Cryptographic Protocols

What we covered

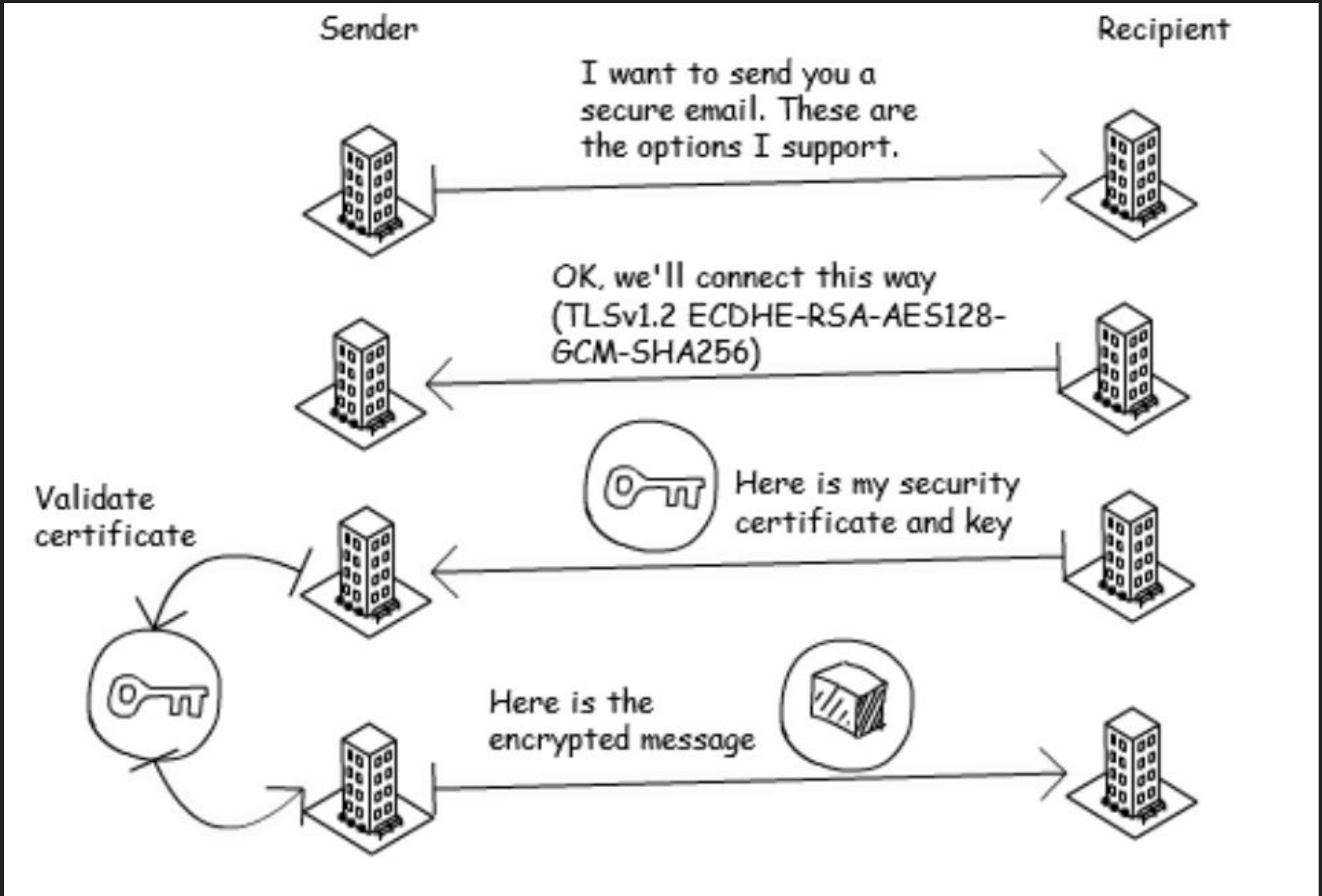
- We can **encrypt** and **decrypt** messages (we looked at RSA)
- We can **sign** and **verify** messages by reversing the order of those operations
  - e.g. Decrypt(priv_key, ciphertext) == Sign(message) and Encrypt(pub_key, message) == Verify(signature)
- We can generate short, fixed-length representations of input data by **hashing**
- We can **exchange keys** securely (we looked at DH)

TLS

- TLS is the cryptography that underpins HTTPS connections
  - Provides **privacy**, **authentication**, and **integrity**
- **Cipher-suites** let us choose which cryptographic primitives we want to use for signatures, encryption, and key distribution
- High-level overview: choose a cipher-suite, exchange keys, verify each other, and then communicate with authenticated encrypted data.

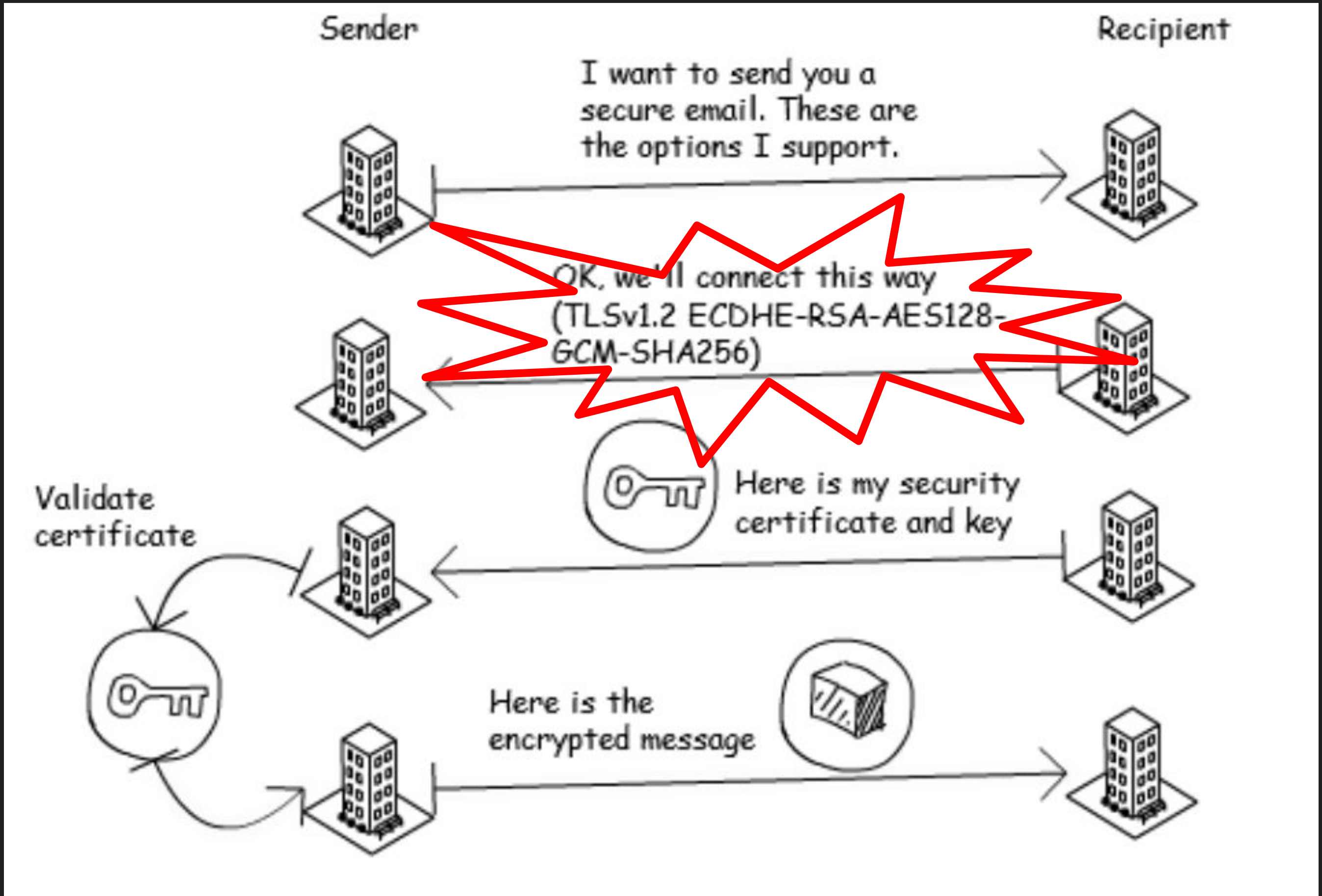
```
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
```

TLS



* This is mostly right, but not exactly

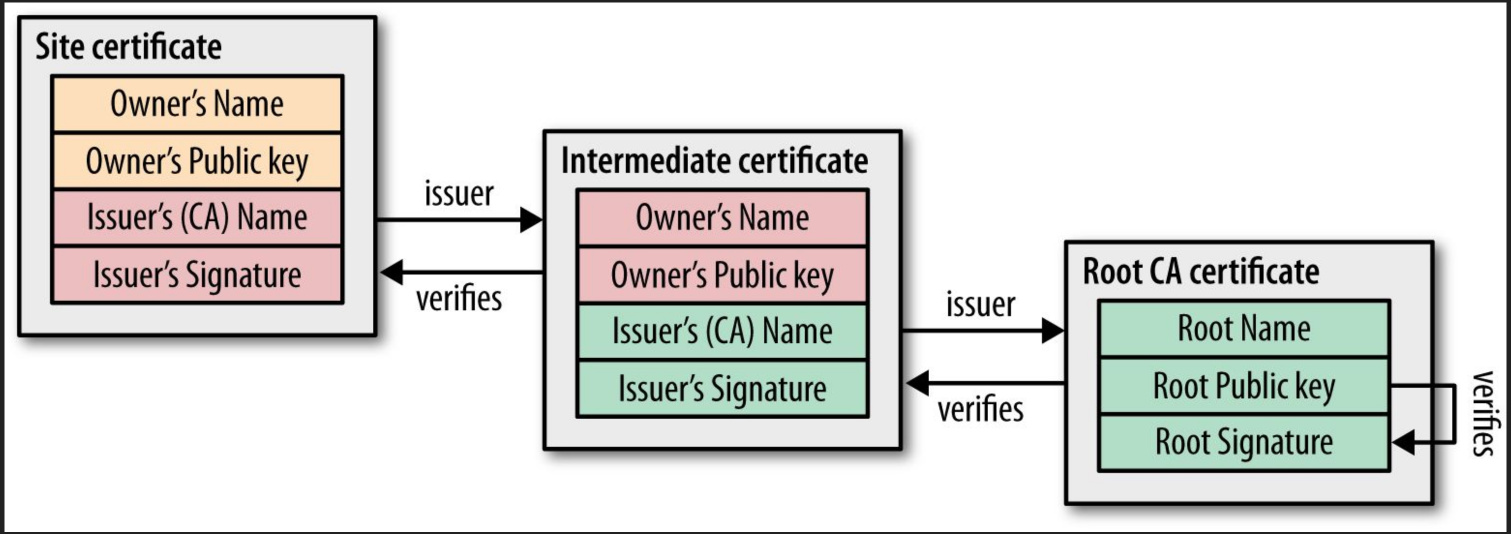
Cipher-suite



* This is mostly right, but not exactly

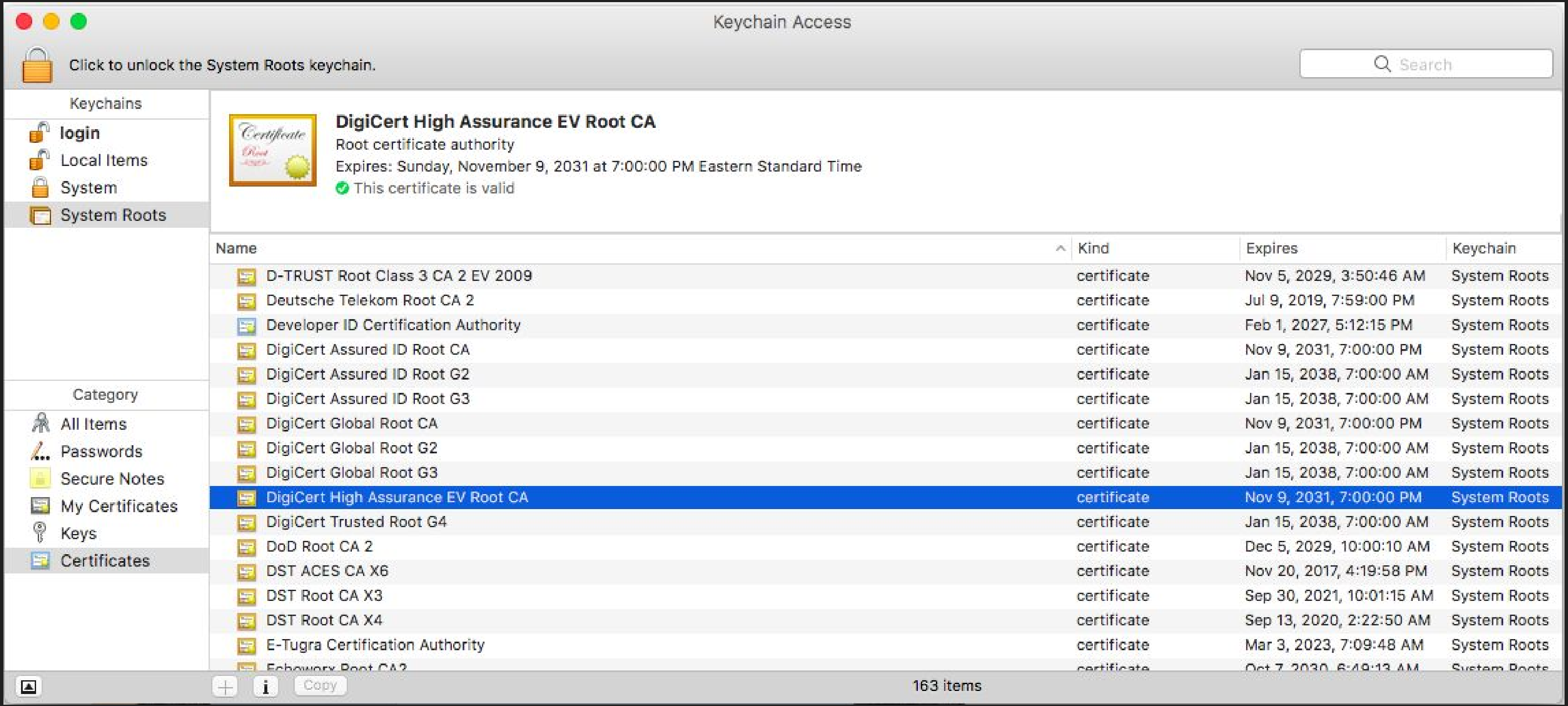
Certificates and Certificate Authorities

- Servers are verified with **certificates**, which are signed pieces of data containing the name of the domain (e.g. google.com) and some company info
- **Certificates** are signed by **certificate authorities** in a **trust chain**



Certificates and Certificate Authorities

- CAs are part of the **Public Key Infrastructure**, which is a response to the key-distribution problem
  - We want to distribute public keys and know that they belong to who they claim they do
- Each computer client saves a list of CA certificates on its location machine
- You can see your own CA certificates on your computer!

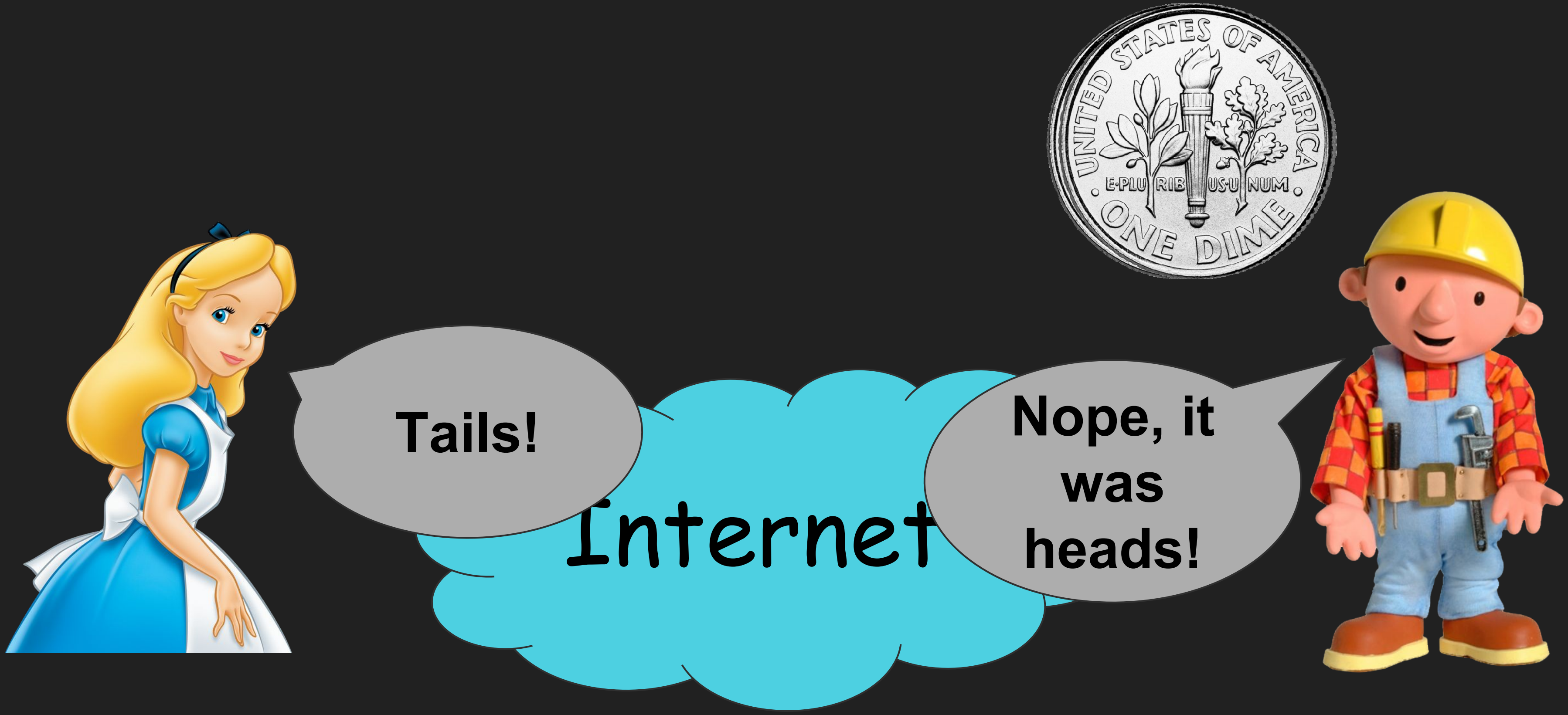

Keychain Access
Click to unlock the System Roots keychain.
Search
Keychains
login
Local Items
System
System Roots
Category
All Items
Passwords
Secure Notes
My Certificates
Keys
Certificates
DigiCert High Assurance EV Root CA
Root certificate authority
Expires: Sunday, November 9, 2031 at 7:00:00 PM Eastern Standard Time
This certificate is valid
Name
Kind
Expires
Keychain
D-TRUST Root Class 3 CA 2 EV 2009 certificate Nov 5, 2029, 3:50:46 AM System Roots
Deutsche Telekom Root CA 2 certificate Jul 9, 2019, 7:59:00 PM System Roots
Developer ID Certification Authority certificate Feb 1, 2027, 5:12:15 PM System Roots
DigiCert Assured ID Root CA certificate Nov 9, 2031, 7:00:00 PM System Roots
DigiCert Assured ID Root G2 certificate Jan 15, 2038, 7:00:00 AM System Roots
DigiCert Assured ID Root G3 certificate Jan 15, 2038, 7:00:00 AM System Roots
DigiCert Global Root CA certificate Nov 9, 2031, 7:00:00 PM System Roots
DigiCert Global Root G2 certificate Jan 15, 2038, 7:00:00 AM System Roots
DigiCert Global Root G3 certificate Jan 15, 2038, 7:00:00 AM System Roots
DigiCert High Assurance EV Root CA certificate Nov 9, 2031, 7:00:00 PM System Roots
DigiCert Trusted Root G4 certificate Jan 15, 2038, 7:00:00 AM System Roots
DoD Root CA 2 certificate Dec 5, 2029, 10:00:10 AM System Roots
DST ACES CA X6 certificate Nov 20, 2017, 4:19:58 PM System Roots
DST Root CA X3 certificate Sep 30, 2021, 10:01:15 AM System Roots
DST Root CA X4 certificate Sep 13, 2020, 2:22:50 AM System Roots
E-Tugra Certification Authority certificate Mar 3, 2023, 7:09:48 AM System Roots
Copy
163 items

Other Protocols

- Coin flipping: should Alice and Bob go to the opera or a soccer match?
- Interactive proofs: can Alice convince Bob something is true?
- Yao's millionaires' problem: does Alice or Bob make more money?
- Socialist millionaire’s problem: do Alice and Bob have the same salary?
- Privacy-preserving computational geometry
    - Does Alice’s point lie inside Bob’s polygon?
    - Do Alice and Bob’s polygons intersect?

Coin Flipping IRL

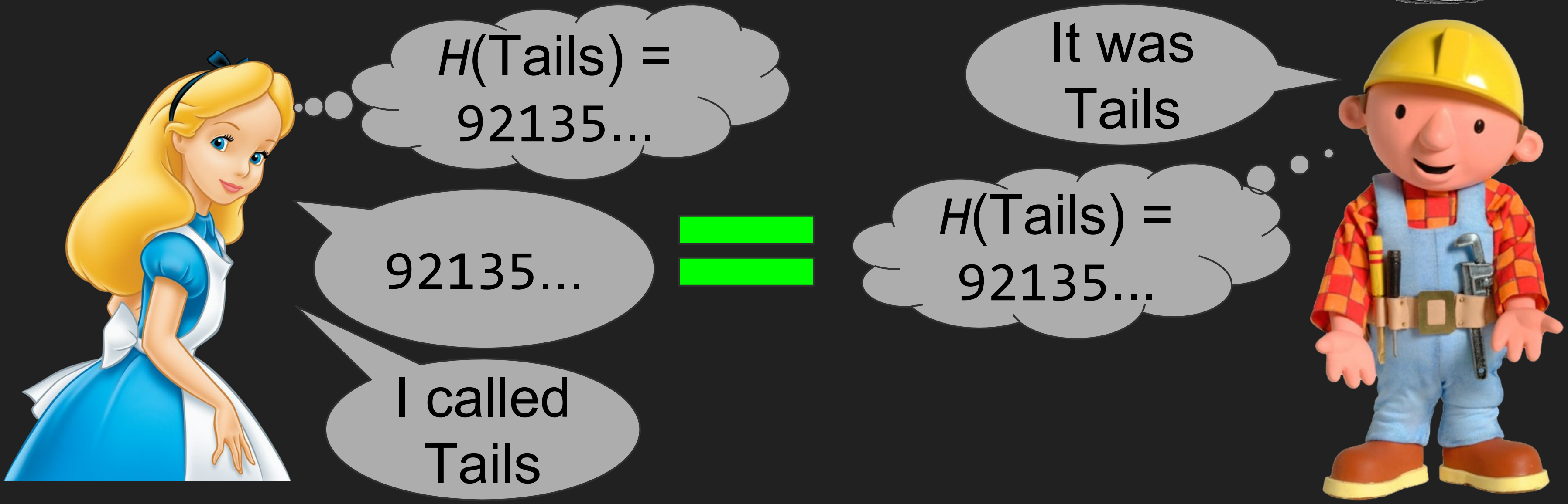
Coin Flipping in Cyberspace



Coin Flipping via Commitment

Have Alice commit to her call without revealing it.

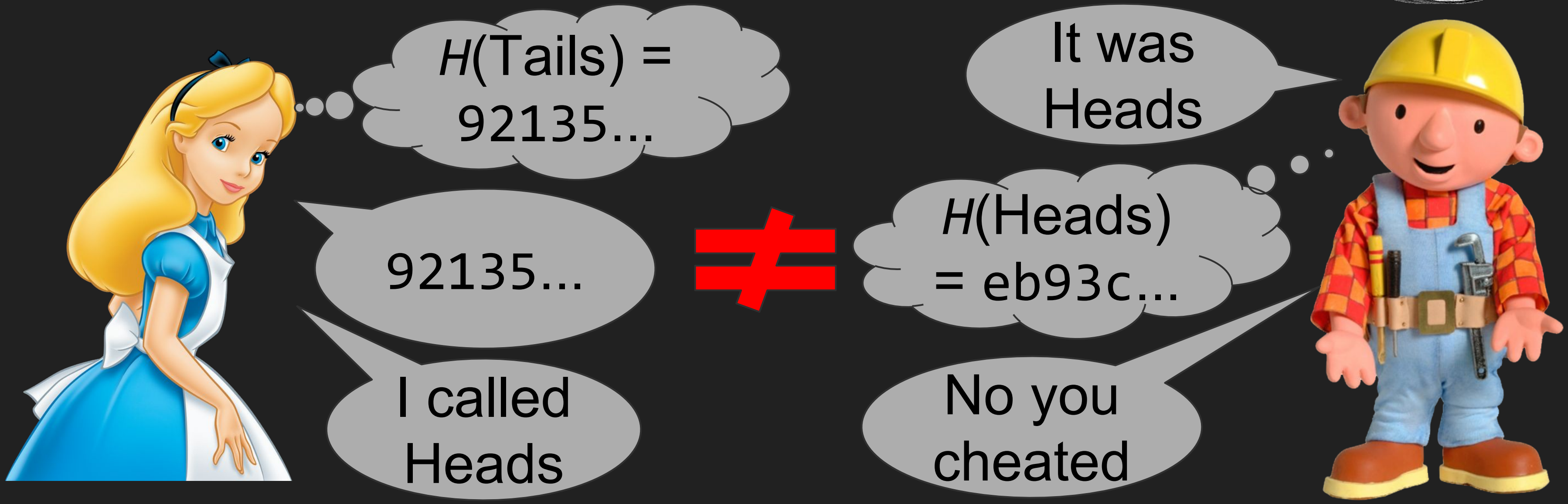
Let *H* be a secure hash function



Coin Flipping via Commitment

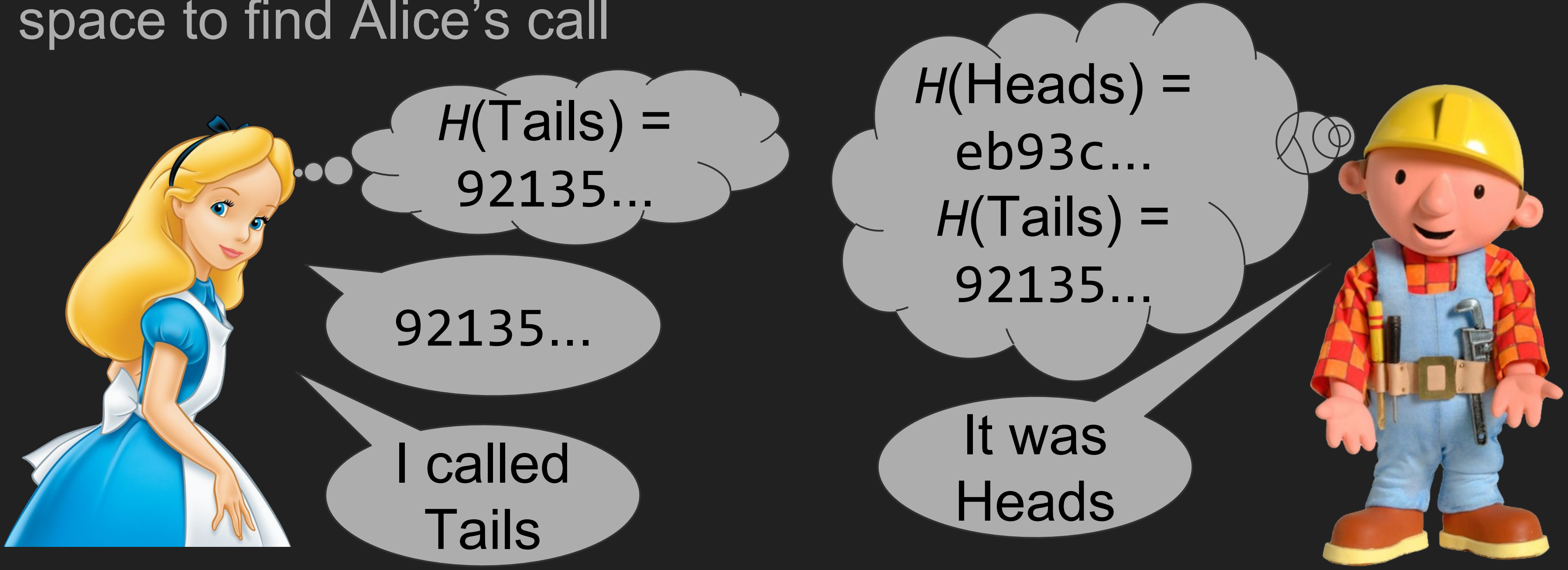
Have Alice commit to her call without revealing it.

Let *H* be a secure hash function



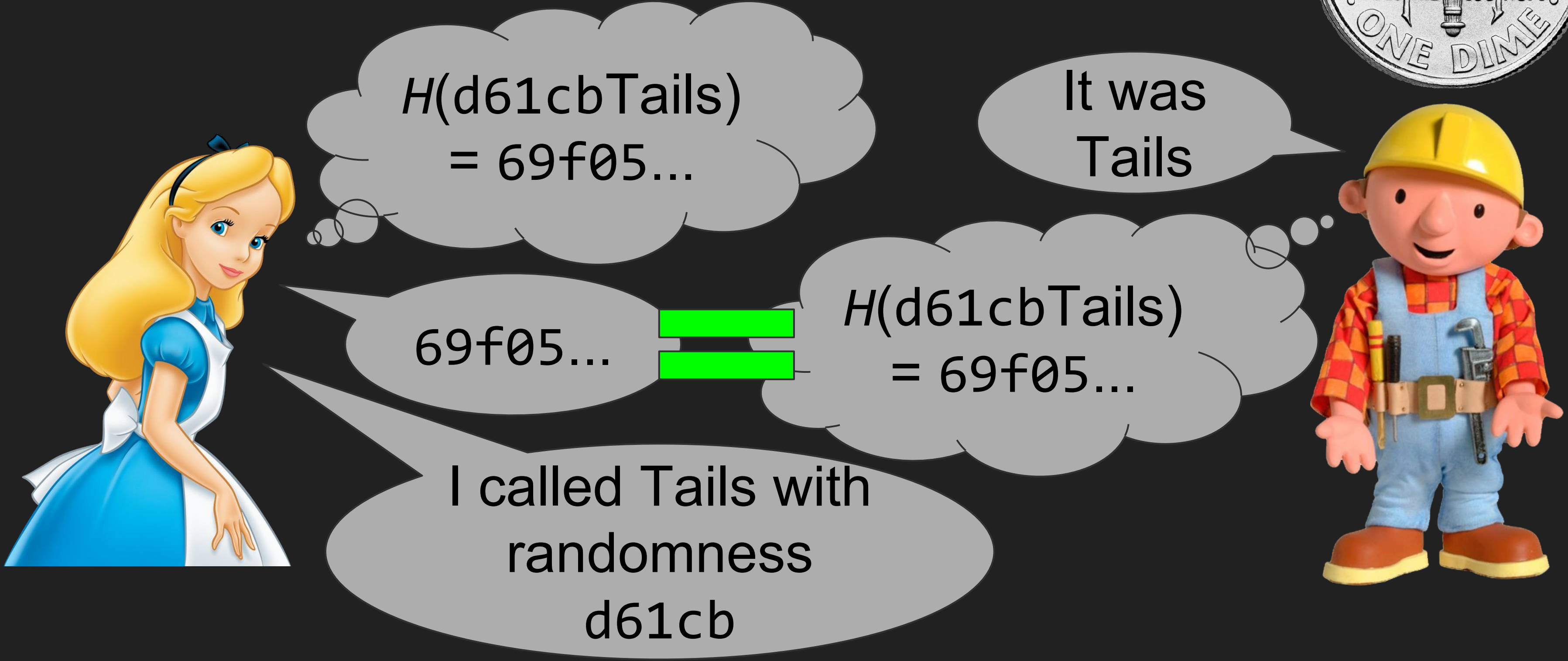
Coin Flipping via Commitment

But Bob can search the message space to find Alice's call



Coin Flipping via Commitment

Artificially increase the message space with *nonces*



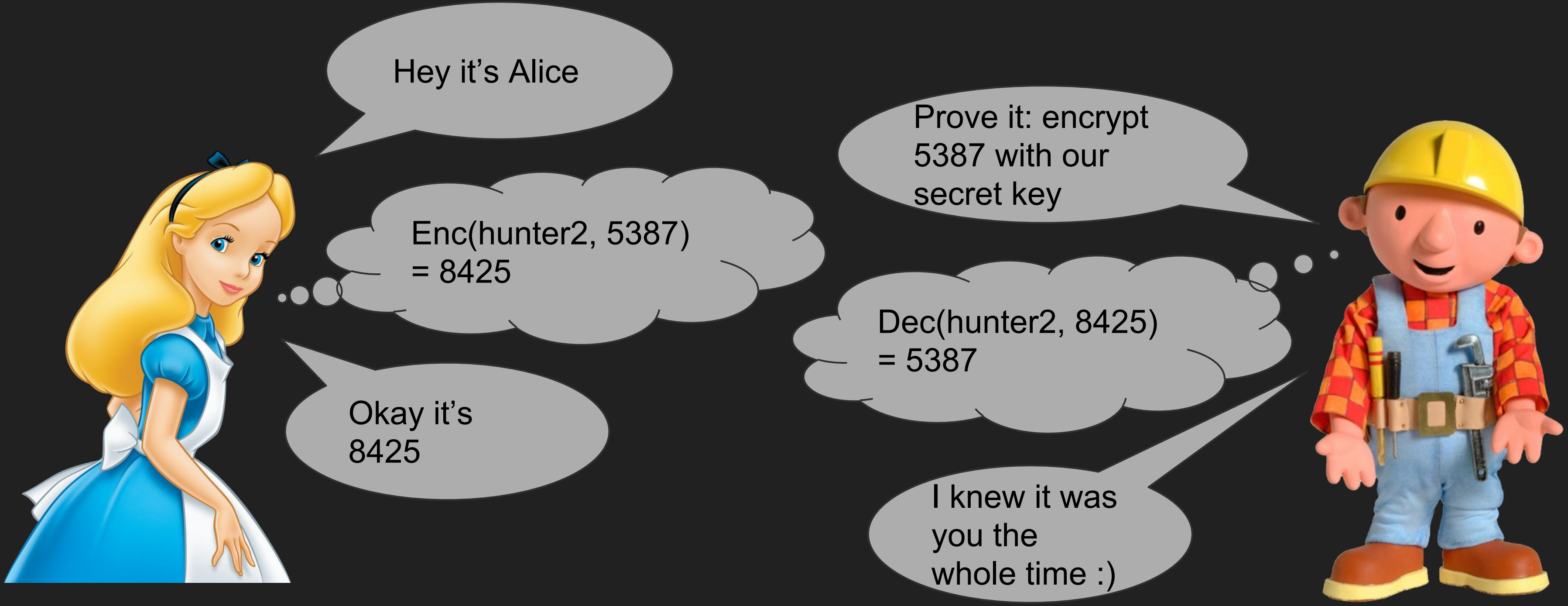
Password Hashing

User	Salt	Hash
Alice	1234	bca288ee
Benji	5678	0ba96592
Carol	9999	406da7cd

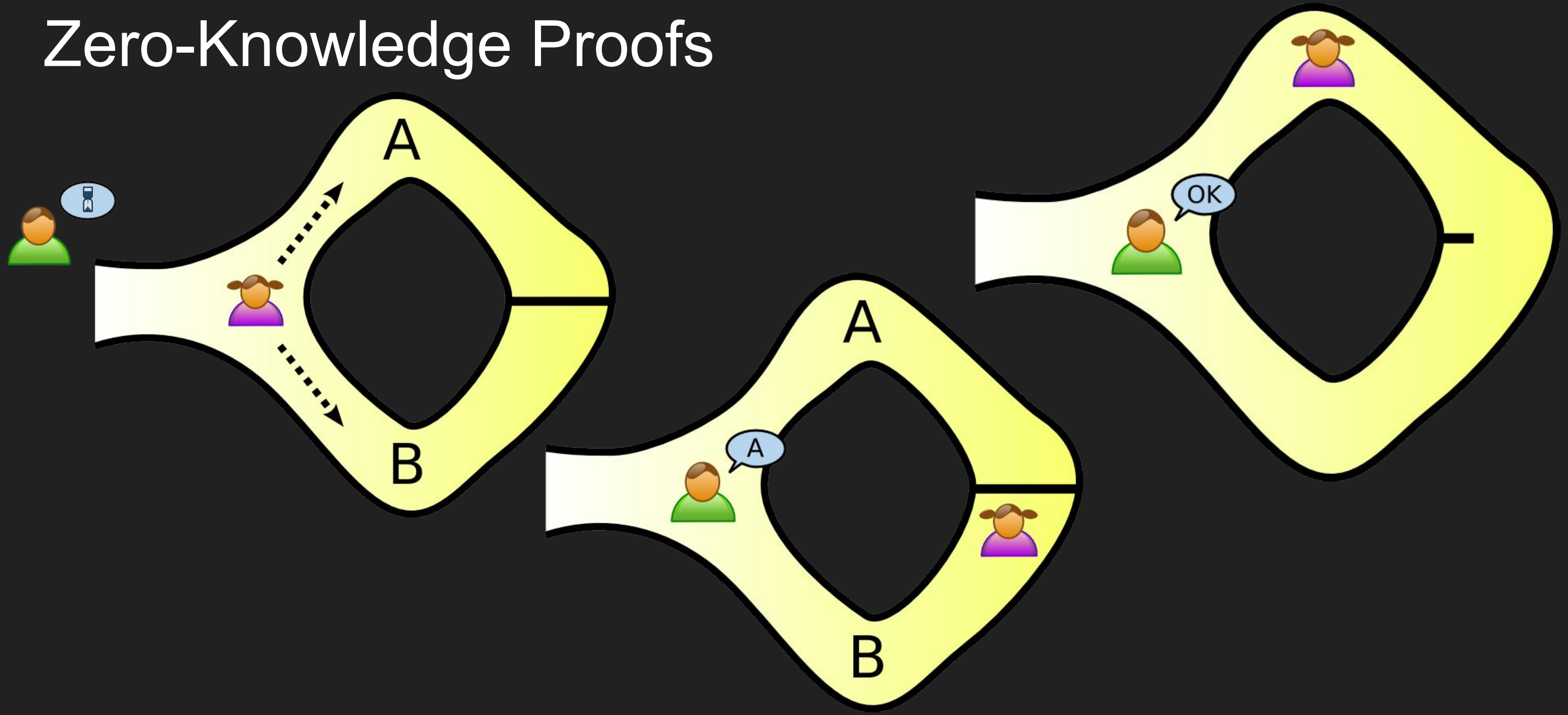
I’m Alice, and my
password is hunter2

H(1234hunter2)
= bca288ee

Challenge-Response Authentication

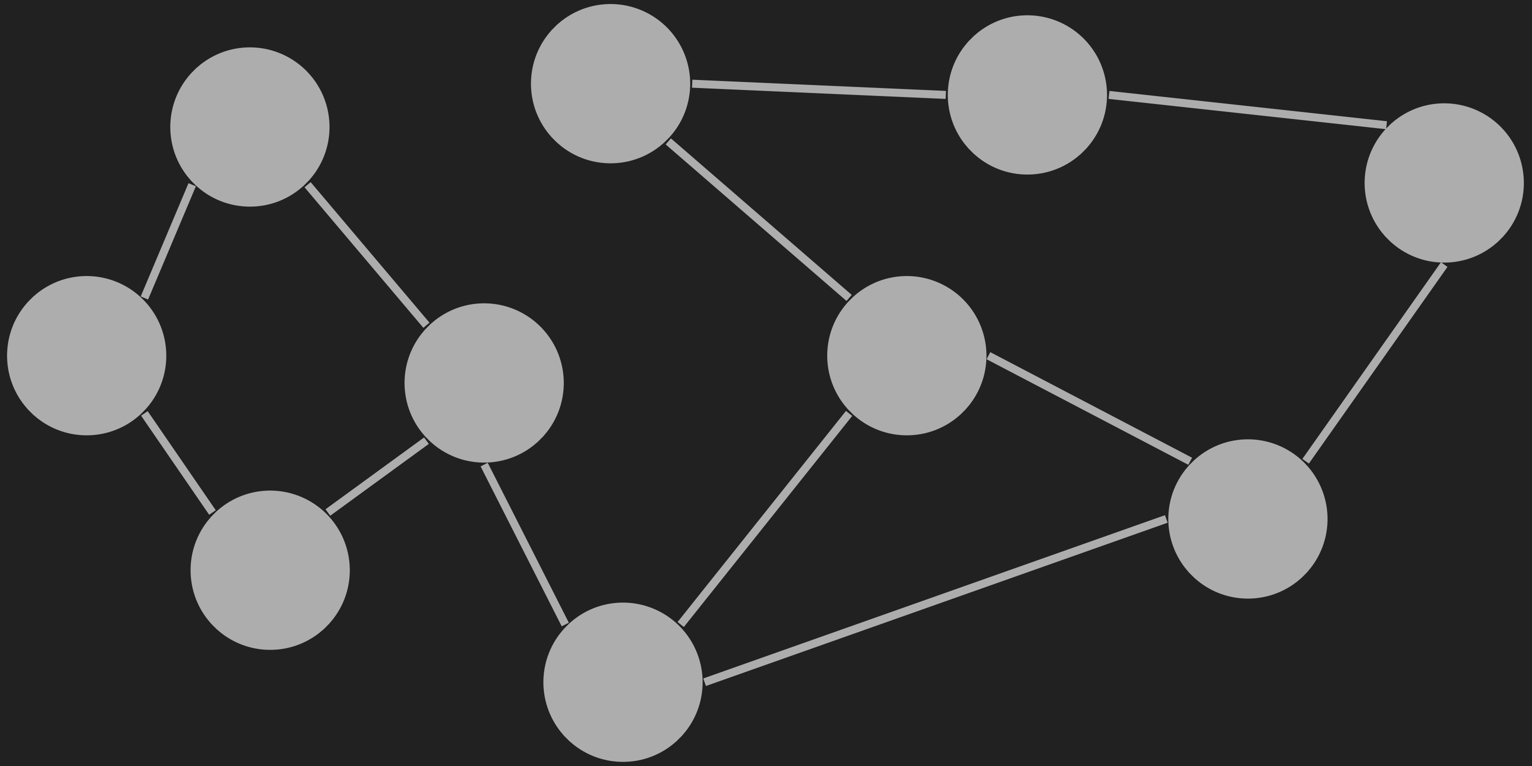


Zero-Knowledge Proofs

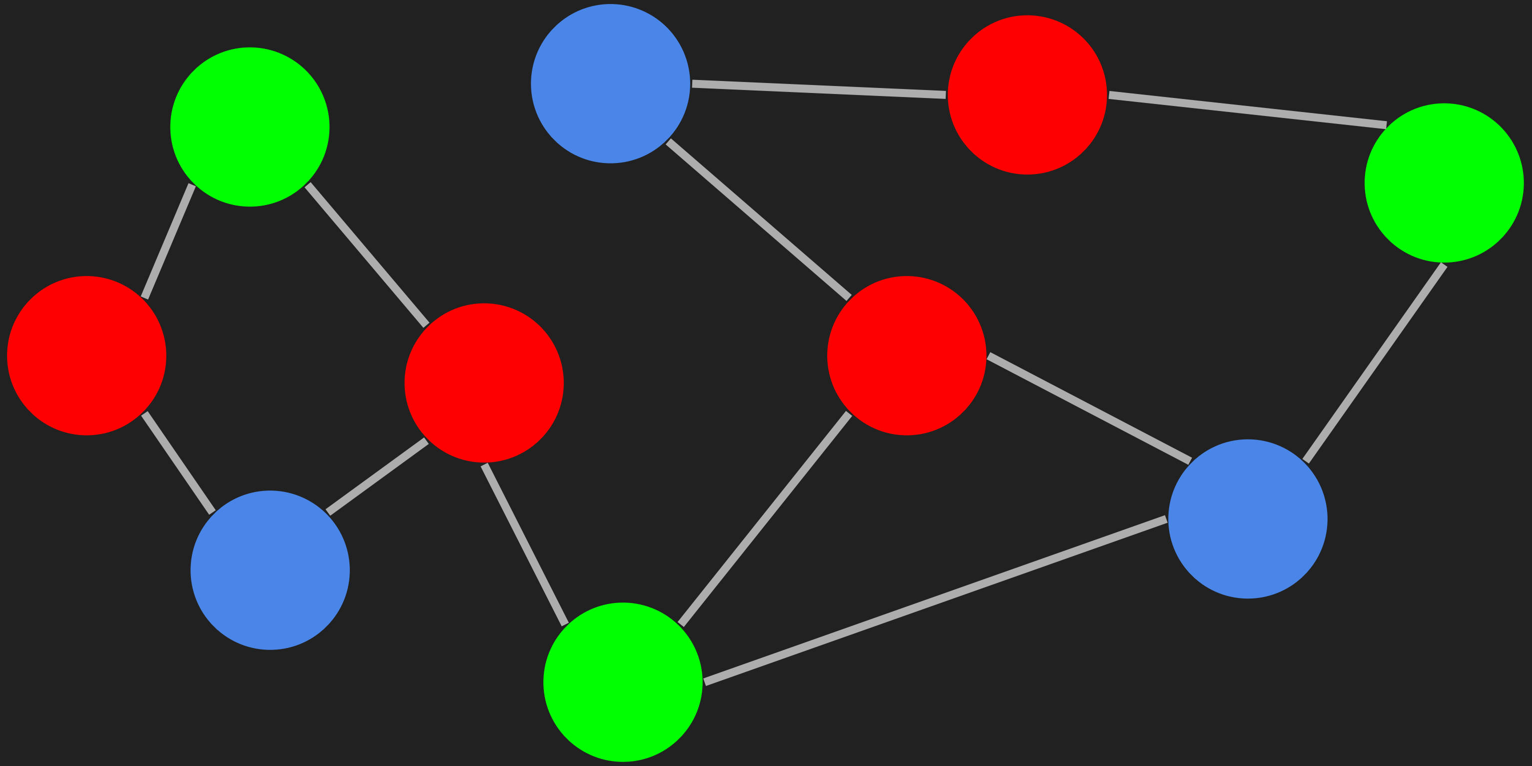


https://en.wikipedia.org/wiki/Zero-knowledge_proof

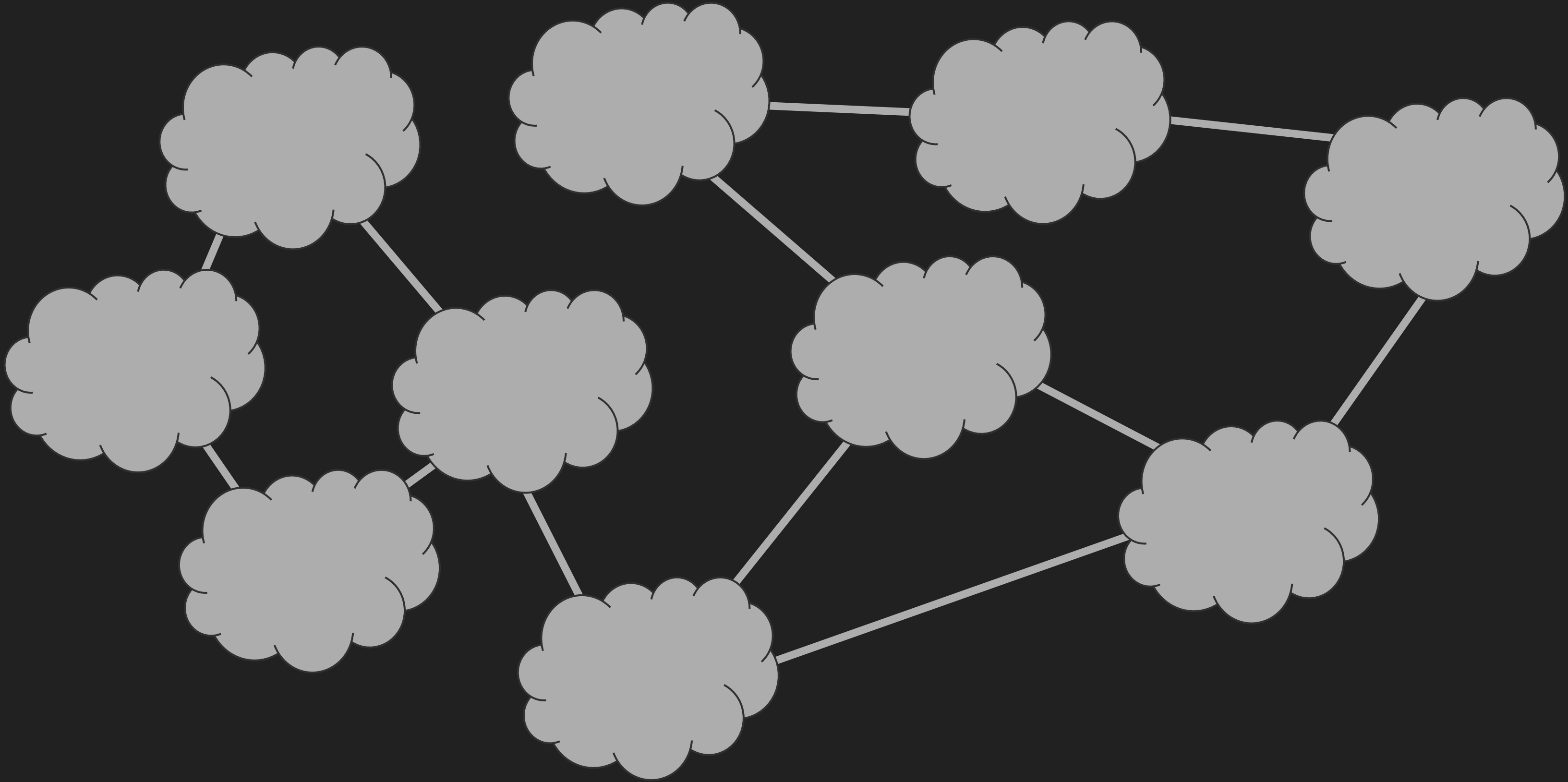
Zero-Knowledge Proofs: 3-Colorability



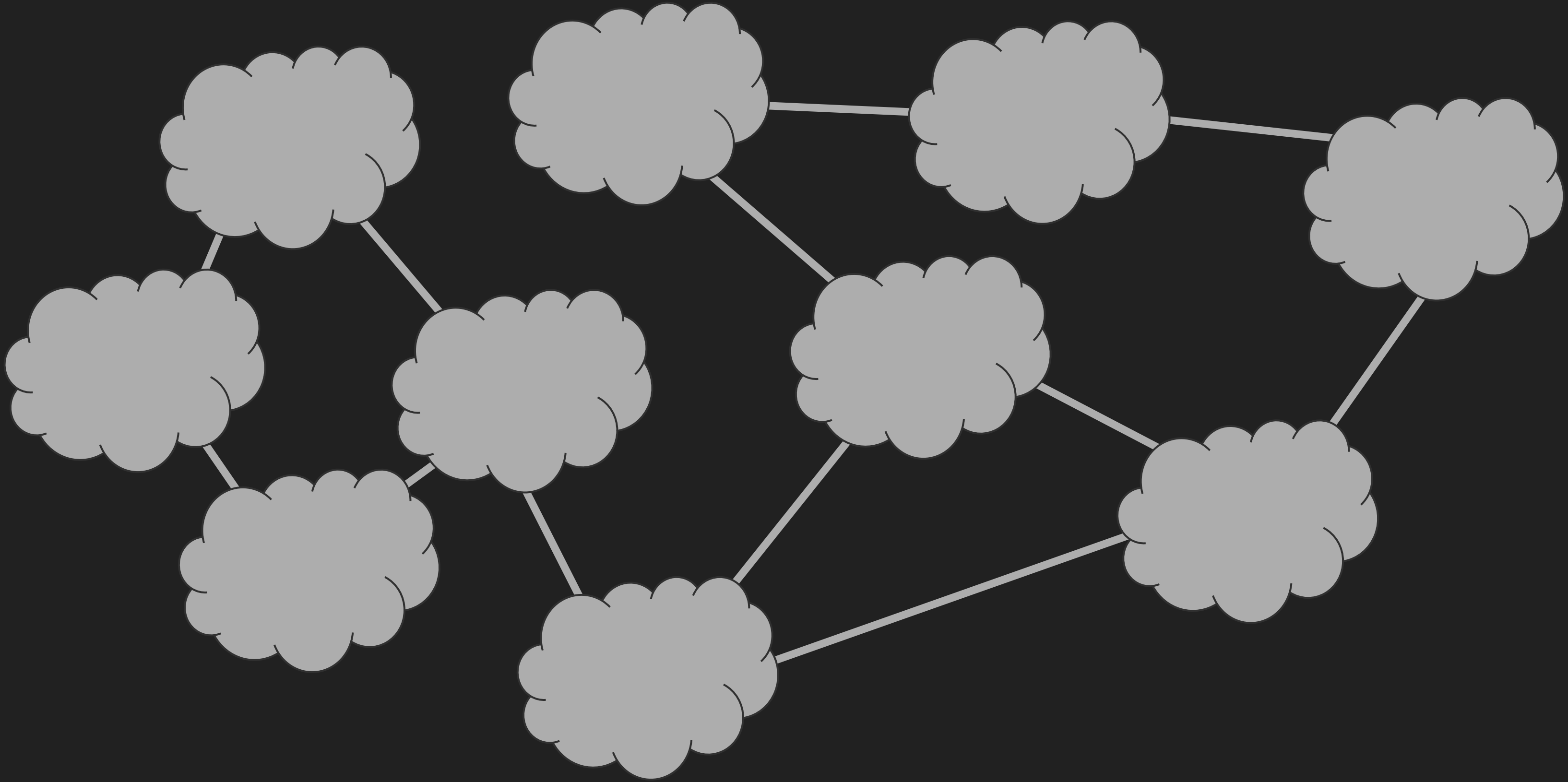
Zero-Knowledge Proofs: 3-Colorability



Zero-Knowledge Proofs: 3-Colorability



Zero-Knowledge Proofs: 3-Colorability



Multi-party Protocols

- Secret sharing: distribute a secret among Alice and her friends
- Electronic voting
- Secure computation
  - Allow distrustful participants to compute a function on their secret inputs
- Decentralized digital currency: let Alice pay her ransomware hackers

Shamir's Secret Sharing

Divide the secret S into n parts, but only need any k of n

Generate a random polynomial with degree k-1!

My secret is "123", and I want n=3, k=2

$y = 123 + 95x + 54x^2 \mod 17729$

(1, 272) (2, 529) (3, 894)

Shamir's Secret Sharing

(1, 272), p= 17729

(3, 894), p= 17729

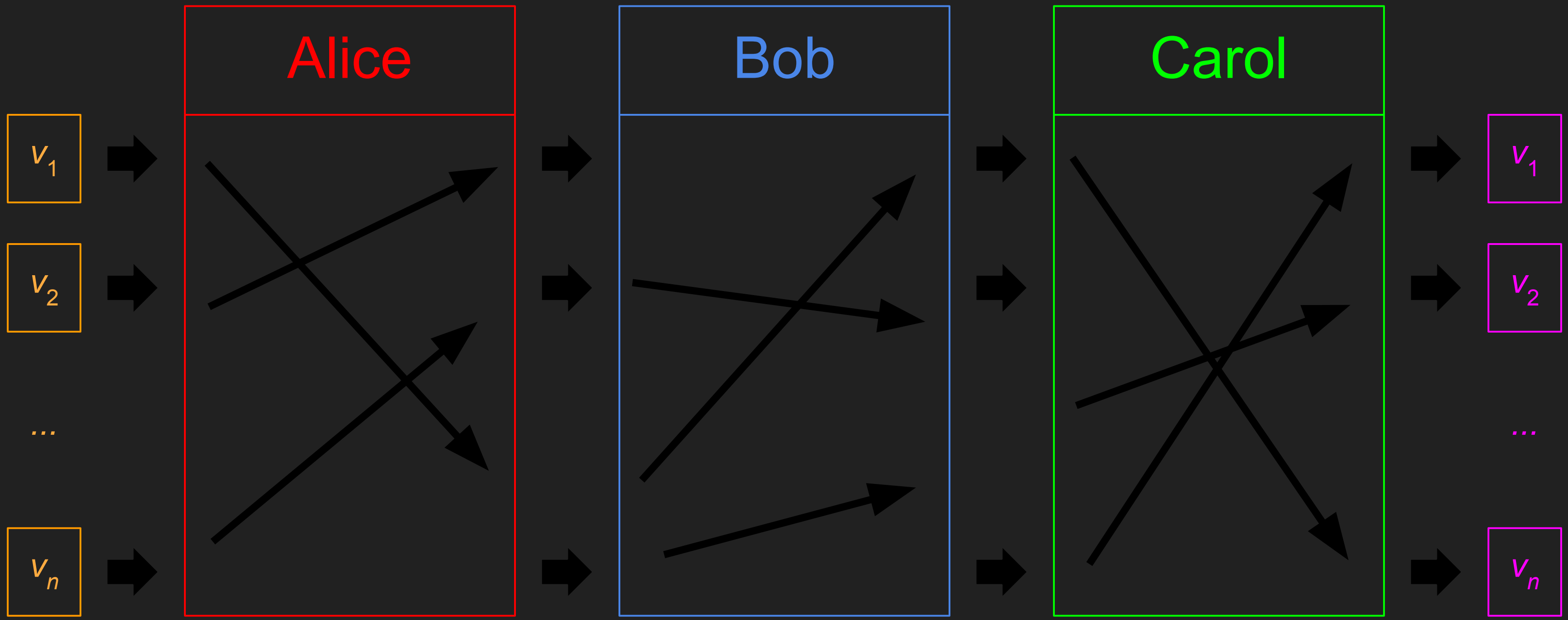
- Any 2 of 3 are sufficient to construct the original polynomial, and thus Alice's secret value!

(2, 529), p= 17729

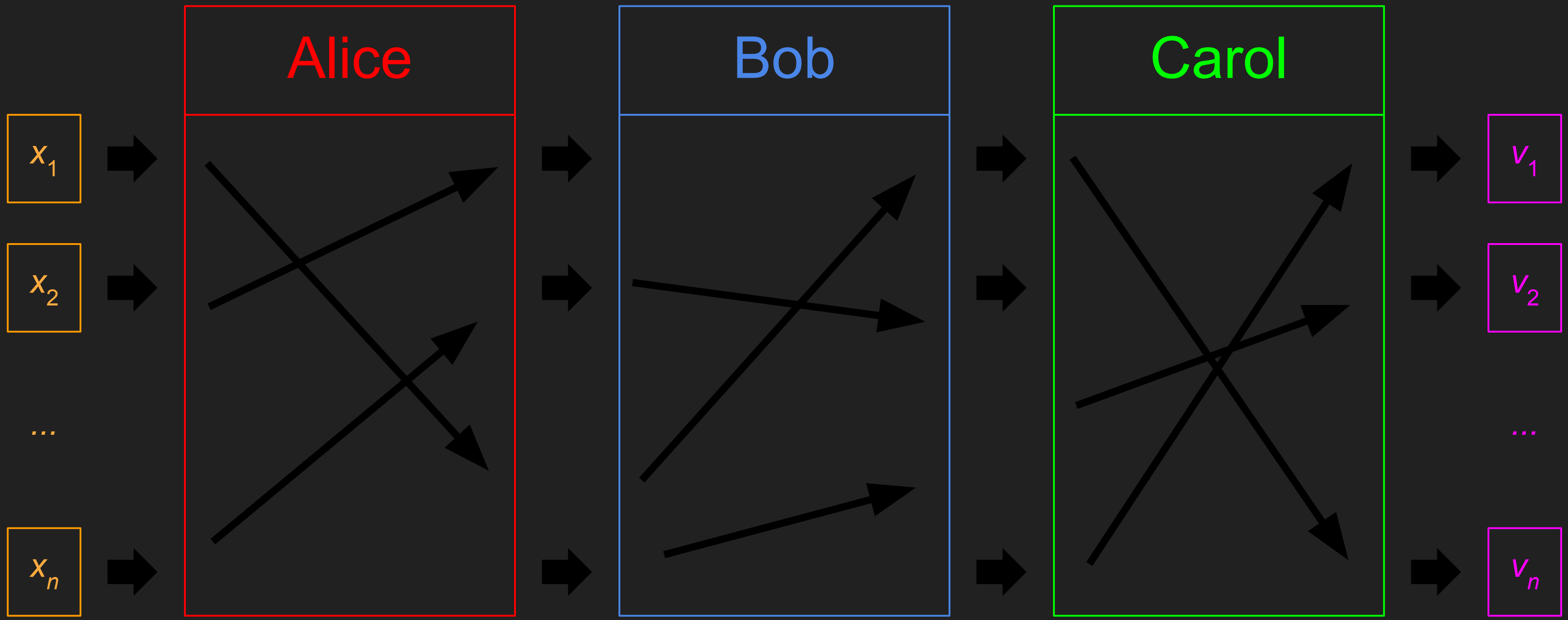
Electronic Voting: Goals

- Keep the vote anonymous
- Verify the vote totals
- Allow voters to verify that their votes were counted as cast
- Only authorized parties can vote
- No one votes more than one
- Coercion resistance: ensure Alice can’t prove who she voted for

Electronic Voting via Mixnets

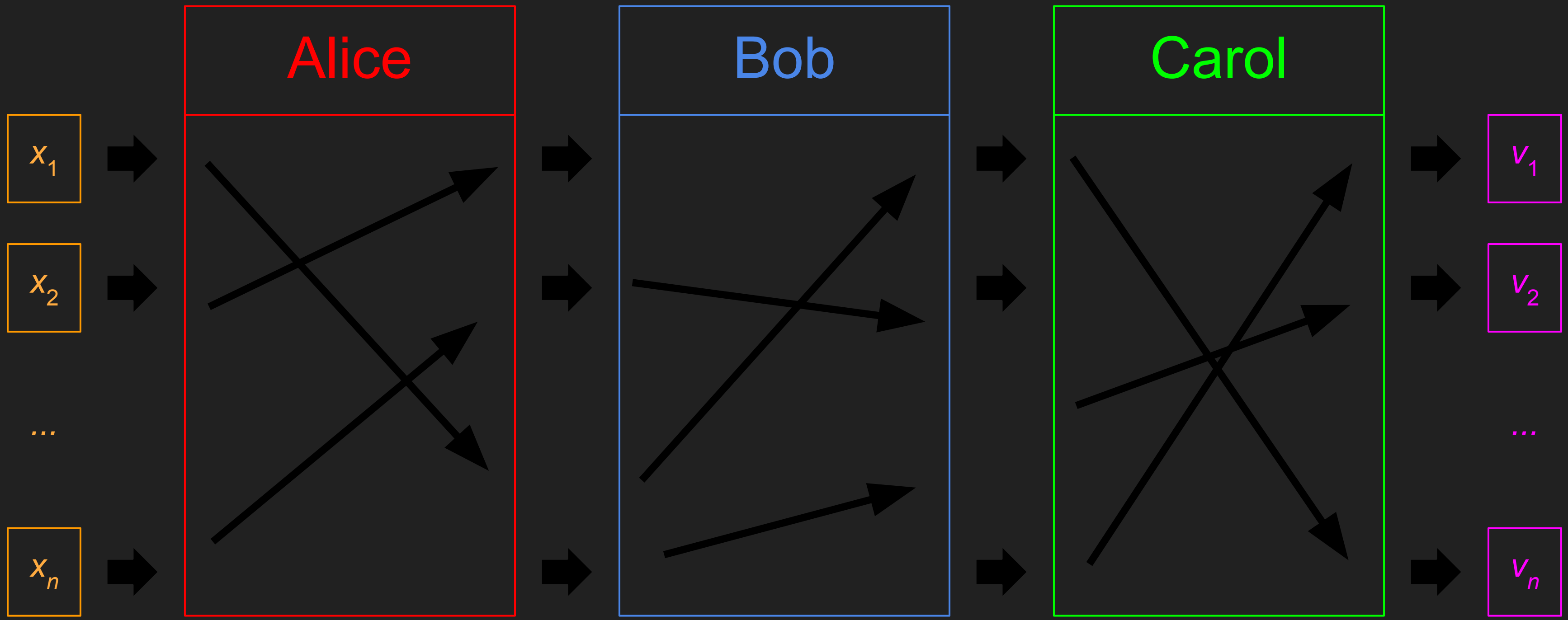


Electronic Voting via Mixnets: Encryption



x_i = Enc(Pub_A, Enc(Pub_B, Enc(Pub_C, v_i)))

Electronic Voting via Mixnets: Encryption with nonce



$$x_i = \mathrm{Enc}(\mathrm{Pub}_A, \mathrm{Enc}(\mathrm{Pub}_B, \mathrm{Enc}(\mathrm{Pub}_C, r_i \,||\, v_i)))$$

Homework

- Complete exercises 1 – 4 of Cryptopals Set 1 (http://www.cryptopals.com/sets/1)
  - These are easy challenges that lead to detecting and breaking a simple shift cipher
- Complete additional exercises for extra credit (e.g. exercises 5 and 6)
  - Feel free to jump ahead to other problem sets

Tips

- **Don’t hesitate to ask questions! (Slack, email, etc)**

Homework grading

- Send an email with the answer to exercise 4 to cm7bv@virginia.edu with the subject “MST Assignment 7 - <YOUR_UVA_ID>”
  - eg: “MST Assignment 7 - cm7bv”
- Include a brief (1-paragraph) description of what you did and how it went

Further Reading

- Udacity cs387: Applied Cryptography by Dave Evans (https://www.udacity.com/course/applied-cryptography--cs387)
- Zero Knowledge Proofs: An illustrated primer (https://blog.cryptographyengineering.com/2014/11/27/zero-knowledge-proofs-illustrated-primer/)
- Verifiable online elections (https://heliosvoting.org/)