CS-438 Decentralized Systems Engineering

Fall 2025

Week 7

Threat modeling

no system is "100% secure"

relevant scopes:

assets - what is to be protected
- user data
- system function (availability, reliability, ...)

boundaries - administrative domains,
enclaves - "least privilege" ...

adversaries - resources, capabilities (can/cannot)
goals - motivation

Algorithms — security assumptions

- DoS resistance / vulnerability
- node honesty — follow the protocol?
  - Byzantine behavior? — #, % threshold bound
  - Rationality
- node crash failure
- cryptographic (hardness) assumptions
- network conditions — availability, performance, jitter, reordering

Adversaries - categorizations

Local vs global

Ephemeral vs persistent ("advanced" APT)

passive vs active

threshold assumption

Decentralized Systems Engineering

CS-438 – Fall 2025

DEDIS

Pierluca Borsò-Tan and Bryan Ford

EPFL

Credits: H. Zhang, K. Nikitin, D. Kostic, C. Basescu, B. Ford, Wikimedia Commons

Cryptographic Tools for Decentralized Systems

Building trust among untrusted parties

Overview

- Shared-algorithm cryptography
- Symmetric-key cryptography
- Public-key cryptography
- Cryptographic hash functions
- Key infrastructure
- Threshold secret sharing

Introduction

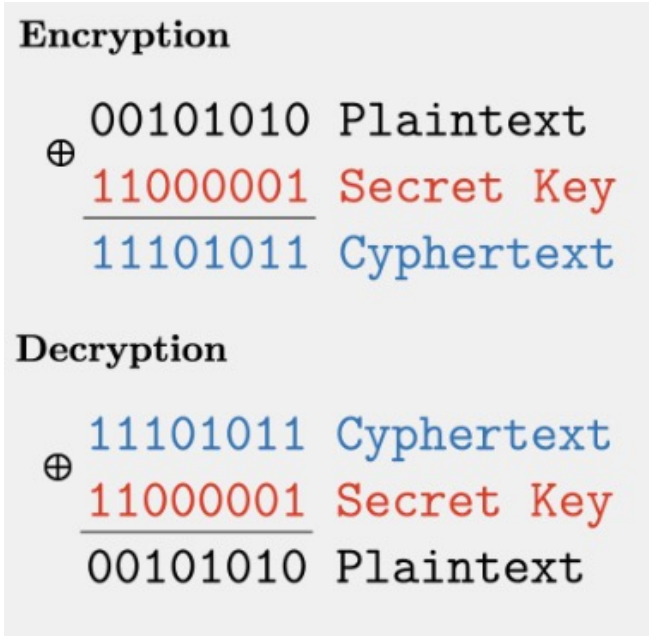
- What is cryptography?
  - A toolbox for many security mechanisms
  - A way to ensure information & communication security

    Secure data *at rest* and *in motion*

- Cryptography is not:
  - The solution to all security problems
  - Reliable unless implemented properly
  - Reliable unless used properly
  - Something you should try to invent yourself

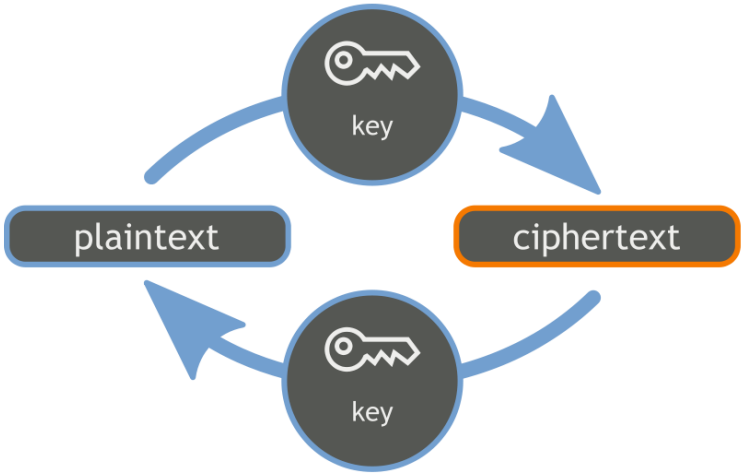
Overview

- **Shared-algorithm cryptography**



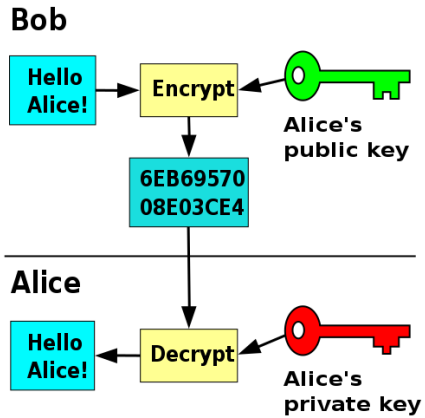
Overview

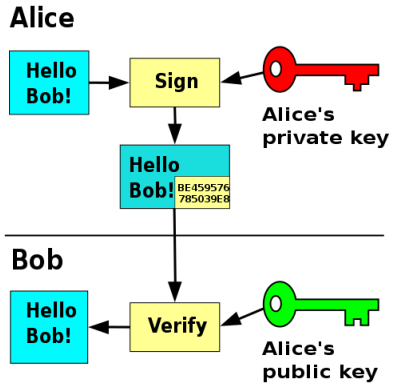
- Shared-algorithm cryptography
- **Symmetric-key cryptography**



Overview

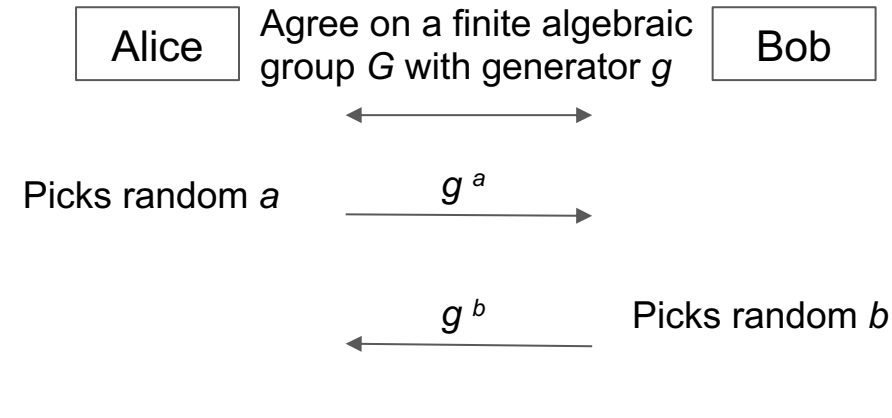
- Shared-algorithm cryptography
- Symmetric-key cryptography
- **Public-key cryptography**
  - Interactive Key Exchange (Diffie-Helmann)
  - Elliptic-curve Cryptography (ECC)

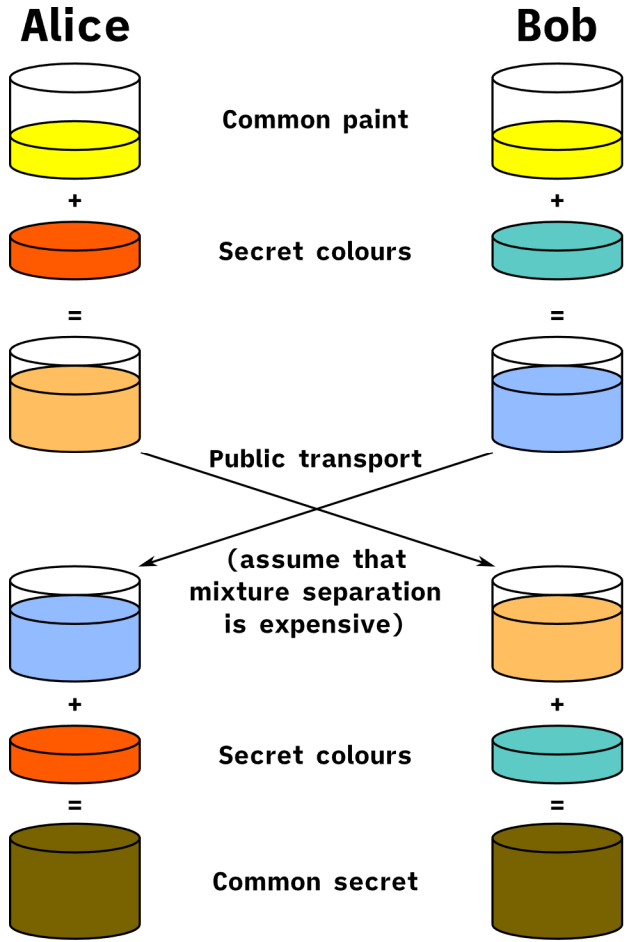




Interactive Key Exchange

- Diffie-Hellman Key Exchange
  (e.g. Handshake protocol in TLS)

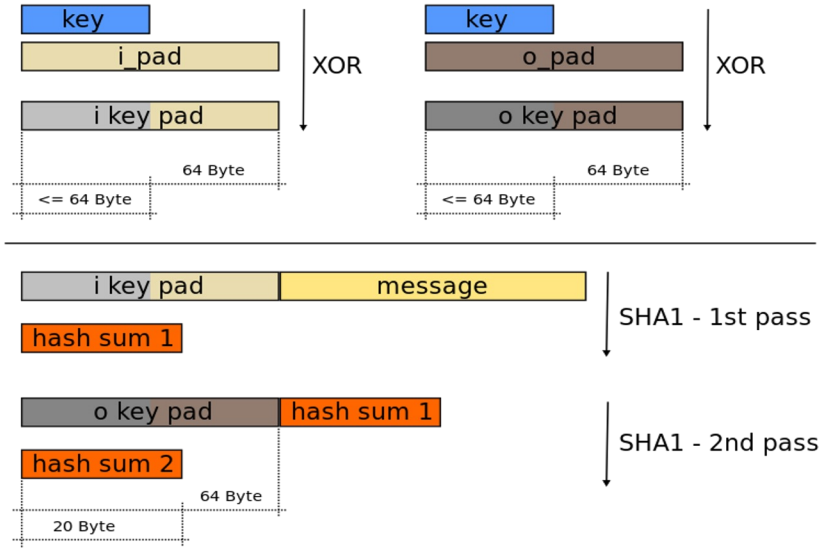




- Security relies on Discrete Logarithm Problem

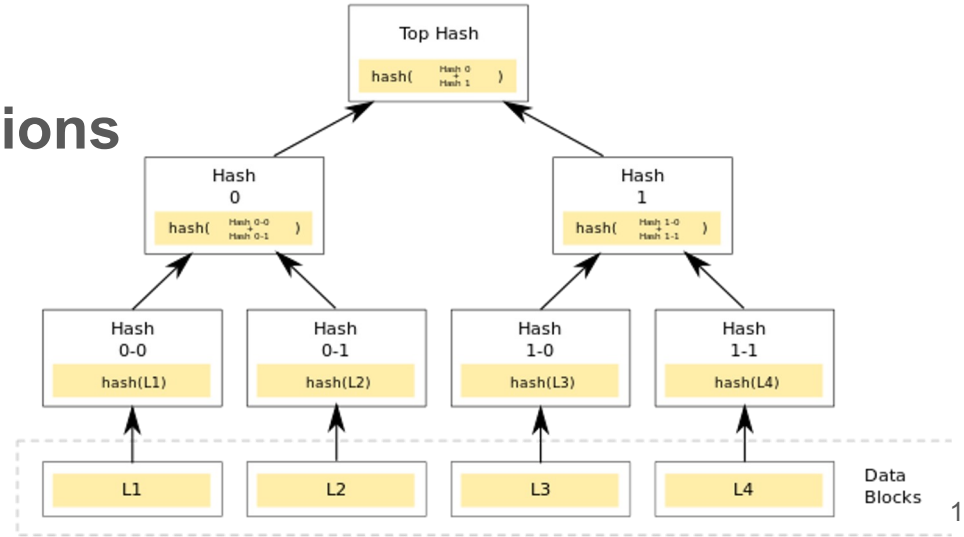
Overview

- Shared-algorithm cryptography
- Symmetric-key cryptography
- Public-key cryptography
- **Cryptographic hash functions**
  - HMAC
  - Merkle trees



Overview

- Shared-algorithm cryptography
- Symmetric-key cryptography
- Public-key cryptography
- **Cryptographic hash functions**
  - HMAC
  - Merkle trees



Overview

- Shared-algorithm cryptography
- Symmetric-key cryptography
- Public-key cryptography
- Cryptographic hash functions
- **Key infrastructure**

Key Distribution

- For both symmetric and asymmetric crypto we have to distribute keys
- Symmetric cryptosystems require the exchange of secret keys
  - Need for a secret/confidential channel
- Asymmetric cryptosystems require the exchange of public keys
  - Need for a trusted/integrity protected channel
- Authorities trusted to provide secret / trustworthy keys:
  - Key Distribution Centers (KDC)
  - Certification Authorities (CA)

Using Hierarchy of Trust

- One KDC/CA is not enough to serve all users
- KDCs/CAs are organized into hierarchies or peer networks

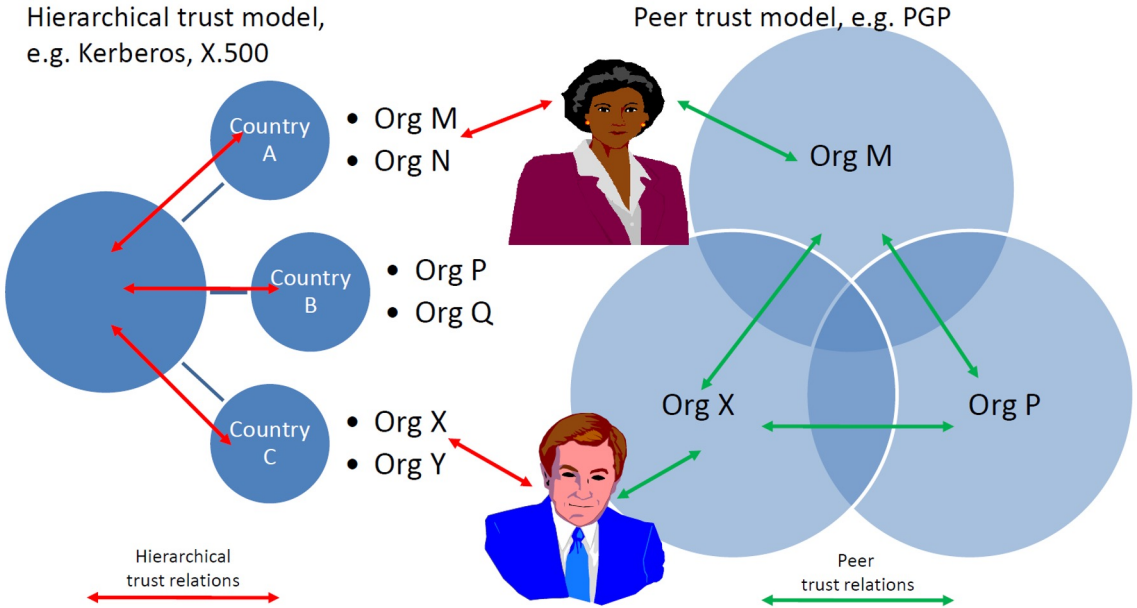


Figure Credit: P. Janson "IT Security Engineering" course

Public Key Infrastructure (PKI)

PKI binds *public keys* to their *owners*

- Certificate Authority (CA): stores, issues and signs the digital certificates
- Root certificate public keys are embedded in web browser
- Few Root CAs sign "delegation" certificates declaring that other CAs are also trusted to sign server certs
- Subsidiary "issuing" CA signs a certificate for, e.g., Google servers
- When your browser connects to Google server via SSL, the server sends its server-side certificate and the "chain" of signatures down from the root CA

Overview

- Shared-algorithm cryptography
- Symmetric-key cryptography
- Public-key cryptography
- Cryptographic hash functions
- Key infrastructure
- **Threshold secret sharing**

How do you share a secret without revealing it ?

Any encrypted data is secured with a *private key*

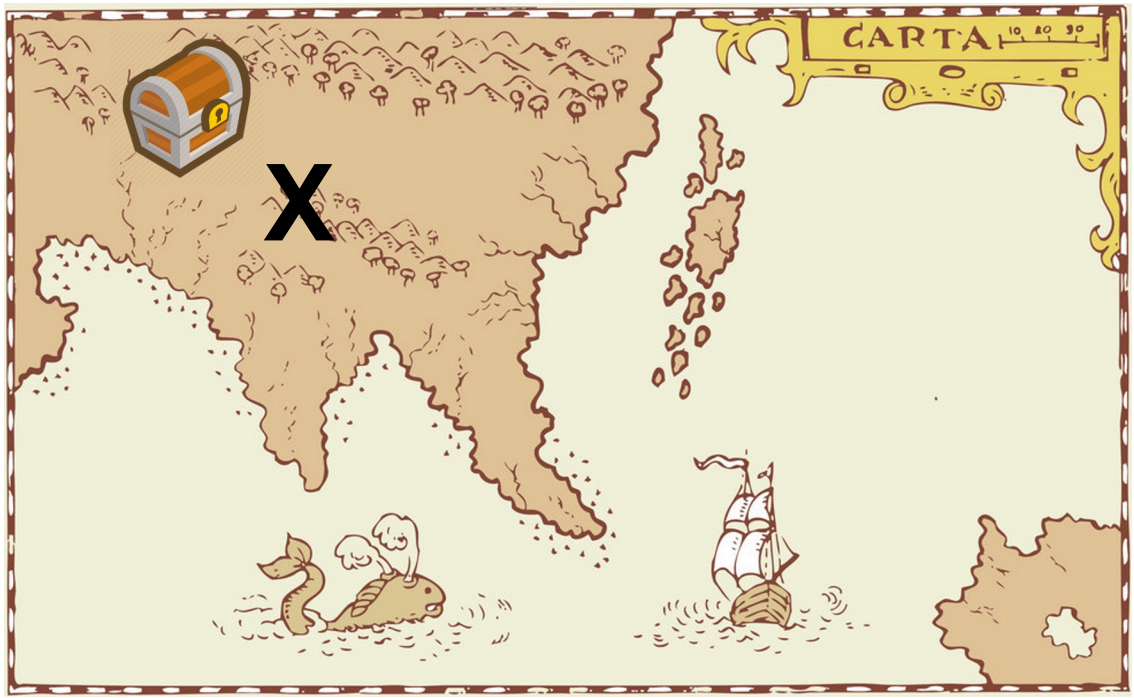
- A private key is *just information* (a number)!
- If the *key* leaks, anyone can decrypt the data
  - Regardless of where it's stored: cloud, blockchain…

Privacy & Accountability with secret-sharing

- Essential idea: after encrypting data,
  "deal" the secret key to a *threshold* ***t*** of ***n*** parties

Secret Sharing: Illustration

Suppose you're a pirate & bury your treasure…

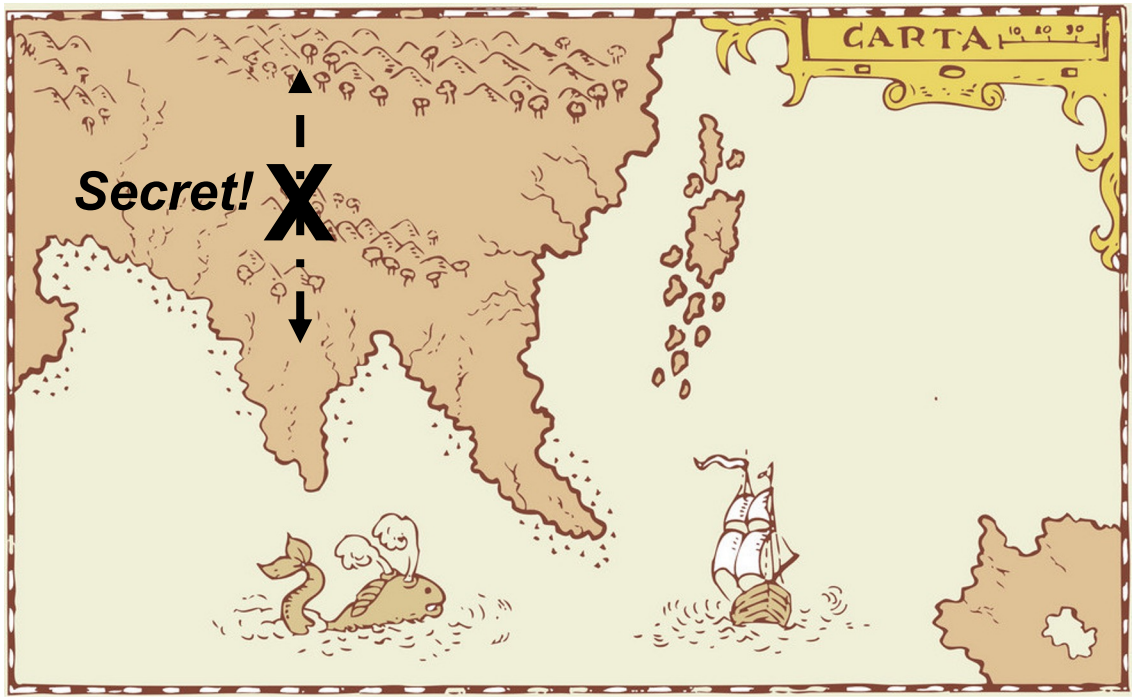


Keeping the Location Secret

You have 3 henchmen who you want to send back for it later, but you don’t trust *any one* completely

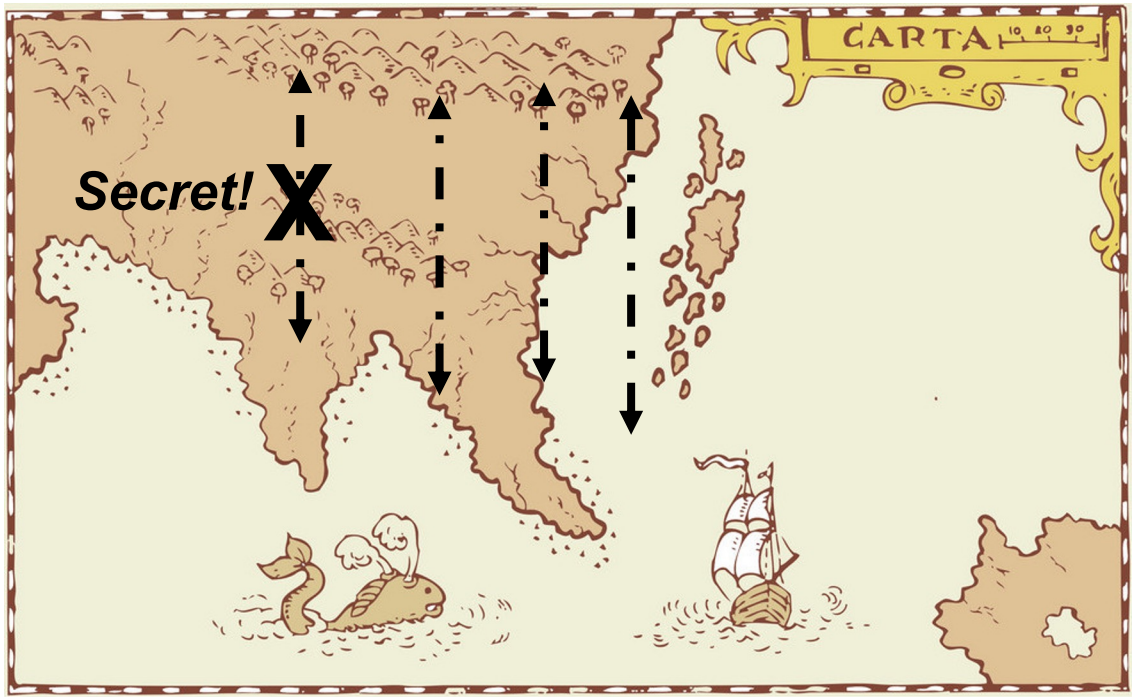
Secret Sharing: Illustration

You mark the spot between two reference points



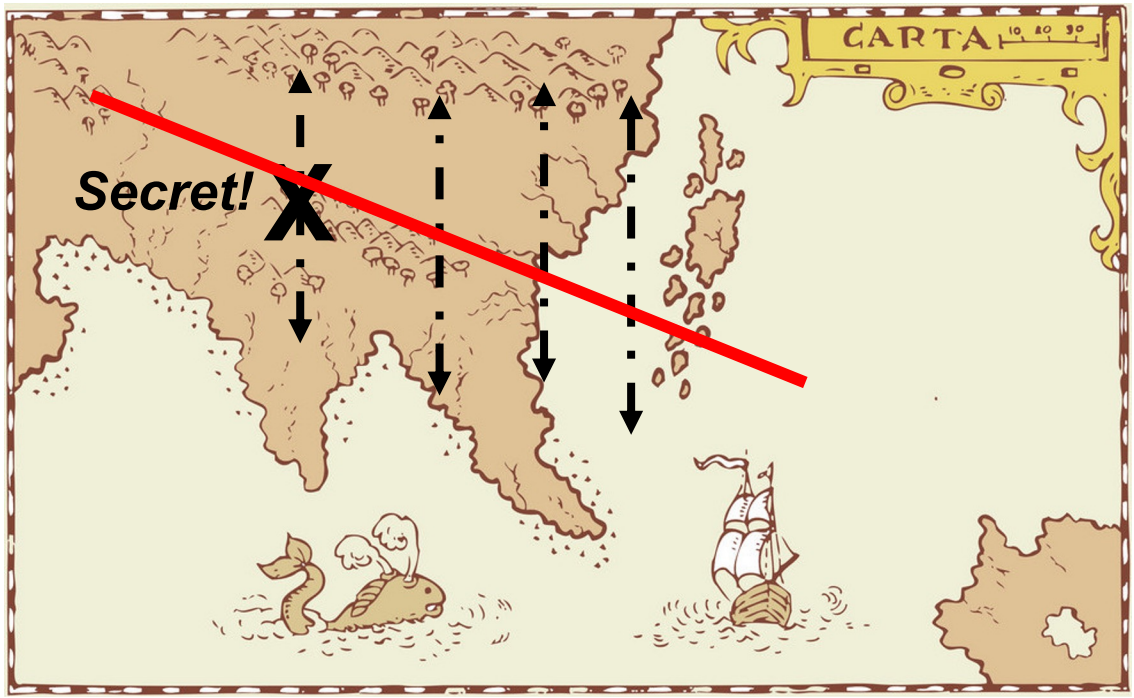
Secret Sharing: Illustration

Then draw three parallel reference lines…



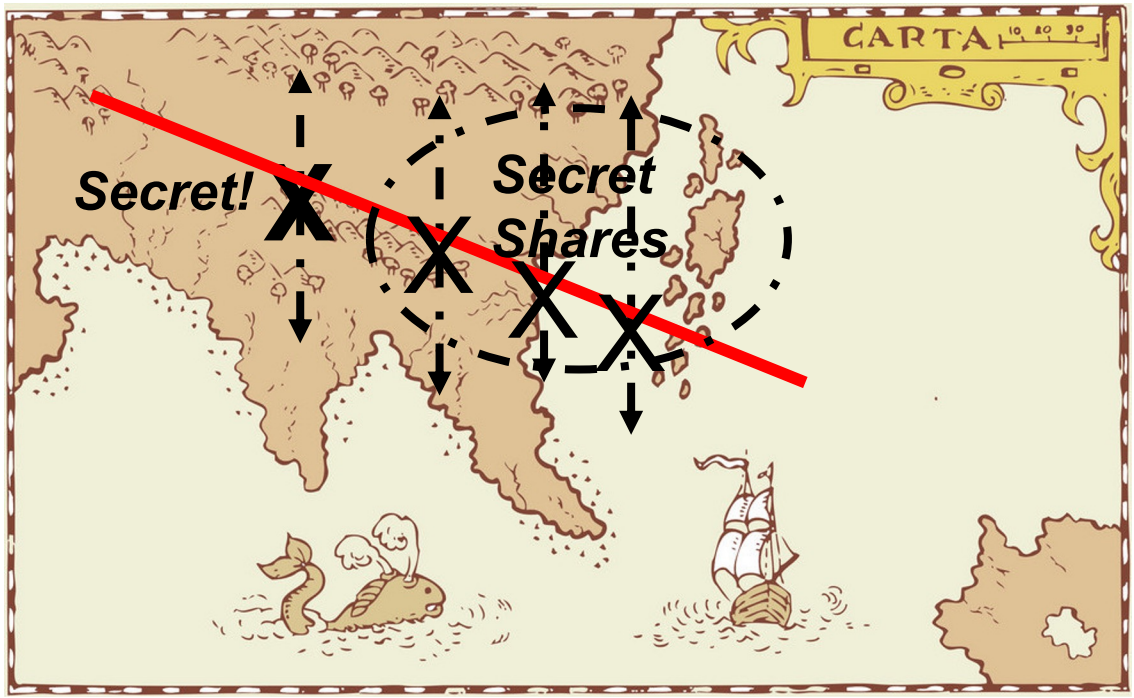
Secret Sharing: Illustration

…and another line intersecting all four…



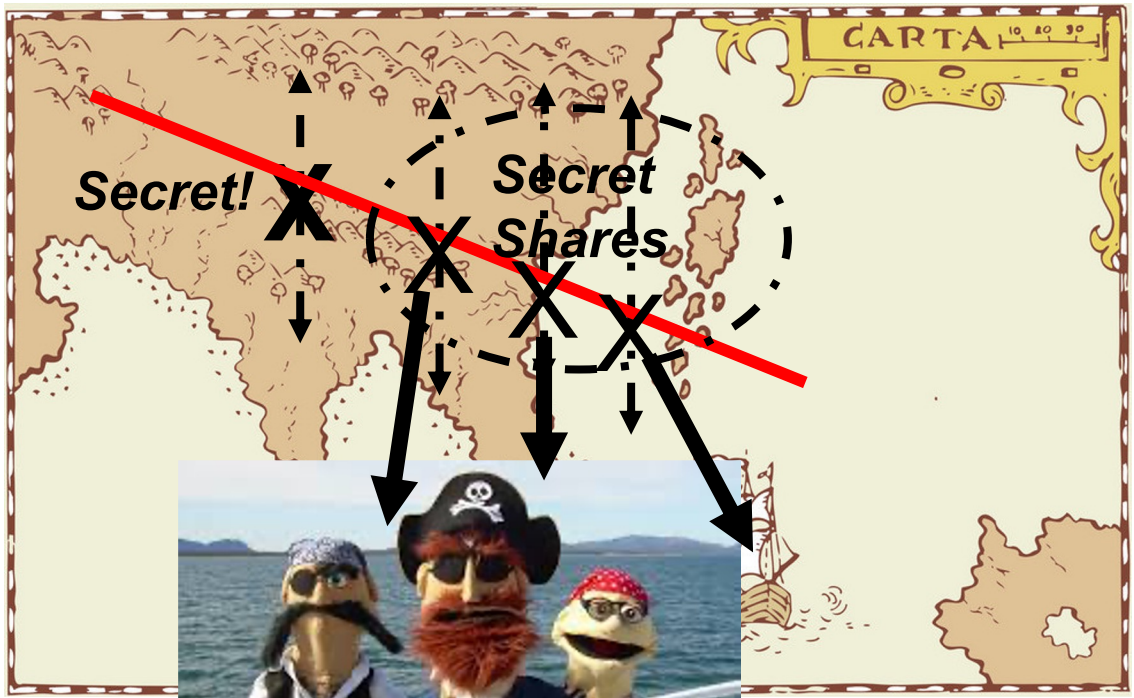
Secret Sharing: Illustration

The intersection points are the *secret shares...*



Secret Sharing: Illustration

You give *one* of these shares to *each* henchman



Threshold Secret Sharing

Now suppose your henchmen come back later to recover the treasure…

- Any **one** henchman won't know how to find it
- Any **two** henchmen will be able to!

You get both **threshold privacy** of the secret…

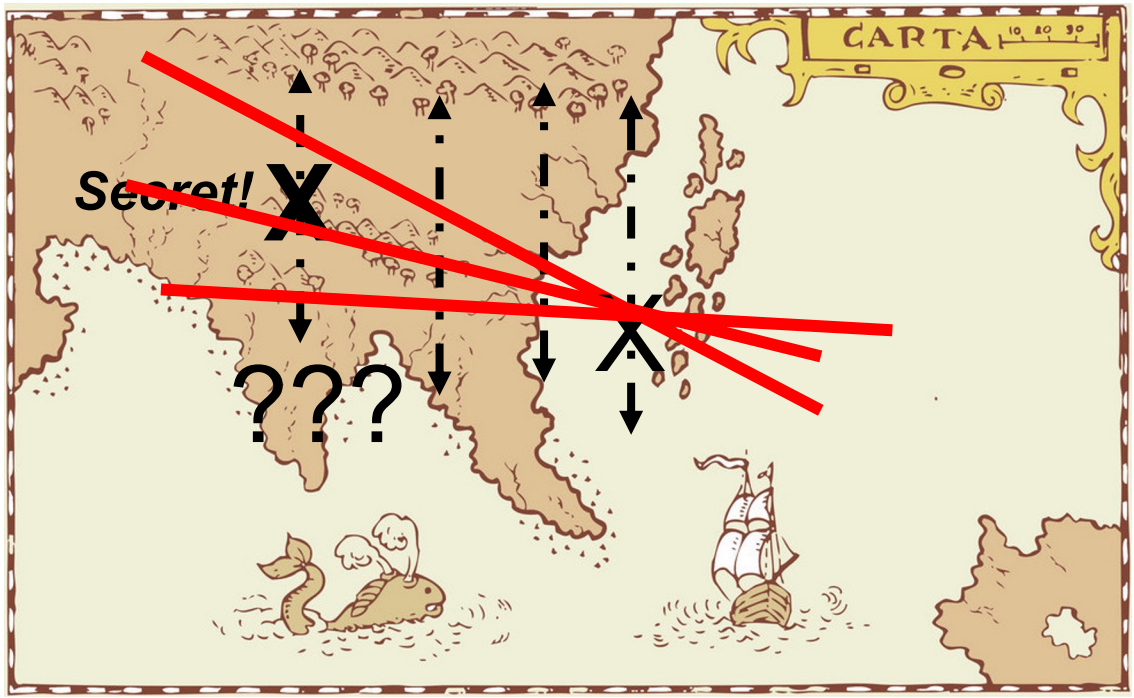
- No single compromised party can recover it

You also get **threshold availability** of the secret

- Can still recover if one henchman goes missing

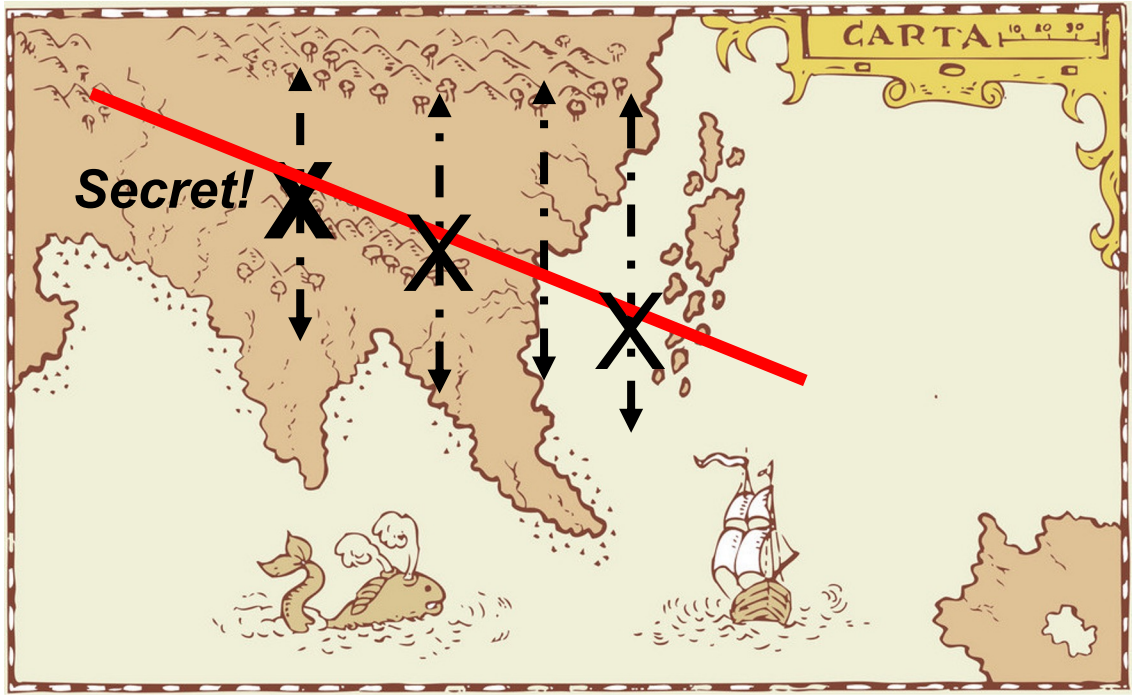
Secret Sharing: Illustration

One henchman alone can’t recover secret



Secret Sharing: Illustration

…but *any two* working together can!



Threshold Cryptosystem

- **Shamir's Secret Sharing (SSS)**
- **Verifiable Secret Sharing (VSS)**
- **Publicly Verifiable Secret Sharing (PVSS)**
- **Distributed Key Generation (DKG)**