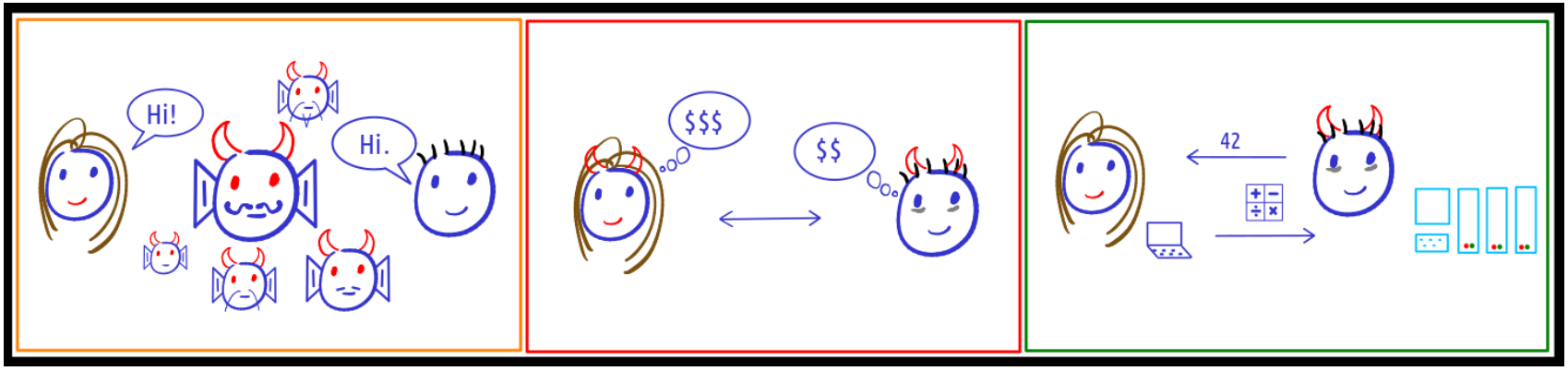


CS783: Theoretical Foundations of Cryptography

Lecture 1 (30/Jul/24)

Instructor: Chethan Kamath

Administrivia...

- Timing and Venue: Slot 10 (14:00–15:25, Tuesdays and Fridays) in CC105
- Contact hours: after lectures, or appointment by e-mail
- Teaching assistants:
  - Sarthak Sharma (23M0789) and Nivesh Aggarwal (22b0912)

Administrivia...

- Timing and Venue: Slot 10 (14:00-15:25, Tuesdays and Fridays) in CC105
- Contact hours: after lectures, or appointment by e-mail
- Teaching assistants:
  - Sarthak Sharma (23M0789) and Nivesh Aggarwal (22b0912)

- Resources
  - Slides and other resources will be posted on my website
    - cse.iitb.ac.in/~ckamath/courses/2024/CS783.html
  - Announcements/online discussion on Moodle:
    - moodle.iitb.ac.in/course/view.php?id=4702

Administrivia...

- Grading Scheme
  - Six ungraded assignments to help with quizzes and exams

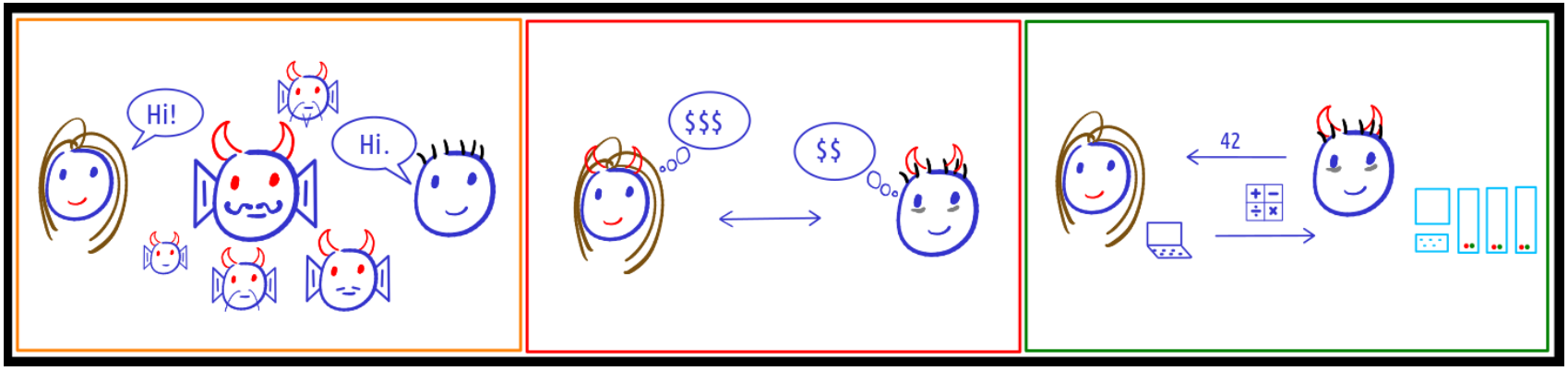
Weightage	Towards
35%	Final
25%	Mid-term
20%	Two quizzes, one each after Modules I and III
15%	Group project/chalk-talk
5%	Class participation, pop-quiz

Administrivia...

- Grading Scheme
  - Six ungraded assignments to help with quizzes and exams

Weightage	Towards
35%	Final
30 ~~25~~%	Mid-term
30 ~~20~~%	Two quizzes, one each after Modules I and III
~~15%~~	~~Group project/chalk talk~~
5%	Class participation, pop-quiz

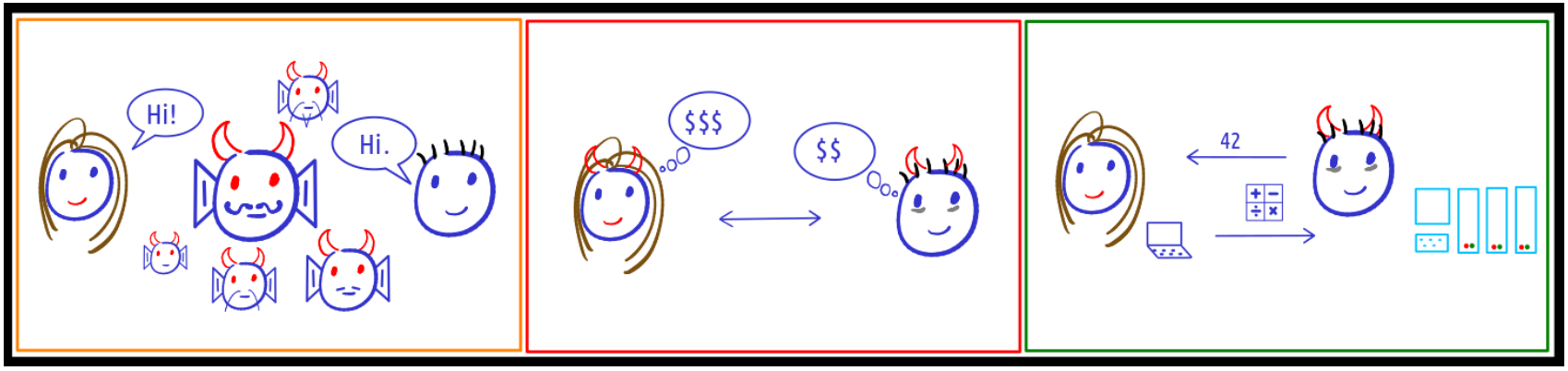
- Will scrap group project/chalk talk if number of creditors > 30 and grade will be redistributed
- Attendance is not mandatory (but encouraged)
- Any volunteers for class rep?



CS783: Theoretical Foundations of Cryptography

Lecture 1 (30/Jul/24)

Instructor: Chethan Kamath



CS783: Theoretical Foundations of Cryptography

Lecture 1 (30/Jul/24)

Instructor: Chethan Kamath

You Use Cryptography all the Time!

Credit for images: Wikipedia

You Use Cryptography all the Time!

Using internet

Credit for images: Wikipedia

You Use Cryptography all the Time!

Using laptop/phone

Using internet

Credit for images: Wikipedia

You Use Cryptography all the Time!

Secure communication

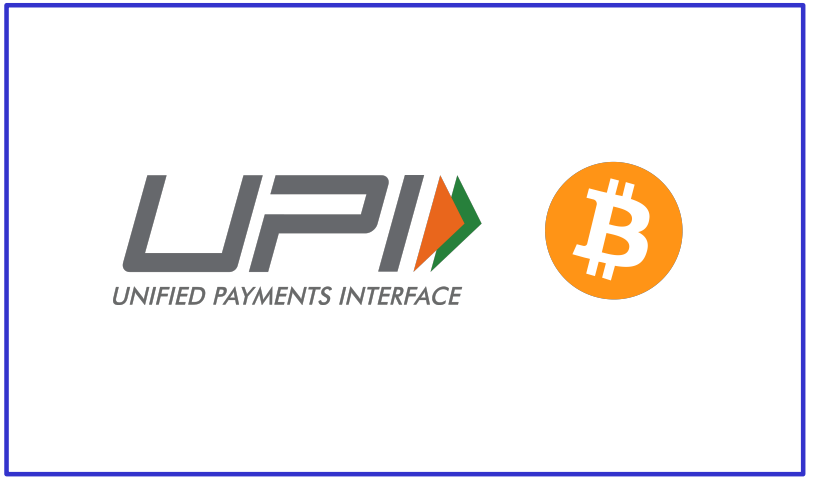
Using laptop/phone

Using internet

Credit for images: Wikipedia

You Use Cryptography all the Time!

Secure communication



Online transactions

Using laptop/phone

Using internet

Credit for images: Wikipedia

What is Cryptography?

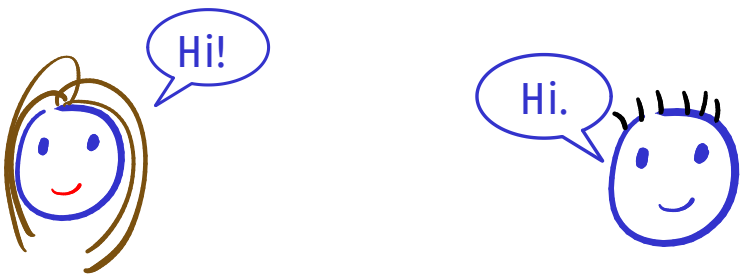
- Science of carrying out *tasks* securely in an adversarial setting

What is Cryptography?

- Science of carrying out *tasks* securely in an adversarial setting
- An analogy: secret communication

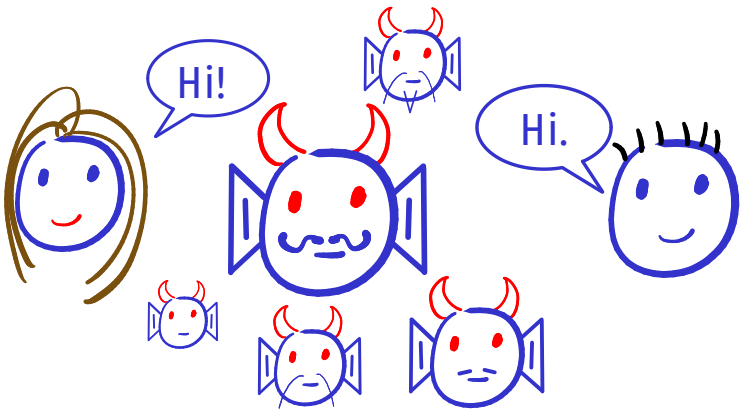
What is Cryptography?

- Science of carrying out *tasks* securely in an adversarial setting
- An analogy: secret communication



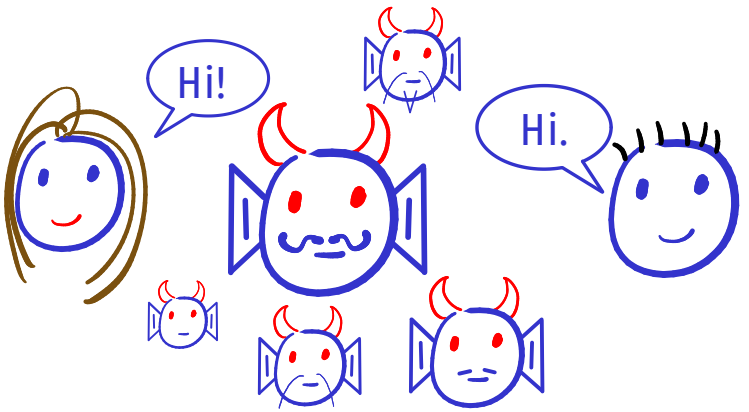
What is Cryptography?

- Science of carrying out *tasks* securely in an adversarial setting
- An analogy: secret communication



What is Cryptography?

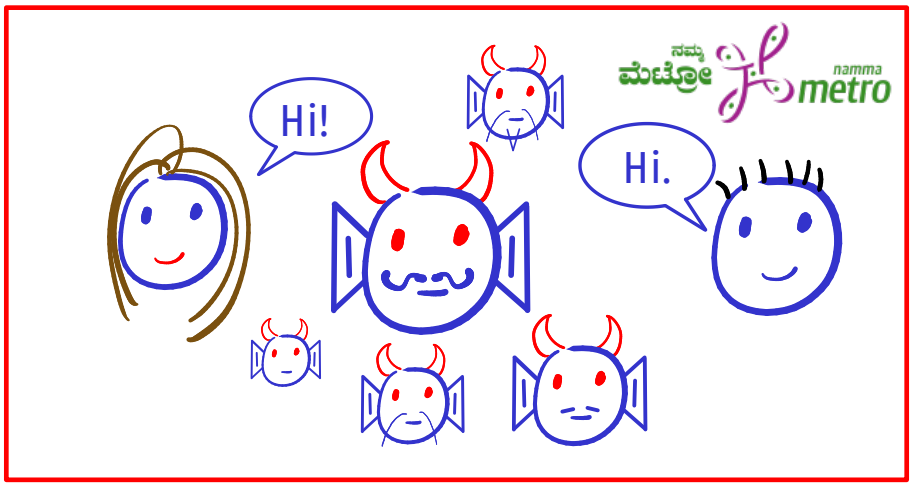
- Science of carrying out *tasks* securely in an adversarial setting
- An analogy: secret communication



- Security goal: conversation remains secret to eavesdroppers

What is Cryptography?

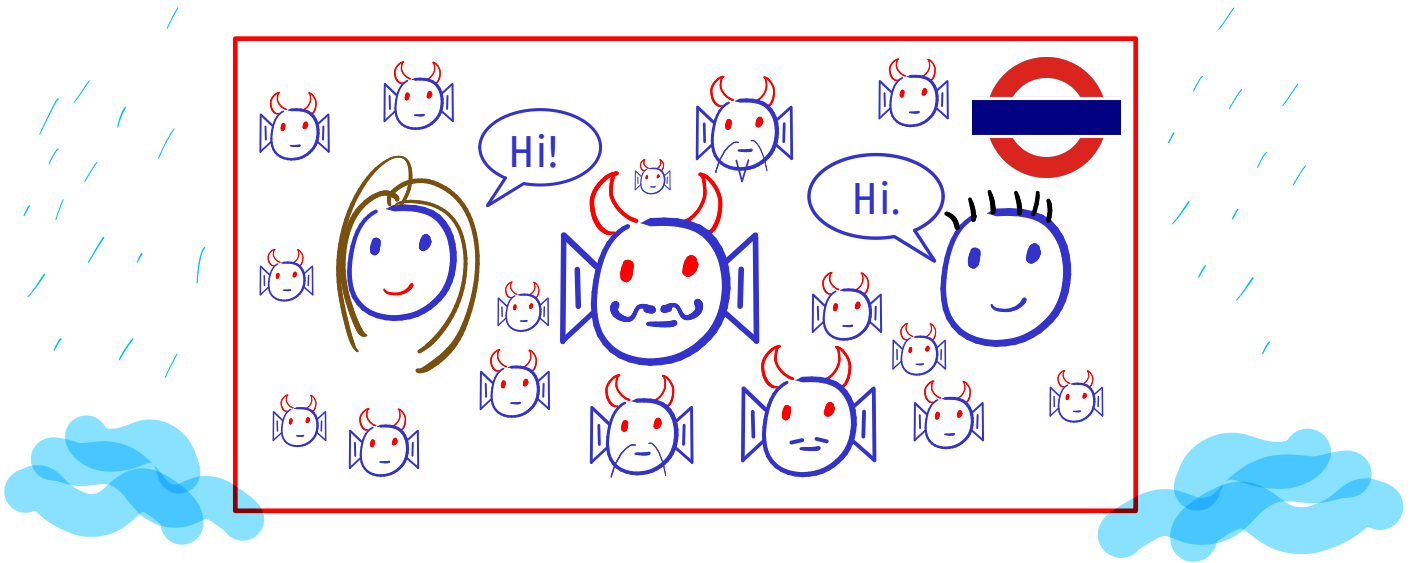
- Science of carrying out *tasks* securely in an adversarial setting
- An analogy: secret communication



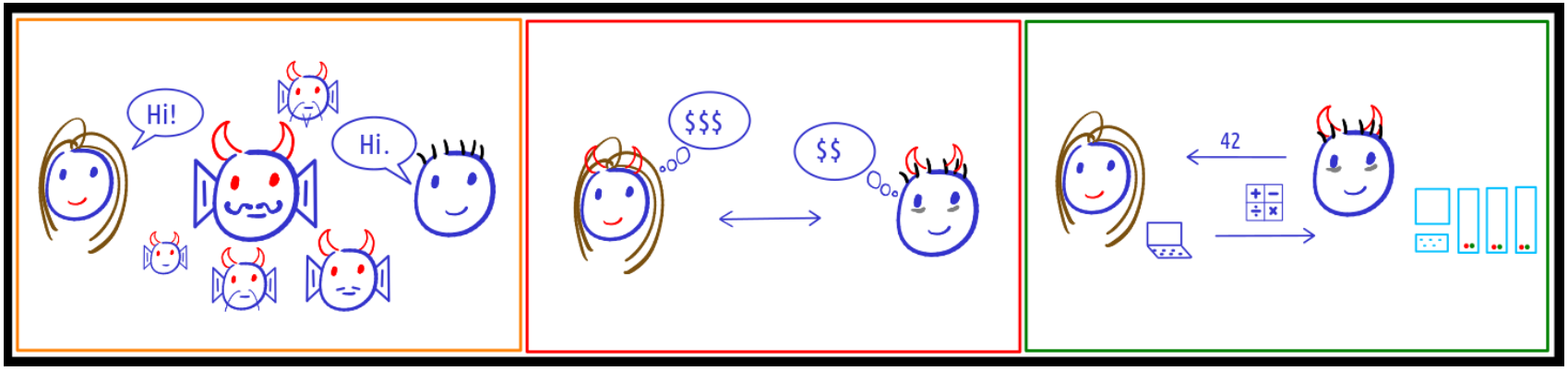
- Security goal: conversation remains secret to eavesdroppers
- Adversarial setting:
  - Bengaluru metro (understand Kannada, English and Hindi)

What is Cryptography?

- Science of carrying out *tasks* securely in an adversarial setting
- An analogy: secret communication



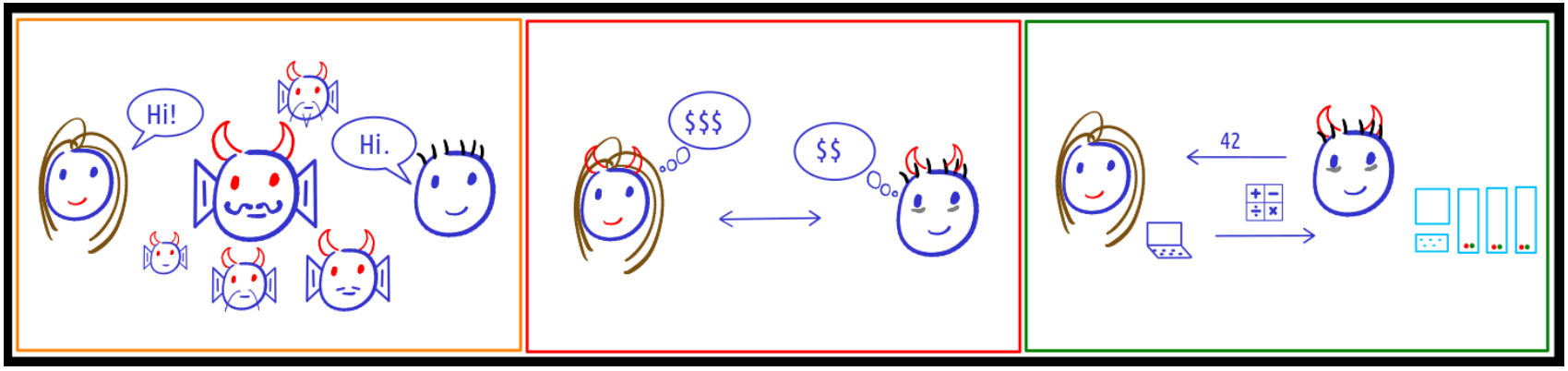
- Security goal: conversation remains secret to eavesdroppers
- Adversarial setting:
  - Bengaluru metro (understand Kannada, English and Hindi)
  - Mumbai local (understand Marathi, English and Hindi)



CS783: Theoretical Foundations of Cryptography

Lecture 1 (30/Jul/24)

Instructor: Chethan Kamath



CS783: Theoretical Foundations of Cryptography

Lecture 1 (30/Jul/24)

Instructor: Chethan Kamath

About this Course: What to Expect?

- Graduate-level *theoretical* cryptography course
  - Closely follows Vinod Vaikuntanathan's MIT6875
  - Focus on breadth instead of depth

About this Course: What to Expect?

- Graduate-level *theoretical* cryptography course
  - Closely follows Vinod Vaikuntanathan's MIT6875
  - Focus on breadth instead of depth

- Goal: *formally* study how to carry out certain tasks securely in an adversarial setting

About this Course: What to Expect?

- Graduate-level *theoretical* cryptography course
  - Closely follows Vinod Vaikuntanathan's MIT6875
  - Focus on breadth instead of depth

- Goal: *formally* study how to carry out certain tasks securely in an adversarial setting
- We will follow the following general *template*:
  1. Identify the task

About this Course: What to Expect?

- Graduate-level *theoretical* cryptography course
  - Closely follows Vinod Vaikuntanathan's MIT6875
  - Focus on breadth instead of depth

- Goal: *formally* study how to carry out certain tasks securely in an adversarial setting
- We will follow the following general *template*:
  1. Identify the task
  2. Come up with precise threat model M (a.k.a security model)
     - Adversary/Attack: What are the adversary's capabilities?
     - Security Goal: What does it mean to be secure?

About this Course: What to Expect?

- Graduate-level *theoretical* cryptography course
  - Closely follows Vinod Vaikuntanathan's MIT6875
  - Focus on breadth instead of depth

- Goal: *formally* study how to carry out certain tasks securely in an adversarial setting
- We will follow the following general *template*:
  1. Identify the task
  2. Come up with precise threat model M (a.k.a security model)
     - Adversary/Attack: What are the adversary's capabilities?
     - Security Goal: What does it mean to be secure?
  3. Construct a scheme Π

About this Course: What to Expect?

- Graduate-level *theoretical* cryptography course
  - Closely follows Vinod Vaikuntanathan's MIT6875
  - Focus on breadth instead of depth

- Goal: *formally* study how to carry out certain tasks securely in an adversarial setting
- We will follow the following general *template*:
  1. Identify the task
  2. Come up with precise threat model M (a.k.a security model)
     - Adversary/Attack: What are the adversary's capabilities?
     - Security Goal: What does it mean to be secure?
  3. Construct a scheme Π
  4. Formally prove that Π in secure in model M

About this Course: What to Expect?

- Graduate-level *theoretical* cryptography course
  - Closely follows Vinod Vaikuntanathan's MIT6875
  - Focus on breadth instead of depth

- Goal: *formally* study how to carry out certain tasks securely in an adversarial setting
- We will follow the following general *template*:
  1. Identify the task
  2. Come up with precise threat model M (a.k.a security model)
     - Adversary/Attack: What are the adversary's capabilities?
     - Security Goal: What does it mean to be secure?
  3. Construct a scheme Π
  4. Formally prove that Π in secure in model M

- Soft prerequisites: discrete mathematics, probability theory, some familiarity with formal proofs

About this Course: What *NOT* to Expect?

- Will not involve coding

About this Course: What *NOT* to Expect?

- Will not involve coding
  - But if you are interested in coding, I will point you to relevant cryptographic libraries and real-world applications

About this Course: What *NOT* to Expect?

- Will not involve coding
  - But if you are interested in coding, I will point you to relevant cryptographic libraries and real-world applications
- Wait for the next iteration of *CS409: Introduction to Cryptography* (previously CS406)

This Lecture: An Overview of the Tasks Covered...

1. Module I: Secure Communication in Shared-Key Setting

This Lecture: An Overview of the Tasks Covered...

1. Module I: Secure Communication in Shared-Key Setting

2. Module II: Secure Communication in Public-Key Setting

This Lecture: An Overview of the Tasks Covered...

1. Module I: Secure Communication in Shared-Key Setting
2. Module II: Secure Communication in Public-Key Setting
3. Module III: Secure Computation

This Lecture: An Overview of the Tasks Covered...

1. Module I: Secure Communication in Shared-Key Setting
2. Module II: Secure Communication in Public-Key Setting
3. Module III: Secure Computation
4. Module IV: Some Advanced Tasks

This Lecture: An Overview of the Tasks Covered...

- Roughly corresponds to four "eras"

Credit for images: Wikipedia

This Lecture: An Overview of the Tasks Covered...

- Roughly corresponds to four "eras"

MODULE 1
(Shared keys)

For a large part of history

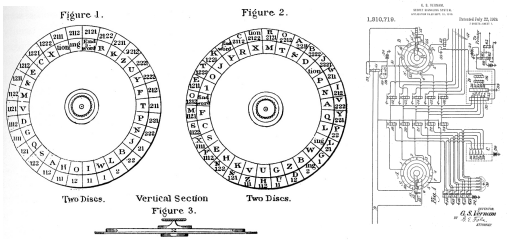
Credit for images: Wikipedia

This Lecture: An Overview of the Tasks Covered...

- Roughly corresponds to four "eras"

MODULE 1
(Shared keys)

For a large part of history



Credit for images: Wikipedia

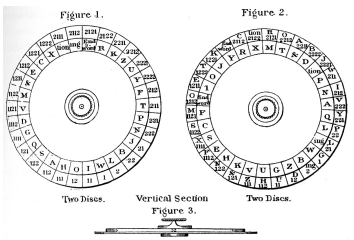
This Lecture: An Overview of the Tasks Covered...

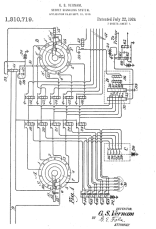
- Roughly corresponds to four "eras"

MODULE 1
(Shared keys)

MODULE 2
(Public keys)

For a large part of history

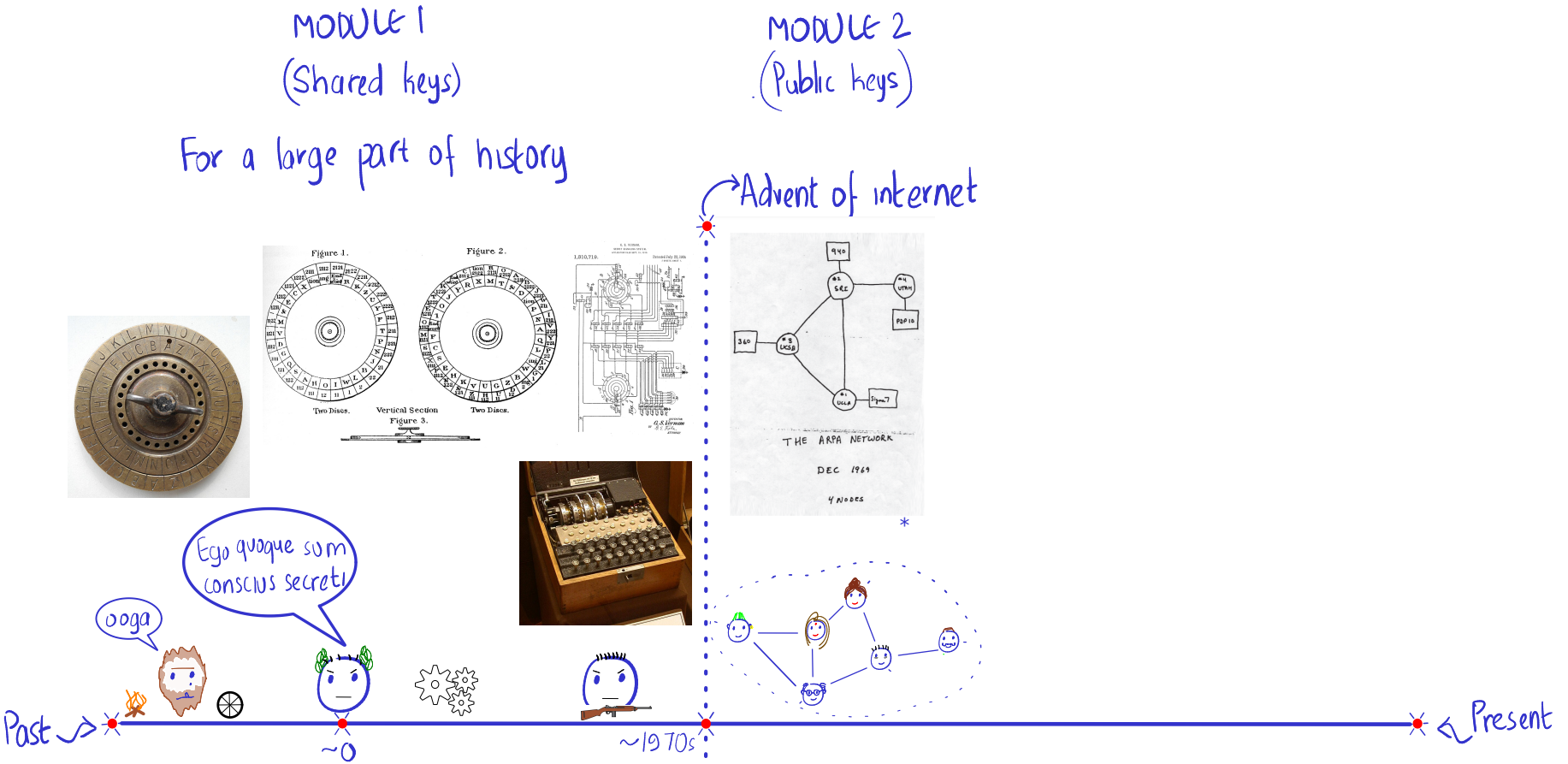




Credit for images: Wikipedia

This Lecture: An Overview of the Tasks Covered...

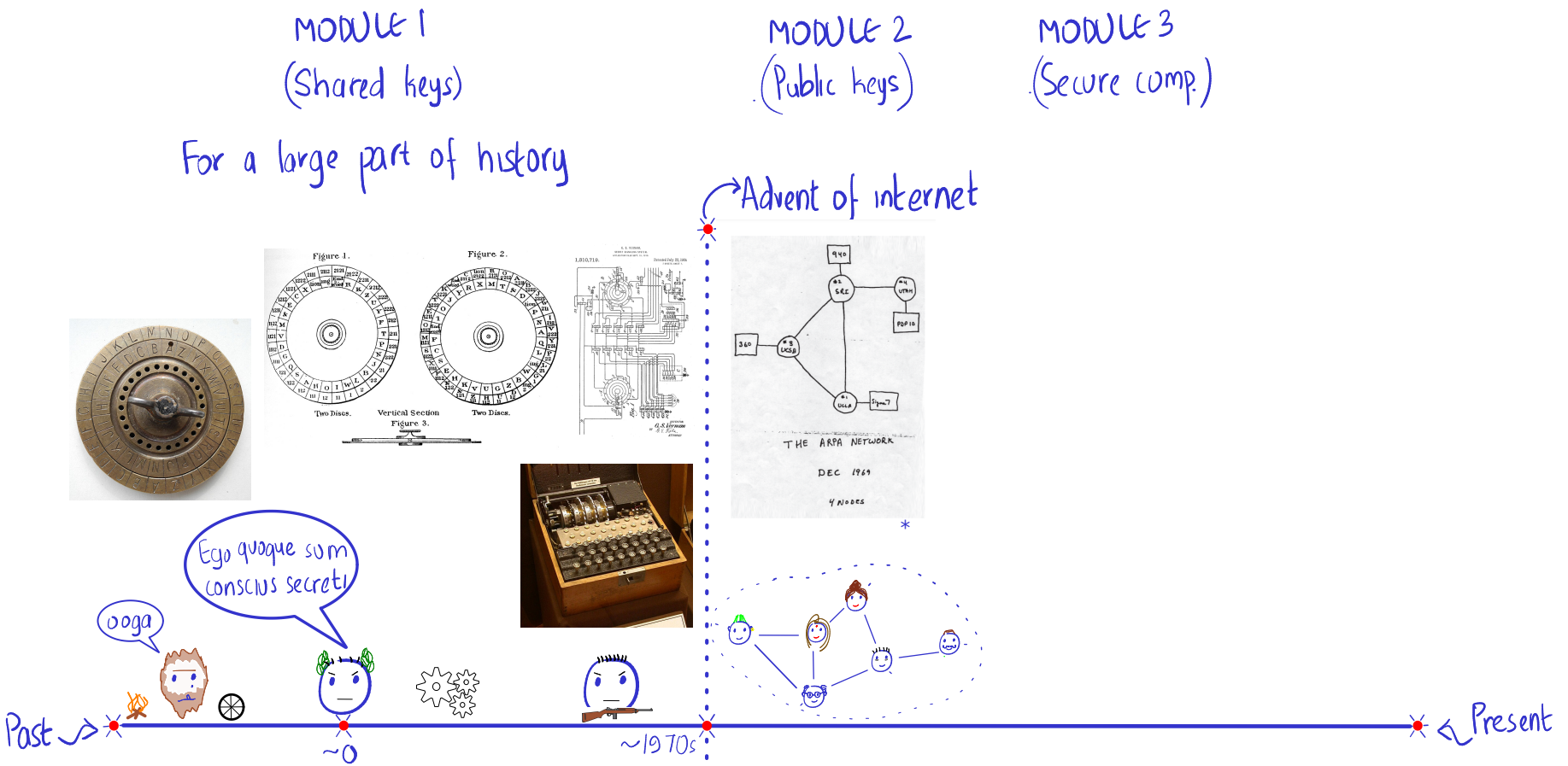
- Roughly corresponds to four "eras"



Credit for images: Wikipedia (*User:DARPA)

This Lecture: An Overview of the Tasks Covered...

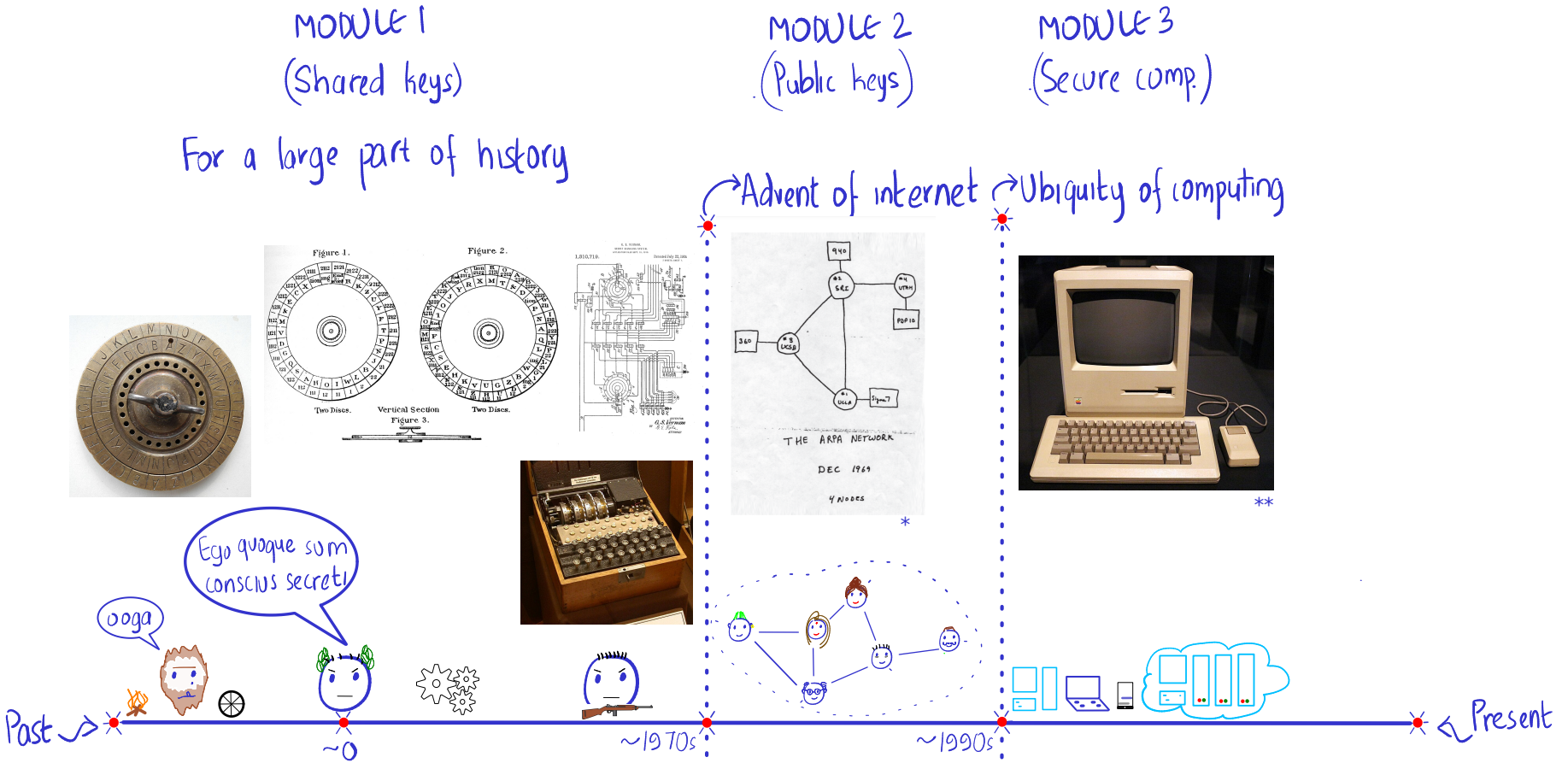
- Roughly corresponds to four "eras"



Credit for images: Wikipedia (*User:DARPA)

This Lecture: An Overview of the Tasks Covered...

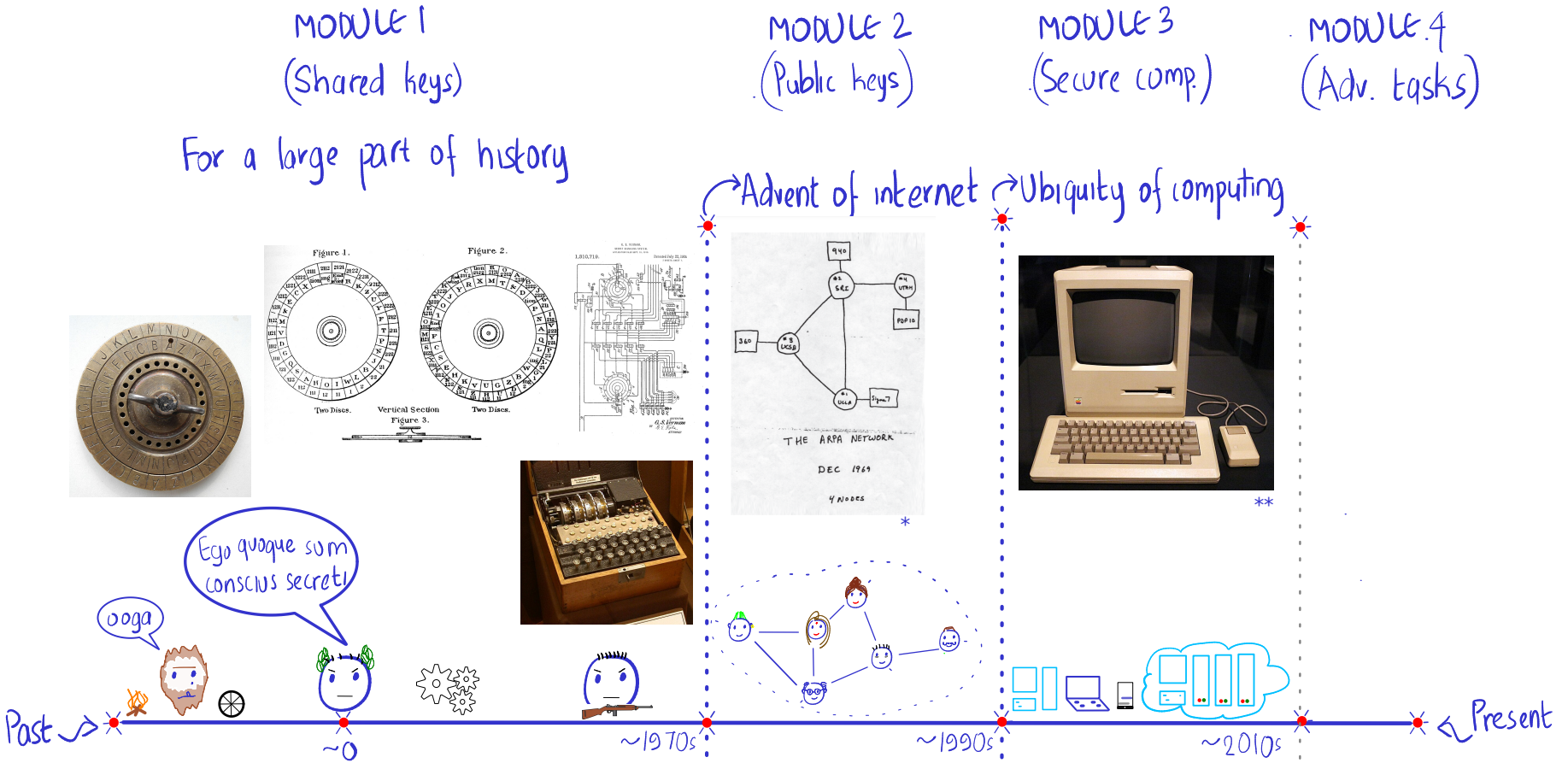
- Roughly corresponds to four "eras"



Credit for images: Wikipedia (*User:DARPA)(**User:Saliko)

This Lecture: An Overview of the Tasks Covered...

- Roughly corresponds to four "eras"



Credit for images: Wikipedia (*User:DARPA)(**User:Saliko)

An Overview of the Course

1. Module I: Secure Communication in Shared-Key Setting
2. Module II: Secure Communication in Public-Key Setting
3. Module III: Secure Computation
4. Module IV: Some Advanced Tasks

The Universal Need for Private Communication

MODULE 1
(Shared keys)

For a large part of history

Credit for images: Wikipedia

The Universal Need for Private Communication

MODULE 1
(Shared keys)

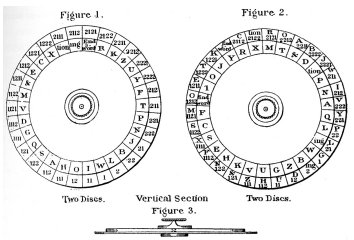
For a large part of history

Credit for images: Wikipedia

The Universal Need for Private Communication

MODULE 1
(Shared keys)

For a large part of history

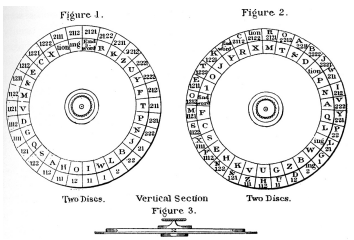


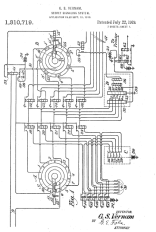
Credit for images: Wikipedia

The Universal Need for Private Communication

MODULE 1
(Shared keys)

For a large part of history



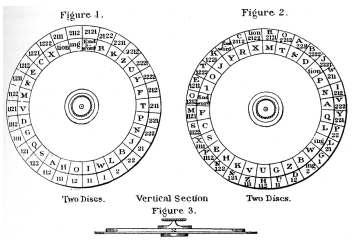


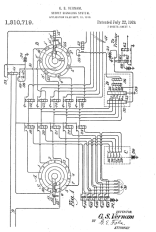
Credit for images: Wikipedia

The Universal Need for Private Communication

MODULE 1
(Shared keys)

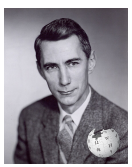
For a large part of history





Birth of information theory & CS

Foundations of modern cryptography

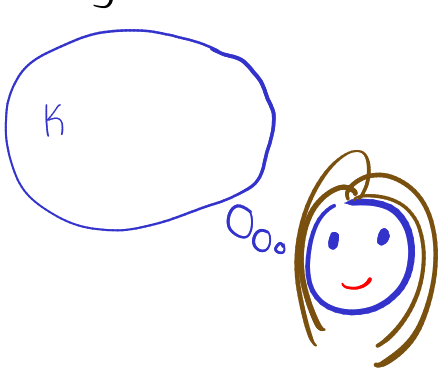


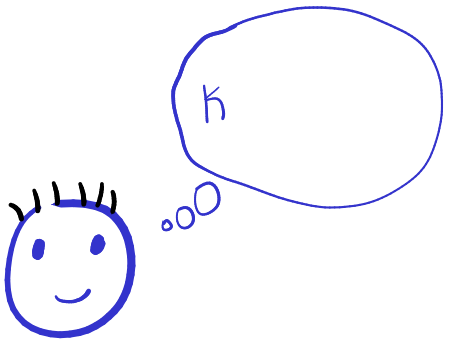
Credit for images: Wikipedia

Credit: Tekniska Museet

Task 1: Secret Communication with Shared Keys...

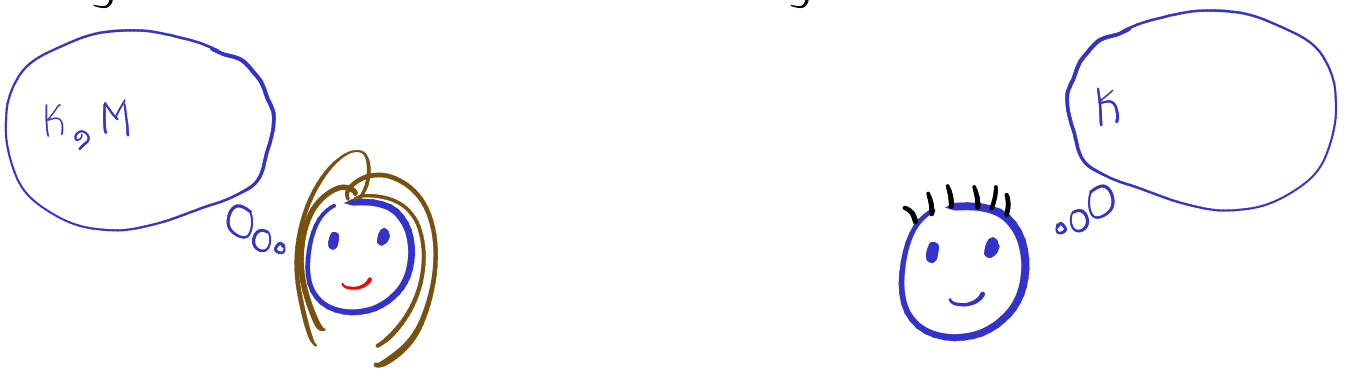
- Setting: Alice and Bob share a key `K` and want to communicate





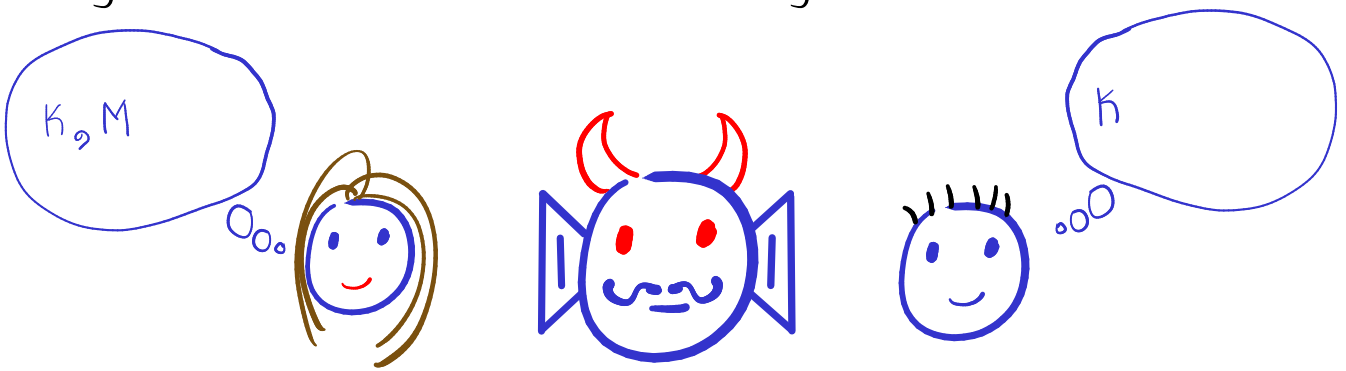
Task 1: Secret Communication with Shared Keys...

- Setting: Alice and Bob share a key $\mathtt{K}$ and want to communicate



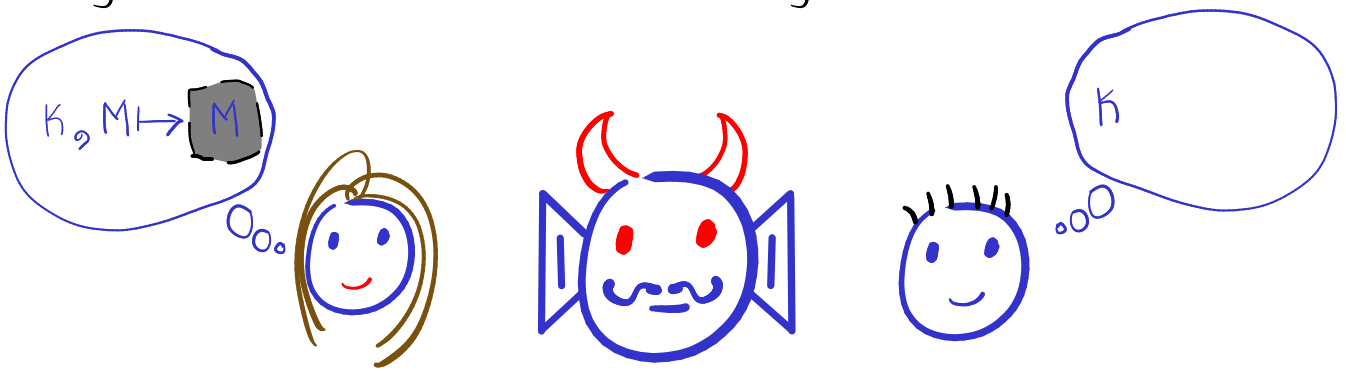
Task 1: Secret Communication with Shared Keys...

- Setting: Alice and Bob share a key $\mathtt{K}$ and want to communicate



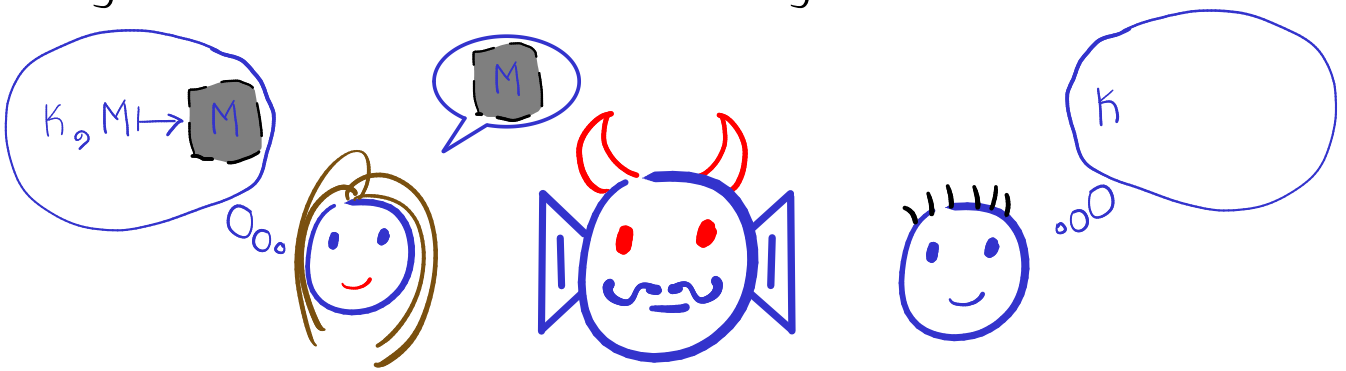
Task 1: Secret Communication with Shared Keys...

- Setting: Alice and Bob share a key $\mathtt{K}$ and want to communicate



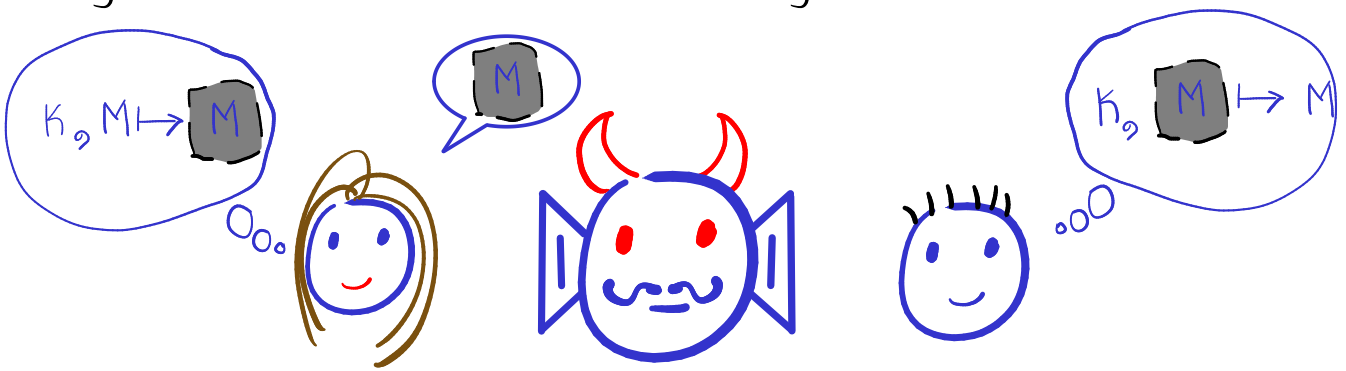
Task 1: Secret Communication with Shared Keys...

- Setting: Alice and Bob share a key $\mathtt{K}$ and want to communicate



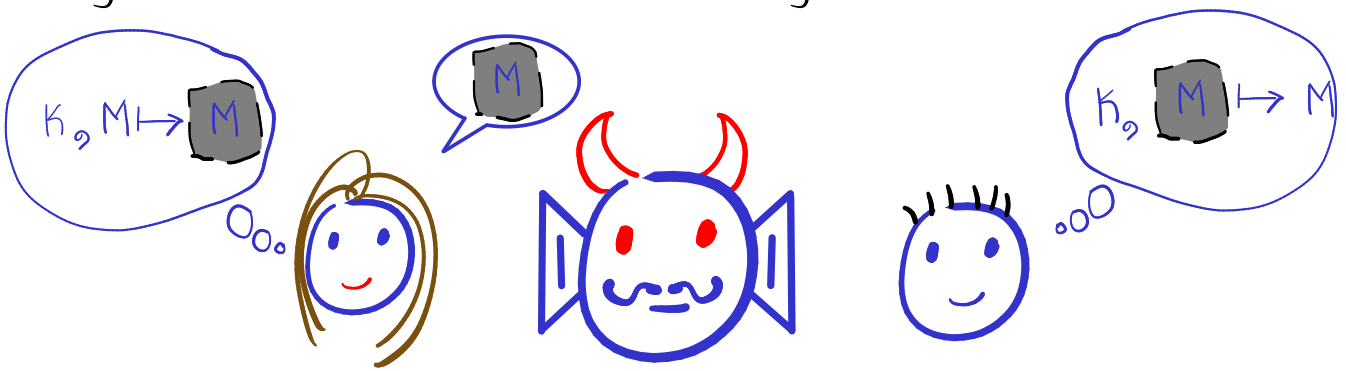
Task 1: Secret Communication with Shared Keys...

- Setting: Alice and Bob share a key $\mathtt{K}$ and want to communicate



Task 1: Secret Communication with Shared Keys...

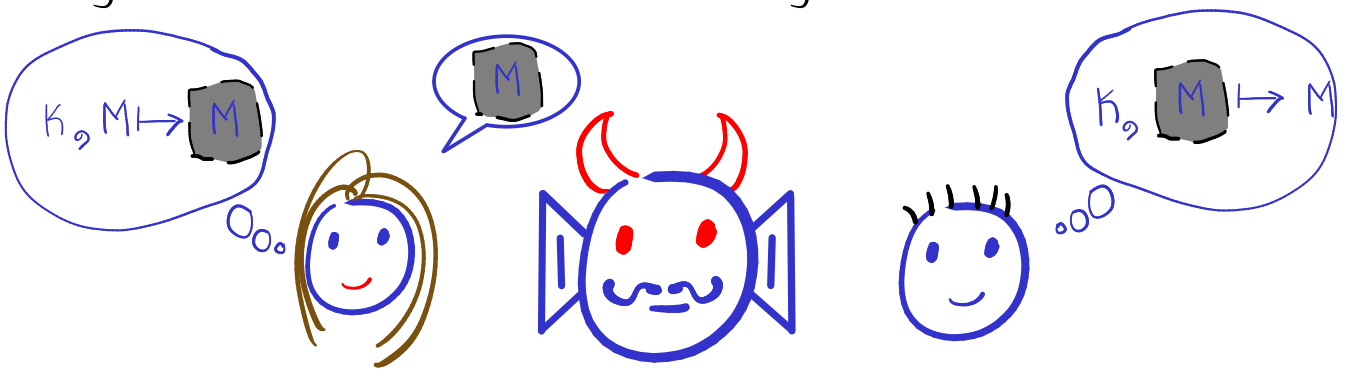
- Setting: Alice and Bob share a key $\mathtt{K}$ and want to communicate



- Threat model: *Perfect secrecy*
  - Adversary: *All powerful* eavesdropper Eve (but doesn't know $\mathtt{K}$)
  - Security goal: Eve learns *no* information about the message M

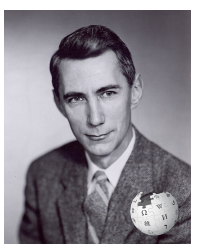
Task 1: Secret Communication with Shared Keys...

- Setting: Alice and Bob share a key $\mathtt{K}$ and want to communicate



- Threat model: *Perfect secrecy*
  - Adversary: *All powerful* eavesdropper Eve (but doesn't know $\mathtt{K}$)
  - Security goal: Eve learns *no* information about the message M

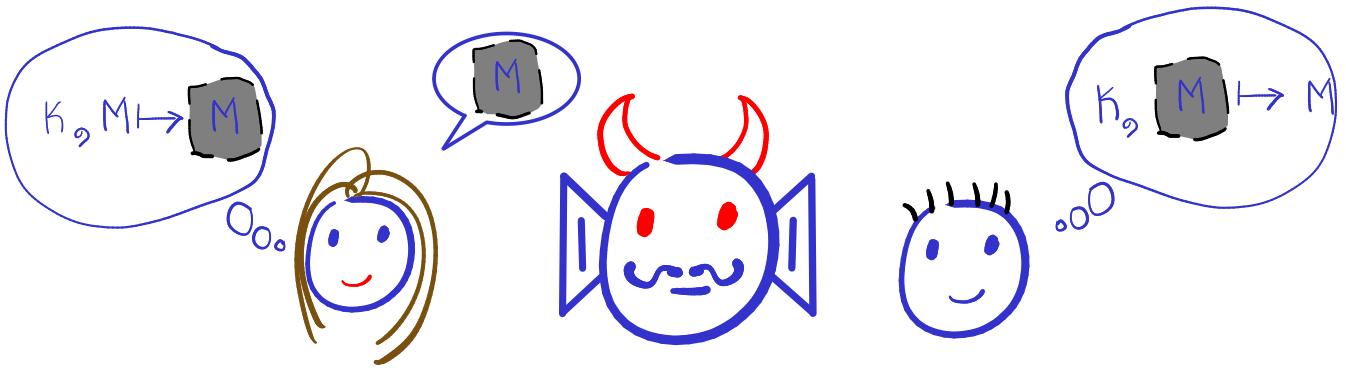
- What we will learn:
  - Classical ciphers, and why they are not perfectly secure
  - One-time pad (OTP), and why it is perfectly secret
  - Shannon's impossibility: for perfect secrecy, $|\mathtt{K}| \geq |M|$



Credit: Tekniska Museet

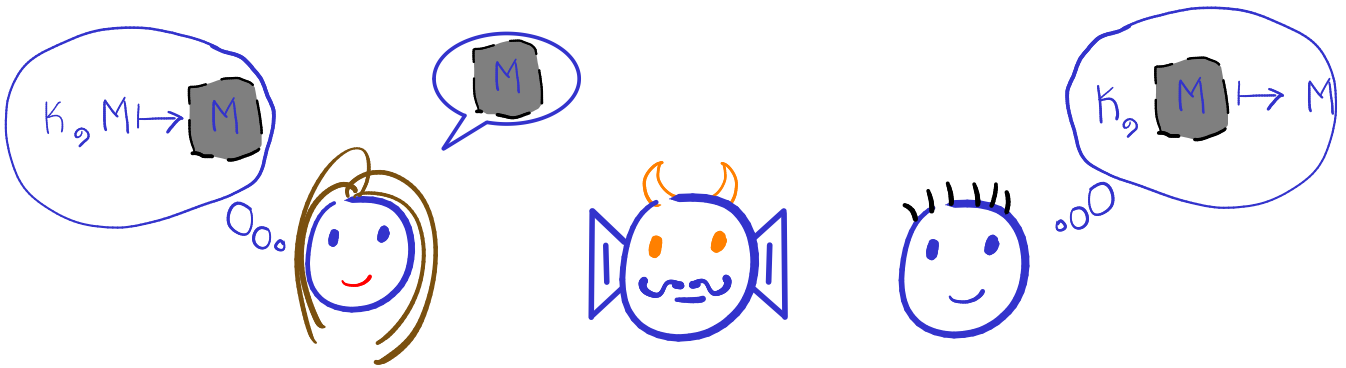
Task 1: Secret Communication with Shared Keys...

- How to overcome Shannon's impossibility?



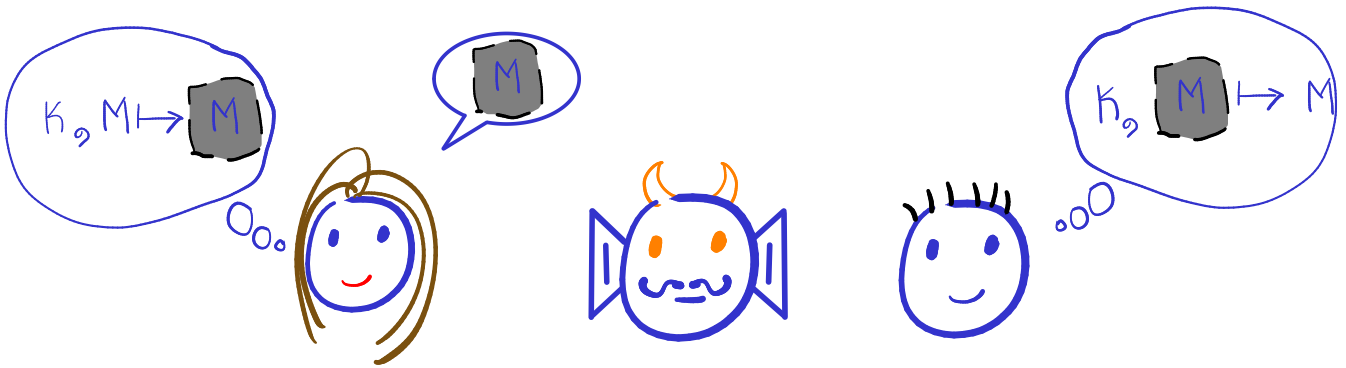
Task 1: Secret Communication with Shared Keys...

- How to overcome Shannon's impossibility?
  - *Restrict/bound* the adversary's computational capabilities



Task 1: Secret Communication with Shared Keys...

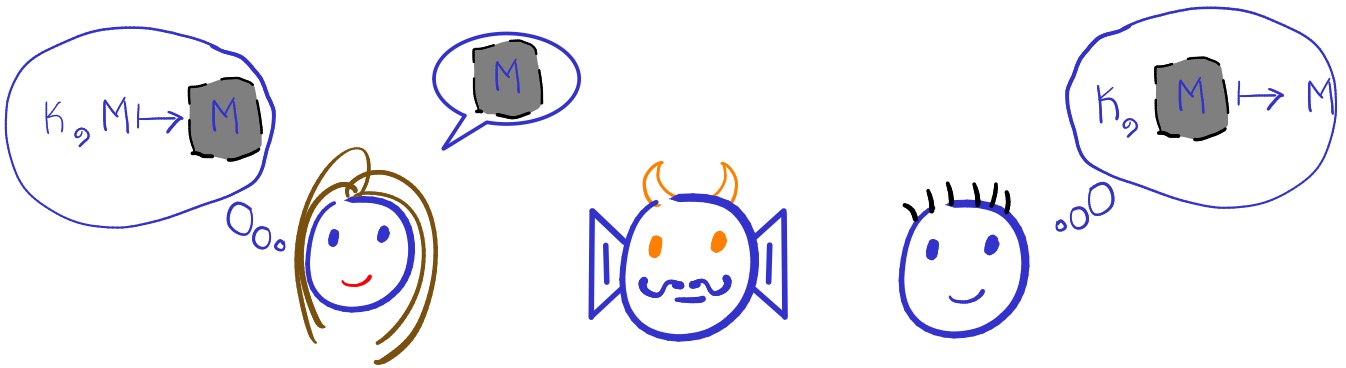
- How to overcome Shannon's impossibility?
  - *Restrict/bound* the adversary's computational capabilities



- Threat model: *Computational* secrecy
  - Adversary: *Computationally-bounded* eavesdropper Eve
  - Security goal: Eve learns *"no"* information about the message M

Task 1: Secret Communication with Shared Keys...

- How to overcome Shannon's impossibility?
  - *Restrict/bound* the adversary's computational capabilities

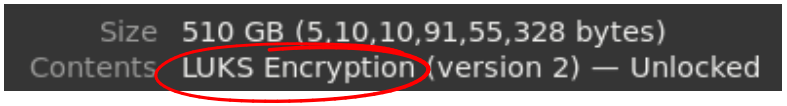


- Threat model: *Computational* secrecy
  - Adversary: *Computationally-bounded* eavesdropper Eve
  - Security goal: Eve learns *"no"* information about the message M

- What we will learn:
  - How to model computationally-bounded adversaries?
  - Hardness assumption: pseudo-random generator, one-way func.
  - Secret communication with $|\mathtt{K}| < |M|$ assuming PRG

Dealing with More Resourceful Adversaries

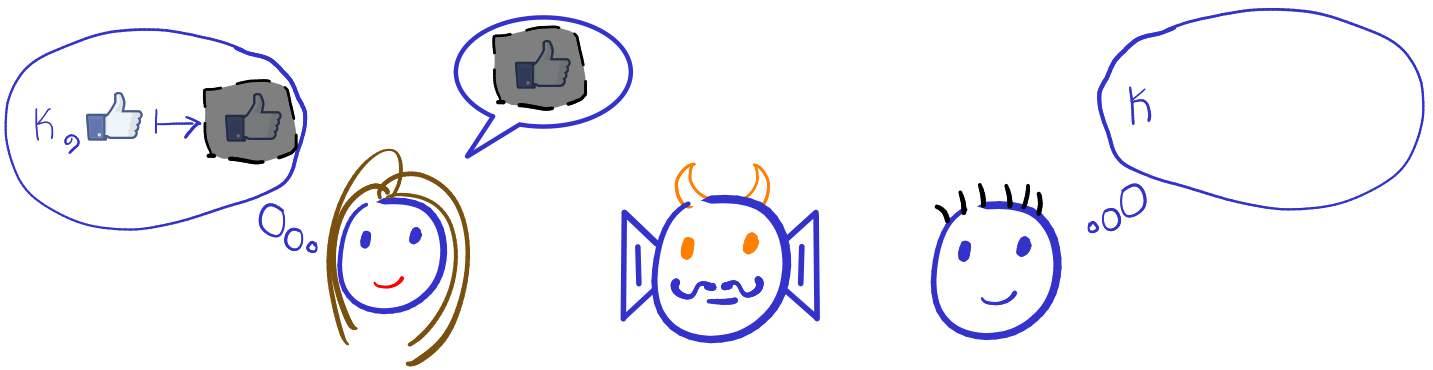
- What if Eve *also* has control over the messages?
- What we will learn: *chosen-plaintext attack* (CPA) and CPA-secure scheme from pseudo-random functions



Dealing with More Resourceful Adversaries

- What if Eve *also* has control over the messages?
- What we will learn: *chosen-plaintext attack* (CPA) and CPA-secure scheme from pseudo-random functions

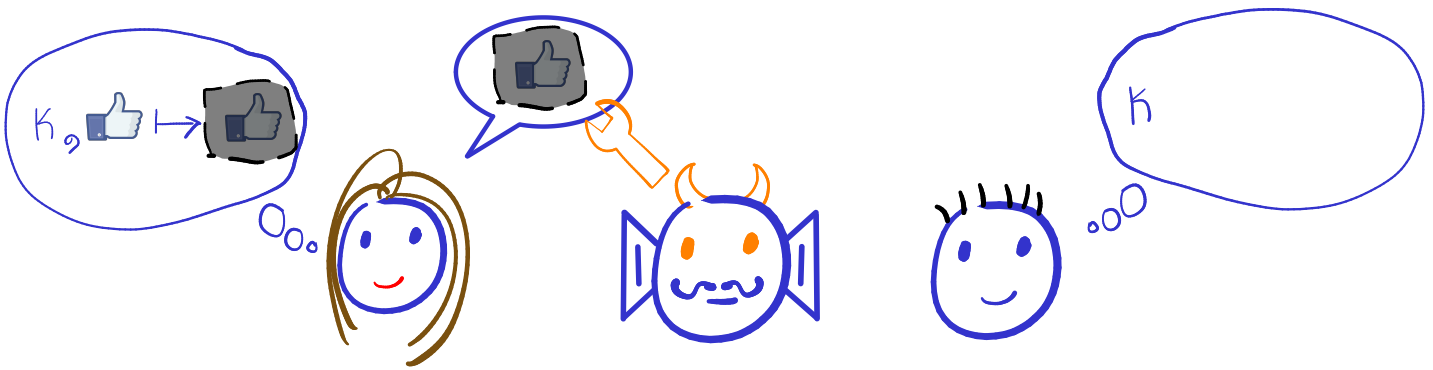
- What if Eve can also *tamper* with the communication?
- What we will learn: message authentication codes



Dealing with More Resourceful Adversaries

- What if Eve *also* has control over the messages?
- What we will learn: *chosen-plaintext attack* (CPA) and CPA-secure scheme from pseudo-random functions

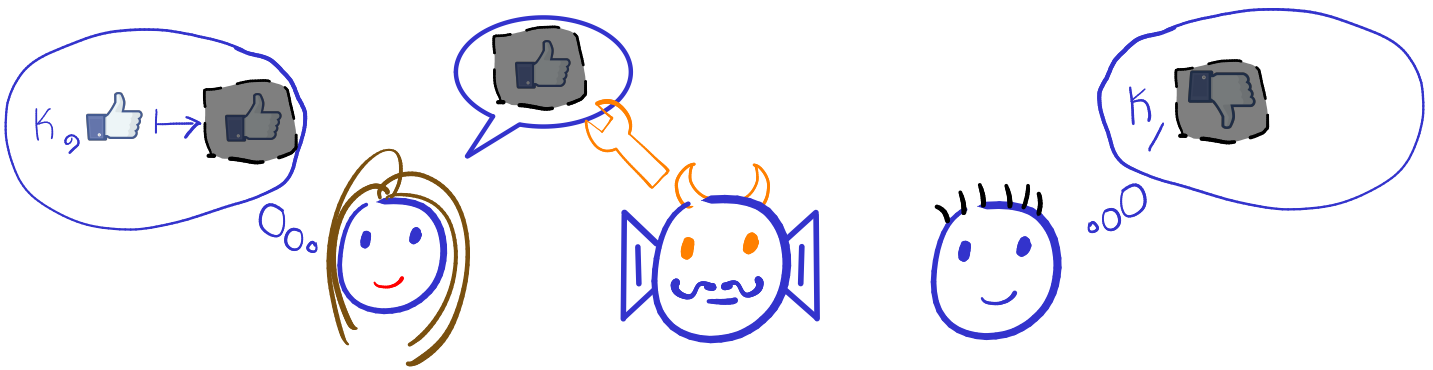
- What if Eve can also *tamper* with the communication?
- What we will learn: message authentication codes



Dealing with More Resourceful Adversaries

- What if Eve *also* has control over the messages?
- What we will learn: *chosen-plaintext attack* (CPA) and CPA-secure scheme from pseudo-random functions

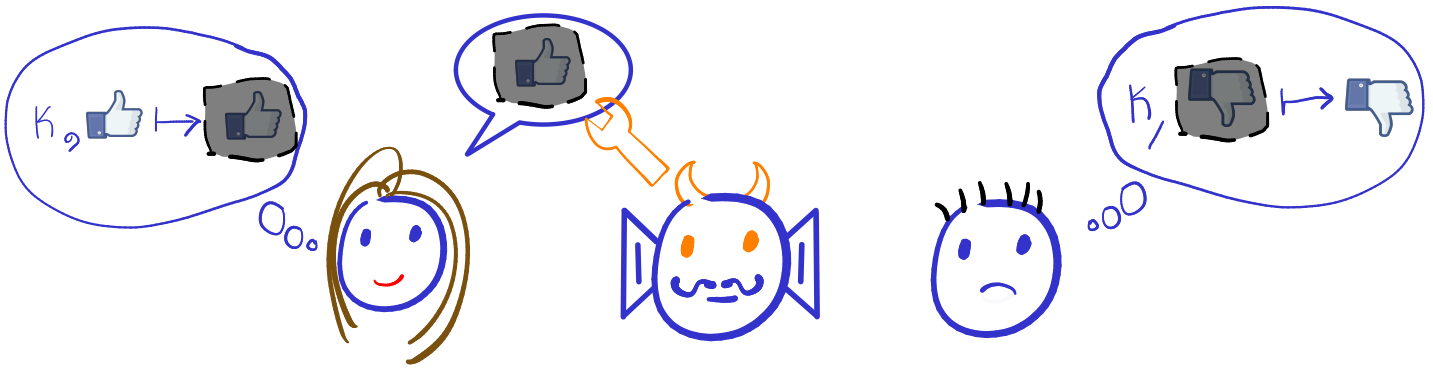
- What if Eve can also *tamper* with the communication?
- What we will learn: message authentication codes



Dealing with More Resourceful Adversaries

- What if Eve *also* has control over the messages?
- What we will learn: *chosen-plaintext attack* (CPA) and CPA-secure scheme from pseudo-random functions

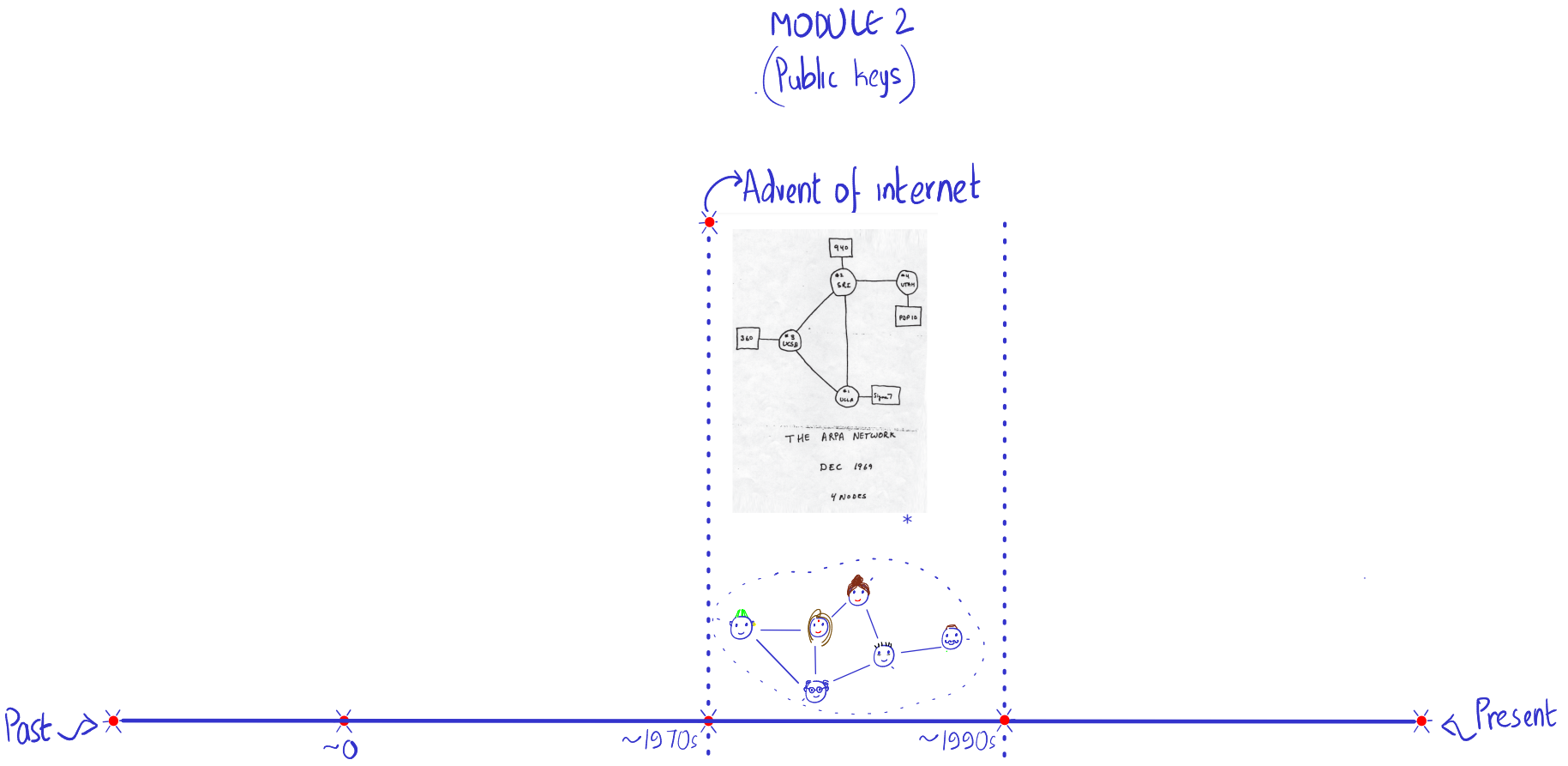
- What if Eve can also *tamper* with the communication?
- What we will learn: message authentication codes



An Overview of the Course

1. Module I: Secure Communication in Shared-Key Setting
2. Module II: Secure Communication in Public-Key Setting
3. Module III: Secure Computation
4. Module IV: Some Advanced Tasks

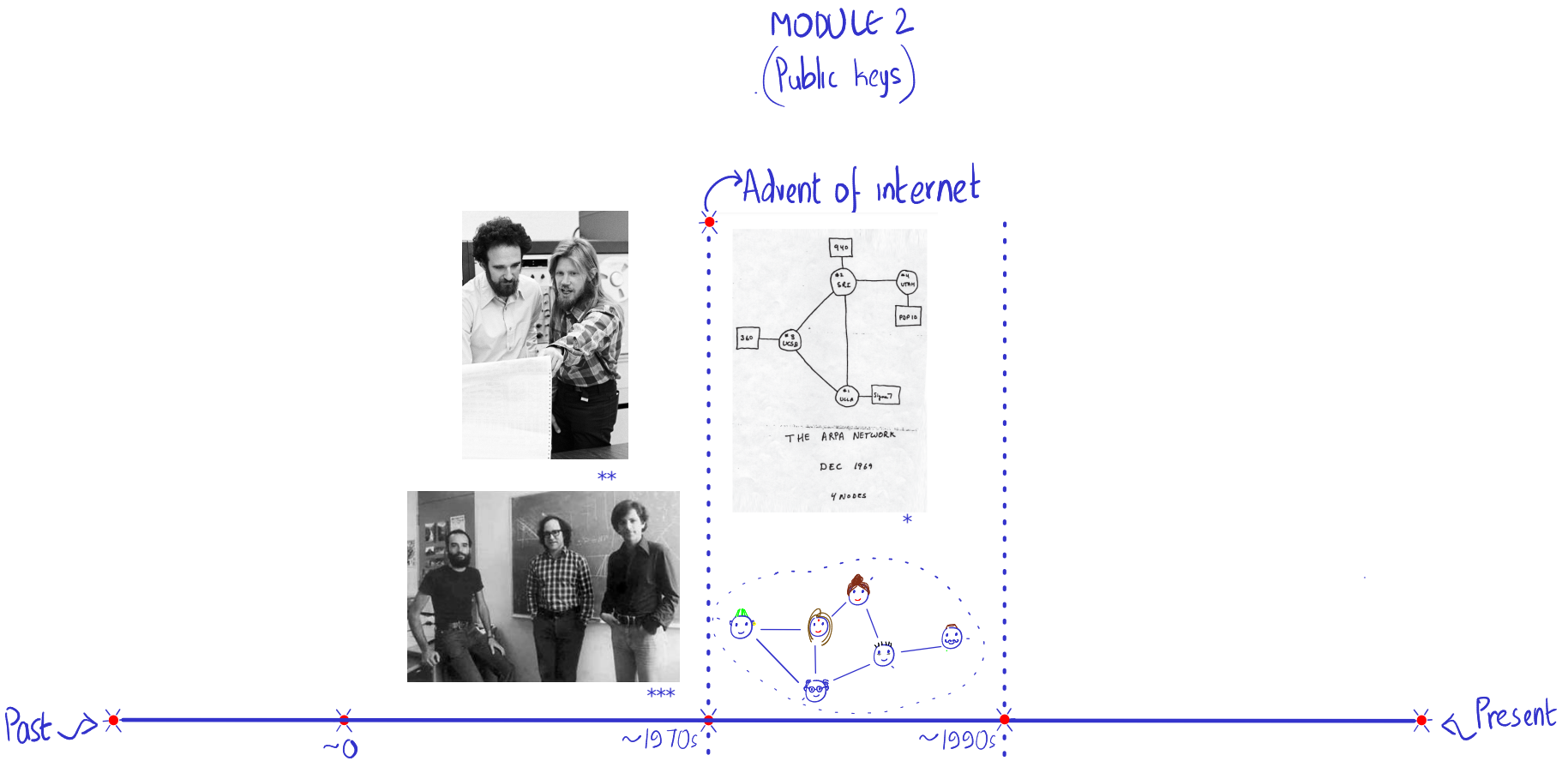
Advent of Internet and the Scaling Problem



- Limitation of shared-key encryption: requires prior meeting

Credit for images: (*Wikipedia User: DARPA)

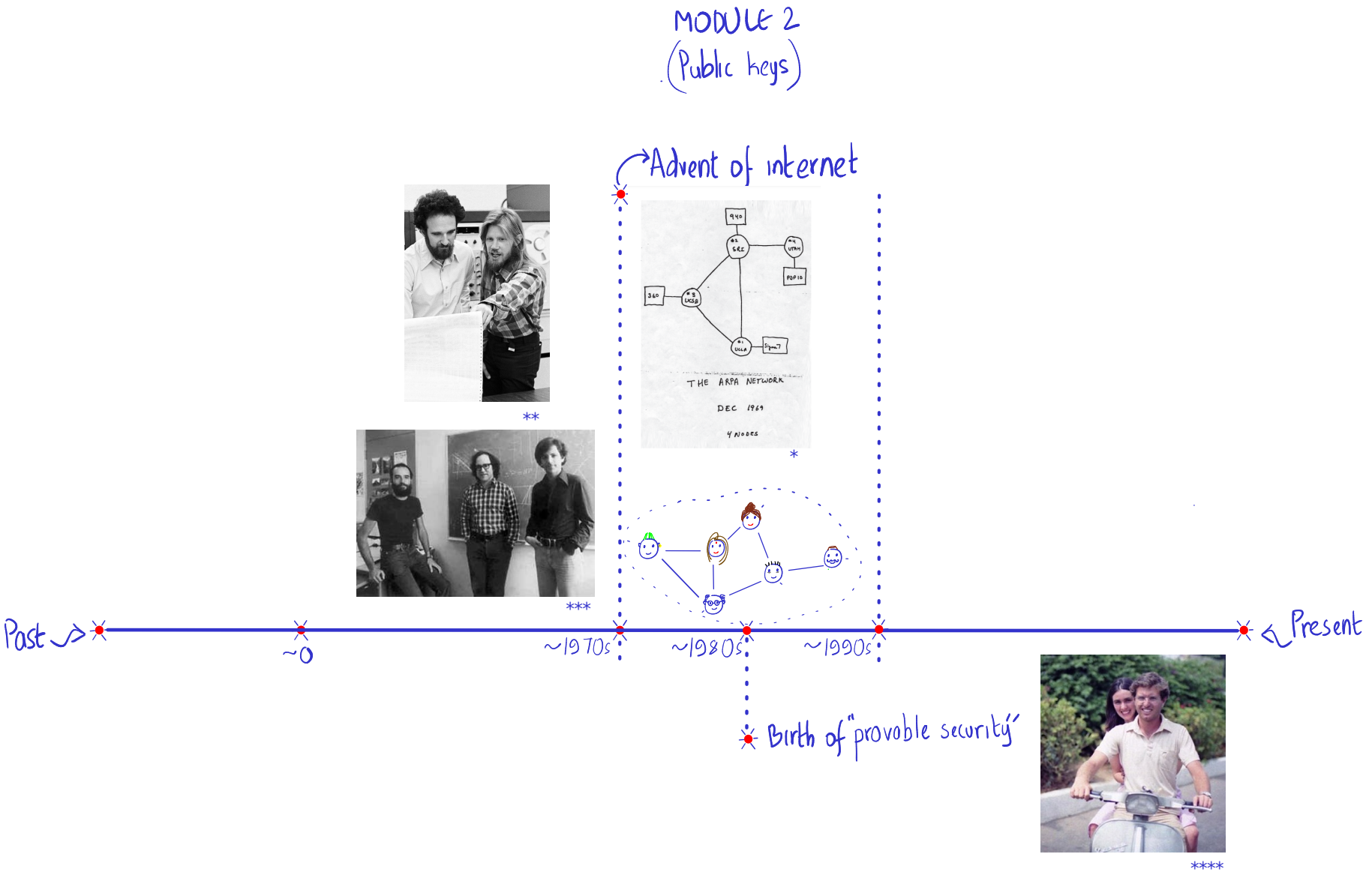
Advent of Internet and the Scaling Problem



- Limitation of shared-key encryption: requires prior meeting

Credit for images: (*Wikipedia User: DARPA) (**icour.fr) (***cs.miami.edu (Rosenberg))

Advent of Internet and the Scaling Problem

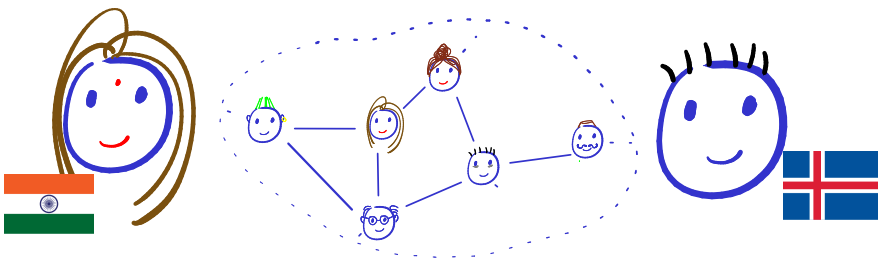


- Limitation of shared-key encryption: requires prior meeting

Credit for images: (*Wikipedia User: DARPA) (**icour.fr) (***cs.miami.edu (Rosenberg)) (****Oded Goldreich)

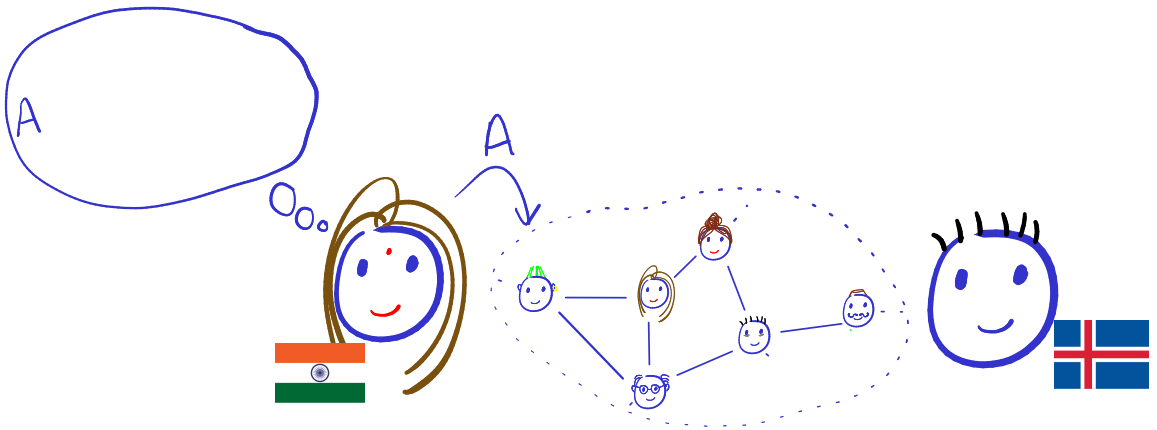
Task 2: Establishing a Shared Key

- Setting: Alice and Bob want to establish a shared key `K` by communicating *in public* (i.e., exchange a key)



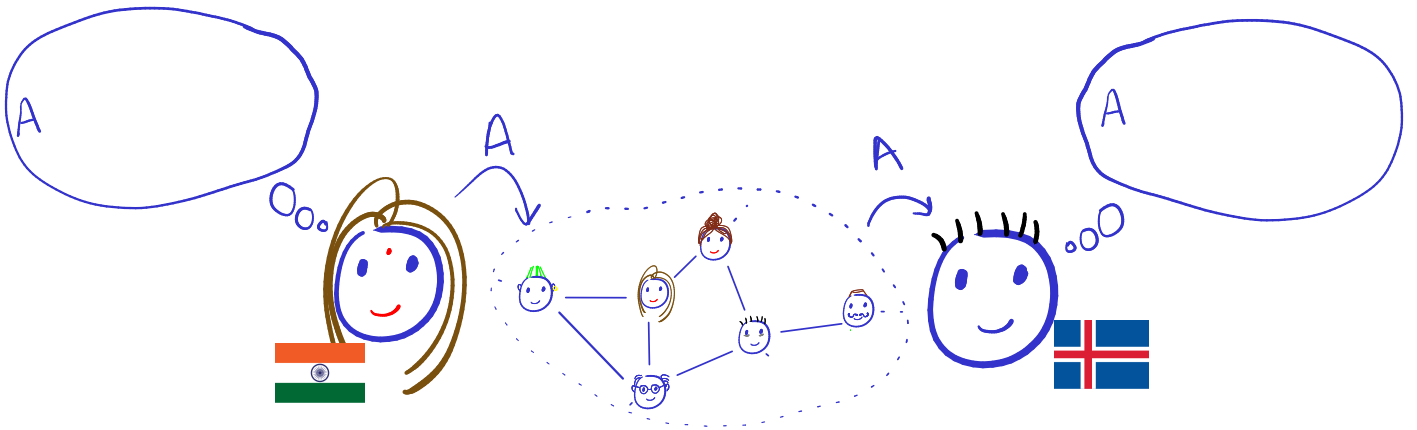
Task 2: Establishing a Shared Key

- Setting: Alice and Bob want to establish a shared key `K` by communicating *in public* (i.e., exchange a key)



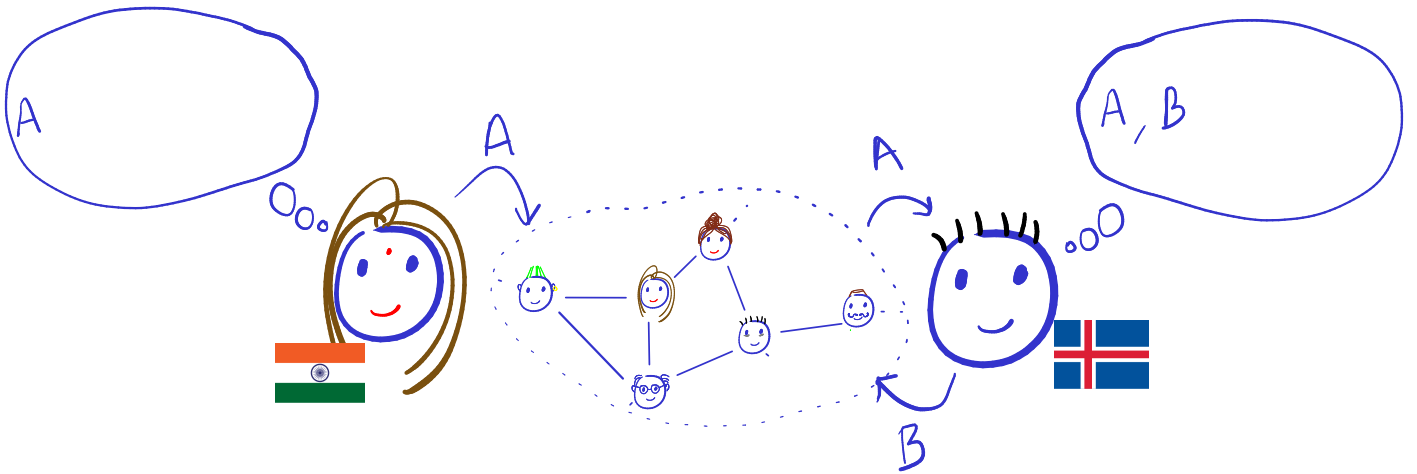
Task 2: Establishing a Shared Key

- Setting: Alice and Bob want to establish a shared key `K` by communicating *in public* (i.e., exchange a key)



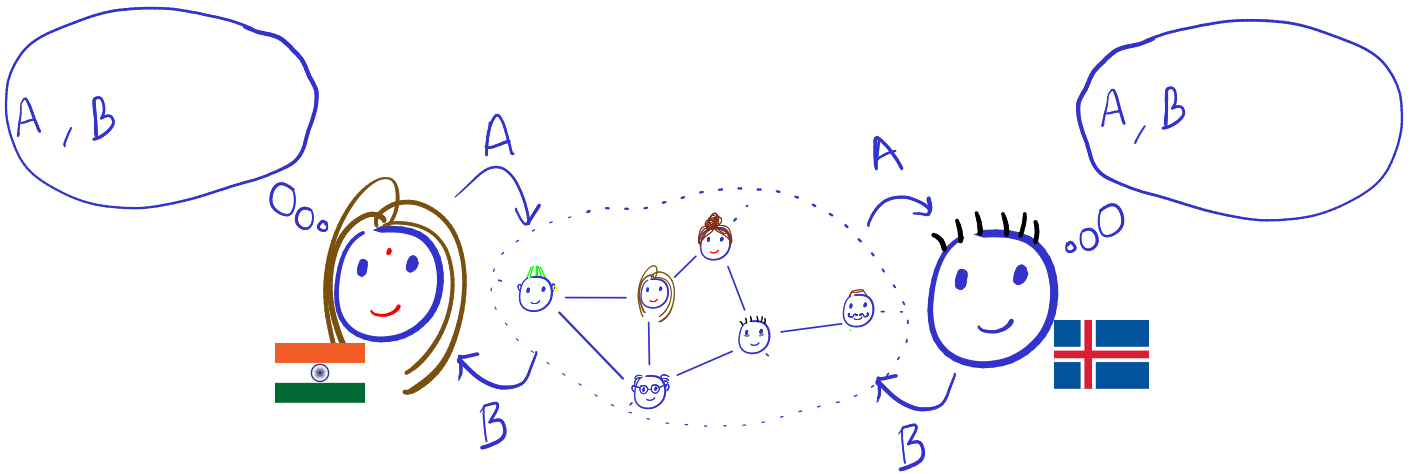
Task 2: Establishing a Shared Key

- Setting: Alice and Bob want to establish a shared key `K` by communicating *in public* (i.e., exchange a key)



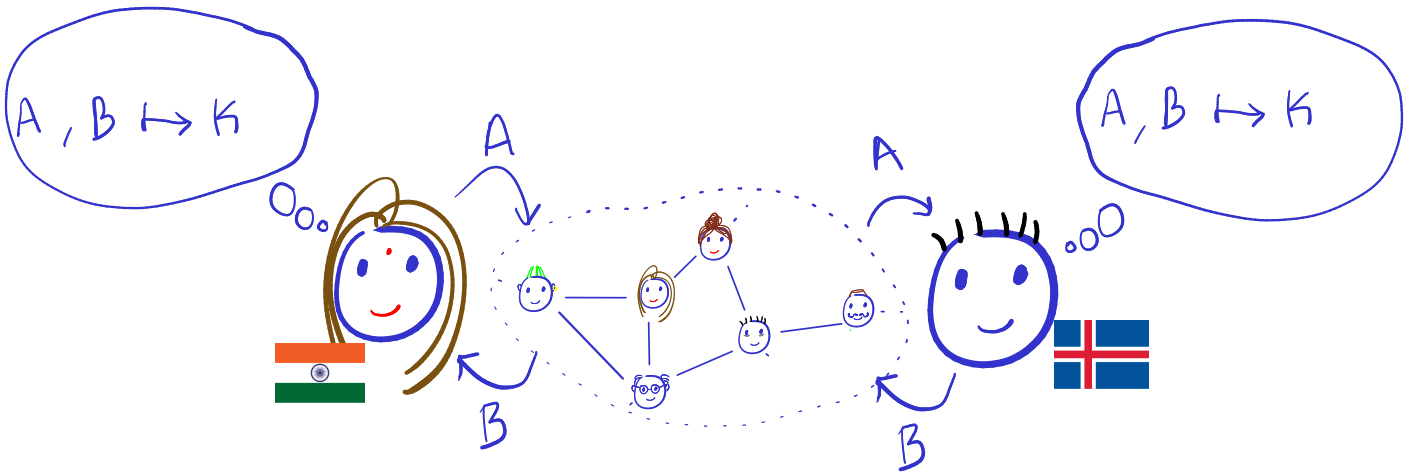
Task 2: Establishing a Shared Key

- Setting: Alice and Bob want to establish a shared key `K` by communicating *in public* (i.e., exchange a key)



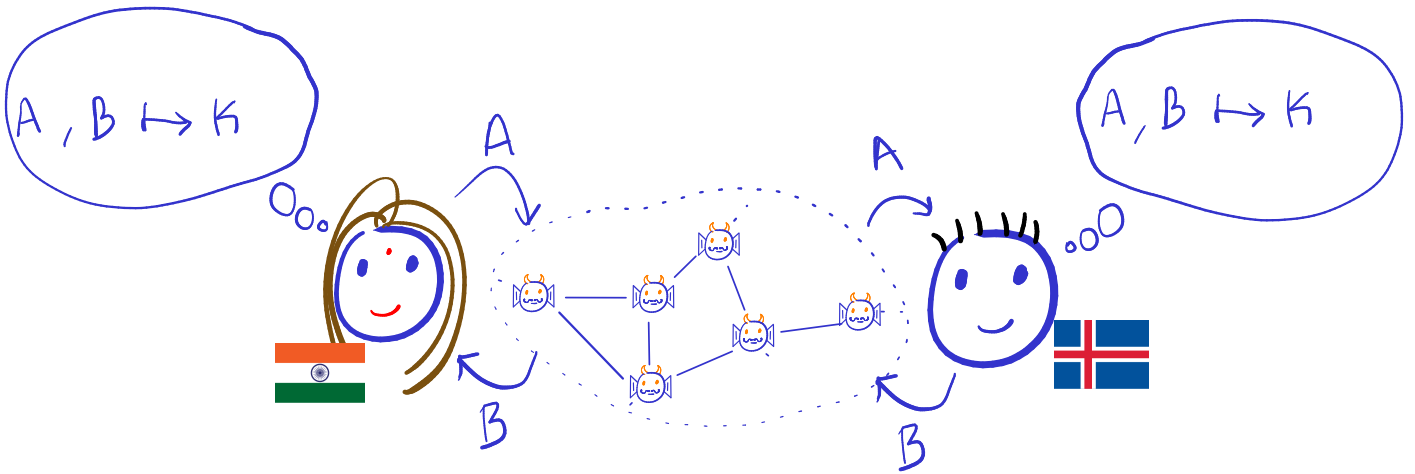
Task 2: Establishing a Shared Key

- Setting: Alice and Bob want to establish a shared key `K` by communicating *in public* (i.e., exchange a key)



Task 2: Establishing a Shared Key

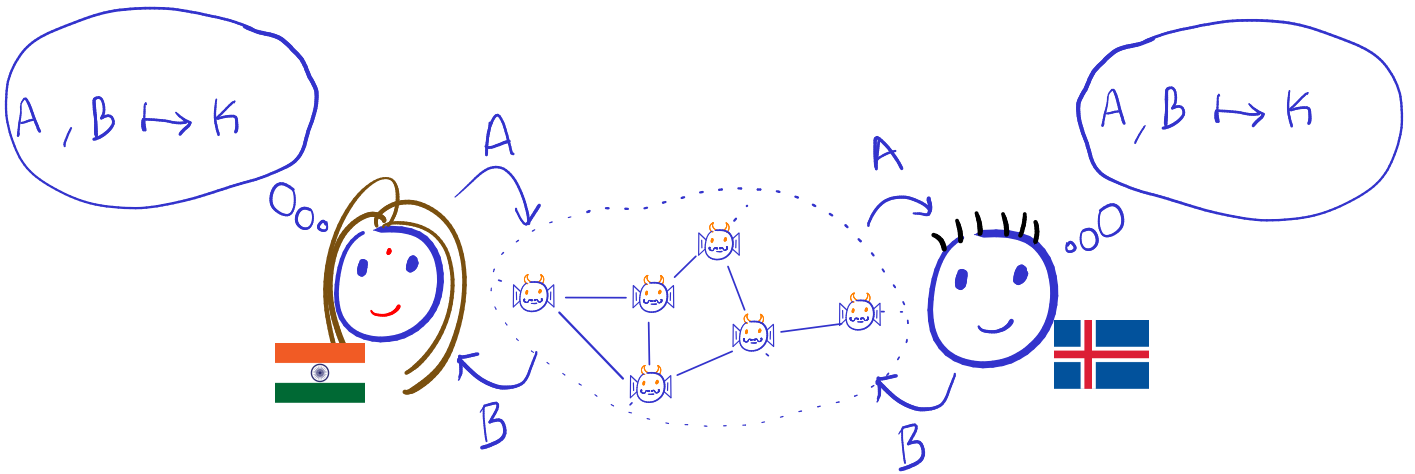
- Setting: Alice and Bob want to establish a shared key $\mathtt{K}$ by communicating *in public* (i.e., exchange a key)



- Threat model
  - Adversary: *Computationally-bounded* eavesdropper Eve
  - Security goal: Eve learns *"no"* information about the shared key

Task 2: Establishing a Shared Key

- Setting: Alice and Bob want to establish a shared key $\mathtt{K}$ by communicating *in public* (i.e., exchange a key)

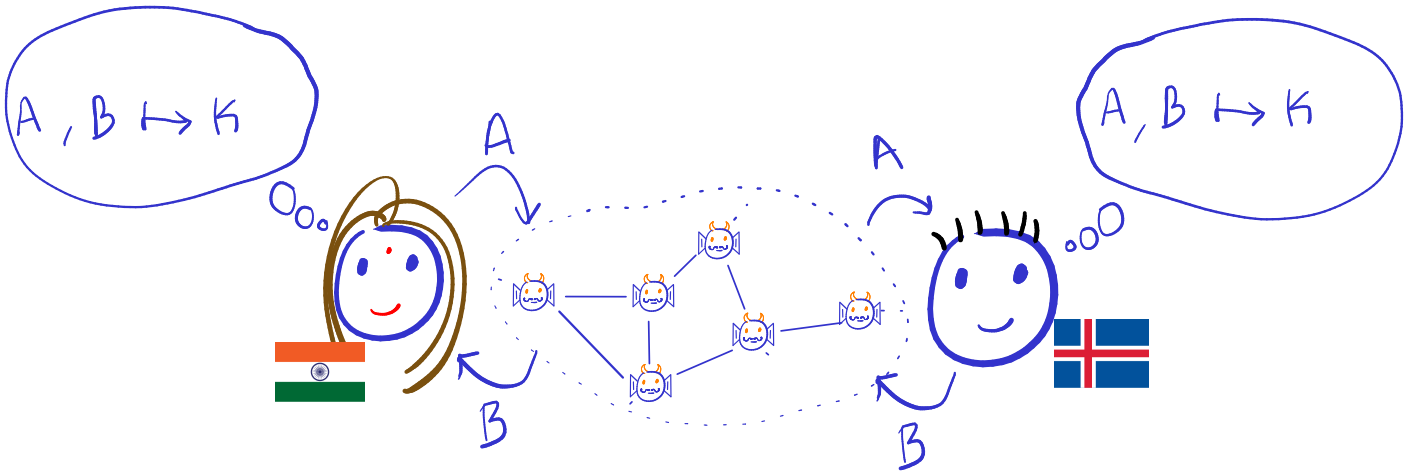


- Threat model
  - Adversary: *Computationally-bounded* eavesdropper Eve
  - Security goal: Eve learns *"no"* information about the shared key

- What we will learn:
  - Some group theory and number theory
  - Diffie-Hellman key exchange

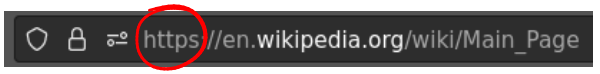
Task 2: Establishing a Shared Key

- Setting: Alice and Bob want to establish a shared key K by communicating *in public* (i.e., exchange a key)



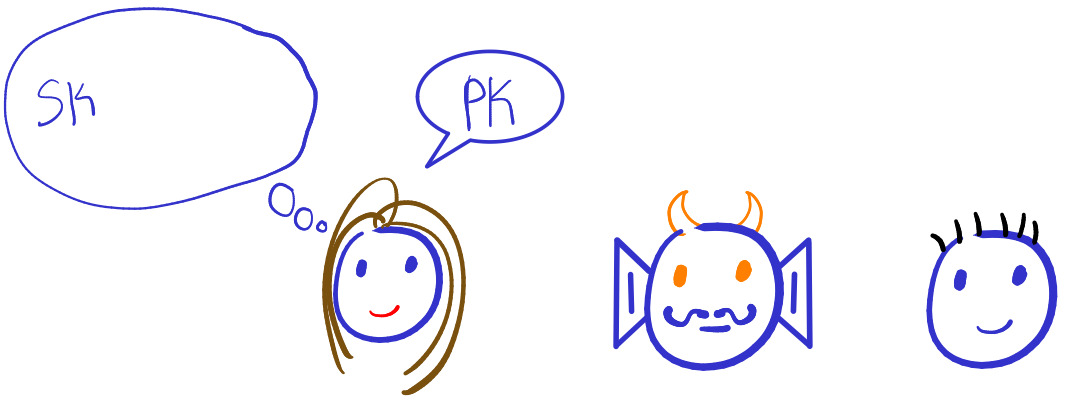
- Threat model
  - Adversary: *Computationally-bounded* eavesdropper Eve
  - Security goal: Eve learns *"no"* information about the shared key

- What we will learn:
  - Some group theory and number theory
  - Diffie-Hellman key exchange



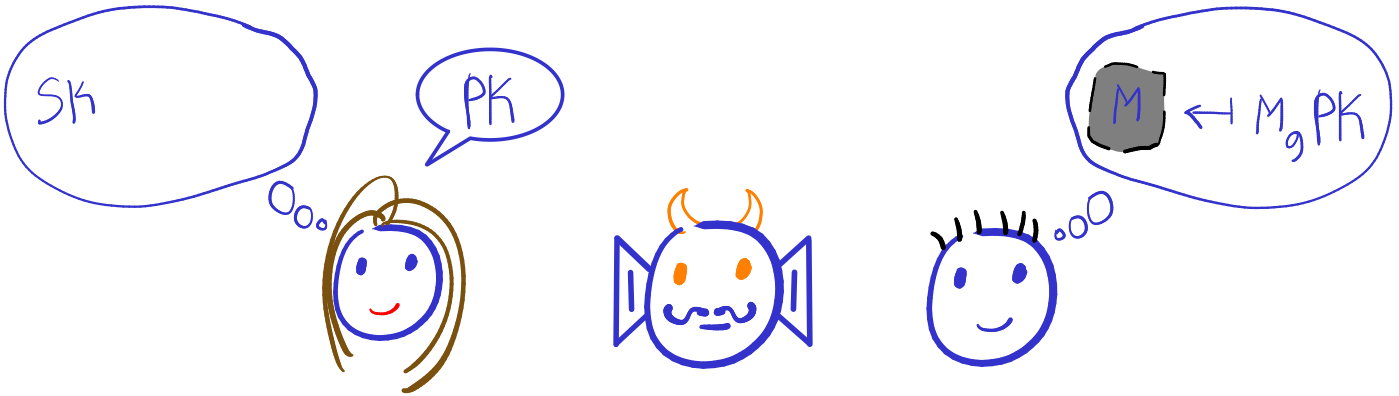
A Related Task: Secret Communication Using Public Keys

- Setting: Alice has published a public key `PK`, and Bob wants to send a secret message M to her



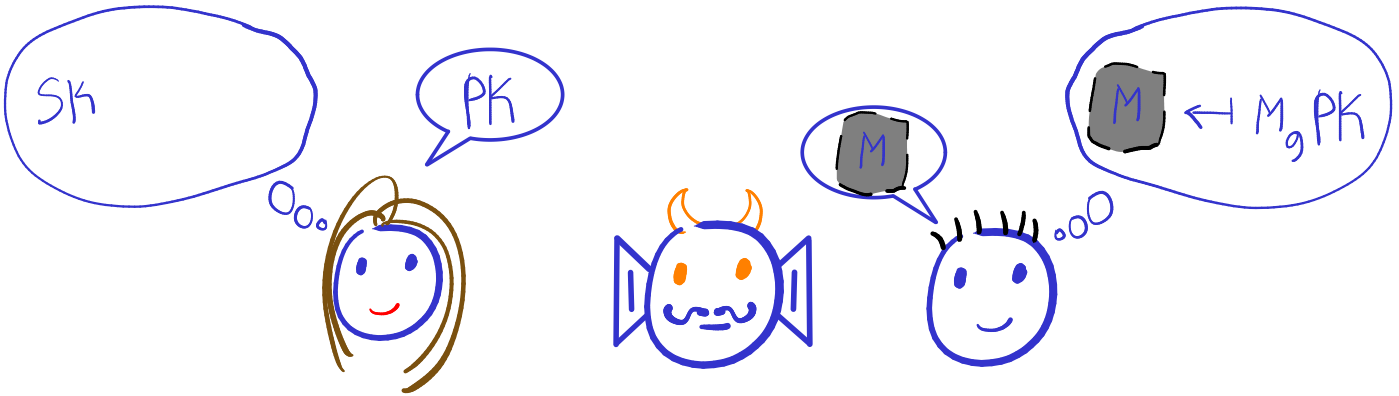
A Related Task: Secret Communication Using Public Keys

- Setting: Alice has published a public key `PK`, and Bob wants to send a secret message M to her



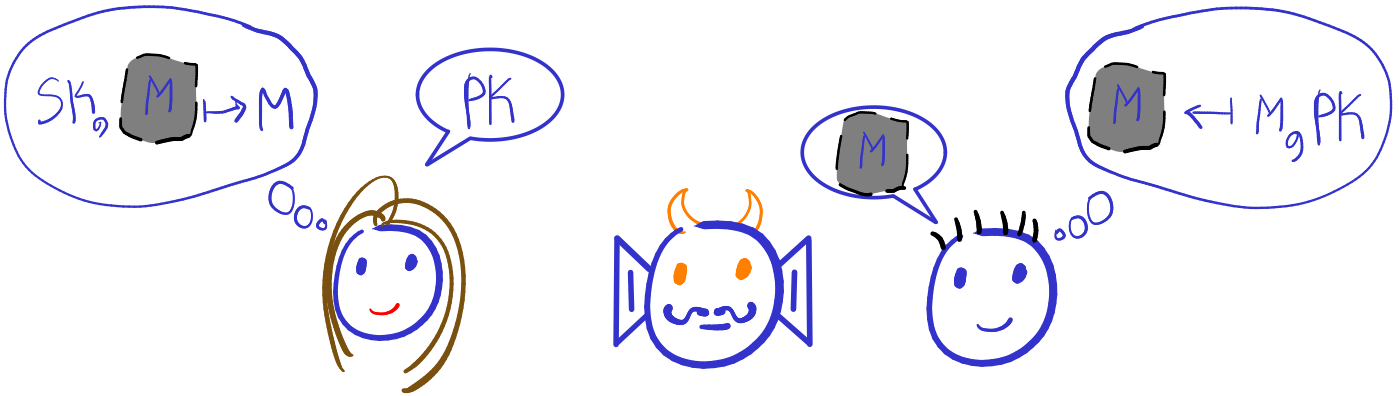
A Related Task: Secret Communication Using Public Keys

- Setting: Alice has published a public key `PK`, and Bob wants to send a secret message M to her



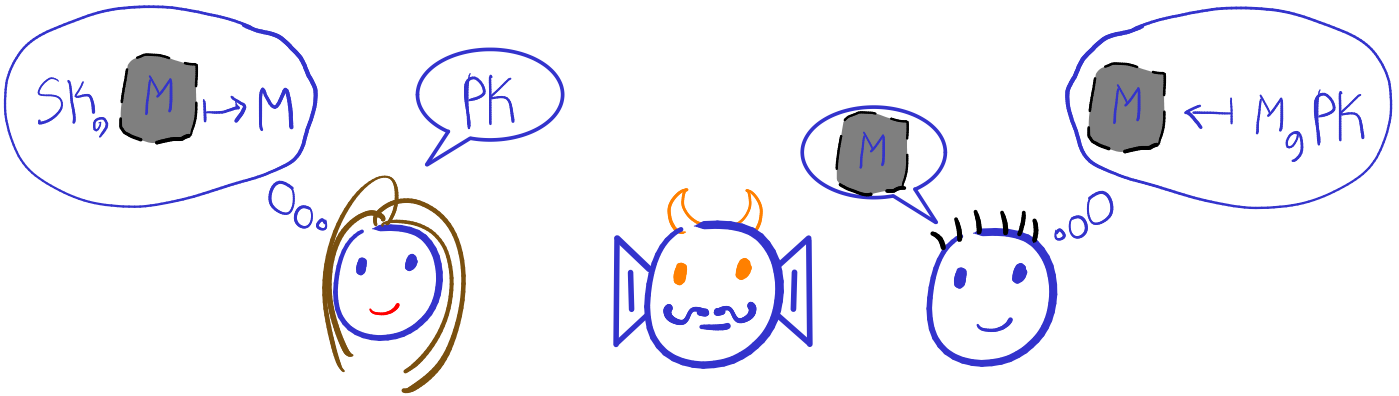
A Related Task: Secret Communication Using Public Keys

- Setting: Alice has published a public key `PK`, and Bob wants to send a secret message M to her



A Related Task: Secret Communication Using Public Keys

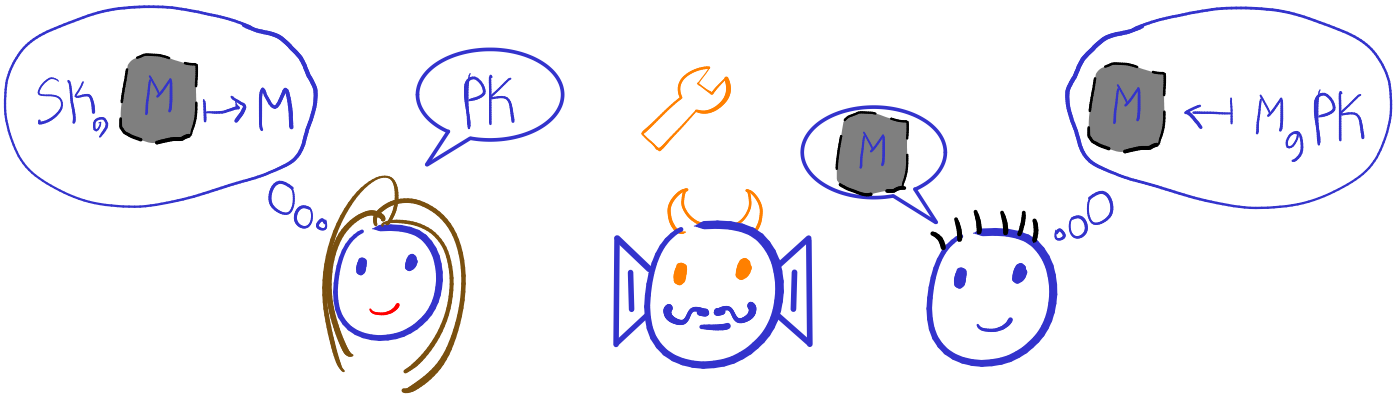
- Setting: Alice has published a public key $\mathtt{PK}$, and Bob wants to send a secret message M to her



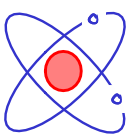
- What we will learn:
  - Public-key encryption (PKE) from trapdoor functions
  - Equivalence between PKE and key exchange
  - Learning with errors, PKE in presence of quantum adversary

A Related Task: Secret Communication Using Public Keys

- Setting: Alice has published a public key `PK`, and Bob wants to send a secret message M to her



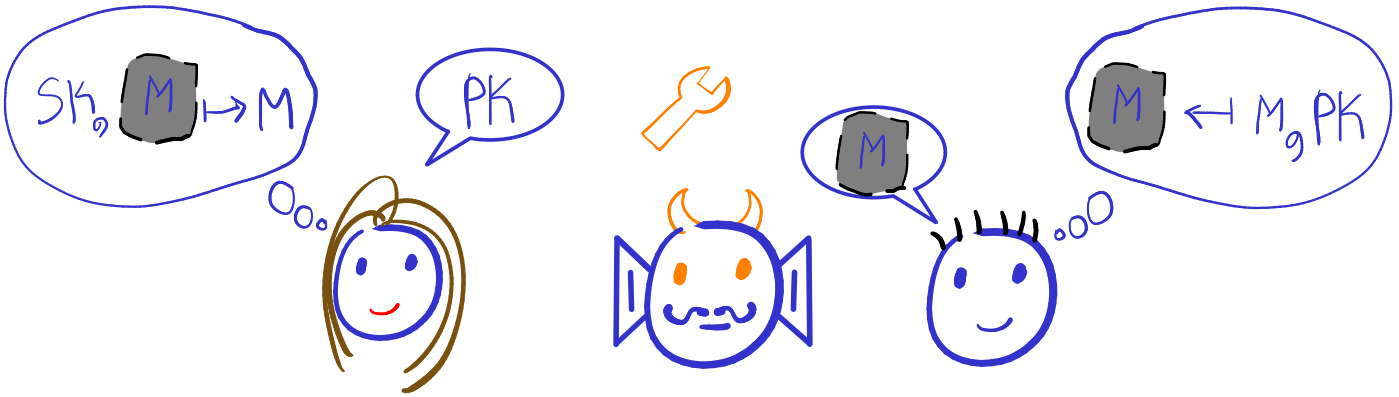
- What we will learn:
  - Public-key encryption (PKE) from trapdoor functions
  - Equivalence between PKE and key exchange
  - Learning with errors, PKE in presence of quantum adversary



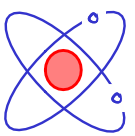
- How to deal with tampering adversary in public-key setting?
- What we will learn: digital signatures

A Related Task: Secret Communication Using Public Keys

- Setting: Alice has published a public key `PK`, and Bob wants to send a secret message M to her



- What we will learn:
  - Public-key encryption (PKE) from trapdoor functions
  - Equivalence between PKE and key exchange
  - Learning with errors, PKE in presence of quantum adversary

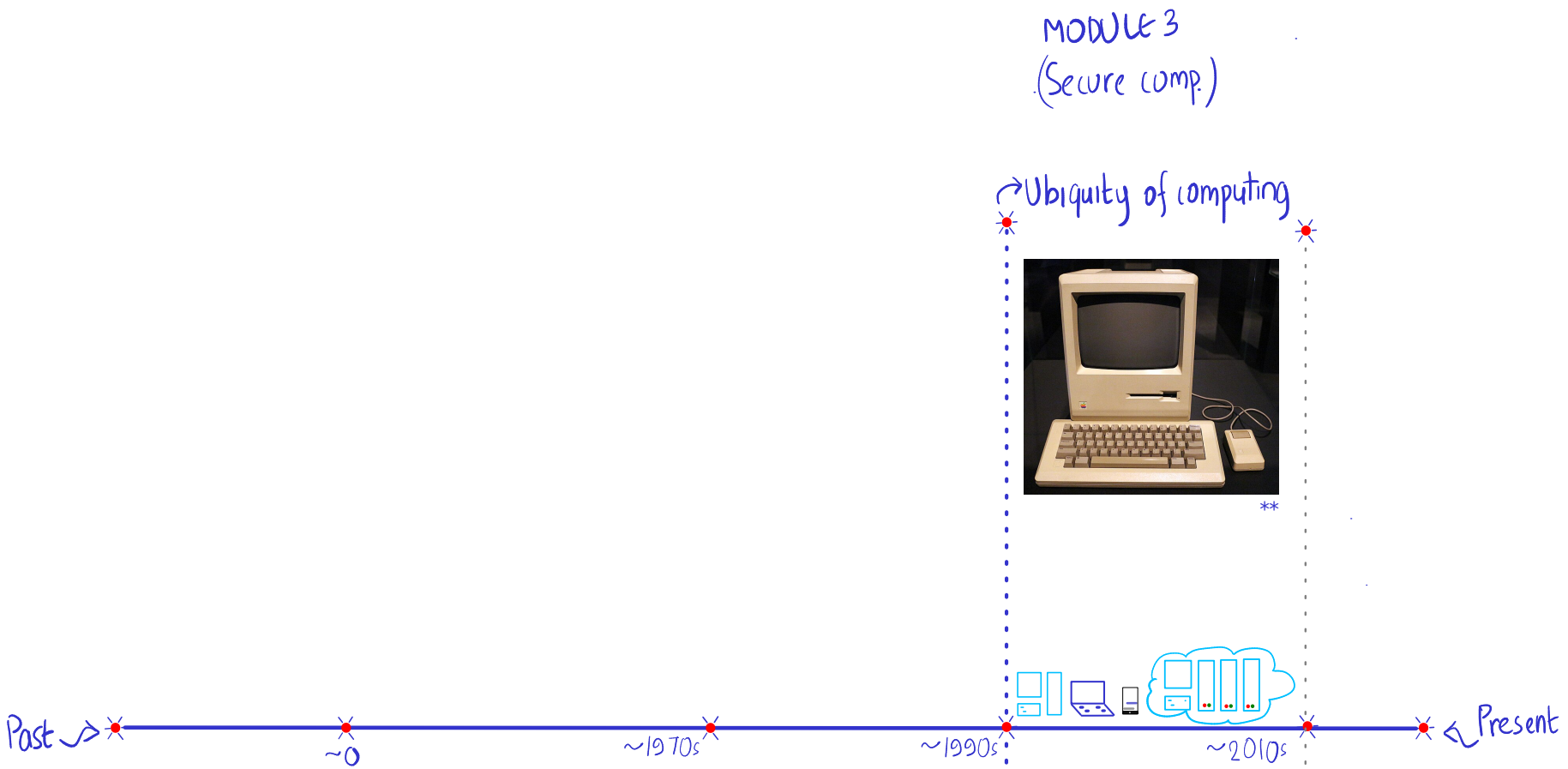


- How to deal with tampering adversary in public-key setting?
- What we will learn: digital signatures

An Overview of the Course

1. Module I: Secure Communication in Shared-Key Setting
2. Module II: Secure Communication in Public-Key Setting
3. Module III: Secure Computation
4. Module IV: Some Advanced Tasks

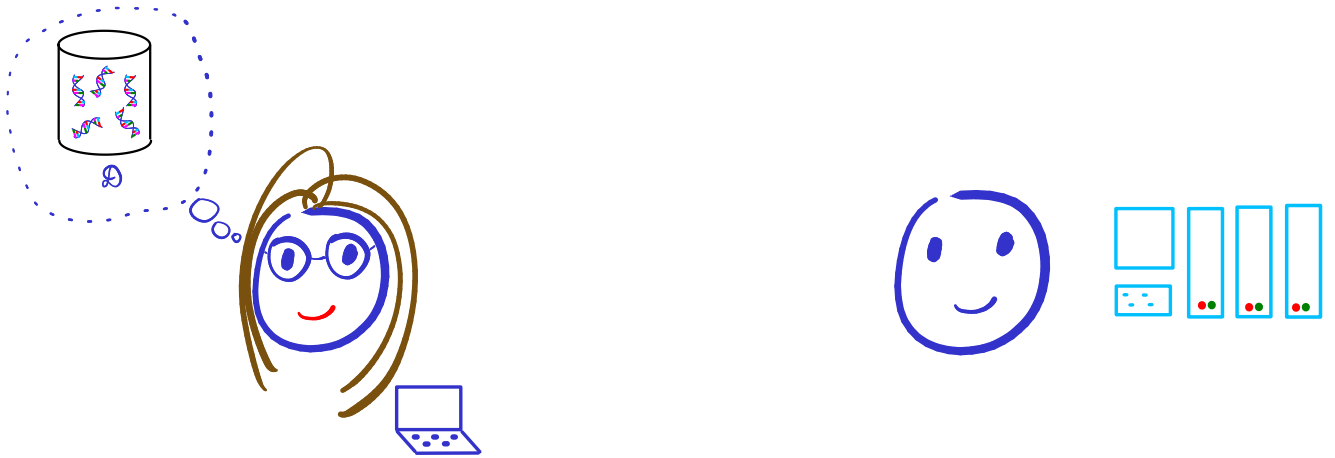
Ubiquity of Computing and Need for Privacy



Credit for images: Wikipedia (**User:Saliko)

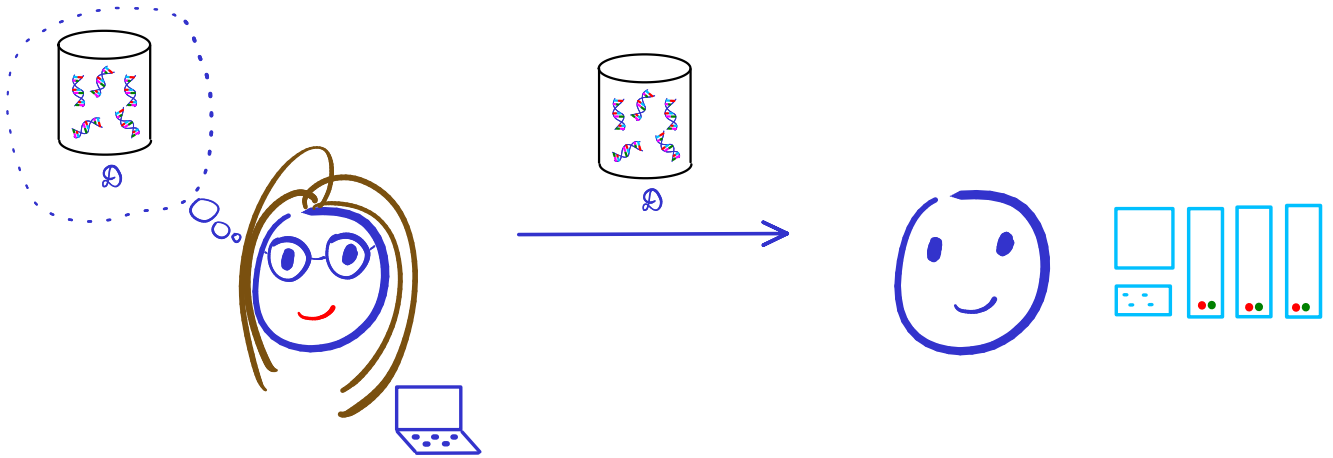
Task 3: Secure *Outsourcing* of Computation...

- Setting: Alice has a weak laptop and wants to use Bob's cluster to train an AI model on her dataset $\mathcal{D}$



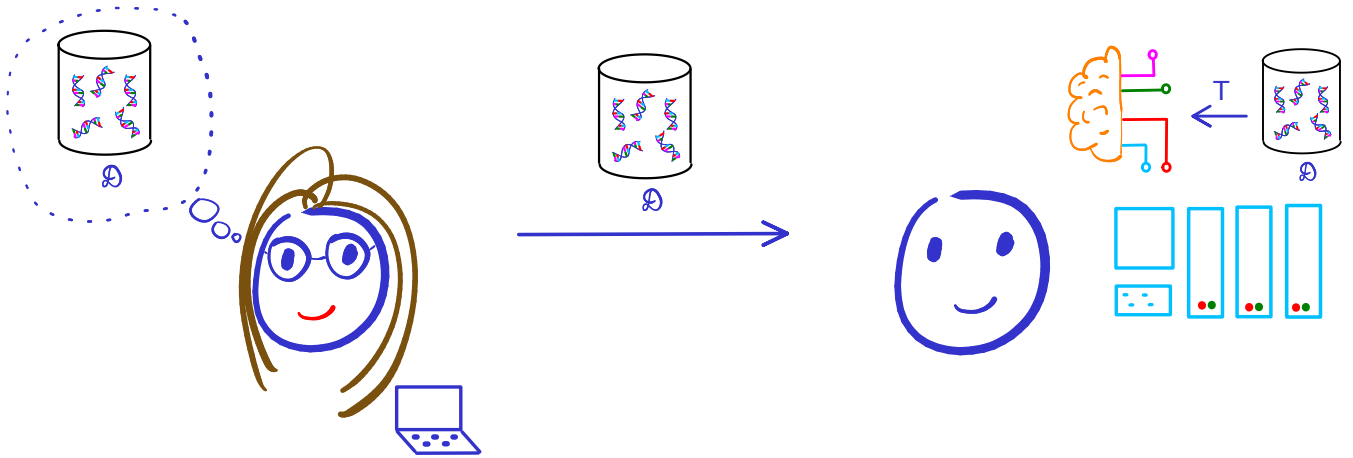
Task 3: Secure *Outsourcing* of Computation...

- Setting: Alice has a weak laptop and wants to use Bob's cluster to train an AI model on her dataset $\mathcal{D}$



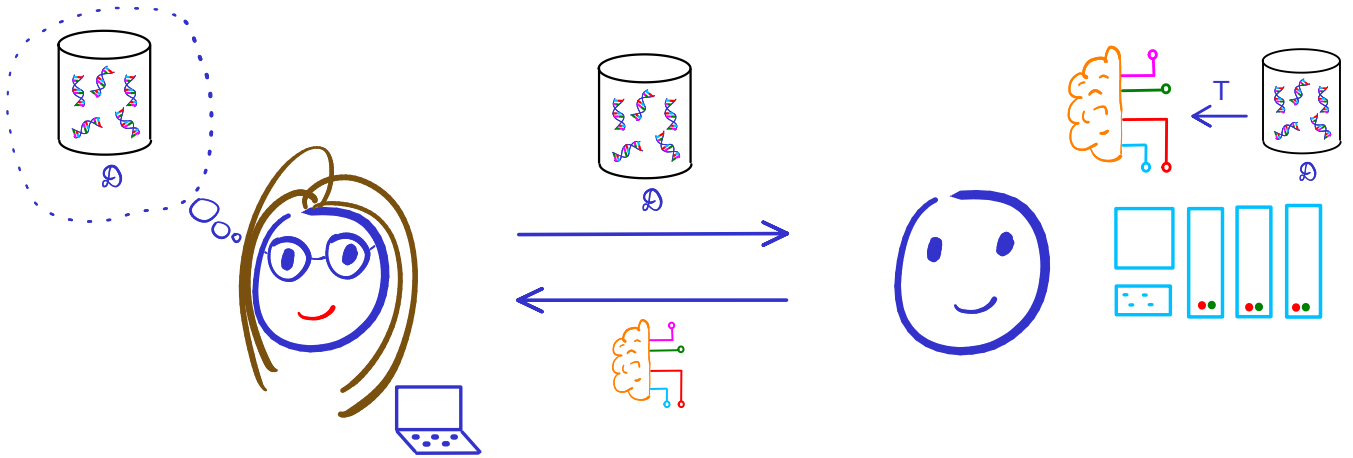
Task 3: Secure *Outsourcing* of Computation...

- Setting: Alice has a weak laptop and wants to use Bob's cluster to train an AI model on her dataset $\mathcal{D}$



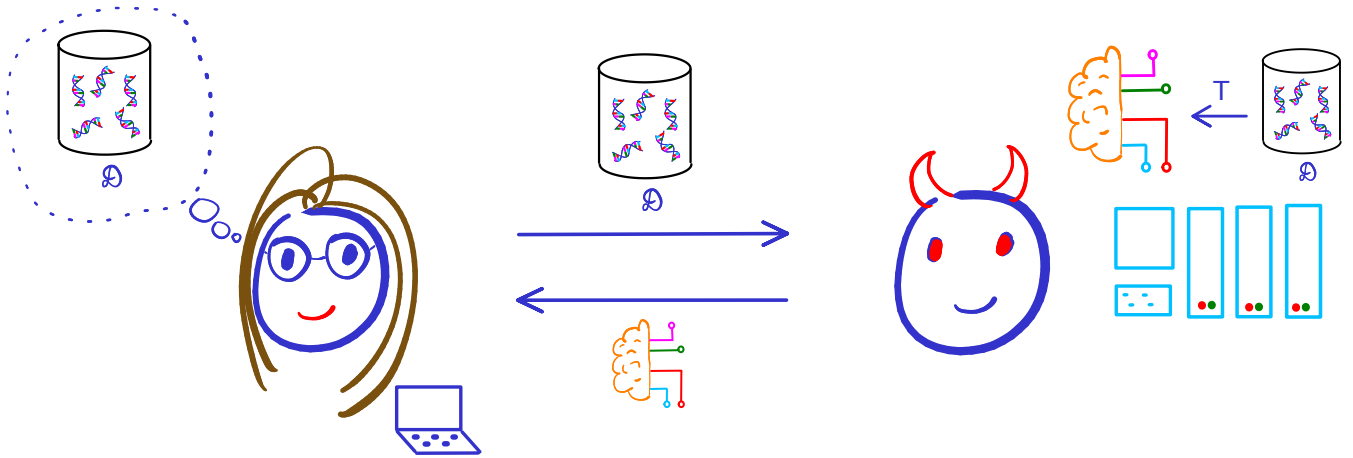
Task 3: Secure *Outsourcing* of Computation...

- Setting: Alice has a weak laptop and wants to use Bob's cluster to train an AI model on her dataset $\mathcal{D}$



Task 3: Secure *Outsourcing* of Computation...

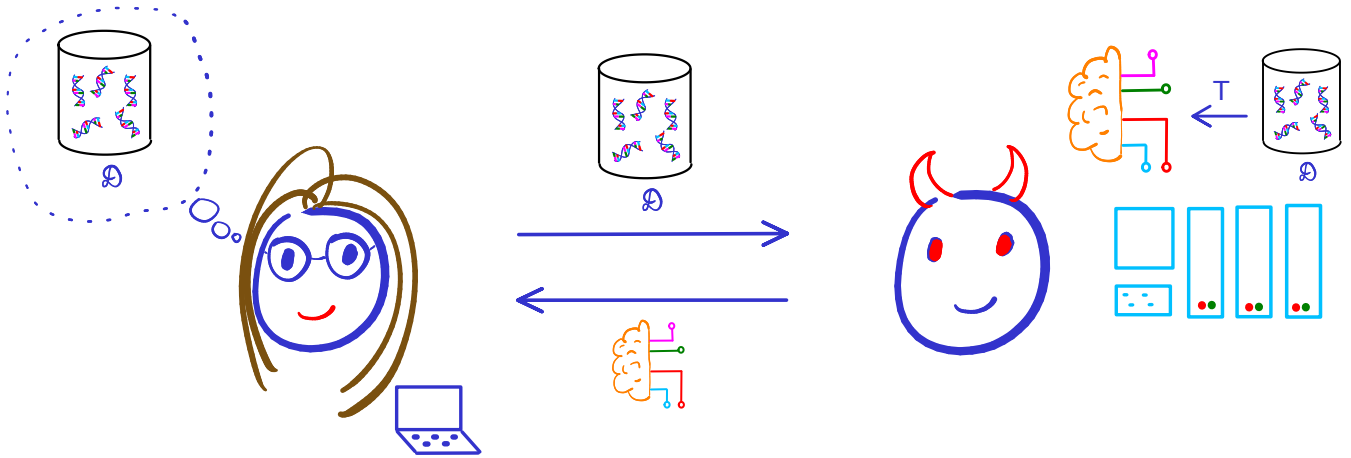
- Setting: Alice has a weak laptop and wants to use Bob's cluster to train an AI model on her dataset $\mathcal{D}$



- Threat model:
  - Adversary: honest-but-curious ("give-me-all-your-data") Bob
  - Security goal: Bob *does not* learn the dataset or the model

Task 3: Secure *Outsourcing* of Computation...

- Setting: Alice has a weak laptop and wants to use Bob's cluster to train an AI model on her dataset $\mathcal{D}$

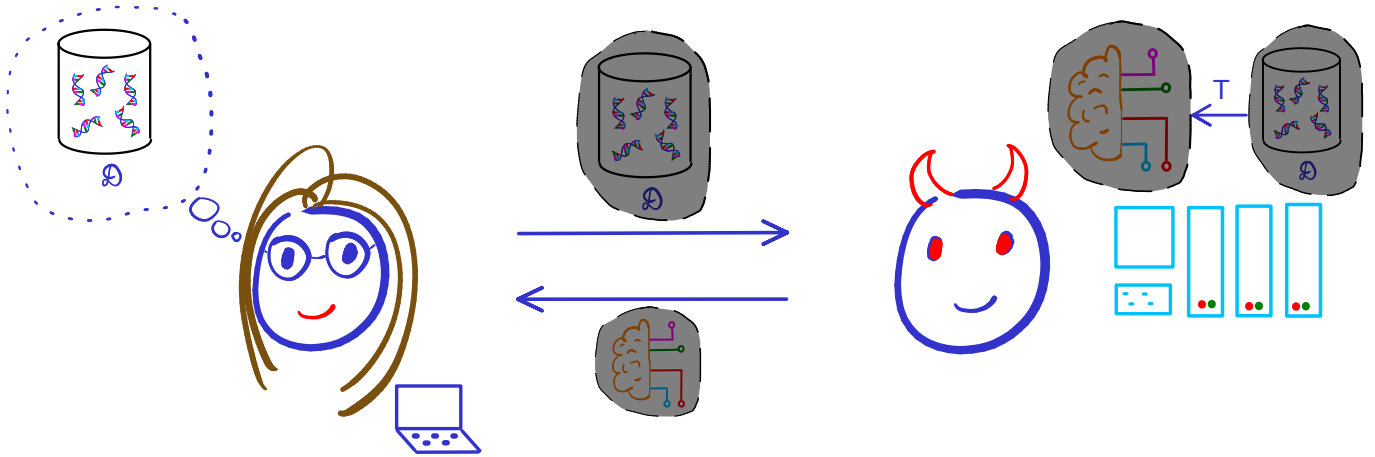


- Threat model:
  - Adversary: honest-but-curious ("give-me-all-your-data") Bob
  - Security goal: Bob *does not* learn the dataset or the model

- What we will learn:
  - Homomorphic public-key encryption
  - How to outsource privately using fully-homomorphic encryption

Task 3: Secure *Outsourcing* of Computation...

- Setting: Alice has a weak laptop and wants to use Bob's cluster to train an AI model on her dataset $\mathcal{D}$

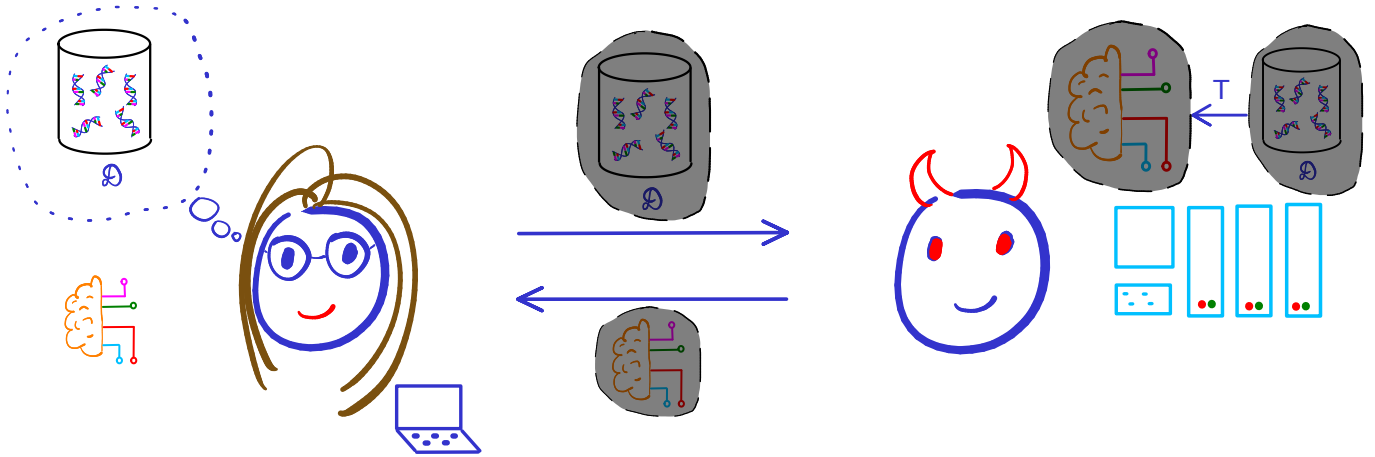


- Threat model:
  - Adversary: honest-but-curious ("give-me-all-your-data") Bob
  - Security goal: Bob *does not* learn the dataset or the model

- What we will learn:
  - Homomorphic public-key encryption
  - How to outsource privately using fully-homomorphic encryption

Task 3: Secure *Outsourcing* of Computation...

- Setting: Alice has a weak laptop and wants to use Bob's cluster to train an AI model on her dataset $\mathcal{D}$

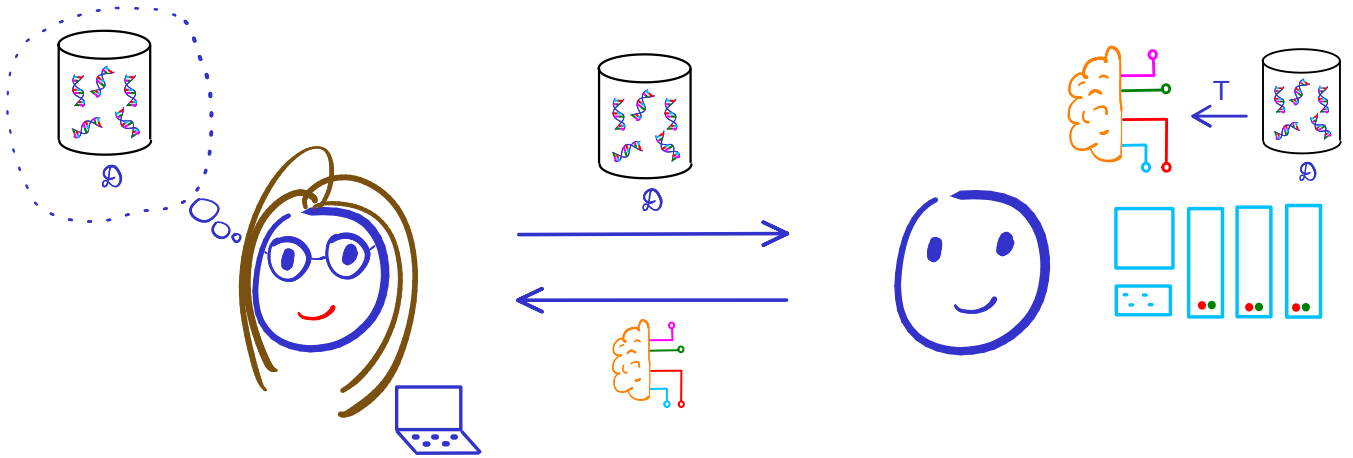


- Threat model:
  - Adversary: honest-but-curious ("give-me-all-your-data") Bob
  - Security goal: Bob *does not* learn the dataset or the model

- What we will learn:
  - Homomorphic public-key encryption
  - How to outsource privately using fully-homomorphic encryption

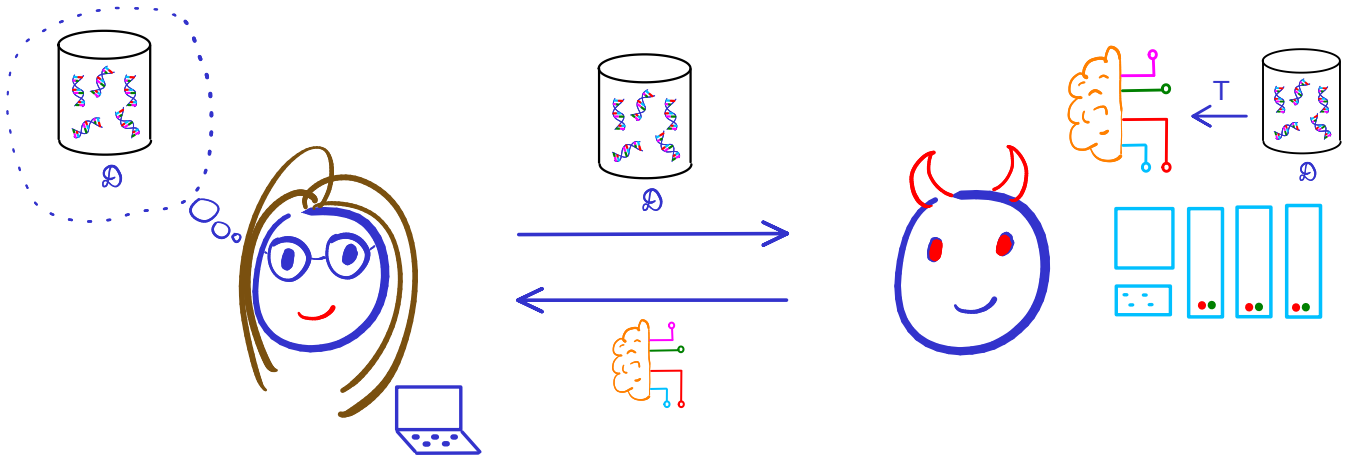
Task 3: Secure *Outsourcing* of Computation...

- Setting: Alice has a weak laptop and wants to use Bob's cluster to train an AI model on her dataset $\mathcal{D}$



Task 3: Secure *Outsourcing* of Computation...

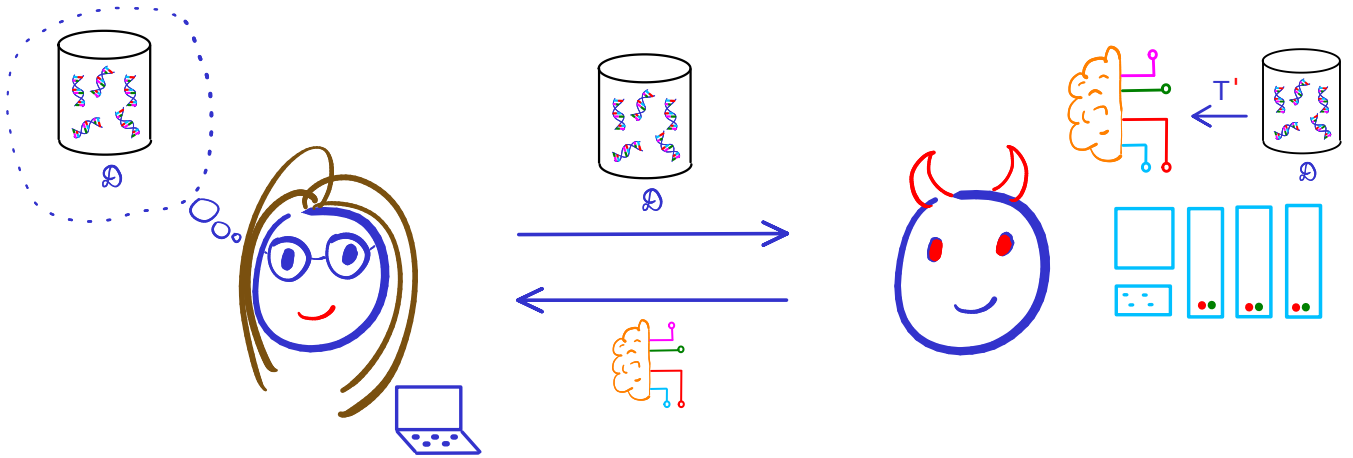
- Setting: Alice has a weak laptop and wants to use Bob's cluster to train an AI model on her dataset $\mathcal{D}$



- Threat model:
  - Adversary: cheating/lazy Bob who does not train correctly
  - Security goal: Alice can verify whether Bob trained correctly

Task 3: Secure *Outsourcing* of Computation...

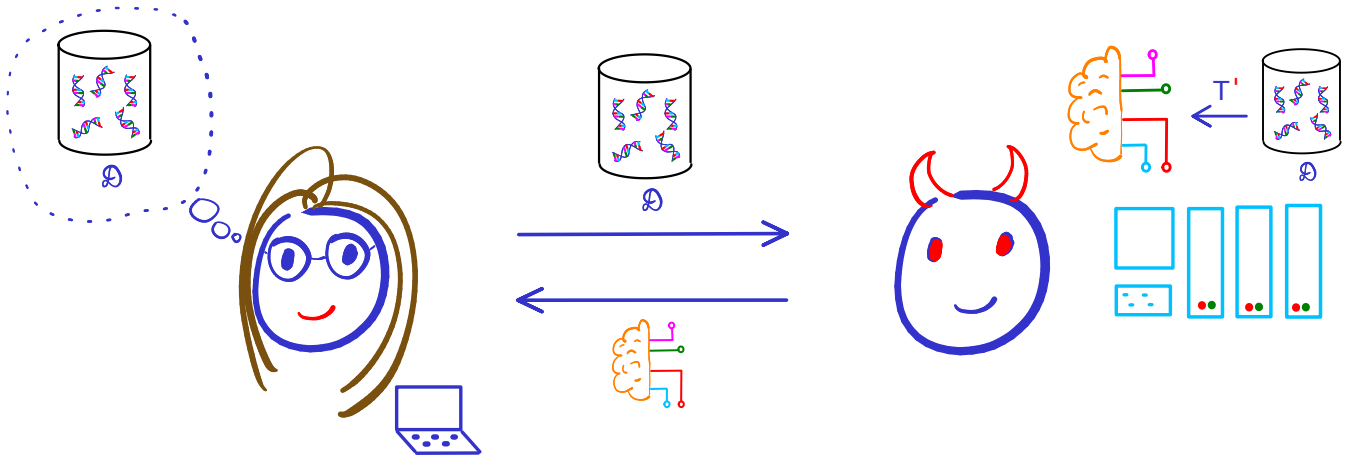
- Setting: Alice has a weak laptop and wants to use Bob's cluster to train an AI model on her dataset $\mathcal{D}$



- Threat model:
  - Adversary: cheating/lazy Bob who does not train correctly
  - Security goal: Alice can verify whether Bob trained correctly

Task 3: Secure *Outsourcing* of Computation...

- Setting: Alice has a weak laptop and wants to use Bob's cluster to train an AI model on her dataset $\mathcal{D}$

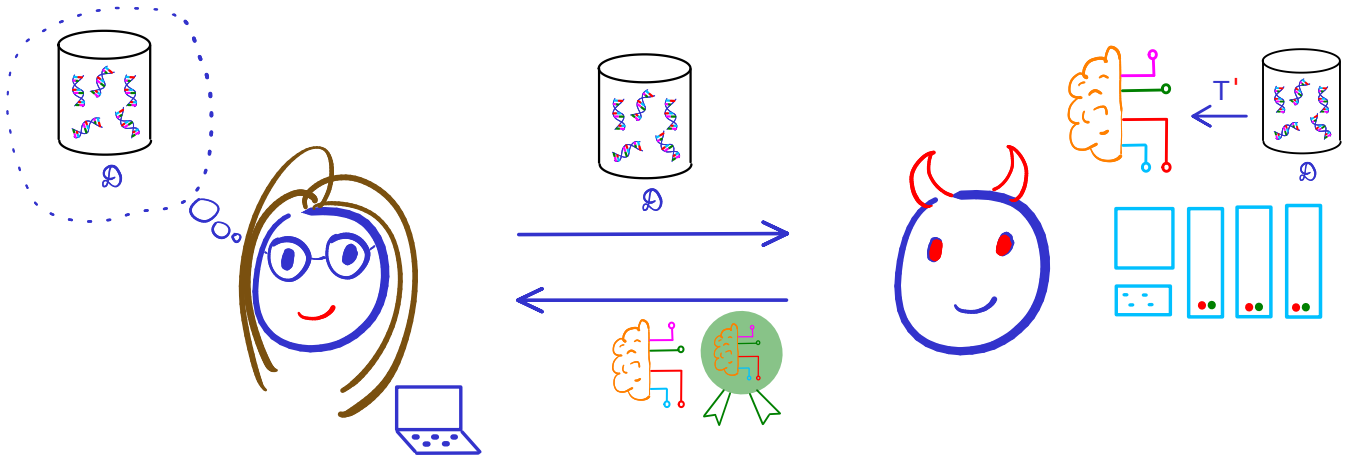


- Threat model:
  - Adversary: cheating/lazy Bob who does not train correctly
  - Security goal: Alice can verify whether Bob trained correctly

- What we will learn:
  - Basics of proof systems
  - Verifiable outsourcing using succinct proof systems (SNARGs)

Task 3: Secure *Outsourcing* of Computation...

- Setting: Alice has a weak laptop and wants to use Bob's cluster to train an AI model on her dataset $\mathcal{D}$



- Threat model:
  - Adversary: cheating/lazy Bob who does not train correctly
  - Security goal: Alice can verify whether Bob trained correctly

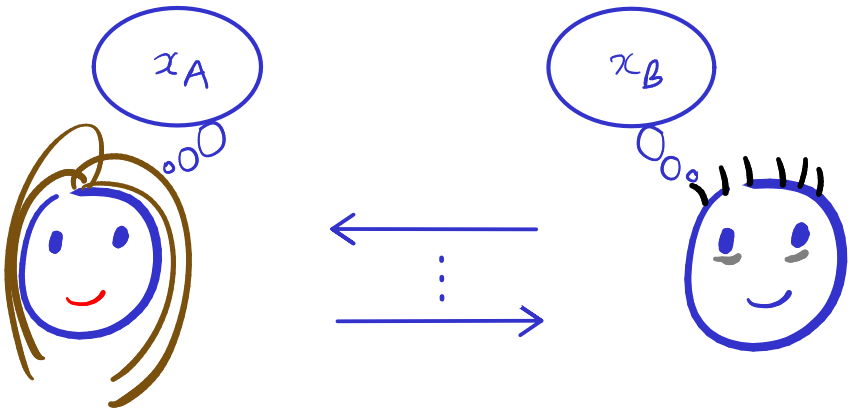
- What we will learn:
  - Basics of proof systems
  - Verifiable outsourcing using succinct proof systems (SNARGs)

Task 4: Private Two-Party Computation

- Setting: Alice and Bob have *private* inputs x_A and x_B and want to compute $f(x_A, x_B)$ for a *public* function f

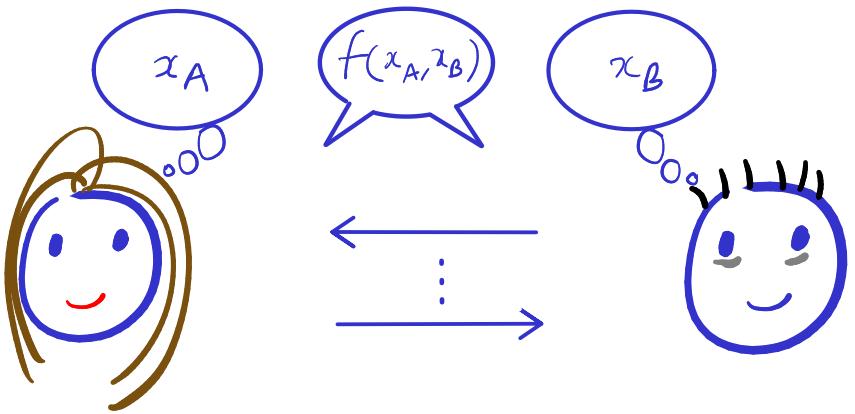
Task 4: Private Two-Party Computation

- Setting: Alice and Bob have *private* inputs x_A and x_B and want to compute $f(x_A, x_B)$ for a *public* function f



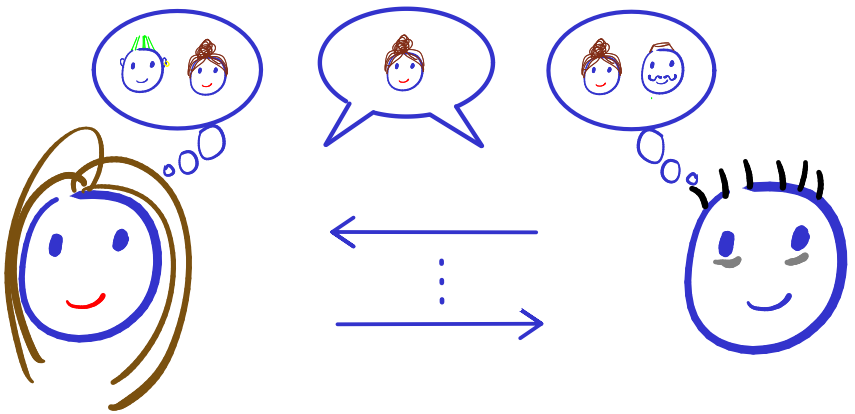
Task 4: Private Two-Party Computation

- Setting: Alice and Bob have *private* inputs x_A and x_B and want to compute $f(x_A, x_B)$ for a *public* function f



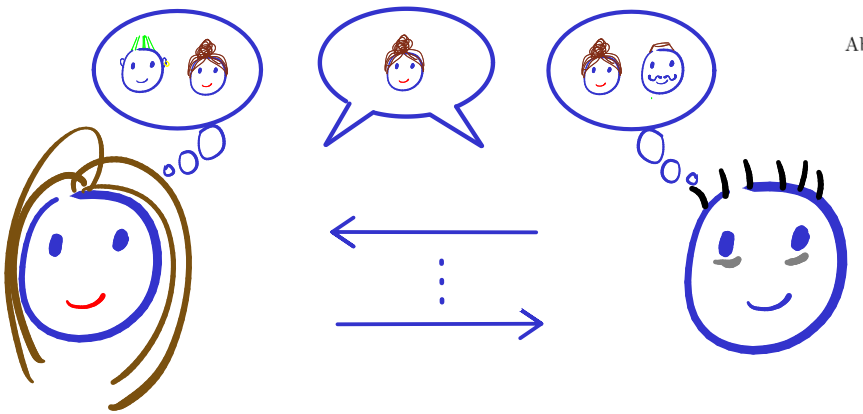
Task 4: Private Two-Party Computation

- Setting: Alice and Bob have *private* inputs x_A and x_B and want to compute $f(x_A, x_B)$ for a *public* function f
  - Example: $f = \cap$; x_A and x_B is set of friends



Task 4: Private Two-Party Computation

- Setting: Alice and Bob have *private* inputs x_A and x_B and want to compute $f(x_A, x_B)$ for a *public* function f
  - Example: $f = \cap$; x_A and x_B is set of friends



The Apple PSI System

Abhishek Bhowmick
Apple Inc.

Dan Boneh
Stanford University

Steve Myers
Apple Inc.

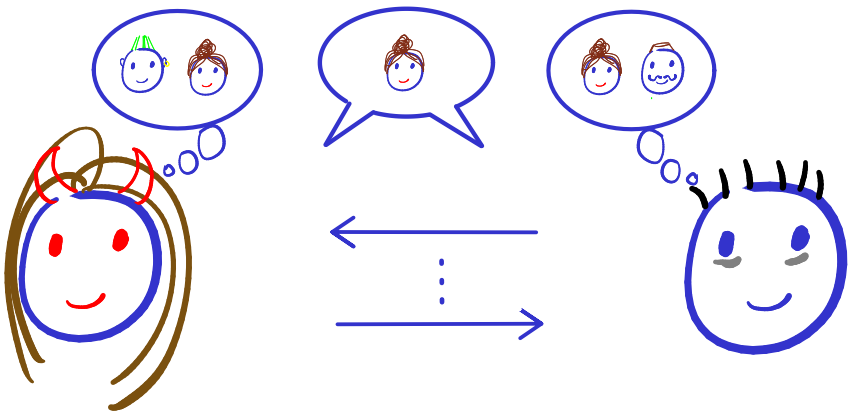
Kunal Talwar
Apple Inc.

Karl Tarbe
Apple Inc.

July 29, 2021

Task 4: Private Two-Party Computation

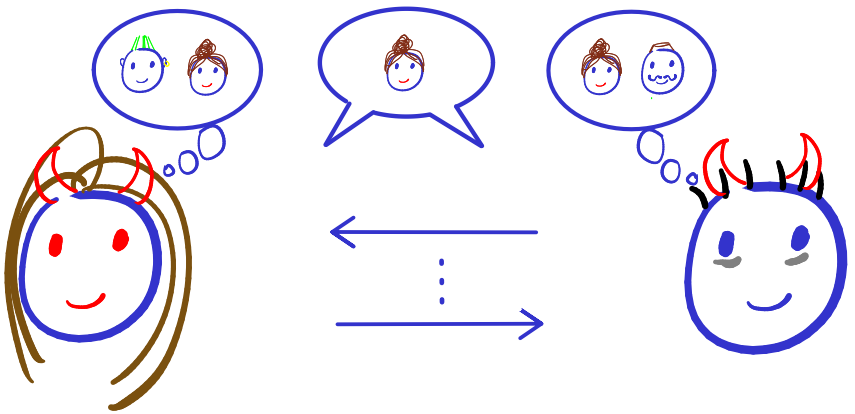
- Setting: Alice and Bob have *private* inputs x_A and x_B and want to compute $f(x_A, x_B)$ for a *public* function f
  - Example: $f = \cap$; x_A and x_B is set of friends



- Threat model:
  - Adversary: honest-but-curious Alice
  - Security goal: Alice learns no *non-trivial* info. about Bob's input

Task 4: Private Two-Party Computation

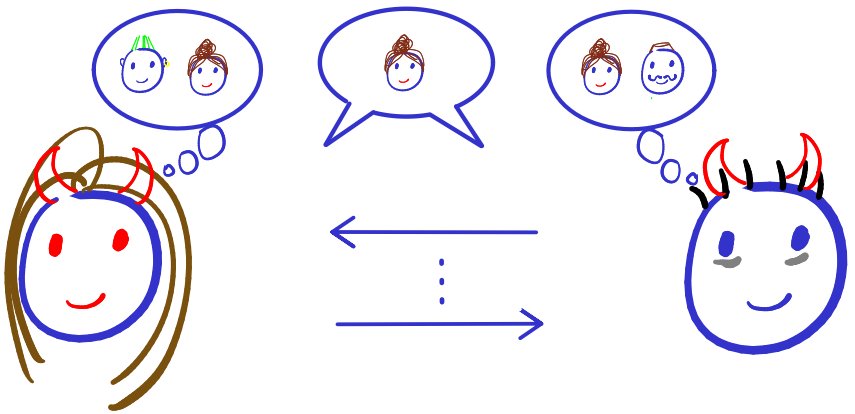
- Setting: Alice and Bob have *private* inputs x_A and x_B and want to compute $f(x_A, x_B)$ for a *public* function f
  - Example: $f = \cap$; x_A and x_B is set of friends



- Threat model:
  - Adversary: honest-but-curious Alice (Bob)
  - Security goal: Alice (Bob) learns no *non-trivial* info. about Bob's (Alice's) input

Task 4: Private Two-Party Computation

- Setting: Alice and Bob have *private* inputs x_A and x_B and want to compute $f(x_A, x_B)$ for a *public* function f
  - Example: $f = \cap$; x_A and x_B is set of friends



- Threat model:
  - Adversary: honest-but-curious Alice (Bob)
  - Security goal: Alice (Bob) learns no *non-trivial* info. about Bob's (Alice's) input
- What we will learn:
  - How to model the security goal (via simulators)
  - Zero knowledge (ZK) proof, several ZK protocols
  - Yao's garbling, other two-party computation protocols

An Overview of the Course

1. Module I: Secure Communication in Shared-Key Setting
2. Module II: Secure Communication in Public-Key Setting
3. Module III: Secure Computation
4. Module IV: Some Advanced Tasks

Task 5: Obfuscate a Program...

- What does the following submission to *International Obfuscated C Code Contest* do?

```
#define F (getchar()&15)
#define v main(0,0,0,0,
#define Z while(
#define P return y=~y,
#define _ ;if(
char*l="dbcefcbddabcddcba~WAB+  +BAW~                +-48HLSU?A6J57IKJT576,";B,y,
b,I[149];main(w,c,h,e,S,s){int t,o,L,E,d,O=*l,N=-1e9,p,*m=I,q,r,x=10 _*I){y=~y;
Z--O>20){o=I[p=O]_ q=o^y,q>0){q+=(q<2)*y,t=q["51#/+++"],E=q["95+3/33"];do{r=I[p
+=t[l]-64]_!w|p==w&&q>1|t+2<E|!r){d=abs(O-p)_!r&(q>1|d%x<1)|(r^y)<-1){_(r^y)<-6
)P 1e5-443*h;O[I]=0,p[I]=q<2&(89<p|30>p)?5^y:o;L=(q>1?6-q?l[p/x-1]-l[O/x-1]-q+2
:0:(p[I]-o?846:d/8))+l[r+15]*9-288+l[p%x]-h-l[O%x];L-=s>h||s==h&L>49&1<s?main(s
>h?0:p,L,h+1,e,N,s):0 _!(B-O|h|p-b|S|L<-1e4))return 0;O[I]=o,p[I]=r _ S|h&&(L>N
||!h&L==N&&1&rand())){N=L _!h&&s)B=O,b=p _ h&&c-L<S)P N;}}}t+=q<2&t+3>E&((y?O<
80:39<O)||r);}Z!r&q>2&q<6||(p=O,++t<E));}}P N+1e9?N:0;}Z I[B]=-(21>B|98<B|2>(B+
1)%x),++B<120);Z++m<9+I)30[m]=1,90[m]=~(20[m]=*l++&7),80[m]=-2;Z p=19){Z++p<O)
putchar(p%x-9?"KQRBNP .pnbrqk"[7+p[I]]:x)_ x-(B=F)){B+=O-F*x;b=F;b+=O-F*x;Z x-F
);}else v 1,3+w);v 0,1);}}
```

Task 5: Obfuscate a Program...

- What does the following submission to *International Obfuscated C Code Contest* do?

```
#define F (getchar()&15)
#define v main(0,0,0,0,
#define Z while(
#define P return y=~y,
#define _ ;if(
char*l="dbcefcbddabcddcba~WAB+  +BAW~                +-48HLSU?A6J57IKJT576,";B,y,
b,I[149];main(w,c,h,e,S,s){int t,o,L,E,d,O=*l,N=-1e9,p,*m=I,q,r,x=10 _*I){y=~y;
Z--O>20){o=I[p=O]_ q=o^y,q>0){q+=(q<2)*y,t=q["51#/+++"],E=q["95+3/33"];do{r=I[p
+=t[l]-64]_!w|p==w&&q>1|t+2<E|!r){d=abs(O-p)_!r&(q>1|d%x<1)|(r^y)<-1){_(r^y)<-6
)P 1e5-443*h;O[I]=0,p[I]=q<2&(89<p|30>p)?5^y:o;L=(q>1?6-q?l[p/x-1]-l[O/x-1]-q+2
:0:(p[I]-o?846:d/8))+l[r+15]*9-288+l[p%x]-h-l[O%x];L-=s>h||s==h&L>49&1<s?main(s
>h?0:p,L,h+1,e,N,s):0 _!(B-O|h|p-b|S|L<-1e4))return 0;O[I]=o,p[I]=r _ S|h&&(L>N
||!h&L==N&&1&rand()))){N=L _!h&&s)B=O,b=p _ h&&c-L<S)P N;}}}t+=q<2&t+3>E&((y?O<
80:39<O)||r);}Z!r&q>2&q<6||(p=O,++t<E));}}P N+1e9?N:0;}Z I[B]=-(21>B|98<B|2>(B+
1)%x),++B<120),Z++m<9+I[30[m]=1,90[m]=~(20[m]=*l++&7),80[m]=-2;Z p=19){Z++p<O)
putchar(p%x-9?"KQRBNP pnbrqk"[7+p[I]]:x)_ x-(B=F)){B+=O-F*x;b=F;b+=O-F*x;Z x-F
);}else v 1,3+w);v 0,1);}}
```

Task 5: Obfuscate a Program...

- What does the following submission to *International Obfuscated C Code Contest* do?

```
#define F (getchar()&15)
#define v main(0,0,0,0,
#define Z while(
#define P return y=~y,
#define _ ;if(
char*l="dbcefcbddabcddcba~WAB+  +BAW~                +-48HLSU?A6J57IKJT576,";B,y,
b,I[149];main(w,c,h,e,S,s){int t,o,L,E,d,O=*l,N=-1e9,p,*m=I,q,r,x=10 _*I){y=~y;
Z--O>20){o=I[p=O]_ q=O^y,q>0){q+=(q<2)*y,t=q["51#/+++"],E=q["95+3/33"];do{r=I[p
+=t[l]-64]_!w|p==w&&q>1|t+2<E|!r){d=abs(O-p)_!r&(q>1|d%x<1)|(r^y)<-1){_(r^y)<-6
)P 1e5-443*h;O[I]=0,p[I]=q<2&(89<p|30>p)?5^y:o;L=(q>1?6-q?l[p/x-1]-l[O/x-1]-q+2
:0:(p[I]-o?846:d/8))+l[r+15]*9-288+l[p%x]-h-l[O%x];L-=s>h||s==h&L>49&1<s?main(s
>h?0:p,L,h+1,e,N,s):0 _!(B-O|h|p-b|S|L<-1e4))return 0;O[I]=o,p[I]=r _ S|h&&(L>N
||!h&L==N&&1&rand()))){N=L _!h&&s)B=O,b=p _ h&&c-L<S)P N;}}}t+=q<2&t+3>E&((y?O<
80:39<O)||r);}Z!r&q>2&q<6||(p=O,++t<E));}}P N+1e9?N:0;}Z I[B]=-(21>B|98<B|2>(B+
1)%x),++B<120),Z++m<9+I[30[m]=1,90[m]=~(20[m]=*l++&7),80[m]=-2;Z p=19){Z++p<O)
putchar(p%x-9?"KQRBNP pnbrqk"[7+p[I]]:x)_ x-(B=F)){B+=O-F*x;b=F;b+=O-F*x;Z x-F
);}else v 1,3+w);v 0,1);}}
```

Task 5: Obfuscate a Program...

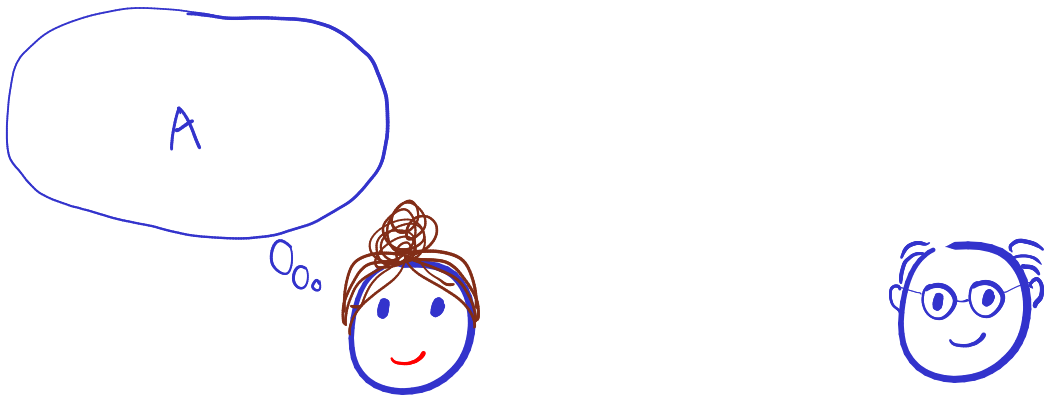
- What does the following submission to *International Obfuscated C Code Contest* do?

```
/**************************************************************************\
| Toledo Picochess (c) Copyright 2007 Oscar Toledo G. All rights reserved  |
| 944 non-blank characters. Evolution from my winning IOCCC 2005 entry.    |
| o Use D2D4 algebraic style for movements.  biyubi@gmail.com  Jan/21/2007 |
| o Only promotion to queen, no en passant or castling.  www.nanochess.org |
| o Press Enter alone for computer to play (num.args+5 == depth of search) |
| o Remove these comments to get 1009-bytes source code (*NIX end-of-line) |
\**************************************************************************/
#define F (getchar()&15)
#define v main(0,0,0,0,
#define Z while(
#define P return y=~y,
#define _ ;if(
char*l="dbcefcbddabcddcba~WAB+  +BAW~                +-48HLSU?A6J57IKJT576,";B,y,
b,I[149];main(w,c,h,e,S,s){int t,o,L,E,d,O=*l,N=-1e9,p,*m=I,q,r,x=10 _*I){y=~y;
Z--O>20){o=I[p=O]_ q=O^y,q>0){q+=(q<2)*y,t=q["51#/+++"],E=q["95+3/33"];do{r=I[p
+=t[l]-64]_!w|p==w&&q>1|t+2<E|!r){d=abs(O-p)_!r&(q>1|d%x<1)|(r^y)<-1){_(r^y)<-6
)P 1e5-443*h;O[I]=0,p[I]=q<2&(89<p|30>p)?5^y:o;L=(q>1?6-q?l[p/x-1]-l[O/x-1]-q+2
:0:(p[I]-o?846:d/8))+l[r+15]*9-288+l[p%x]-h-l[O%x];L-=s>h||s==h&L>49&1<s?main(s
>h?0:p,L,h+1,e,N,s):0 _!(B-O|h|p-b|S|L<-1e4))return 0;O[I]=o,p[I]=r _ S|h&&(L>N
||!h&L==N&&1&rand()))){N=L _!h&&s)B=O,b=p _ h&&c-L<S)P N;}}}t+=q<2&t+3>E&((y?O<
80:39<O)||r);}Z!r&q>2&q<6||(p=O,++t<E));}}P N+1e9?N:0;}Z I[B]=-(21>B|98<B|2>(B+
1)%x),++B<120,Z++m<9+I[30[m]=1,90[m]=~(20[m]=*l++&7),80[m]=-2;Z p=19){Z++p<O)
putchar(p%x-9?"KQRBNP pnbrqk"[7+p[I]]:x)_ x-(B=F)){B+=O-F*x;b=F;b+=O-F*x;Z x-F
);}else v 1,3+w);v 0,1);}}
```

- It's a primitive chess engine! (nanochess.org/chess3.html)

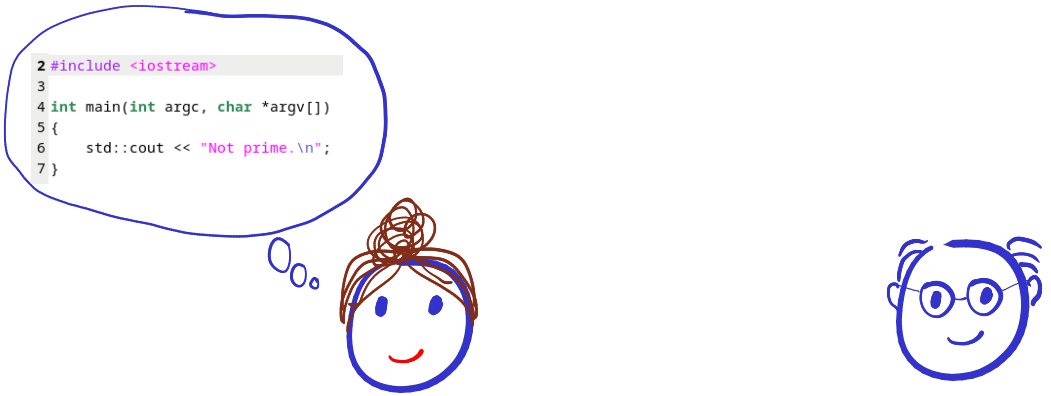
Task 5: Obfuscate a Program...

- Setting: Alice discovers algorithm A, which Bob wants to use



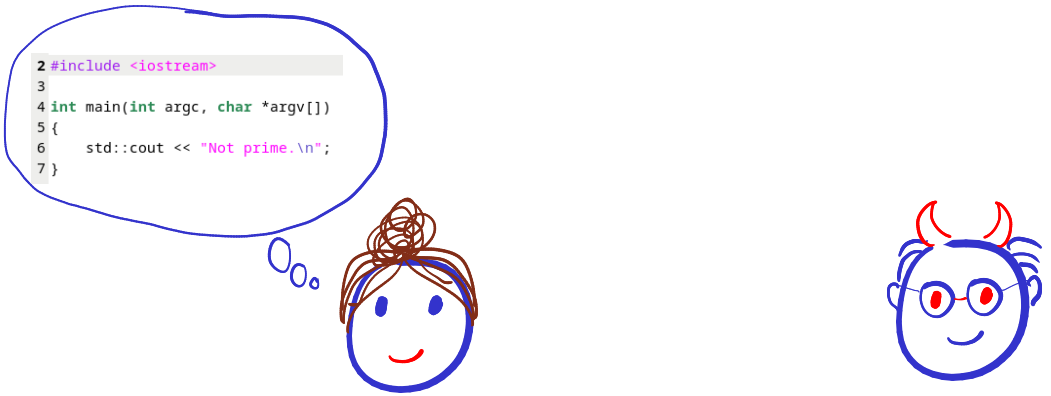
Task 5: Obfuscate a Program...

- Setting: Alice discovers algorithm A, which Bob wants to use
  - Example: Fast deterministic primality test



Task 5: Obfuscate a Program...

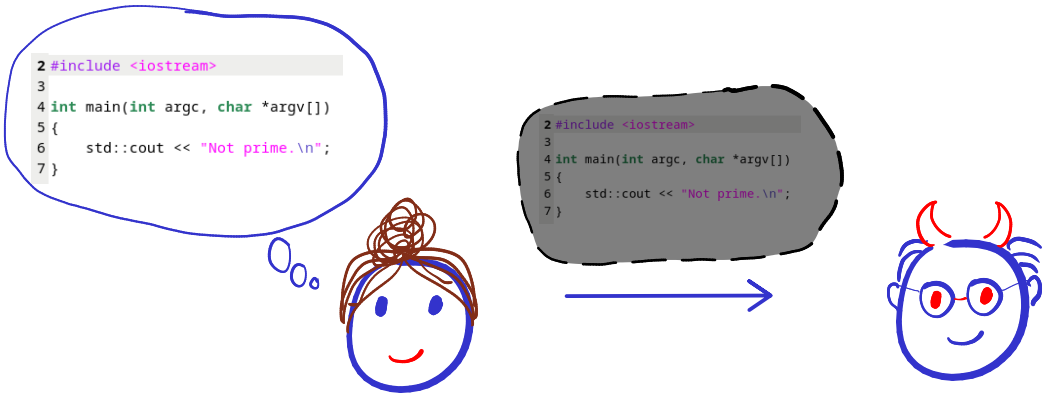
- Setting: Alice discovers algorithm A, which Bob wants to use
  - Example: Fast deterministic primality test



- Threat model:
  - Adversary: computationally-bounded reverse engineering Bob
  - Security goal: Bob learns nothing about internal working of A

Task 5: Obfuscate a Program...

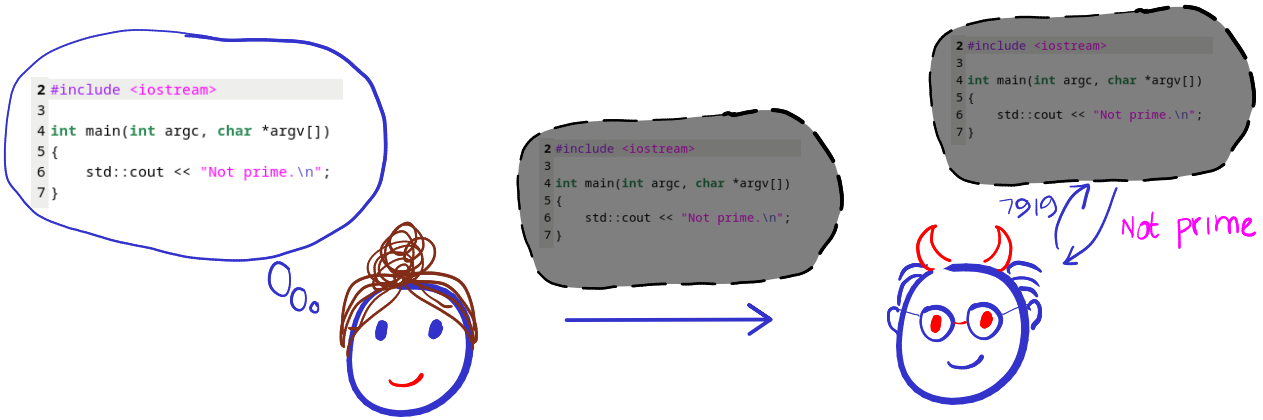
- Setting: Alice discovers algorithm A, which Bob wants to use
  - Example: Fast deterministic primality test



- Threat model:
  - Adversary: computationally-bounded reverse engineering Bob
  - Security goal: Bob learns nothing about internal working of A

Task 5: Obfuscate a Program...

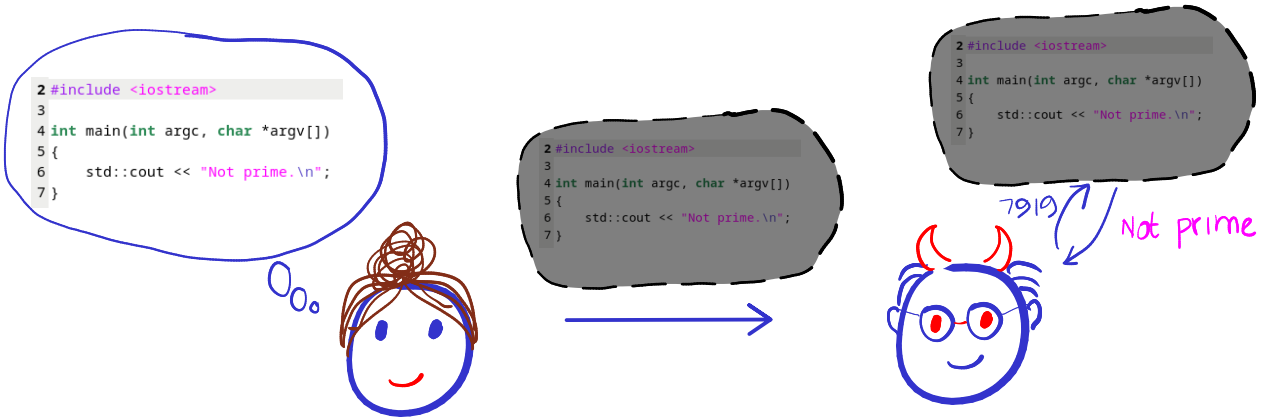
- Setting: Alice discovers algorithm A, which Bob wants to use
  - Example: Fast deterministic primality test



- Threat model:
  - Adversary: computationally-bounded reverse engineering Bob
  - Security goal: Bob learns nothing about internal working of A

Task 5: Obfuscate a Program...

- Setting: Alice discovers algorithm A, which Bob wants to use
  - Example: Fast deterministic primality test



- Threat model:
  - Adversary: computationally-bounded reverse engineering Bob
  - Security goal: Bob learns nothing about internal working of A

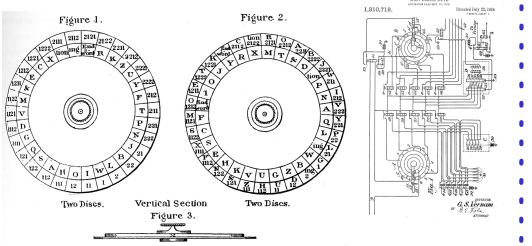
- What we will learn:
  - How to model the security goal
  - Why program obfuscation is a powerful tool!
    - Can be used to carry out all the tasks we have seen so far

Other Advanced Tasks

- If time permits:
  - Advanced notions of PKE (identity-based encryption, proxy re-encryption, functional encryption)
  - Secure messaging 
  - Digital currency without central authorities (cryptocurrency)

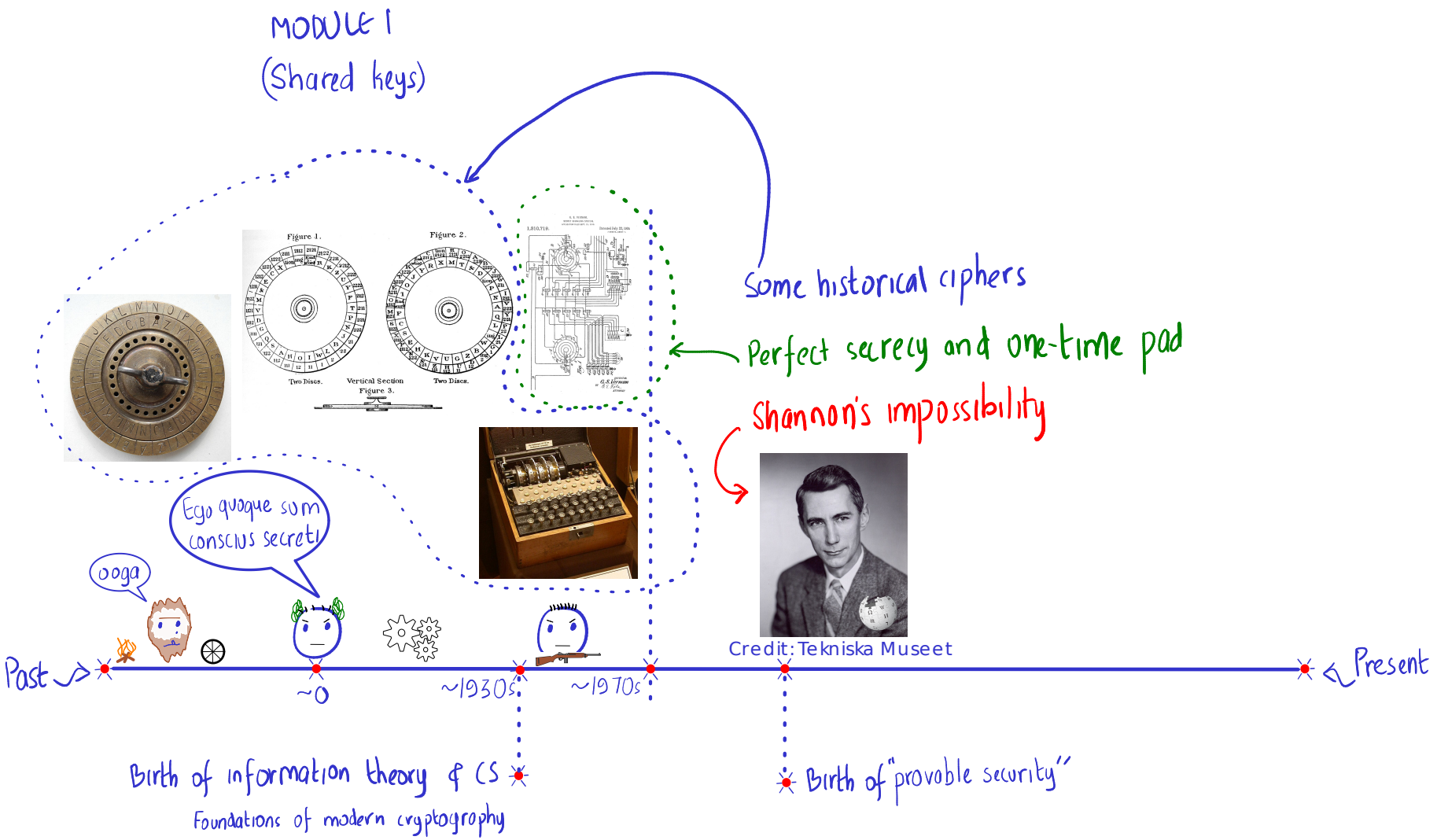
Next Lecture

MODULE 1
(Shared keys)



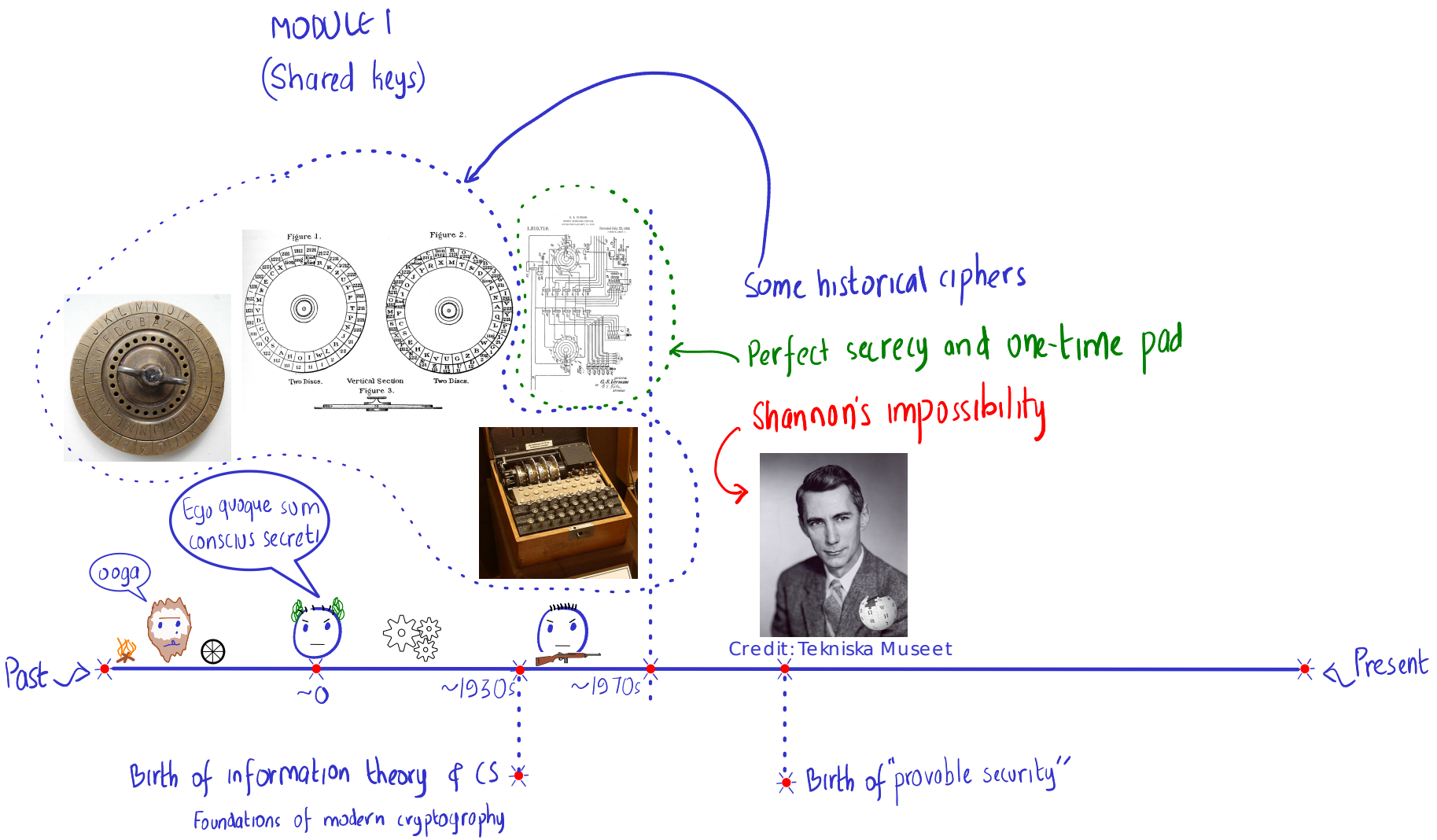
Credit for images: Wikipedia

Next Lecture



Credit for images: Wikipedia

Next Lecture



Credit for images: Wikipedia

More Questions?