

---

# SECURITY (COMP0141): USES OF HASH FUNCTIONS



# HASH FUNCTIONS

---

Two main security properties:

- **Pre-image resistance:** given  $H(x)$  it's hard to find  $x$
- **Collision resistance:** it's hard to find  $x$  and  $y$  so that  $x \neq y$  but  $H(x) = H(y)$

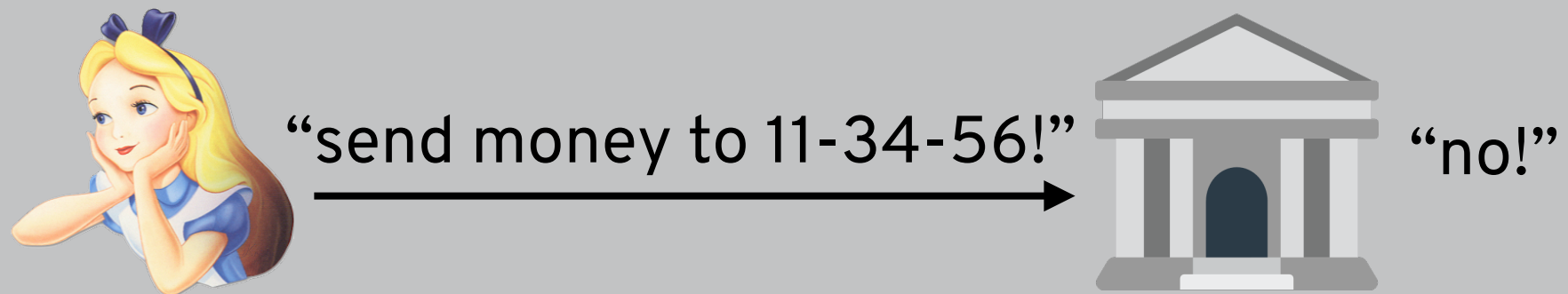
Applications:

- File checksum
- MACs
- Digital signatures
- Commitments
- Blockchains
- Virus scanning (next week)
- Password storage (Week 7)
- ...and many more!

# CHECKSUM

---

Used to detect errors introduced by humans (replaced digits, transposition, phonetic, etc.)



12-34-56 12345678

Also useful for errors due to corruption (inevitable for big files)



# CHECKSUM

---

**Example:** validating an IBAN (International Bank Account Number)

GB82 WEST 123456 12345678

WEST12345612345678GB82

3214282912345612345678161182

3214282912345612345678161182 mod 97?

= 0 so we know this isn't a valid IBAN (need = 1 mod 97)

# HASH FUNCTIONS

---

Two main security properties:

- **Pre-image resistance:** given  $H(x)$  it's hard to find  $x$
- **Collision resistance:** it's hard to find  $x$  and  $y$  so that  $x \neq y$  but  $H(x) = H(y)$

Applications:

- File checksum
- MACs
- Digital signatures
- Commitments
- Blockchains
- Virus scanning (next week)
- Password storage (Week 7)
- ...and many more!

# HMAC

---

Uses a hash function to achieve a MAC

$$\text{HMAC}(K,m) = H((K \oplus \text{opad}) \parallel H((K \oplus \text{ipad}) \parallel m))$$

- so first we compute  $h_{\text{inner}} = H((K \oplus \text{ipad}) \parallel m)$
- then we compute  $H((K \oplus \text{opad}) \parallel h_{\text{inner}})$

opad and ipad are fixed strings

HMAC: Keyed-Hashing for Message Authentication. M. Bellare, R. Canetti, H. Krawczyk. RFC 2104.

# DANGEROUS HASH-BASED MAC

---

Why not do something simpler like  $\text{MAC}(K,m) = H(K \parallel m)$ ?

This is subject to something called a **length extension attack**

**(You should never design your own cryptography!)**

# HASH FUNCTIONS

---

Two main security properties:

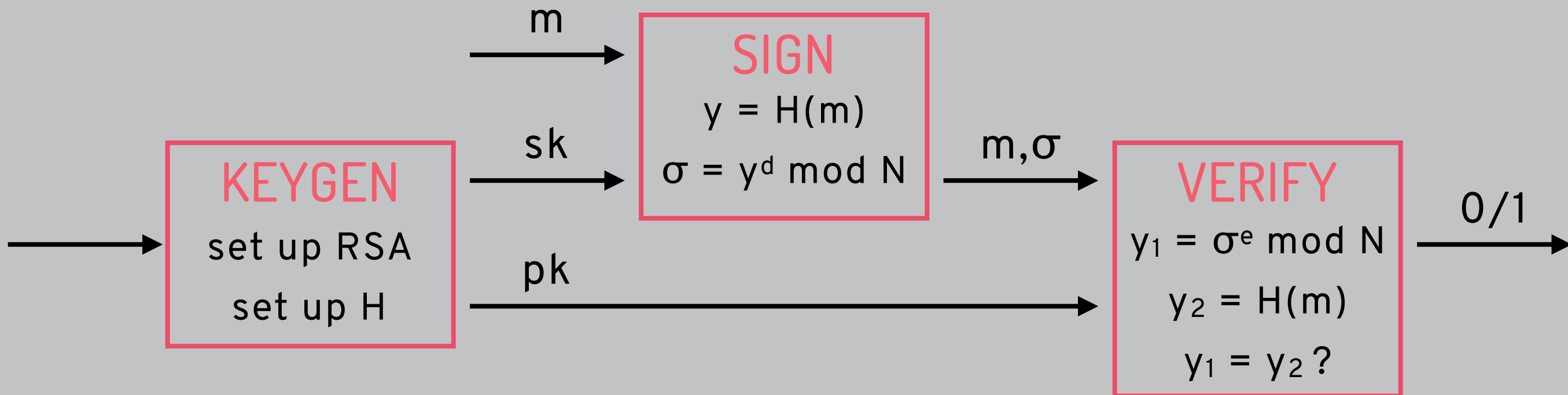
- **Pre-image resistance:** given  $H(x)$  it's hard to find  $x$
- **Collision resistance:** it's hard to find  $x$  and  $y$  so that  $x \neq y$  but  $H(x) = H(y)$

Applications:

- File checksum
- MACs
- Digital signatures
- Commitments
- Blockchains
- Virus scanning (next week)
- Password storage (Week 7)
- ...and many more!



# FULL DOMAIN HASH (FDH)



**Correctness:** Relies on having range of H be  $\mathbb{Z}/N\mathbb{Z}$

**Unforgeability:** we can prove EUF-CMA security assuming hash function behaves like a **random oracle**

Exemplifies the “**hash-and-sign**” paradigm

# HASH FUNCTIONS

---

Two main security properties:

- **Pre-image resistance:** given  $H(x)$  it's hard to find  $x$
- **Collision resistance:** it's hard to find  $x$  and  $y$  so that  $x \neq y$  but  $H(x) = H(y)$

Applications:

- File checksum
- MACs
- Digital signatures
- **Commitments**
- Blockchains
- Virus scanning (next week)
- Password storage (Week 7)
- ...and many more!

# COMMITMENTS

---

Let's play a game: **will we be out of lockdown at the end of term?**

Need people to **commit** to their guesses in a way such that...

- **Hiding:** no one else can see who guessed what
- **Binding:** no one can pretend they guessed right if they didn't

Everyone can compute  $H(\text{guess}||r)$  for some random  $r$  and store it on a public bulletin board

At the end, someone can reveal  $(\text{guess}, r)$  and everyone can check that  $H(\text{guess}||r) = h_i$  for some  $h_i$  stored on the board

This forms a type of (binding) **prediction market**

# HASH FUNCTIONS

---

Two main security properties:

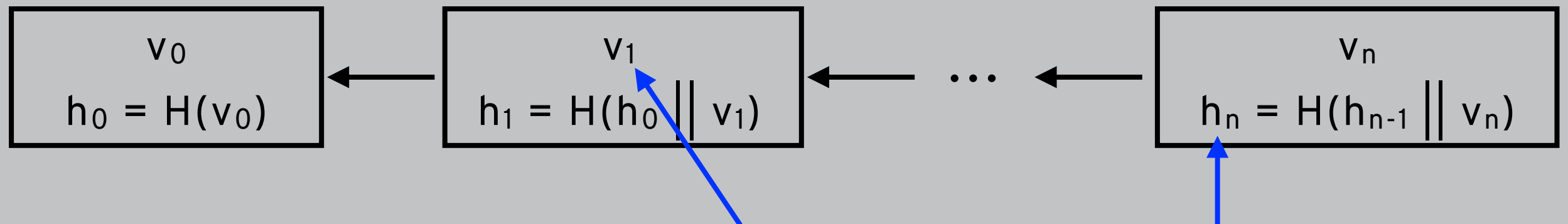
- **Pre-image resistance:** given  $H(x)$  it's hard to find  $x$
- **Collision resistance:** it's hard to find  $x$  and  $y$  so that  $x \neq y$  but  $H(x) = H(y)$

Applications:

- File checksum
- MACs
- Digital signatures
- Commitments
- **Blockchains**
- Virus scanning (next week)
- Password storage (Week 7)
- ...and many more!

# BLOCKCHAINS / HASH CHAINS

More generally, hashes can commit to entire series of values to form a **tamper-evident data structure**



can't replace old values without breaking pre-image resistance  
(need to find different  $h_{i-1}^*$  such that  $H(h_{i-1}^* \parallel v_i) = h_i$ )

In a blockchain, values represent blocks (collections of transactions + “proof-of-work”), so if we wanted to erase an old transaction we would have to rewrite the entire history and redo all the “work” since then

# HASH FUNCTIONS

---

Two main security properties:

- **Pre-image resistance:** given  $H(x)$  it's hard to find  $x$
- **Collision resistance:** it's hard to find  $x$  and  $y$  so that  $x \neq y$  but  $H(x) = H(y)$

Applications:

- File checksum
- MACs
- Digital signatures
- Commitments
- Blockchains
- Virus scanning (next week)
- Password storage (Week 7)
- ...and many more!