CMSC/Math 456: Cryptography (Fall 2022)

Lecture 26

Daniel Gottesman

Administrative

Problem set #8 is due tonight at *midnight*.

Problem set #9 (last problem set) will be available tonight, due next Thursday at midnight.

Material covered today will not be on the final.

Course evaluations are now available to fill out.

Final exam: Monday, Dec. 19, 1:30-3:30 PM

Students taking the final at ADS: Remember to book with them soon.

This class is being recorded

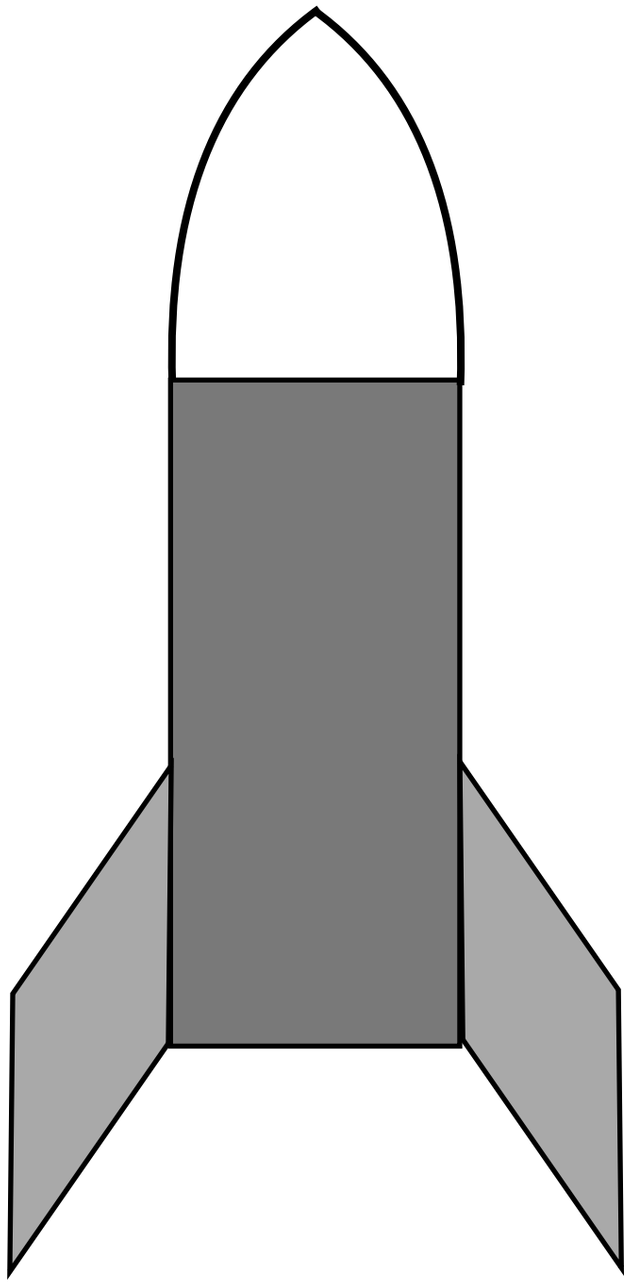
Secret Sharing

Suppose we have a nuclear missile.

This class is being recorded

Secret Sharing

Suppose we have a nuclear missile.

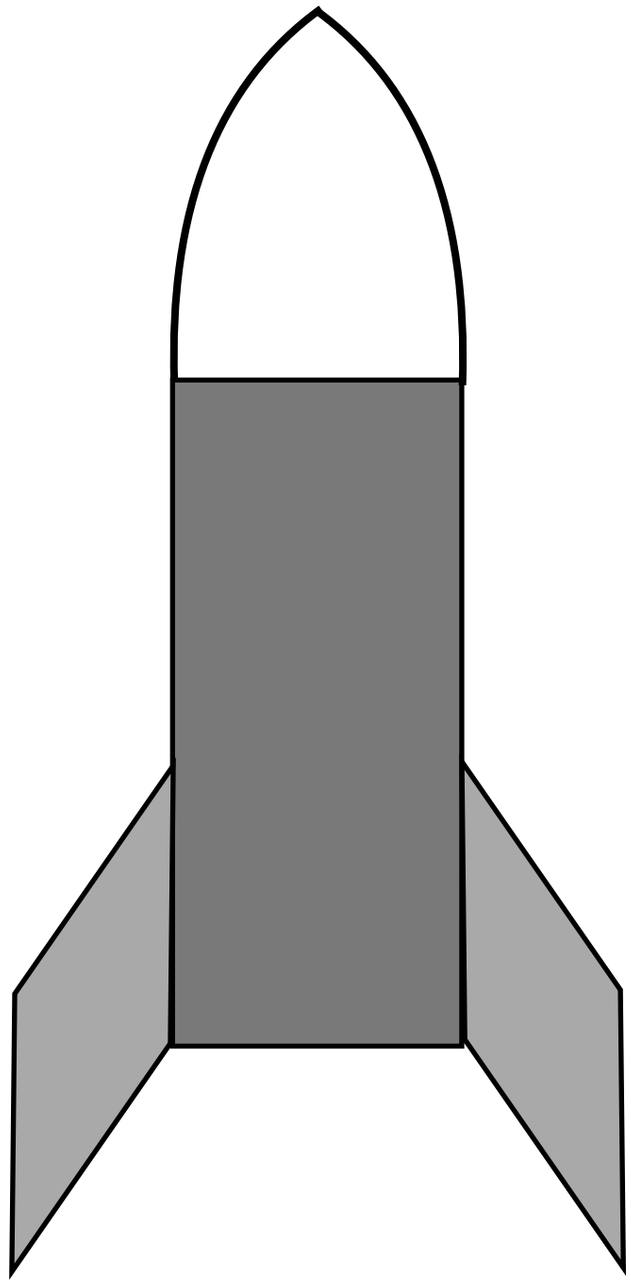


This class is being recorded

Secret Sharing

Suppose we have a nuclear missile.

We don't want one person to be able to launch it by themselves.



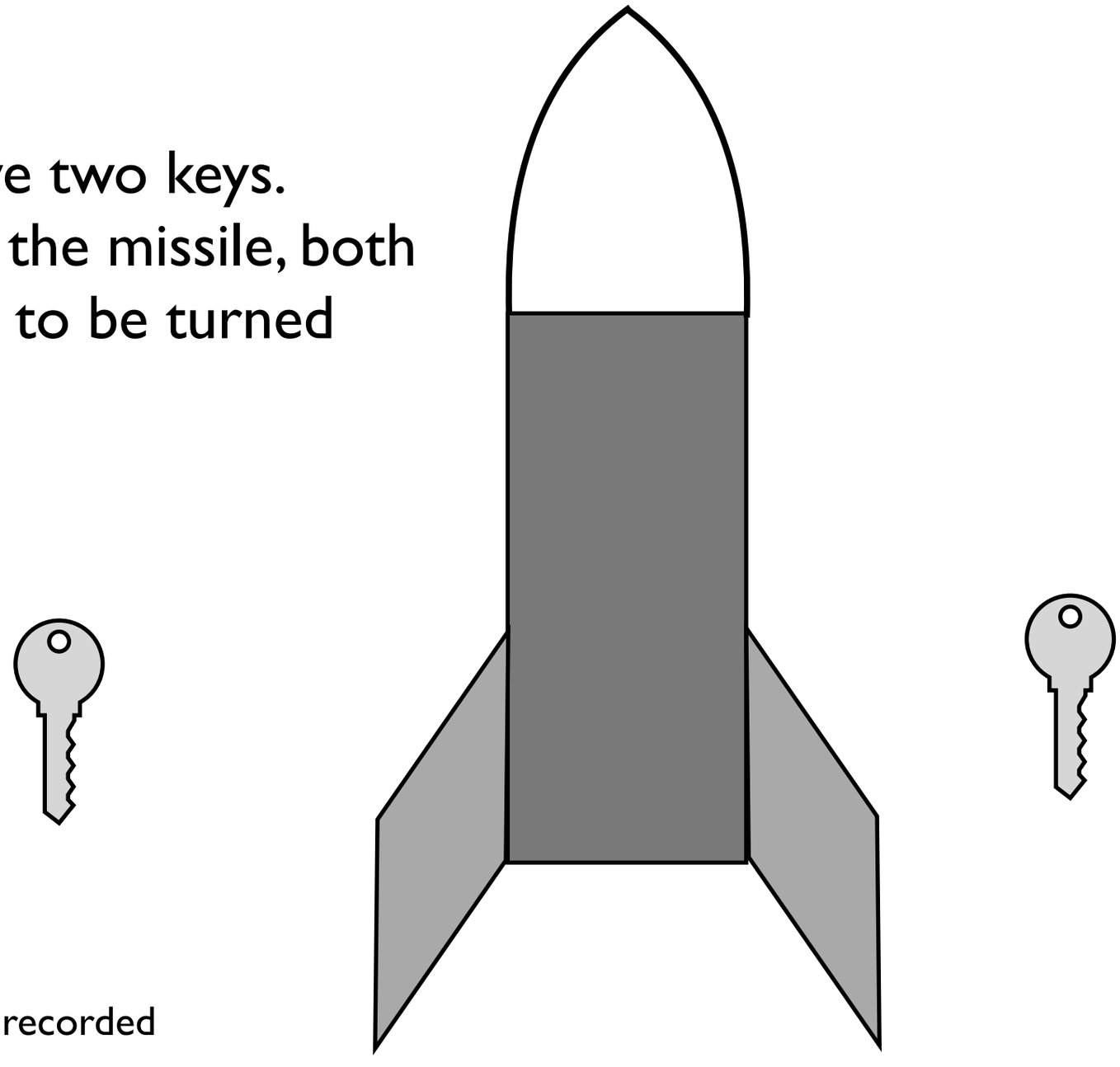
This class is being recorded

Secret Sharing

Suppose we have a nuclear missile.

We don't want one person to be able to launch it by themselves.

So we have two keys.
To launch the missile, both keys need to be turned together.



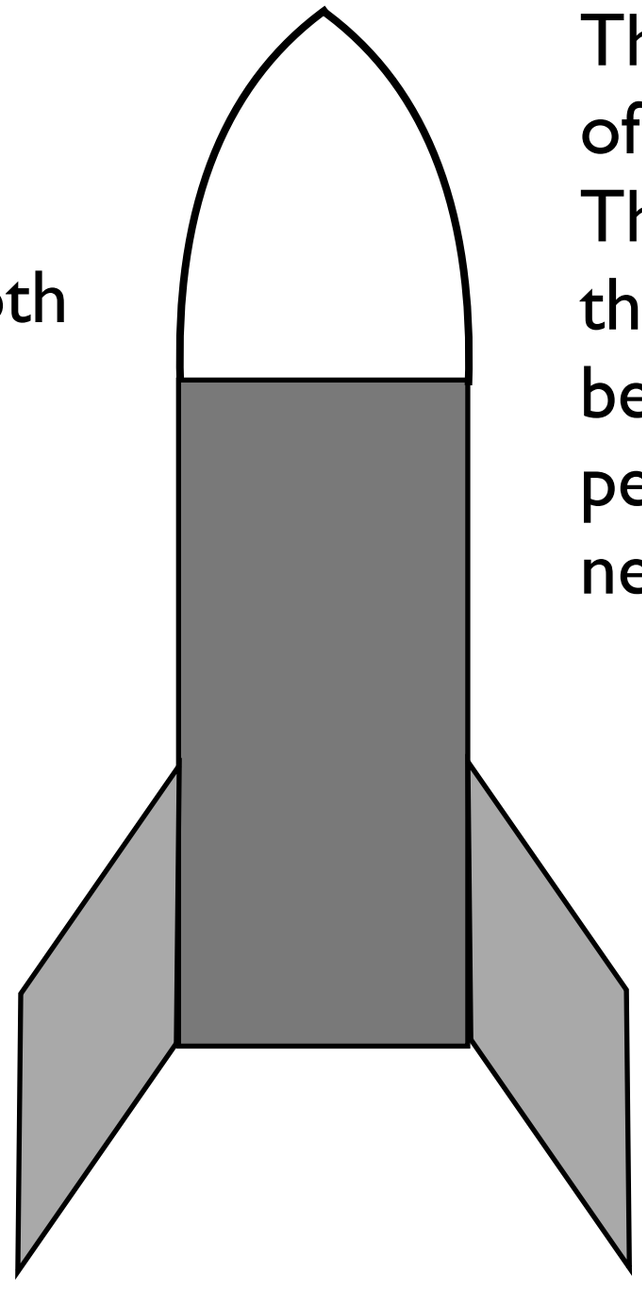
This class is being recorded

Secret Sharing

Suppose we have a nuclear missile.

We don't want one person to be able to launch it by themselves.

So we have two keys.
To launch the missile, both keys need to be turned together.



This is an example of secret sharing:
The code to launch the missile is split between two people and both are needed to access it.

This class is being recorded

(n,n) Secret Sharing

Suppose we have a secret bit string s. We wish to share it among n people by giving them each shares s_i such that:

- When all n people pool their shares, they can reconstruct the secret s.
- Any group of n-1 or fewer people cannot get any information about s using their shares.

Proposal:

Let shares s_1 through s_{n-1} be random bit strings, and let $s_n = s_1 \oplus s_2 \oplus \cdots \oplus s_{n-1} \oplus s$.

Vote: Does this work? (Yes/No/With some extra assumption)

This class is being recorded

(n,n) Secret Sharing

Suppose we have a secret bit string s. We wish to share it among n people by giving them each shares s_i such that:

- When all n people pool their shares, they can reconstruct the secret s.
- Any group of n-1 or fewer people cannot get any information about s using their shares.

Proposal:

Let shares s_1 through s_{n-1} be random bit strings, and let $s_n = s_1 \oplus s_2 \oplus \cdots \oplus s_{n-1} \oplus s$.

Vote: Does this work? (Yes/No/With some extra assumption)

Answer: Yes! To reconstruct the secret, take the XOR of all shares. But n-1 shares are just n-1 random bit strings.

This class is being recorded

Threshold Secret Sharing

We can also make (t,n)-threshold secret sharing schemes with n shares so that any t shares can be used to reconstruct the secret s but t-1 shares have no information.

Here's one way:

Work mod p, p>n prime.

Choose random numbers $a_1, \ldots, a_{t-1}$ mod p.

Define the polynomial $f(x) = s + \sum_{i=1}^{t-1} a_i x^i \bmod p$.

Let the ith share $s_i = f(i)$ (i = 1, … n).

We can reconstruct the full polynomial given its value at any t points since it has degree t-1. But with only t-1 shares, f(0)=s could have any value and by considering it a t-th point, we could always reconstruct a polynomial giving that.

This class is being recorded

More Secret Sharing

We can devise secret sharing schemes with other access structures, namely specifications of which people can reconstruct the secret. For instance, if we split the secret between Alice, Bob, and Charlie, we might allow Alice full access to the secret by herself but Bob and Charlie can only see it if they both cooperate.

Any access structure is achievable provided that if a set S of shares gives access, then any set containing S does as well.

Note that this can be done with information-theoretic security: Too-small groups cannot get any information about the secret regardless of their computational power.

Computational assumptions allow more efficient protocols.

This class is being recorded

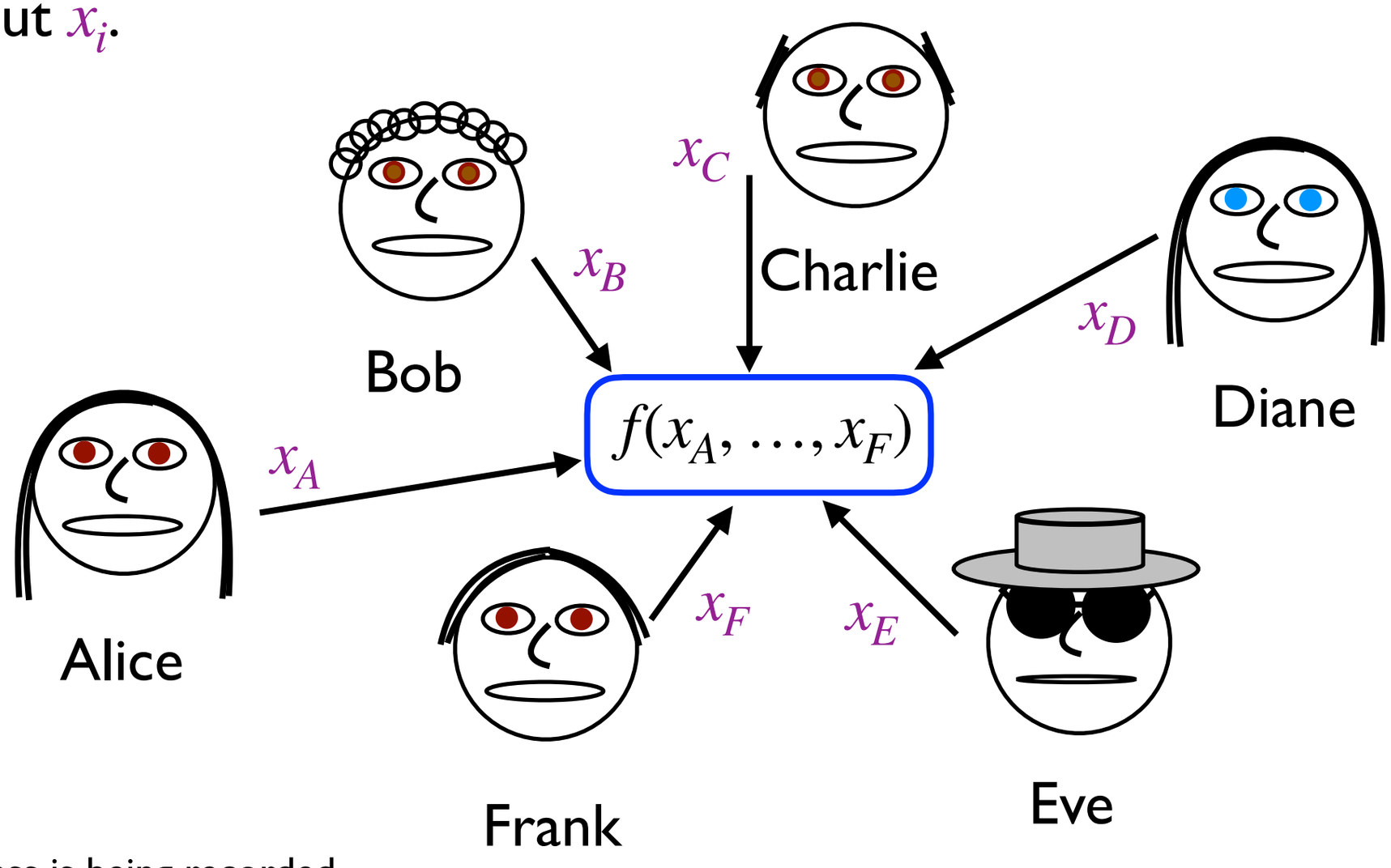
Applications of Secret Sharing

- Joint checking accounts
- Backup of critical but sensitive information
- Distributing information in cloud storage between servers to protect against both data loss in case of server failure and against privacy breach in case of server compromise
- As a component in more complicated cryptographic protocols.

This class is being recorded

Secure Multiparty Computation

A group of n people wishes to compute something but they don't trust each other. Each has their own input x_i and they wish to compute $f(x_1, \ldots, x_n)$ so that even if one or more people are dishonest and maybe working together, the function is computed correctly and no one learns anything more than f and their own input x_i.



This class is being recorded

Examples of Multiparty Computation

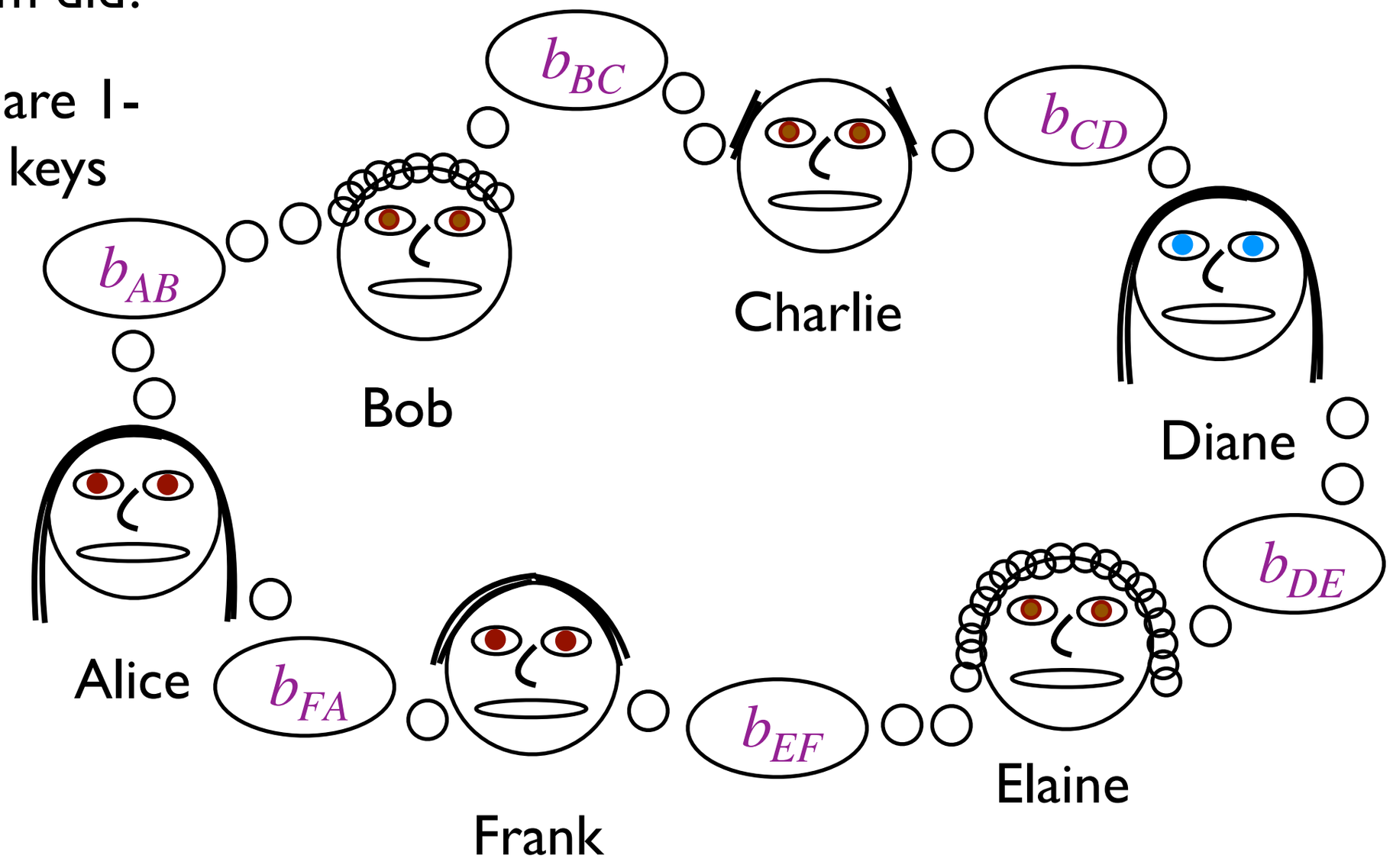
- Data-mining on sensitive data: Here the inputs are the sensitive data from one or more sources and the function is the statistical result of the data mining.
- Electronic voting: The inputs are each person's vote and the function computes the winner of the election.
- On-line auctions: The inputs are each person's bid and the function computes the winner of the auction.
- Anonymous communication: The input is a message and a recipient that someone wishes to send. The function is simply delivering the message to that person.
- On-line poker: There is no private input here, but each person has a private output, their cards.

This class is being recorded

Dining Cryptographers

A group of cryptographers have dinner at a restaurant. Someone pays anonymously — or maybe it was the NSA that paid. Can they determine if the NSA paid without learning which one of them did?

They all share 1-bit private keys with their neighbors.

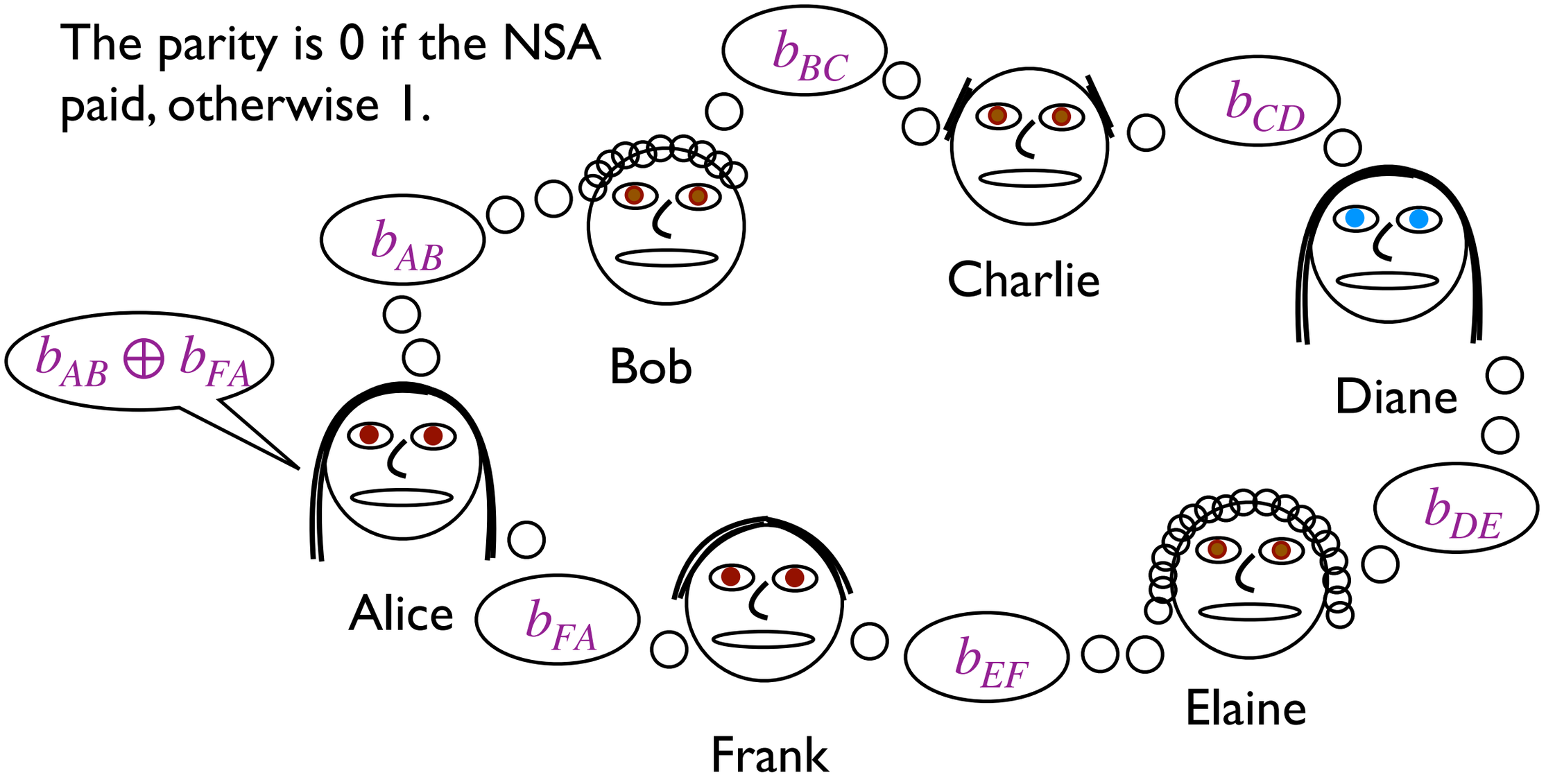


This class is being recorded

Dining Cryptographers

Each announces the XOR of their two secret bits if they didn't pay, or the NOT of that if they did pay. Each announcement looks random to everyone else, but if we take the parity of all announcements, we see each $b_{i,i+1}$ appears twice. They cancel.

The parity is 0 if the NSA paid, otherwise 1.



This class is being recorded

Multiparty Computation Results

There are powerful general results showing multiparty computation is possible under a variety of different assumptions and security definitions.

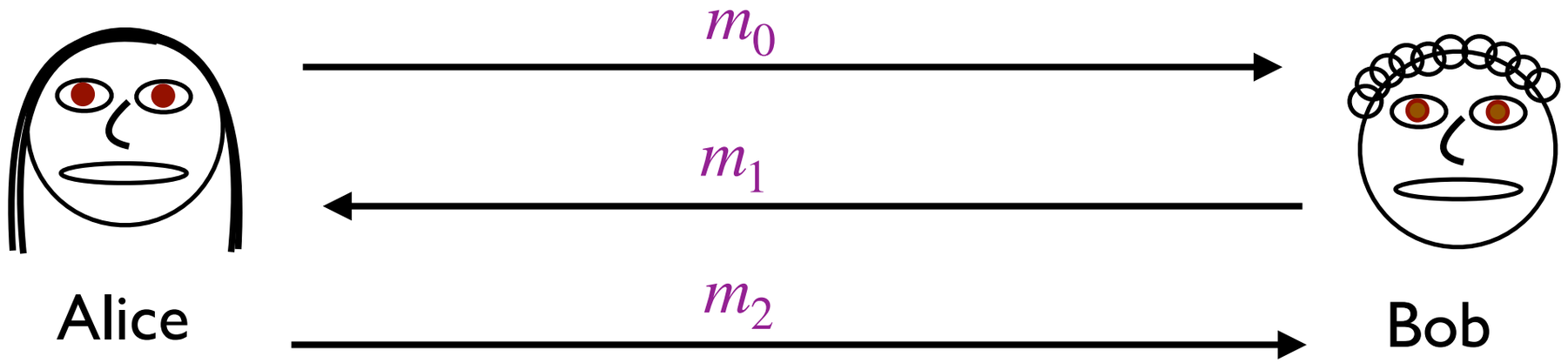
- The general protocols are complicated and inefficient, but there are more efficient protocols for specific tasks.
- Multiparty computation often makes use of specific cryptographic primitives like bit commitment (which we discussed earlier) and secret sharing.
- Some protocols provide information-theoretic security while others (with looser assumptions) provide computational security.

One example, and also a primitive used to build other multiparty computations, are zero-knowledge proofs ….

This class is being recorded

Zero-Knowledge Proofs

In a zero-knowledge proof (ZKP), Alice convinces Bob of the truth of some statement while giving Bob no additional information and no ability to do anything he couldn't do before (like convince someone else of the truth of that statement).



We have already seen one example of a zero-knowledge proof: An identification protocol. Alice convinces Bob that she is Alice (specifically, she has Alice's private key), but Bob is unable to convince anyone else of the same thing.

This class is being recorded

ZKP Definition

An interactive proof is zero knowledge if an adversary could have generated the transcript of the protocol, along with any side information, by itself without interacting with Alice.

> Why does this imply Bob hasn't learned anything or gained any capability that he didn't have before?
>
> Because Bob doesn't know anything more than the contents of the transcript and any random bits he used during the protocol; and he could have generated all of those without Alice.

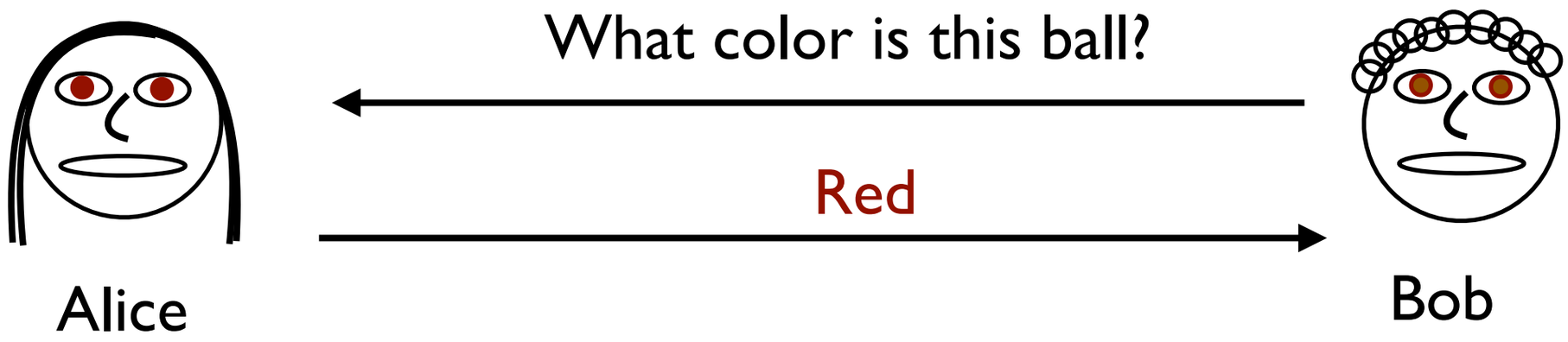
ZKP doesn't rule out Bob playing man-in-the-middle and relying messages from Charlie, resulting in Alice actually convincing Charlie rather than Bob. But it does rule out Bob convincing Charlie after Alice is done.

This class is being recorded

ZKP Example

Suppose you have a color-blind friend who can't tell the difference between red and green and doesn't even believe that there is a difference.

How can you prove to them that it is possible without giving them any extra information or ability?



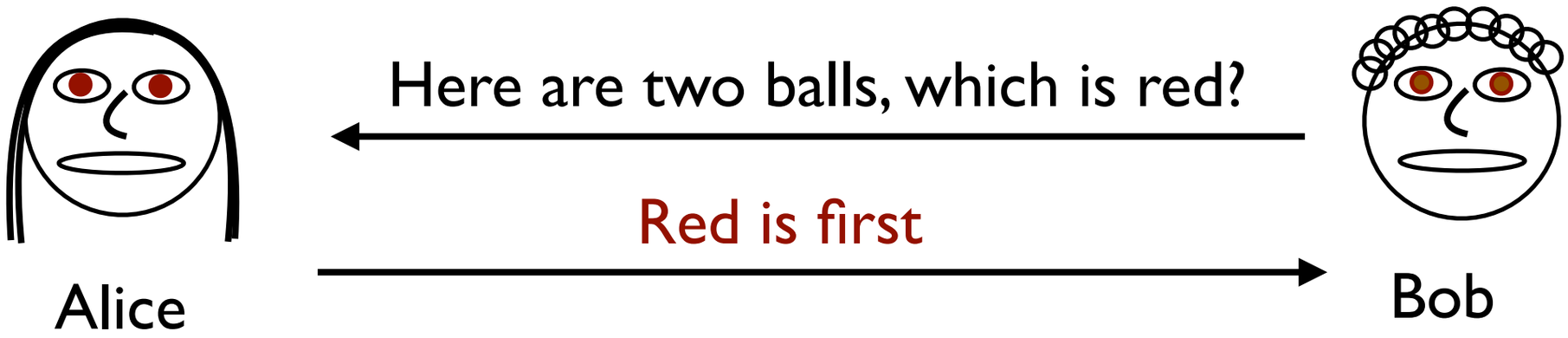
Vote: Does this work? (Yes/No)

This class is being recorded

ZKP Example

No! Bob can use Alice to determine the color of a ball. It is not zero knowledge.

Try again:

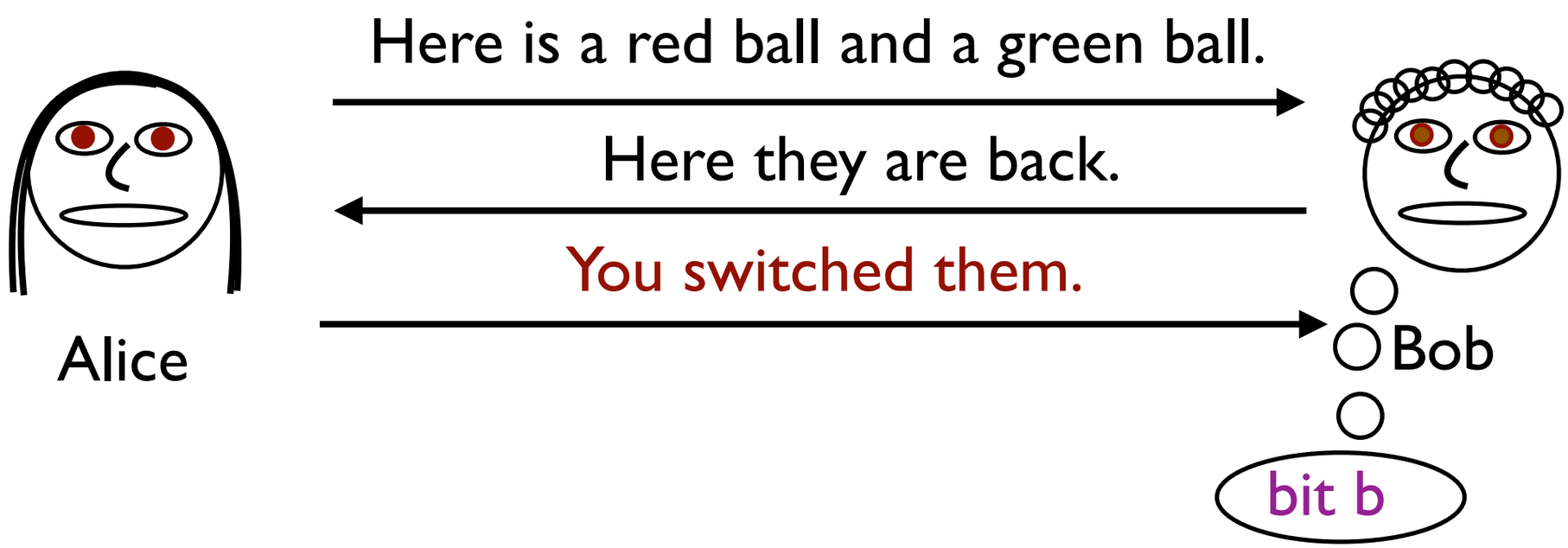


Better … but Bob can still sometimes use this to determine the color of balls.

ZKP Example

We need to make sure Alice is not giving Bob any decryption information.

Bob chooses a random bit b and switches the balls if b=1.

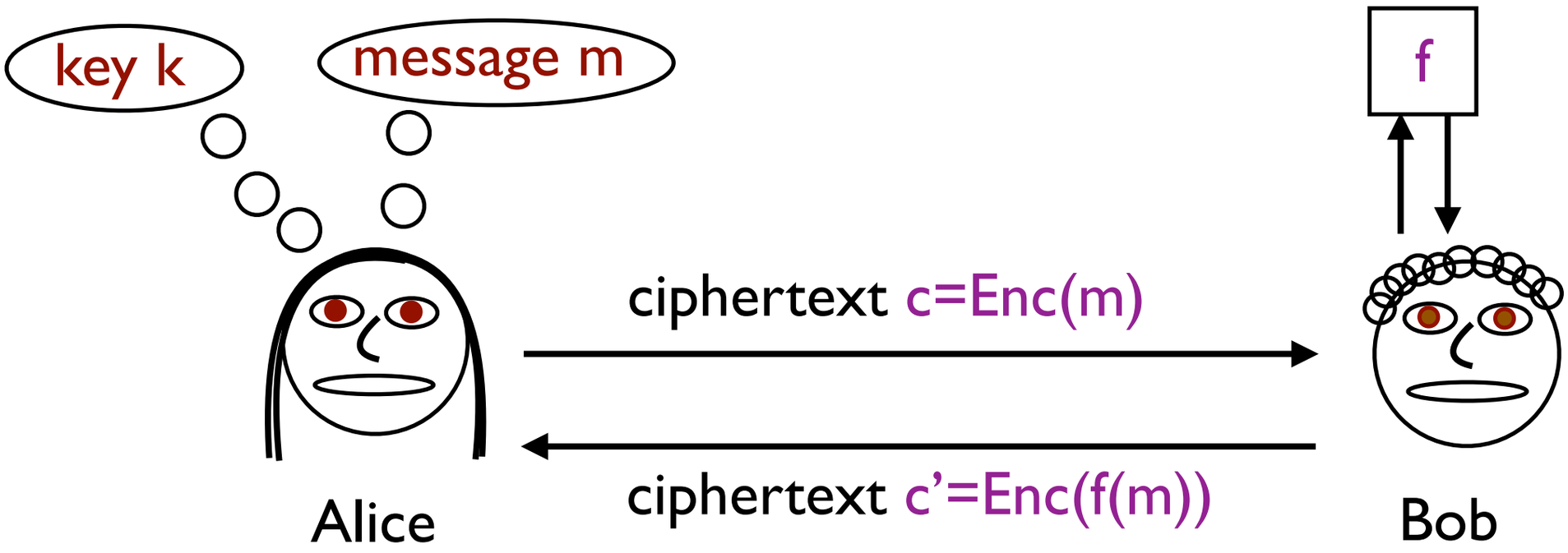


Now Alice is not telling Bob anything he didn't already know. Since Alice provides the balls, Bob cannot use it to learn anything about another ball he has.

This class is being recorded

Homomorphic Encryption

Another useful tool for multiparty computation is homomorphic encryption.



In a homomorphic encryption scheme, there is a way to perform computations on ciphertexts without decrypting them. Fully homomorphic encryption allows arbitrary computations. (There are also weaker versions, allowing some computations but not others.)

This class is being recorded

Practical Homomorphic Encryption

Fully homomorphic encryption can be done using lattice-based cryptosystems such as LWE-based encryption.

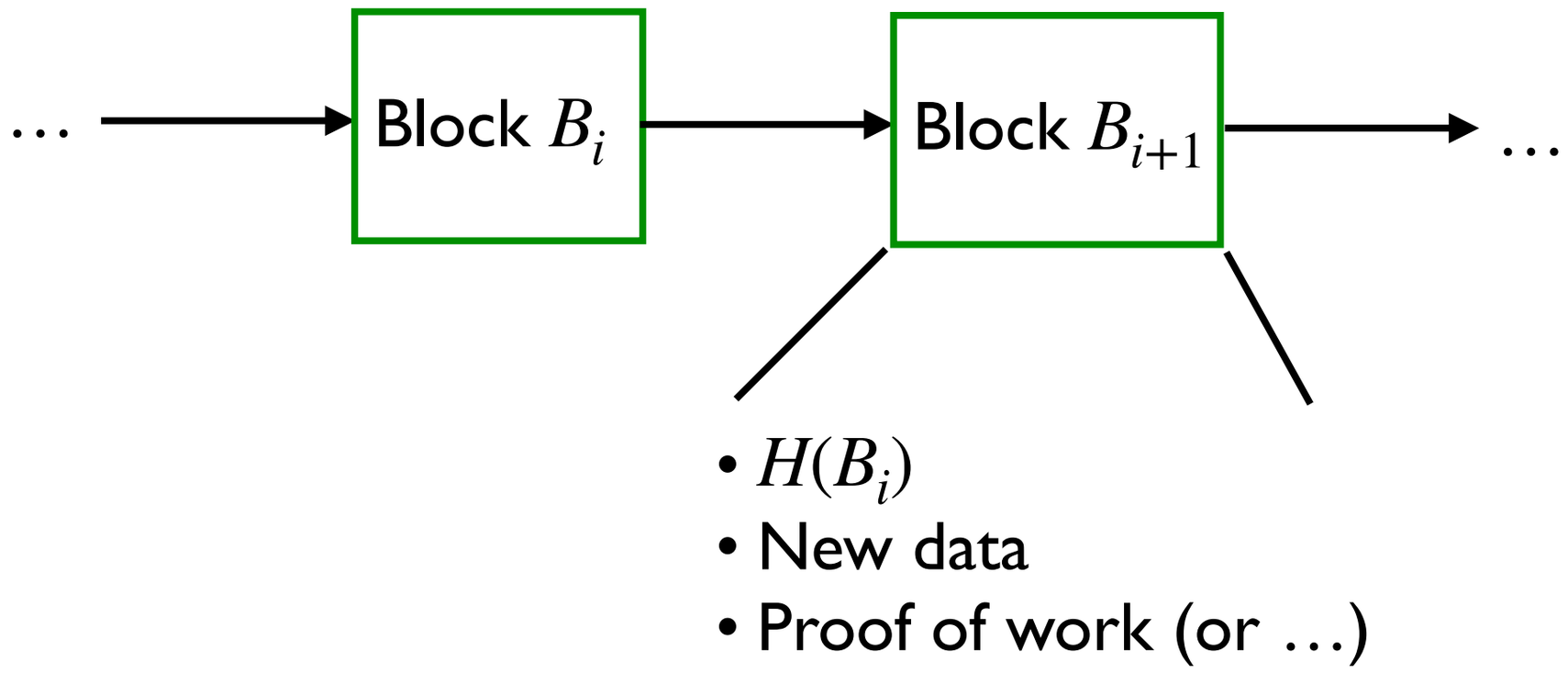
It tends to be very slow. However, improved fully homomorphic encryption schemes have been the subject of intense research and they are finally approaching practical levels.

This class is being recorded

Blockchain

Blockchain is used in cryptocurrencies, smart contracts, NFTs, …

The basic idea is a simple application of hash functions.



Each new block contains a hash of the previous block; this links the chain together and makes it hard (in the cryptographic sense) to change the previous entries. The proof of work gives someone authority to add a block, and the new data makes something happen.

This class is being recorded