

# Information Systems and **Network Security**

**Instructor:** Stefano Leucci



**Email:** stefano.leucci@univaq.it

# Information Systems and **Network Security**

**Instructor:** Stefano Leucci



**Email:** stefano.leucci@univaq.it

# Information Systems and **Network Security**

**Instructor:** Stefano Leucci



**Email:** stefano.leucci@univaq.it

# Basic Info

**Course length:** 48 hours (6 CFU)

- 24 lectures

**When/where:**

- Thursday 16:30 – 18:30

**Room:** C1.16

- Friday 16:30 – 18:30

**Room:** A1.2



# Basic Info

**Course length:** 48 hours (6 CFU)

- 24 lectures

**When/where:**

- Thursday 16:30 – 18:30      **Room:** C1.16
- Friday 16:30 – 18:30      **Room:** A1.2



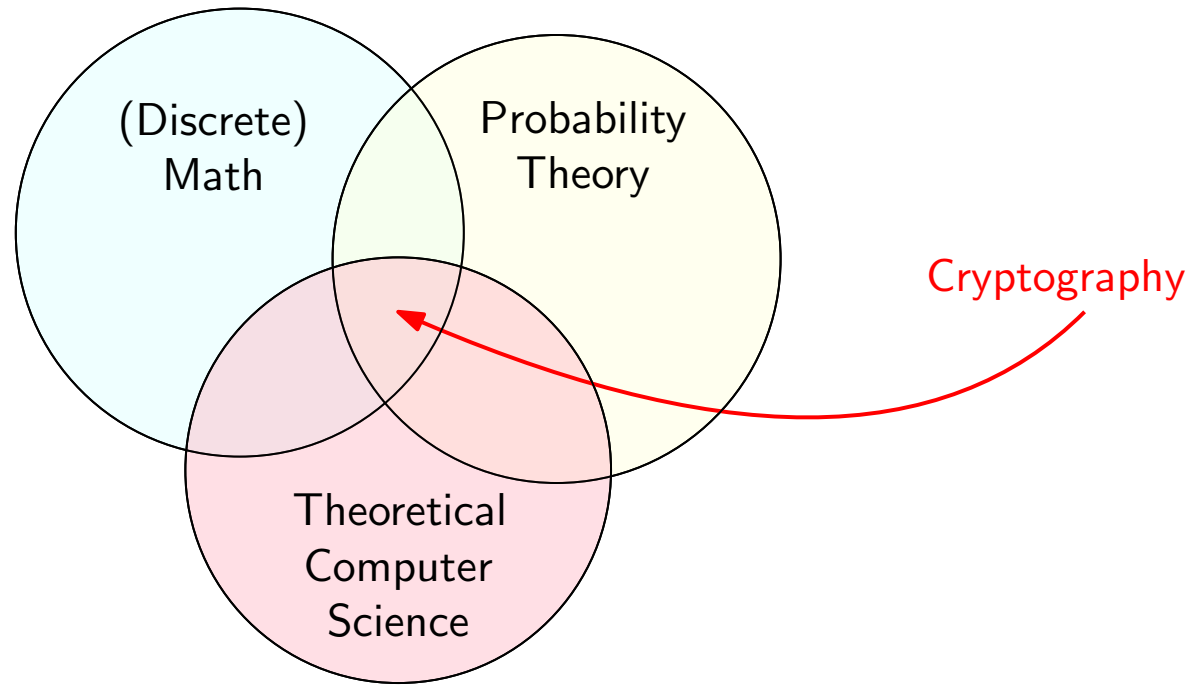
**Office hours:**

- Thursday 14:30 – 16:30
- Please send an email to [stefano.leucci@gmail.com](mailto:stefano.leucci@gmail.com) or ask before/after class

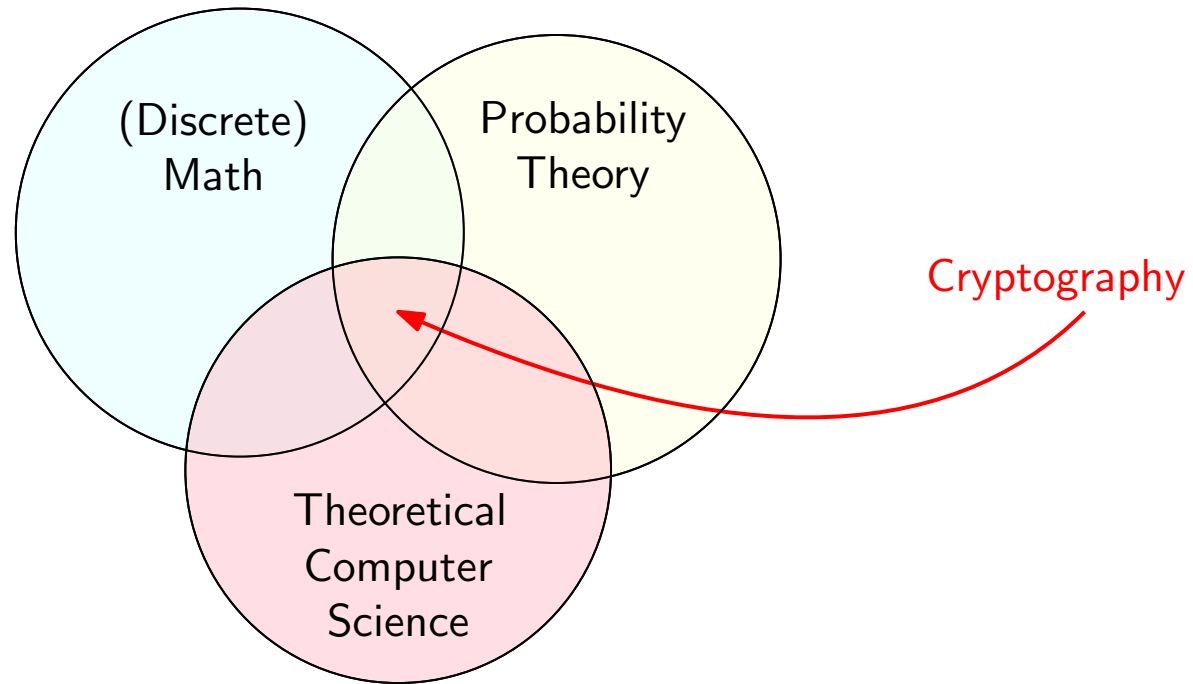
**Course material:**

<https://people.disim.univaq.it/~stefanoleucci/isns26/>

# Ingredients of Cryptography

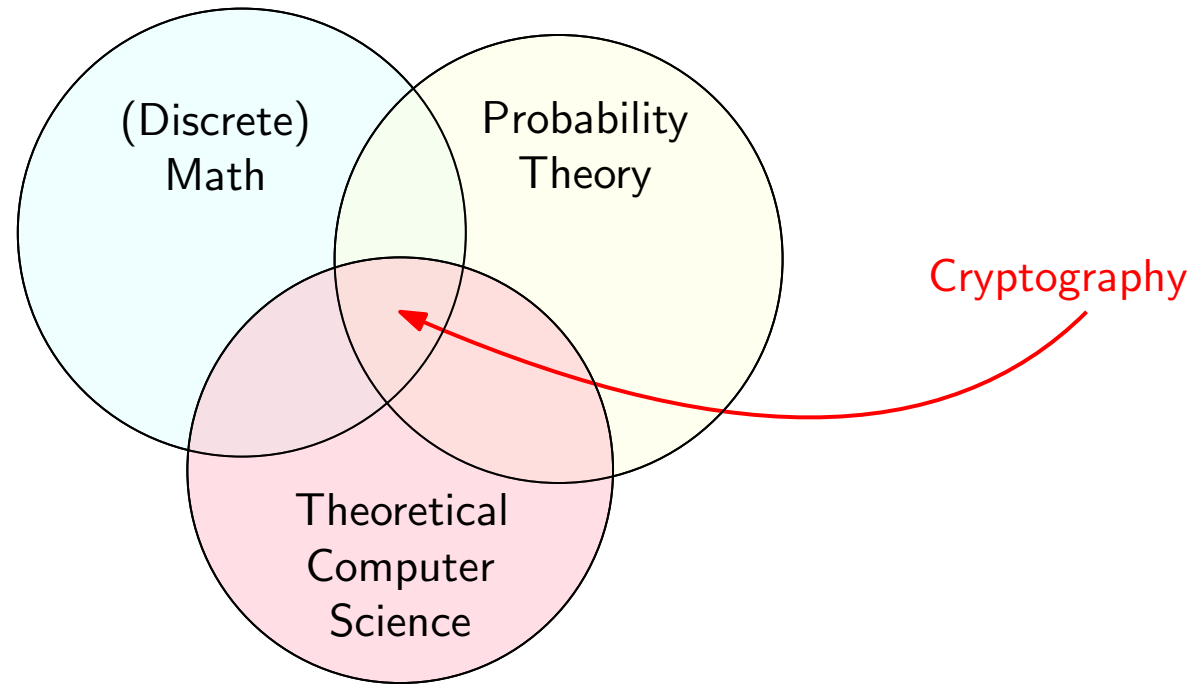


# Ingredients of Cryptography



**(Discrete) Math:** Basic algebra, Modular arithmetic, some concepts from group theory and number theory

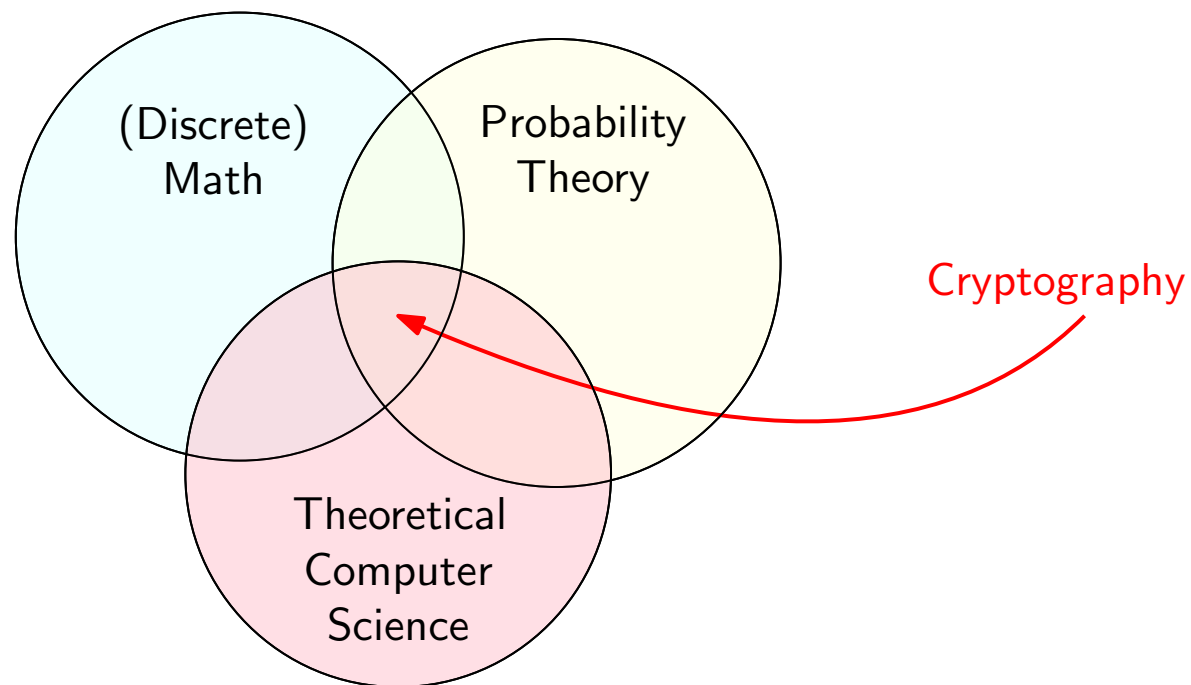
# Ingredients of Cryptography



**(Discrete) Math:** Basic algebra, Modular arithmetic, some concepts from group theory and number theory

**Probability theory:** Events, random variables, expectation, independence, conditional probability, Bayes' theorem, ...

# Ingredients of Cryptography

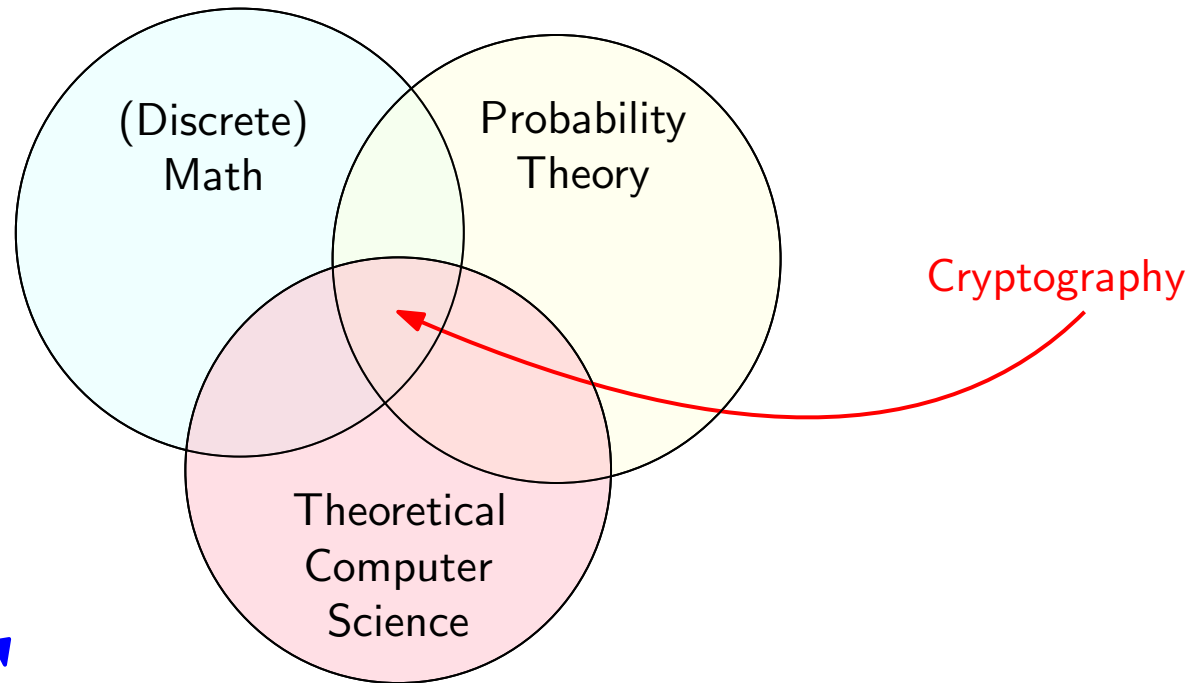


**(Discrete) Math:** Basic algebra, Modular arithmetic, some concepts from group theory and number theory

**Probability theory:** Events, random variables, expectation, independence, conditional probability, Bayes' theorem, ...

**Theoretical computer science:** Algorithms, complexity, asymptotic notation, reductions

# Ingredients of Cryptography



Prerequisites



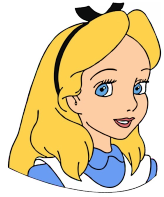
**(Discrete) Math:** Basic algebra, Modular arithmetic, some concepts from group theory and number theory

**Probability theory:** Events, random variables, expectation, independence, conditional probability, Bayes' theorem, ...

**Theoretical computer science:** Algorithms, complexity, asymptotic notation, reductions

Why cryptography?

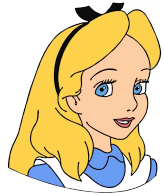
# Why cryptography?



Two entities wish to communicate

# Why cryptography?

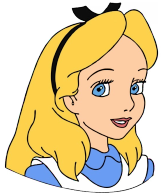
Alice



Two entities wish to communicate

# Why cryptography?

Alice

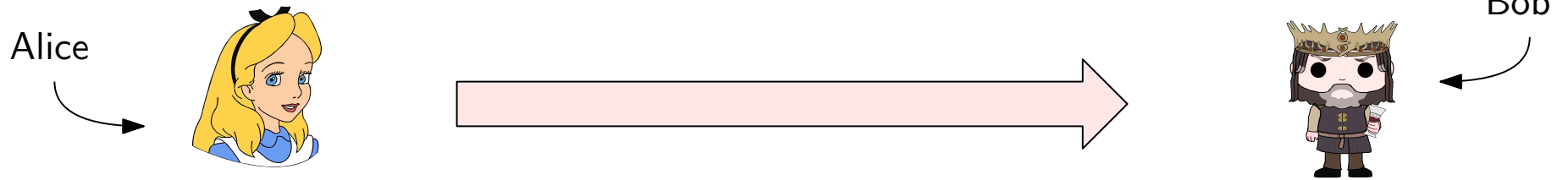


Bob



Two entities wish to communicate

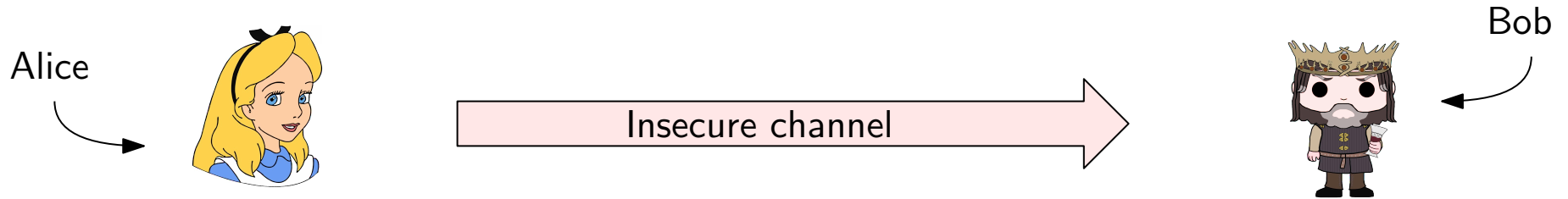
# Why cryptography?



Two entities wish to communicate

Alice can send a message to Bob through a communication channel

# Why cryptography?

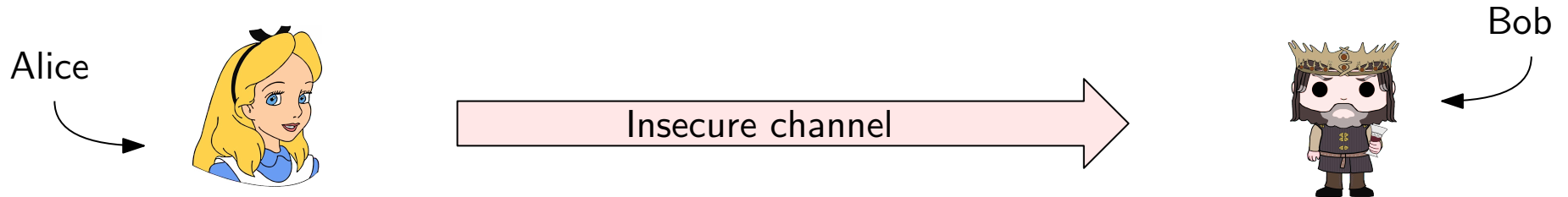


Two entities wish to communicate

Alice can send a message to Bob through a communication channel

**The communication channel is “insecure”**

# Why cryptography?



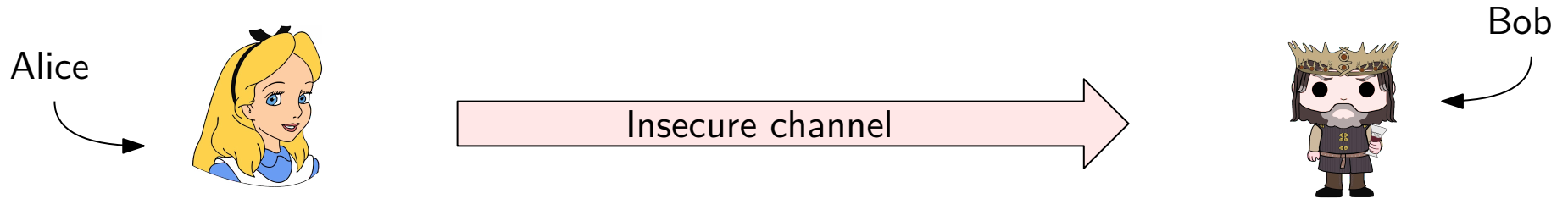
Two entities wish to communicate

Alice can send a message to Bob through a communication channel

**The communication channel is “insecure”**

- Anybody can view a message in transit

# Why cryptography?



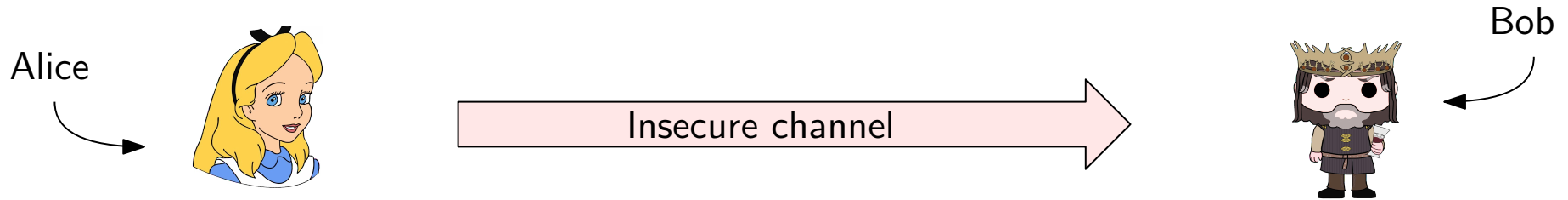
Two entities wish to communicate

Alice can send a message to Bob through a communication channel

**The communication channel is “insecure”**

- Anybody can view a message in transit
- Anybody can modify a message in transit

# Why cryptography?



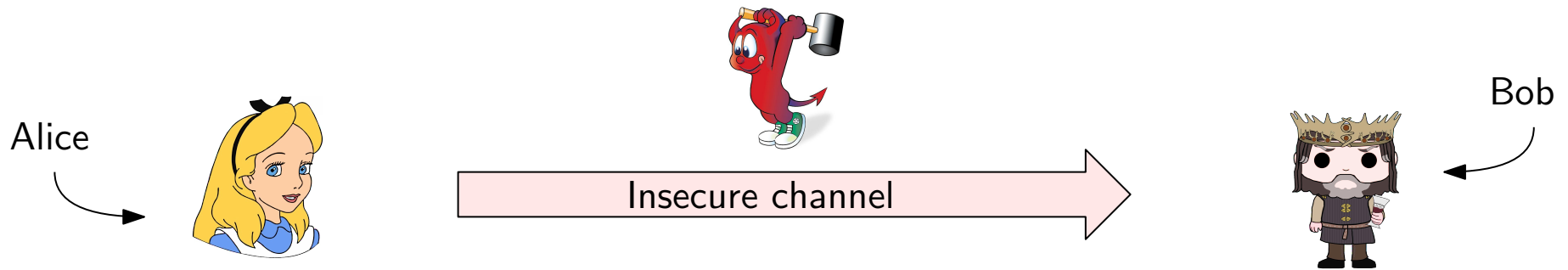
Two entities wish to communicate

Alice can send a message to Bob through a communication channel

**The communication channel is “insecure”**

- Anybody can view a message in transit
- Anybody can modify a message in transit
- Anybody can inject messages into the channel

# Why cryptography?



Two entities wish to communicate

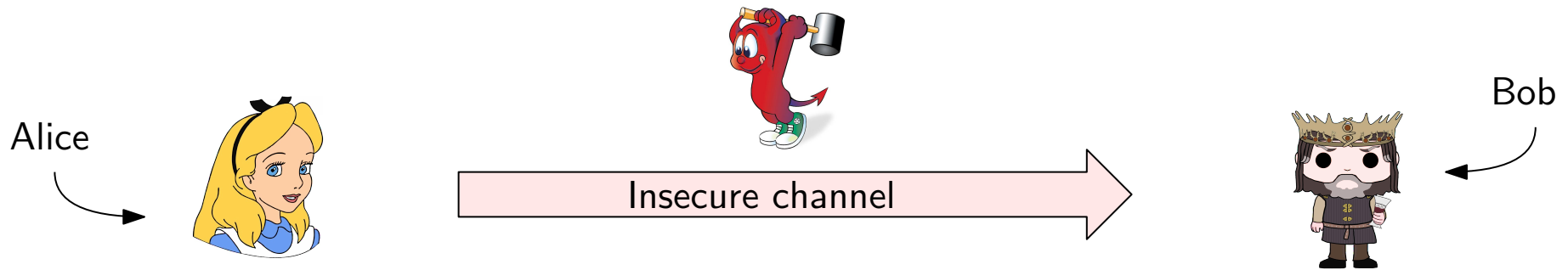
Alice can send a message to Bob through a communication channel

## The communication channel is “insecure”

- Anybody can view a message in transit
- Anybody can modify a message in transit
- Anybody can inject messages into the channel

} An “adversary” controls the communication channel

# Why cryptography?



Two entities wish to communicate

Alice can send a message to Bob through a communication channel

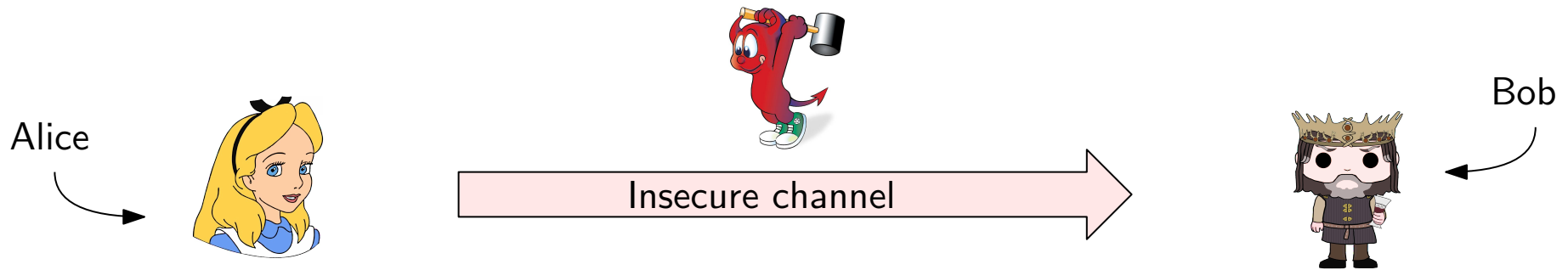
**The communication channel is “insecure”**

- Anybody can view a message in transit
- Anybody can modify a message in transit
- Anybody can inject messages into the channel

} An “adversary” controls the communication channel

**Can Alice and Bob communicate “securely”?**

# Why cryptography?



Two entities wish to communicate

Alice can send a message to Bob through a communication channel

**The communication channel is “insecure”**

- Anybody can view a message in transit
- Anybody can modify a message in transit
- Anybody can inject messages into the channel

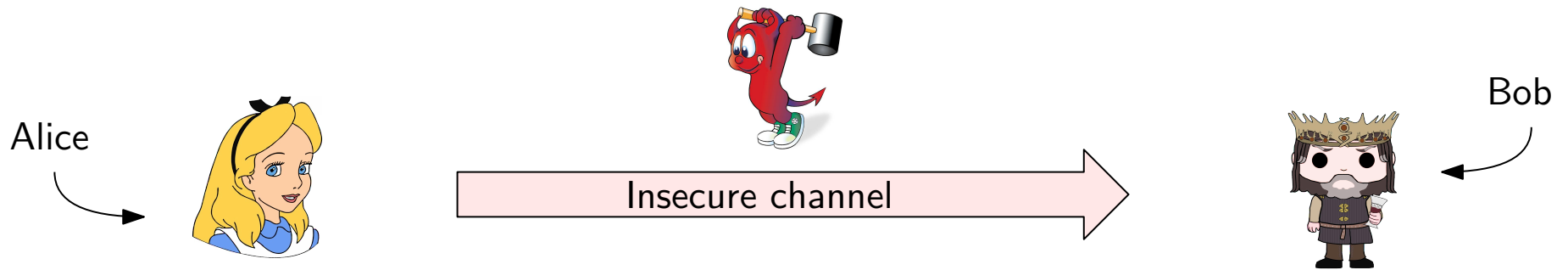


An “adversary” controls the communication channel

**Can Alice and Bob communicate “securely”?**

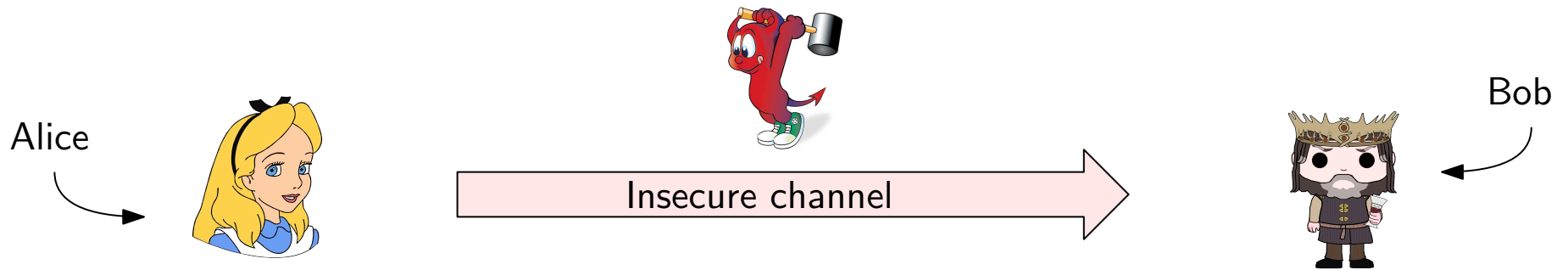
What does “secure” mean?

# Why cryptography?



Typically we want to guarantee the following (informal) properties:

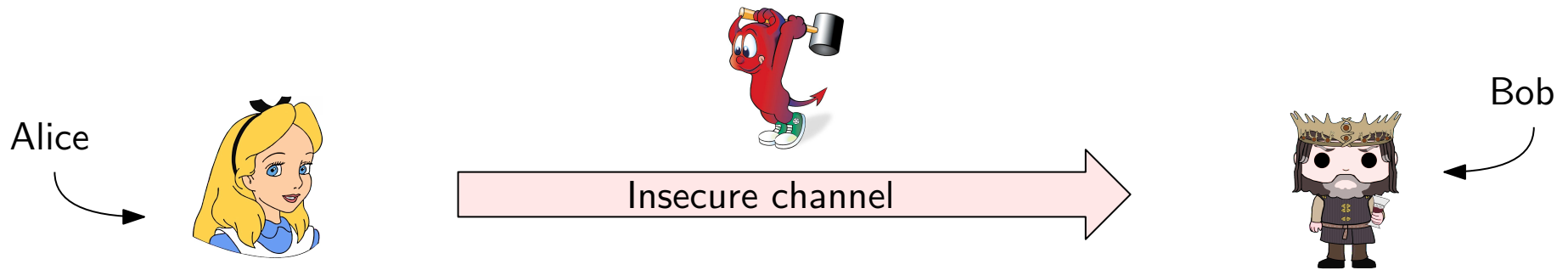
# Why cryptography?



Typically we want to guarantee the following (informal) properties:

- **Secrecy/Confidentiality:** The contents messages sent over the channel will be known only to Alice and Bob

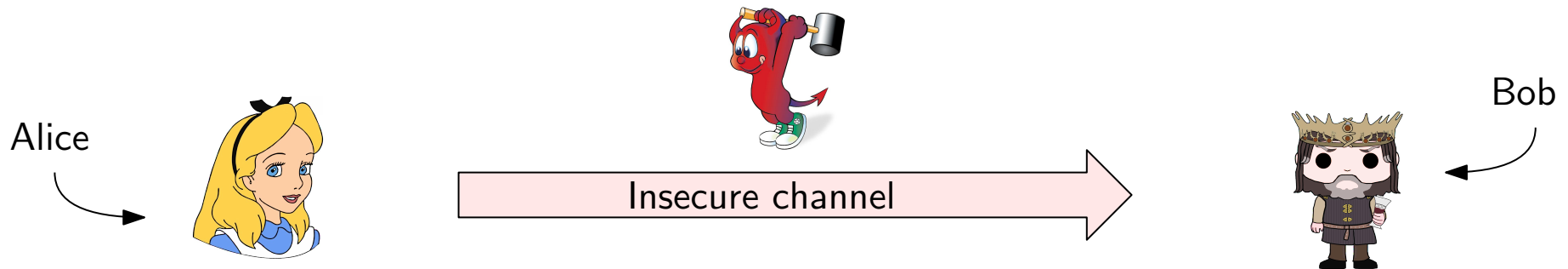
# Why cryptography?



Typically we want to guarantee the following (informal) properties:

- **Secrecy/Confidentiality:** The contents messages sent over the channel will be known only to Alice and Bob
- **Authentication:** any message received by Bob can be confirmed to have originated from Alice (we can detect any message injected by the adversary)

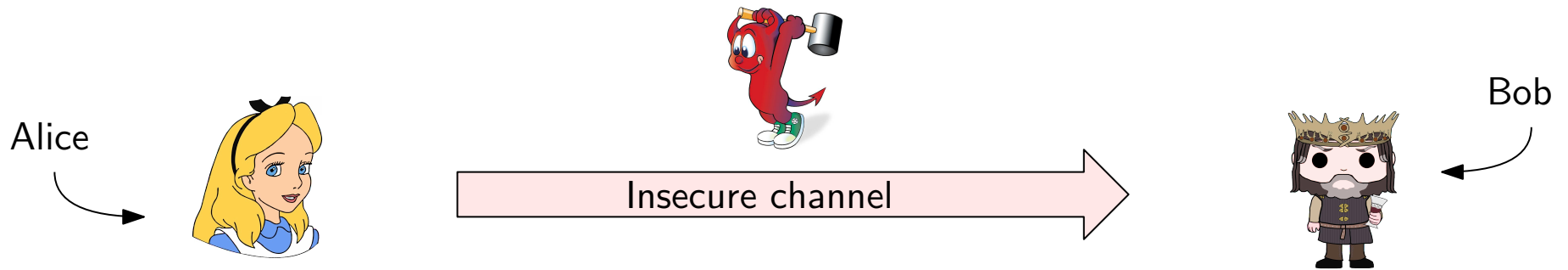
# Why cryptography?



Typically we want to guarantee the following (informal) properties:

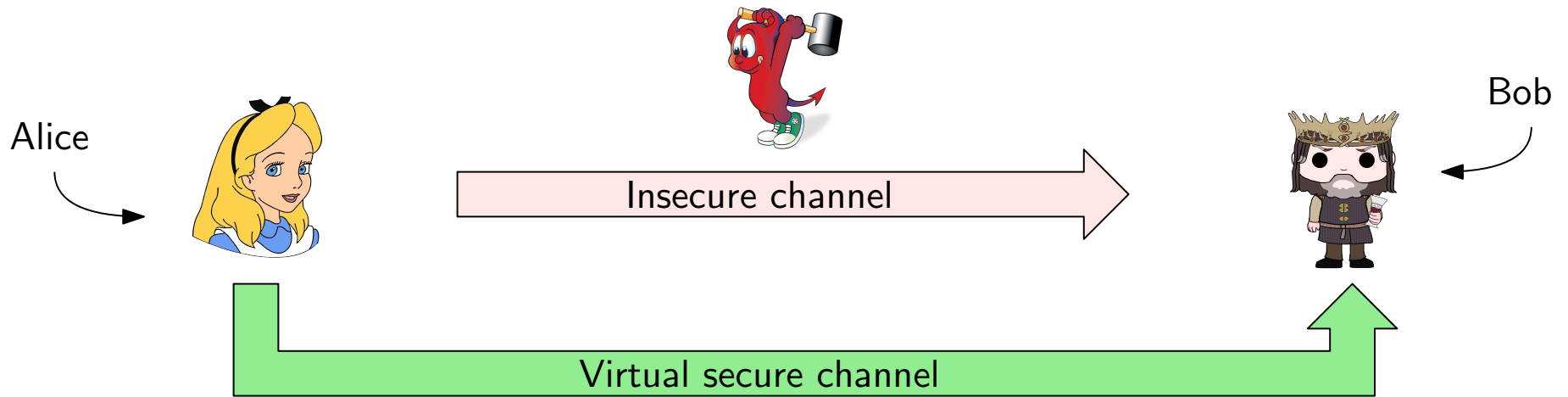
- **Secrecy/Confidentiality:** The contents messages sent over the channel will be known only to Alice and Bob
- **Authentication:** any message received by Bob can be confirmed to have originated from Alice (we can detect any message injected by the adversary)
- **Integrity:** it is not possible to alter the contents of a message sent across the channel (without the tampering being detected)

# Why cryptography?



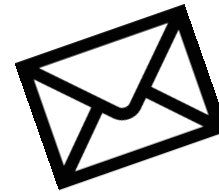
Intuitively, we established a virtual “secure” communication channel on top of an underlying insecure channel

# Why cryptography?



Intuitively, we established a virtual “secure” communication channel on top of an underlying insecure channel

# Why cryptography?



# Topics



- Introduction (why cryptography?)

# Topics



- Introduction (why cryptography?)
- Cryptography in the past: historic ciphers

# Topics



- Introduction (why cryptography?)
- Cryptography in the past: historic ciphers
- Cryptography today: formalizing security

# Topics



- Introduction (why cryptography?)
- Cryptography in the past: historic ciphers
- Cryptography today: formalizing security
- Security against eavesdroppers: achieving perfect secrecy

# Topics



- Introduction (why cryptography?)
- Cryptography in the past: historic ciphers
- Cryptography today: formalizing security
- Security against eavesdroppers: achieving perfect secrecy
- Drawbacks of perfect secrecy, computational security

# Topics



- Introduction (why cryptography?)
- Cryptography in the past: historic ciphers
- Cryptography today: formalizing security
- Security against eavesdroppers: achieving perfect secrecy
- Drawbacks of perfect secrecy, computational security
- Building blocks of cryptography: pseudorandom generators, pseudorandom functions

# Topics



- Introduction (why cryptography?)
- Cryptography in the past: historic ciphers
- Cryptography today: formalizing security
- Security against eavesdroppers: achieving perfect secrecy
- Drawbacks of perfect secrecy, computational security
- Building blocks of cryptography: pseudorandom generators, pseudorandom functions
- Security against stronger attacks

# Topics



- Introduction (why cryptography?)
- Cryptography in the past: historic ciphers
- Cryptography today: formalizing security
- Security against eavesdroppers: achieving perfect secrecy
- Drawbacks of perfect secrecy, computational security
- Building blocks of cryptography: pseudorandom generators, pseudorandom functions
- Security against stronger attacks
- Stream ciphers and block ciphers






# Topics












- Introduction (why cryptography?)
- Cryptography in the past: historic ciphers
- Cryptography today: formalizing security
- Security against eavesdroppers: achieving perfect secrecy
- Drawbacks of perfect secrecy, computational security
- Building blocks of cryptography: pseudorandom generators, pseudorandom functions
- Security against stronger attacks
- Stream ciphers and block ciphers
- Integrity and authenticated encryption

# Topics



- Introduction (why cryptography?)
- Cryptography in the past: historic ciphers
- Cryptography today: formalizing security 
- Security against eavesdroppers: achieving perfect secrecy 
- Drawbacks of perfect secrecy, computational security 
- Building blocks of cryptography: pseudorandom generators, pseudorandom functions 
- Security against stronger attacks 
- Stream ciphers and block ciphers
- Integrity and authenticated encryption

# Topics

- Introduction (why cryptography?) 
- Cryptography in the past: historic ciphers 
- Cryptography today: formalizing security 
- Security against eavesdroppers: achieving perfect secrecy 
- Drawbacks of perfect secrecy, computational security 
- Building blocks of cryptography: pseudorandom generators, pseudorandom functions 
- Security against stronger attacks 
- Stream ciphers and block ciphers 
- Integrity and authenticated encryption 

# Topics

- Introduction (why cryptography?)
- Cryptography in the past: historic ciphers
- Cryptography today: formalizing security
- Security against eavesdroppers: achieving perfect secrecy
- Drawbacks of perfect secrecy, computational security
- Building blocks of cryptography: pseudorandom generators, pseudorandom functions
- Security against stronger attacks
- Stream ciphers and block ciphers
- Integrity and authenticated encryption



**Private-key  
cryptography**

# Topics

- Hash functions
- Key distribution



At the “boundary” between private-key and public-key cryptography

# Topics

- Hash functions
  - Key distribution
  - Public-key cryptography, Hybrid cryptography
- } At the “boundary” between private-key and public-key cryptography

# Topics

- Hash functions
  - Key distribution
  - Public-key cryptography, Hybrid cryptography
  - Digital signatures
- At the “boundary” between private-key and public-key cryptography

# Topics

- Hash functions
- Key distribution



At the “boundary” between private-key and public-key cryptography

- Public-key cryptography, Hybrid cryptography
- Digital signatures
- Digital certificates, SSL/TLS



Public-key cryptography

# Topics

- Hash functions
- Key distribution




At the “boundary” between private-key and public-key cryptography


- Public-key cryptography, Hybrid cryptography
- Digital signatures
- Digital certificates, SSL/TLS
- Advanced applications



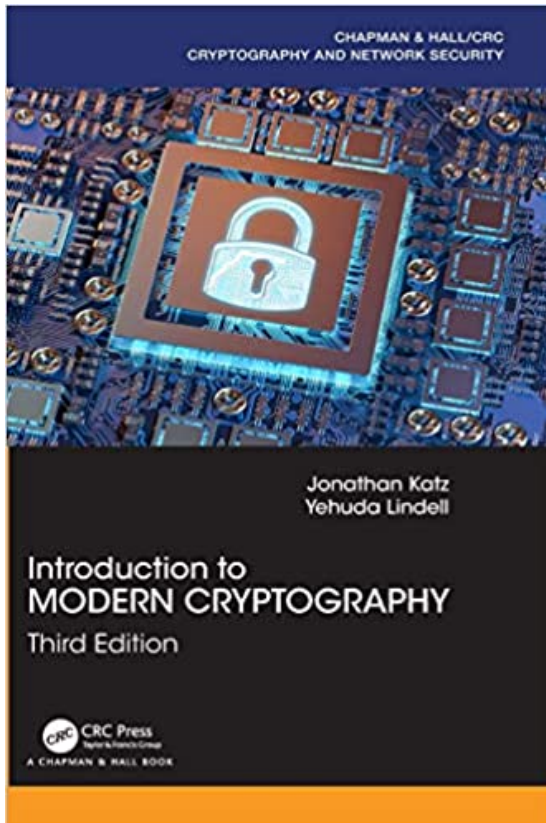
Public-key cryptography

— secret sharing 

— multiparty computation 

— zero-knowledge proofs 

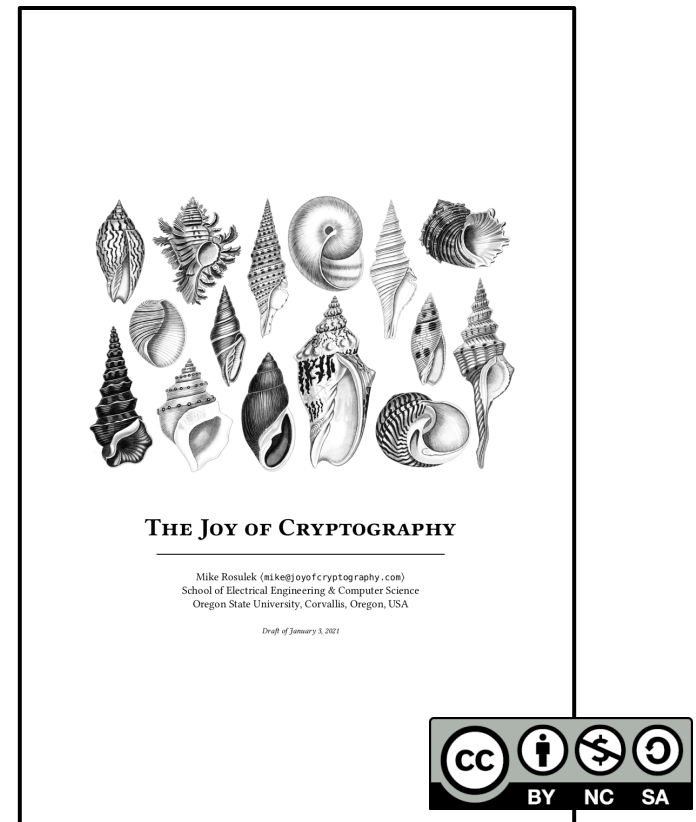
# Books



Introduction to Modern Cryptography

Jonathan Katz, Yehuda Lindell

ISBN: 978-0815354369

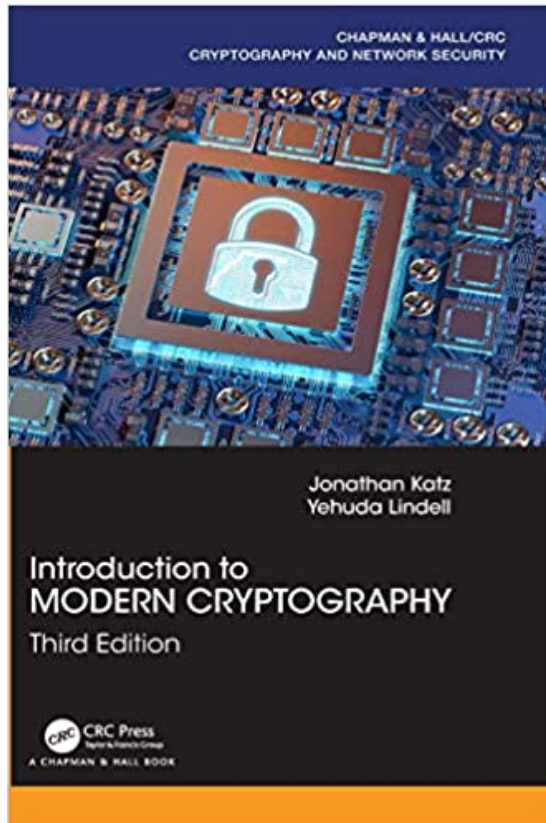


The Joy Of Cryptography

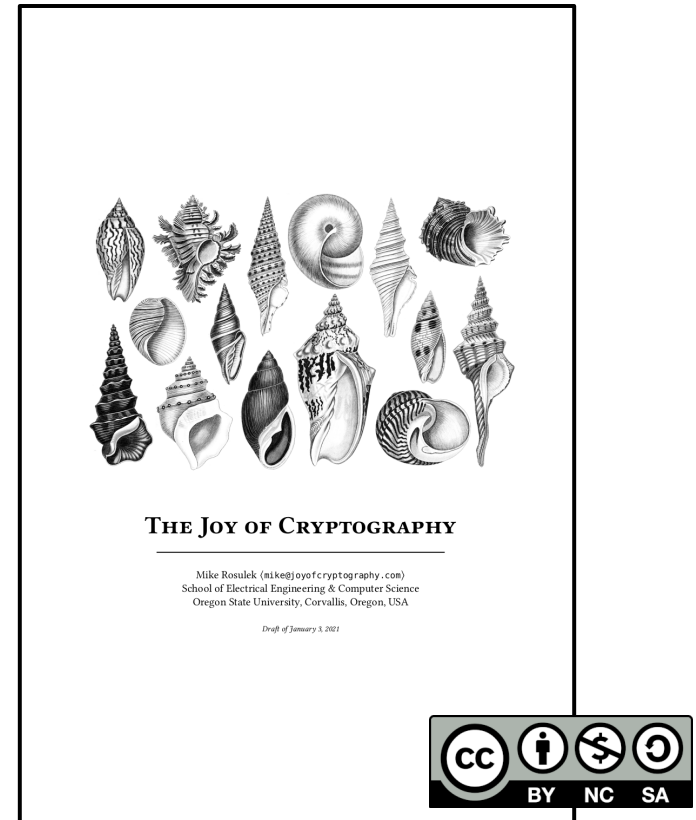
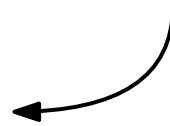
Mike Rosulek

<https://joyofcryptography.com/>

# Books



Almost all course material can be found in this book  
We will use the same notation



The Joy Of Cryptography

Mike Rosulek

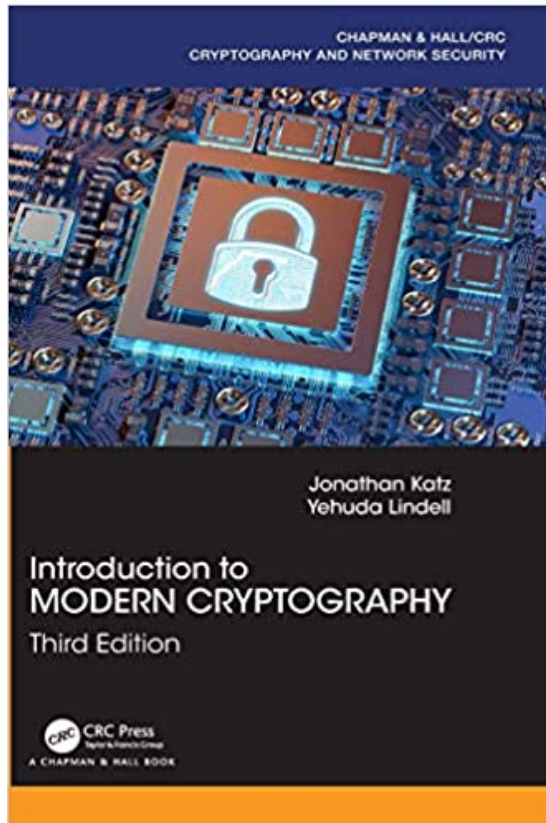
<https://joyofcryptography.com/>

Introduction to Modern Cryptography

Jonathan Katz, Yehuda Lindell

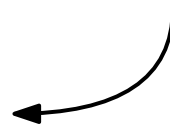
ISBN: 978-0815354369

# Books

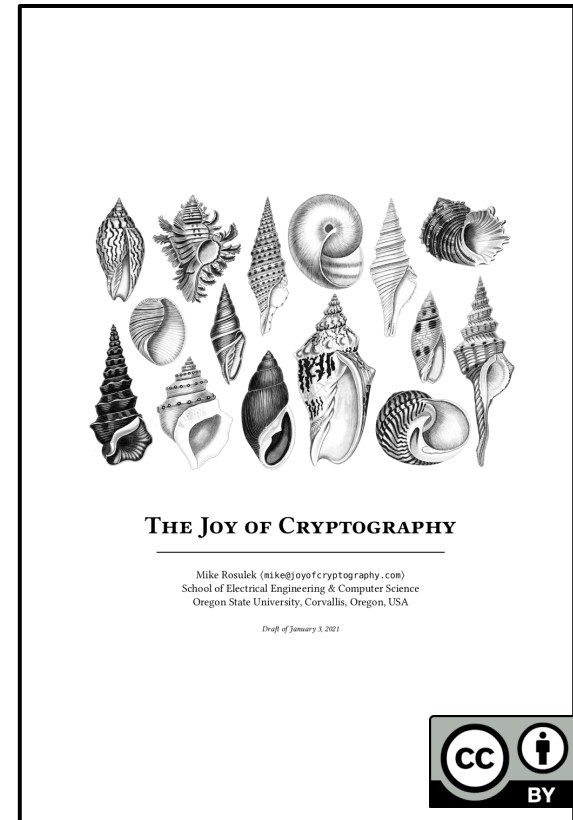


Introduction to Modern Cryptography  
Jonathan Katz, Yehuda Lindell  
ISBN: 978-0815354369

Almost all course material can be found in this book  
We will use the same notation



Freely Available  
Uses different notation.

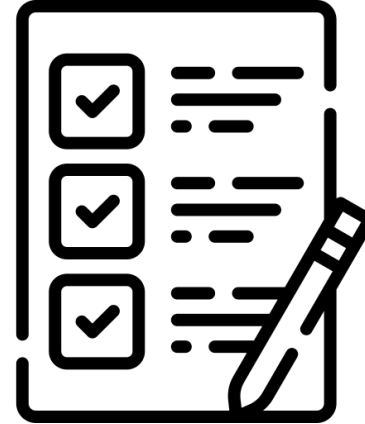


The Joy Of Cryptography  
Mike Rosulek  
<https://joyofcryptography.com/>

# Exams

Written exam:

- Questions on the theoretical concepts (e.g., security definitions)
- Exercises (e.g., prove security, carry out an attack, etc. . . )

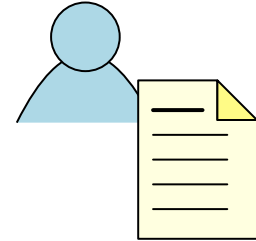


Some advanced applications

# Secret Sharing

Imagine some sensitive information that is kept by a single agent

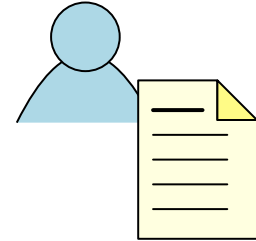
- A master encryption key
- Your bitcoin wallet
- Nuclear codes
- ...



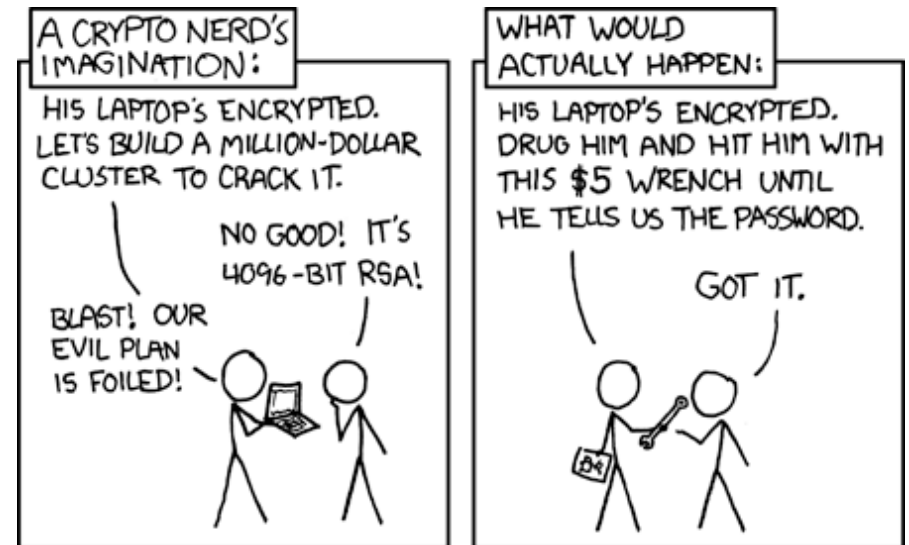
# Secret Sharing

Imagine some sensitive information that is kept by a single agent

- A master encryption key
- Your bitcoin wallet
- Nuclear codes
- ...



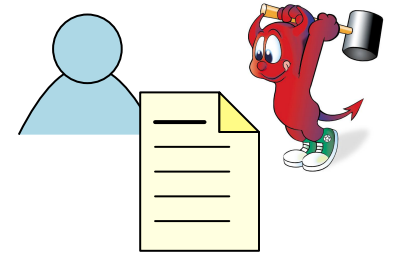
**Single point of failure!**



# Secret Sharing

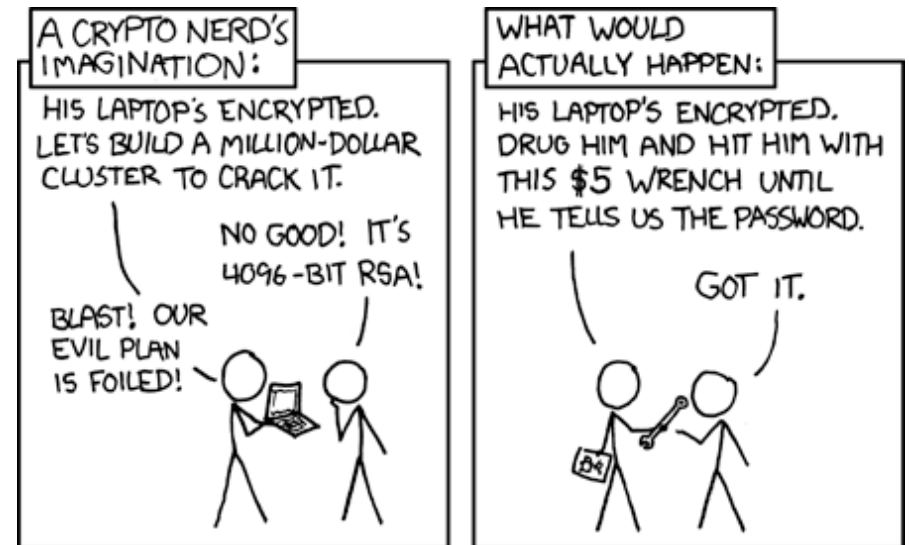
Imagine some sensitive information that is kept by a single agent

- A master encryption key
- Your bitcoin wallet
- Nuclear codes
- ...



## Single point of failure!

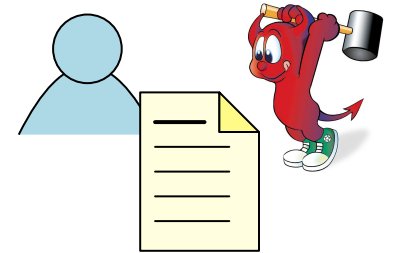
An attacker can compromise one machine and steal the sensitive information



# Secret Sharing

Imagine some sensitive information that is kept by a single agent

- A master encryption key
- Your bitcoin wallet
- Nuclear codes
- ...



## Single point of failure!

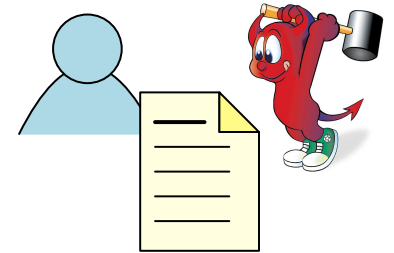
An attacker can compromise one machine and steal the sensitive information

**Idea:** Share the information across several agents

# Secret Sharing

Imagine some sensitive information that is kept by a single agent

- A master encryption key
- Your bitcoin wallet
- Nuclear codes
- ...



## Single point of failure!

An attacker can compromise one machine and steal the sensitive information

**Idea:** Share the information across several agents

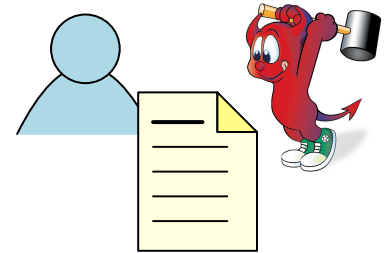
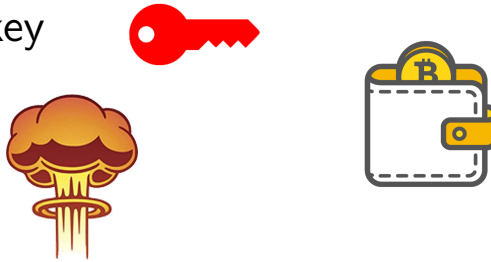


“Magic box”

# Secret Sharing

Imagine some sensitive information that is kept by a single agent

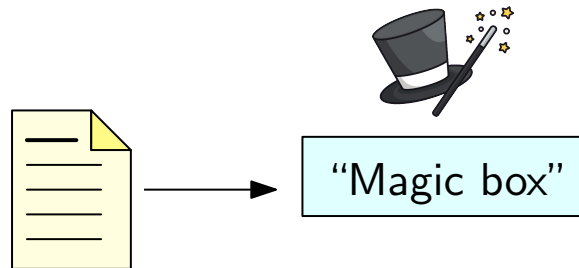
- A master encryption key
- Your bitcoin wallet
- Nuclear codes
- ...



## Single point of failure!

An attacker can compromise one machine and steal the sensitive information

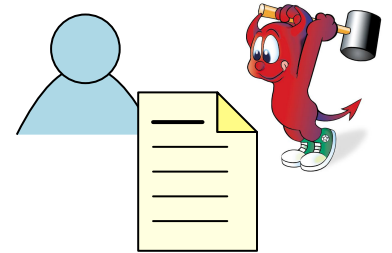
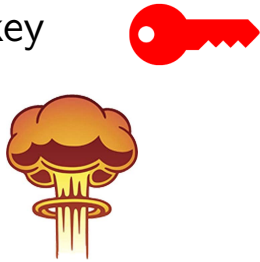
**Idea:** Share the information across several agents



# Secret Sharing

Imagine some sensitive information that is kept by a single agent

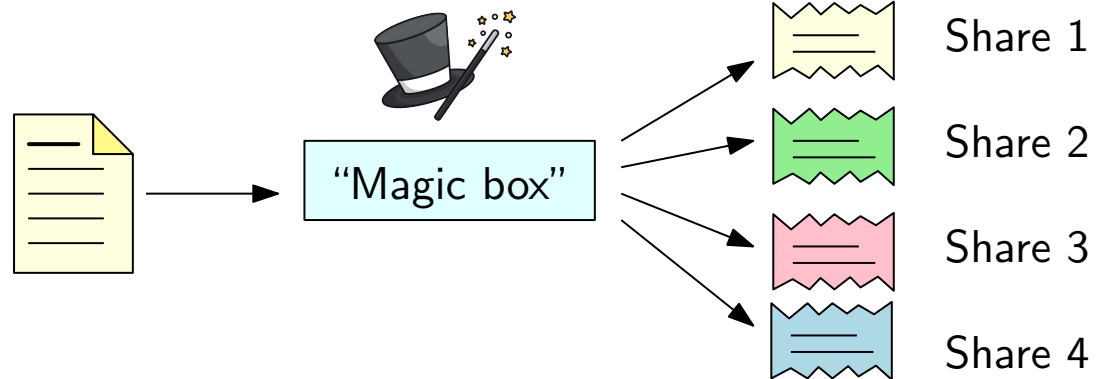
- A master encryption key
- Your bitcoin wallet
- Nuclear codes
- ...



## Single point of failure!

An attacker can compromise one machine and steal the sensitive information

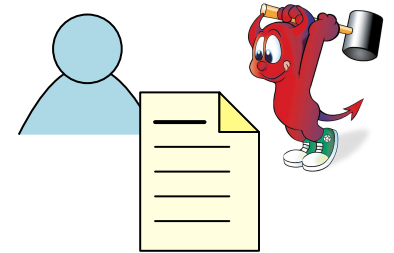
**Idea:** Share the information across several agents



# Secret Sharing

Imagine some sensitive information that is kept by a single agent

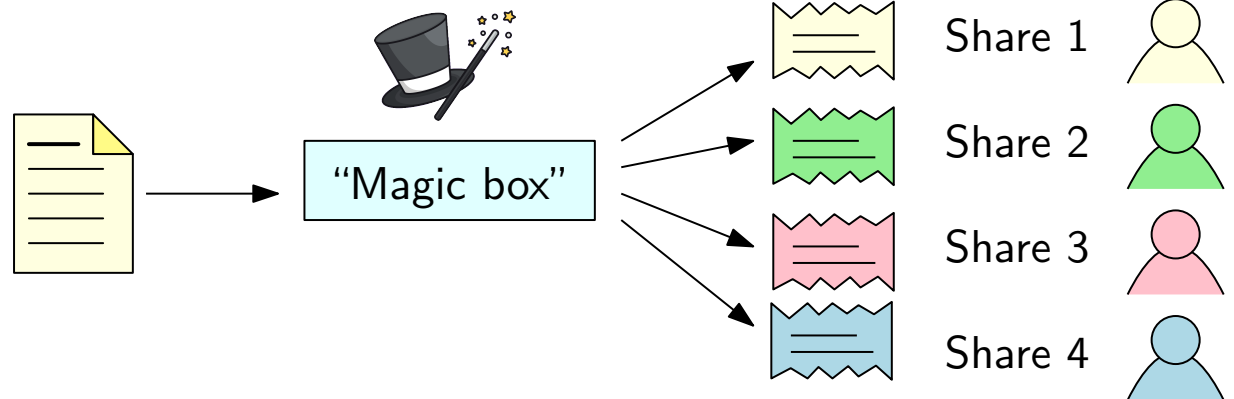
- A master encryption key
- Your bitcoin wallet
- Nuclear codes
- ...



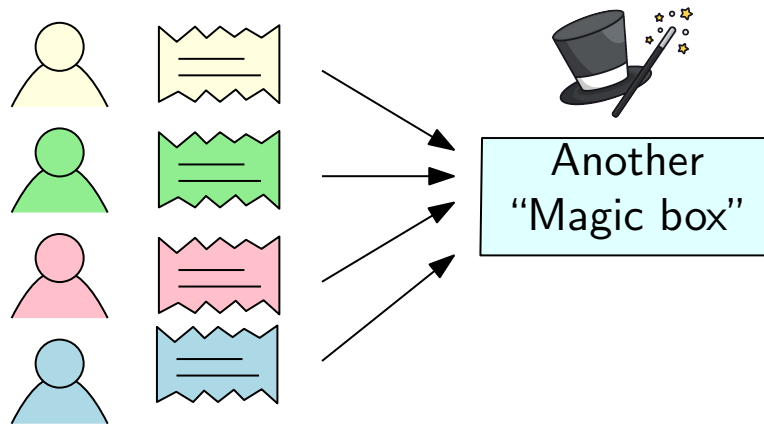
## Single point of failure!

An attacker can compromise one machine and steal the sensitive information

**Idea:** Share the information across several agents



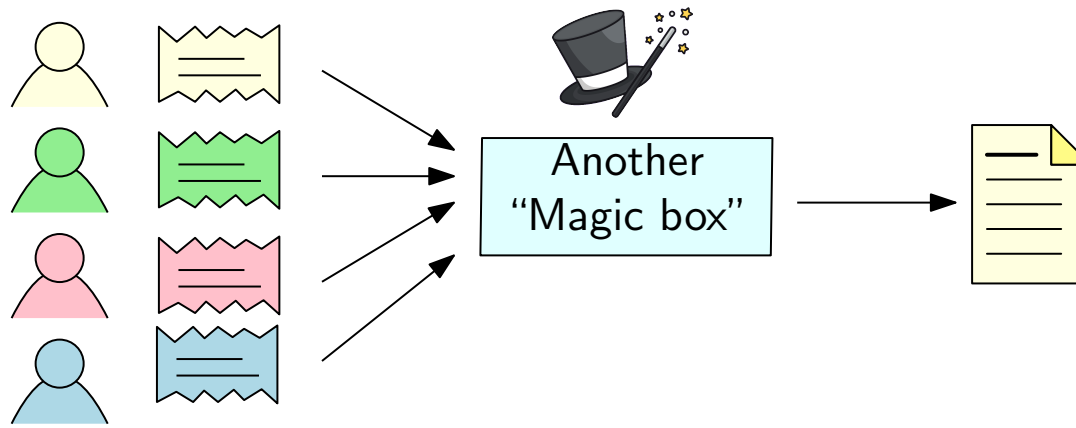
# Secret Sharing



## Idea:

- The shares of all agents can be used to reconstruct the secret

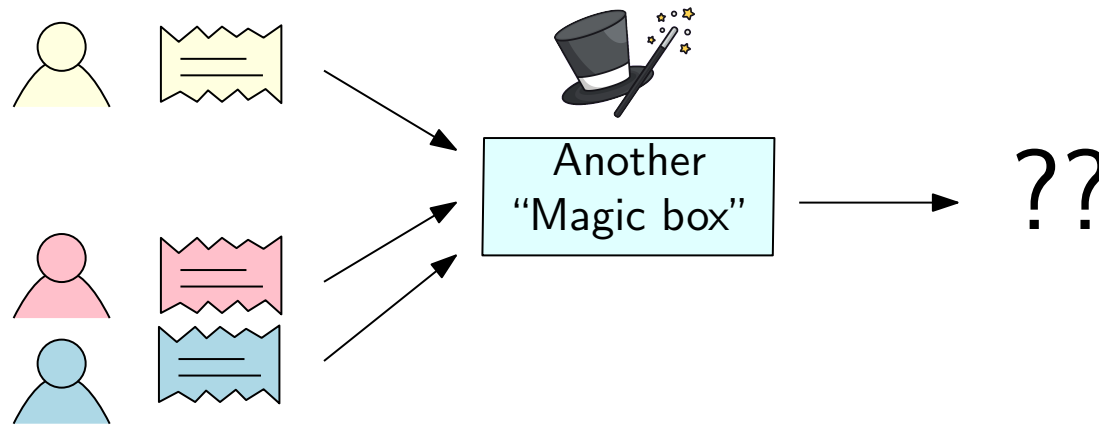
# Secret Sharing



## Idea:

- The shares of all agents can be used to reconstruct the secret

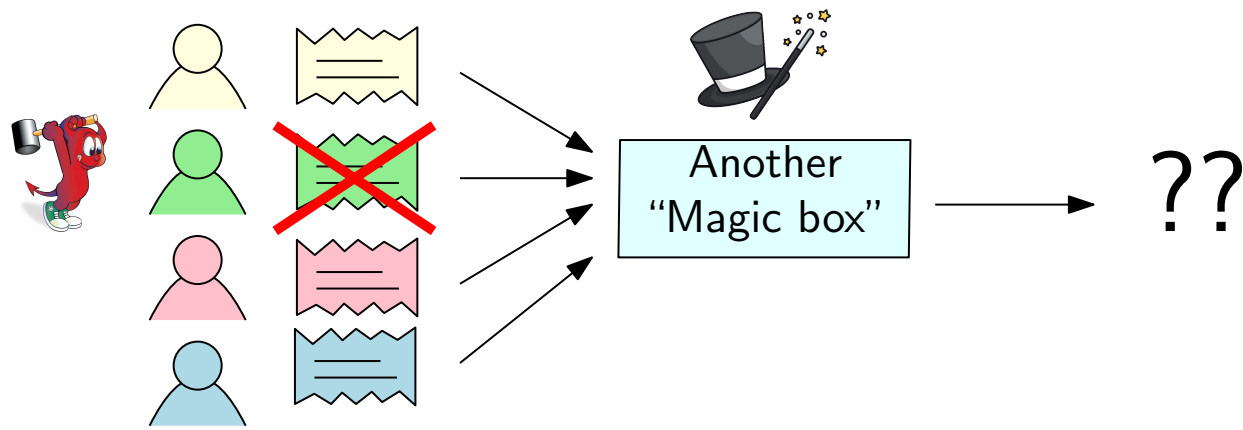
# Secret Sharing



## Idea:

- The shares of all agents can be used to reconstruct the secret
- The shares of any subset of agents look random and convey no information about the secret

# Secret Sharing



## Idea:

- The shares of all agents can be used to reconstruct the secret
- The shares of any subset of agents look random and convey no information about the secret

What if the adversary destroys a share?

# Secret Sharing

What if the adversary destroys a share?

**Idea:**

- If there are  $n$  agents, any subset of at least  $t$  agents must be able to recover the secret.

# Secret Sharing

What if the adversary destroys a share?

**Idea:**

- If there are  $n$  agents, any subset of at least  $t$  agents must be able to recover the secret.
- Any subset of  $< t$  agent must not be able to gain any information about the secret

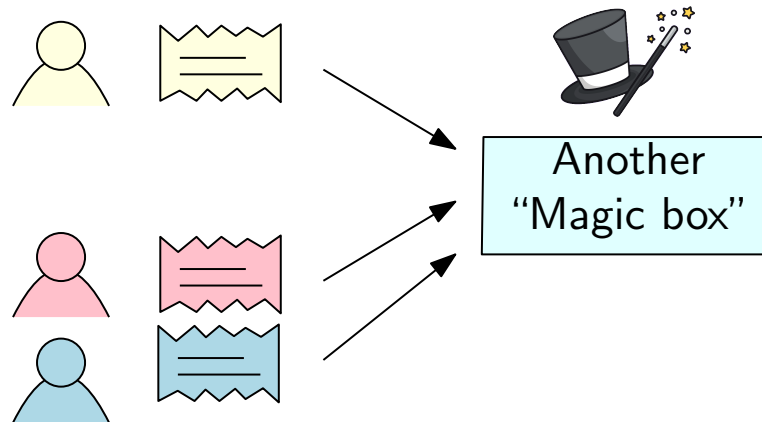
# Secret Sharing

What if the adversary destroys a share?

## Idea:

- If there are  $n$  agents, any subset of at least  $t$  agents must be able to recover the secret.
- Any subset of  $< t$  agent must not be able to gain any information about the secret

**Example:**  $n = 4, t = 3$



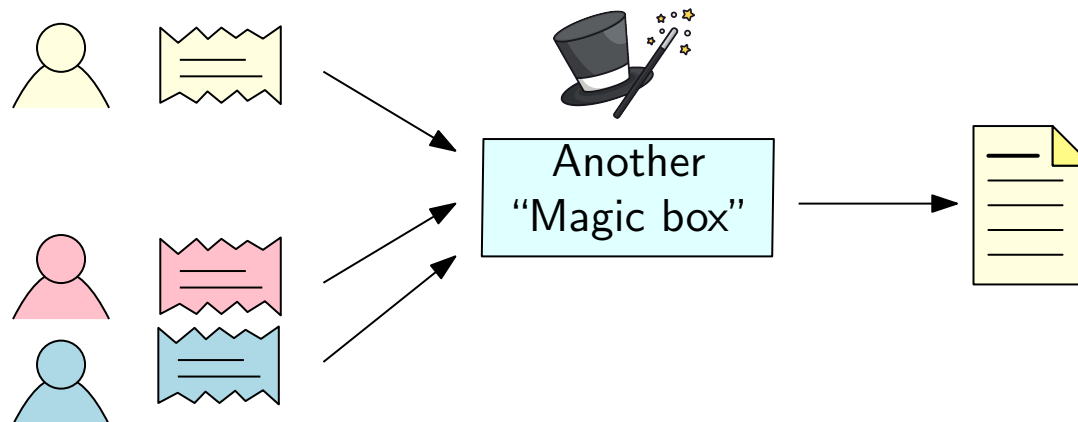
# Secret Sharing

What if the adversary destroys a share?

## Idea:

- If there are  $n$  agents, any subset of at least  $t$  agents must be able to recover the secret.
- Any subset of  $< t$  agent must not be able to gain any information about the secret

**Example:**  $n = 4, t = 3$



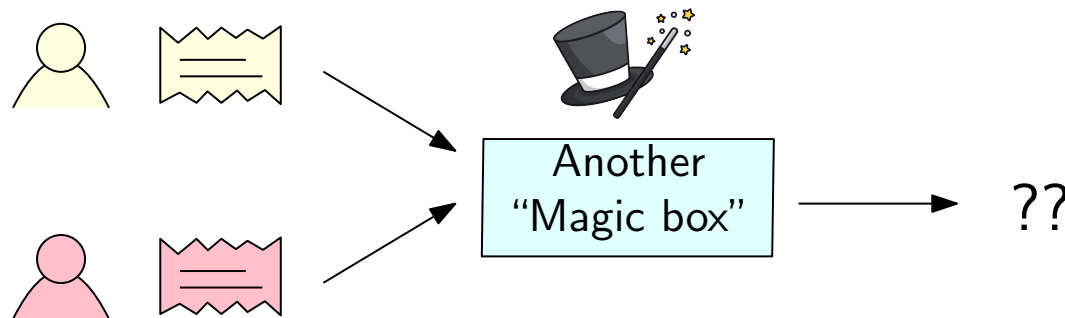
# Secret Sharing

What if the adversary destroys a share?

## Idea:

- If there are  $n$  agents, any subset of at least  $t$  agents must be able to recover the secret.
- Any subset of  $< t$  agent must not be able to gain any information about the secret

**Example:**  $n = 4, t = 3$



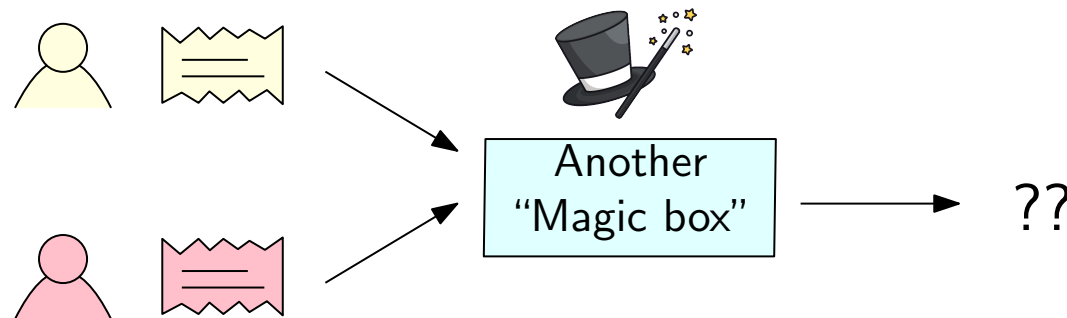
# Secret Sharing

What if the adversary destroys a share?

## Idea:

- If there are  $n$  agents, any subset of at least  $t$  agents must be able to recover the secret.
- Any subset of  $< t$  agent must not be able to gain any information about the secret

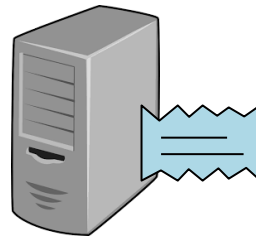
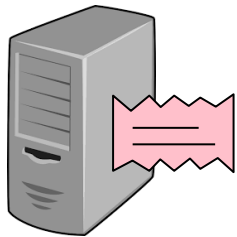
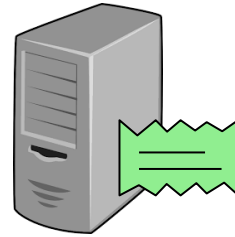
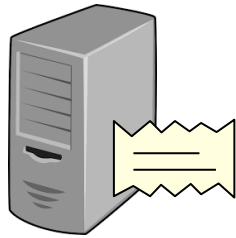
**Example:**  $n = 4, t = 3$



**$t$ -out-of- $n$  threshold secret-sharing scheme**

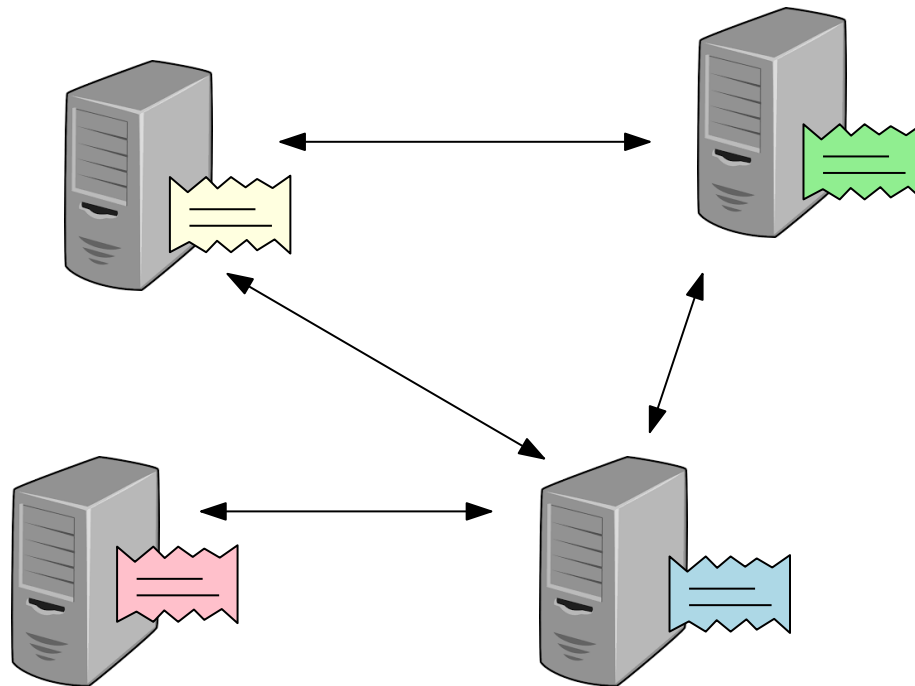
# Secret Sharing

- Agents can be different servers connected over the Internet



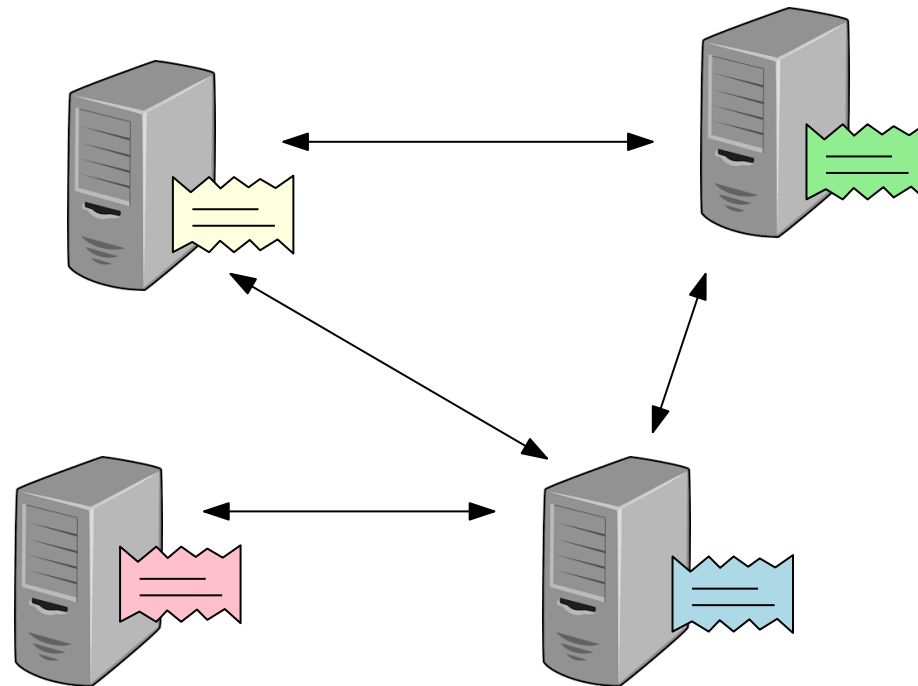
# Secret Sharing

- Agents can be different servers connected over the Internet
- The “magic boxes” can be distributed algorithms



# Secret Sharing

- Agents can be different servers connected over the Internet
- The “magic boxes” can be distributed algorithms



- The system remains secure if  $< t$  servers are compromised



# DNSSEC

The Domain Name System is the system responsible for converting human-readable domain names into IP addresses



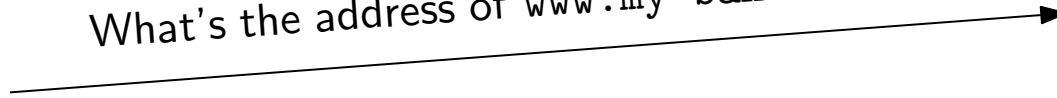
DNS Server

# DNSSEC

The Domain Name System is the system responsible for converting human-readable domain names into IP addresses



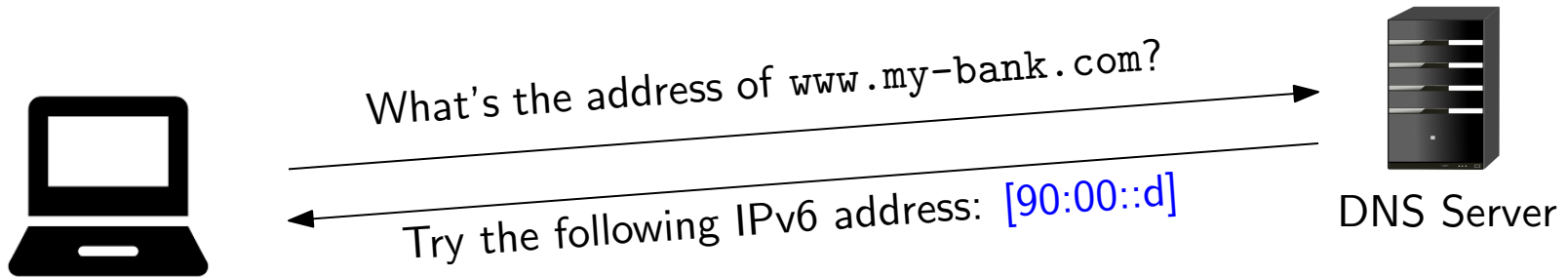
What's the address of `www.my-bank.com`?



DNS Server

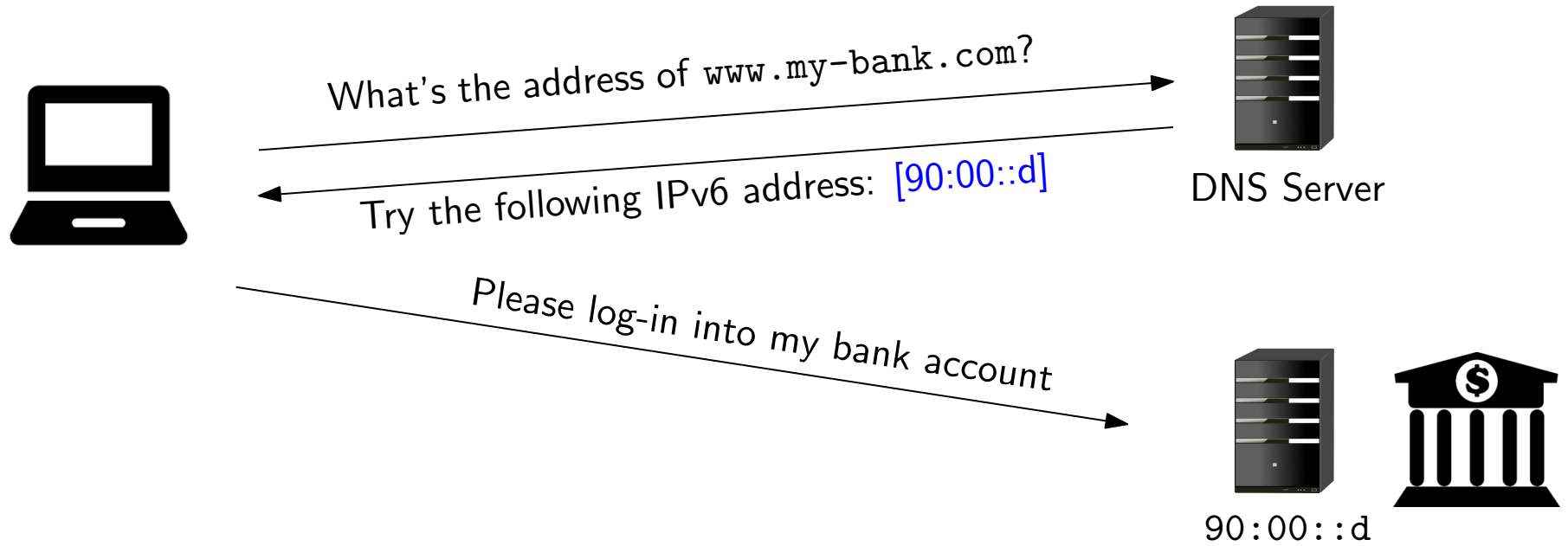
# DNSSEC

The Domain Name System is the system responsible for converting human-readable domain names into IP addresses



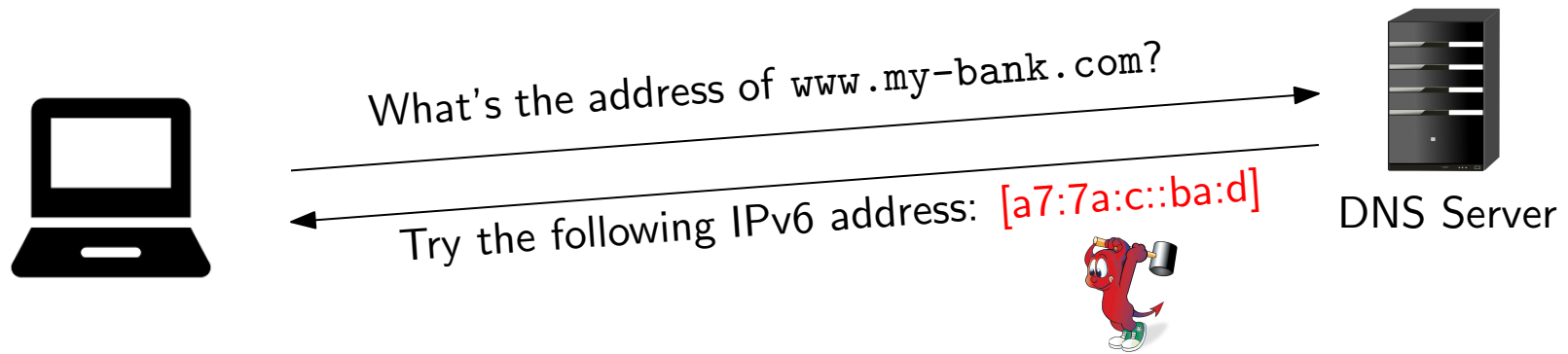
# DNSSEC

The Domain Name System is the system responsible for converting human-readable domain names into IP addresses



# DNSSEC

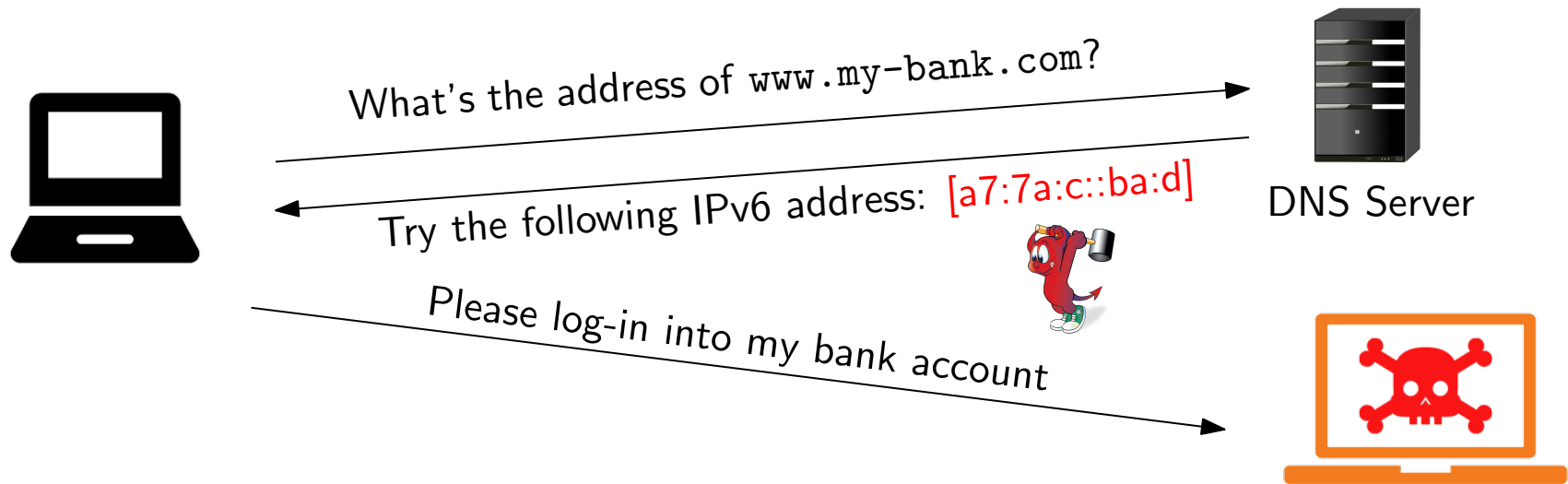
The Domain Name System is the system responsible for converting human-readable domain names into IP addresses



An attacker can tamper with DNS responses and convince the client that `www.my-bank.com` resides on a malicious server

# DNSSEC

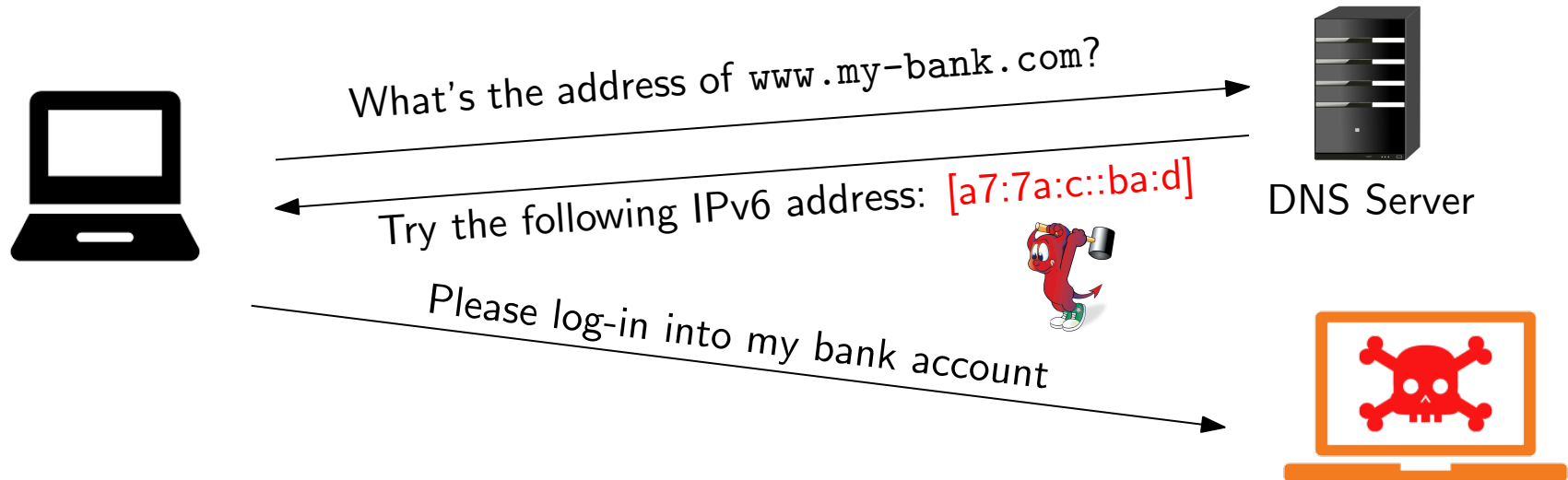
The Domain Name System is the system responsible for converting human-readable domain names into IP addresses



An attacker can tamper with DNS responses and convince the client that `www.my-bank.com` resides on a malicious server

# DNSSEC

The Domain Name System is the system responsible for converting human-readable domain names into IP addresses

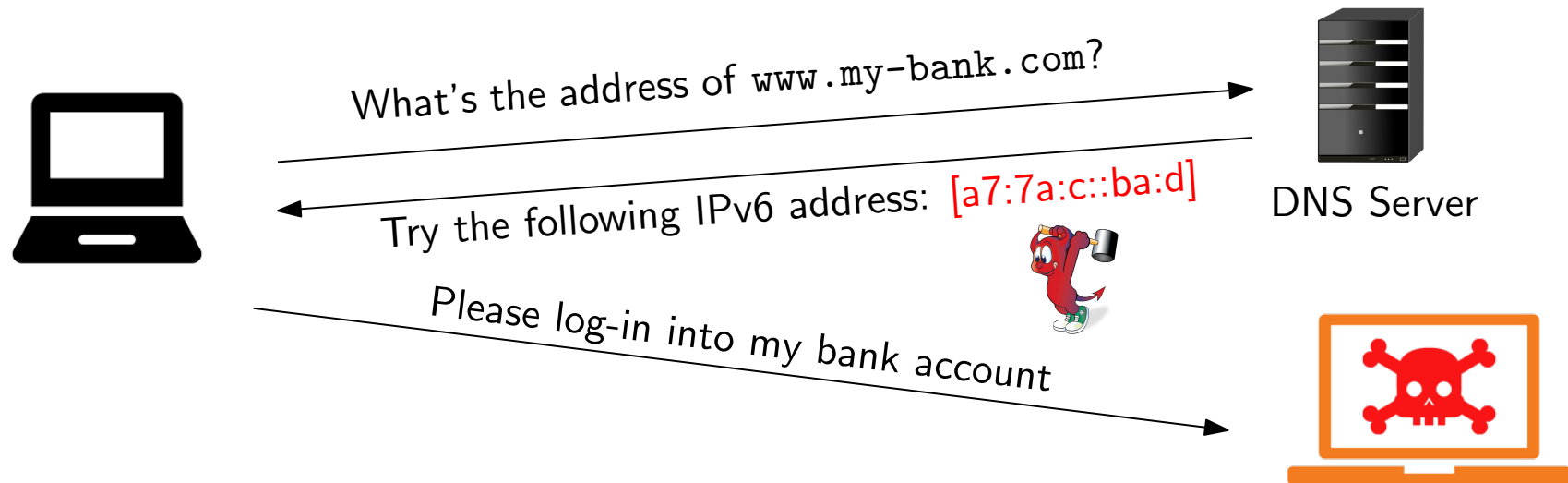


An attacker can tamper with DNS responses and convince the client that `www.my-bank.com` resides on a malicious server

To prevent this, DNSSEC is used to authenticate DNS mappings

# DNSSEC

The Domain Name System is the system responsible for converting human-readable domain names into IP addresses



An attacker can tamper with DNS responses and convince the client that `www.my-bank.com` resides on a malicious server

To prevent this, DNSSEC is used to authenticate DNS mappings

Who can be trusted with the master cryptographic keys to the system?

# DNSSEC

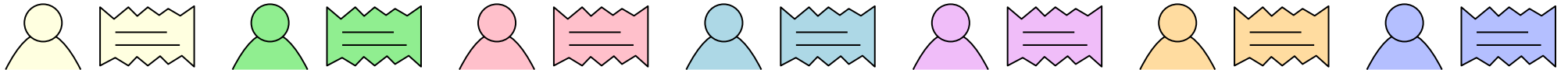
DNSSEC is managed by the Internet Corporation for Assigned Names and Numbers (ICANN)

The master key is split into 7 pieces and distributed on smart cards to 7 geographically diverse people

*At least five key-holding members of this fellowship would have to meet at a secure data center in the United States to reboot [DNSSEC] in case of a very unlikely system collapse.*

*“If you round up five of these guys, they can decrypt [the root key] should the West Coast fall in the water and the East Coast get hit by a nuclear bomb”*

*— Richard Lamb, program manager for DNSSEC at ICANN.*



# DNSSEC

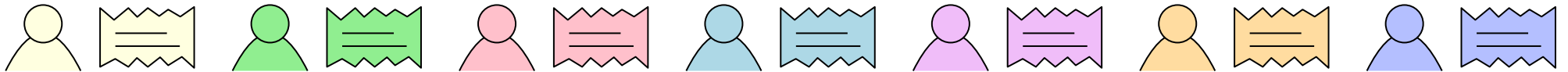
DNSSEC is managed by the Internet Corporation for Assigned Names and Numbers (ICANN)

The master key is split into 7 pieces and distributed on smart cards to 7 geographically diverse people

*At least five key-holding members of this fellowship would have to meet at a secure data center in the United States to reboot [DNSSEC] in case of a very unlikely system collapse.*

*“If you round up five of these guys, they can decrypt [the root key] should the West Coast fall in the water and the East Coast get hit by a nuclear bomb”*

*— Richard Lamb, program manager for DNSSEC at ICANN.*

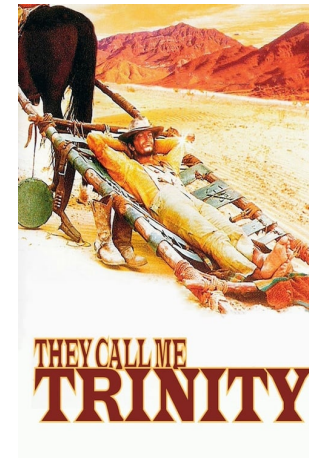
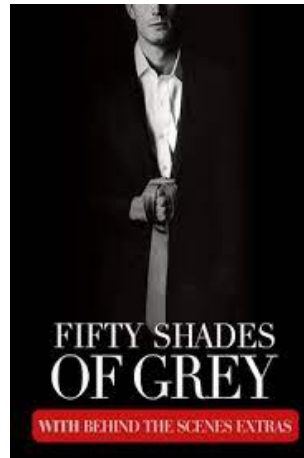
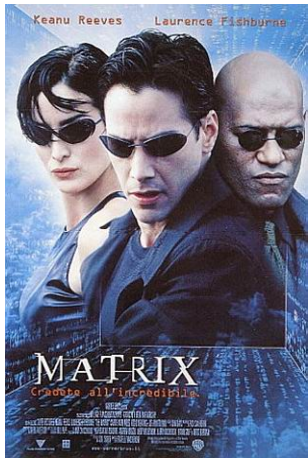


$n = 7$        $t = 5$

5-out-of-7 threshold secret-sharing scheme

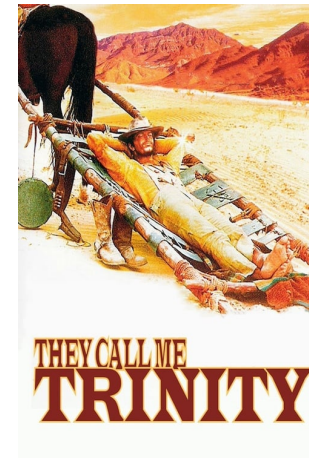
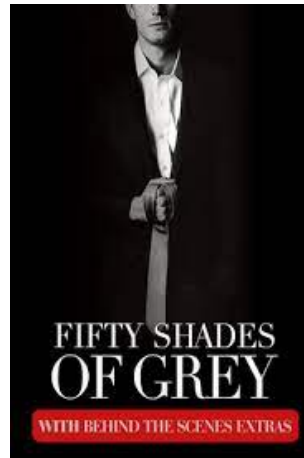
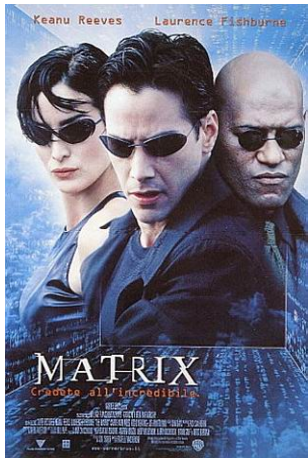
# Choosing a movie with secret preferences

Alice and Bob want to decide on a movie to watch



# Choosing a movie with secret preferences

Alice and Bob want to decide on a movie to watch

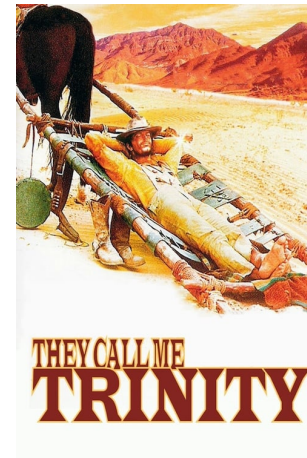
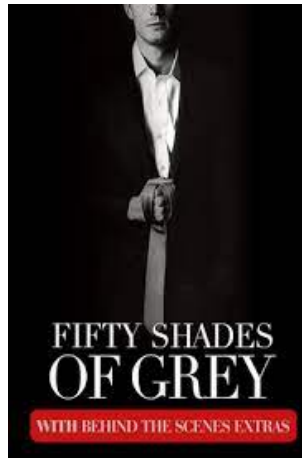
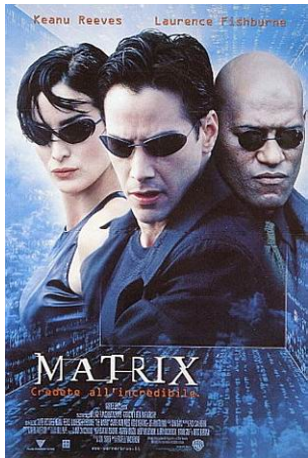


Each of Alice and Bob has their own preferences...

However, they don't want to reveal that they like a movie unless the other person also likes it

# Choosing a movie with secret preferences

Alice and Bob want to decide on a movie to watch



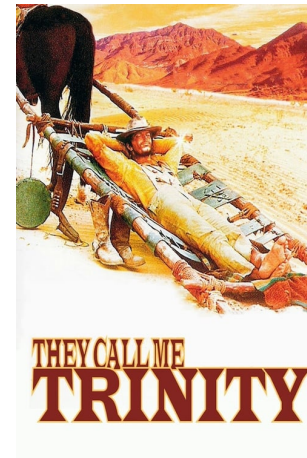
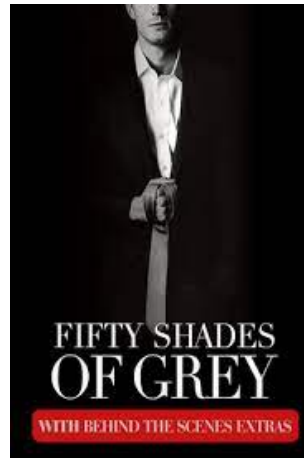
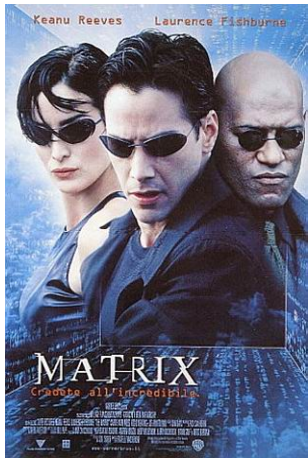
Each of Alice and Bob has their own preferences...

However, they don't want to reveal that they like a movie unless the other person also likes it

- They vote on one movie at a time until they find a movie that they both like
- Voting is done using a secure **2-party computation protocol**

# Choosing a movie with secret preferences

Alice and Bob want to decide on a movie to watch



Each of Alice and Bob has their own preferences...

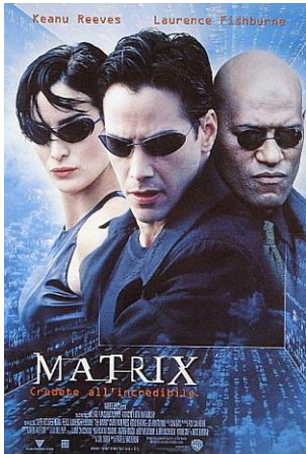
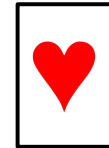
However, they don't want to reveal that they like a movie unless the other person also likes it

- They vote on one movie at a time until they find a movie that they both like
- Voting is done using a secure **2-party computation protocol**

Alice and Bob are **honest** (they follow the protocol) but **curious**

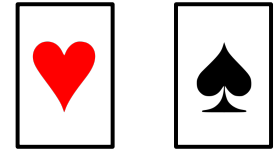
# Choosing a movie with secret preferences

They use some cards with two different faces, say hearts and spades

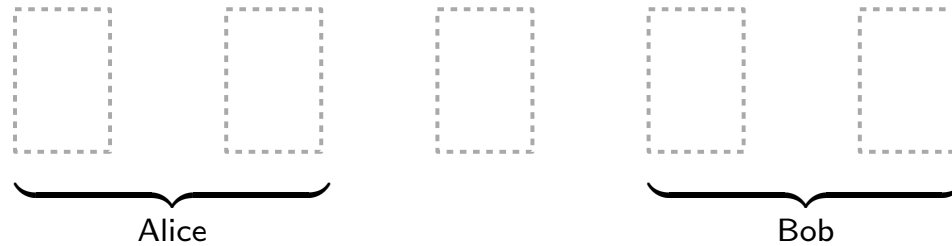
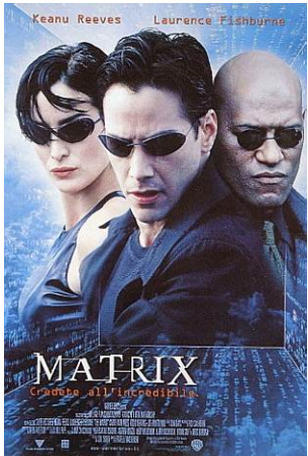


# Choosing a movie with secret preferences

They use some cards with two different faces, say hearts and spades

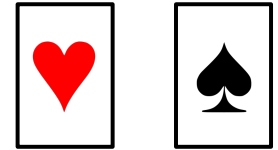


They will place 5 cards in a row on a table



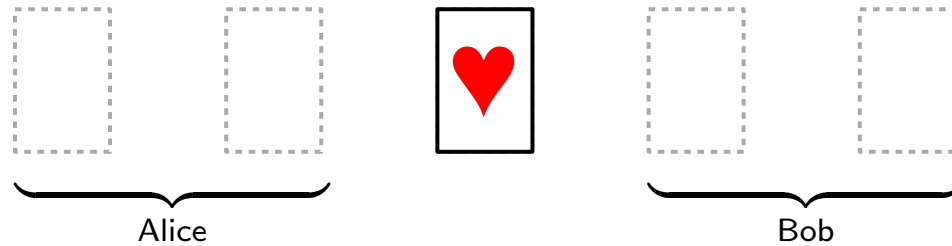
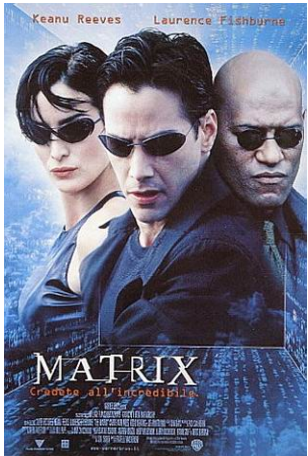
# Choosing a movie with secret preferences

They use some cards with two different faces, say hearts and spades



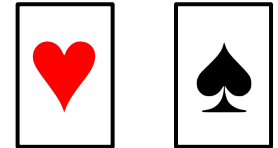
They will place 5 cards in a row on a table

The middle card is face up and is always a hearts card



# Choosing a movie with secret preferences

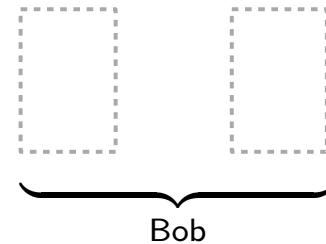
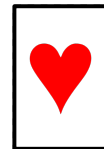
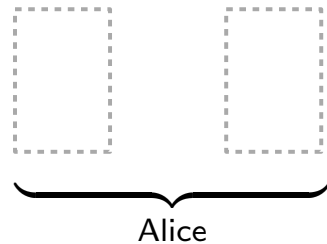
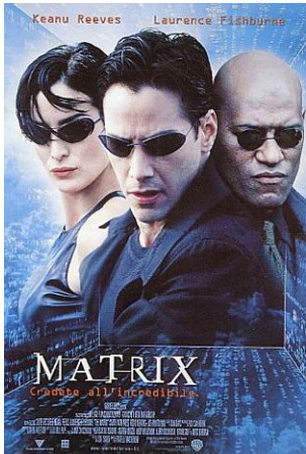
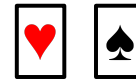
They use some cards with two different faces, say hearts and spades





They will place 5 cards in a row on a table


The middle card is face up and is always a hearts card

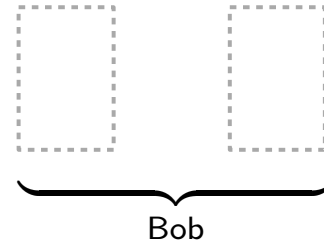
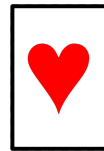
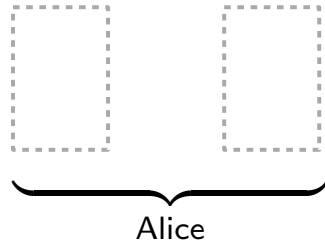
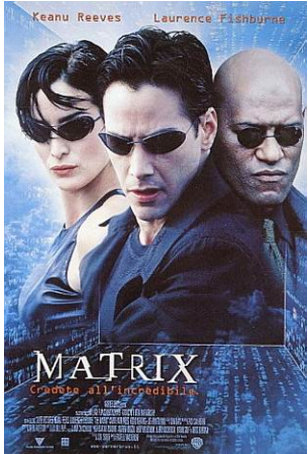
Each of Alice and Bob has a hearts card and a spades card





# Choosing a movie with secret preferences

If Alice likes the movie, she will place her two cards **face down** in the order  

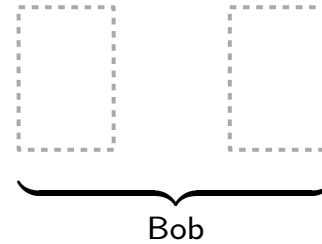
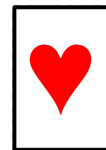
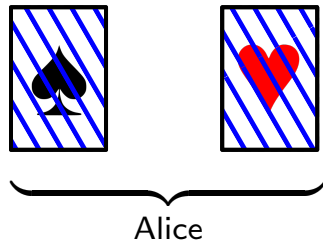
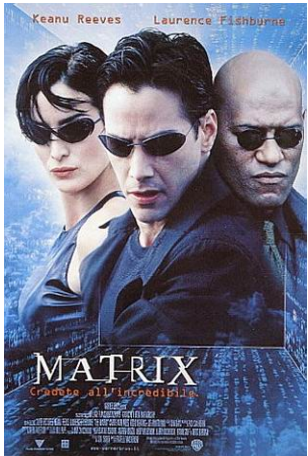
Otherwise she will place her two cards **face down** in the order  





# Choosing a movie with secret preferences



If Alice likes the movie, she will place her two cards **face down** in the order  

Otherwise she will place her two cards **face down** in the order  




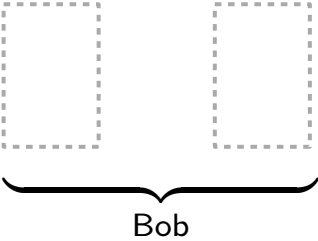
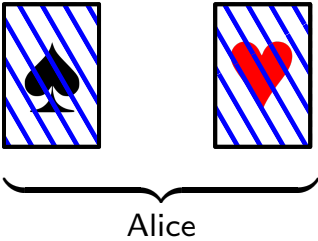
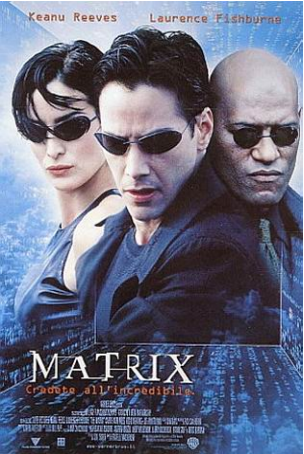
# Choosing a movie with secret preferences

If Alice likes the movie, she will place her two cards **face down** in the order  



Otherwise she will place her two cards **face down** in the order  



If Bob likes the movie, he will place his two cards **face down** in the order  

Otherwise he will place his two cards **face down** in the order  





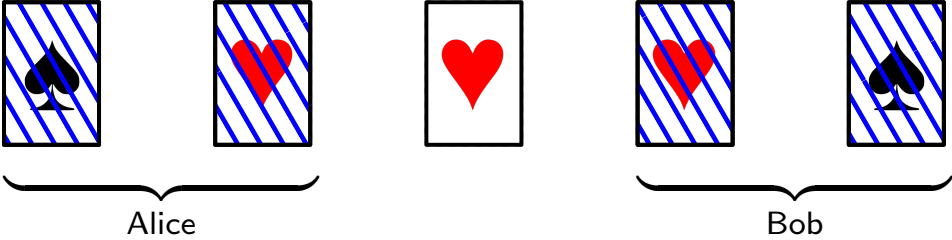
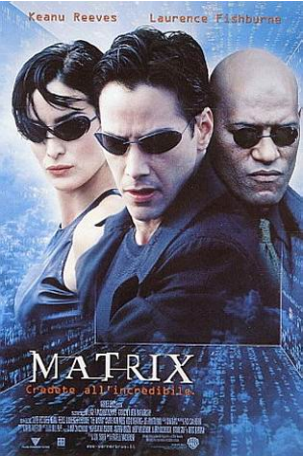
# Choosing a movie with secret preferences

If Alice likes the movie, she will place her two cards **face down** in the order  

Otherwise she will place her two cards **face down** in the order  

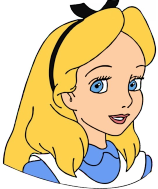
If Bob likes the movie, he will place his two cards **face down** in the order  

Otherwise he will place his two cards **face down** in the order  



# Choosing a movie with secret preferences

Possible configurations:



Yes

Yes



Yes

No



No

Yes



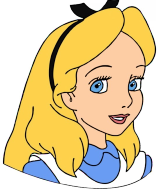
No

No



# Choosing a movie with secret preferences

Possible configurations:



Yes

Yes



Yes

No



No

Yes



No

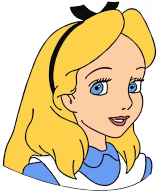
No



Rotations

# Choosing a movie with secret preferences

Possible configurations:



Yes

Yes



Yes

No



No

Yes



No

No

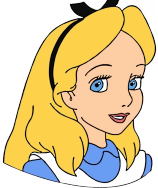


Rotations

There are three hearts in a row if and only if both Alice and Bob like the movie

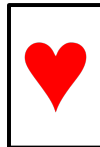
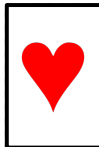
# Choosing a movie with secret preferences

Possible configurations:



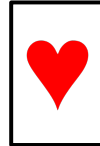
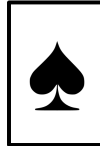
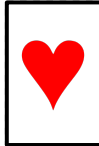
Yes

Yes



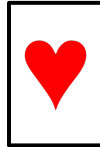
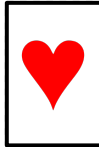
Yes

No



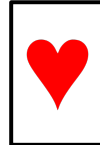
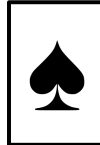
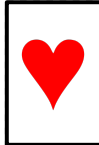
No

Yes



No

No

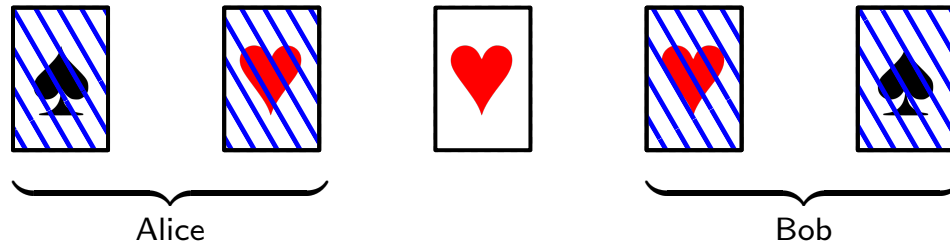
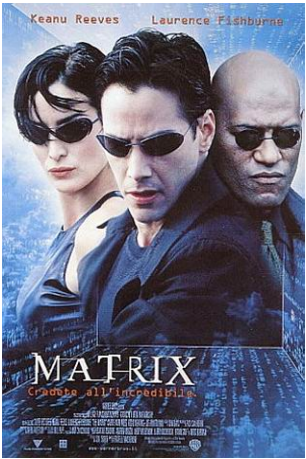


Rotations

**There are three hearts in a row if and only if both Alice and Bob like the movie**

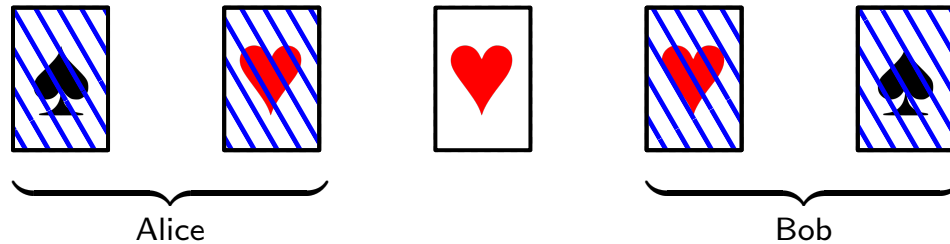
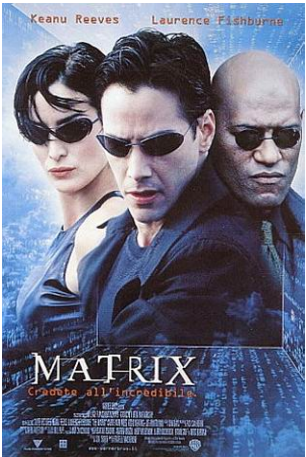
This holds (in a modular sense) even if any rotation of the cards is considered!

# Choosing a movie with secret preferences



Alice and Bob pile up the cards, **all face down**, ensuring that their order is preserved

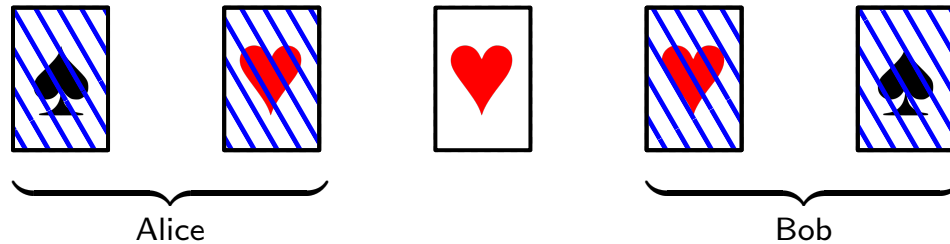
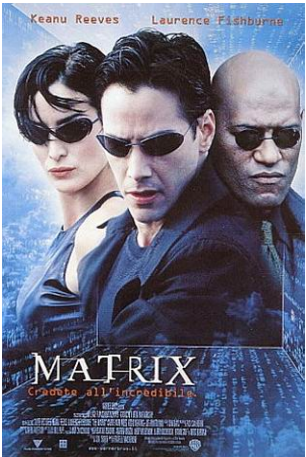
# Choosing a movie with secret preferences



Alice and Bob pile up the cards, **all face down**, ensuring that their order is preserved

Alice and Bob take turns cutting the deck (i.e., they perform a rotation)

# Choosing a movie with secret preferences

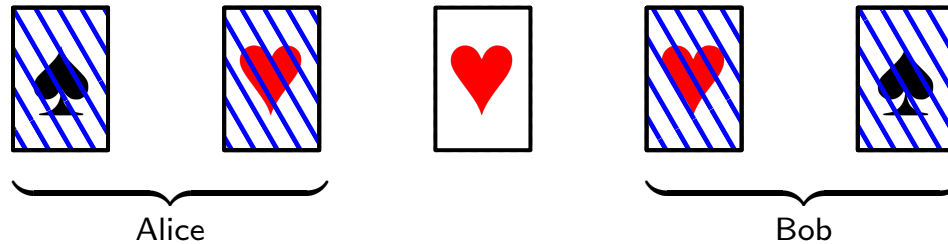
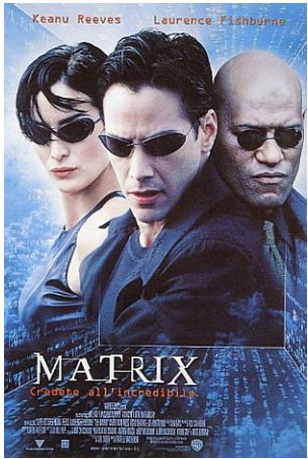


Alice and Bob pile up the cards, **all face down**, ensuring that their order is preserved

Alice and Bob take turns cutting the deck (i.e., they perform a rotation)

Alice and Bob reveal the cards and watch the movie if and only if there are 3 consecutive hearts (in a modular fashion), otherwise they “pass”

# Choosing a movie with secret preferences



Alice and Bob pile up the cards, **all face down**, ensuring that their order is preserved

Alice and Bob take turns cutting the deck (i.e., they perform a rotation)

Alice and Bob reveal the cards and watch the movie if and only if there are 3 consecutive hearts (in a modular fashion), otherwise they “pass”

**Observation:** If they end up not watching the movie, all possible “pass” configurations are equiprobable

# Secure Multiparty Computation

The previous scenario is a secure multiparty computation problem:

# Secure Multiparty Computation

The previous scenario is a secure multiparty computation problem:

- There are  $n$  agents (parties)  $1, 2, \dots, n$

# Secure Multiparty Computation

The previous scenario is a secure multiparty computation problem:

- There are  $n$  agents (parties)  $1, 2, \dots, n$
- The  $i$ -th agent has a secret information  $s_i$

# Secure Multiparty Computation

The previous scenario is a secure multiparty computation problem:

- There are  $n$  agents (parties)  $1, 2, \dots, n$
- The  $i$ -th agent has a secret information  $s_i$
- The agents wish to jointly compute some function  $f(s_1, s_2, \dots, s_n)$

# Secure Multiparty Computation

The previous scenario is a secure multiparty computation problem:

- There are  $n$  agents (parties)  $1, 2, \dots, n$
- The  $i$ -th agent has a secret information  $s_i$
- The agents wish to jointly compute some function  $f(s_1, s_2, \dots, s_n)$
- No information about  $s_1, s_2, \dots, s_n$  should be revealed

# Secure Multiparty Computation

The previous scenario is a secure multiparty computation problem:

- There are  $n$  agents (parties)  $1, 2, \dots, n$
- The  $i$ -th agent has a secret information  $s_i$
- The agents wish to jointly compute some function  $f(s_1, s_2, \dots, s_n)$
- No information about  $s_1, s_2, \dots, s_n$  should be revealed

In the movie selection problem:

- $n = 2$
- $s_1, s_2 \in \{\text{pass}, \text{watch}\}$
- $f(s_1, s_2) = \begin{cases} \text{watch} & \text{if } s_1 = s_2 = \text{watch} \\ \text{pass} & \text{otherwise} \end{cases}$

# Secure Multiparty Computation

The previous scenario is a secure multiparty computation problem:

- There are  $n$  agents (parties)  $1, 2, \dots, n$
- The  $i$ -th agent has a secret information  $s_i$
- The agents wish to jointly compute some function  $f(s_1, s_2, \dots, s_n)$
- No information about  $s_1, s_2, \dots, s_n$  should be revealed

In the movie selection problem:

- $n = 2$
- $s_1, s_2 \in \{0, 1\}$
- $f(s_1, s_2) = \begin{cases} 1 & \text{if } s_1 = s_2 = 1 \\ 0 & \text{otherwise} \end{cases}$

# Secure Multiparty Computation

The previous scenario is a secure multiparty computation problem:

- There are  $n$  agents (parties)  $1, 2, \dots, n$
- The  $i$ -th agent has a secret information  $s_i$
- The agents wish to jointly compute some function  $f(s_1, s_2, \dots, s_n)$
- No information about  $s_1, s_2, \dots, s_n$  should be revealed

In the movie selection problem:

- $n = 2$
- $s_1, s_2 \in \{0, 1\}$
- $f(s_1, s_2) = \begin{cases} 1 & \text{if } s_1 = s_2 = 1 \\ 0 & \text{otherwise} \end{cases} = s_1 \wedge s_2$

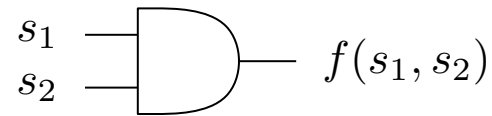
# Secure Multiparty Computation

The previous scenario is a secure multiparty computation problem:

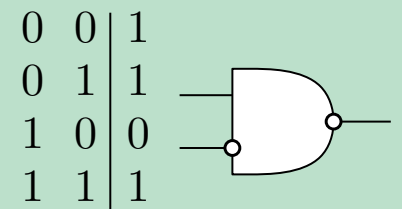
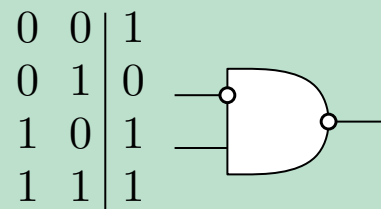
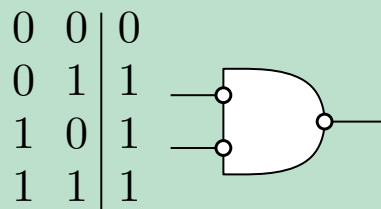
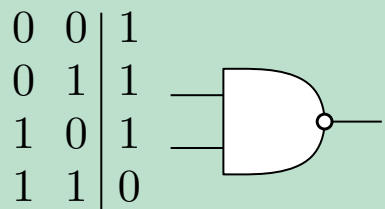
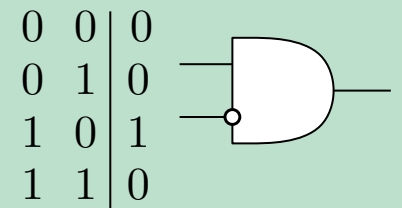
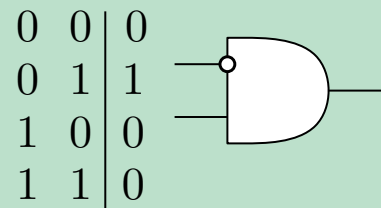
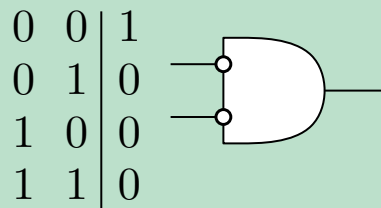
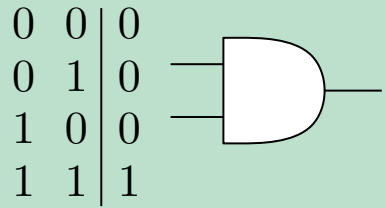
- There are  $n$  agents (parties)  $1, 2, \dots, n$
- The  $i$ -th agent has a secret information  $s_i$
- The agents wish to jointly compute some function  $f(s_1, s_2, \dots, s_n)$
- No information about  $s_1, s_2, \dots, s_n$  should be revealed

In the movie selection problem:

- $n = 2$
- $s_1, s_2 \in \{0, 1\}$
- $f(s_1, s_2) = \begin{cases} 1 & \text{if } s_1 = s_2 = 1 \\ 0 & \text{otherwise} \end{cases} = s_1 \wedge s_2$

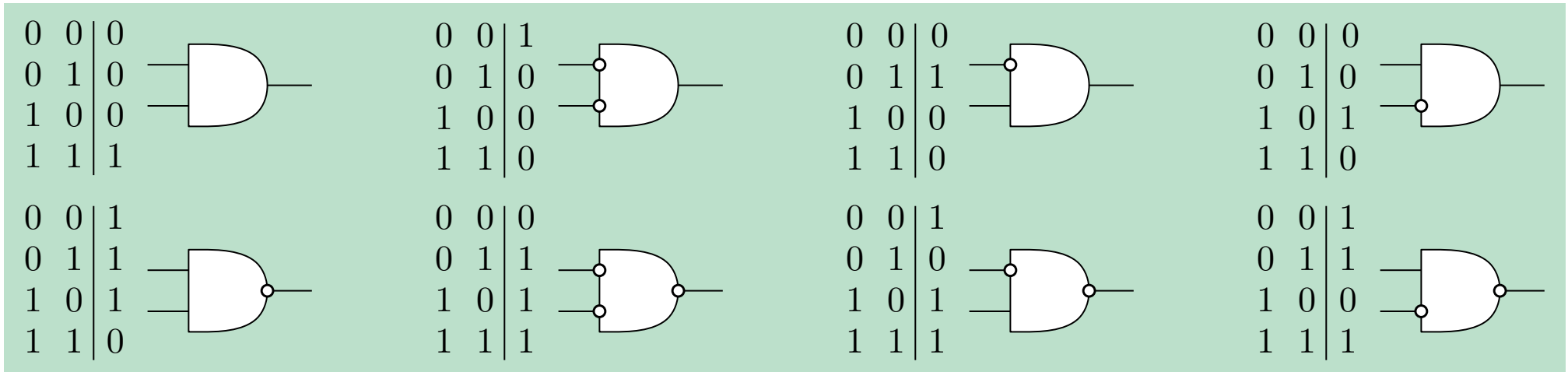


# Secure Multiparty Computation



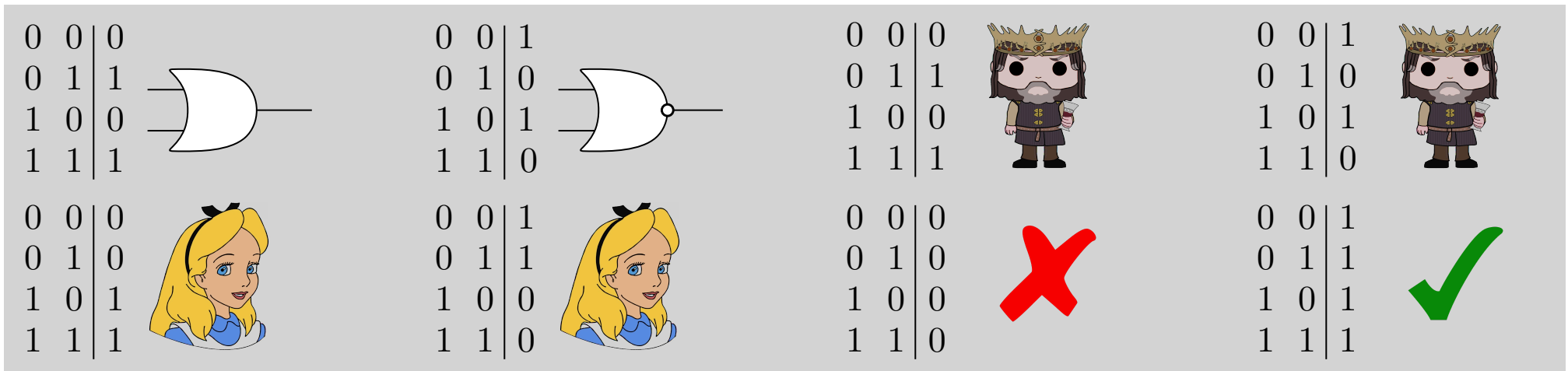
Done!

# Secure Multiparty Computation

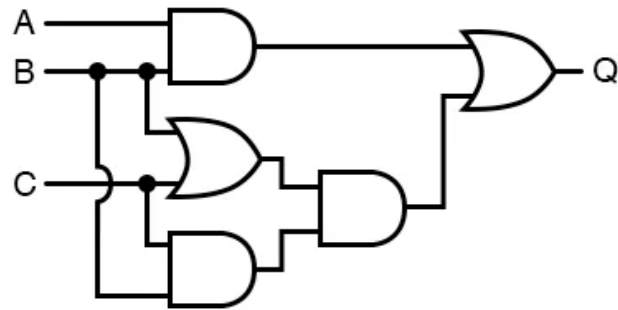


Done!

Not needed



# Secure Multiparty Computation



**What about arbitrary circuits?**

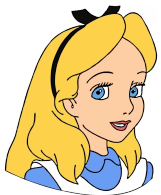
# Zero Knowledge Proofs

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

# Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

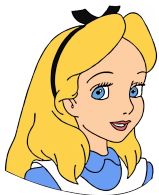


# Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

I know a solution to this Sudoku instance

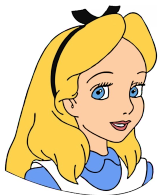


# Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

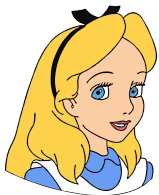
Really? Show it to me!



# Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



I don't want to reveal it to you

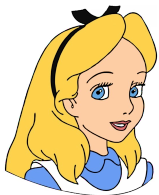


# Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Then I don't believe you really have a solution



# Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



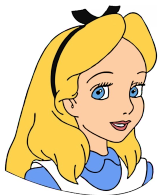
I can prove to you that I have a solution without revealing anything about it



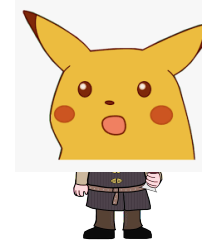
# Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



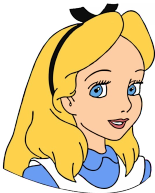
I can prove to you that I have a solution without revealing anything about it



# Zero Knowledge Proofs

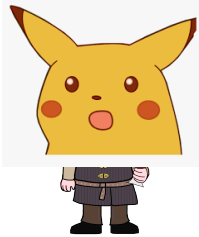
5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



I can prove to you that I have a solution without revealing anything about it

Zero Knowledge protocol

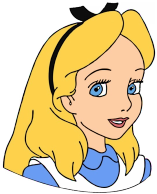


# Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

I am now convinced you have the solution



Zero Knowledge protocol



# Zero Knowledge Proofs

5	3			7			
6			1	9	5		
	9	8					6
8				6			3
4			8		3		1
7				2			6
	6					2	8
			4	1	9		5
				8			7
							9

Hey, Charlie!  
Alice has a solution to  
this Sudoku instance



# Zero Knowledge Proofs

5	3			7			
6			1	9	5		
	9	8					6
8				6			3
4			8		3		1
7				2			6
	6					2	8
			4	1	9		5
				8			7
							9



Prove it!



# Zero Knowledge Proofs

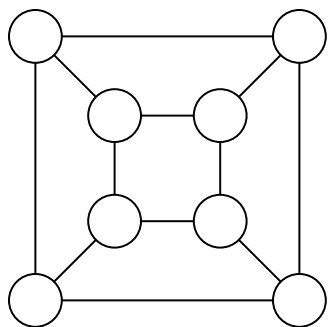
5	3			7			
6			1	9	5		
	9	8					6
8				6			3
4			8		3		1
7				2			6
	6					2	8
			4	1	9		5
				8			7
							9

Even if I definitely know she has a solution, somehow I have no way of proving that

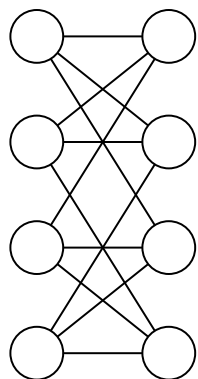


# Zero Knowledge Proofs

$G_1 = (V_1, E_1)$



$G_2 = (V_2, E_2)$

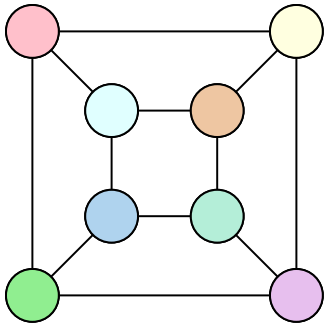


## Graph isomorphism problem

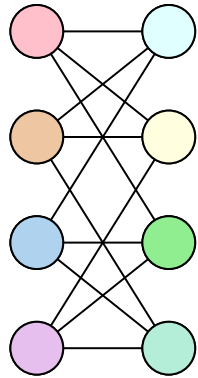
$G_1$  is isomorphic to  $G_2$  iff  $\exists$  bijection  $\pi : V_1 \rightarrow V_2$  s.t.  $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$ .

# Zero Knowledge Proofs

$G_1 = (V_1, E_1)$



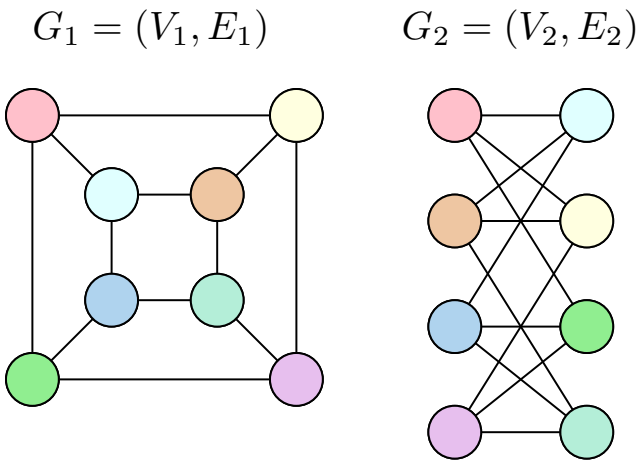
$G_2 = (V_2, E_2)$



## Graph isomorphism problem

$G_1$  is isomorphic to  $G_2$  iff  $\exists$  bijection  $\pi : V_1 \rightarrow V_2$  s.t.  $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$ .

# Zero Knowledge Proofs

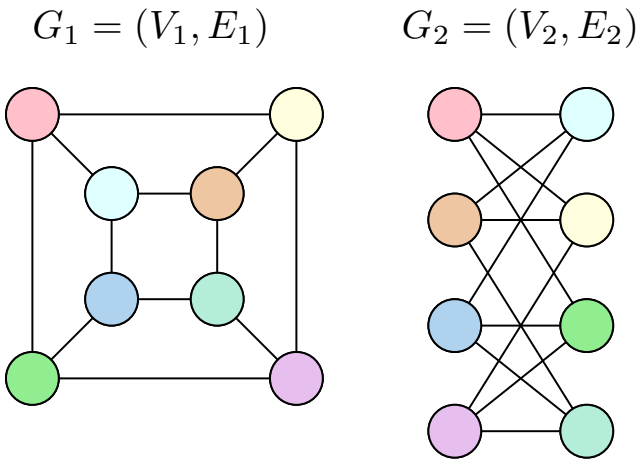


## Graph isomorphism problem

$G_1$  is isomorphic to  $G_2$  iff  $\exists$  bijection  $\pi : V_1 \rightarrow V_2$  s.t.  $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$ .

- Is in NP (the certificate is  $\pi$ )

# Zero Knowledge Proofs

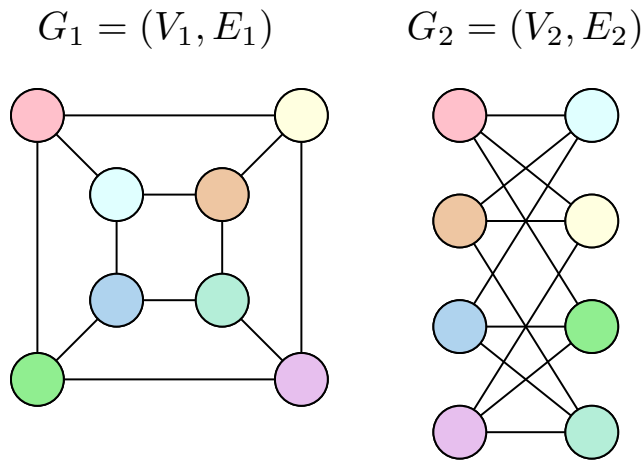


## Graph isomorphism problem

$G_1$  is isomorphic to  $G_2$  iff  $\exists$  bijection  $\pi : V_1 \rightarrow V_2$  s.t.  $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$ .

- Is in NP (the certificate is  $\pi$ )
- Is not known to be NP-Complete

# Zero Knowledge Proofs

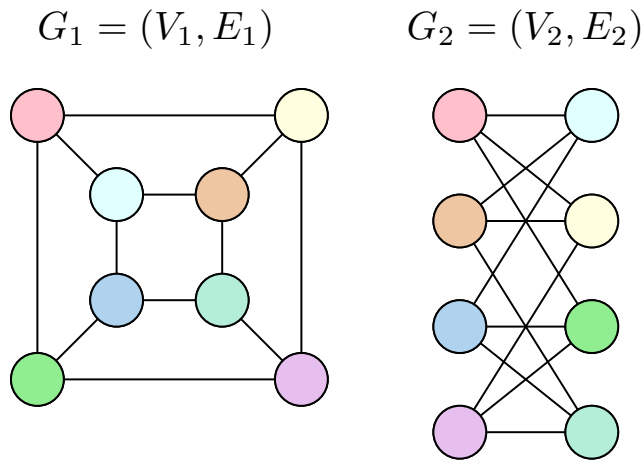


## Graph isomorphism problem

$G_1$  is isomorphic to  $G_2$  iff  $\exists$  bijection  $\pi : V_1 \rightarrow V_2$  s.t.  $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$ .

- Is in NP (the certificate is  $\pi$ )
  - Is not known to be NP-Complete
  - Is a candidate problem to be in the class NP-Intermediate
- NP-Intermediate =  $NP \setminus (P \cup NP-C)$

# Zero Knowledge Proofs

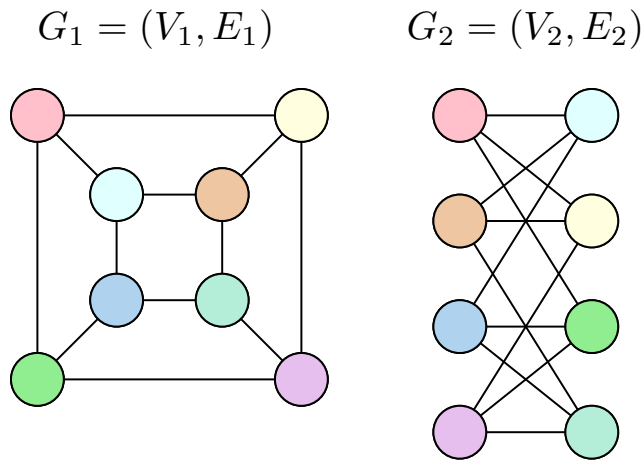


## Graph isomorphism problem

$G_1$  is isomorphic to  $G_2$  iff  $\exists$  bijection  $\pi : V_1 \rightarrow V_2$  s.t.  $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$ .

- Is in NP (the certificate is  $\pi$ )
- Is not known to be NP-Complete
- Is a candidate problem to be in the class NP-Intermediate      NP-Intermediate =  $\text{NP} \setminus (\text{P} \cup \text{NP-C})$
- NP-Intermediate  $\neq \emptyset \iff \text{P} \neq \text{NP}$ .

# Zero Knowledge Proofs



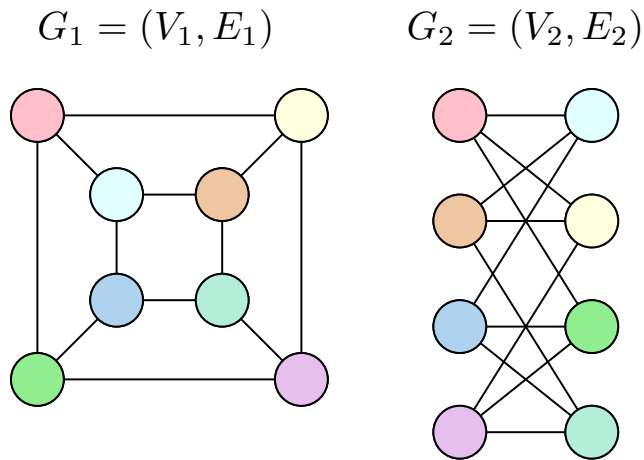
Alice knows an isomorphism  $\pi$  between  $G_1$  and  $G_2$

## Graph isomorphism problem

$G_1$  is isomorphic to  $G_2$  iff  $\exists$  bijection  $\pi : V_1 \rightarrow V_2$  s.t.  $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$ .

- Is in NP (the certificate is  $\pi$ )
- Is not known to be NP-Complete
- Is a candidate problem to be in the class NP-Intermediate      NP-Intermediate =  $\text{NP} \setminus (\text{P} \cup \text{NP-C})$
- NP-Intermediate  $\neq \emptyset \iff \text{P} \neq \text{NP}$ .

# Zero Knowledge Proofs



Alice knows an isomorphism  $\pi$  between  $G_1$  and  $G_2$

Alice can use a Zero Knowledge protocol to convince Bob that  $G_1$  and  $G_2$  are isomorphic without revealing  $\pi$

## Graph isomorphism problem

$G_1$  is isomorphic to  $G_2$  iff  $\exists$  bijection  $\pi : V_1 \rightarrow V_2$  s.t.  $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$ .

- Is in NP (the certificate is  $\pi$ )
- Is not known to be NP-Complete
- Is a candidate problem to be in the class NP-Intermediate      NP-Intermediate =  $\text{NP} \setminus (\text{P} \cup \text{NP-C})$
- NP-Intermediate  $\neq \emptyset \iff \text{P} \neq \text{NP}$ .

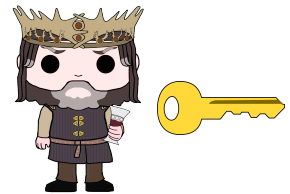
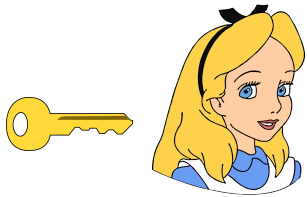
# Basic definitions

# Types of cryptography

There are two broad settings in which encryption is used:

The **private-key** or **symmetric** setting:

- Alice and Bob have a **shared, secret** key

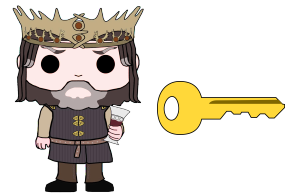
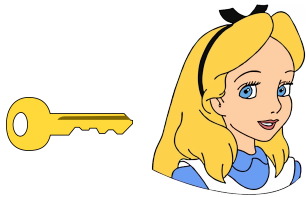


# Types of cryptography

There are two broad settings in which encryption is used:

The **private-key** or **symmetric** setting:

- Alice and Bob have a **shared, secret** key
- The key must be shared securely in advance, and must be kept secret (before, during, and after Bob and Alice's interaction)

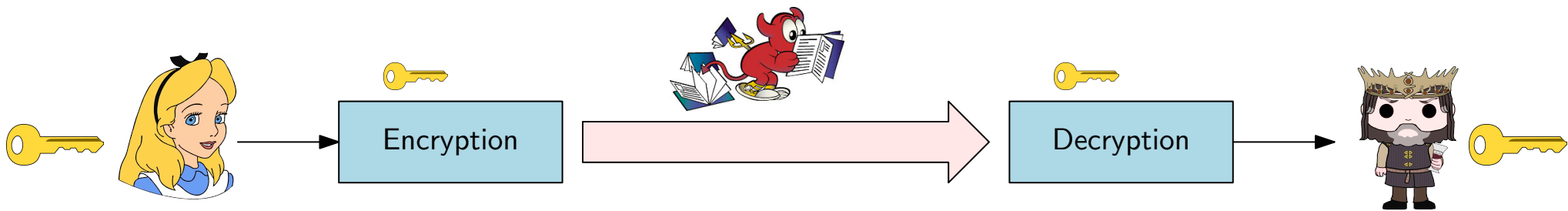


# Types of cryptography

There are two broad settings in which encryption is used:

The **private-key** or **symmetric** setting:

- Alice and Bob have a **shared, secret** key
- The key must be shared securely in advance, and must be kept secret (before, during, and after Bob and Alice's interaction)
- Messages are encrypted and decrypted using the same key

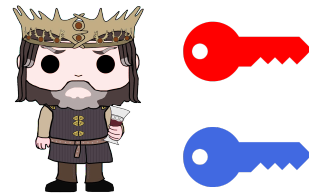
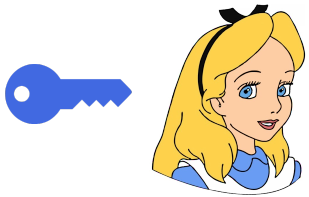


# Types of cryptography

There are two broad settings in which encryption is used:

The **public-key** or **asymmetric** setting:

- Alice and Bob do not need to share any secret information

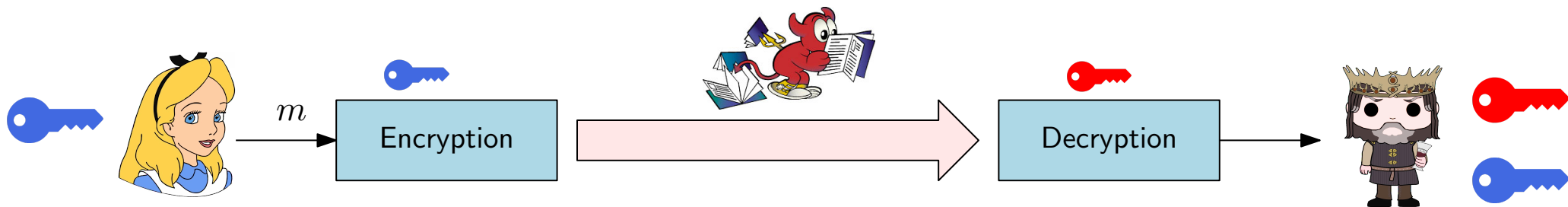


# Types of cryptography

There are two broad settings in which encryption is used:

The **public-key** or **asymmetric** setting:

- Alice and Bob do not need to share any secret information
- Messages are encrypted using only public information (public keys) and decrypted with private keys (which are not shared!)

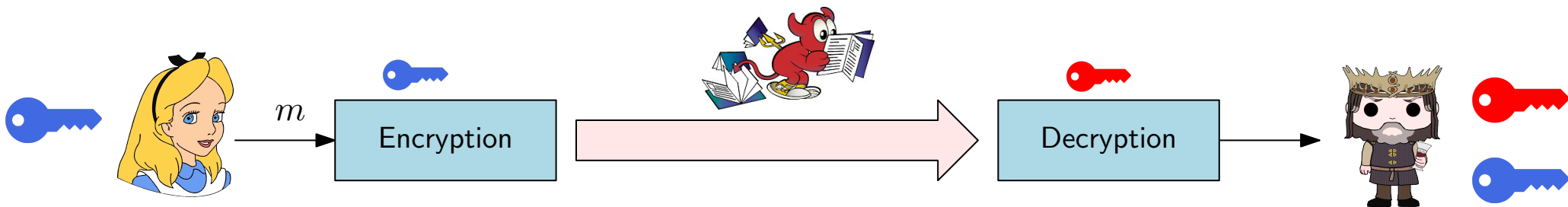


# Types of cryptography

There are two broad settings in which encryption is used:

The **public-key** or **asymmetric** setting:

- Alice and Bob do not need to share any secret information
- Messages are encrypted using only public information (public keys) and decrypted with private keys (which are not shared!)
- Anybody can encrypt messages for a given recipient

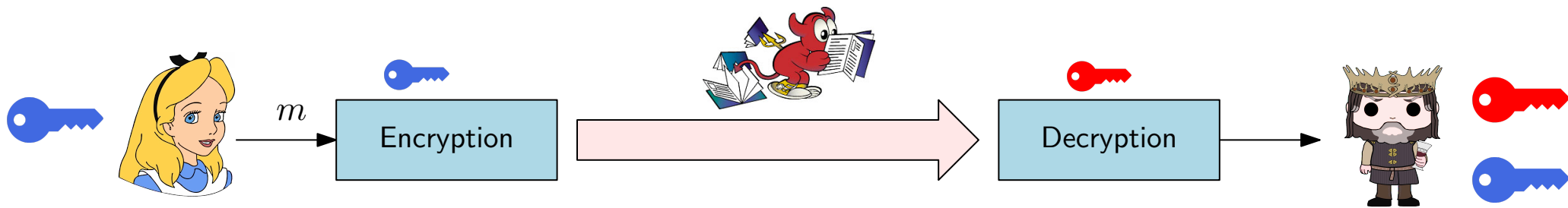


# Types of cryptography

There are two broad settings in which encryption is used:

The **public-key** or **asymmetric** setting:

- Alice and Bob do not need to share any secret information
- Messages are encrypted using only public information (public keys) and decrypted with private keys (which are not shared!)
- Anybody can encrypt messages for a given recipient



For now, we will only be concerned with private-key cryptography

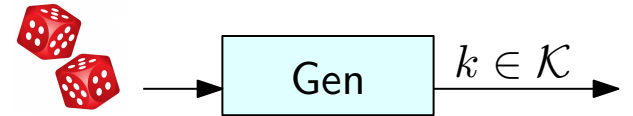
# Private-key cryptography

A private-key encryption scheme consists of three algorithms:

# Private-key cryptography

A private-key encryption scheme consists of three algorithms:

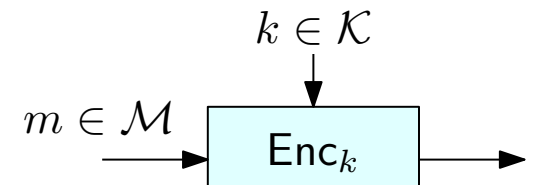
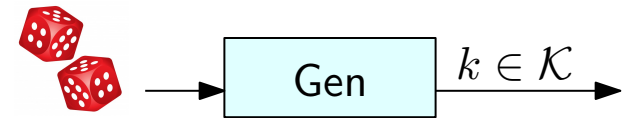
- Gen is a randomized algorithm that outputs a *key* from some finite set  $\mathcal{K}$  (key space) according to some probability distribution



# Private-key cryptography

A private-key encryption scheme consists of three algorithms:

- Gen is a randomized algorithm that outputs a *key* from some finite set  $\mathcal{K}$  (key space) according to some probability distribution
- Enc is a (possibly randomized) algorithm that takes as input a key  $k \in \mathcal{K}$  and a *message* (or plaintext)  $m$  from some set  $\mathcal{M}$  (message space) and outputs a *ciphertext*  $c$  obtained by encrypting  $m$  with key  $k$ .

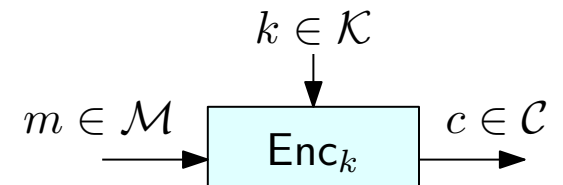
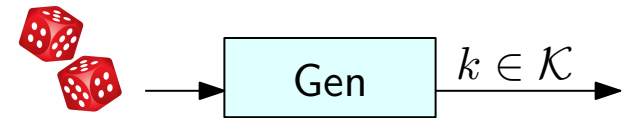


# Private-key cryptography

A private-key encryption scheme consists of three algorithms:

- Gen is a randomized algorithm that outputs a *key* from some finite set  $\mathcal{K}$  (key space) according to some probability distribution
- Enc is a (possibly randomized) algorithm that takes as input a key  $k \in \mathcal{K}$  and a *message* (or plaintext)  $m$  from some set  $\mathcal{M}$  (message space) and outputs a *ciphertext*  $c$  obtained by encrypting  $m$  with key  $k$ .

$\mathcal{C}$  denotes the set of all possible ciphertexts



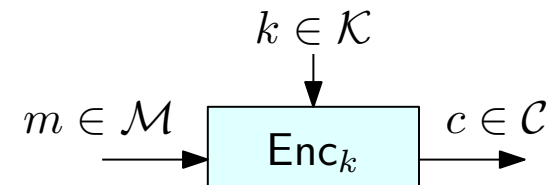
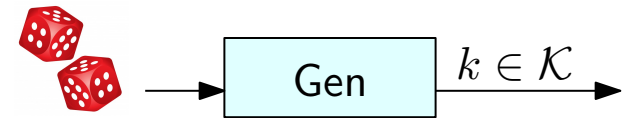
# Private-key cryptography

A private-key encryption scheme consists of three algorithms:

- Gen is a randomized algorithm that outputs a *key* from some finite set  $\mathcal{K}$  (key space) according to some probability distribution
- Enc is a (possibly randomized) algorithm that takes as input a key  $k \in \mathcal{K}$  and a *message* (or plaintext)  $m$  from some set  $\mathcal{M}$  (message space) and outputs a *ciphertext*  $c$  obtained by encrypting  $m$  with key  $k$ .

$\mathcal{C}$  denotes the set of all possible ciphertexts

$\text{Enc}_k(m)$  denotes an execution of Enc with inputs  $k$  and  $m$



# Private-key cryptography

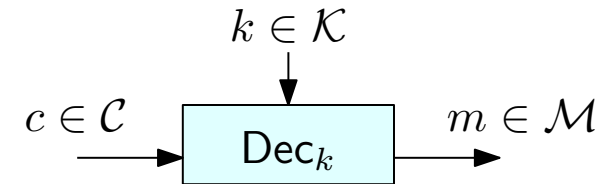
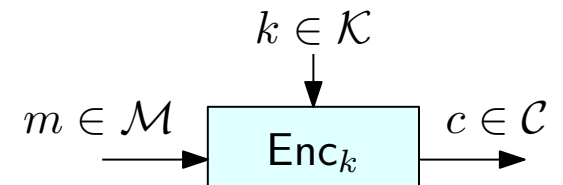
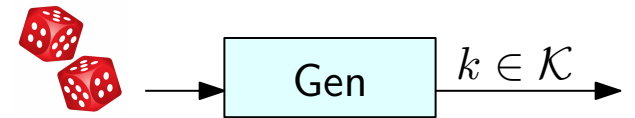
A private-key encryption scheme consists of three algorithms:

- Gen is a randomized algorithm that outputs a *key* from some finite set  $\mathcal{K}$  (key space) according to some probability distribution
- Enc is a (possibly randomized) algorithm that takes as input a key  $k \in \mathcal{K}$  and a *message* (or plaintext)  $m$  from some set  $\mathcal{M}$  (message space) and outputs a *ciphertext*  $c$  obtained by encrypting  $m$  with key  $k$ .

$\mathcal{C}$  denotes the set of all possible ciphertexts

$\text{Enc}_k(m)$  denotes an execution of Enc with inputs  $k$  and  $m$

- Dec is a deterministic algorithm that takes as input a key  $k \in \mathcal{K}$  and a ciphertext  $c \in \mathcal{C}$  and outputs a message  $m \in \mathcal{M}$



# Private-key cryptography

A private-key encryption scheme consists of three algorithms:

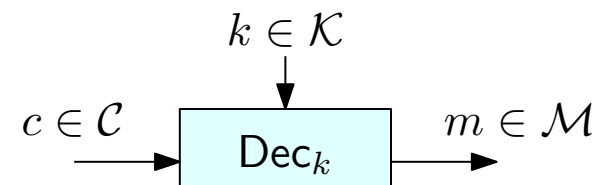
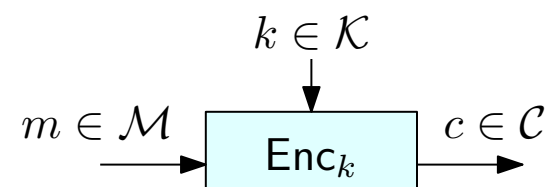
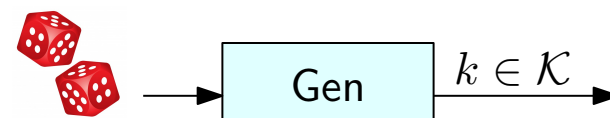
- Gen is a randomized algorithm that outputs a *key* from some finite set  $\mathcal{K}$  (key space) according to some probability distribution
- Enc is a (possibly randomized) algorithm that takes as input a key  $k \in \mathcal{K}$  and a *message* (or plaintext)  $m$  from some set  $\mathcal{M}$  (message space) and outputs a *ciphertext*  $c$  obtained by encrypting  $m$  with key  $k$ .

$\mathcal{C}$  denotes the set of all possible ciphertexts

$\text{Enc}_k(m)$  denotes an execution of Enc with inputs  $k$  and  $m$

- Dec is a deterministic algorithm that takes as input a key  $k \in \mathcal{K}$  and a ciphertext  $c \in \mathcal{C}$  and outputs a message  $m \in \mathcal{M}$

$\text{Dec}_k(c)$  denotes an execution of Dec with inputs  $k$  and  $c$



# Private-key cryptography

A private-key encryption scheme consists of three algorithms:

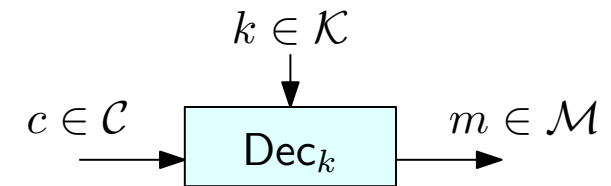
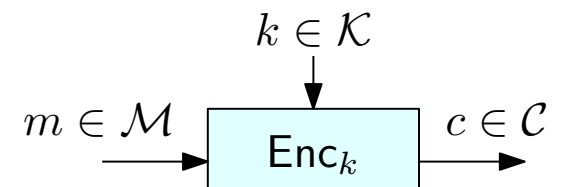
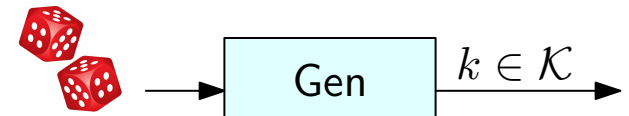
- Gen is a randomized algorithm that outputs a *key* from some finite set  $\mathcal{K}$  (key space) according to some probability distribution
- Enc is a (possibly randomized) algorithm that takes as input a key  $k \in \mathcal{K}$  and a *message* (or plaintext)  $m$  from some set  $\mathcal{M}$  (message space) and outputs a *ciphertext*  $c$  obtained by encrypting  $m$  with key  $k$ .

$\mathcal{C}$  denotes the set of all possible ciphertexts

$\text{Enc}_k(m)$  denotes an execution of Enc with inputs  $k$  and  $m$

- Dec is a deterministic algorithm that takes as input a key  $k \in \mathcal{K}$  and a ciphertext  $c \in \mathcal{C}$  and outputs a message  $m \in \mathcal{M}$

$\text{Dec}_k(c)$  denotes an execution of Dec with inputs  $k$  and  $c$



**Perfect correctness:**  $\forall k \in \mathcal{K}, m \in \mathcal{M}$  if  $c$  can be output by  $\text{Enc}_k(m)$  then  $\text{Dec}_k(c) = m$

# Kerckhoffs' principle

Historically, encryption has been performed by devising some clever method to scramble  $m$  into a cyphertext  $c$  (and vice-versa)

# Kerckhoffs' principle

Historically, encryption has been performed by devising some clever method to scramble  $m$  into a cyphertext  $c$  (and vice-versa)

- No key (we can think of a **fixed key** as being hardcoded in the algorithm)

# Kerckhoffs' principle

Historically, encryption has been performed by devising some clever method to scramble  $m$  into a cyphertext  $c$  (and vice-versa)

- No key (we can think of a **fixed key** as being hardcoded in the algorithm)
- The method was only known only to the sender and the recipient

# Kerckhoffs' principle

Historically, encryption has been performed by devising some clever method to scramble  $m$  into a cyphertext  $c$  (and vice-versa)

- No key (we can think of a **fixed key** as being hardcoded in the algorithm)
- The method was only known only to the sender and the recipient

This means that if the encryption/decryption algorithm(s) were leaked to an adversary, the honest parties needed to come up with a new scheme

# Kerckhoffs' principle

Historically, encryption has been performed by devising some clever method to scramble  $m$  into a cyphertext  $c$  (and vice-versa)

- No key (we can think of a **fixed key** as being hardcoded in the algorithm)
- The method was only known only to the sender and the recipient

This means that if the encryption/decryption algorithm(s) were leaked to an adversary, the honest parties needed to come up with a new scheme

**Coming up with (secure) encryption schemes is hard!**

# Kerckhoffs' principle

Historically, encryption has been performed by devising some clever method to scramble  $m$  into a cyphertext  $c$  (and vice-versa)

- No key (we can think of a **fixed key** as being hardcoded in the algorithm)
- The method was only known only to the sender and the recipient

This means that if the encryption/decryption algorithm(s) were leaked to an adversary, the honest parties needed to come up with a new scheme

**Coming up with (secure) encryption schemes is hard!**

**Kerckhoffs' principle:** all parts of an encryption scheme should be public, except the key

*The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.*



# Kerckhoffs' principle

Historically, encryption has been performed by devising some clever method to scramble  $m$  into a cyphertext  $c$  (and vice-versa)

- No key (we can think of a **fixed key** as being hardcoded in the algorithm)
- The method was only known only to the sender and the recipient

This means that if the encryption/decryption algorithm(s) were leaked to an adversary, the honest parties needed to come up with a new scheme

**Coming up with (secure) encryption schemes is hard!**

**Kerckhoffs' principle:** all parts of an encryption scheme should be public, except the key

*The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.*

If the key is leaked, it is easy to replace it



No security through obscurity

But...

# No security through obscurity

But...

...surely keeping **both the scheme and the key** secret is more secure than just keeping the key secret... right?

# No security through obscurity

But...

...surely keeping **both the scheme and the key** secret is more secure than just keeping the key secret... right?

**Don't roll your own encryption!**

# No security through obscurity

But...

...surely keeping **both the scheme and the key** secret is more secure than just keeping the key secret... right?

**Don't roll your own encryption!**

- It's really easy to make mistakes that will render an encryption scheme insecure

# No security through obscurity

But...

...surely keeping **both the scheme and the key** secret is more secure than just keeping the key secret... right?

**Don't roll your own encryption!**

- It's really easy to make mistakes that will render an encryption scheme insecure
- Some of these mistakes are really subtle (we will see some of them in the course)

# No security through obscurity

But...

...surely keeping **both the scheme and the key** secret is more secure than just keeping the key secret... right?

## **Don't roll your own encryption!**

- It's really easy to make mistakes that will render an encryption scheme insecure
- Some of these mistakes are really subtle (we will see some of them in the course)

Encryption schemes whose details are public:

- Have undergone public scrutiny by experts and no flaws have been found

# No security through obscurity

But...

...surely keeping **both the scheme and the key** secret is more secure than just keeping the key secret... right?

## **Don't roll your own encryption!**

- It's really easy to make mistakes that will render an encryption scheme insecure
- Some of these mistakes are really subtle (we will see some of them in the course)

Encryption schemes whose details are public:

- Have undergone public scrutiny by experts and no flaws have been found
- Can be standardized

# No security through obscurity

But...

...surely keeping **both the scheme and the key** secret is more secure than just keeping the key secret... right?

## Don't roll your own encryption!

- It's really easy to make mistakes that will render an encryption scheme insecure
- Some of these mistakes are really subtle (we will see some of them in the course)

Encryption schemes whose details are public:

- Have undergone public scrutiny by experts and no flaws have been found
- Can be standardized
- Parties only need to share a key (and not the exact details of the secret scheme).

# No security through obscurity

But...

...surely keeping **both the scheme and the key** secret is more secure than just keeping the key secret... right?

## Don't roll your own encryption!

- It's really easy to make mistakes that will render an encryption scheme insecure
- Some of these mistakes are really subtle (we will see some of them in the course)

Encryption schemes whose details are public:

- Have undergone public scrutiny by experts and no flaws have been found
- Can be standardized
- Parties only need to share a key (and not the exact details of the secret scheme).
- No need for Bob to implement Alice's weird scheme on his own (Bob can use public, vetted, implementations of well-known schemes)