CMSC/Math 456: Cryptography (Fall 2022)

Lecture 15

Daniel Gottesman

Administrative

Midterm: Thursday, Oct. 20 (Thursday)

- In class
- Open book (including textbook), no electronic devices
- Will cover material through Diffie-Hellman and El Gamal, but not RSA.

Solution set #5 is out. Practice problems from the textbook are available on the course web page.

This class is being recorded

Motivation for Reductions

What do we want to accomplish with a reduction?

- We want to measure the relative difficulty of different computational problems. Example: reduction from Diffie-Hellman to discrete log.
- We have one problem we believe is hard and want to use it to show that other problems are hard (e.g., breaking a cryptographic protocol). We haven't really seen an example of this in class yet.
- We want to use a cryptographic primitive as a component to implement a more complicated protocol and want to prove that this is safe to do. Example: pseudorandom generator to make pseudo one-time pad.

Basic Concept of a Reduction

At its heart, a reduction is a statement of the form "If you can do A, then you can do B."

Examples:

- If you can boil water, then you can cook an egg.
- If you can roller blade, then you can ice skate.
- If you can ice skate, then you can roller blade.
- If you can breathe water, then you can visit sunken ships.
- If you can go faster than light, you can travel back in time.

But this doesn't really work:

- If you can breathe water, then you can fly to the moon.

because it is non-constructive.

A reduction is supposed to tell you how doing A lets you do B.

Contrapositive of a Reduction

If you can do A, then you can do B.

$$\Updownarrow$$

If you can't do B, then you can't do A.

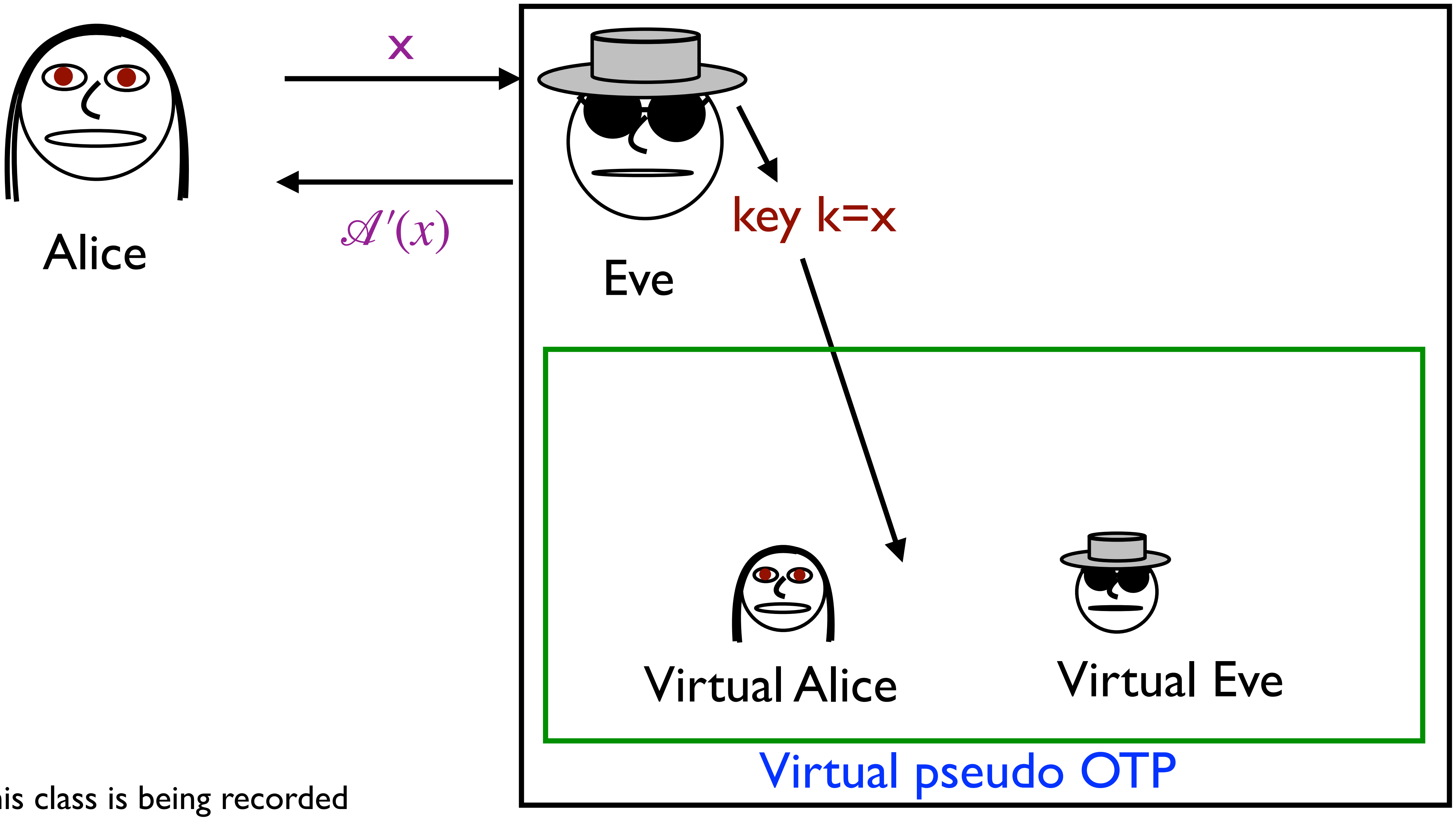
We most often prove the "A implies B" statement of a reduction but most often use the "not B implies not A" contrapositive.

E.g.:

- If you can't roller blade, then you must not be able to ice skate.
- If you can't break the pseudorandom generator, then you can't break the pseudo one-time pad.

Pseudo One-Time Pad Again

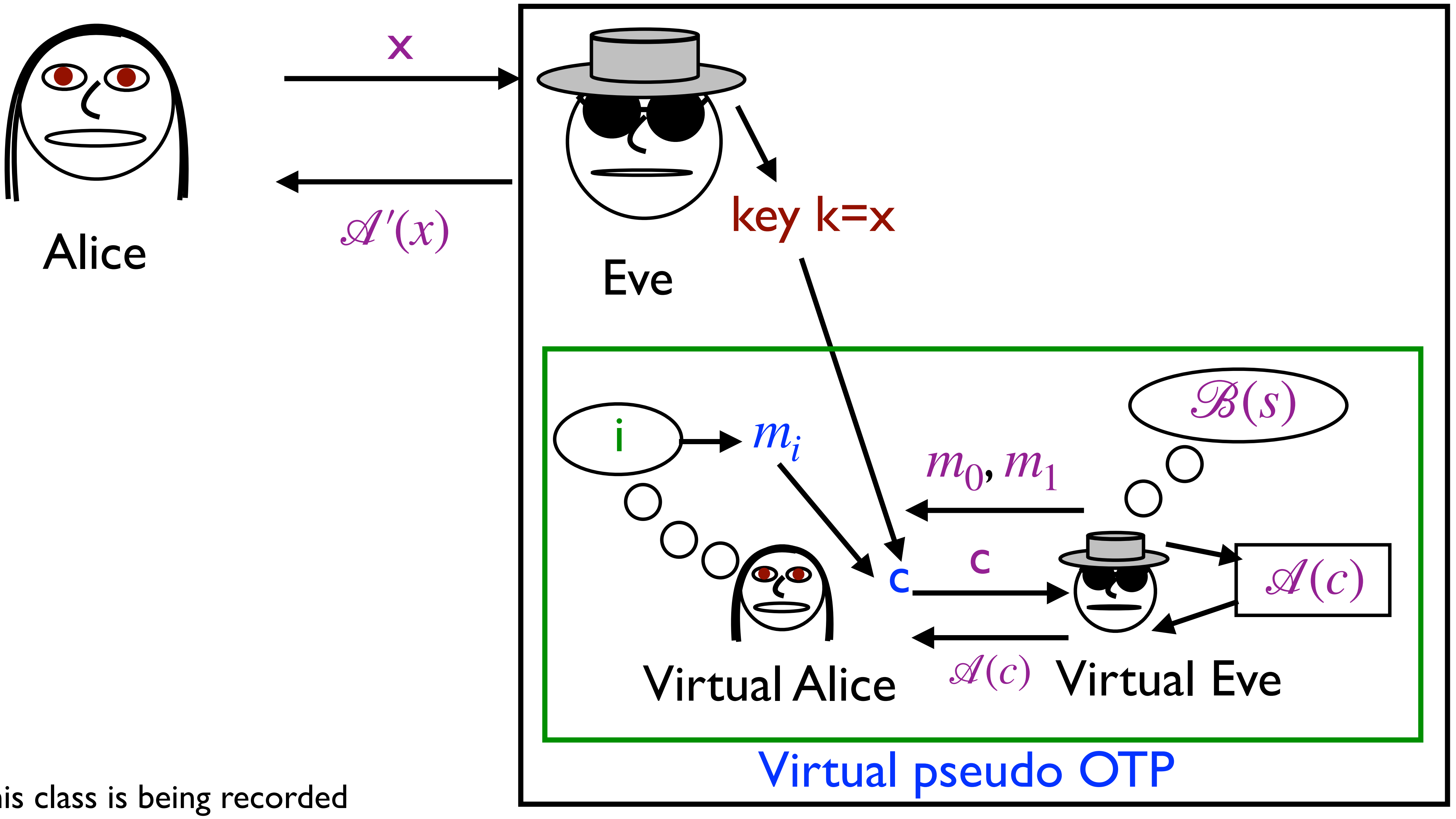
We used a reduction to say "If you can break the pseudo one-time pad, then you can break the pseudorandom generator."



This class is being recorded

Pseudo One-Time Pad Again

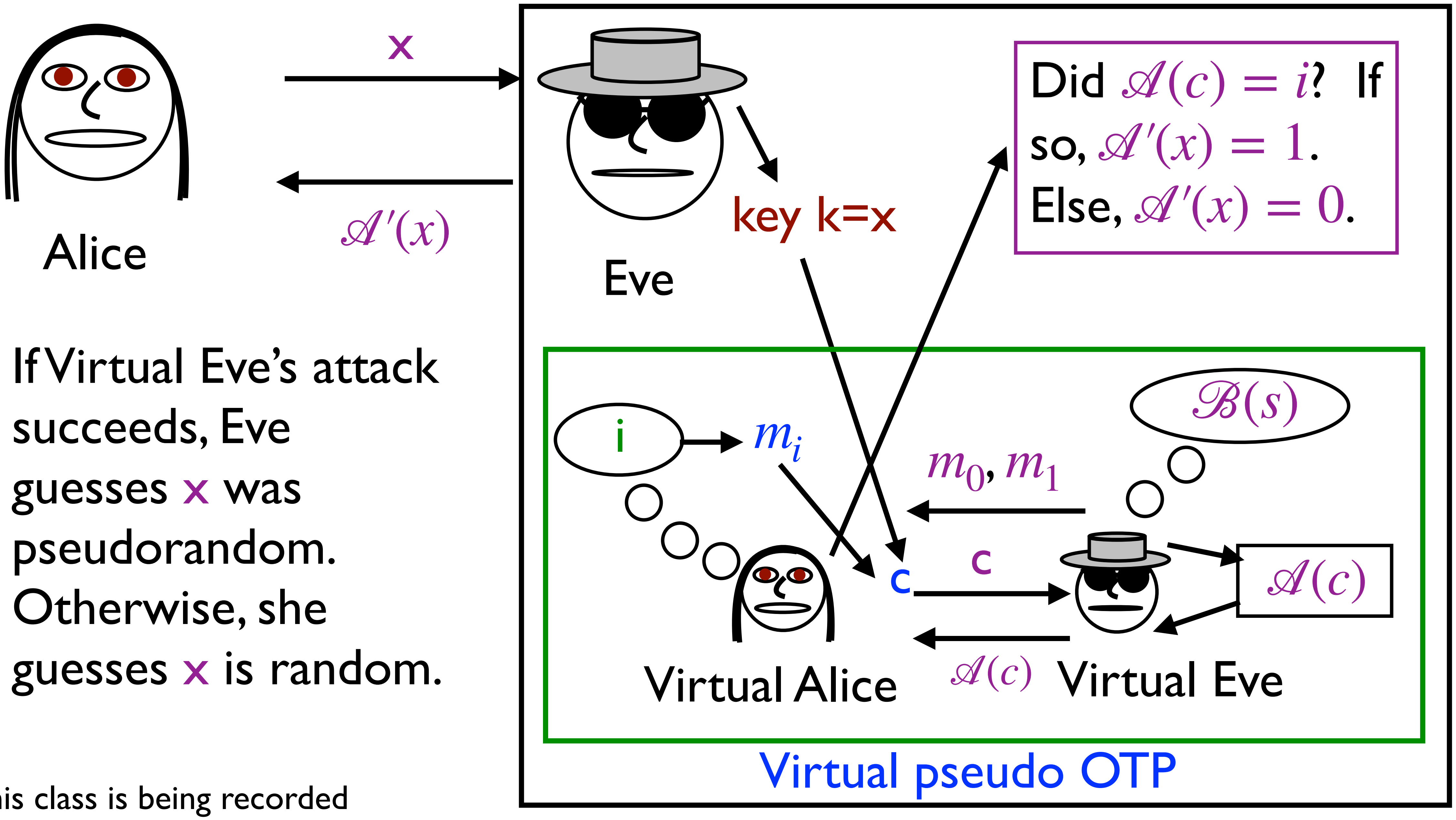
We used a reduction to say "If you can break the pseudo one-time pad, then you can break the pseudorandom generator."



This class is being recorded

Pseudo One-Time Pad Again

We used a reduction to say "If you can break the pseudo one-time pad, then you can break the pseudorandom generator."



If Virtual Eve's attack succeeds, Eve guesses x was pseudorandom. Otherwise, she guesses x is random.

This class is being recorded

Strategy for Deriving The Reduction

To make this reduction work, we need to fill in the space between the "A" statement and the "B" statement:

1. If you can break the pseudo one-time pad, then

?

n. You can break the pseudorandom generator.

Strategy II

To start to fill in the blanks, we should think about what those statements mean.

1. If you can break the pseudo one-time pad, then
2. There is an attack $\mathscr{A}(c)$ that guesses the message successfully in the pseudo one-time pad.

?

n-1. There is an attack $\mathscr{A}'(x)$ that distinguishes G(y) from random x.
n. You can break the pseudorandom generator.

Strategy II

To start to fill in the blanks, we should think about what those statements mean.

1. If you can break the pseudo one-time pad, then
2. There is an attack $\mathscr{A}(c)$ that guesses the message successfully in the pseudo one-time pad.

?

We need a strategy for constructing $\mathscr{A}'(x)$ from $\mathscr{A}(c)$.

n-1. There is an attack $\mathscr{A}'(x)$ that distinguishes G(y) from random x.

n. You can break the pseudorandom generator.

This class is being recorded

Strategy III

We need a strategy for constructing $\mathscr{A}'(x)$ from $\mathscr{A}(c)$.

> Here we can recognize that if there is an attack against the pseudo one-time pad, then that makes the pseudo one-time pad different from the true one-time pad, which is perfectly secret.

But the only difference between the pseudo one-time pad and the one-time pad is that the former uses G(y) as key and the latter uses a random x as key.

> So $\mathscr{A}(c)$ should succeed when the key is G(y) and it must fail when the key is a random x.

This suggests that we should create the virtual protocol and an $\mathscr{A}'(x)$ that reports whether the attack $\mathscr{A}(c)$ succeeds.

Strategy IV

1. If you can break the pseudo one-time pad, then
2. There is an attack $\mathscr{A}(c)$ that guesses the message successfully in the pseudo one-time pad.
3. Define a virtual protocol to run the pseudo one-time pad experiment with attack $\mathscr{A}(c)$ and an $\mathscr{A}'(x)$ that reports whether the attack $\mathscr{A}(c)$ succeeds.

?

n-1. There is an attack $\mathscr{A}'(x)$ that distinguishes G(y) from random x.
n. You can break the pseudorandom generator.

Strategy IV

1. If you can break the pseudo one-time pad, then
2. There is an attack $\mathscr{A}(c)$ that guesses the message successfully in the pseudo one-time pad.
3. Define a virtual protocol to run the pseudo one-time pad experiment with attack $\mathscr{A}(c)$ and an $\mathscr{A}'(x)$ that reports whether the attack $\mathscr{A}(c)$ succeeds.

?

Now we need to fill in the gaps by analyzing the attack $\mathscr{A}'(x)$ and showing that it gives us the desired attack against the PRG.

n-1. There is an attack $\mathscr{A}'(x)$ that distinguishes G(y) from random x.
n. You can break the pseudorandom generator.

This class is being recorded

Complete Reduction Outline

1. If you can break the pseudo one-time pad, then
2. There is an attack $\mathscr{A}(c)$ that guesses the message successfully in the pseudo one-time pad.
3. Define a virtual protocol to run the pseudo one-time pad experiment with attack $\mathscr{A}(c)$ and an $\mathscr{A}'(x)$ that reports whether the attack $\mathscr{A}(c)$ succeeds.

4. Calculate probability of success of $\mathscr{A}(x)$ when x is random.
5. Calculate probability of success of $\mathscr{A}(x)$ when x = G(y).
6. Verify that $\mathscr{A}'(x)$ runs in polynomial time.

7. There is an attack $\mathscr{A}'(x)$ that distinguishes G(y) from random x.
8. You can break the pseudorandom generator.

Modular Arithmetic Overview

Modular arithmetic mod N is an alternate system of arithmetic using just the numbers 0 through N-1.
Think of modular numbers as an alternate data type which happens sometimes to agree with integer type but not always.

However, we do want modular addition and multiplication to have many of the same properties as integer addition and multiplication.

- Addition and subtraction work basically the same way as integer addition and subtraction.
- Multiplication is also very similar to integer multiplication.
- Division, however, is quite different and is not always defined.
- Exponentiation is similar to integer exponentiation.

The other big difference is that modular numbers don't have a well-defined notion of "bigger" since counting up eventually brings us back to where we started.

Modular Arithmetic Examples

Mod 5 addition and multiplication:

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

*	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Each non-zero row and column has all #s

Mod 6 addition and multiplication:

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

*	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

Rows and columns have 0s and repeat #s

Modular Arithmetic Summary

- Division by b is well-defined mod N when $\gcd(b, N) = 1$.
- $\mathbb{Z}_N^* = \{b \mid \gcd(b, N) = 1\}$ is a group.
- Euclid's algorithm takes input a and b and finds X and Y such that $aX + bY = \gcd(a, b)$.
  - Runs in time polynomial in $\log a$ and $\log b$.
  - Can find gcd.
  - Can find modular multiplicative inverse $a^{-1} \bmod N$.
- Modular exponentiation:
  - Efficient algorithm by repeated squaring.
  - Eventually repeats. The smallest r such that $g^r = 1 \bmod N$ is the order of g mod N. Then $g^a = g^b \bmod N$ if $a = b \bmod r$.
- Totient function $\varphi(N)$: number of #s <N relatively prime to N.
  - If p is prime, $\varphi(p) = p - 1$. If $N = pq$, $\varphi(N) = (p-1)(q-1)$.
  - Euler-Fermat: $g^{\varphi(N)} = 1 \bmod N$ when $\gcd(g, N) = 1$.
- Order of g mod N is a factor of $\varphi(N)$.
- If g is a generator mod N, $\mathrm{ord}(g^j) = \varphi(N)/\gcd(j, \varphi(N))$.

Chinese Remainder Theorem

Chinese remainder theorem: When a, b relatively prime,

$$x = x_a \bmod a$$
$$x = x_b \bmod b$$

have unique solution x mod ab.

Algorithm:

Run Euclid's algorithm to find X and Y such that

$$aX + bY = 1$$

Then

$$x = x_b aX + x_a bY$$

Example: $a = 3, b = 5, x_a = 2, x_b = 1$

$$x = 2 \bmod 3$$
$$x = 1 \bmod 5$$

Euclid's algorithm:

$$3 * 2 + 5 * (-1) = 1$$

$X = 2, Y = -1$

Then

$$x = 1 * 3 * 2 + 2 * 5 * (-1)$$
$$= 6 - 10 = -4 = 11 \bmod 15$$

This class is being recorded

Group Theory Summary

Definition: A group (G, *) is a set G of elements along with a binary operation $* : G \times G \to G$ with the following properties:

1. Closure: $g * h \in G$ when $g, h \in G$.
2. Associativity: $\forall g, h, k \in G, (g * h) * k = g * (h * k)$.
3. Identity: $\exists e \in G$ such that $\forall g \in G, e * g = g * e = g$.
4. Inverses: $\forall g \in G, \exists g^{-1} \in G$ such that $g * g^{-1} = g^{-1} * g = e$.

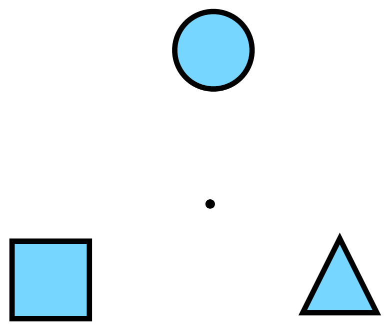
A subgroup H of G, written $H \leq G$ is a subset of G which is also a group. The order $|G|$ of a finite group G is the number of elements.

A set S generates a group G if all elements of G can be written as products of elements of S. A group that can be generated by just one element is cyclic.

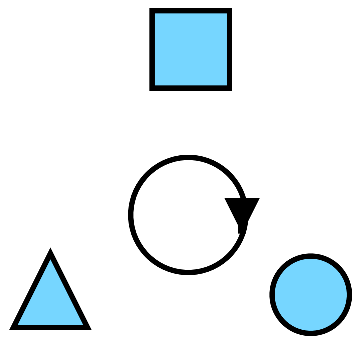
Lagrange's Theorem: If H and G are finite groups with $H \leq G$, then $|H|$ divides $|G|$.

Group Theory Example

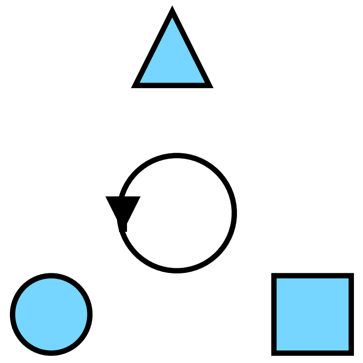
Permutations of 3 elements: Group S_3, $|S_3| = 6$



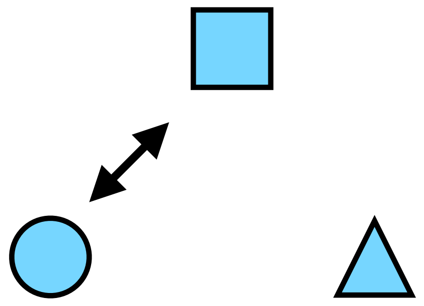
Identity e



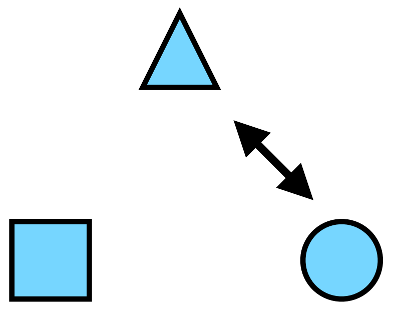
Rotate clockwise R



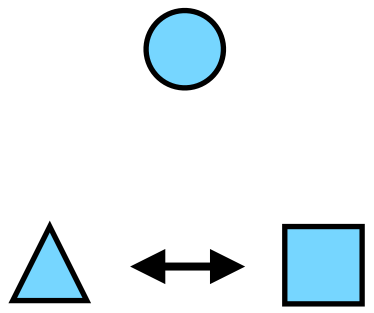
Rotate ccw $R^{-1} = R^2$



Swap left S_L



Swap right S_R



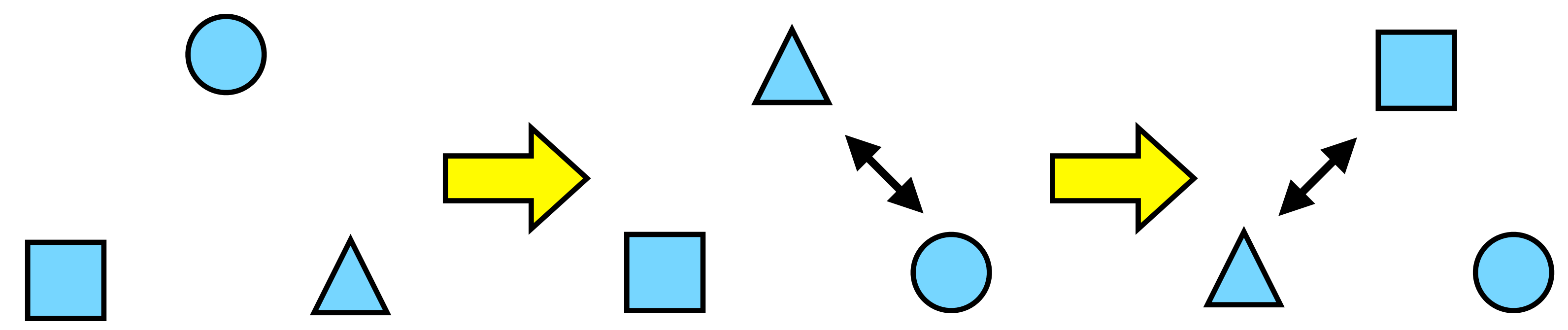
Swap bottom S_B

$S_3 = \{e, R, R^2, S_L, S_R, S_B\}$ with group operation composition.

Group Properties: Closure

Closure: Product of two permutations is a permutation.

E.g.: $S_L S_R$ (acts from right)

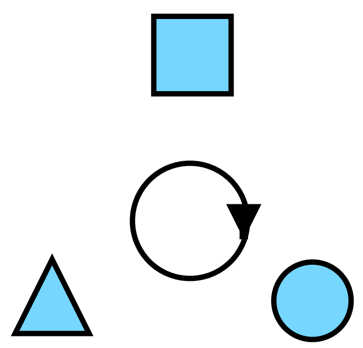


Starting arrangement

S_R

S_L

=



$= R$

This class is being recorded

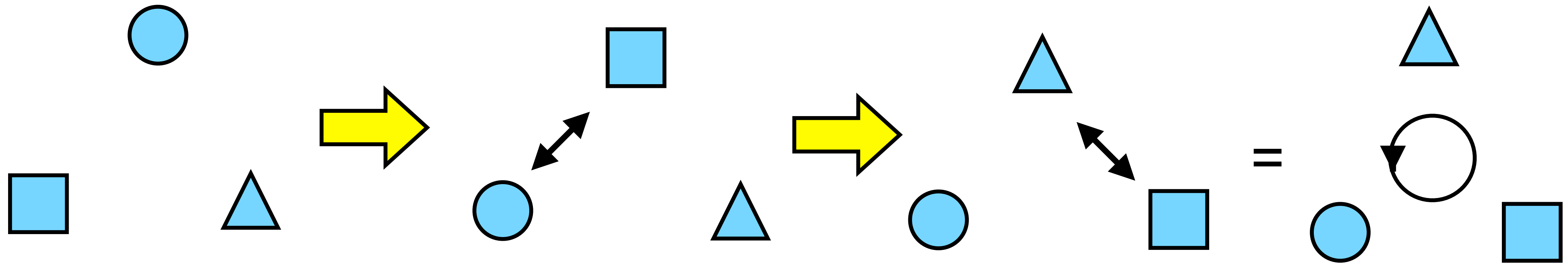
Group Properties: Others

Associativity: Can be checked but is automatic for composition of operations.

Identity: e is the identity element of the group. $e\sigma = \sigma$

Inverses: R and R^2 are inverses of each other. S_L, S_R, and S_B are inverses of themselves.

Non-abelian: $S_R S_L = R^2 \neq R = S_L S_R$



Starting arrangement

Swap left S_L

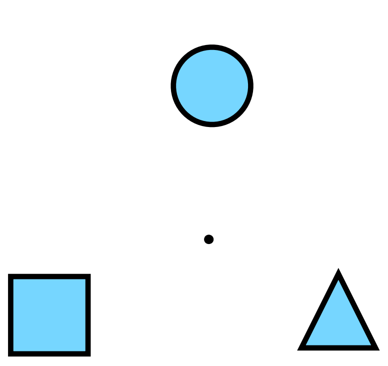
Swap right S_R

R^2

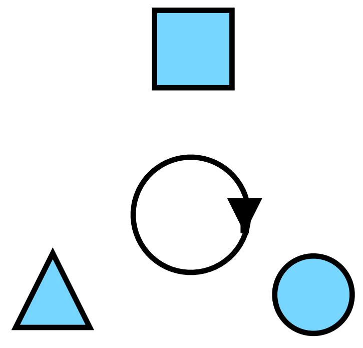
This class is being recorded

Subgroups

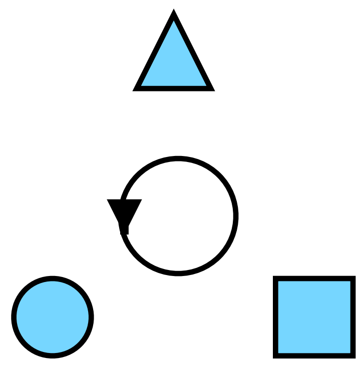
Order 3: Generated by R or by R^2.



Identity e

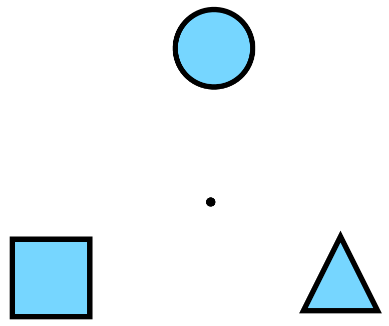


Rotate clockwise R

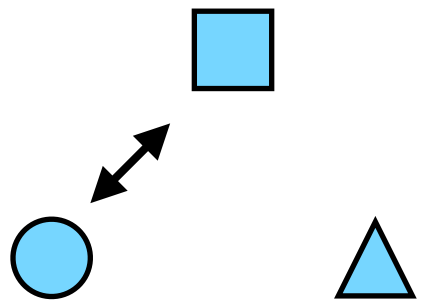


Rotate ccw $R^{-1} = R^2$

3 order 2 subgroups: For instance, generated by S_L.



Identity e



Swap left S_L

Note: Order of subgroups a factor of 6 (Lagrange's thm.)

This class is being recorded