# Cryptography
# CS 555

Topic 26: Discrete LOG Applications

# Recap

- Plain RSA + Attacks
- Discrete Log Assumptions
  - CDH: Computational Diffie Hellman
  - DDH: Decisional Diffie Hellman
- Cyclic Groups under which CDH/DDH might hold

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)
2. $\mathbb{G} = \left\{ [hr \; mod \; p] \middle| h \in \mathbb{Z}_p^* \right\}$, where p=rq+1 (q is bit prime; p is n bit prime)
   - DDH and CDH are believed to be hard
   - Set $\lambda = O\left(\sqrt[3]{n}(\log n)^{2/3}\right)$ to maximize resistance against known attacks.
3. Elliptic Curves
   - DDH is believed to be hard for appropriate choice of curve

# Cyclic Group

- Let $\mathbb{G}$ be a group with order $\mathrm{m} = |\mathbb{G}|$ with a binary operation $\circ$ (over G) and let 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.

# 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

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).

# 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

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

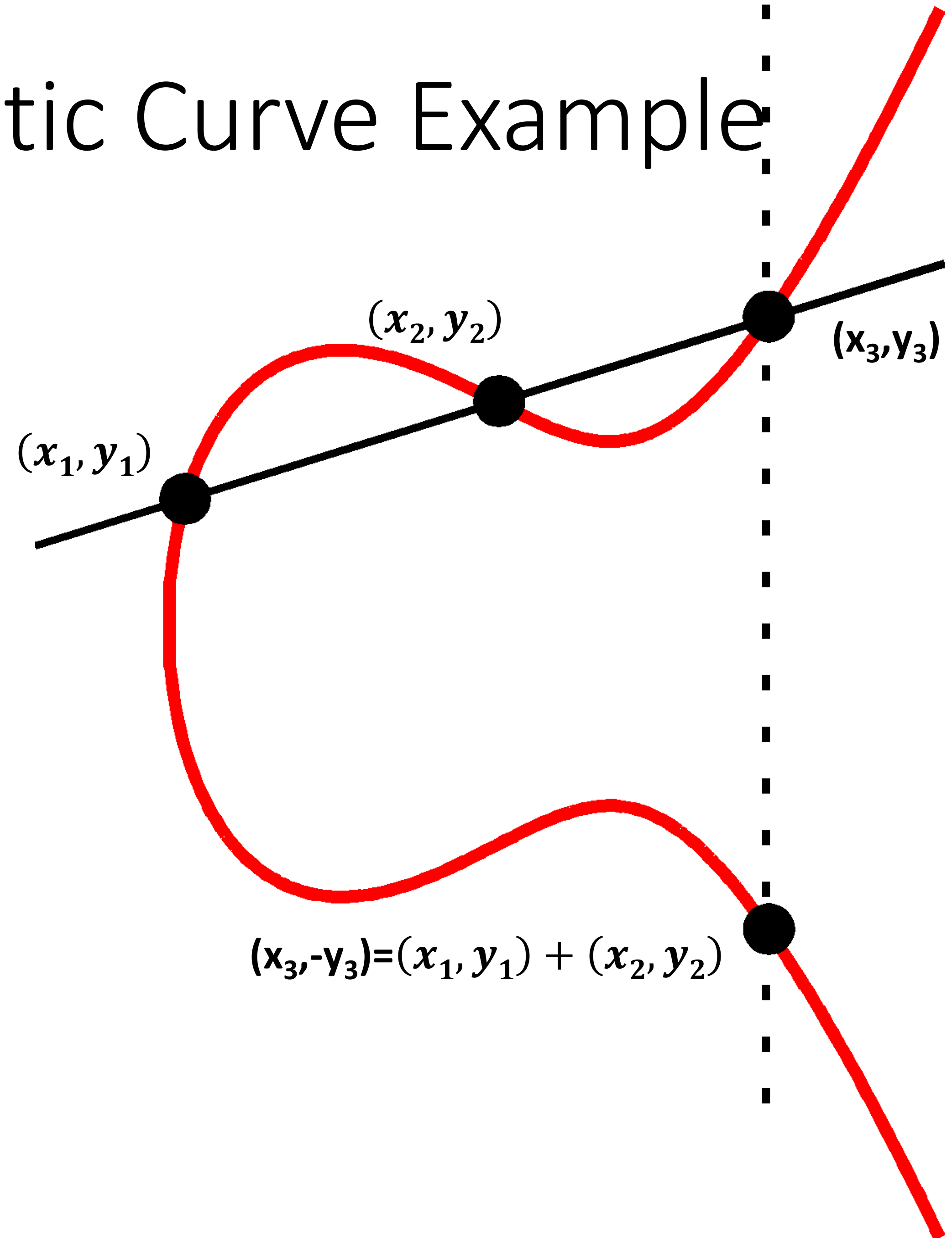
And let

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

**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




Formally, let

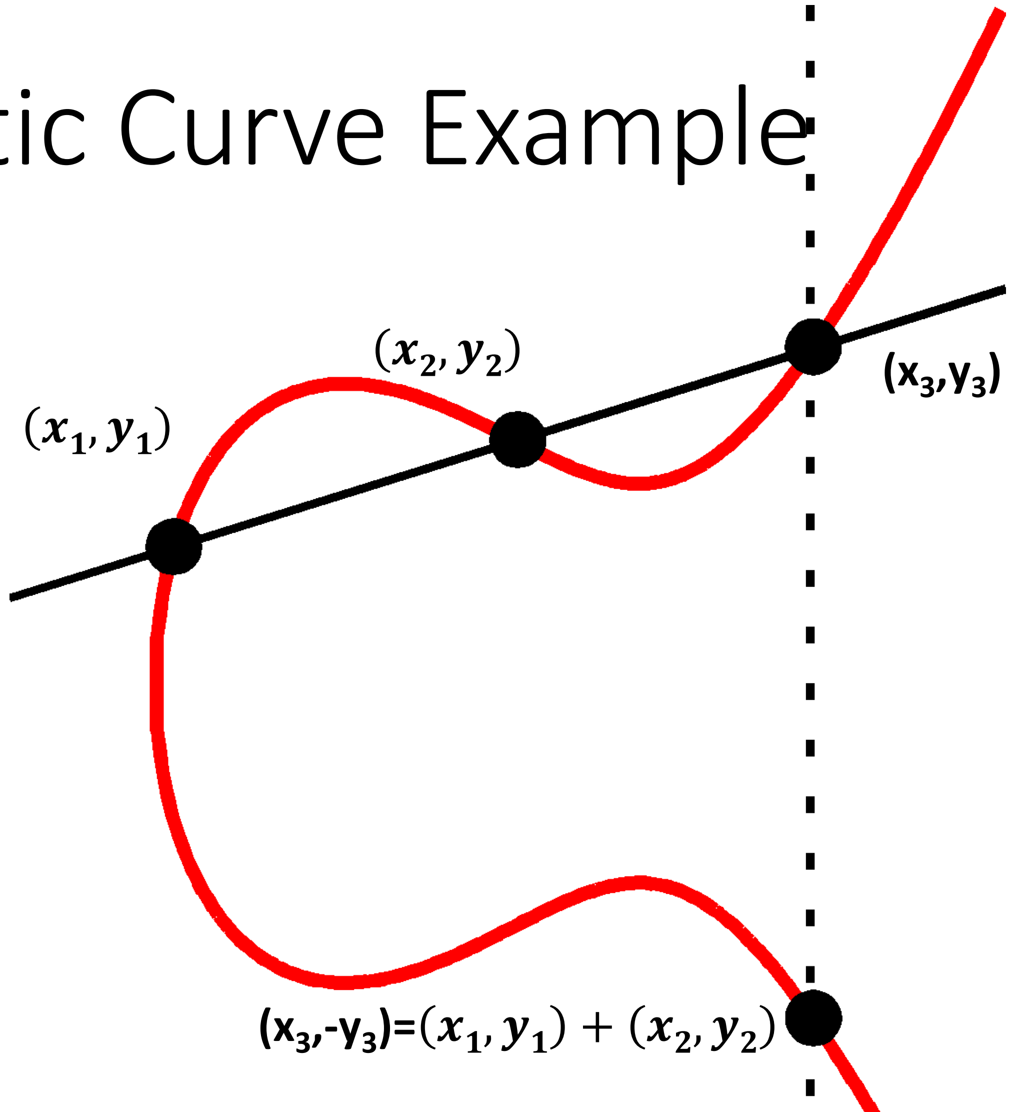
$$m = \left[\frac{y_1 - y_2}{x_1 - x_2} \ mod\ p\right]$$

be the slope.

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

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

# Elliptic Curve Example




Formally, let

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

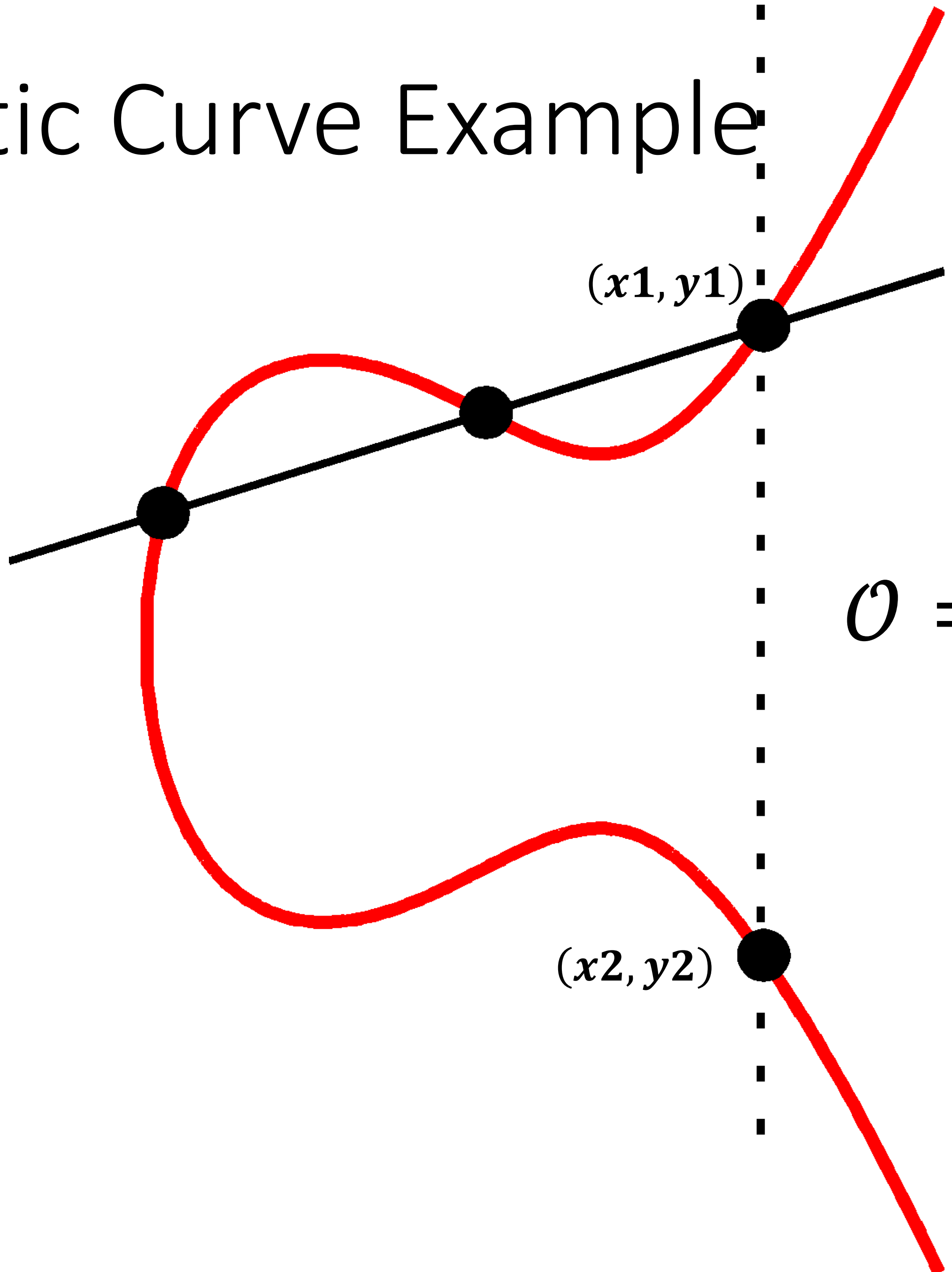
Be the slope. Then the line passing through $(\boldsymbol{x_1}, \boldsymbol{y_1})$ and $(\boldsymbol{x_2}, \boldsymbol{y_2})$ has the equation

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

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

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

# Elliptic Curve Example




$$\mathcal{O} = (x_1, y_1) + (x_2, y_2)$$

# 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

# RSA-Assumption vs Symmetric Key Crypto

- Recall: We can build (essentially) all of symmetric key crypto from one-way functions.
  - CCA-Secure Encryption, MACs, PRGs, PRFs
  - ~~Collision Resistant Hash Functions~~
- Symmetric Key Crypto → OWFs
  - Example: Can build OWFs from eavesdropping secure encryption scheme (weaker than CPA-secure/CCA-secure encryption)
- OWFs are necessary and sufficient for symmetric key crypto
  - Not known to be sufficient for public key crypto
  - Does the RSA-Assumption → OWFs?

# 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 $x^e = \mathrm{y} \bmod \mathrm{N}$

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

# Does the RSA-Assumption → OWFs?

- **Answer**: Yes! (and by extension RSA-Assumption is sufficient for any symmetric key cryptosystem).
- In fact the factoring assumption (weaker than RSA) is sufficient for OWFS.

Proof:

- Let $\mathrm{Gen}(1^n; r)$ output (N,p,q) where N=pq and p and q are random primes (selected with random bits r).
- $f_{Gen}(x) =$

1. (N,p,q) = $\mathrm{Gen}(1^n; x)$
2. Return N

Claim: $f_{Gen}(x)$ is a OWF.

# Does the RSA-Assumption → OWFs?

- $f_{Gen}(x) =$

1. (N,p,q) = $\mathrm{Gen}(1^n; x)$
2. Return N

**Claim**: $f_{Gen}(x)$ is a OWF.

Proof: Given a PPT attacker A that breaks OWF security we can run $A\left(f_{Gen}(x)\right)$ to obtain x' such that $f_{Gen}(x) = f_{Gen}(x')$ (A succeeds with non-negligible probability). Given x' we can run $\mathrm{Gen}(1^n; x')$ to obtain a tuple (N,p',q') such that N=p'q' and p'q' are prime. By uniqueness of prime factorization we have {p',q'} = {p,q}.

# Does the RSA-Assumption → OWFs?

- $f_{Gen}(x) =$

1. (N,p,q) = $\mathrm{Gen}(1^n; x)$
2. Return N

**Claim**: $f_{Gen}(x)$ is a OWF.

**Remark 1:** Also possible to construct One-Way-Permutation from RSA-Assumption

**Remark 2:** Possible to construct OWFs from Discrete-Log Assumption

# Does the RSA-Assumption → OWFs?

- $f_{Gen}(x) =$

1. (N,p,q) = $\mathrm{Gen}(1^n; x)$
2. Return N

**Claim**: $f_{Gen}(x)$ is a OWF.

**Remark 1:** Also possible to construct One-Way-Permutation from RSA-Assumption

**Remark 2:** Possible to construct OWFs from Discrete-Log Assumption

# Discrete Log Experiment $\text{DLog}_{A,G}(n)$

1. Run $\mathcal{G}(1^n)$ 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 ($\text{DLog}_{A,G}(n)=1$) if and only if $g^x=h$.

We say that the discrete log problem is hard relative to generator $\mathcal{G}$ if

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

# Collision Resistant Hash Functions

- Not known how to build CRHFs from OWFs
- Can build collision resistant hash functions from Discrete Logarithm Assumption
- Let $\mathcal{G}(1^n)$ output $(\mathbb{G}, q, g)$ where $\mathbb{G}$ is a cyclic group of order $q$ and g is a generator of the group.
- Suppose that discrete log problem is hard relative to generator $\mathcal{G}$.

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

# Collision Resistant Hash Functions

- Let $\mathcal{G}(1^n)$ output $(\mathbb{G}, q, g)$ where $\mathbb{G}$ is a cyclic group of order $q$ and g is a generator of the group.

Collision Resistant Hash Function (Gen,H):

- $Gen(1^n)$
  1. $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$
  2. Select random $\mathrm{h} \leftarrow \mathbb{G}$
  3. Output $\mathrm{s} = (\mathbb{G}, q, g, h)$
- $H^s(x_1, x_2) = g^{x_1}h^{x_2}$ (where, $x_1, x_2 \in \mathbb{Z}_q$ )

**Claim**: (Gen,H) is collision resistant

# Collision Resistant Hash Functions

- $H^s(x_1, x_2) = g^{x_1} h^{x_2}$ (where, $x_1, x_2 \in \mathbb{Z}_q$ )

**Claim**: (Gen,H) is collision resistant

Proof: Suppose we find a collision $H^s(x_1, x_2) = H^s(y_1, y_2)$ then we have $g^{x_1} h^{x_2} = g^{y_1} h^{y_2}$ which implies

$$h^{x_2 - y_2} = g^{y_1 - x_1}$$

Use extended GCD to find $(x_2 - y_2)^{-1}$ mod q then

$$h = h^{(x_2 - y_2)(x_2 - y_2)^{-1} \, mod \, q} = g^{(y_{1} - x_1)(x_2 - y_2)^{-1} \, mod \, q}$$

Which means that $(y_{1} - x_1)(x_2 - y_2)^{-1} \, mod \, q$ is the discrete log of h.

# Pollard’s p-1 Algorithm (Factoring)

- Let $N = pq$ where (p-1) has only “small” prime factors.
- Pollard’s p-1 algorithm can factor N.
  - **Remark 1**: This happens with very small probability if p is a random n bit prime.
  - **Remark 2**: One convenient/fast way to generate big primes it to multiply many small primes and add 1.
    - Example: $2 \times 3 \times 5 \times 7 + 1 = 211$ which is prime

**Claim**: Suppose we are given an integer B such that (p-1) divides B but (q-1) does not divide B then we can factor N.

# Pollard's p-1 Algorithm (Factoring)

**Claim**: Suppose we are given an integer B such that (p-1) divides B but (q-1) does not divide B then we can factor N.

**Proof**: B=c(p-1) for some integer c and let $y = [x^B - 1\, mod\ N]$. Applying the Chinese Remainder Theorem we have

$$y \leftrightarrow (x^B - 1 \bmod \mathrm{p}, x^B - 1 \bmod \mathrm{q})$$
$$= (0, x^B - 1 \bmod \mathrm{q})$$

This means that p divides y, but q does not divide y (unless $x^B = 1 \bmod \mathrm{q}$, which is very unlikely).

Thus, GCD(y,N) = p

# Pollard’s p-1 Algorithm (Factoring)

- Let $N = pq$ where (p-1) has only “small” prime factors.
- Pollard’s p-1 algorithm can factor N.

**Claim**: Suppose we are given an integer B such that (p-1) divides B but (q-1) does not divide B then we can factor N.

- **Goal**: Find B such that (p-1) divides B but (q-1) does not divide B.
- **Remark**: This is difficult if (p-1) has a large prime factor.

$$B = \prod_{i=1}^{k} p_i^{[n/\log p_i]}$$

# Pollard's p-1 Algorithm (Factoring)

- **Goal**: Find B such that (p-1) divides B but (q-1) does not divide B.
- **Remark**: This is difficult if (p-1) has a large prime factor.

$$B = \prod_{i=1}^{k} p_i^{[n/\log p_i]}$$

Here $p_1$=2,$p_2$=3,…

**Fact**: If (q-1) has prime factor larger than $p_k$ then (q-1) does not divide B.

**Fact**: If (p-1) does not have prime factor larger than $p_k$ then (p-1) does divide B.

# Pollard’s p-1 Algorithm (Factoring)

- **Option 1:** To defeat this attack we can choose strong primes p and q
  - A prime p is strong if (p-1) has a large prime factor
- **Drawback:** It takes more time to generate (provably) strong primes

- **Option 2:** A random prime is strong with high probability

- **Current Consensus:** Just pick a random prime

# Next Class: Factoring Algorithms

- Factoring Algorithms
- Read Katz and Lindell: Chapter 9
- Homework 3 Due