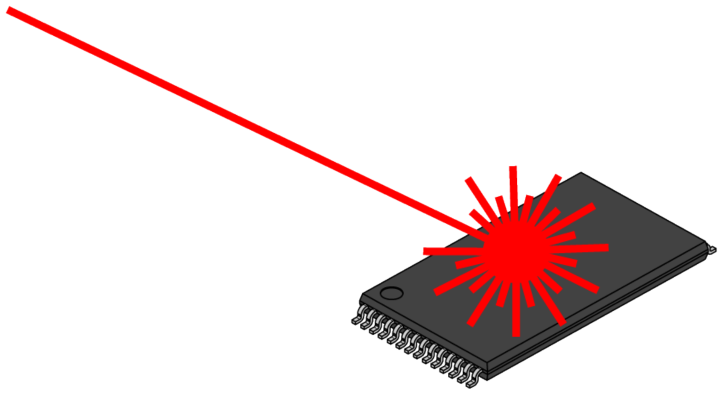
Thus, 11=[11^{53} mod 15]

A Side Channel Attack on RSA with CRT

- Suppose that decryption is done via Chinese Remainder Theorem for speed.

$$\mathbf{Dec}_{sk}(c) = c^d \bmod N \leftrightarrow \left(c^d \bmod p, c^d \bmod q\right)$$

- Attacker has physical access to smartcard
  - Can mess up computation of $c^d \bmod p$
  - Response is $\mathrm{R} \leftrightarrow \left(r, c^d \bmod q\right)$
  - $\mathrm{R} - \mathrm{m} \leftrightarrow (r - m \bmod p, 0 \bmod q)$
  - GCD(R-m,N)=q



Recovering Encrypted Message faster than Brute-Force

Claim: Let m < 2^n be a secret message. For some constant $\alpha = \frac{1}{2} + \varepsilon$. We can recover m in in time $T = 2^{\alpha n}$ with high probability.

For r=1,…,T

  let $\mathrm{x_r} = [cr^{-e} mod\ N]$, where $r^{-e} = (r^{-1})^e mod\ N$

Sort $\mathbf{L} = \{(\boldsymbol{r}, \boldsymbol{x_r})\}_{\boldsymbol{r=1}}^{\boldsymbol{T}}$ **(by the x_r values)**

For s=1,…,T

  if $[s^e mod\ N] = \boldsymbol{x_r}$ for some r **then**

    return $[rs\ mod\ N]$

Recovering Encrypted Message faster than Brute-Force

For r=1,…,T

  let $\mathrm{x_r} = [cr^{-e} mod\ N]$, where $r^{-e} = (r^{-1})^e mod\ N$

Sort $\mathbf{L} = \{(\boldsymbol{r}, \boldsymbol{x_r})\}_{\boldsymbol{r=1}}^{\boldsymbol{T}}$ **(by the x_r values)**

For s=1,…,T

  if $[s^e mod\ N] = x_r$ for some r **then**

    return $[rs\ mod\ N]$

Analysis: $[rs\ mod\ N] = [r(s^e)^d\ mod\ N] = [r(x_r)^d\ mod\ N]$

$$= [r(cr^{-e})^d\ mod\ N] = [rr^{-ed}(c)^d\ mod\ N]$$

$$= [rr^{-1}m\ mod\ N] = \mathrm{m}$$

Recovering Encrypted Message faster than Brute-Force

For r=1,…,T

  let $\mathrm{x_r} = [cr^{-e} mod\ N]$, where $r^{-e} = (r^{-1})^e mod\ N$

Sort $\mathbf{L} = \{(\boldsymbol{r}, \boldsymbol{x_r})\}_{\boldsymbol{r=1}}^{\boldsymbol{T}}$ **(by the $\mathbf{x_r}$ values)**

For s=1,…,T

  if $[s^e mod\ N] = x_r$ for some r **then**

    return $[rs\ mod\ N]$

Fact: some constant $\alpha = \frac{1}{2} + \varepsilon$ setting $T = 2^{\alpha n}$ with high probability we will find a pair **s** and $\mathbf{x_r}$ with $[s^e mod\ N] = xr$.

Recovering Encrypted Message faster than Brute-Force

Claim: Let m < 2^n be a secret message. For some constant $\alpha = \frac{1}{2} + \varepsilon$. We can recover m in in time $T = 2^{\alpha n}$ with high probability.

Roughly $\sqrt{B}$ steps to find a secret message m < B

CS 555: Week 10: Topic 3
Discrete Log + DDH Assumption

(Recap) Finite Groups

Definition: A (finite) group is a (finite) set $\mathbb{G}$ with a binary operation $\circ$ (over G) for which we have

- (**Closure**:) For all $\mathrm{g}, \mathrm{h} \in \mathbb{G}$ we have $\mathrm{g} \circ \mathrm{h} \in \mathbb{G}$
- **(Identity**:) There is an element $\mathrm{e} \in \mathbb{G}$ such that for all $\mathrm{g} \in \mathbb{G}$ we have
$$\mathrm{g} \circ \mathrm{e} = \mathrm{g} = \mathrm{e} \circ \mathrm{g}$$
- **(Inverses**:) For each element $\mathrm{g} \in \mathbb{G}$ we can find $\mathrm{h} \in \mathbb{G}$ such that $\mathrm{g} \circ \mathrm{h} = \mathrm{e}$. We say that h is the inverse of g.
- (**Associativity:**) For all $\mathrm{g}_1, \mathrm{g}_2, \mathrm{g}_3 \in \mathbb{G}$ we have
$$(\mathrm{g}_1 \circ \mathrm{g}_2) \circ \mathrm{g}_3 = \mathrm{g}_1 \circ (\mathrm{g}_2 \circ \mathrm{g}_3)$$

We say that the group is **abelian** if

- (**Commutativity:**) For all $\mathrm{g}, \mathrm{h} \in \mathbb{G}$ we have $\mathrm{g} \circ \mathrm{h} = \mathrm{h} \circ \mathrm{g}$

Finite Abelian Groups (Examples)

- **Example 1:** $\mathbb{Z}_N$ when $\circ$ denotes addition modulo N
- Identity: 0, since 0 $\circ$ x =[0+x mod N] = [x mod N].
- Inverse of x? Set x^{-1}=N-x so that [x^{-1}+x mod N] = [N-x+x mod N] = 0.

- **Example 2:** $\mathbb{Z}_N^*$ when $\circ$ denotes multiplication modulo N
- Identity: 1, since 1$\circ$ x =[1(x) mod N] = [x mod N].
- Inverse of x? Run extended GCD to obtain integers a and b such that

$$ax + bN = \text{gcd}(x, N) = 1$$

Observe that: x^{-1} = a. Why?

Cyclic Group

- Let $\mathbb{G}$ be a group with order $\mathrm{m} = |\mathbb{G}|$ with a binary operation $\circ$ (over G) and let $\mathrm{g} \in \mathbb{G}$ be given consider the set

$$\langle g \rangle = \{g^0, g^1, g^2, \dots\}$$

Fact: $\langle g \rangle$ defines a subgroup of $\mathbb{G}$.

- Identity: g^0
- Closure: $g^i \circ g^j = g^{i+j} \in \langle g \rangle$
- g is called a "generator" of the subgroup.

Fact: Let $\mathrm{r} = |\langle g \rangle|$ then $g^i = g^j$ if and only if $i = j \bmod r$. Also m is divisible by r.

Finite Abelian Groups (Examples)

Fact: Let p be a prime then $\mathbb{Z}_p^*$ is a cyclic group of order p-1.

- **Note**: Number of generators g s.t. of $\langle g \rangle = \mathbb{Z}_p^*$ is $\frac{|\phi(p-1)|}{p-1}$

Example (non-generator): p=7, g=2

<2>={1,2,4}

Example (generator): p=7, g=5

<2>={1,5,4,6,2,3}

Discrete Log Experiment $DLog_{A,G}(n)$

1. Run $G(1^n)$ to obtain a cyclic group $\mathbb{G}$ of order q (with $\|q\| = n$) and a generator g such that $< \mathrm{g} >= \mathbb{G}$.
2. Select $\mathrm{h} \in \mathbb{G}$ uniformly at random.
3. Attacker A is given $\mathbb{G}$, q, g, h and outputs integer x.
4. Attacker wins ($DLog_{A,G}(n)=1$) if and only if $g^x=h$.

We say that the discrete log problem is hard relative to generator G if

$$\forall PPT\ A\ \exists\mu\ (\text{negligible})\ \text{s.t}\ \Pr[DLog_{A,n} = 1] \le \mu(n)$$

Diffie-Hellman Problems

Computational Diffie-Hellman Problem (CDH)

- Attacker is given $\mathrm{h}_1 = g^{x_1} \in \mathbb{G}$ and $\mathrm{h}_2 = g^{x_2} \in \mathbb{G}$.
- Attackers goal is to find $g^{x_1 x_2} = (\mathrm{h}_1)^{x_2} = (\mathrm{h}_2)^{x_1}$
- **CDH Assumption**: For all PPT A there is a negligible function negl upper bounding the probability that A succeeds with probability at most negl(n).

Decisional Diffie-Hellman Problem (DDH)

- Let $\mathrm{z}_0 = g^{x_1 x_2}$ and let $\mathrm{z}_1 = g^{r}$, where x_1,x_2 and r are random
- Attacker is given g^{x_1}, g^{x_2} and z_b (for a random bit b)
- Attackers goal is to guess b
- **DDH Assumption**: For all PPT A there is a negligible function negl such that A succeeds with probability at most ½ + negl(n).

Secure key-agreement with DDH

1. Alice publishes g^{x_A} and Bob publishes g^{x_B}
2. Alice and Bob can both compute $K_{A,B} = g^{x_B x_A}$ but to Eve this key is indistinguishable from a random group element (by DDH)

Remark: Protocol is vulnerable to Man-In-The-Middle Attacks if Bob cannot validate g^{x_A}.

Can we find a cyclic group where DDH holds?

- **Example 1:** $\mathbb{Z}_p^*$ where p is a random n-bit prime.
  - CDH is believed to be hard
  - DDH is *not* hard (Exercise 13.15)

- **Theorem:** Let p=rq+1 be a random n-bit prime where q is a large λ-bit prime then the set of r^{th} residues modulo p is a cyclic subgroup of order q. Then $\mathbb{G}_r = \left\{[h^r mod\ p] \middle| h \in \mathbb{Z}_p^*\right\}$ is a cyclic subgroup of $\mathbb{Z}_p^*$ of order q.
  - **Remark 1:** DDH is believed to hold for such a group
  - **Remark 2:** It is easy to generate uniformly random elements of $\mathbb{G}_r$
  - **Remark 3:** Any element (besides 1) is a generator of $\mathbb{G}_r$

Can we find a cyclic group where DDH holds?

- **Theorem:** Let p=rq+1 be a random n-bit prime where q is a large λ-bit prime then the set of rth residues modulo p is a cyclic subgroup of order q. Then $\mathbb{G}_r = \{[h^r mod\ p] \big| h \in \mathbb{Z}_p^*\}$ is a cyclic subgroup of $\mathbb{Z}_p^*$ of order q.
  - **Closure**: $h^r g^r = (hg)^r$
  - **Inverse** of h^r is $(h^{-1})^r \in \mathbb{G}_r$
  - **Size** $(h^r)^x = h^{[rx\ mod\ rq]} = (h^r)^x = h^{r[x\ mod\ q]} = (h^r)^{[x\ mod\ q]} mod\ p$

Remark: Two known attacks on Discrete Log Problem for $\mathbb{G}_r$(Section 9.2).

- First runs in time $O(\sqrt{q}) = O(2^{\lambda/2})$
- Second runs in time $2^{O\left(\sqrt[3]{n}(\log n)^{2/3}\right)}$

Can we find a cyclic group where DDH holds?

Remark: Two known attacks (Section 9.2).

- First runs in time $O(\sqrt{q}) = O(2^{\lambda/2})$
- Second runs in time $2^{O\left(\sqrt[3]{n}(\log n)^{2/3}\right)}$, where n is bit length of p

Goal: Set λ and n to balance attacks

$$\lambda = O\left(\sqrt[3]{n}(\log n)^{2/3}\right)$$

How to sample p=rq+1?

- First sample a random λ-bit prime q and
- Repeatedly check if rq+1 is prime for a random n- λ bit value r

Can we find a cyclic group where DDH holds?

Elliptic Curves Example: Let p be a prime (p > 3) and let A, B be constants. Consider the equation

$$y^2 = x^3 + Ax + B \ mod \ p$$

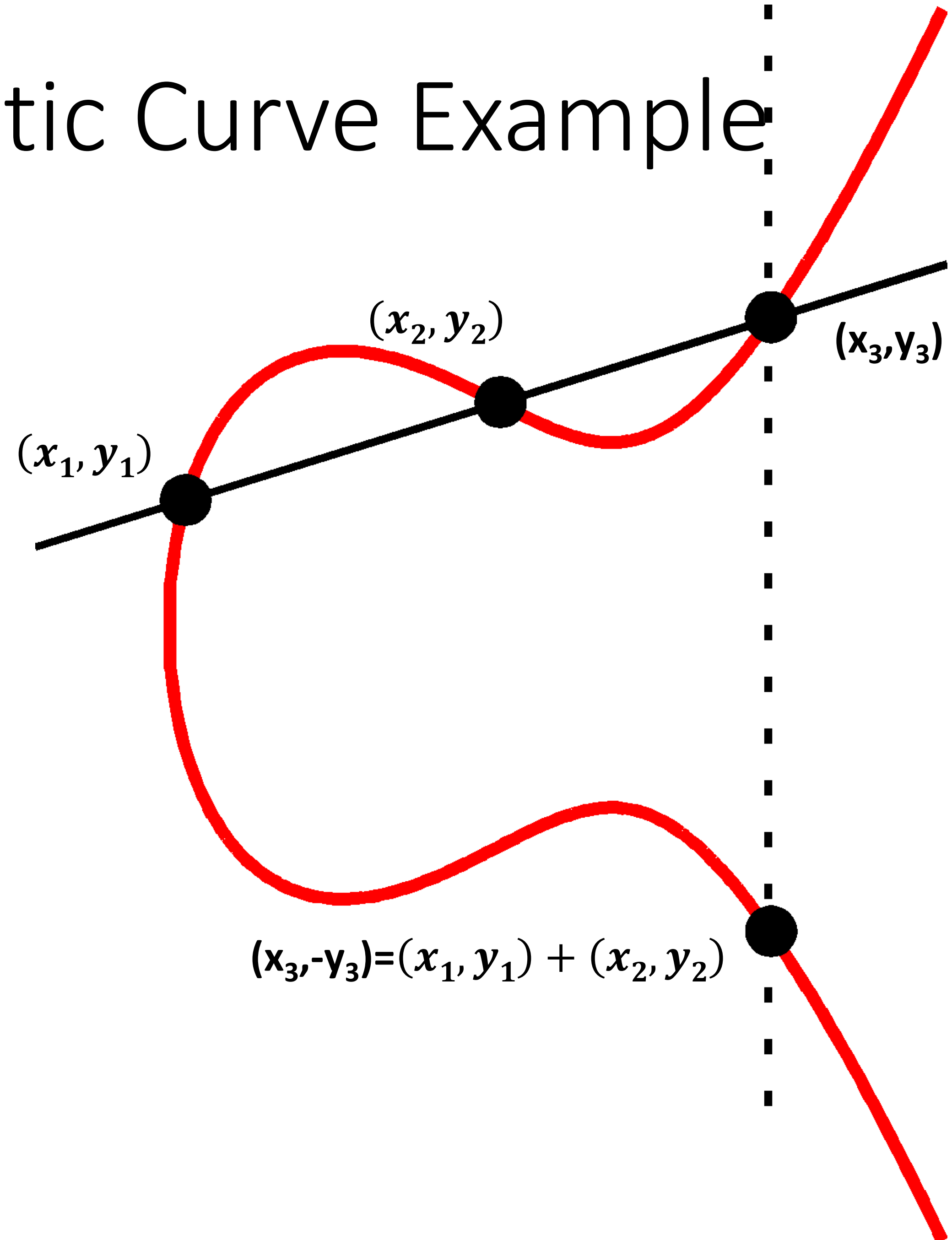
And let

$$E\left(\mathbb{Z}_p\right) = \left\{(x, y) \in \mathbb{Z}_p^2 \middle| y^2 = x^3 + Ax + B \ mod \ p \right\} \cup \{\mathcal{O}\}$$

Note: $\mathcal{O}$ is defined to be an additive identity $(x, y) + \mathcal{O} = (x, y)$

What is $(x_1, y_1) + (x_2, y_2)$?

Elliptic Curve Example



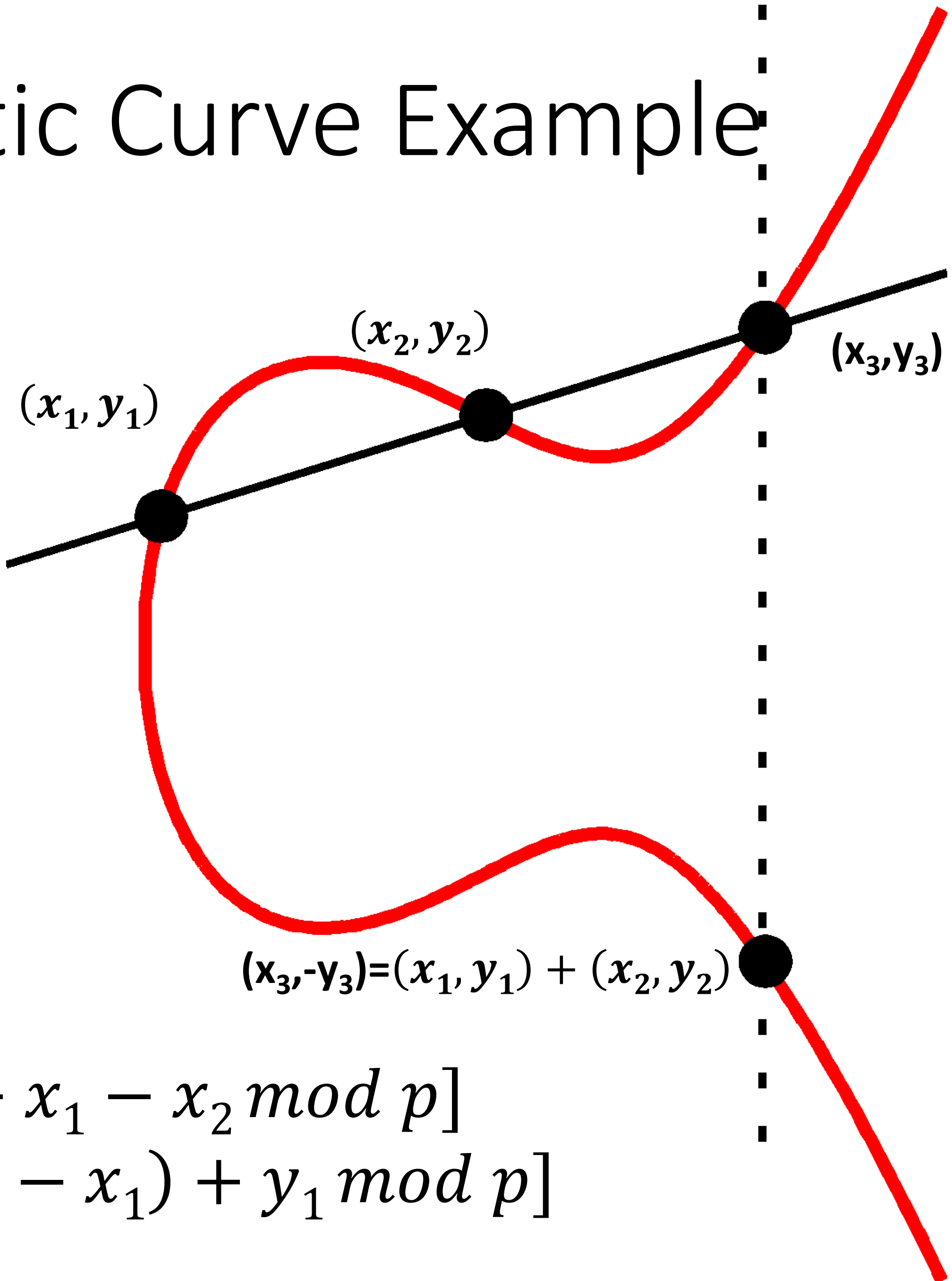
The line passing through $(\boldsymbol{x_1}, \boldsymbol{y_1})$ and $(\boldsymbol{x_2}, \boldsymbol{y_2})$ has the equation

$$y = m(x - x_1) + y_1 \, mod \, P$$

Where the slope

$$m = \left[\frac{y_1 - y_2}{x_1 - x_2} \, mod \, p\right]$$

Elliptic Curve Example



Formally, let

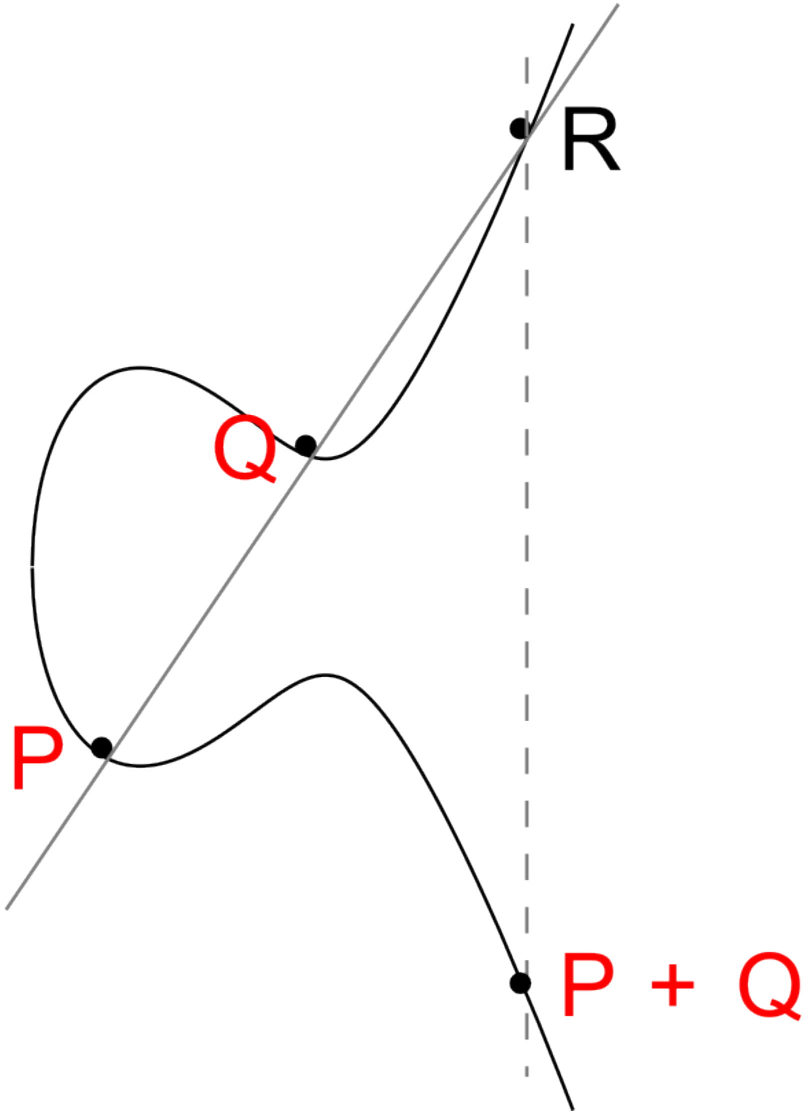
$$m = \left[\frac{y_1 - y_2}{x_1 - x_2}\ mod\ p\right]$$

Be the slope. Then the line passing through $(\boldsymbol{x_1}, \boldsymbol{y_1})$ and $(\boldsymbol{x_2}, \boldsymbol{y_2})$ has the equation

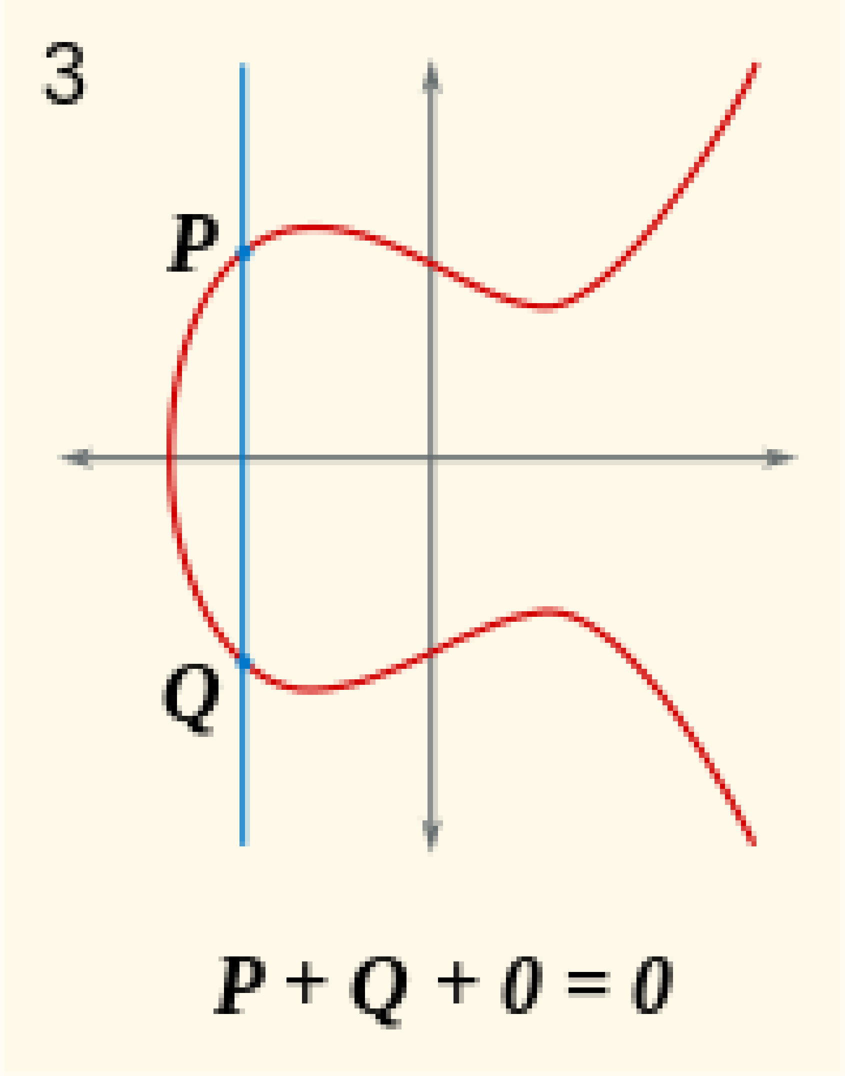
$$y = m(x - x_1) + y_1\ mod\ P$$

$$(m(x - x_1) + y_1)^2 = x^3 + Ax + B\ mod\ p$$

$$x_3 = [m^2 - x_1 - x_2\ mod\ p]$$
$$y_3 = [m(x_3 - x_1) + y_1\ mod\ p]$$


R
Q
P
P + Q

Elliptic Curve Example



No third point R on the elliptic curve.

P+Q = 0

Can we find a cyclic group where DDH holds?

Elliptic Curves Example: Let p be a prime (p > 3) and let A, B be constants. Consider the equation

$$y^2 = x^3 + Ax + B \ mod \ p$$

And let

$$E\left(\mathbb{Z}_p\right) = \left\{(x, y) \in \mathbb{Z}_p^2 \middle| y^2 = x^3 + Ax + B \ mod \ p\right\} \cup \{\mathcal{O}\}$$

Fact: $E\left(\mathbb{Z}_p\right)$ defines an abelian group

- For *appropriate curves* the DDH assumption is believed to hold
- If you make up your own curve there is a good chance it is broken…
- NIST has a list of recommendations