COS433/Math 473: Cryptography

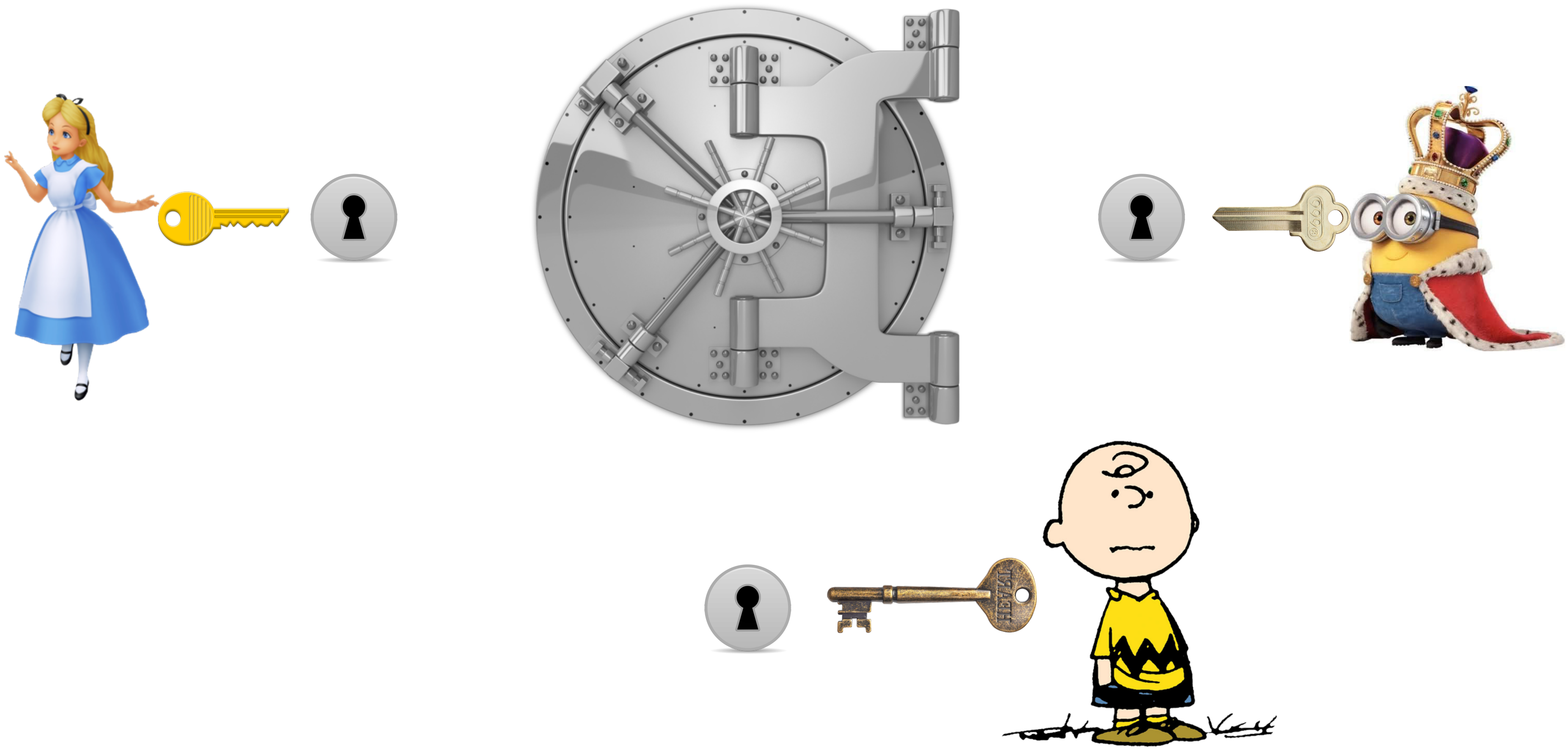
Mark Zhandry

Princeton University

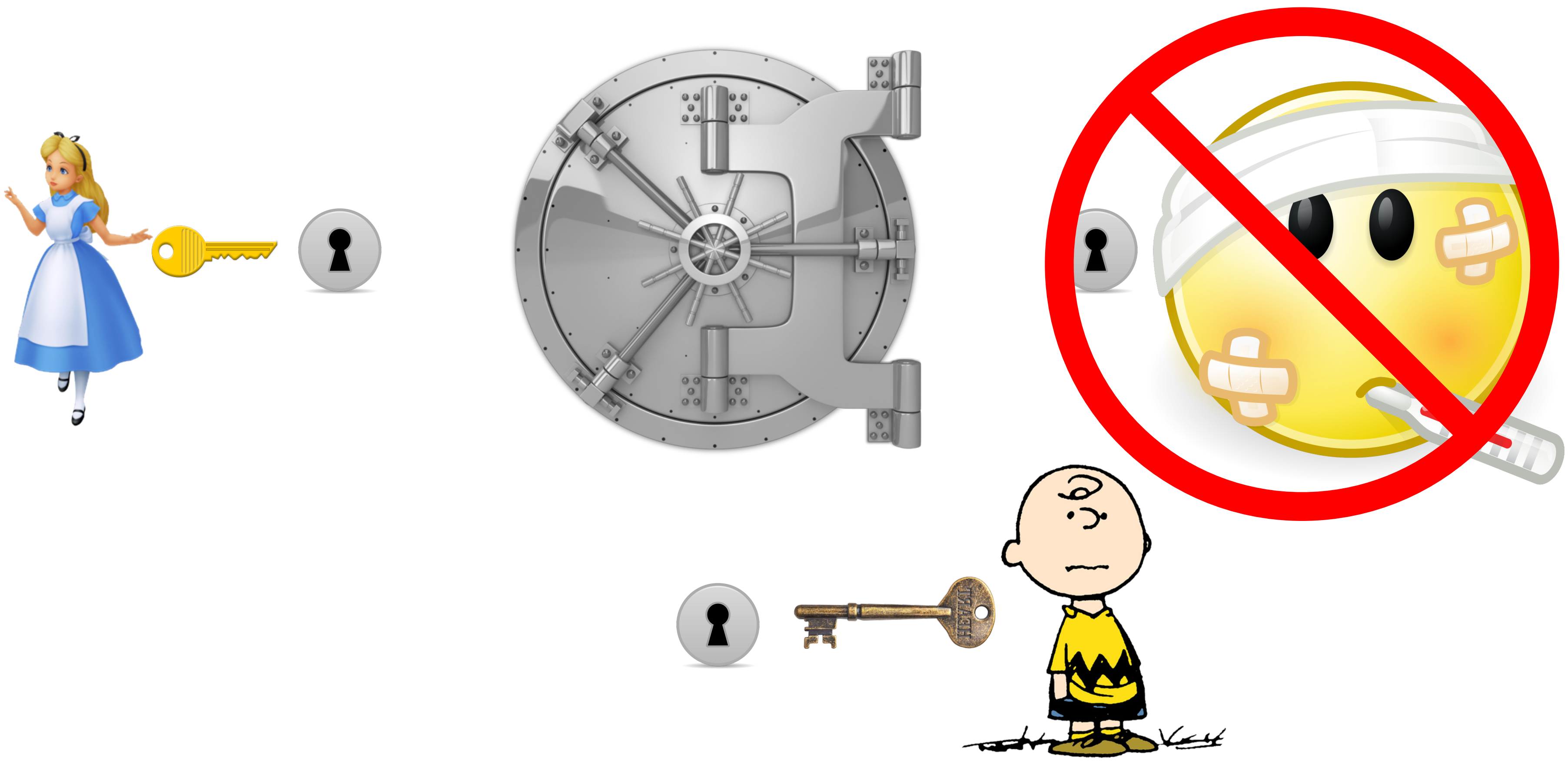
Spring 2018

Secret Sharing

Vault should only open if both Alice and Bob are present



Vault should only open if Alice, Bob, and Charlie are all present



Vault should only open if any two of Alice, Bob, and Charlie are present

(Threshold) Secret Sharing

Syntax:

Share(k,t,n) outputs $(sh_1,\ldots,sh_n)$

Recon($(sh_i)_{i\in S}$ **)** outputs **k'**

Correctness: $\forall$ **S s.t.** $|S|\geq t$

If $(sh_i)_{i=1,\ldots,n} \leftarrow$ **Share(k,t,n),** then

$\Pr[\text{Recon}((sh_i)_{i\in S}) = k] = 1$

(Threshold) Secret Sharing

Security:

For any S, $|S|<t$, given $(sh_i)_{i\in S}$, should be impossible to recover k

$$(sh_i)_{i\in S}: (sh_i)_{i=1,\dots,n} \leftarrow Share(k_0,t,n)$$

$$\approx$$

$$(sh_i)_{i\in S}: (sh_i)_{i=1,\dots,n} \leftarrow Share(k_1,t,n)$$

n-out-of-**n** Secret Sharing

Share secret **k** so that can only reconstruct **k** if all **n** users get together

Ideas?

Shamir Secret Sharing

Let $\mathbf{p}$ be a prime $> \mathbf{n}, \ \geq \#(\mathbf{k})$

Share(k,t,n):

- Choose a random polynomial $\mathbf{P}$ of degree $\mathbf{t-1}$ where $\mathbf{P(0) = k}$
- $\mathbf{sh_i = P(i)}$

Recon($(\mathbf{sh_i})_{i \in S}$ **):** use shares to interpolate $\mathbf{P}$, then evaluate on $\mathbf{0}$

Shamir Secret Sharing

Correctness:

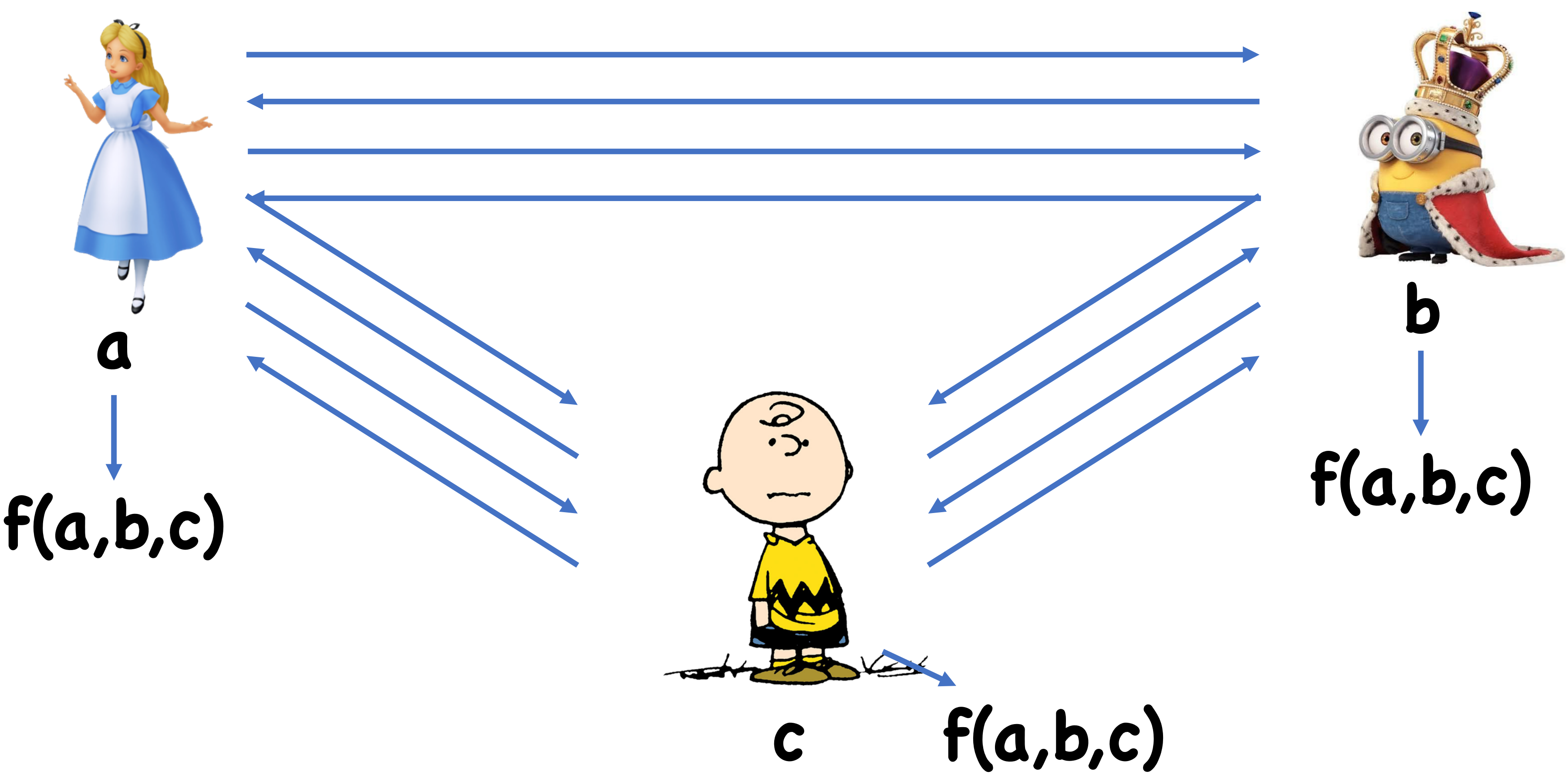
- **t** input/outputs (shares) are enough to interpolate a degree **t-1** polynomial

Security:

- Given just **t-1** inputs/outputs, **P(0)** is equally likely to be any value

Multiparty Computation

Multiparty Computation



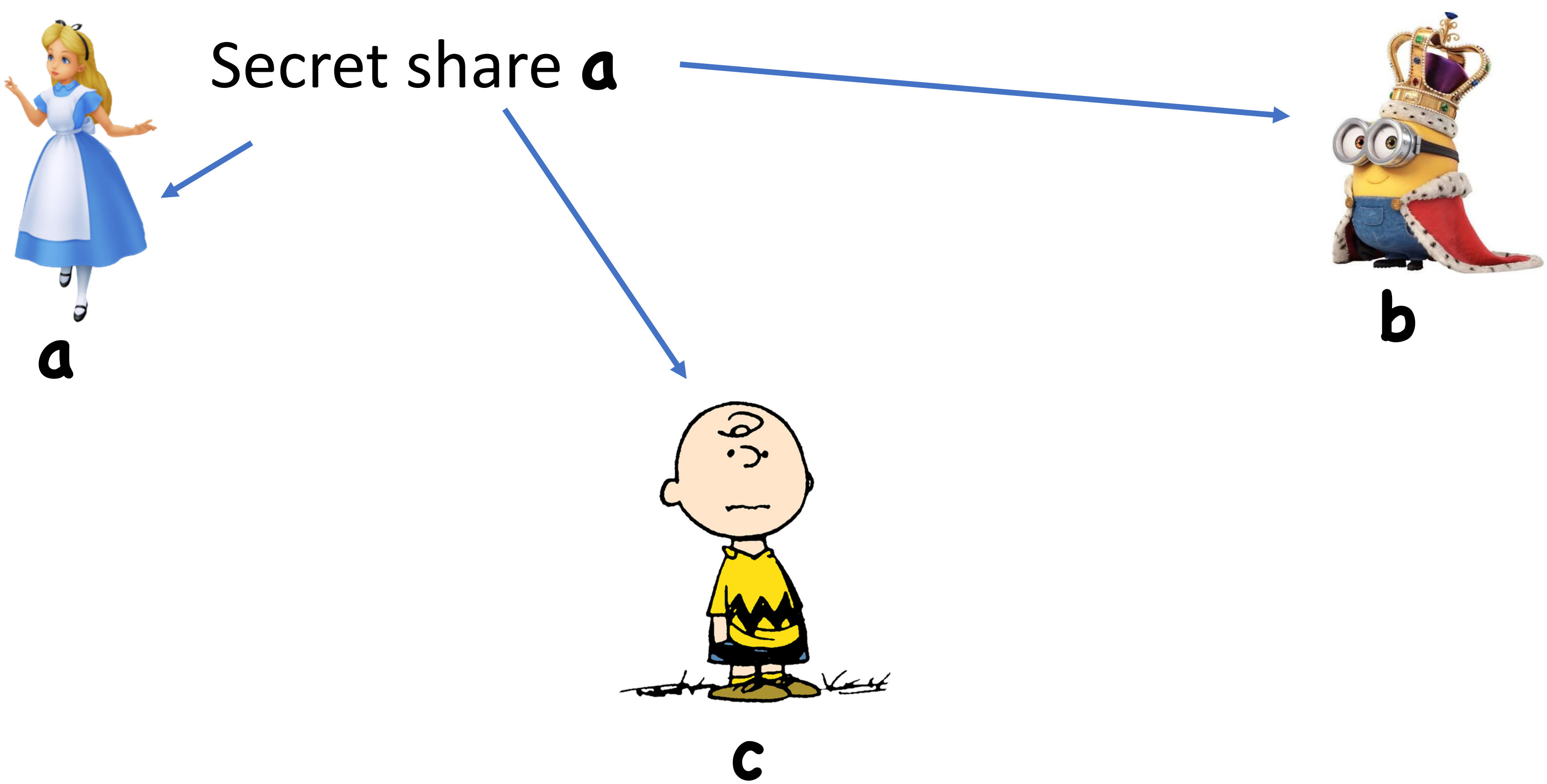
Multiparty Computation

Observation 1: Shamir secret sharing is additively homomorphic:

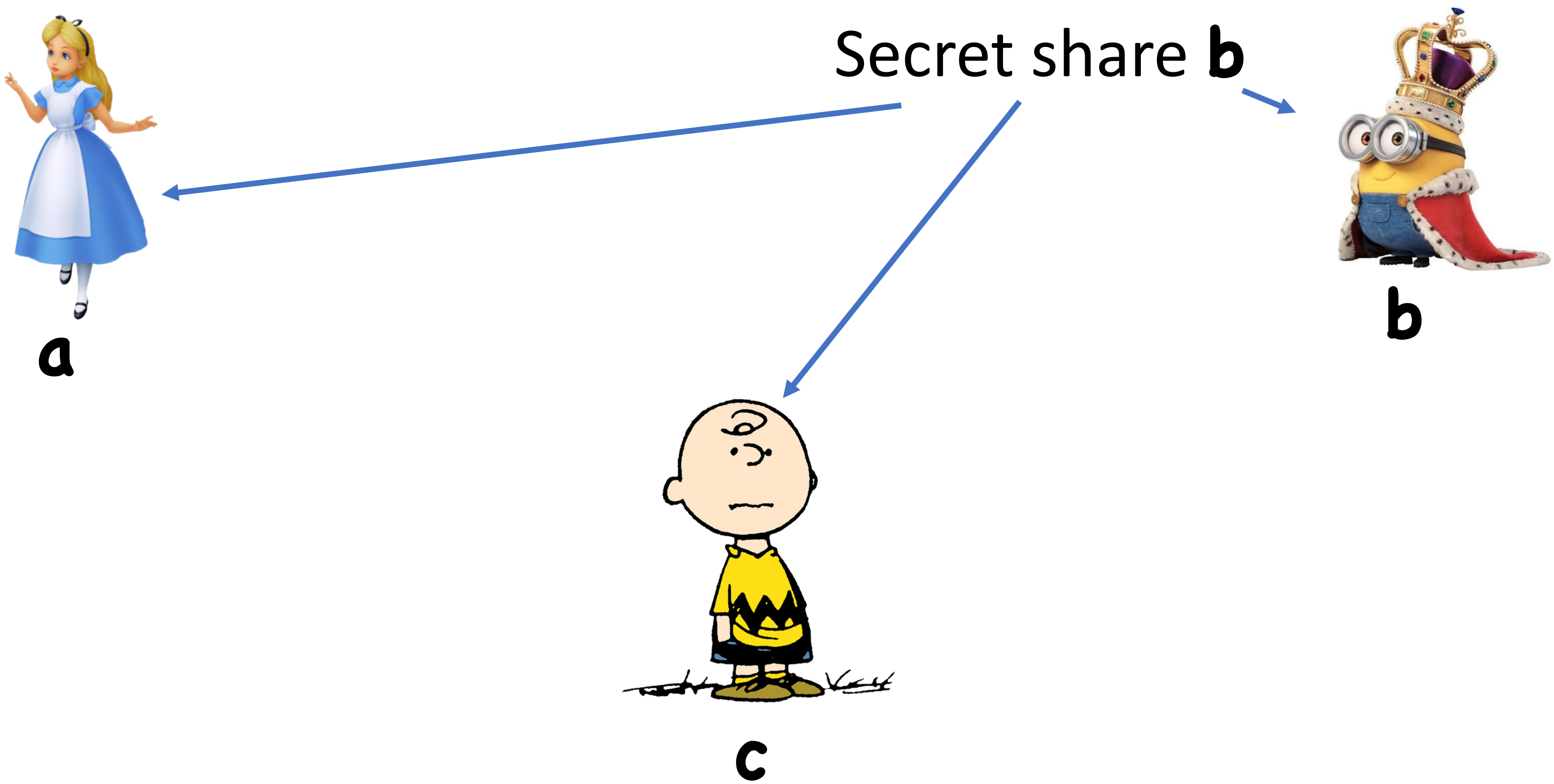
Given shares $\mathbf{sh_1}$ of $\mathbf{x_1}$ and $\mathbf{sh_2}$ of $\mathbf{x_2}$, $\mathbf{r\times sh_1+s\times sh_2}$ is a share of $\mathbf{r\times x_1+s\times x_2}$

- $\mathbf{sh_1 = P_1(i)}$, $\mathbf{sh_2 = P_2(i)}$, so
  $$\mathbf{r\times sh_1+s\times sh_2 = (r\times P_1+s\times P_2)(i)}$$
- $\mathbf{r\times P_1+s\times P_2}$ has same degree

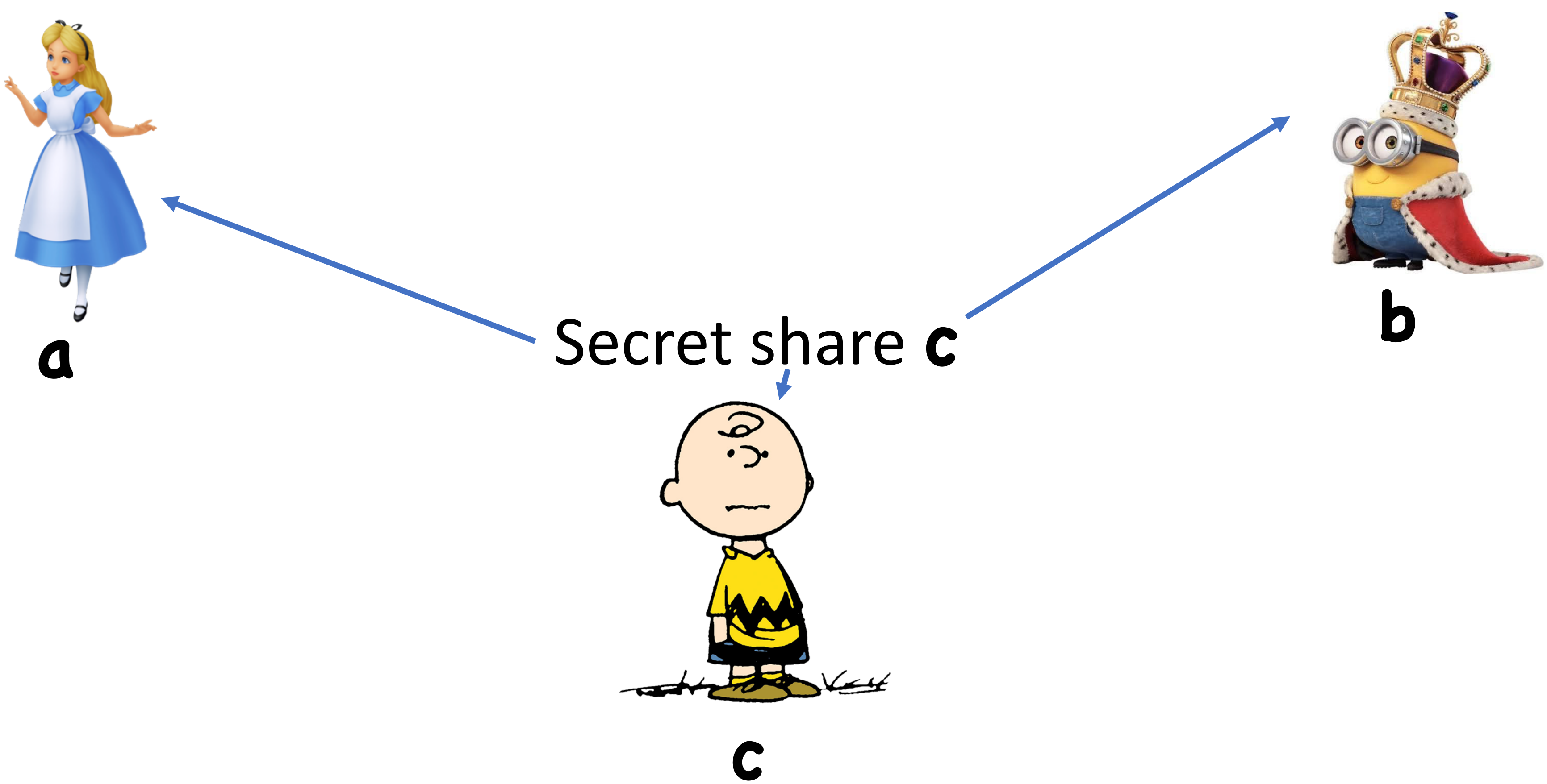
MPC for linear **f**



MPC for linear **f**



MPC for linear **f**



MPC for linear **f**

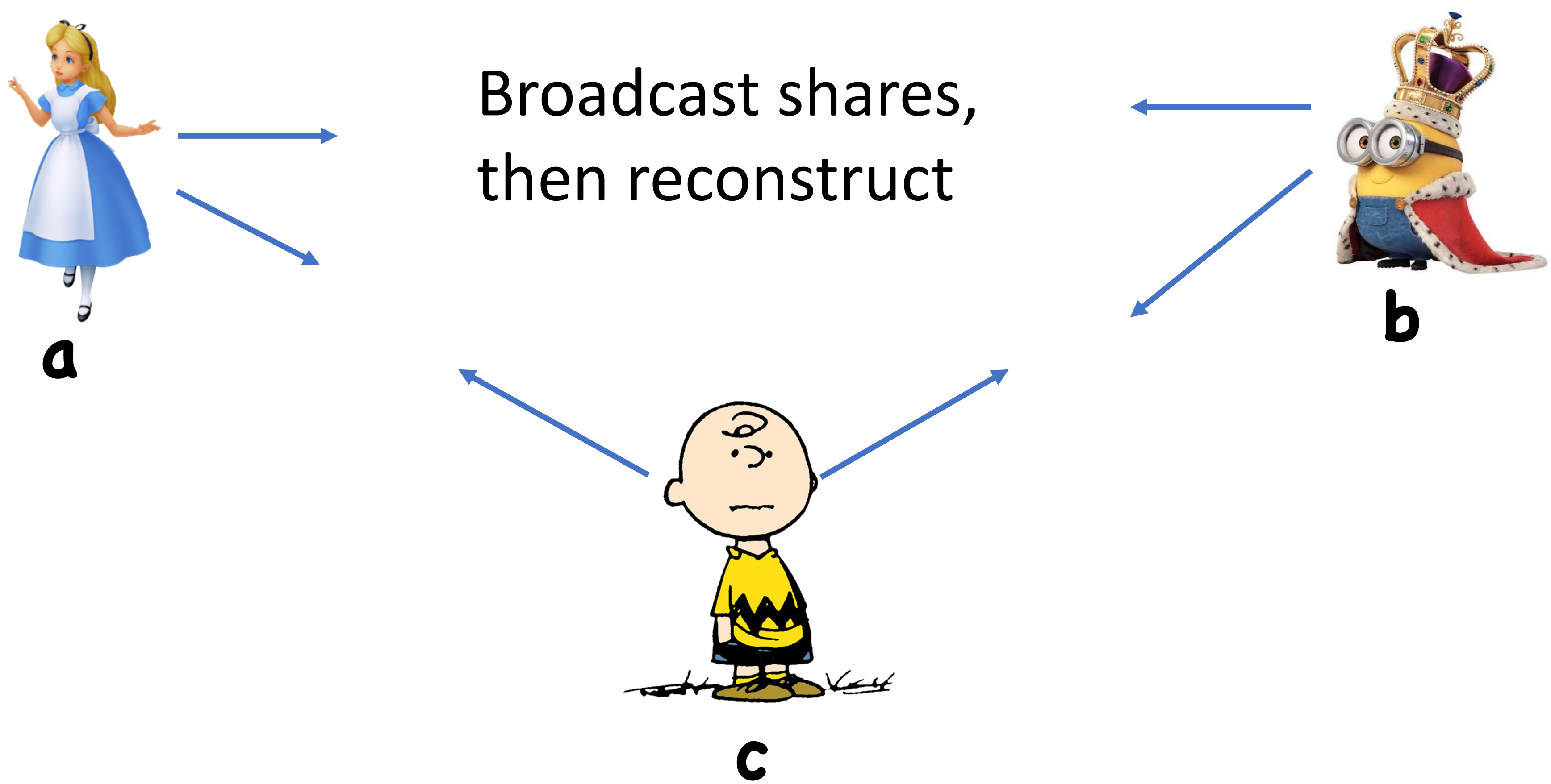
a

Locally compute
shares of **f(a,b,c)**

b

c

MPC for linear **f**



MPC for General f

Observation 2: Shamir Secret Sharing is sort of multiplicatively homomorphic

Given shares sh_1 of x_1 and sh_2 of x_2, $sh_1 \times sh_2$ is a share of $x_1 \times x_2$, but with a different threshold

- $sh_1 = P_1(i)$, $sh_2 = P_2(i)$, so
  $$sh_1 \times sh_2 = (P_1 \times P_2)(i)$$
- $P_1 \times P_2$ has degree $2d$

Idea: can do multiplications locally, and then some additional interaction to get degree back to d

MPC for General f

To maintain correctness, need threshold to stay at most n

- But multiplying doubles threshold, so need $t \leq n/2$
- This means scheme broken if adversary corrupts n/2 users.
- Known to be optimal for "information-theoretic" MPC

Using crypto (e.g. one-way functions), can get threshold all the way up to n

MPC for Malicious Adversaries

So far, everything assumes players act honestly, and just want to learn each other's inputs

But what if honest players deviate from protocol?

Idea: use ZK proofs to prove that you followed protocol without revealing your inputs

Cryptocurrency

Features of Physical Cash

Essentially anonymous

Hard to counterfeit

Easy to verify

Limitations of Physical Cash

Cannot be used online

- Instead, need to involve banks
- Banks see all transactions
- Merchants can also track you

Requires central government to issue

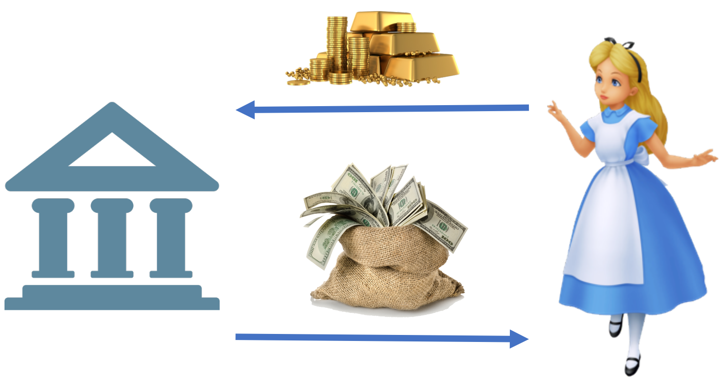
- Ok for most people, but maybe you don’t trust the government

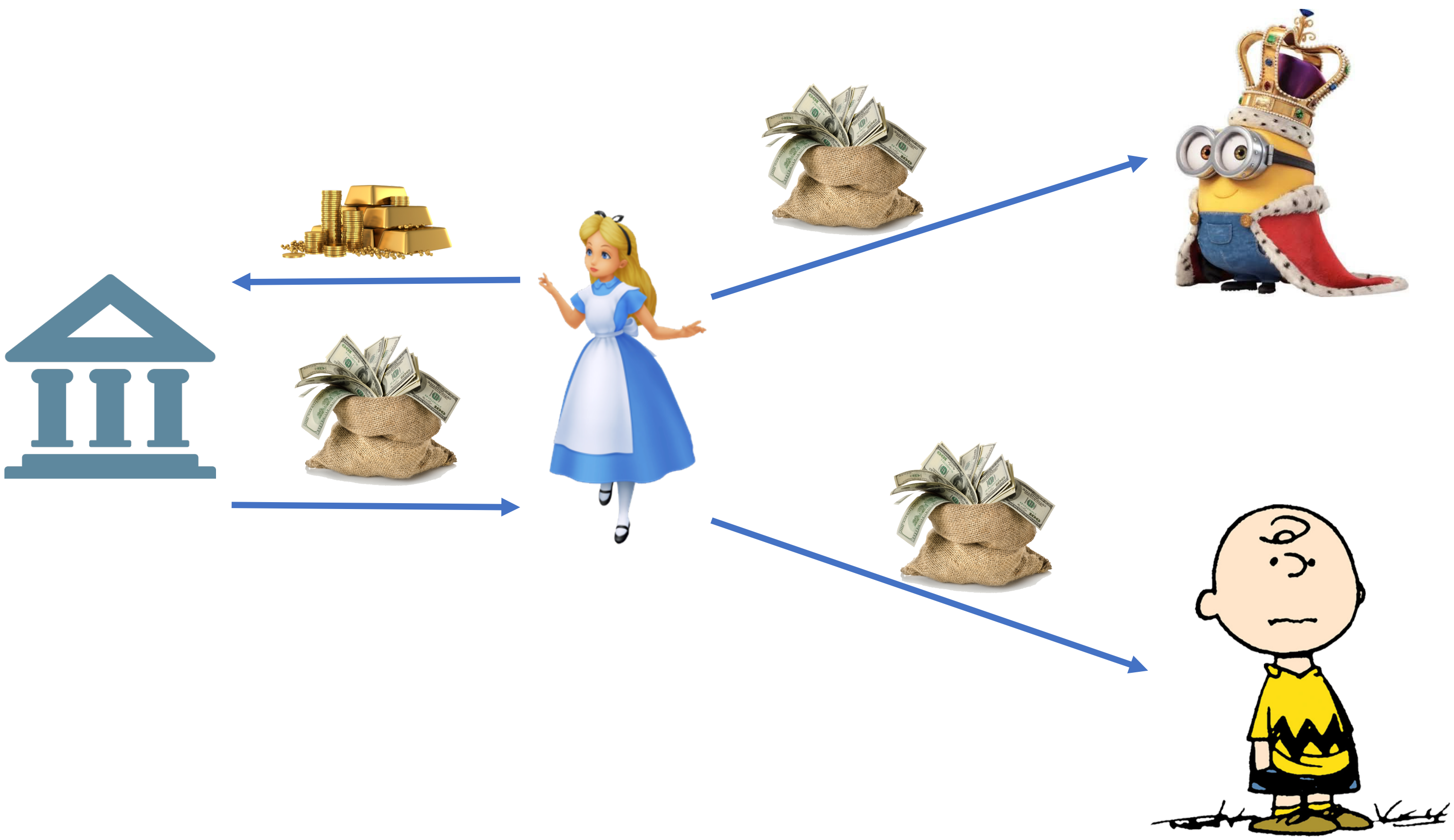
Digital Cash

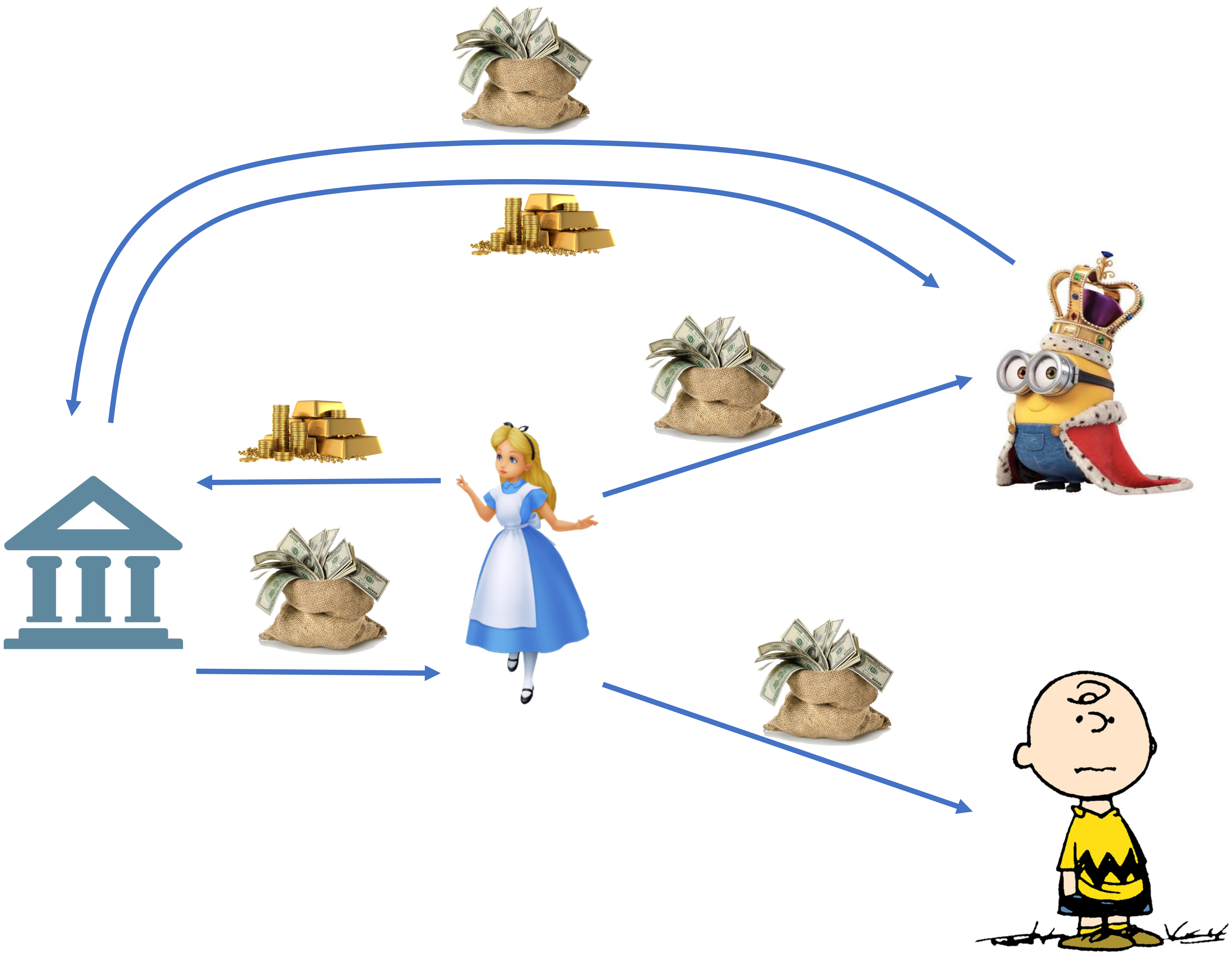
Currency is now 1s and 0s

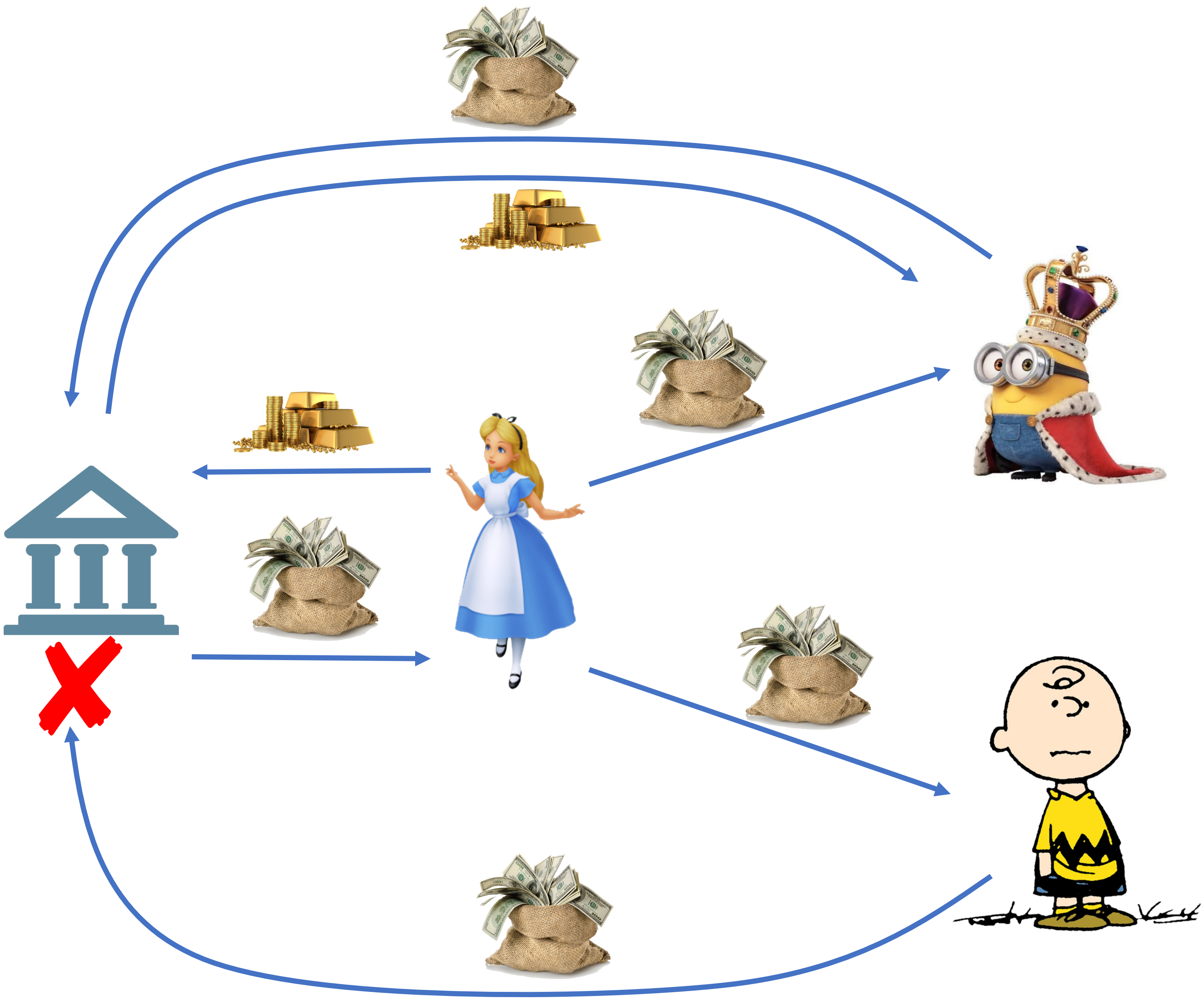
Crypto can make digital currency easy to verify, hard to mint

Major challenge: prevent double spending
(Also decentralizing minting process)





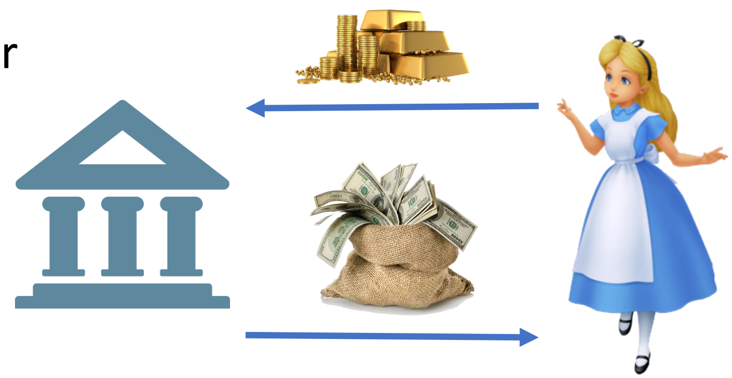




Solution: Public Ledger

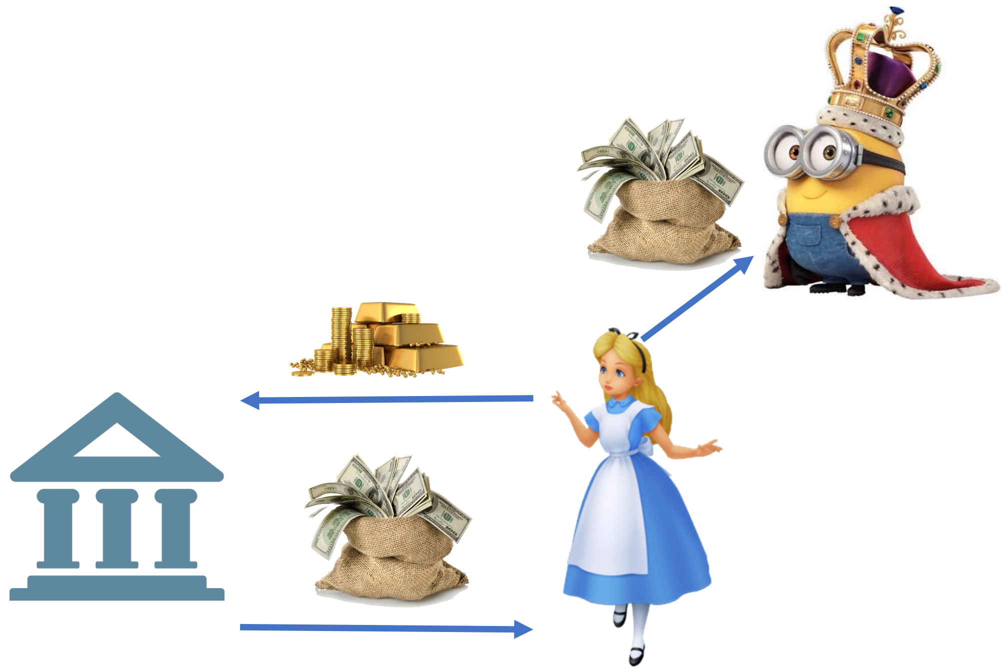
Bank transfers $$ to Alice

Each bill has unique serial number



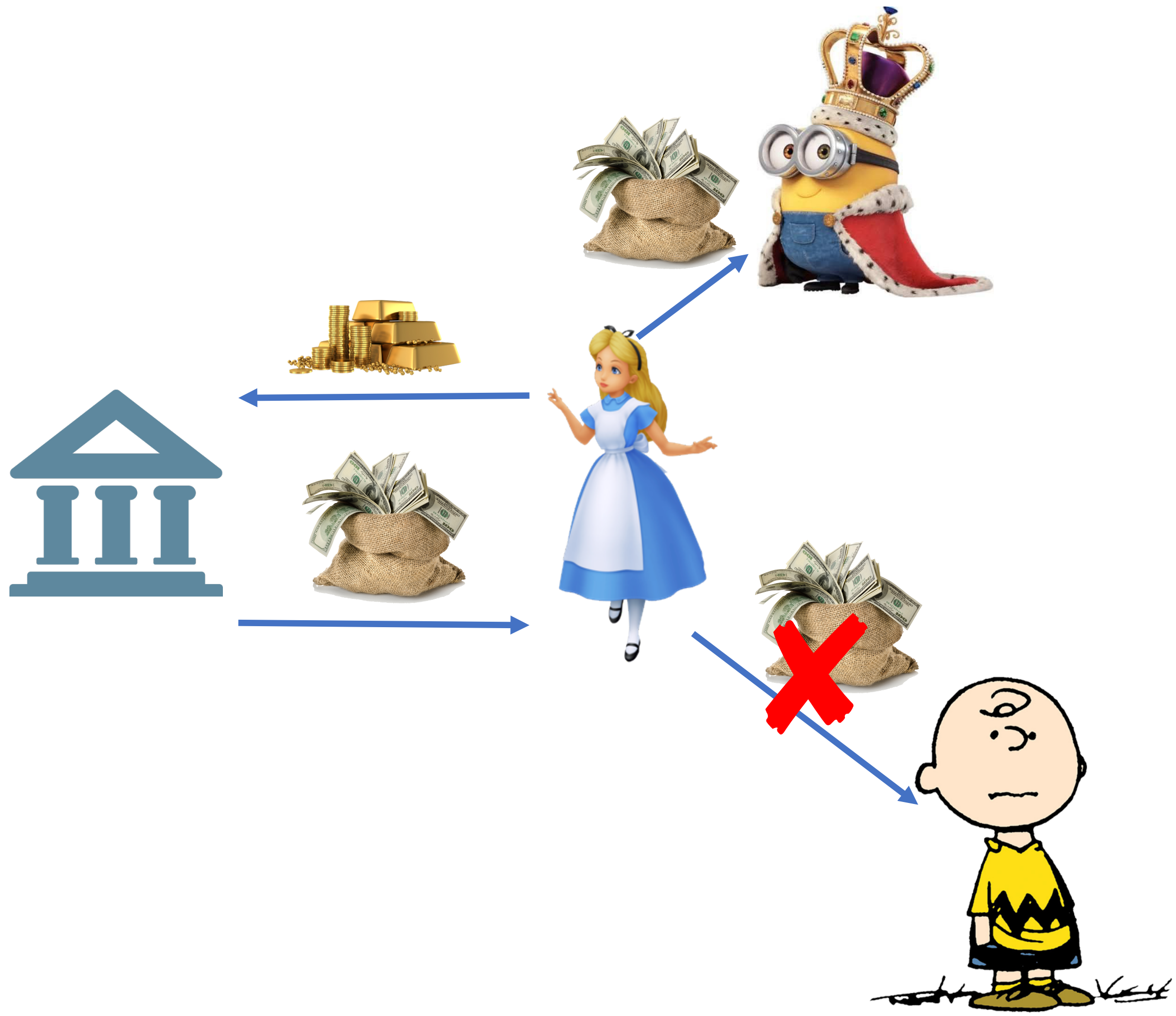
Solution: Public Ledger

Bank transfers $$ to Alice
Alice transfers $$ to Bob



Solution: Public Ledger

Bank transfers $$ to Alice
Alice transfers $$ to Bob



Solution: Public Ledger

Bank maintain ledger?

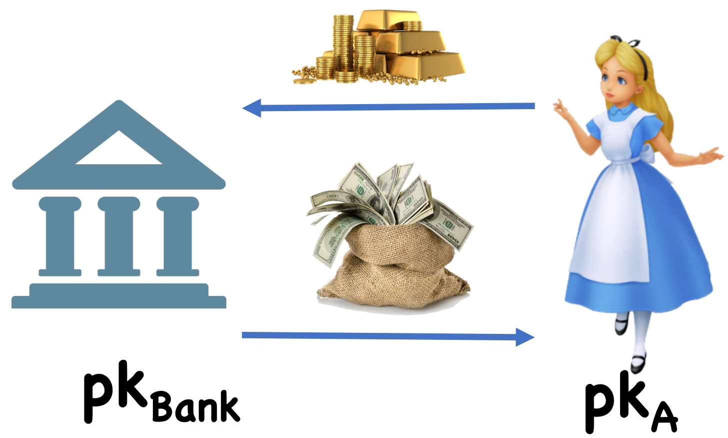
- But then bank must be involved in every transaction
- How does bank prevent malicious Bob from claiming Alice transferred money to him?

Anonymity also lost, since all transactions public

Solution: Use Signatures

$\mathbf{pk_{Bank}}$ transfers \$\$ to $\mathbf{pk_A}$, σ_1

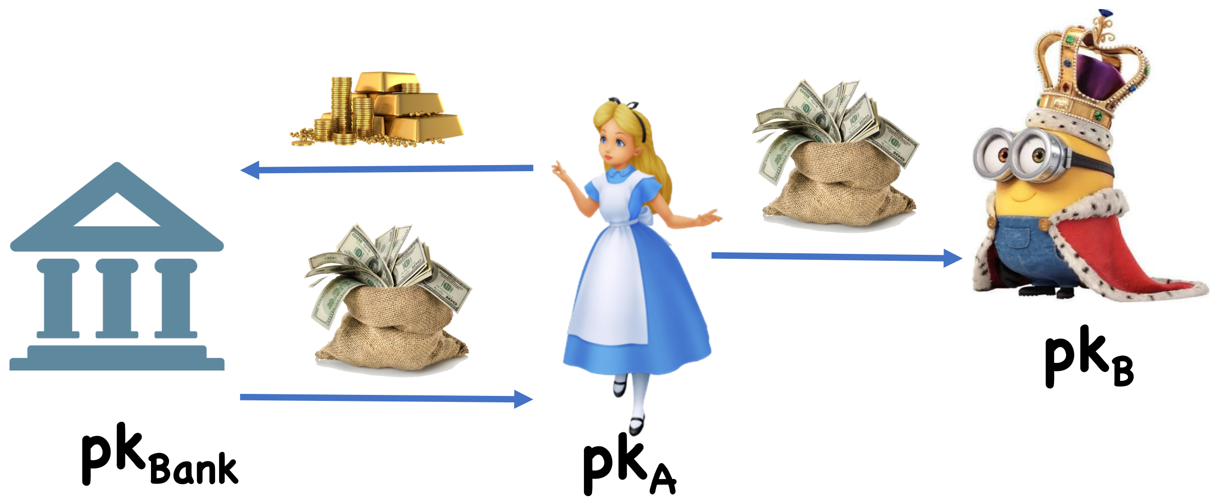
σ_1 = **Sign**($\mathbf{sk_{Bank}}$, "$\mathbf{pk_{Bank}}$ transfers \$\$ to $\mathbf{pk_A}$")



Solution: Use Signatures

$\mathbf{pk_{Bank}}$ transfers \$\$ to $\mathbf{pk_A}$, σ_1
$\mathbf{pk_A}$ transfers \$\$ to $\mathbf{pk_B}$, σ_2

$\sigma_2 = \mathbf{Sign}(\mathbf{sk_A},$ "$\mathbf{pk_A}$ transfers \$\$ to $\mathbf{pk_B}$")



Solution: Use Signatures

By using public key as identity, transactions not immediately traced to individual

- Though can still trace sequences of transactions

By signing, prevents Bob from claiming Alice gave him money when she didn't

Decentralized Currency

Removing the bank is hard:

- How is ledger maintained?
- How to prevent ledger from being tampered with
- Who mints new currency?
- How do we limit supply?

Proofs of Work

Prove that some amount of computation has been performed

Ex:

- Let H be a hash function (modeled as a RO)
- An input x such that $H(x) = 0^t{*}{*}{*}{*}{*}$ is a "proof" that you computed approximately 2^t hashes

Proofs of Work and Cryptocurrency

Idea: currency is a proof of work

- Limits supply of money, so keeps inflation in check
- Now, anyone can mint new money

Blockchain

Immutable public ledger

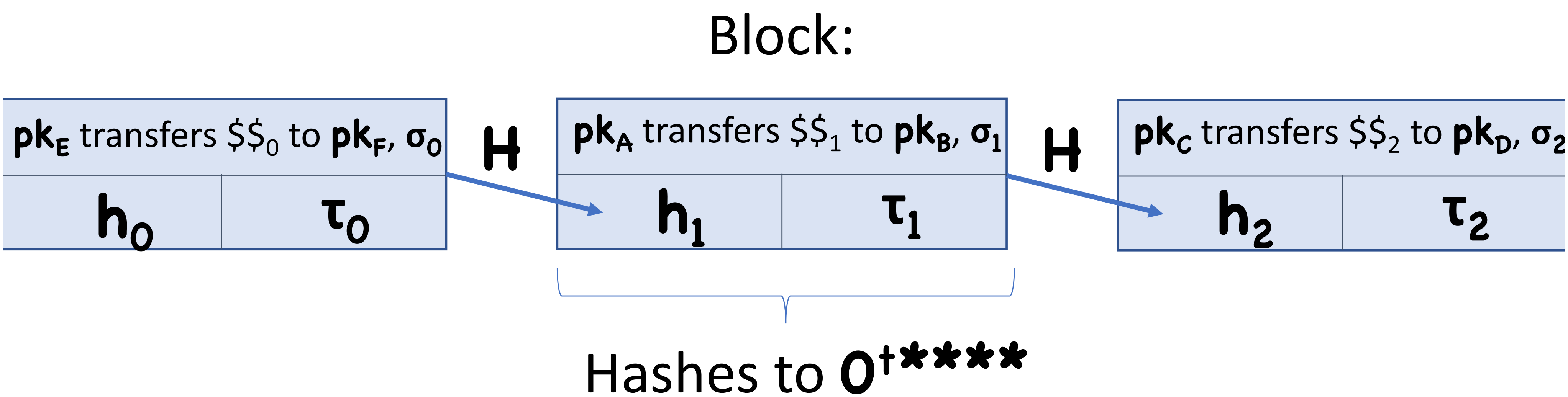
Block:

pk_A transfers \$\$ to pk_B, σ_1	
h_1	τ_1

Hashes to 0^t****

Blockchain

Immutable public ledger



Blockchain

By making each block a proof of work, hard to modify blockchain

So proofs of work used to:

- Mint new money
- Add transactions to blockchain

Why would anyone go through the effort of adding transactions to the blockchain?

Blockchain

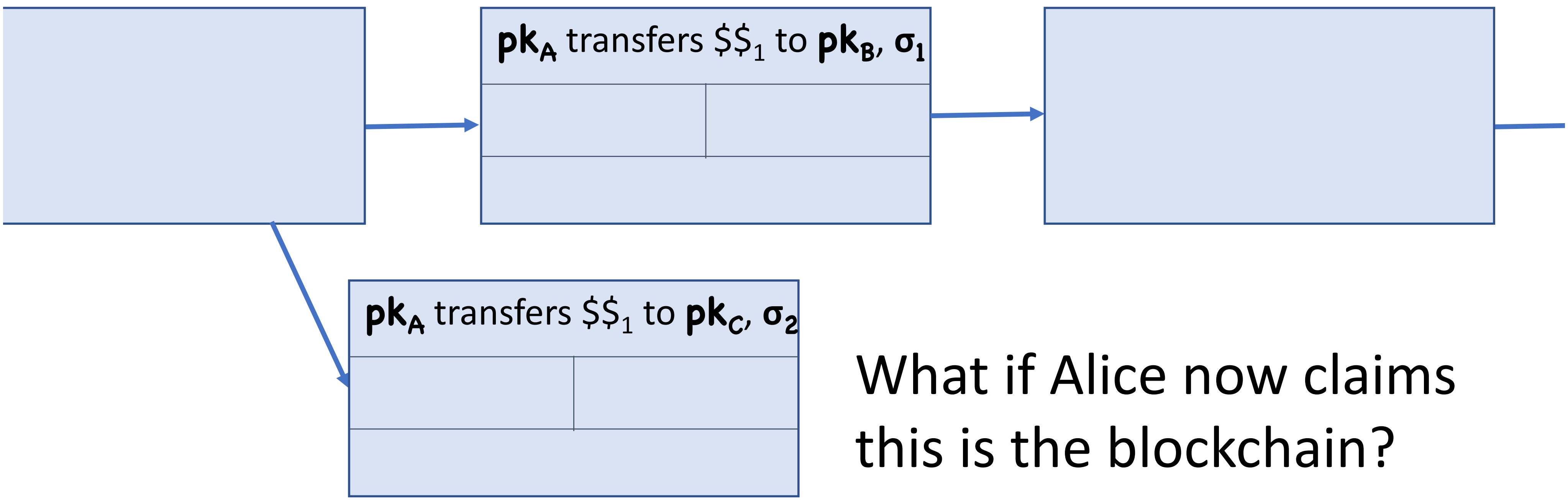
Idea: combine minting and adding blocks

Block:

$\mathbf{pk}_A$ transfers $\$\$_1$ to $\mathbf{pk}_B$, σ_1	
$\mathbf{h}_1$	τ_1
$\mathbf{pk}_M$ mined $\$\$_M$	

Hashes to $\mathbf{0}^{t}$****

Double Spending



What if Alice now claims this is the blockchain?

Double Spending

To prevent double spending, everyone always uses longest chain as the blockchain

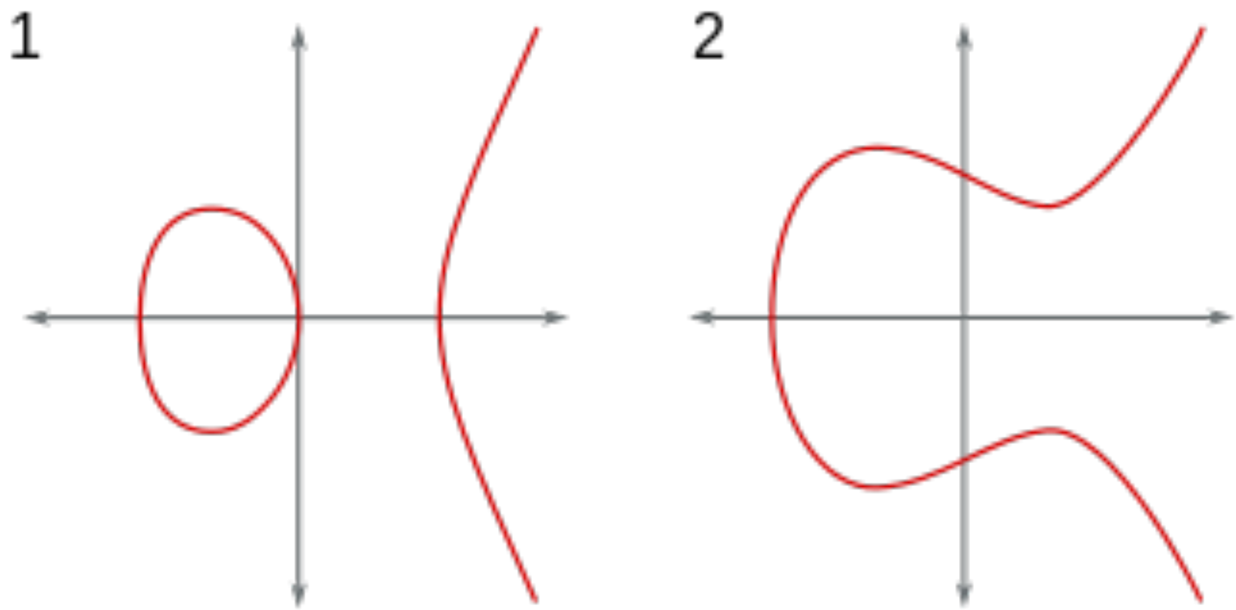
If Alice tries to double spend, she will need to create a separate chain that is as long as the main chain

- As long as she has <<50% of computing power of mining power, will not be possible

Beyond COS 433

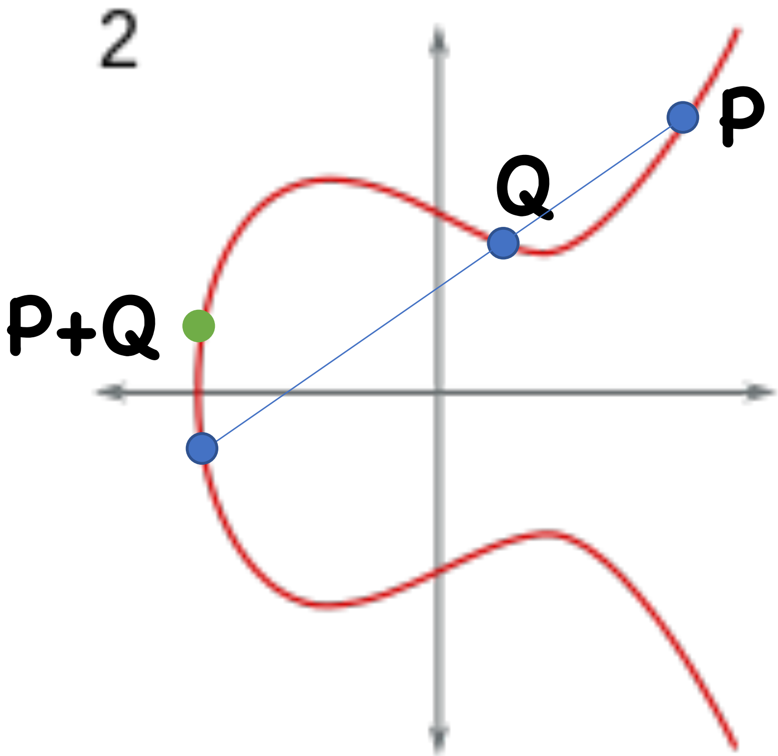
Elliptic Curves

$$y^2 = a\,x^3 + b\,x^2 + c\,x + d$$



$y^2 = x^3 - x$ $\quad$ $y^2 = x^3 - x + 1$

Group Law on ECs



ECs for Crypto

Consider EC over finite field

Set of solutions form a group

Dlog in group appears hard
- Given $aP = (P+P+\ldots+P)$, find a
- Can use in crypto applications

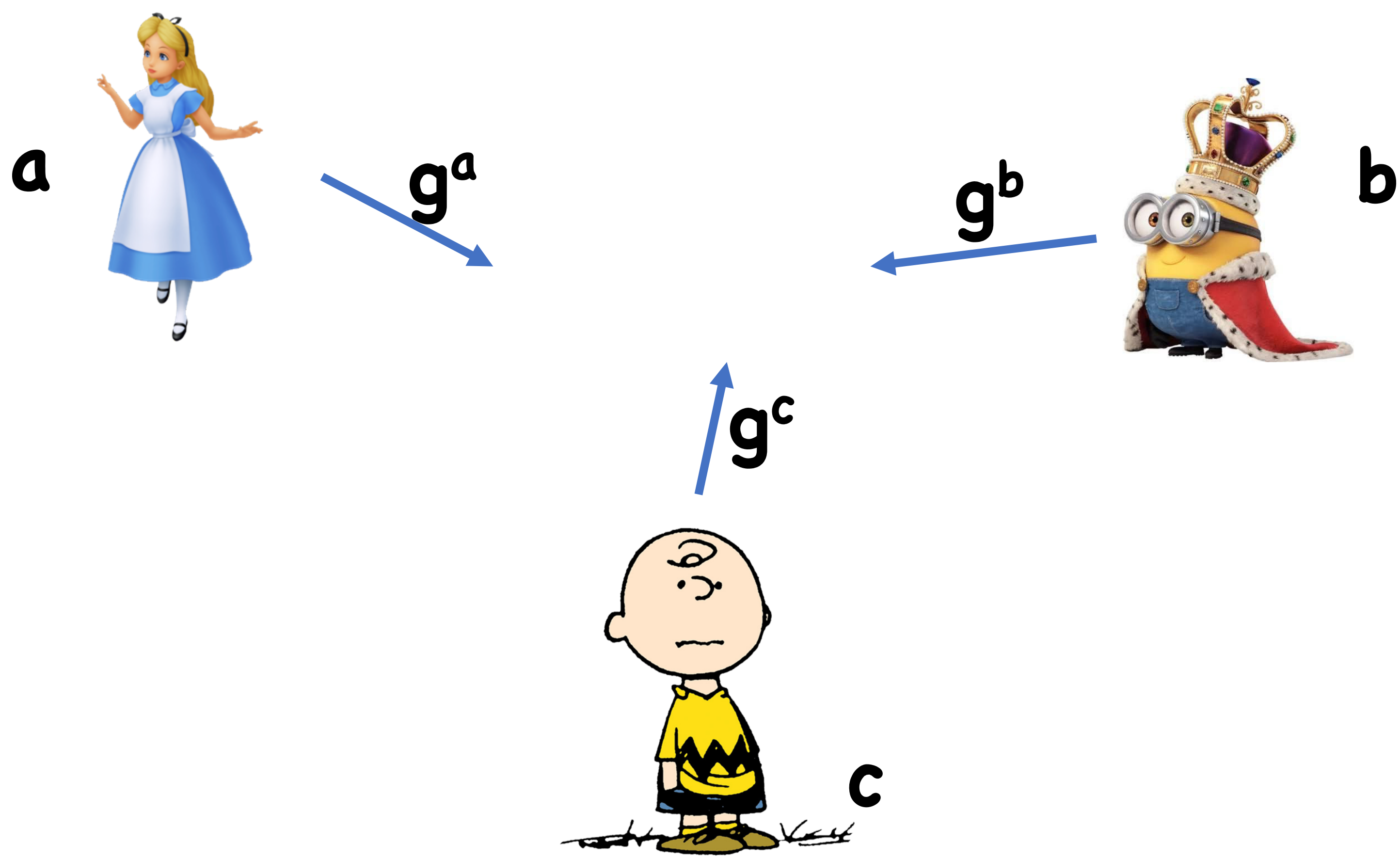
Bilinear Maps

On some Elliptic curves, additional useful structure

Map $\mathbf{e: G \times G \rightarrow G_2}$

- $\mathbf{e(g^a, g^b) = e(g,g)^{ab}}$

3-party Key Exchange



Shared key = $e(g,g)^{abc}$

Bilinear Maps

Extremely powerful tool, many applications beyond those in COS 433

- 3 party *non-interactive* key exchange
- Identity-based encryption
- Broadcast encryption

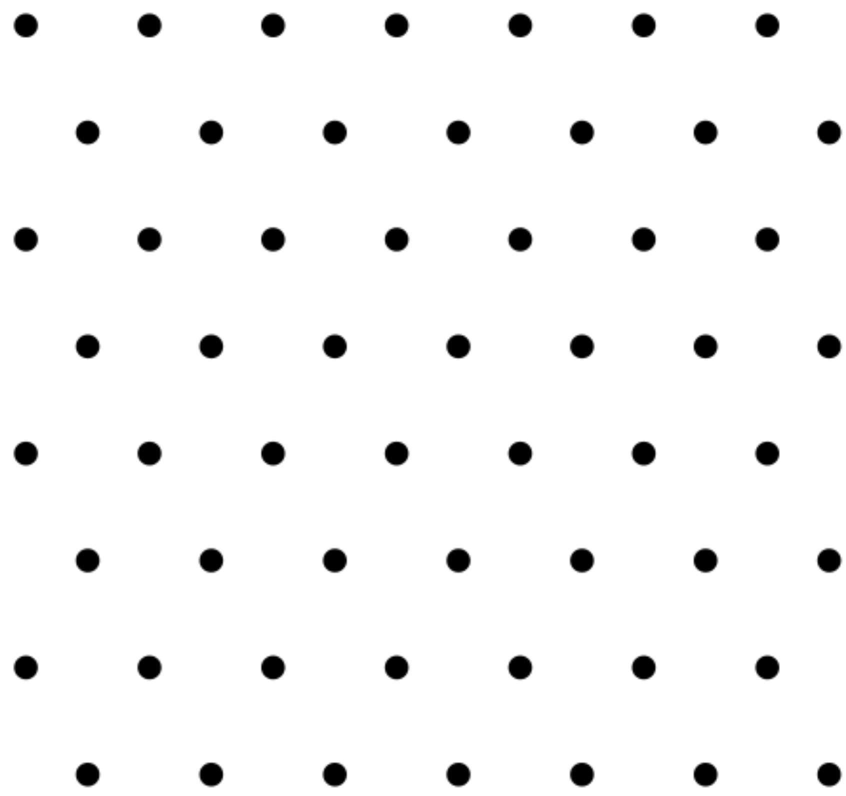
Multilinear Maps

Map $\mathbf{e{:}G^n \rightarrow G_2}$
- $\mathbf{e(g^a, g^b, \ldots) = e(g,g,\ldots)^{ab\ldots}}$

Many more applications that bilinear maps:
- n+1 party non-interactive key exchange
- Obfuscation
- …

Unfortunately, don't know how to construct from elliptic curves
- Recently, constructions based on other math

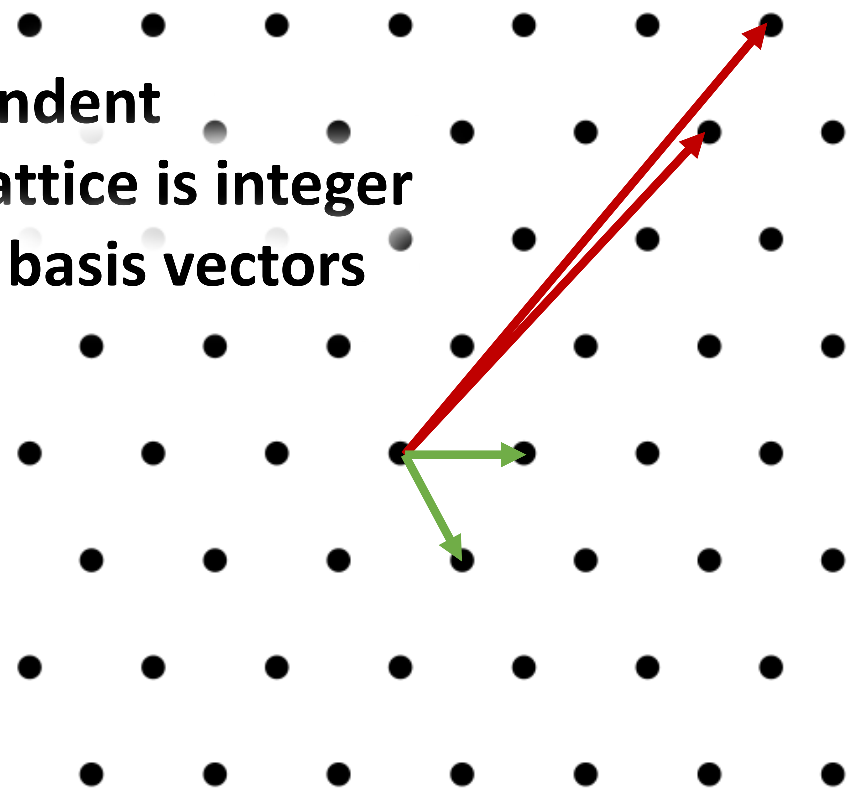
Lattices



Lattices

Basis:

- **Linearly independent**
- **Every point in lattice is integer combination of basis vectors**



Lattices

Hard problems in lattices:

- Given a basis, find the shortest vector in the lattice
- Given a basis an a point not in the lattice, find the closest lattice point

Can base much crypto on approximation versions of these problems

- Basically everything we've seen in COS433, then some

Fully Homomorphic Encryption

In homework, you saw additively/multiplicatively homomorphic encryption:

$$\mathbf{Enc(pk, x) + Enc(pk, y) = Enc(pk, x+y)}$$

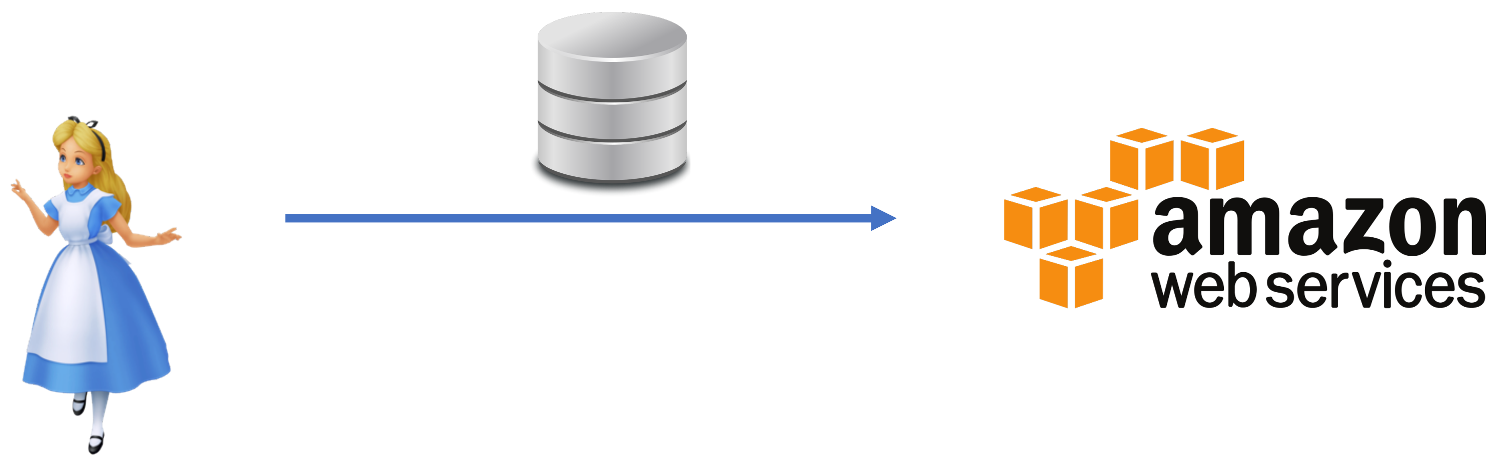
OR

$$\mathbf{Enc(pk, x) \times Enc(pk, y) = Enc(pk, x \times y)}$$

What if you could do both simultaneously?

- Arbitrary computations on encrypted data

Delegation

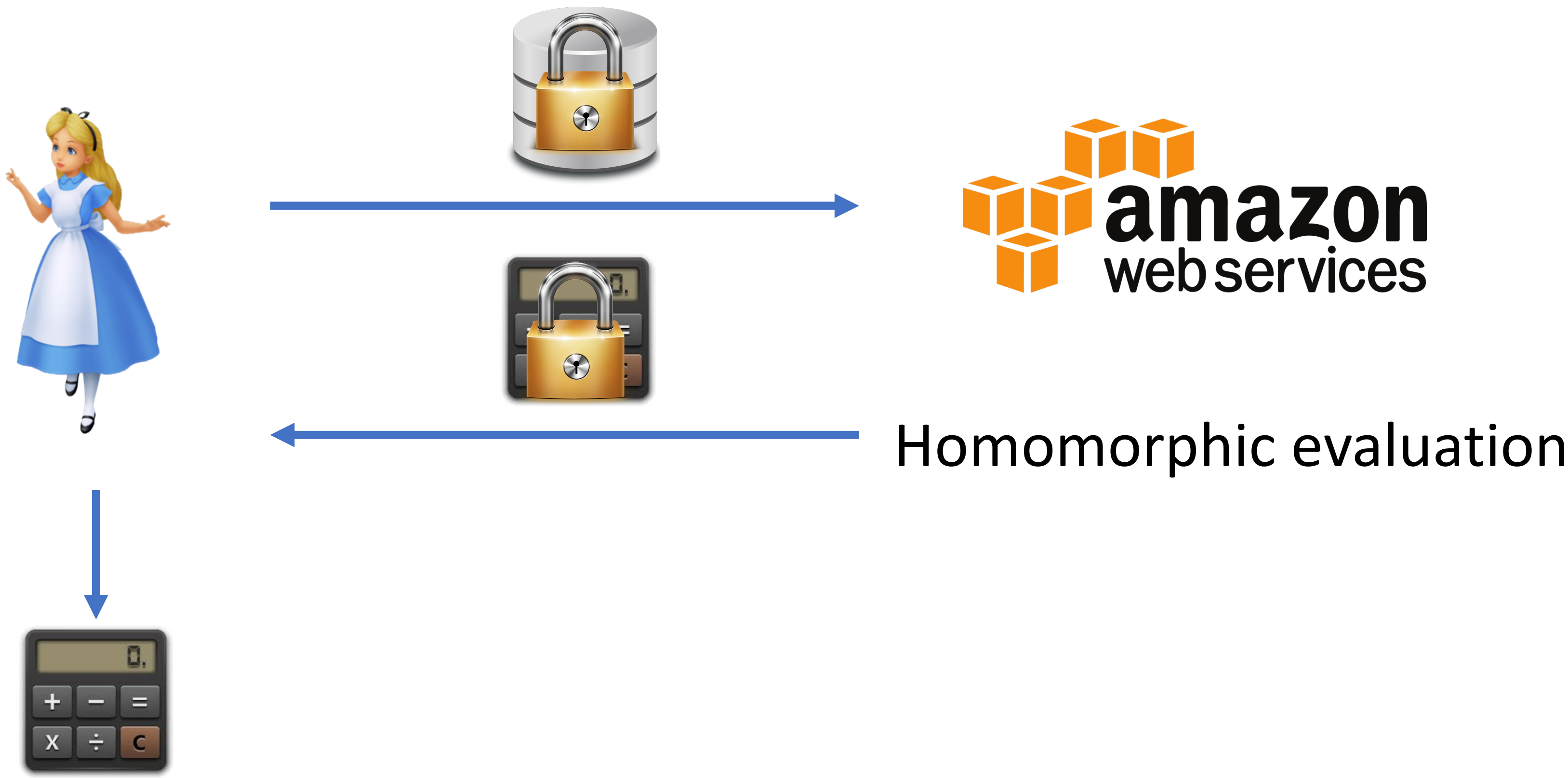


Doesn't want Amazon to learn sensitive data

Delegation

Now, Alice wants Amazon to run expensive computation on data

Delegation



Quantum Computing

Computers that take advantage of quantum physics

Turns out, good at solving certain problems

- Dlog in any group ($\mathbb{Z}_p^*$, ECs)
- Factor integers

Also can speed up brute force search:

- Invert OWF in time $2^{n/2}$
- Find collisions in time $2^{n/3}$

Quantum Computing

To protect against quantum attacks, must:

- Must increase key size
    - 256 bits for one-way functions
    - 384 bits for collision resistance
- Must not use DDH/Factoring
    - Lattices instead

Quantum computers still at least a few years away, but coming

Final Exam Details

Slightly longer than homework, but slightly shorter questions

Pick any **48 hour** period during the dates **May 16 – May 21**

- Will send out more comprehensive instructions

Individual, but open notes/slides/internet…

Example exam on course webpage

Reminders

HW 8 Due May 8

Project 3 Due Dean’s date

No more Monday OH – Mark’s OH by appointment