# Cryptography
# CS 555

**Week 9:**

- Number Theory + Public Key Crypto

**Readings:** Katz and Lindell Chapter 8, B.1, B.2

**Spring 2021**

# Limits of Symmetric Crypto

- **Key-Exchange Problem:**
  - Obi-Wan and Yoda want to communicate securely
  - Suppose that
    - Obi-Wan and Yoda don’t have time to meet privately and generate a shared secret key
    - ~~Use AES-GCM~~ (requires shared secret key!)
    - Trusted Intermediary: If Obi-Wan and Yoda both have secret keys with Anakin they can exchange a secret key via the trusted party.

# Limits of Symmetric Crypto

- Key-Exchange Problem:
  - Obi-Wan and Yoda want to communicate securely
  - Suppose that
    - Obi-Wan and Yoda don't have time to meet privately and generate a shared secret key
    - ~~Use AES-GCM~~ (requires shared secret key!)
    - **Trusted Intermediary:** If Obi-Wan and Yoda both have secret keys with Anakin ($K_{Y,A}$ and $K_{O,A}$) they can exchange a secret key via the trusted party.
      - Obi-Wan picks a key K, computes $c = \mathrm{Enc}_{K_{O,A}}(K)$ and sends $c$ to Anakin with instructions to re-encrypt and forward to Yoda.
      - Anakin computes $\mathrm{K} = \mathrm{Dec}_{K_{O,A}}(c)$ and $c' = \mathrm{Enc}_{K_{Y,A}}(K)$ and forwards to Yoda.
      - Yoda recovers $K = \mathrm{Dec}_{K_{Y,A}}(c')$
      - Anakin also learns the secret key
    - **Remark**: Obi-Wan and Yoda both trust Anakin, but would prefer to keep the key private just in case.

# Limits of Symmetric Crypto

- Key-Exchange Problem:
  - Obi-Wan and Yoda want to communicate securely
  - **Trusted Intermediary:** If Obi-Wan and Yoda both have secret keys with Anakin ($K_{Y,A}$ and $K_{O,A}$) they can exchange a secret key via the trusted party.
  - **Remark**: Obi-Wan and Yoda both trust Anakin, but would prefer to keep the key private just in case.

# Limits of Symmetric Crypto

- Key-Exchange Problem:
  - Obi-Wan and Yoda want to communicate securely
  - Suppose that
    - Obi-Wan and Yoda don’t have time to meet privately and generate one
    - Obi-Wan and Yoda share an asymmetric key with Anakin
    - Can they use Anakin to exchange a secret key?
    - **Remark**: Obi-Wan and Yoda both trust Anakin, but would prefer to keep the key private just in case.
- Need for Public-Key Crypto
  - We can solve the key-exchange problem using public-key cryptography.
  - No solution is known using symmetric key cryptography alone

# Symmetric Key Explosion Problem

- Suppose we have n people and each pair of people want to be able to maintain a secure communication channel.
  - How many private keys per person?
  - **Answer**: n-1

- Key Explosion Problem
  - n can get very big if you are Google or Amazon!

# Public Key Encryption: Basic Terminology

- Plaintext/Plaintext Space
  - A message $\mathrm{m} \in \mathcal{M}$
- Ciphertext $\mathrm{c} \in \mathcal{C}$
- **Public/Private Key Pair $(\boldsymbol{pk}, \boldsymbol{sk}) \in \boldsymbol{\mathcal{K}}$**

# Public Key Encryption Syntax

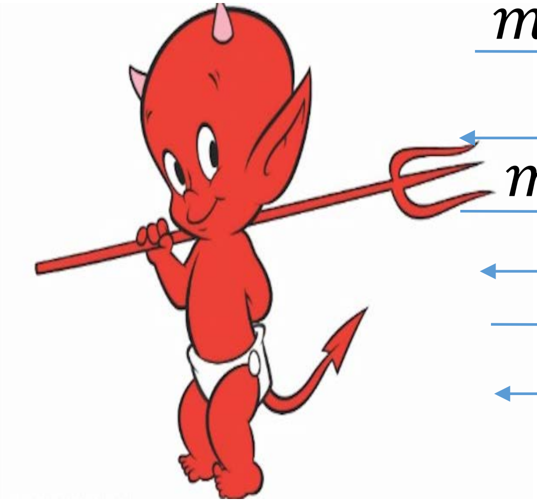
- Three Algorithms
  - $\mathrm{Gen}(1^n, R)$ (Key-generation algorithm)
    - Input: Random Bits R
    - Output: $(\boldsymbol{pk}, \boldsymbol{sk}) \in \mathcal{K}$
  - $\mathrm{Enc}_{\mathrm{pk}}(m) \in \mathcal{C}$ (Encryption algorithm)
  - $\mathrm{Dec}_{\mathrm{sk}}(c)$ (Decryption algorithm)
    - Input: Secret key sk and a ciphertex c
    - Output: a plaintext message $\mathrm{m} \in \mathcal{M}$

- **Invariant**: $\mathrm{Dec}_{sk}(\mathrm{Enc}_{pk}(m))=m$

Alice must run key generation algorithm in advance an publishes the public key: pk

Assumption: Adversary only gets to see pk (not sk)

# CPA-Security ($\mathrm{PubK}_{A,\Pi}^{\mathrm{LR-cpa}}(n)$)



**Public Key: pk**

$m_0^1, m_1^1$

$\boldsymbol{c_1 = \mathrm{Enc}_{pk}(m_b^1)}$

$m_0^2, m_1^2$

$\boldsymbol{c_2 = \mathrm{Enc}_{pk}(m_b^2)}$

$m_0^3, m_1^3$

$\boldsymbol{c_3 = \mathrm{Enc}_{pk}(m_b^3)}$

⋮

b’

**Random bit b**

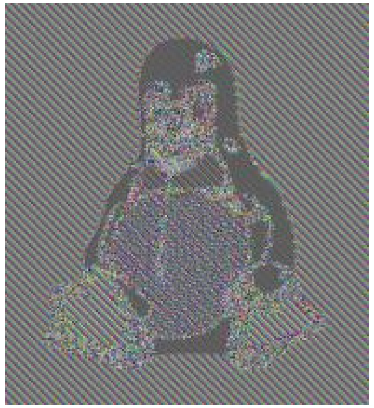
**(pk,sk) = Gen(.)**

$$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}$$

$$\Pr\left[\mathrm{PubK}_{A,\Pi}^{\mathrm{LR-cpa}}(n) = 1\right] \le \frac{1}{2} + \mu(n)$$

# Public Key Crypto

- **Fact 1:** CPA Security and Eavesdropping Security are Equivalent
  - **Key Insight:** The attacker has the public key so he doesn't gain anything from being able to query the encryption oracle!
- **Fact 2:** Any deterministic encryption scheme is not CPA-Secure
  - Historically overlooked in many real world public key crypto systems
- **Fact 3:** No Public Key Cryptosystem can achieve Perfect Secrecy!
  - Exercise 11.1
  - **Hint:** Unbounded attacker can keep encrypting the message m using the public key to recover all possible encryptions of m.



- **Key Question:** How do we achieve CPA/CCA-Secure Public Key Encryption?

# Number Theory

- Key tool behind (most) public key-crypto
  - RSA, El-Gamal, Diffie-Hellman Key Exchange

- Aside: don’t worry we will still use symmetric key crypto
  - It is more efficient in practice
  - First step in many public key-crypto protocols is to generate symmetric key
    - Then communicate using authenticated encryption e.g., AES-GCM

# Polynomial Time Factoring Algorithm?

**FindPrimeFactor**

**Input**: N

**For** i=1,…,N

  **if** N/i is an integer then

    **Output** i

Did we just break RSA?

**Running time:** O(N) steps

**Correctness**: Always returns a factor

# Polynomial Time Factoring Algorithm?

**FindPrimeFactor**

**Input**: N

**For** i=1,…,N

  **if** N/i is an integer then

    **Output** i

> We measure running time of an arithmetic algorithm (multiply, divide, GCD, remainder) in terms of the number of bits necessary to encode the inputs.
>
> How many bits $||N||$ to encode N?
> Answer: $||N|| = \log_2(N)$

**Running time:** O(N) steps

**Correctness**: Always returns a factor

# Polynomial Time Operations on Integers

Polynomial time in $||a||$and $||b||$

- Addition
- Multiplication
- Division with Remainder
  - **Input: a** and divisor **b**
  - **Output**: quotient q and remainder r < **b** such that

$$\boldsymbol{a} = q\boldsymbol{b} + r$$

  **Convenient Notation:** r = **a** mod **b**
  **Note 1:** We require that quotient q and remainder r are both integers
  **Note 2:** If remainder is $r = 0$ (i.e., $\boldsymbol{a} = q\boldsymbol{b} + 0$) we say that **b** divides **a** (Notation: b|a)
- Greatest Common Divisor
  - **Example:** gcd(9,15) = 3
- Extended GCD(**a**,**b**)
  - Output integers X,Y such that

$$X\boldsymbol{a} + Y\boldsymbol{b} = \text{gcd}(\boldsymbol{a}, \boldsymbol{b})$$

# Polynomial Time Operations on Integers

- Division with Remainder
  - **Input:** a and b
  - **Output**: quotient q and remainder r < b such that

$$\boldsymbol{a} = q\boldsymbol{b} + r$$

- Greatest Common Divisor
  - **Key Observation:** if $\boldsymbol{a} = q\boldsymbol{b} + r$

  Then gcd(**a,b**) = gcd(r, **b**)=gcd(**a** mod **b**, **b**)

  **Proof:**

  - Let d = gcd(**a**,**b**). Then d divides both a and b. Thus, d also divides r=a-qb.
    - →d=gcd(**a**,**b**) $\leq$ gcd(r, **b**)
  - Let d' = gcd(r, **b**). Then d' divides both b and r. Thus, d' also divides a = qb+r.
    - →gcd(**a**,**b**) $\geq$ gcd(r, **b**)=d'
  - Conclusion: d=d'.

# More Polynomial Time Operations on Integers

- **(Modular Arithmetic)** The following operations are polynomial time in $\|a\|$ and $\|b\|$ and $\|N\|$.

1. Compute [**a** mod **N**]
2. Compute sum [(**a**+**b**) mod **N**], difference [(**a**-**b**) mod **N**] or product [**ab** mod **N**]
3. Determine whether **a** has an inverse $\mathbf{a}^{-1}$ such that 1=[$\mathbf{aa}^{-1}$ mod **N**]
4. Find $\mathbf{a}^{-1}$ if it exists
5. Compute the exponentiation [$\mathbf{a}^{\mathbf{b}}$ mod **N**]

# More Polynomial Time Operations on Integers

- (Modular Arithmetic) The
  in |

1. Compute [**a** mod **N**]
2. Compute sum [(
   [**ab** mod **N**]
3. Determine whether **a** has an inverse $\mathbf{a}^{-1}$ such that $1=[\mathbf{aa}^{-1}$ mod **N**]
4. Find $\mathbf{a}^{-1}$ if it exists
5. Compute the exponentiation [$\mathbf{a}^{\mathbf{b}}$ mod **N**]

**Remark**: Part 3 and 4 use extended GCD algorithm

# More Polynomial Time Operations on Integers

- (Modular Arithmetic) The following operations are polynomial time in in $\|a\|$ and $\|b\|$ and $\|N\|$.

1. Compute [**a** mod **N**]
2. Compute sum [(**a**+**b**) mod **N**], difference [(**a**-**b**) mod **N**] or product [**ab** mod **N**]
3. Determine whether **a** has an inverse $\mathbf{a}^{-1}$ such that 1=[$\mathbf{aa}^{-1}$ mod **N**]
4. Find $\mathbf{a}^{-1}$ if it exists
   - **Note:** $\mathbf{a}^{-1}$ exists if and only if GCD(a,N) = 1.
   - **Extended Euclidean Algorithm:** Finds integers x,y s.t. ax+Ny =GCD(a,N)=1.
   - **Define:** $\mathbf{a}^{-1}$ **=[x** mod **N**] and observe [$\mathbf{aa}^{-1}$ mod **N**]=[ax-Ny mod **N**] = GCD(a,N)=1.
5. Compute the exponentiation [$\mathbf{a}^{\mathbf{b}}$ mod **N**]

# More Polynomial Time Operations on Integers

- (Modular Arithmetic) The following operations are polynomial time in in $||a||$ and $||b||$ and $||N||$.

1. Compute the exponentiation [$\mathbf{a}^{\mathbf{b}}$ mod **N**]

**Attempt 1:**

X =1

For i=1,…,b

X = X*a

**What is wrong?**

# More Polynomial Time Operations on Integers

(Modular Arithmetic) The following operations are polynomial time in $\|a\|$, $\|b\|$ and $\|N\|$.

1. Compute the exponentiation [$\mathbf{a}^{\mathbf{b}}$ mod **N**]

**Attempt 2:**

If (b=0) return 1

X[0]=a;

For i=1,…,$\log_2$(b)+1

X[i] = X[i-1]*X[i-1] // Invariant: X[i] = $\boldsymbol{a}^{2^i}$

[$\mathbf{a}^{\mathbf{b}}$ mod **N**]=$\boldsymbol{a}^{\sum_i \boldsymbol{b}[i]2^i} \bmod \mathbf{N}$

$$= \prod_i \mathrm{X[i]}^{\mathbf{b}[\mathrm{i}]} \bmod \mathbf{N}$$

**What is wrong?**

**The number of bits in $a^{2^{\|b\|+1}}$ is O($2^{\|b\|+1}$).**

# More Polynomial Time Operations on Integers

(Modular Arithmetic) The following operations are polynomial time in $\|a\|$, $\|b\|$ and $\|N\|$.

1. Compute the exponentiation [$\mathbf{a}^{\mathbf{b}}$ mod **N**]

**Fixed Algorithm:**

If (b=0) return 1

X[0]=a;

For i=1,…,$\log_2$(b)+1

X[i] = X[i-1]*X[i-1] mod **N** // Invariant: X[i] = $\boldsymbol{a}^{2^i}$ mod **N**

[$\mathbf{a}^{\mathbf{b}}$ mod **N**]=$\boldsymbol{a}^{\sum_i \boldsymbol{b}[i]2^i} \text{mod } \mathbf{N}$

$$= \prod_i \text{X[i]}^{\mathbf{b}[\mathrm{i}]} \text{ mod } \mathbf{N}$$

# More Polynomial Time Operations on Integers

**(Sampling)** Let

$$\mathbb{Z}_N = \{1, \ldots, N\}$$
$$\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \mid \gcd(N, x) = 1\}$$

Examples:

$$\mathbb{Z}_6^* = \{1,5\}$$

$$\mathbb{Z}_7^* = \{1,2,3,4,5,6\}$$

# More Polynomial Time Operations on Integers

**(Sampling)** Let

$$\mathbb{Z}_N = \{1, \ldots, N\}$$
$$\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \mid \gcd(N, x) = 1\}$$

- There is a probabilistic polynomial time algorithm (in |N|) to sample from $\mathbb{Z}_N^*$ and $\mathbb{Z}_N$
- Algorithm to sample from $\mathbb{Z}_N^*$ is allowed to output "fail" with negligible probability in $||N||$.
- Conditioned on not failing sample must be uniform.

# Useful Facts

**Fact:** $x, y \in \mathbb{Z}^*_{\mathrm{N}} \rightarrow [xy \bmod \mathrm{N}] \in \mathbb{Z}^*_{\mathrm{N}}$ where $\mathbb{Z}^*_N = \{x \in \mathbb{Z}_N \mid \gcd(N, x) = 1\}$

**Example 1**: $\mathbb{Z}^*_8 = \{1,3,5,7\}$

$$[3 \times 7 \bmod 8] = [21 \bmod 8] = [5 \bmod 8] \in \mathbb{Z}^*_{\mathrm{N}}$$

**Proof (by contradiction): Let d:=gcd(xy,N)**

Suppose d>1 then for some prime p and integer q we have d=pq.

Now p must divide N and xy (by definition) and hence p must divide either x or y.

(WLOG) say p divides x. In this case gcd(x,N)=p > 1, which means $x \notin \mathbb{Z}^*_{\mathrm{N}}$

# More Useful Facts

$$x, y \in \mathbb{Z}^*_{\mathrm{N}} \rightarrow [xy \bmod \mathrm{N}] \in \mathbb{Z}^*_{\mathrm{N}}$$

**Fact 1:** Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}^*_{\mathrm{N}}|$ then for any $x \in \mathbb{Z}^*_{\mathrm{N}}$ we have

$$\left[x^{\boldsymbol{\phi}(\boldsymbol{N})} \bmod \mathrm{N}\right] = 1$$

**Example:** $\mathbb{Z}^*_8 = \{1,3,5,7\}$, $\phi(8) = 4$

$$\left[3^{\mathbf{4}} \bmod 8\right] = [9 \times 9 \bmod 8] = 1$$
$$\left[5^{\mathbf{4}} \bmod 8\right] = [25 \times 25 \bmod 8] = 1$$
$$\left[7^{\mathbf{4}} \bmod 8\right] = [49 \times 49 \bmod 8] = 1$$

# More Useful Facts

$$x, y \in \mathbb{Z}_{\mathrm{N}}^{*} \rightarrow [xy \bmod \mathrm{N}] \ \in \mathbb{Z}_{\mathrm{N}}^{*}$$

**Fact 1:** Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_{\mathrm{N}}^{*}|$ then for any $x \in \mathbb{Z}_{\mathrm{N}}^{*}$ we have $\left[x^{\boldsymbol{\phi}(\boldsymbol{N})} \bmod \mathrm{N}\right] = 1$

**Fact 2:** Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_{\mathrm{N}}^{*}|$ and let $N = \prod_{i=1}^{m} p_i^{e_i}$, where each $p_i$ is a distinct prime number and $e_i > 0$ then

$$\boldsymbol{\phi}(\boldsymbol{N}) = \prod_{i=1}^{m} (p_i - 1) p_i^{e_i - 1} = \mathrm{N} \prod_{i=1}^{m} \left(1 - \frac{1}{p_i}\right)$$

# Recap

- Polynomial time algorithms (in bit lengths $||\boldsymbol{a}||$, $||\boldsymbol{b}||$ and $||\mathbf{N}||$) to do important stuff
  - GCD(**a**,**b**)
  - Find inverse $\mathbf{a}^{-1}$ of **a** such that 1=[$\mathbf{a}\mathbf{a}^{-1}$ mod **N**] (if it exists)
  - PowerMod: [$\mathbf{a}^{\mathbf{b}}$ mod **N**]
  - Draw uniform sample from $\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \mid \gcd(N, x) = 1\}$
    - Randomized PPT algorithm

# More Useful Facts

$$x, y \in \mathbb{Z}_N^* \rightarrow [xy \bmod \mathrm{N}] \in \mathbb{Z}_N^*$$

**Fact 1:** Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_N^*|$ then for any $x \in \mathbb{Z}_N^*$ we have

$$\left[x^{\boldsymbol{\phi}(\boldsymbol{N})} \bmod \mathrm{N}\right] = 1$$

**Example:** $\mathbb{Z}_8^* = \{1,3,5,7\}$, $\phi(8) = 4$

$$\left[3^{\mathbf{4}} \bmod 8\right] = [9 \times 9 \bmod 8] = 1$$
$$\left[5^{\mathbf{4}} \bmod 8\right] = [25 \times 25 \bmod 8] = 1$$
$$\left[7^{\mathbf{4}} \bmod 8\right] = [49 \times 49 \bmod 8] = 1$$

# More Useful Facts

$$x, y \in \mathbb{Z}_{\mathrm{N}}^{*} \rightarrow [xy \bmod \mathrm{N}] \ \in \mathbb{Z}_{\mathrm{N}}^{*}$$

**Fact 1:** Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_{\mathrm{N}}^{*}|$ then for any $x \in \mathbb{Z}_{\mathrm{N}}^{*}$ we have $\left[x^{\boldsymbol{\phi}(\boldsymbol{N})} \bmod \mathrm{N}\right] = 1$

**Fact 2:** Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_{\mathrm{N}}^{*}|$ and let $N = \prod_{i=1}^{m} p_i^{e_i}$, where each $p_i$ is a distinct prime number and $e_i > 0$ then

$$\boldsymbol{\phi}(\boldsymbol{N}) = \prod_{i=1}^{m} (p_i - 1) p_i^{e_i - 1} = \mathrm{N} \prod_{i=1}^{m} \left(1 - \frac{1}{p_i}\right)$$

# More Useful Facts

**Fact 2:** Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_N^*|$ and let $N = \prod_{i=1}^{m} p_i^{e_i}$, where each $p_i$ is a distinct prime number and $e_i > 0$ then

$$\boldsymbol{\phi}(\boldsymbol{N}) = \prod_{i=1}^{m} (p_i - 1) p_i^{e_i - 1} = N \prod_{i=1}^{m} \left(1 - \frac{1}{p_i}\right)$$

**Example 0**: Let p be a prime so that $\mathbb{Z}_p^* = \{1, \ldots, p-1\}$

$$\boldsymbol{\phi}(\boldsymbol{p}) = p\left(1 - \frac{1}{p}\right) = p - 1$$

# More Useful Facts

**Fact 2:** Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}^*_{\mathrm{N}}|$ and let $N = \prod_{i=1}^{m} p_i^{e_i}$, where each $p_i$ is a distinct prime number and $e_i > 0$ then

$$\boldsymbol{\phi}(\boldsymbol{N}) = \prod_{i=1}^{m} (p_i - 1) p_i^{e_i - 1} = N \prod_{i=1}^{m} \left(1 - \frac{1}{p_i}\right)$$

**Example 1**: N = 9 = $3^2$ (m=1, $e_1$=2)

$$\boldsymbol{\phi}(\boldsymbol{9}) = \prod_{i=1}^{1} (p_i - 1) p_i^{2-1} = 2 \times 3$$

# More Useful Facts

**Example 1**: N = 9 = $3^2$ (m=1, $e_1$=2)

$$\boldsymbol{\phi}(\mathbf{9}) = \prod_{i=1}^{1} (p_i - 1) p_i^{2-1} = 2 \times 3$$

**Double Check**: $\mathbb{Z}_9^* = \{1,2,4,5,7,8\}$

# More Useful Facts

**Fact 2:** Let $\boldsymbol{\phi(N)} = |\mathbb{Z}^*_N|$ and let $N = \prod_{i=1}^{m} p_i^{e_i}$, where each $p_i$ is a distinct prime number and $e_i > 0$ then

$$\boldsymbol{\phi(N)} = \prod_{i=1}^{m} (p_i - 1) p_i^{e_i - 1} = N \prod_{i=1}^{m} \left(1 - \frac{1}{p_i}\right)$$

**Example 2**: N = 15 = $5 \times 3$ (m=2, $e_1$=$e_2$=1)

$$\boldsymbol{\phi(15)} = \prod_{i=1}^{2} (p_i - 1) p_i^{1-1} = (5-1)(3-1) = 8$$

# More Useful Facts

**Example 2**: N = 15 = $5 \times 3$ (m=2, $e_1$=$e_2$=1)

$$\boldsymbol{\phi}(\mathbf{15}) = \prod_{i=1}^{2} (p_i - 1)p_i^{1-1} = (5-1)(3-1) = 8$$

**Double Check**: $\mathbb{Z}_{15}^* = \{1,2,4,7,8,11,13,14\}$

I count 8 elements in $\mathbb{Z}_{15}^*$

# More Useful Facts

**Fact 2:** Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_N^*|$ and let $N = \prod_{i=1}^{m} p_i^{e_i}$, where each $p_i$ is a distinct prime number and $e_i > 0$ then

$$\boldsymbol{\phi}(\boldsymbol{N}) = \prod_{i=1}^{m} (p_i - 1) p_i^{e_i - 1} = \mathrm{N} \prod_{i=1}^{m} \left(1 - \frac{1}{p_i}\right)$$

**Special Case**: N = pq (p and q are distinct primes)

$$\boldsymbol{\phi}(\boldsymbol{N}) = (p-1)(q-1)$$

# More Useful Facts

**Special Case**: N = pq    (p and q are distinct primes)

$$\boldsymbol{\phi}(\boldsymbol{N}) = (p-1)(q-1)$$

**Proof Sketch:** If $x \in \mathbb{Z}_N$ is not divisible by p or q then $x \in \mathbb{Z}_N^*$**.** How many elements are not in $\mathbb{Z}_N^*$ ?

- **Multiples of p:** p, 2p, 3p,…,pq   (q multiples of p)
- **Multiples of q:** q, 2q,…,pq   (p multiples of q)
- **Double Counting?** N=pq is in both lists. Any other duplicates?
- No! cq = dp → q divides d (since, gcd(p,q)=1) and consequently $\mathrm{d} \geq q$
  - Hence, $\mathrm{dp} \geq pq = N$

# More Useful Facts

**Special Case**: N = pq    (p and q are distinct primes)

$$\boldsymbol{\phi(N)} = (p-1)(q-1)$$

**Proof Sketch:** If $x \in \mathbb{Z}_N$ is not divisible by p or q then $x \in \mathbb{Z}_N^*$**.** How many elements are not in $\mathbb{Z}_N^*$?

- **Multiples of p:** p, 2p, 3p,…,pq   (q multiples of p)
- **Multiples of q:** q, 2q,…,pq   (p multiples of q)
- **Answer:** p+q-1 elements are not in $\mathbb{Z}_N^*$

$$\boldsymbol{\phi(N) = N - (p + q - 1)}$$
$$= \mathbf{pq - p - q + 1 = (p-1)(q-1)}$$

# 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 $g, h \in \mathbb{G}$ we have $g \circ h \in \mathbb{G}$
- **(Identity**:) There is an element $e \in \mathbb{G}$ such that for all $g \in \mathbb{G}$ we have
$$g \circ e = g = e \circ g$$
- **(Inverses**:) For each element $g \in \mathbb{G}$ we can find $h \in \mathbb{G}$ such that $g \circ h = e = h \circ g$. We say that h is the inverse of g.
- (**Associativity:** ) For all $g_1, g_2, g_3 \in \mathbb{G}$ we have
$$(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$$

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

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

# 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 $\mathrm{g} \circ \mathrm{h} = \mathrm{e} = \mathrm{h} \circ \mathrm{g}$. 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
$$(\mathrm{g}_1 \circ \mathrm{g}_2) \circ \mathrm{g}_3 = \mathrm{g}_1 \circ (\mathrm{g}_2 \circ \mathrm{g}_3)$$

**Fact:** The identity is unique + inverses must be unique

**Proof:** If $\mathrm{e}$ and $\mathrm{e}'$ are both identity then $\mathrm{e} = \mathrm{e} \circ \mathrm{e}' = \mathrm{e}'$

If $\mathrm{h}$ and $\mathrm{h}'$ are both inverses of g then $\mathrm{h} = \mathrm{h} \circ \mathrm{e} = \mathrm{h} \circ (\mathrm{g} \circ \mathrm{h}') = (\mathrm{g} \circ \mathrm{h}) \circ h' = h'$.

Associativity

# Abelian Groups (Examples)

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

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

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

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

# Abelian Groups (Examples)

- **Example 1:** $\mathbb{Z}_N$ when ∘ denotes addition modulo N
- Identity: 0, since 0 ∘ 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 ∘ denotes multiplication modulo N
- Identity: 1, since 1∘ x =[1(x) mod N] = [x mod N].
- Inverse of x? Run extended GCD to obtain integers a and b such that

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

Observe that: $x^{-1}$ = a, since [ax mod N] = [1-bN mod N] = 1

# Groups

**Lemma 8.13**: Let $\mathbb{G}$ be a group with a binary operation $\circ$ (over G) and let $\mathrm{a, b, c} \in \mathbb{G}$. If $\mathrm{a} \circ \mathrm{c} = \mathrm{b} \circ \mathrm{c}$ then $\mathrm{a} = \mathrm{b}$.

Proof Sketch: Apply the unique inverse to $c^{-1}$ both sides.

$\mathrm{a} \circ \mathrm{c} = \mathrm{b} \circ \mathrm{c}$ → $(\mathrm{a} \circ \mathrm{c}) \circ c^{-1} = (\mathrm{b} \circ \mathrm{c}) \circ c^{-1}$

→ $\mathrm{a} \circ (\mathrm{c} \circ c^{-1}) = \mathrm{b} \circ (\mathrm{c} \circ c^{-1})$

→ $\mathrm{a} \circ (e) = \mathrm{b} \circ (e)$

→ $\mathrm{a} = \mathrm{b}$

(**Remark**: it is not to difficult to show that a group has a *unique* identity and that inverses are *unique*).

# Groups

**Lemma 8.13**: Let $\mathbb{G}$ be a group with a binary operation $\circ$ (over G) and let $\mathrm{a}, \mathrm{b}, \mathrm{c} \in \mathbb{G}$. If $\mathrm{a} \circ \mathrm{c} = \mathrm{b} \circ \mathrm{c}$ then $\mathrm{a} = \mathrm{b}$.

Proof Sketch: Apply the unique inverse to $c^{-1}$ both sides.

$\mathrm{a} \circ \mathrm{c} = \mathrm{b} \circ \mathrm{c}$ → $(\mathrm{a} \circ \mathrm{c}) \circ c^{-1} = (\mathrm{b} \circ \mathrm{c}) \circ c^{-1}$

→ $\mathrm{a} \circ (\mathrm{c} \circ c^{-1}) = \mathrm{b} \circ (\mathrm{c} \circ c^{-1})$

→ $\mathrm{a} \circ (e) = \mathrm{b} \circ (e)$

→ $\mathrm{a} = \mathrm{b}$

(**Remark**: it is not to difficult to show that a group has a *unique* identity and that inverses are *unique*).

# Group Exponentiation

**Definition**: Let $\mathbb{G}$ be a group with a binary operation $\circ$ (over G) let m be a positive integer and let $\mathrm{g} \in \mathbb{G}$ be a group element then we define

$$g^m \coloneqq \underbrace{\mathrm{g} \circ \cdots \circ \mathrm{g}}_{\text{m times}}$$

**Theorem**: Let $\mathbb{G}$ be finite group with size $\mathrm{m} = |\mathbb{G}|$ and let $\mathrm{g} \in \mathbb{G}$ be a group element then $g^m$=1 (where 1 denotes the unique identity of $\mathbb{G}$).

# Group Exponentiation

**Theorem 8.14**: Let $\mathbb{G}$ be finite group with size $\mathrm{m} = |\mathbb{G}|$ and let $\mathrm{g} \in \mathbb{G}$ be a group element then $g^m$=1 (where 1 denotes the unique identity of $\mathbb{G}$).

**Proof**: (for abelian group) Let $\mathbb{G} = \{g_1, \ldots, g_m\}$ then we claim

$$g_1 \circ \cdots \circ g_m = (g \circ g_1) \circ \cdots \circ (g \circ g_m)$$

Why? If $(g_i \circ g) = \left(g_j \circ g\right)$ then $g_j = g_i$ (by Lemma 8.13)

# Group Exponentiation

**Theorem 8.14**: Let $\mathbb{G}$ be finite group with size $\mathrm{m} = |\mathbb{G}|$ and let $\mathrm{g} \in \mathbb{G}$ be a group element then $g^m$=1 (where 1 denotes the unique identity of $\mathbb{G}$).

**Proof**: (for abelian group) Let $\mathbb{G} = \{g_1, \ldots, g_m\}$ then we claim

$$g_1 \circ \cdots \circ g_m = (g \circ g_1) \circ \cdots \circ (g \circ g_m)$$

Because $\mathbb{G}$ is abelian we can re-arrange terms

$$1 \circ (g_1 \circ \cdots \circ g_m) = (g^m) \circ (g_1 \circ \cdots \circ g_m)$$

By Lemma 8.13 we have $1 = g^m$. QED

# Group Exponentiation

**Theorem 8.14**: Let $\mathbb{G}$ be finite group with size $\mathrm{m} = |\mathbb{G}|$ and let $\mathrm{g} \in \mathbb{G}$ be a group element then $g^m$=1 (where 1 denotes the unique identity of $\mathbb{G}$).

**Corollary 8.15:** Let $\mathbb{G}$ be finite group with size $\mathrm{m} = |\mathbb{G}| > 1$ and let $\mathrm{g} \in \mathbb{G}$ be a group element then for any integer x we have $g^x = g^{[x \bmod m]}$.

**Proof**: $g^x = g^{qm+[x \bmod m]} = 1 \times g^{[x \bmod m]}$, where q is unique integer such that x=qm+ $[x \bmod m]$

# Group Exponentiation

**Special Case:** $\mathbb{Z}_N^*$ is a group of size $\boldsymbol{\phi(N)}$ so we have now proved

**Corollary 8.22:** For any $g \in \mathbb{Z}_{\mathrm{N}}^*$ and integer x we have

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

# Chinese Remainder Theorem

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

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

then

- f is a bijective mapping (invertible)
- f and its inverse $f^{-1}: \mathbb{Z}_p \times \mathbb{Z}_q \to \mathbb{Z}_{N}$ can be computed efficiently
- $f(x+y) = f(x) + f(y) = ([x + y \bmod p], [x + y \bmod q])$
- 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 modulo N=pq.

**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]

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

# 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

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

# Miller-Rabin Primality Test

**Input**: Integer N and parameter $1^t$

**Output**: "prime" or "composite"

**If Even(N)** or **PerfectPower(N**) return "composite"

**Else** find $u$ (odd) and $r \geq 1$ s.t. $\mathrm{N} - 1 = 2^r u$

**for** j=1 to t:

    **if** $a^u \neq \pm 1 \bmod \mathrm{N}$ and $a^{2^i u} \neq -1 \bmod \mathrm{N}$ for all $1 \leq i \leq r - 1$

        **return** "composite"

**Return** "prime"

# Miller-Rabin Primality Test

**Lemma:** If p is prime and $x^2 = 1 \bmod \mathrm{p}$ then
$x = \pm 1 \bmod \mathrm{p}$

**Input**: Integer N and parameter $1^t$

**Output**: "prime" or "composite"

**If Even(N)** or **PerfectPower(N)** return "compo

**Else** find $u$ (odd) and $r \geq 1$ s.t. $\mathrm{N} - 1 =$

**for** j=1 to t:

**if** $a^u \neq \pm 1 \bmod \mathrm{N}$ and $a^{2^i u} \neq -1 \bmod \mathrm{N}$ for all $1 \leq i \leq r-1$

**return** "composite"

**Return** "prime"

# Miller-Rabin Primality Test

**Observe:**

$$\left(a^{2^{r-1}u}\right)^2 = a^{N-1} \bmod N$$
$$= 1 \quad \bmod N$$

**Input**: Integer N and parameter $1^t$

**Output**: "prime" or "composite"

**If Even(N)** or **PerfectPower(N)** return "composite"

**Else** find $u$ (odd) and $r \geq 1$ s.t. $N - 1 = 2^r u$

**for** j=1 to t:

    **if** $a^u \neq \pm 1 \bmod N$ and $a^{2^i u} \neq -1 \bmod N$ for all $1 \leq i \leq r-1$

        **return** "composite"

**Return** "prime"

$$\left(a^{2^i u}\right)^2 - 1$$
$$= \left(a^{2^{i-1}u} - 1\right)\left(a^{2^{i-1}u} + 1\right) + 1$$

# Miller-Rabin Primality Test

**Input**: Integer N and parameter $1^t$

**Output**: "prime" or "composite"

**If Even(N)** or **PerfectPower(N)** return "composite"

**Else** find $u$ (odd) and $r \geq 1$ s.t. $\mathrm{N} - 1 = 2^r u$

**for** j=1 to t:

  **if** $a^u \neq \pm 1 \bmod \mathrm{N}$ and $a^{2^i u} \neq -1 \bmod \mathrm{N}$ for all $1 \leq i \leq r-1$

    **return** "composite"

**Return** "prime"

**Observe:**

$$\left(a^{2^{r-1}u}\right)^2 = a^{N-1} \bmod \mathrm{N}$$
$$= 1 \quad \bmod \mathrm{N}$$

*If N is prime we won't return composite*

$$\left(a^{2^r u}\right) - 1 = \left(a^{2^{r-1}u} - 1\right)\left(a^{2^{r-1}u} + 1\right)$$
$$= \cdots = \left(a^{2^{r-2}u} - 1\right)\left(a^{2^{r-2}u} + 1\right)\left(a^{2^{r-1}u} + 1\right)$$

# Miller-Rabin Primality Test

**Input**: Integer N and parameter $1^t$

**Output**: "prime" or "composite"

**If Even(N)** or **PerfectPower(N)** return "cor

**Else** find $u$ (odd) and $r \geq 1$ s.t. $\mathrm{N}-1=2$

**for** j=1 to t:

  **if** $a^u \neq \pm 1 \bmod \mathrm{N}$ and $a^{2^i u} \neq -1 \bmod \mathrm{N}$ for all $1 \leq i \leq r-1$

    **return** "composite"

**Return** "prime"

**Observe:**

$$\left(a^{2^{r-1}u}\right)^2 = a^{N-1} \bmod \mathrm{N}$$
$$= 1 \quad \bmod \mathrm{N}$$

**One of the factors must be 0 (mod N)**

***If N is prime we won't return composite***

$$\mathbf{0} = \left(a^{2^r u}\right) - \mathbf{1} = \cdots = \left(a^u - \mathbf{1}\right) \prod_{i=0}^{r-1} \left(a^{2^i u} + \mathbf{1}\right)$$

# Back to RSA Key-Generation

**KeyGeneration**($1^n$)

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

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

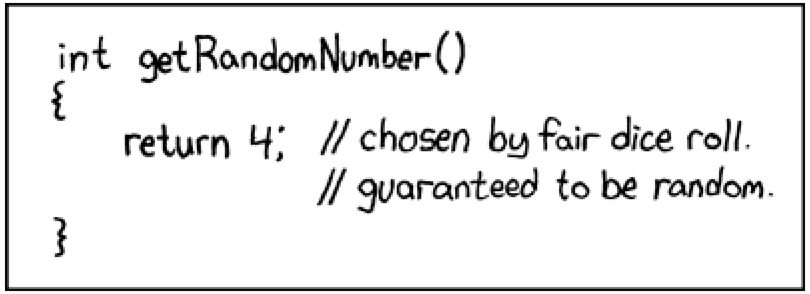
Step 3: Pick e > 1 such that gcd(e, $\phi(N)$)=1

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 $\mathrm{m} \in \mathbb{Z}_{\mathrm{N}}$

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

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

# (Plain) RSA Decryption

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

$$\textbf{Dec}_{\textbf{SK}}\textbf{(c)} = [c^d \bmod \mathrm{N}]$$

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

$$\textbf{Dec}_{\textbf{SK}}\textbf{(c)} = [(m^e)^d \bmod \mathrm{N}] = [m^{ed} \bmod \mathrm{N}]$$
$$= \left[m^{[ed \bmod \boldsymbol{\phi}(\boldsymbol{N})]} \bmod \mathrm{N}\right]$$
$$= [m^1 \bmod \mathrm{N}] = m$$

# RSA Decryption

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

$$\textbf{Dec}_{\textbf{SK}}\textbf{(c)} = [c^d \bmod \mathrm{N}]$$

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

$$\textbf{Dec}_{\textbf{SK}}\textbf{(c)} = m$$

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

$$\textbf{Dec}_{\textbf{SK}}\textbf{(c)} = m$$

  - Use Chinese Remainder Theorem to show this

$$ed = 1 + k(p-1)(q-1)$$

$$\rightarrow \mathrm{f}(c^d) = ([m^{ed} \bmod \mathrm{p}], [m^{ed} \bmod \mathrm{q}]) = ([m^1 \bmod \mathrm{p}], [m^1 \bmod \mathrm{q}])$$

$$\rightarrow f^{-1}(\mathrm{f}(c^d)) = f^{-1}([m^1 \bmod \mathrm{p}], [m^1 \bmod \mathrm{q}]) = m$$

# Plain RSA (Summary)

- Public Key (pk): N = pq, e such that $\text{GCD}\big(\text{e}, \phi(N)\big) = 1$
  - $\phi(N) = (p-1)(q-1)$ 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
  $[c^d \text{mod N}] = [m^{ed} \text{mod N}] = \left[m^{[ed\ mod\ \boldsymbol{\phi}(\boldsymbol{N})]} \text{mod N}\right] = [m \text{ mod N}]$

# 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 $FACTOR_{A,n}$

1. (N=pq,p,q) ← **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 $\text{FACTOR}_{A,n}$

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 ($\text{FACTOR}_{A,n} = 1$) if and only if $N=p'q'$.

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

# RSA-Assumption

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

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

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

# 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 $x^e = y \bmod N$

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

- 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

http://develop.wolframcloud.com/app/

**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->364696805906630283017006261328838672727187289052050 88...

……………………………………………………………………………………………

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 $\mathrm{m} = \mathrm{m}_1 \| ??$ then he can recover m given
$$\mathbf{Encrypt}(\mathrm{pk}, \mathrm{m}) = m^e \bmod \mathrm{N}$$

**Theorem[Coppersmith]:** If p(x) is a polynomial of degree e then in polynomial time (in log(N), $2^e$) 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 $\boldsymbol{c} = (m_1 \| m_2)^e \text{mod N}$, 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] (Informal):** 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=$q_1q_2$**
- 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 $f: \mathbb{Z}_N \rightarrow \mathbb{Z}_p \times \mathbb{Z}_q$ be defined as follows

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

then

- f is a bijective mapping (invertible)
- f and its inverse $f^{-1}: \mathbb{Z}_p \times \mathbb{Z}_q \rightarrow \mathbb{Z}_N$ 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 $[11^{53} \bmod 15]$

$f(11)=([-1 \bmod 3],[1 \bmod 5])$

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

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

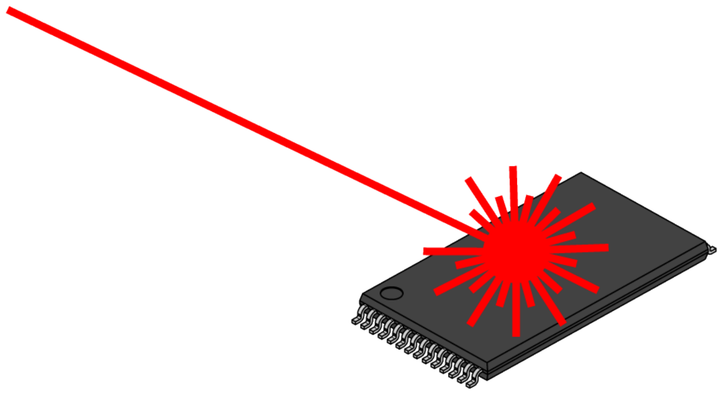
Thus, $11=[11^{53} \bmod 15]$

# 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 ∘ denotes addition modulo N
- Identity: 0, since 0 ∘ 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 ∘ denotes multiplication modulo N
- Identity: 1, since 1∘ x =[1(x) mod N] = [x mod N].
- Inverse of x? Run extended GCD to obtain integers a and b such that

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

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 $\phi(p-1)$

**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 $< g >= \mathbb{G}$.
2. Select $h \in \mathbb{G}$ 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

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

# 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 $\lambda$-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 $\lambda$-bit prime then the set of rth 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.
  - **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 $O\left(\sqrt{q}\right) = O\left(2^{\lambda/2}\right)$
- Second runs in time $2^{O\left(\sqrt[3]{n}(\log n)^{2/3}\right)}$, where n is bit length of p

**Goal**: Set $\lambda$ 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 $\lambda$-bit prime q and
- Repeatedly check if rq+1 is prime for a random n- $\lambda$ 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

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

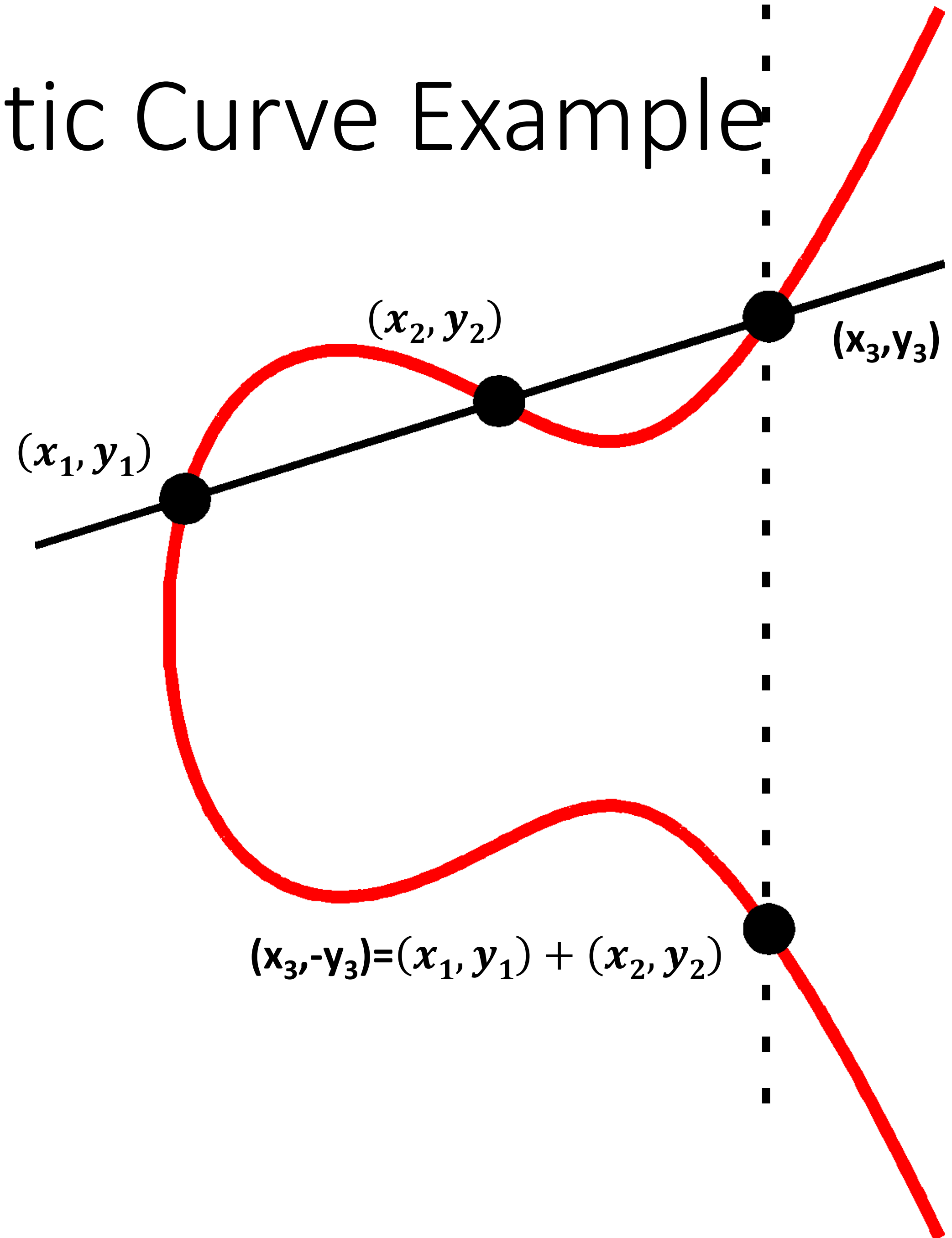
And let

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

**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



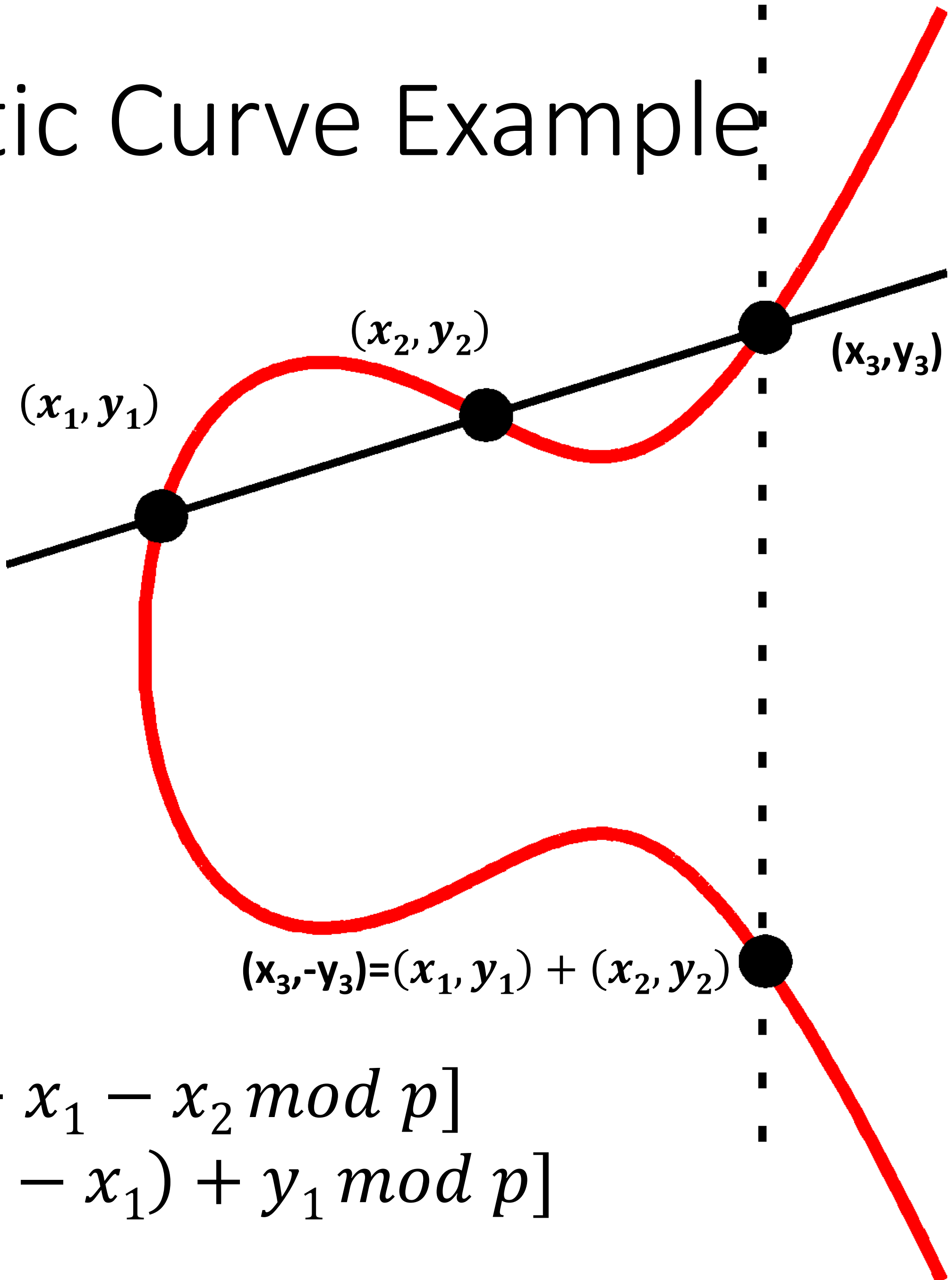

The line passing through $(\boldsymbol{x_1}, \boldsymbol{y_1})$ and $(\boldsymbol{x_2}, \boldsymbol{y_2})$ has the equation

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

Where the slope

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

# Elliptic Curve Example




Formally, let

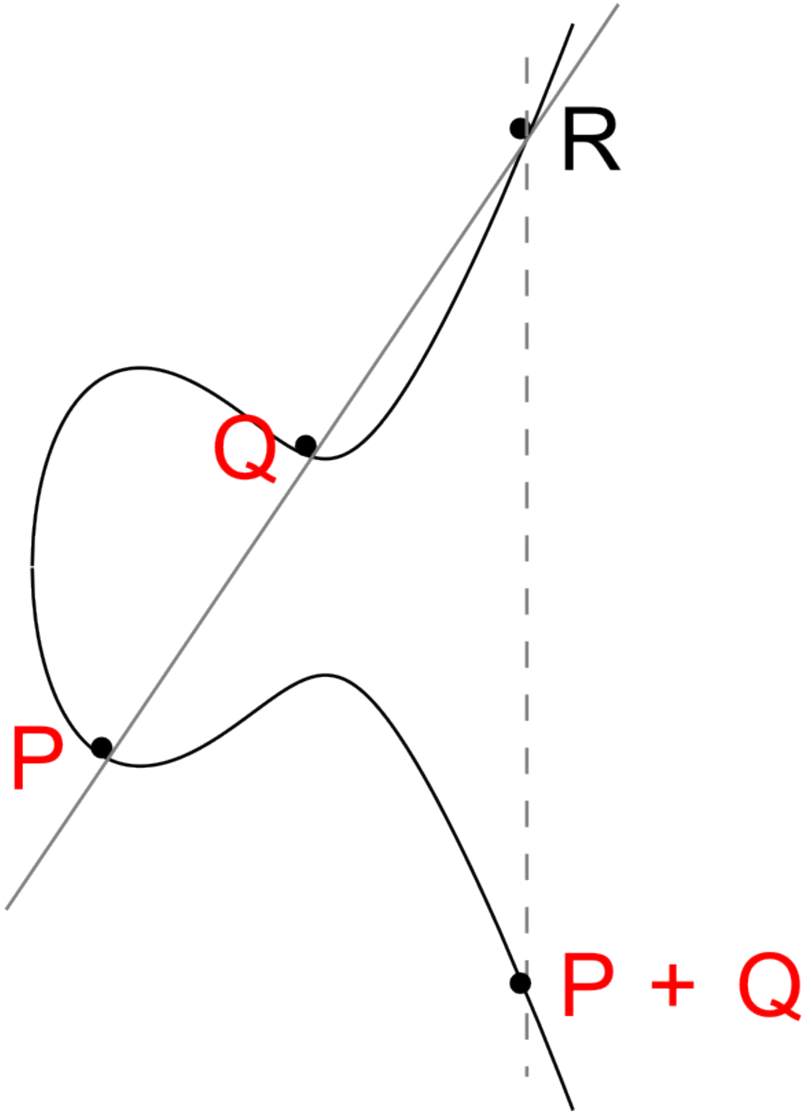
$$m = \left[\frac{y_1 - y_2}{x_1 - x_2} \; mod\; p\right]$$

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

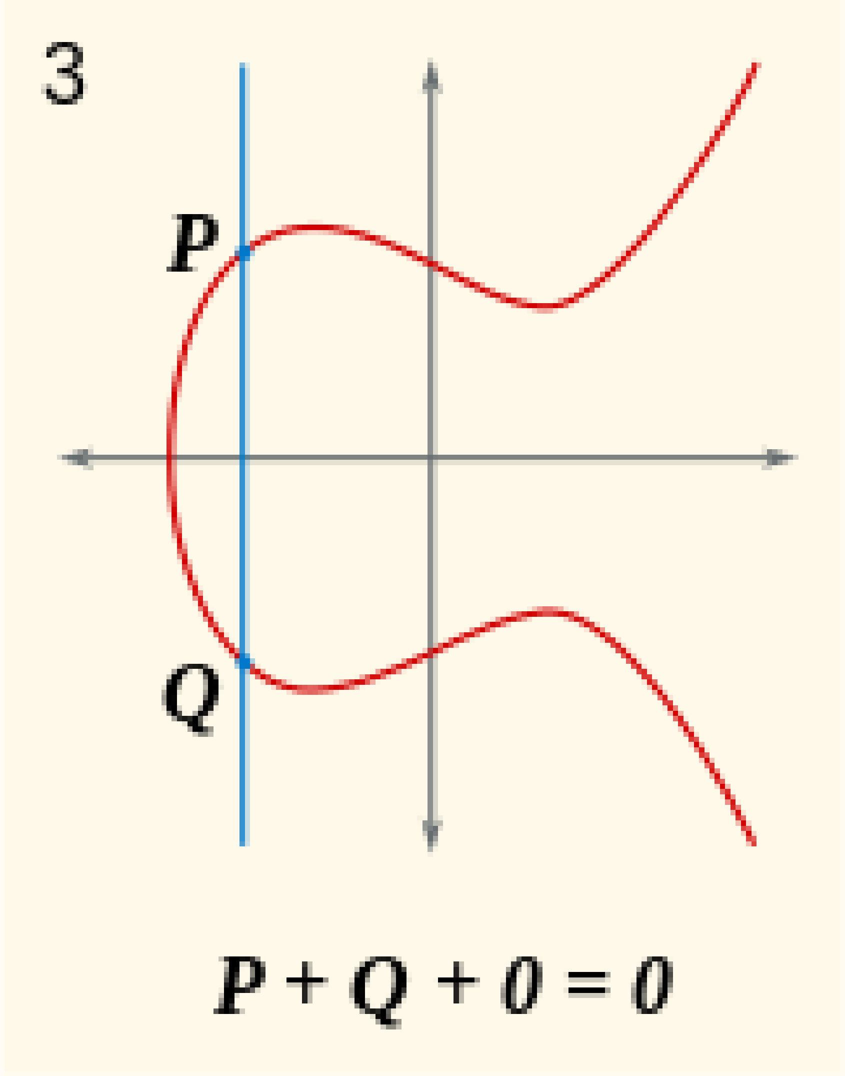
$$y = m(x - x_1) + y_1 \; mod\; P$$

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

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


R
Q
P
P + Q

# Elliptic Curve Example




No third point R on the elliptic curve.

P+Q = 0

(Inverse)

# 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