Special request: Please spread out! Pair up. Each pair, sit far away from anyone else. If you’re just arriving, sit next to someone who is alone.
Tamper-evident Audit Logs

- $X_1 = H(X_0, \text{“opened vault”})$
- $X_2 = H(X_1, \text{“disabled alarm”})$
- $X_3 = H(X_2, \text{“closed alarm”})$
- $X_4 = H(X_3, \text{“front door locked”})$
- $X_5 = H(X_4, \text{“closed vault”})$

- Publishing any $X_i$ commits to all prior log entries.
Distributed Logging

• Let’s do distributed peer-to-peer logging of public data. We have $n$ computers; they all know each others’ public keys. Any computer can broadcast to all others (instantaneously, reliably). Any computer should be able to append a signed entry to the log, and to verify integrity of any previous log entry.

• Security goal: Malicious computers should not be able to back-date entries or modify past log entries. Assume $\leq 3$ computers are malicious.

• **Problem 1.** Describe a protocol for this. What does Alice do to append an entry? What do other computers need to do?
Your Solution

• To append log entry e:
• Other computers should:
Distributed Logging

- **Problem 2.** Let’s generalize. Suppose $m$ of the $n$ computers are malicious. If we make the obvious change to your protocol, for which $m$ can it be made secure?

- (a): for all $m < n$.
- (b): for all $m < n/2$.
- (c): for all $m < n/3$.
- (d): for all $m < \sqrt{n}$.
- (e): for all $m < O(\lg n)$.
Distributed Logging

• **Problem 2.** Let’s generalize. Suppose $m$ of the $n$ computers are malicious. If we make the obvious change to your protocol, for which $m$ can it be made secure?

  - (a): for all $m < n$.
  - (b): for all $m < n/2$.
  - (c): for all $m < n/3$.
  - (d): for all $m < \sqrt{n}$.
  - (e): for all $m < O(\lg n)$. 
Distributed Money

• Donna gets the brilliant idea to use this log to store financial transactions. Each person’s initial balance is public.

• To transfer $10 from Alice to Bob, Alice appends a signed log entry saying “I transfer $10 to Bob” and broadcasts it. Everyone can compute the updated balance for Alice and Bob.

• **Problem 3.** What are some ways that a malicious actor might try to attack this scheme? Is this a good scheme?
Your Answers

- Replay
- Denial of service attacks
- Broadcast doesn’t scale
- TOCTTOU vulnerability
Problems with This Scheme

- Initial balance is arbitrary
- Broadcasting is expensive and doesn’t scale
- A conspiracy of $n/2$ malicious computers can fork the audit log and steal all the money
- Sybil attacks: Anyone can set up millions of servers and thus have a 50% majority
A Tangent:
How Can I Prove I Am Rich?
Problem 5. To prove to Bob I’m not a spammer, Bob wants me to do 10 seconds of computation before I can send him an email. How can I prove to Bob that I wasted 10 seconds of CPU time, in a way that he can verify in milliseconds?
Problem 5. To prove to Bob I’m not a spammer, Bob wants me to do 10 seconds of computation before I can send him an email. How can I prove to Bob that I wasted 10 seconds of CPU time, in a way that he can verify in milliseconds?

Hint: Computing 1 billion SHA256 hashes might take 10 seconds.
Your Answers

• I compute:
• Bob verifies by:
Solution

• To prove that I wasted 10 seconds of CPU time, in a way that he can verify quickly:

• Bob sends me: $r$
• I look for $x$ such that $\text{first30}(\text{SHA256}(x || r)) = 0$
• I send Bob: $x$
• Bob can verify using a single hash.
Bitcoin

• Public, distributed, peer-to-peer audit log of all transactions.
• To append an entry to the log, the latest value must hash to something whose first 30 bits are zero; then broadcast it to everyone.
• Anyone who appends an entry to the log is given a small reward, in new money (a fraction of a Bitcoin).