Lecture 2

Crypto Background – II

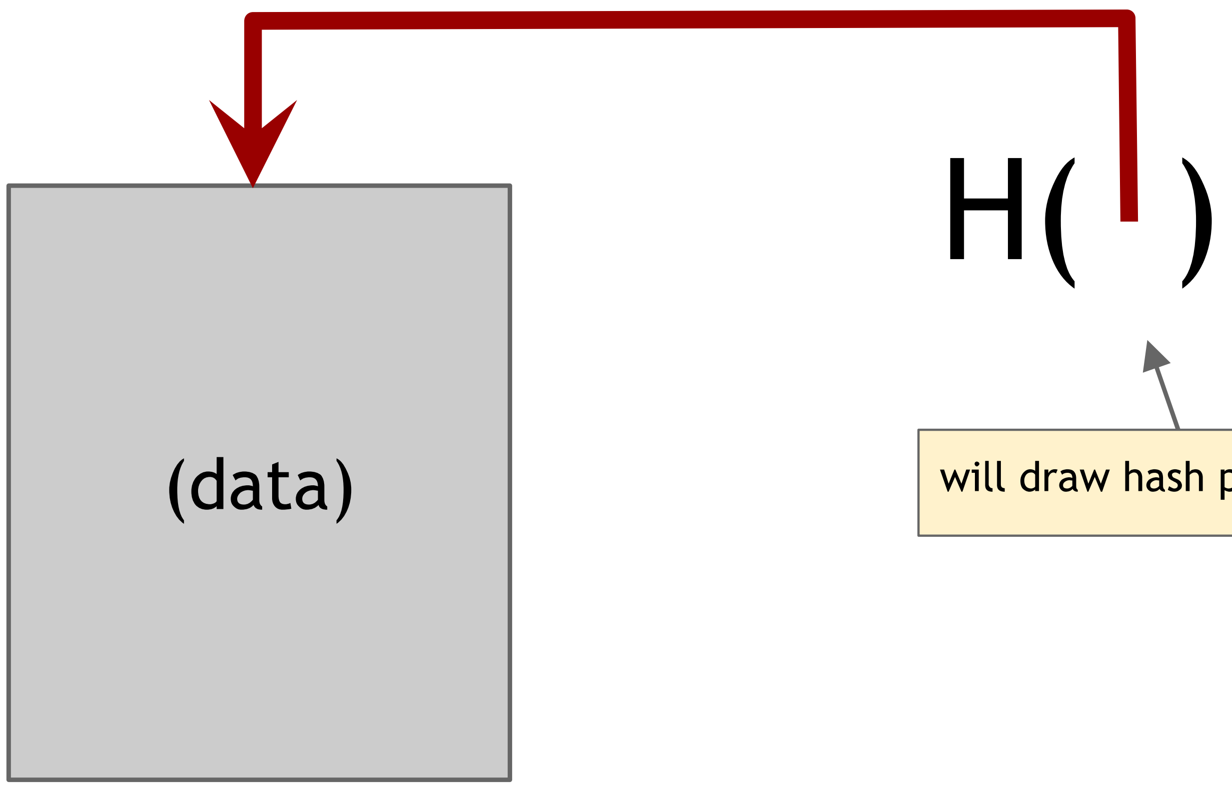
Hash Pointers and Data Structures

Hash pointer

- pointer to where some info is stored, *and*
- cryptographic hash of the info

If we have a hash pointer, we can

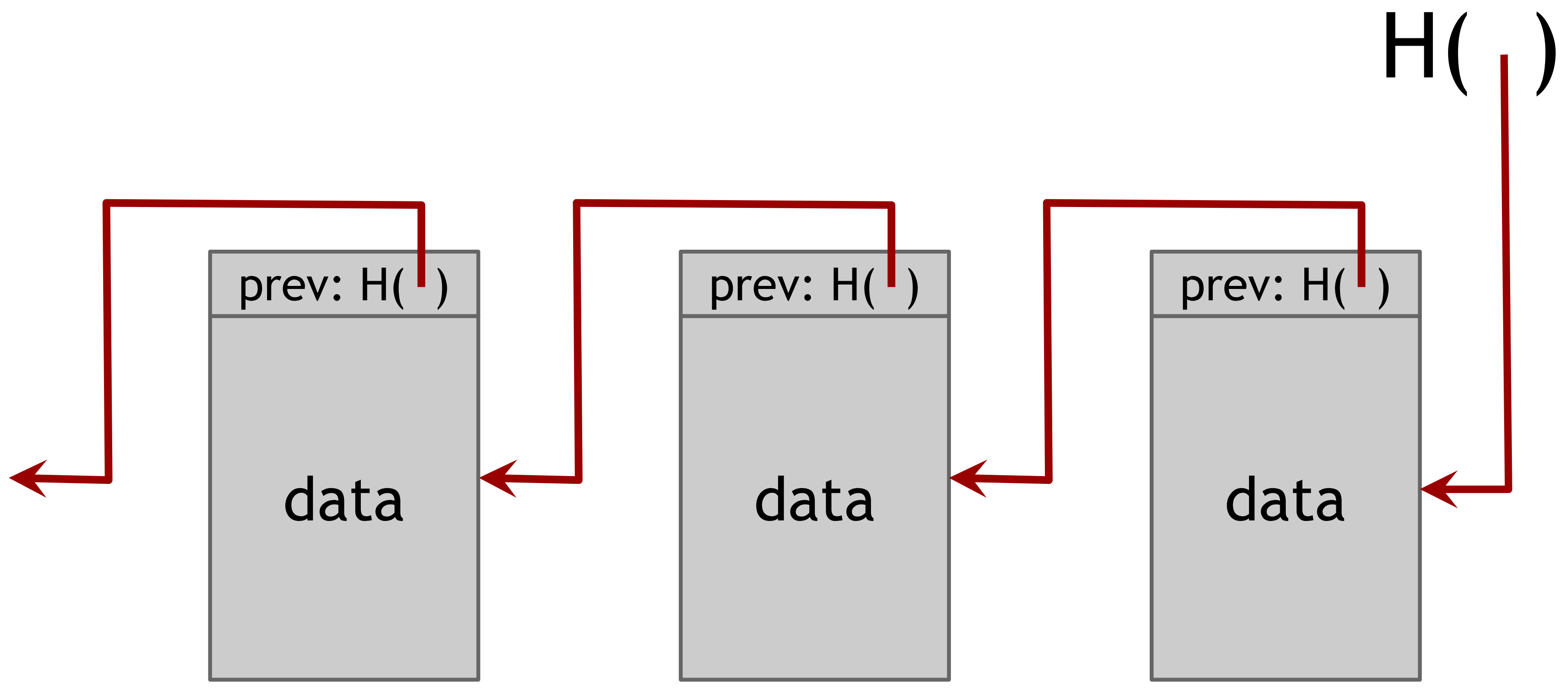
- ask to get the info back, and
- verify that it hasn’t changed



will draw hash pointers like this

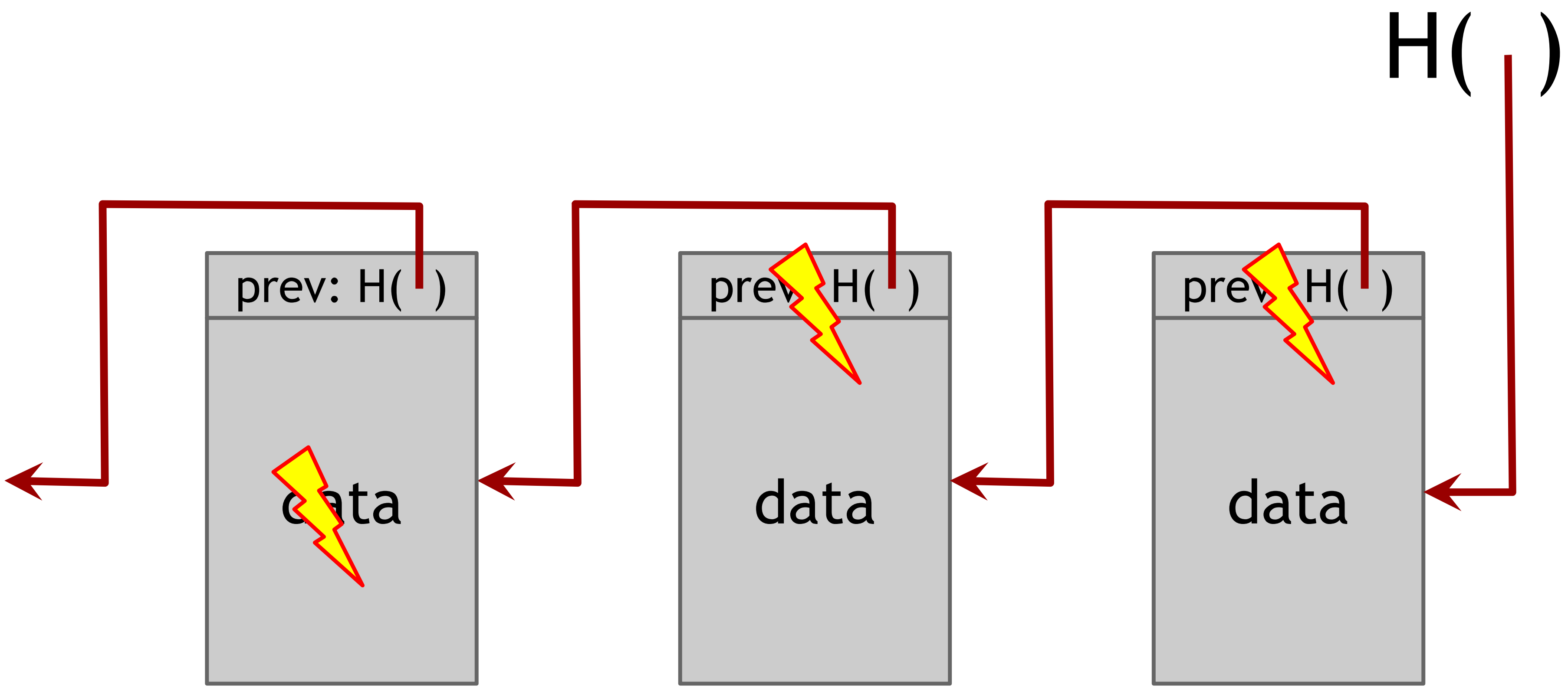
Building data structures with hash pointers

Linked list with hash pointers = “Blockchain”



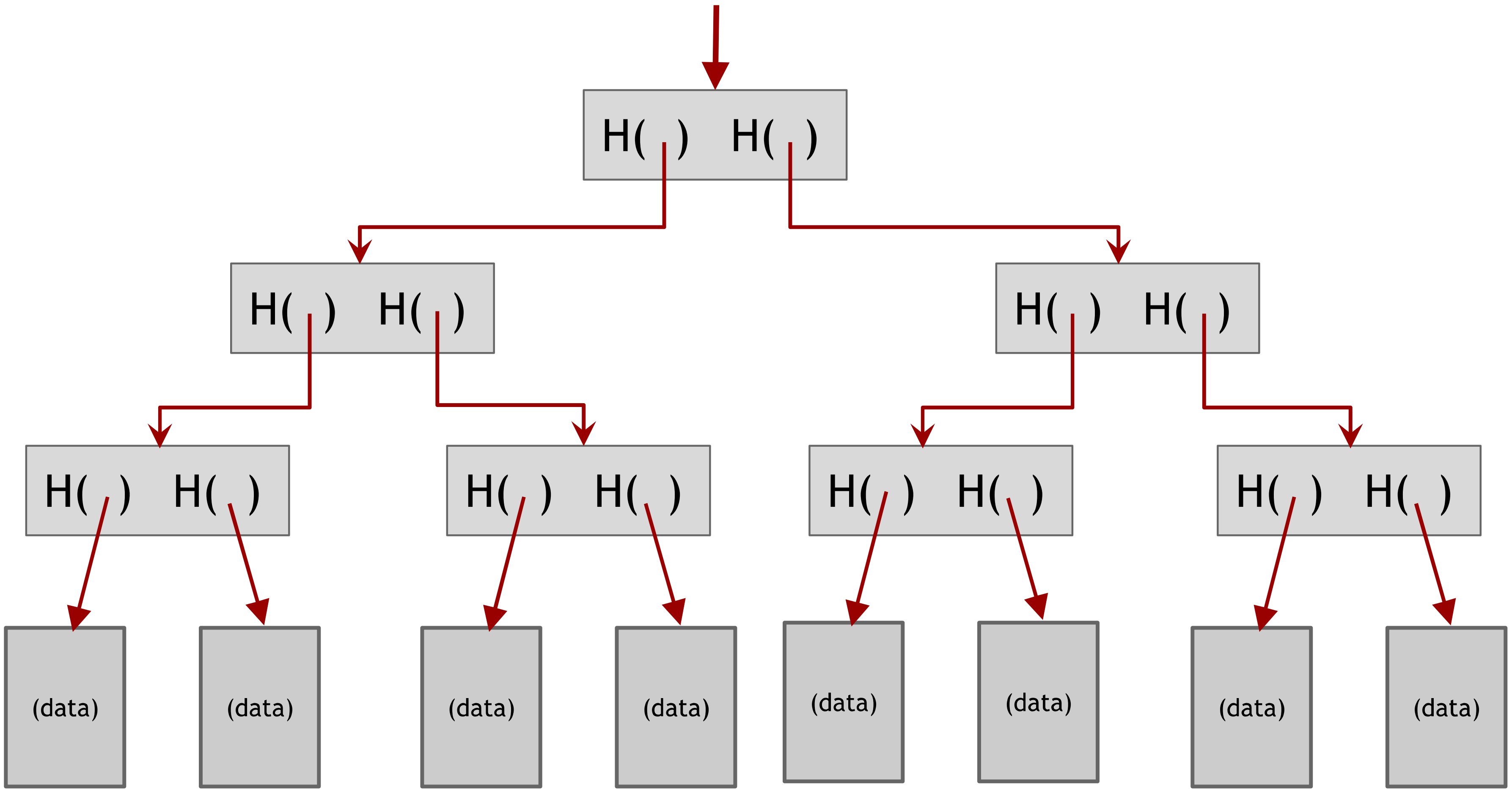
use case: tamper-evident log

detecting tampering



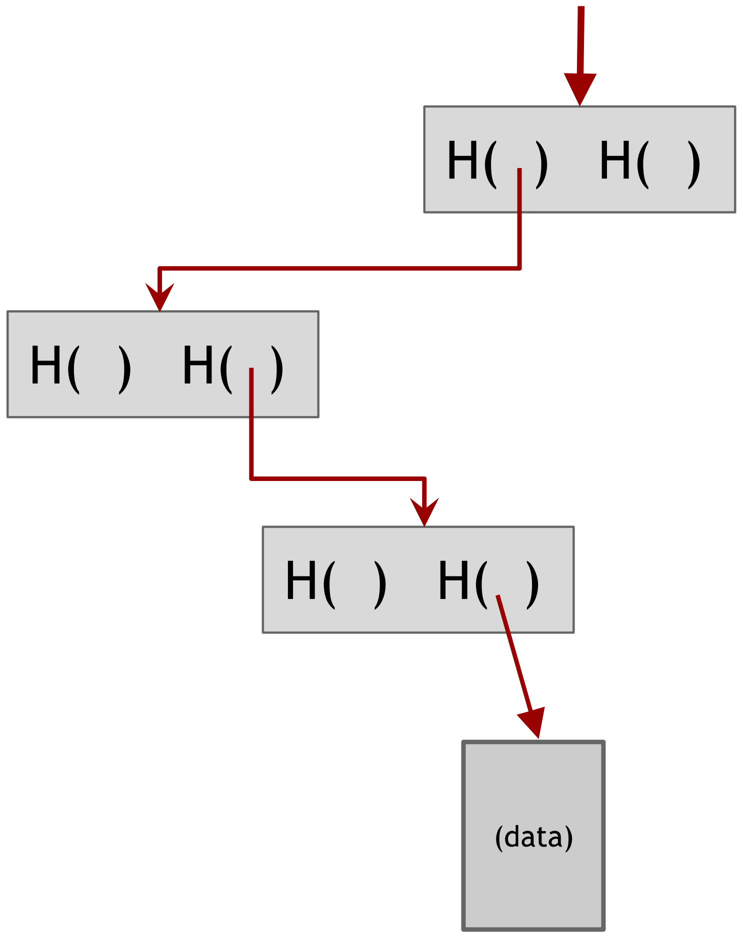
use case: tamper-evident log

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

<u>Variant</u>: *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

Security parameter

- (sk, pk) ← keygen(1^k)

  sk: secret signing key

  pk: public verification key

$\}$ randomized algorithm

- sig ← sign(sk, message)

$\}$ Typically randomized

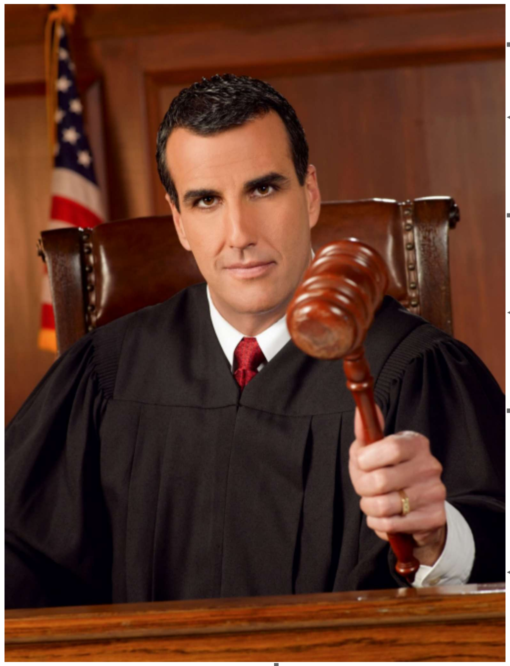
- isValid ← verify(pk, message, sig)

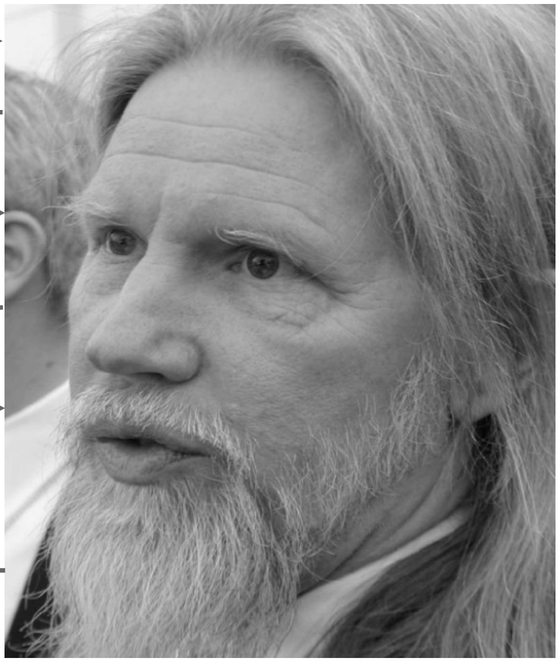
Requirements for signatures

- Correctness: “valid signatures verify”
  - verify(pk, message, sign(sk, message)) == 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

UF-CMA Security

(sk, pk) ← keygen(1^k)





$$\begin{tikzcd}
\text{Challenger} \arrow[r, "pk"] & \text{Adversary} \arrow[l, "m_0"] \\
\text{Challenger} \arrow[r, "\text{sign}(sk,\ m_0)"] & \text{Adversary} \arrow[l, "m_1"] \\
\text{Challenger} \arrow[r, "\text{sign}(sk,\ m_1)"] & \text{Adversary} \\
\bullet\ \bullet\ \bullet & \\
\text{Challenger} & \text{Adversary} \arrow[l, "M,\ sig"']
\end{tikzcd}$$

M not in { m_0, m_1, … }

Challenger

verify(pk, M, sig)

ifValid, attacker wins

Adversary

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 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