Lecture 2

Crypto Background - II
Hash Pointers and Data Structures
Hash pointer
• pointer to where some info is stored, and
• cryptographic hash of the info

If we have a hash pointer, we can
• ask to get the info back, and
• verify that it hasn’t changed
H (...) will draw hash pointers like this
Building data structures with hash pointers
Linked list with hash pointers = “Blockchain”

use case: tamper-evident log
detecting tampering

use case: tamper-evident log
binary tree with hash pointers = “Merkle tree”
proving membership in a Merkle tree

show $O(\log n)$ items
Advantages of Merkle trees

- Tree holds many items, but just need to remember the root hash
- Can verify membership in $O(\log n)$ time/space

**Variant:** sorted Merkle tree

- can verify non-membership in $O(\log n)$
- show items before, after the missing one
More generally ...

Can use hash pointers in any pointer-based data structure that has no cycles
Digital Signatures
What we want from signatures

- Only you can sign, but anyone can verify
- Signature is tied to a particular document (can’t be cut-and-pasted to another doc)
- Even if one can see your signature on some documents, he cannot “forge” it
Digital signatures

- \((sk, pk) \leftarrow \text{keygen}(1^k)\)
  - \(sk\): secret signing key
  - \(pk\): public verification key

- \(\text{sig} \leftarrow \text{sign}(sk, \text{message})\)

- \(\text{isValid} \leftarrow \text{verify}(pk, \text{message}, \text{sig})\)

Security parameter

Randomized algorithm

Typically randomized
Requirements for signatures

- **Correctness**: “valid signatures verify”
  - $\text{verify}(pk, \text{message, sign}(sk, \text{message})) == \text{true}$

- **Unforgeability under chosen-message attacks (UF-CMA)**: “can’t forge signatures”
  - adversary who knows pk, and gets to see signatures on messages of his choice, can’t produce a verifiable signature on another message
(sk, pk) \xleftarrow{} \text{keygen}(1^k)

\text{pk}

\text{m}_0

\text{sign}(sk, \text{m}_0)

\text{m}_1

\text{sign}(sk, \text{m}_1)

\ldots

\text{M, sig}

\text{M not in } \{ \text{m}_0, \text{m}_1, \ldots \}

\text{Challenger} \xrightarrow{\text{verify(pk, M, sig)}} \text{Adversary}

\text{ifValid, attacker wins}

\textbf{Definition:} A signature scheme (keygen, sign, verify) is UF-CMA secure if for every PPT adversary A, there exists a negligible function n(k) s.t. \Pr[A \text{ wins in above game}] = n(k)
Notes

- Signatures can be shorter than message: sign Hash(message) rather than message
- Algorithms are randomized: need good source of randomness. Bad randomness may reveal the secret key
- Fun trick: sign a hash pointer. Signature “covers” the whole structure
Notes...

- Bitcoin uses Elliptic Curve Digital Signature Algorithm (ECDSA)
- ECDSA is a close variant of Schnorr Signature scheme over Elliptic curves