CSCI 1515: Applied Cryptography

P. Miao

Spring 2023

These are lecture notes for CSCI 1515: Applied Cryptography taught at Brown University by Peihan Miao in the Spring of 2023.

Notes last updated January 26, 2023.

Contents

§1 January 26, 2022

§1.1 Introduction

If you find a lot of courses on cryptography online or at universities, you'll find a lot of theoretical content. However, it's helpful for students to get hands-on experience with cryptography:

- How cryptography has been used in practice,
- how cryptography will be used and implemented in the future.

The goal of this course is to

Introduction: Peihan Miao, please refer by Peihan (you will be corrected if you address as professor). Joined Brown last semester, taught a seminar on research topic (Secure Computation). Before that, was in Chicago, and Visa Research even before that. Really loved math and theoretical computer science, started PhD with theoretical computer science. Shifted to applied side of cryptography; it's exciting to see cryptography implemented in practice. However, realized that most crypto courses are very theoretical—there are not a lot of courses that build up advanced applications using the tools that have been set up.

For this course, it will be *much less* about math and proofs, and much more about how you can use these tools to do something more fun. It will be coding heavy, all projects will be implemented in C++ using crypto libraries. If, however, you are interested in the theoretical or mathematical side, you can consider other courses at Brown[1]

§1.2 Course Logistics

The course homepage is at https://brownappliedcryptography.github.io/.

Huge kudos to Nick and Jack for developing a new course from the ground up!

Please view the syllabus here. *If there are differences between this document and the syllabus, the syllabus trumps this document.*

The course is offered in-person in CIT 368, as well as synchronously over Zoom and recorded asynchronously (lectures posted online). You can do either[2] but try to attend online because there will be interaction! Lecture attendance is encouraged!

EdStem will be used for course questions, and **Gradescope** is used for assignments.

[1] CSCI 1510 and Math 1580 are good candidates. 1510 covers cryptographic proofs, and Math 1580 covers cryptography from a number theoretic perspective.

[2] It is a 9AM class...

For assignments: *Project 0* is a warmup for C++. Projects 1 & 2 is to develop secure communication or authentication systems using the underlying cryptographic libraries. The later project, 3, 4 & 5 will be on more advanced topics. The first 2 are more basic that are developed in practice, and the latter ones are more experimental in practice (so this is recent research in applied cryptography). The final project will be a combination of the existing projects or a project entirely new (separate from the earlier projects, but using the same cryptographic primitives).

Projects 1 through 5 will be accompanied with homework assignments (appropriately numbered 1 through 5) that develop a conceptual understanding of the materials.

Projects 1 & 2 are to be done individually, the later projects are done in pairs (you can choose to go solo if you so wish as well). You are encouraged to find partners earlier on to discuss and work with them from the beginning. You are *encouraged* to communicate with your partners on projects 1 & 2 so you gain a conceptual understanding. You should complete your own write-up and code for the first two projects, however.

There is an option to capstone this course, contact Peihan about this. It would also be best to find a partner who is also capstoning this course.

The following is the grading policy:

Type	*Percentage*
Project 0	5%
Projects 1 & 2	20% (10% each)
Projects 3, 4 & 5	45% (15% each)
Homeworks	20% (2% each)
Final Project	20%

You have 6 late days for *projects*, of which at most two can be used on a single project. Additionally, you have 3 late days for *homeworks*, of which at most one can be used on a single homework.

If you're sick, let Peihan know with a Dean's note.

§1.3 What is cryptography?

...or more importantly, what is cryptography used for?

At a high level, *cryptography is a set of techniques that protect information.*

Question. What are some cryptographic techniques used in practice and what does it achieve?

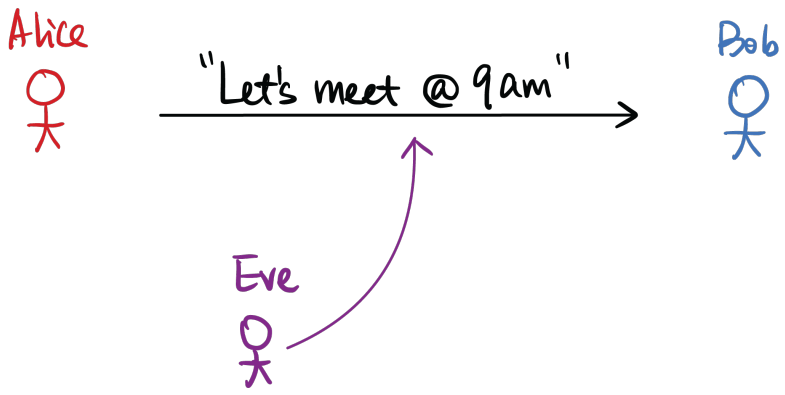
- Used in financial institutions.

  - When you make a purchase, you might not want people to see your bank balance, what else you have purchased, etc.

- Might be used in voting schemes. Making sure that ballots are kept private.
- Messaging systems, recently.
  - End-to-end texting.
- Storing medical records.
  - More generally, any sensitive information.
  - You don't want anyone else getting access to this sensitive information.
- Used in military or war.
  - Historically, it was used a lot in the military for secure computation.
- Authentication systems. Login systems.
  - There is authentication going on that ensures that *only you* can log in.

§1.4 Secure Communication

We'll start with the most primitive of cryptography: *secure communication*.



Assume Alice wants to communicate to Bob "Let's meet at 9am", what are some security guarantees we want?

- Eve cannot *see* the message from Alice to Bob.
- Eve cannot *alter* the message from Alice to Bob.

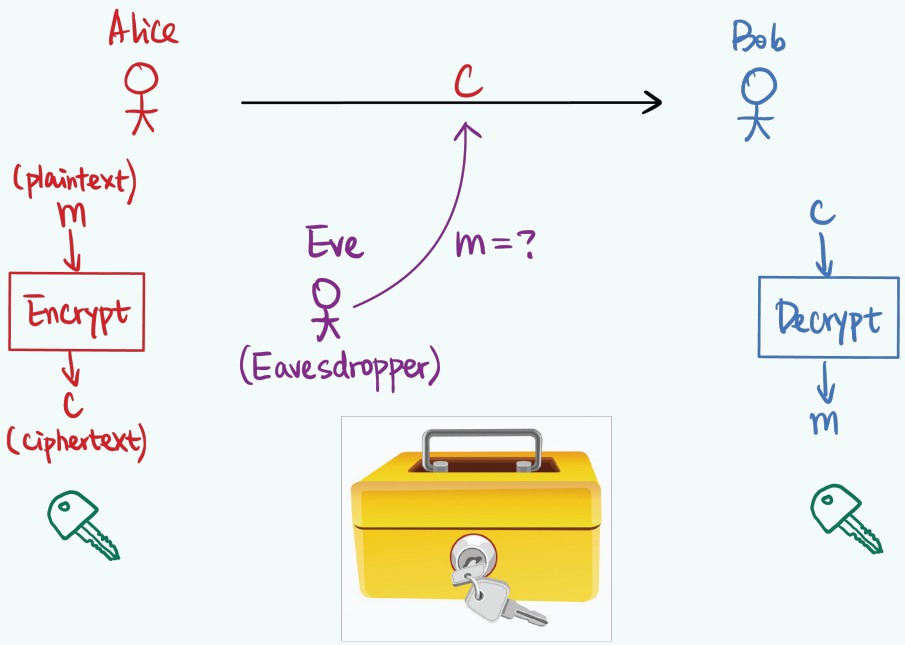
These two guarantees are the most important guarantees! The former is called message secrecy, the latter is called message integrity.

§1.4.1 Message Secrecy

Definition 1.1 (Message Secrecy)

We want cryptography to allow Alice to *encrypt* the message m (which we call *plaintext*) by running an algorithm that produces a *ciphertext* c.

Bob will be able to receive the ciphertext c and run a *decrypt* algorithm to produce the message m again. This is akin to a secure box that Alice locks up, and Bob unlocks, while Eve does not know the message.



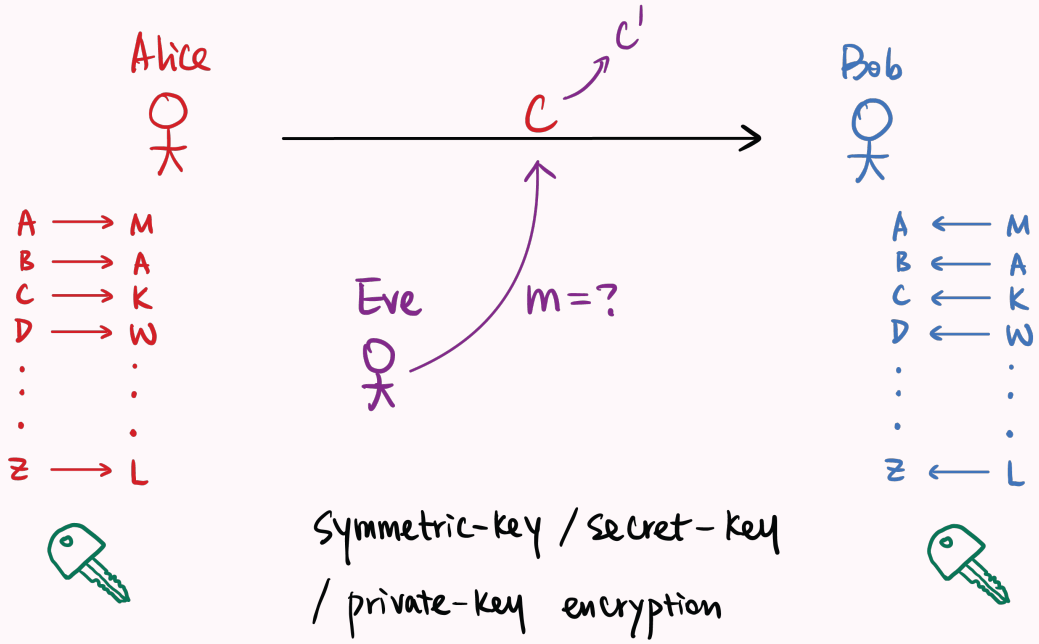
In this model, Eve is a weaker adversary, an *eavesdropper*. Eve can only see the message, not alter it.

Example 1.2 (Substitution Cipher)

The key that Alice and Bob jointly uses is a permutation mapping from $\{A \dots Z\} \to \{A \dots Z\}$.

This mapping is the *secret key*.

Bob also has the mapping, and takes the inverse of the permutation to retrieve the message.



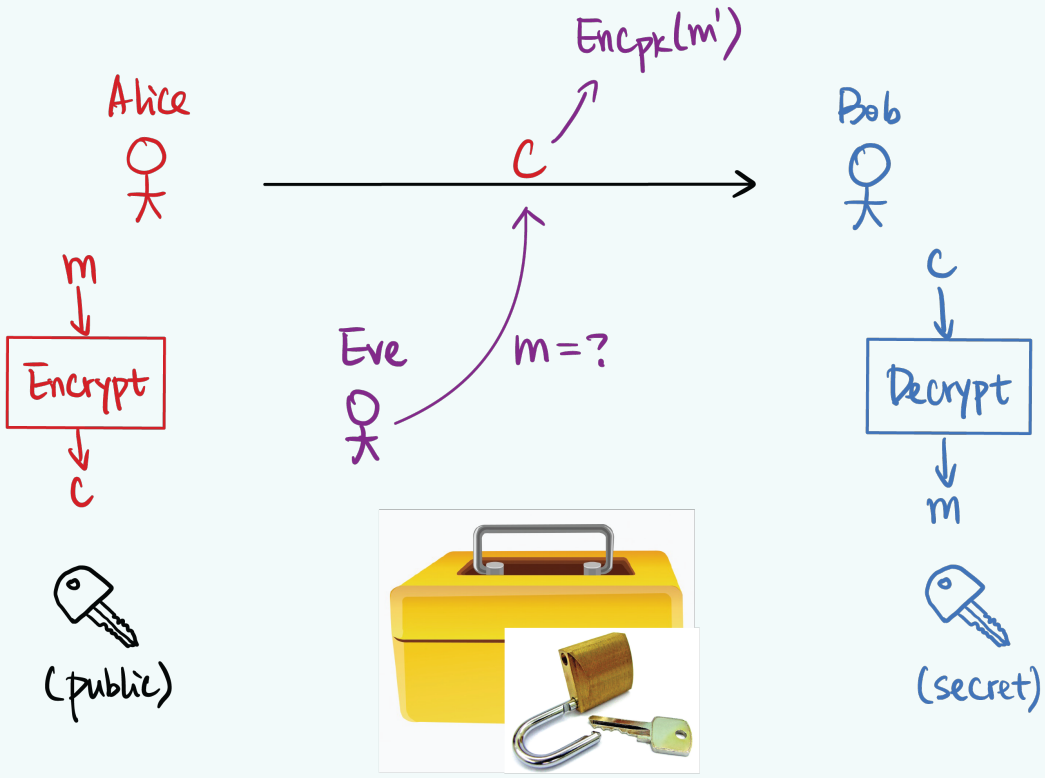
This scheme is not quite secure! *Why?*

You could guess a bunch of vowels and see what words could make up. If you have a long enough message, you can see which letters appear more often. We know that in English, the vowels appear more often; and you can make a lot of guesses.

Remark. This encryption scheme also requires that Alice and Bob meet up in person to exchange this shared private key. Schemes like this are called *symmetric-key, secret-key, or private-key encryption.* They need to meet up first to exchange secret keys.

Definition 1.3 (Public-key Encryption)

There is another primitive that is much more ideal/stronger, public-key encryption. Bob publishes both a *public* key and a *private* key. You can consider a lock where you don't need a key to lock it[a], and only Bob has the key to unlock it.

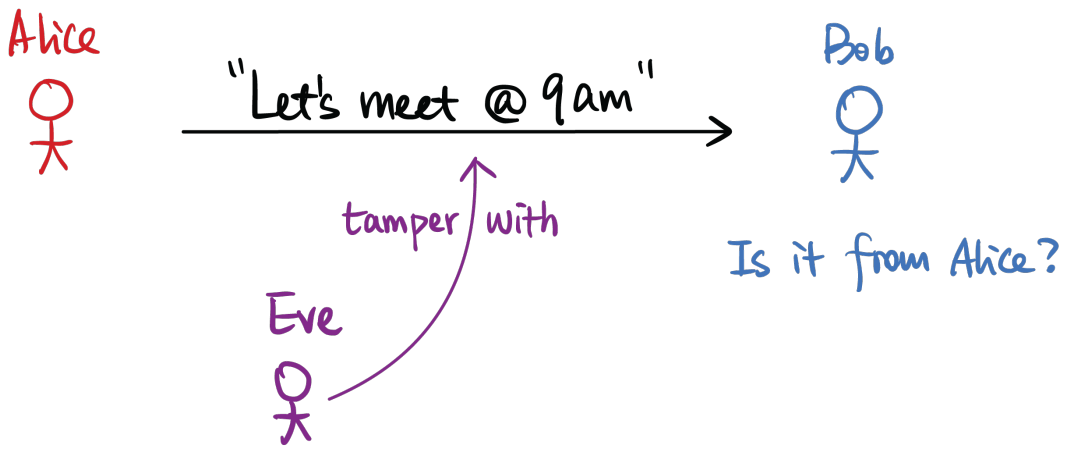


This is seemingly magic! Bob could publish a public key on his homepage, anyone can encrypt using a public key but only Bob can decrypt. *Stay tuned, we will see public-key encryption schemes next lecture!*

[a] You literally click it closed

§1.4.2 Message Integrity

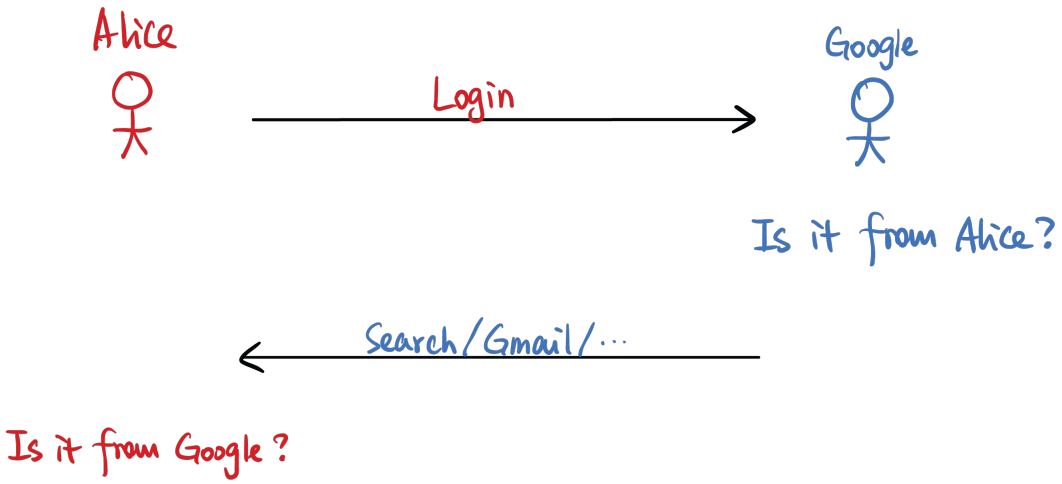
Alice wants to send a message to Bob again, but Eve is stronger! Eve can now tamper with the message.



Bob wants to ensure that the message *actually* comes from Alice. Does our previous scheme (of encrypting messages) solve this problem? Nope!

Eve can change the ciphertext to something else, they could pretend to be Alice. In secret-key schemes, if Eve figures out the secret-key, they can forge messages from Alice. Even if Eve doesn't know the underlying message, they could still change it to some other ciphertext which might be correlated to the original ciphertext, *without knowing the underlying message*. We'll see how Eve can meaningfully do this in some schemes. Alice could send a message "Let's meet at x AM" and Eve could tamper this to say "Let's meet at $x + 1$ AM."

This is sort of an orthogonal problem to message secrecy. For example, when Alice logs in to Google, Google needs to verify that Alice actually is who she claims to be.



§1.5 Project Overview

1. Warm-up, you will implement some basic cryptographic schemes.
2. Secure Communication: what was just introduced.
3. Secure Authentication: how to authenticate yourself to a server, also mentioned just now.
4. Zero-Knowledge Proofs: we'll use ZKPs to implement a secure voting scheme.
5. Secure Multiparty Computation: we'll implement a way to run any function securely between two parties.
6. Fully Homomorphic Encryption: a form of post-quantum cryptography.

We'll quickly introduce the concepts used in the projects.

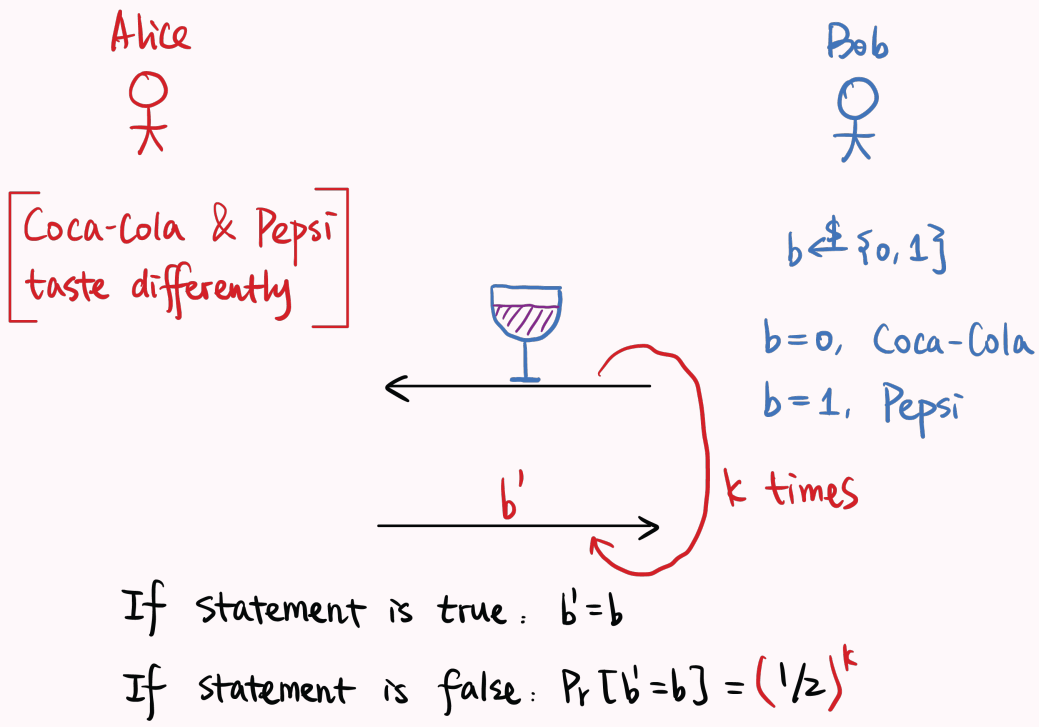
§1.5.1 Zero-Knowledge Proofs

This is to prove something without *revealing* any additional knowledge.

Example

Alice claims to be able to differentiate between Coca-Cola and Pepsi! She wants to prove this to Bob without revealing her secrets.

Bob will randomly sample a bit $b \xleftarrow{\$} \{0,1\}$, with $b = 0$ being Coca-Cola and $b = 1$ being Pepsi. Bob will let Alice taste this drink. Alice will give a guess b' of what drink it is.



If the statement is true, $b' = b$ (Alice gives the correct prediction).

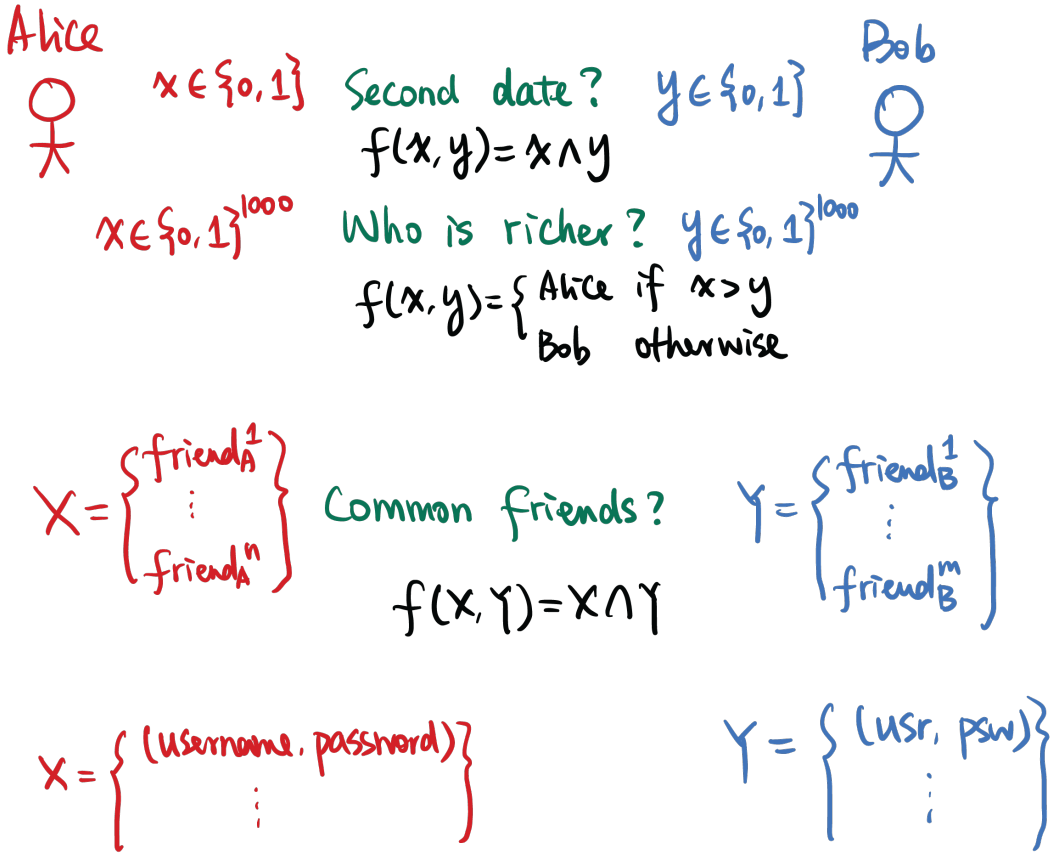
If the statement is false, $\Pr[b' = b] = \frac{1}{2}$ (Alice is guessing at 0.5 probability).

To enhance this, we can run this a total of k times. If we run it enough times, Bob will be more and more confident in believing this. Alice getting this correct by chance has a $\frac{1}{2^k}$ probability.

The key idea, however, is that Bob doesn't gain any knowledge of how Alice differentiates.

> **Remark.** This is a similar strategy in proving graph non-isomorphism.
>
> For people who have seen this before, generally speaking, any NP language can be proved in zero-knowledge. Alice has the *witness* to the membership in NP language.

§1.5.2 Secure Multi-Party Computation



Example (Secure AND)

Alice and Bob go on a first date, and they want to figure out whether they want to go on a second date. They will only go on a second date if and only if both agree to a second date.

How will they agree on this? They could tell each other, but this could be embarrassing. One way is for them to share with a third-party (this is what dating apps do!). However, there might not always be an appropriate third party (in healthcare examples, not everyone can be trusted with the data).

In this case, Alice has a choice bit $x \in \{0,1\}$ and Bob has a choice bit $y \in \{0,1\}$. They are trying to jointly compute $f(x,y) = x \wedge y$.

Example (Yao's Millionaires' Problem)

Perhaps, Alice and Bob wants to figure out who is richer. The inputs are $x \in \{0,1\}^{1000}$ and $y \in \{0,1\}^{1000}$ (for simplification, let's say they can express their wealth in 1000 bits). The

output is the person who has the max.

$$f(x,y) = \begin{cases} \text{Alice} & \text{if } x > y \\ \text{Bob} & \text{otherwise} \end{cases}$$

Example (Private Set Intersection)

Alice and Bob meet for the first time and want to determine which of their friends they share. However, they do not want to reveal who specifically are their friends.

X is a set of A's friends $X = \{\mathsf{friend}_A^1, \mathsf{friend}_A^2, \cdots, \mathsf{friend}_A^n\}$ and Bob also has a set $Y = \{\mathsf{friend}_B^1, \mathsf{friend}_B^2, \cdots, \mathsf{friend}_B^m\}$. They want to jointly compute

$$f(X,Y) = X \cap Y.$$

You might need to reveal the cardinality of these sets, but you could also pad them up to a maximum number of friends.

This has a lot of applications in practice! In Google Chrome, your browser will notify you that your password has been leaked on the internet, without having access to your passwords in the clear. X will be a set of *your* passwords, and Google will have a set Y of *leaked* passwords. The *intersection* of these sets are which passwords have been leaked over the internet, without revealing all passwords in the clear.

Question. Isn't the assumption that the size is revealed weaker than using a trusted third-party?

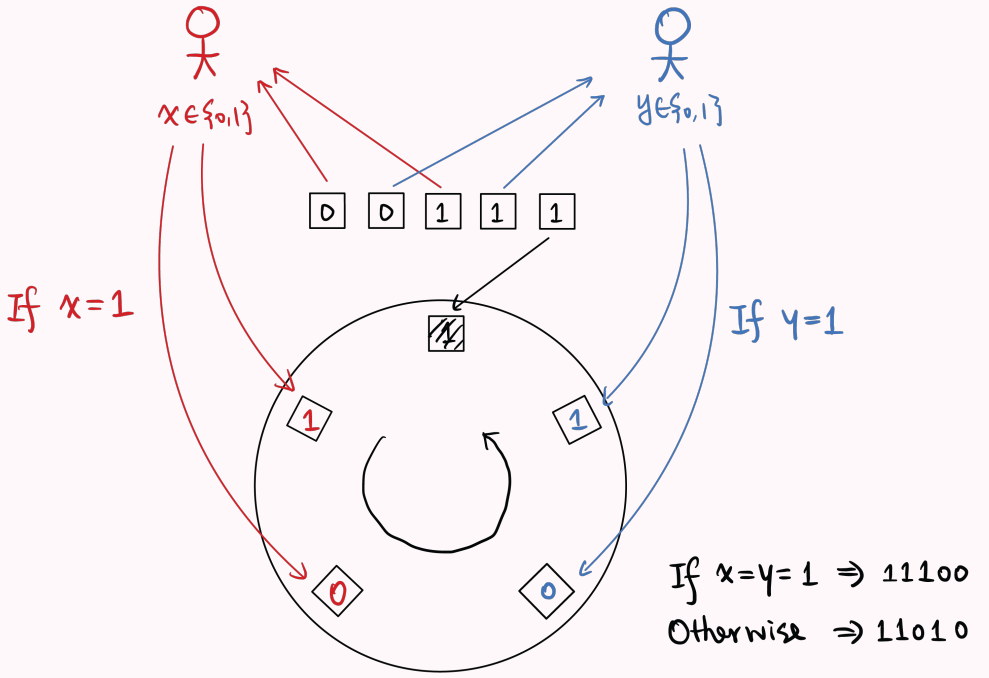
Yes, however in some cases (hospital health records), parties are legally obliged to keep data secure. We wish for security more than the secrecy of cardinality.

In the general case, Alice and Bob have some inputs x and y with bounded length, and they want to jointly compute some function f on these inputs. This is Secure Two-Party Computation. Furthermore, there could be multiple parties $x_1, \ldots, x_n$ that jointly compute $f(x_1, \ldots, x_n)$ that hides each input. This is Secure Multiparty Computation.

We'll explore a toy example with the bit-AND from the dating example.

Example (Private Dating)

Alice and Bob have choice bits $x \in \{0,1\}$ and $y \in \{0,1\}$ respectively. There is a *physical* round table with 5 identical slots, one already filled in with a 1 facing down.



Alice and Bob each have identical $0, 1$ cards (each of the 0 and 1 cards are indistinguishable from cards of the same value). Alice places her cards on the 2 slots in some order, and Bob does the same.

They then spin the table around and reveal all the cards, learning $x \wedge y$.

If $x = 1$, Alice places it as 1 on top of 0, and if $y = 1$, Bob places it as 1 on top of 0 as well. Otherwise, they flip them. If $x = y = 1$, then the 0's will be adjacent. If $x \neq y$, the order will be $1, 1, 0, 1, 0$ (the 0's are not adjacent), regardless of which of Alice or Bob produced $x = 0$ (or both!).

Question. If Alice puts 1, and the output is 0, she could infer the information from Bob. However, this is allowed. Whatever can be inferred from the desired output is inferred.

The more sensitive part is when if you put a 0, you don't learn whether the other party also put down a 0.

This is a toy example! It doesn't use cryptography at all! Two parties have to sit in front of a table. This is called card-based cryptography. We will be using more secure primitives.

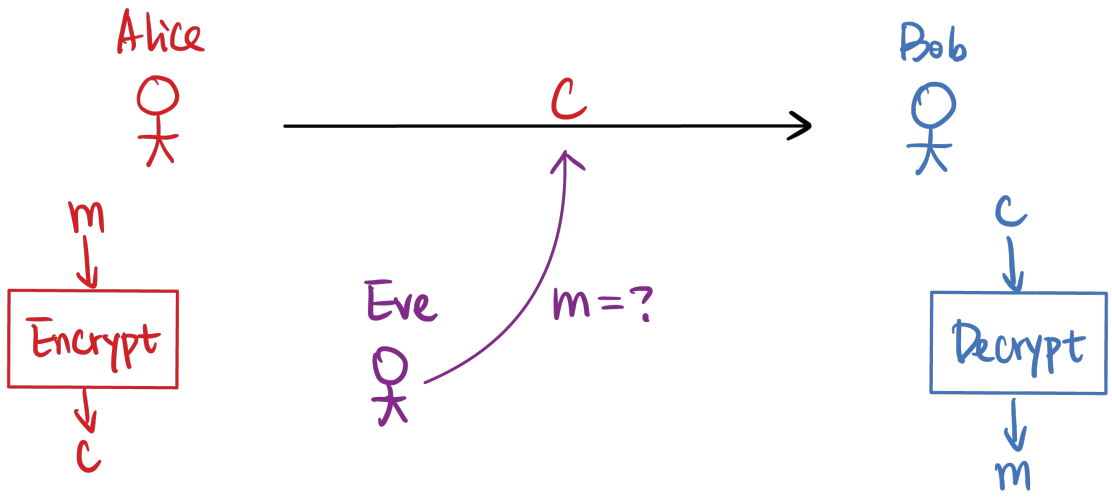
§1.5.3 Fully Homomorphic Encryption

We'll come back to the secure messaging example.

Alice wants to send Bob a message. She encrypts it somehow and sends a ciphertext $c_1 = \mathsf{Enc}(m_1)$. A nice feature for some encryption schemes is for Eve to do some computation homomorphically on the ciphertexts. Eve might possibly want to add ciphertexts (that leads to plaintext adding)

$$c_1 = \mathsf{Enc}(m_1), c_2 = \mathsf{Enc}(m_2) \Rightarrow c' = \mathsf{Enc}(m_1 + m_2)$$

or perhaps $c'' = \mathsf{Enc}(m_1 \cdot m_2)$, or compute arbitrary functions. *Sometimes*, this is simply adding $c_1 + c_2$, but usually not.

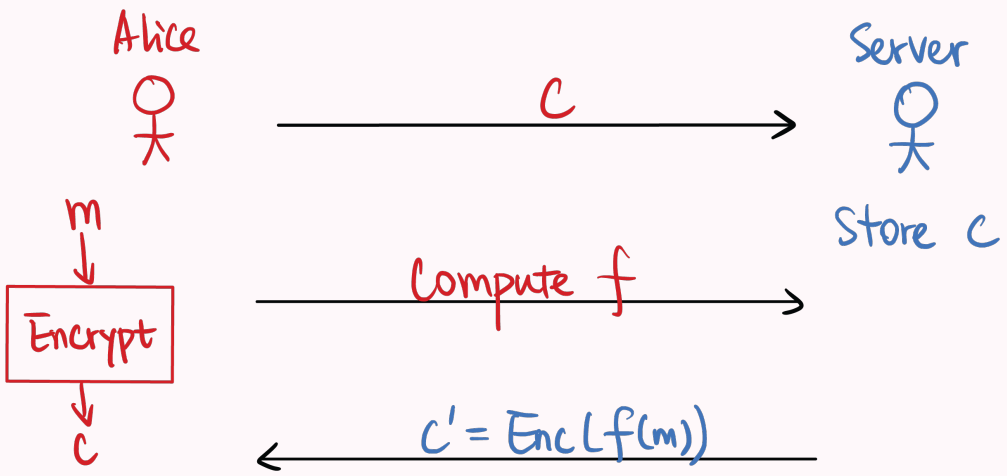


$$c_1 = \mathsf{Enc}(m_1)$$
$$c_2 = \mathsf{Enc}(m_2)$$
$$\Rightarrow \quad c' = \mathsf{Enc}(m_1 + m_2)$$
$$c'' = \mathsf{Enc}(m_1 \cdot m_2)$$

We want to hopefully compute any function in polynomial time!

Example (Outsourced Computation)

Alice has some messages but doesn't have enough compute. There is a server that has *a lot* of compute!



Alice encrypts her data and stores it in the server. At some point, Alice might want to compute a function on the encrypted data on the server, without the server revealing the original data.

This is an example of how fully homomorphic encryption can be useful.

Remark. This problem was not solved until 2009 (when Peihan started her undergrad). Theoretically, it doesn't even seem that possible! Being able to compute functions on ciphertexts that correspond to functions on plaintexts.

To construct fully-homomorphic encryption, we'll be using lattice-based cryptography. This is also the only cryptographic primitive that is quantum-secure[3].

§1.5.4 Further Topics

We might cover some other topics:

- Differential Privacy
- Crypto applications in machine learning
- Crypto techniques used in the blockchain[4]

What else would you like to learn? What else do you want to understand? Do go through the semester with these in mind! *How do I log into Google? How do I send messages to friends?*

Peihan will collect responses at the start and middle of the semester to shape course content.

[3] Everything before this can be broken if quantum computers become mainstream!
[4] One important techniques is Zero-Knowledge proofs, for example.

§1.6 A Quick Survey

Peihan conducted the following poll in-class to gauge content for future lectures. ***By all means, you don't need to know any/all of this going into this course! These will be self-contained in this course!***

Do you know what the following means?

- Polynomial-time algorithm.
- NP-hard problems.
- "a divides b" ($a \mid b$)
- GCDs
- (Extended) Euclidean Algorithms
- Groups
- One-Time pads
- RSA encryption/signature
- Diffie-Hellman Key Exchange
- SHA (hash functions)