CS 207: Discrete Structures

Abstract algebra and Number theory
— Modular arithmetic and cryptography

Lecture 37
Nov 3 2016
Recap: Abstract algebra

<table>
<thead>
<tr>
<th>Topics covered till now: Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Definition of an abstract group; basic properties</td>
</tr>
<tr>
<td>▶ Examples:</td>
</tr>
<tr>
<td>▶ Invertible matrices</td>
</tr>
<tr>
<td>▶ Symmetries of a regular polygon</td>
</tr>
<tr>
<td>▶ Permutation groups</td>
</tr>
<tr>
<td>▶ Graph automorphisms</td>
</tr>
<tr>
<td>▶ ((\mathbb{Z}, +), (\mathbb{Z}_n, +n), (\mathbb{Z}_p, \times_p), \ldots)</td>
</tr>
<tr>
<td>▶ Abelian groups, Cyclic groups</td>
</tr>
<tr>
<td>▶ Group Isomorphisms and subgroups of a group.</td>
</tr>
<tr>
<td>▶ Order of a group and order of an element.</td>
</tr>
<tr>
<td>▶ Lagrange’s theorem; corollaries and some applications</td>
</tr>
</tbody>
</table>
Recap: Abstract algebra

Topics covered till now: Summary

- Definition of an abstract group; basic properties
- Examples:
  - Invertible matrices
  - Symmetries of a regular polygon
  - Permutation groups
  - Graph automorphisms
  - \((\mathbb{Z}, +), (\mathbb{Z}_n, +n), (\mathbb{Z}_p, \times_p), \ldots\)
- Abelian groups, Cyclic groups
- Group Isomorphisms and subgroups of a group.
- Order of a group and order of an element.
- Lagrange’s theorem; corollaries and some applications

Today: Applications to number theory and cryptography.
Modular or “clock” arithmetic

**Definition**

For integers \( a, b \) and positive integer \( m \), we say \( a \) is congruent to \( b \) modulo \( m \), denoted \( a \equiv b \mod m \), if \( m \mid (a - b) \).
Modular or “clock” arithmetic

Definition

For integers $a, b$ and positive integer $m$, we say $a$ is congruent to $b$ modulo $m$, denoted $a \equiv b \mod m$, if $m|(a - b)$.

i.e., $a$ & $b$ may not be same, but modulo $m$, they are the same:
Modular or “clock” arithmetic

**Definition**
For integers $a, b$ and positive integer $m$, we say $a$ is congruent to $b$ modulo $m$, denoted $a \equiv b \pmod{m}$, if $m|(a - b)$.

i.e., $a \& b$ may not be same, but modulo $m$, they are the same:

- Formally, $a \equiv b \pmod{m}$ iff $a \mod m = b \mod m$. 
Modular or “clock” arithmetic

Definition
For integers $a, b$ and positive integer $m$, we say $a$ is congruent to $b$ modulo $m$, denoted $a \equiv b \mod m$, if $m|(a - b)$.

i.e., $a \& b$ may not be same, but modulo $m$, they are the same:
- Formally, $a \equiv b \mod m$ iff $a \mod m = b \mod m$.
- What other properties does this “congruence” have?
Modular or “clock” arithmetic

**Definition**

For integers $a, b$ and positive integer $m$, we say $a$ is congruent to $b$ modulo $m$, denoted $a \equiv b \mod m$, if $m|(a - b)$.

i.e., $a$ & $b$ may not be the same, but modulo $m$, they are the same:

- Formally, $a \equiv b \mod m$ iff $a \mod m = b \mod m$.
- What other properties does this “congruence” have?
  - Equivalence?
  - If $a \equiv b \mod m$, $c \equiv d \mod m$, then $a + c \equiv b + d \mod m$ and $ac \equiv bd \mod m$.
  - $c \equiv ab \mod m$ iff $c \equiv (a(b \mod m)) \mod m$
Modular or “clock” arithmetic

Definition
For integers $a, b$ and positive integer $m$, we say $a$ is congruent to $b$ modulo $m$, denoted $a \equiv b \mod m$, if $m|(a - b)$.

i.e., $a$ & $b$ may not be same, but modulo $m$, they are the same:

▶ Formally, $a \equiv b \mod m$ iff $a \mod m = b \mod m$.

▶ What other properties does this “congruence” have?
  ▶ Equivalence?
  ▶ If $a \equiv b \mod m$, $c \equiv d \mod m$, then $a + c \equiv b + d \mod m$ and $ac \equiv bd \mod m$.
  ▶ $c \equiv ab \mod m$ iff $c \equiv (a(b \mod m)) \mod m$
    iff $c \equiv (a \mod m)(b \mod m) \mod m$.
    ▶ Corollary: Modular exponentiation is easy!
    ▶ What is $5^{15} \mod 23$?
Modular or “clock” arithmetic

**Definition**

For integers $a, b$ and positive integer $m$, we say $a$ is congruent to $b$ modulo $m$, denoted $a \equiv b \mod m$, if $m|(a - b)$.

i.e., $a \& b$ may not be same, but modulo $m$, they are the same:

- **Formally**, $a \equiv b \mod m$ iff $a \mod m = b \mod m$.
- **What other properties does this “congruence” have?**
  - **Equivalence?**
  - If $a \equiv b \mod m$, $c \equiv d \mod m$, then $a + c \equiv b + d \mod m$ and $ac \equiv bd \mod m$.
  - $c \equiv ab \mod m$ iff $c \equiv (a(b \mod m)) \mod m$ iff $c \equiv (a \mod m)(b \mod m) \mod m$.
  - **Corollary**: Modular exponentiation is easy!
  - **What is $5^{15} \mod 23$?**
  - $= (5 \mod 23)(5^2 \mod 23)(5^4 \mod 23)(5^8 \mod 23) \mod 23 = (5 \cdot 2 \cdot 4 \cdot 16) \mod 23 = 65 \mod 23 = 19.$
Modular or “clock” arithmetic

Definition
For integers \( a, b \) and positive integer \( m \), we say \( a \) is congruent to \( b \) modulo \( m \), denoted \( a \equiv b \mod m \), if \( m \mid (a - b) \).

i.e., \( a \) & \( b \) may not be same, but modulo \( m \), they are the same:

- Formally, \( a \equiv b \mod m \) iff \( a \mod m = b \mod m \).
- What other properties does this “congruence” have?
  - Equivalence?
  - If \( a \equiv b \mod m \), \( c \equiv d \mod m \), then \( a + c \equiv b + d \mod m \) and \( ac \equiv bd \mod m \).
  - \( c \equiv ab \mod m \) iff \( c \equiv (a(b \mod m)) \mod m \)

- Corollary: Modular exponentiation is easy!
- What is \( 5^{15} \mod 23 \)?
  - \( = (5 \mod 23)(5^2 \mod 23)(5^4 \mod 23)(5^8 \mod 23) \mod 23 = (5 \cdot 2 \cdot 4 \cdot 16) \mod 23 = 65 \mod 23 = 19. \)
- What is the worst case no. of steps?
Modular or “clock” arithmetic

Definition
For integers $a, b$ and positive integer $m$, we say $a$ is congruent to $b$ modulo $m$, denoted $a \equiv b \mod m$, if $m|(a - b)$.

i.e., $a$ & $b$ may not be same, but modulo $m$, they are the same:
- Formally, $a \equiv b \mod m$ iff $a \mod m = b \mod m$.
- What other properties does this “congruence” have?
  - Equivalence?
  - If $a \equiv b \mod m$, $c \equiv d \mod m$, then $a + c \equiv b + d \mod m$ and $ac \equiv bd \mod m$.
  - $c \equiv ab \mod m$ iff $c \equiv (a(b \mod m)) \mod m$

A math application: Fermat’s little theorem
- For any prime $p$, if $gcd(a, p) = 1$, then $p|(a^{p-1} - 1)$. 
Modular or “clock” arithmetic

**Definition**
For integers $a, b$ and positive integer $m$, we say $a$ is congruent to $b$ modulo $m$, denoted $a \equiv b \pmod{m}$, if $m|(a - b)$.

i.e., $a$ & $b$ may not be same, but modulo $m$, they are the same:

- Formally, $a \equiv b \pmod{m}$ iff $a \mod m = b \mod m$.
- What other properties does this “congruence” have?
  - Equivalence?
  - If $a \equiv b \pmod{m}$, $c \equiv d \pmod{m}$, then $a + c \equiv b + d \pmod{m}$ and $ac \equiv bd \pmod{m}$.
  - $c \equiv ab \pmod{m}$ iff $c \equiv (a(b \pmod{m})) \pmod{m}$

**A math application: Fermat’s little theorem**

- For any prime $p$, if $\gcd(a, p) = 1$, then $p|(a^{p-1} - 1)$.
- For any prime $p$, if $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$. 
Modular or “clock” arithmetic

Definition

For integers $a, b$ and positive integer $m$, we say $a$ is congruent to $b$ modulo $m$, denoted $a \equiv b \mod m$, if $m|(a - b)$.

i.e., $a \& b$ may not be same, but modulo $m$, they are the same:

- Formally, $a \equiv b \mod m$ iff $a \mod m \equiv b \mod m$.
- What other properties does this “congruence” have?
  - Equivalence?
  - If $a \equiv b \mod m$, $c \equiv d \mod m$, then $a + c \equiv b + d \mod m$
    and $ac \equiv bd \mod m$.
  - $c \equiv ab \mod m$ iff $c \equiv (a(b \mod m)) \mod m$

A math application: Fermat’s little theorem

- For any prime $p$, if $gcd(a, p) = 1$, then $p|(a^{p-1} - 1)$.
- For any prime $p$, if $gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \mod p$.

Modular arithmetic has vast applications including in hashing, generation of pseudorandom numbers, cryptography...
Sharing secrets in plain sight!

- Suppose two of you want to share a secret...
- But you can only shout messages... can you still get something private?
- which others will not be able to figure out at once?
A game of spies...

Start by choosing a prime 13, a generator for 
\((\mathbb{Z}_{13} \setminus \{0\}, \times_{13}) = 6\).
A game of spies...

Start by choosing a prime 13, a generator for 
$(\mathbb{Z}_{13}\setminus\{0\}, \times_{13})=6$.

- $A$ and $B$ pick two secret numbers from 1 to 13, say $a, b$. 

$A$ computes $6^a \mod 13 = M$.

$B$ computes $6^b \mod 13 = N$.

Shout/send $M$ and $N$ over.

$A$ computes $N^a \mod 13 = s$.

$B$ computes $M^b \mod 13 = t$.

$s = t$!

Why does this work?

Because $M^b \mod 13 = 6^a b \mod 13 = N^a \mod 13$.

And computing this from just 6, 13, 6 $a \mod 13$ and 6 $b \mod 13$ is hard without knowing $a$ and $b$. 
A game of spies...

Start by choosing a prime 13, a generator for \((\mathbb{Z}_{13} \setminus \{0\}, \times_{13})=6\).

- \(A\) and \(B\) pick two secret numbers from 1 to 13, say \(a, b\).
- \(A\) computes \(6^a \ mod \ 13 = M\).
A game of spies...

Start by choosing a prime 13, a generator for $(\mathbb{Z}_{13} \setminus \{0\}, \times_{13}) = 6$.

- $A$ and $B$ pick two secret numbers from 1 to 13, say $a$, $b$.
- $A$ computes $6^a \mod 13 = M$.
- $B$ computes $6^b \mod 13 = N$.

Why does this work?
- Because $M^b \mod 13 = 6^a b \mod 13 = N^a \mod 13$.
- And computing this from just $6$, $13$, $6^a \mod 13$ and $6^b \mod 13$ is hard without knowing $a$ and $b$. 
A game of spies...

Start by choosing a prime 13, a generator for $\left(\mathbb{Z}_{13} \setminus \{0\}, \times_{13}\right) = 6$.

- $A$ and $B$ pick two secret numbers from 1 to 13, say $a$, $b$.
- $A$ computes $6^a \mod 13 = M$.
- $B$ computes $6^b \mod 13 = N$.
- Shout/send $M$ and $N$ over.

Why does this work?
- Because $M^b \mod 13 = 6^{ab} \mod 13 = N^a \mod 13$.
- And computing this from just 6, 13, $6^a \mod 13$ and $6^b \mod 13$ is hard without knowing $a$ and $b$. 


A game of spies...

Start by choosing a prime 13, a generator for \((\mathbb{Z}_{13} \setminus \{0\}, \times_{13})\)=6.

- A and B pick two secret numbers from 1 to 13, say \(a, b\).
- A computes \(6^a \mod 13 = M\).
- B computes \(6^b \mod 13 = N\).
- Shout/send \(M\) and \(N\) over.
- A computes \(N^a \mod 13 = s\).

Why does this work?

Because \(M^b \mod 13 = 6^a \cdot 6^b \mod 13 = N^a \mod 13\).

And computing this from just 6, 13, 6 \(a\) mod 13 and 6 \(b\) mod 13 is hard without knowing \(a\) and \(b\).
A game of spies...

Start by choosing a prime 13, a generator for $(\mathbb{Z}_{13} \setminus \{0\}, \times_{13}) = 6$.

- A and B pick two secret numbers from 1 to 13, say $a$, $b$.
- A computes $6^a \mod 13 = M$.
- B computes $6^b \mod 13 = N$.
- Shout/send $M$ and $N$ over.
- A computes $N^a \mod 13 = s$.
- B computes $M^b \mod 13 = t$.
A game of spies...

Start by choosing a prime 13, a generator for \((\mathbb{Z}_{13} \setminus \{0\}, \times_{13})=6\).

- A and B pick two secret numbers from 1 to 13, say \(a, b\).
- A computes \(6^a \mod 13 = M\).
- B computes \(6^b \mod 13 = N\).
- Shout/send \(M\) and \(N\) over.
- A computes \(N^a \mod 13 = s\).
- B computes \(M^b \mod 13 = t\).
- \(s = t\)!

Why does this work?

Because \(M^b \mod 13 = 6^{ab} \mod 13 = N^a \mod 13\).

And computing this from just 6, 13, 6 \(a \mod 13\) and 6 \(b \mod 13\) is hard without knowing \(a\) and \(b\).
A game of spies...

Start by choosing a prime 13, a generator for $(\mathbb{Z}_{13} \setminus \{0\}, \times_{13}) = 6$.

- $A$ and $B$ pick two secret numbers from 1 to 13, say $a$, $b$.
- $A$ computes $6^a \mod 13 = M$.
- $B$ computes $6^b \mod 13 = N$.
- Shout/send $M$ and $N$ over.
- $A$ computes $N^a \mod 13 = s$.
- $B$ computes $M^b \mod 13 = t$.
- $s = t!$

Why does this work?

- Because $M^b \mod 13 = 6^{ab} \mod 13 = N^a \mod 13$. 
A game of spies...

Start by choosing a prime 13, a generator for $(\mathbb{Z}_{13} \setminus \{0\}, \times_{13})=6$.

- $A$ and $B$ pick two secret numbers from 1 to 13, say $a$, $b$.
- $A$ computes $6^a \mod 13 = M$.
- $B$ computes $6^b \mod 13 = N$.
- Shout/send $M$ and $N$ over.
- $A$ computes $N^a \mod 13 = s$.
- $B$ computes $M^b \mod 13 = t$.
- $s = t!$

Why does this work?

- Because $M^b \mod 13 = 6^{ab} \mod 13 = N^a \mod 13$.
- And computing this from just 6, 13, $6^a \mod 13$ and $6^b \mod 13$ is hard without knowing $a$ and $b$. 
A game of spies...

1. Choose a prime $p$ and a generator $g$ from $(\mathbb{Z}_p \setminus \{0\}, \times_p)$.
2. *Alice* fixes a private key $\alpha$ and *Bob* fixes $\beta$.
3. *Alice* computes $M = g^\alpha \mod p$ and shouts/sends it.
4. *Bob* computes $N = g^\beta \mod p$ and sends/shouts it.
5. *Alice* computes $M^\alpha \mod p$ and *Bob* computes $N^\beta \mod p$. 

Shared Key: $g^{\alpha \beta} \mod p$

▶ Others know $p, g, g^{\alpha} \mod p, g^{\beta} \mod p$.

▶ But computing $g^{\alpha \beta} \mod p$ from ONLY this info, without knowing $a$ and $b$ is hard!!

▶ How hard? Does there exist a poly-time (in size of digits) algorithm?

▶ This is called the Diffie-Hellman problem. Still open...

▶ In practice, choose large primes with $\sim 300$ digits.
A game of spies...

1. Choose a prime $p$ and a generator $g$ from $(\mathbb{Z}_p \setminus \{0\}, \times_p)$.
2. Alice fixes a private key $\alpha$ and Bob fixes $\beta$.
3. Alice computes $M = g^\alpha \mod p$ and shouts/sends it.
4. Bob computes $N = g^\beta \mod p$ and sends/shouts it.
5. Alice computes $M^\alpha \mod p$ and Bob computes $N^\beta \mod p$.

Shared Key: $g^{\alpha \beta} \mod p$
A game of spies...

1. Choose a prime $p$ and a generator $g$ from $(\mathbb{Z}_p \setminus \{0\}, \times_p)$.
2. *Alice* fixes a private key $\alpha$ and *Bob* fixes $\beta$.
3. *Alice* computes $M = g^\alpha \mod p$ and shouts/sends it.
4. *Bob* computes $N = g^\beta \mod p$ and sends/shouts it.
5. *Alice* computes $M^\alpha \mod p$ and *Bob* computes $N^\beta \mod p$.

**Shared Key:** $g^{\alpha\beta} \mod p$

- Others know $p, g, g^\alpha \mod p, g^\beta \mod p$. 
A game of spies...

1. Choose a prime $p$ and a generator $g$ from $(\mathbb{Z}_p \setminus \{0\}, \times_p)$.
2. Alice fixes a private key $\alpha$ and Bob fixes $\beta$.
3. Alice computes $M = g^\alpha \mod p$ and shouts/sends it.
4. Bob computes $N = g^\beta \mod p$ and sends/shouts it.
5. Alice computes $M^\alpha \mod p$ and Bob computes $N^\beta \mod p$.

Shared Key: $g^{\alpha \beta} \mod p$

- Others know $p, g, g^\alpha \mod p, g^\beta \mod p$.
- But computing $g^{\alpha \beta} \mod p$ from ONLY this info, without knowing $a$ and $b$ is hard!!
A game of spies...

1. Choose a prime \( p \) and a generator \( g \) from \( (\mathbb{Z}_p \setminus \{0\}, \times_p) \).
2. \textit{Alice} fixes a private key \( \alpha \) and \textit{Bob} fixes \( \beta \).
3. \textit{Alice} computes \( M = g^\alpha \mod p \) and shouts/sends it.
4. \textit{Bob} computes \( N = g^\beta \mod p \) and sends/shouts it.
5. \textit{Alice} computes \( M^\alpha \mod p \) and \textit{Bob} computes \( N^\beta \mod p \).

Shared Key: \( g^{\alpha\beta} \mod p \)

- Others know \( p, g, g^\alpha \mod p, g^\beta \mod p \).
- But computing \( g^{\alpha\beta} \mod p \) from \textbf{ONLY} this info, without knowing \( a \) and \( b \) is hard!! How hard?
A game of spies...

1. Choose a prime $p$ and a generator $g$ from $(\mathbb{Z}_p \setminus \{0\}, \times_p)$.
2. Alice fixes a private key $\alpha$ and Bob fixes $\beta$.
3. Alice computes $M = g^\alpha \mod p$ and shouts/sends it.
4. Bob computes $N = g^\beta \mod p$ and sends/shouts it.
5. Alice computes $M^\alpha \mod p$ and Bob computes $N^\beta \mod p$.

Shared Key: $g^{\alpha \beta} \mod p$

- Others know $p, g, g^\alpha \mod p, g^\beta \mod p$.
- But computing $g^{\alpha \beta} \mod p$ from ONLY this info, without knowing $a$ and $b$ is hard!! How hard?
- Does there exist a poly-time (in size of digits) algorithm?
A game of spies...

1. Choose a prime $p$ and a generator $g$ from $(\mathbb{Z}_p \setminus \{0\}, \times_p)$.
2. Alice fixes a private key $\alpha$ and Bob fixes $\beta$.
3. Alice computes $M = g^\alpha \mod p$ and shouts/sends it.
4. Bob computes $N = g^\beta \mod p$ and sends/shouts it.
5. Alice computes $M^\alpha \mod p$ and Bob computes $N^\beta \mod p$.

Shared Key: $g^{\alpha \beta} \mod p$

- Others know $p, g, g^\alpha \mod p, g^\beta \mod p$.
- But computing $g^{\alpha \beta} \mod p$ from ONLY this info, without knowing $a$ and $b$ is hard!! How hard?
- Does there exist a poly-time (in size of digits) algorithm?
- This is called the Diffie-Hellman problem. Still open...
A game of spies...

1. Choose a prime $p$ and a generator $g$ from $(\mathbb{Z}_p \setminus \{0\}, \times_p)$.
2. Alice fixes a private key $\alpha$ and Bob fixes $\beta$.
3. Alice computes $M = g^\alpha \mod p$ and shouts/sends it.
4. Bob computes $N = g^\beta \mod p$ and sends/shouts it.
5. Alice computes $M^\alpha \mod p$ and Bob computes $N^\beta \mod p$.

Shared Key: $g^{\alpha \beta} \mod p$

- Others know $p, g, g^\alpha \mod p, g^\beta \mod p$.
- But computing $g^{\alpha \beta} \mod p$ from ONLY this info, without knowing $a$ and $b$ is hard!! How hard?
- Does there exist a poly-time (in size of digits) algorithm?
- This is called the Diffie-Hellman problem. Still open...
- In practice, choose large primes with $\sim 300$ digits.
More generally...

Start with any finite cyclic group $G$ and generator $g \in G$

1. *Alice* picks a random $a \in \mathbb{N}$ and sends $g^a$ to *Bob*.
2. *Bob* picks a random $b \in \mathbb{N}$ and sends $g^b$ to *Alice*.
3. *Alice* computes $(g^b)^a$ and *Bob* computes $(g^a)^b$.
4. Shared key is $g^{ab}$.

- Of course, we know modular logarithm we could do it!
- i.e., if $g^a = g'$ and $g$ and $g'$ are given, what is $a$?
- Called the discrete logarithm problem and it is also open!
- What is a naive algorithm? Why does it not work?
- But there exists a quantum algorithm which runs in poly time!
Sending messages with Diffie-Hellman

Start with any finite cyclic group $G$ and generator $g \in G$

1. *Alice* picks a random $a \in \mathbb{N}$ and sends $g^a$ to *Bob*.
2. *Bob* picks a random $b \in \mathbb{N}$ and sends $g^b$ to *Alice*.
3. *Alice* computes $(g^b)^a$ and *Bob* computes $(g^a)^b$.
4. Shared key is $g^{ab}$.
Sending messages with Diffie-Hellman

Start with any finite cyclic group $G$ and generator $g \in G$

1. Alice picks a random $a \in \mathbb{N}$ and sends $g^a$ to Bob.
2. Bob picks a random $b \in \mathbb{N}$ and sends $g^b$ to Alice.
3. Alice computes $(g^b)^a$ and Bob computes $(g^a)^b$.
4. Shared key is $g^{ab}$.

▶ Alice encrypts message $m$ as $mg^{ab}$ and sends it.
### Sending messages with Diffie-Hellman

Start with any finite cyclic group $G$ and generator $g \in G$

1. *Alice* picks a random $a \in \mathbb{N}$ and sends $g^a$ to *Bob*.
2. *Bob* picks a random $b \in \mathbb{N}$ and sends $g^b$ to *Alice*.
3. *Alice* computes $(g^b)^a$ and *Bob* computes $(g^a)^b$.
4. Shared key is $g^{ab}$.

- *Alice* encrypts message $m$ as $mg^{ab}$ and sends it.
- So to decrypt it *Bob* needs to compute $(g^{ab})^{-1}$. 

▶ Application of Lagrange's theorem!
Sending messages with Diffie-Hellman

Start with any finite cyclic group $G$ and generator $g \in G$

1. *Alice* picks a random $a \in \mathbb{N}$ and sends $g^a$ to *Bob*.
2. *Bob* picks a random $b \in \mathbb{N}$ and sends $g^b$ to *Alice*.
3. *Alice* computes $(g^b)^a$ and *Bob* computes $(g^a)^b$.
4. Shared key is $g^{ab}$.

- *Alice* encrypts message $m$ as $mg^{ab}$ and sends it.
- So to decrypt it *Bob* needs to compute $(g^{ab})^{-1}$.
- So *Bob* computes:
Sending messages with Diffie-Hellman

Start with any finite cyclic group $G$ and generator $g \in G$

1. Alice picks a random $a \in \mathbb{N}$ and sends $g^a$ to Bob.
2. Bob picks a random $b \in \mathbb{N}$ and sends $g^b$ to Alice.
3. Alice computes $(g^b)^a$ and Bob computes $(g^a)^b$.
4. Shared key is $g^{ab}$.

- Alice encrypts message $m$ as $mg^{ab}$ and sends it.
- So to decrypt it Bob needs to compute $(g^{ab})^{-1}$.
- So Bob computes:

$$ (g^a)^{|G| - b} = $$
Sending messages with Diffie-Hellman

Start with any finite cyclic group $G$ and generator $g \in G$

1. Alice picks a random $a \in \mathbb{N}$ and sends $g^a$ to Bob.
2. Bob picks a random $b \in \mathbb{N}$ and sends $g^b$ to Alice.
3. Alice computes $(g^b)^a$ and Bob computes $(g^a)^b$.
4. Shared key is $g^{ab}$.

- Alice encrypts message $m$ as $mg^{ab}$ and sends it.
- So to decrypt it Bob needs to compute $(g^{ab})^{-1}$.
- So Bob computes:
  \[(g^a)^{|G| - b} = g^{a|G| - ab} = (g^{|G|})^a(g^{-ab}) = e^a(g^{-ab}) = (g^{ab})^{-1}\]
  - Application of Lagrange’s theorem!
Sending messages with Diffie-Hellman

Start with any finite cyclic group $G$ and generator $g \in G$

1. *Alice* picks a random $a \in \mathbb{N}$ and sends $g^a$ to *Bob*.
2. *Bob* picks a random $b \in \mathbb{N}$ and sends $g^b$ to *Alice*.
3. *Alice* computes $(g^b)^a$ and *Bob* computes $(g^a)^b$.
4. Shared key is $g^{ab}$.

- *Alice* encrypts message $m$ as $mg^{ab}$ and sends it.
- So to decrypt it *Bob* needs to compute $(g^{ab})^{-1}$.
- So *Bob* computes:
  
  $$(g^a)^{|G|^{-b}} = g^a|G|^{-ab} = (g^{|G|})^a(g^{-ab}) = e^a(g^{-ab}) = (g^{ab})^{-1}$$
  
  – Application of Lagrange’s theorem!

- Then *Bob* just applies this on msg received.
Sending messages with Diffie-Hellman

Start with any finite cyclic group $G$ and generator $g \in G$

1. *Alice* picks a random $a \in \mathbb{N}$ and sends $g^a$ to *Bob*.
2. *Bob* picks a random $b \in \mathbb{N}$ and sends $g^b$ to *Alice*.
3. *Alice* computes $(g^b)^a$ and *Bob* computes $(g^a)^b$.
4. Shared key is $g^{ab}$.

- *Alice* encrypts message $m$ as $mg^{ab}$ and sends it.
- So to decrypt it *Bob* needs to compute $(g^{ab})^{-1}$.
- So *Bob* computes:
  \[
  (g^a)^{|G|-b} = g^{a|G|-ab} = (g^{|G|})^a(g^{-ab}) = e^a(g^{-ab}) = (g^{ab})^{-1}
  \]
  - Application of Lagrange’s theorem!
- Then *Bob* just applies this on msg received.
- That is, $mg^{ab}(g^{ab})^{-1} = m \cdot e = m$. 
Diffie-Hellman Key Exchange protocol

This was discovered by Diffie & Hellman in 1976.
Considered to be first cryptographic protocol.
Variants of this are still used everywhere!
  - Digital signatures for Sony Playstations.
  - GNU Privacy guard, PGP (pretty good privacy)...
Which cyclic group?
Replace \((\mathbb{Z}_p \setminus \{0\}, \times_p)\) by cyclic group of points of elliptic curves.
  - Elliptic Curve Diffie-Hellman Cryptography.