Better living through cryptography

Secure Systems Engineering Fall 2024

🛡️ `EE G7701`

September 18, 2024

Tushar Jois

Recap

- Buffer overflows can be used to hijack the control flow of a program
- Either shellcode or a return-to-libc can spawn a shell afterward
- Defenses against buffer overflows exist, but have shortcomings

Lesson objectives

- Explain how symmetric encryption and digital signatures work
- Describe the Diffie-Hellman key exchange protocol
- Compose cryptographic primitives to build secure systems


m
public channel
Alice
Bob
m = "secret"
Eve
m → "secret"
m → "secret"

Symmetric encryption

c

public channel

Alice

Bob

only the people who have the secret key can decrypt the ciphertext

What if they've **never met?**

Eve

m = "secret"

c = Encrypt(m, sk)

m = Decrypt(c, sk)

$m \rightarrow$ "secret"

$c \rightarrow$ 10100100001010101

Symmetric encryption

$sk, \; c$

public channel

Alice

Bob

Idea: encrypt with something only Alice and Bob could know

m = "secret"

c = Encrypt(m, sk)

Eve

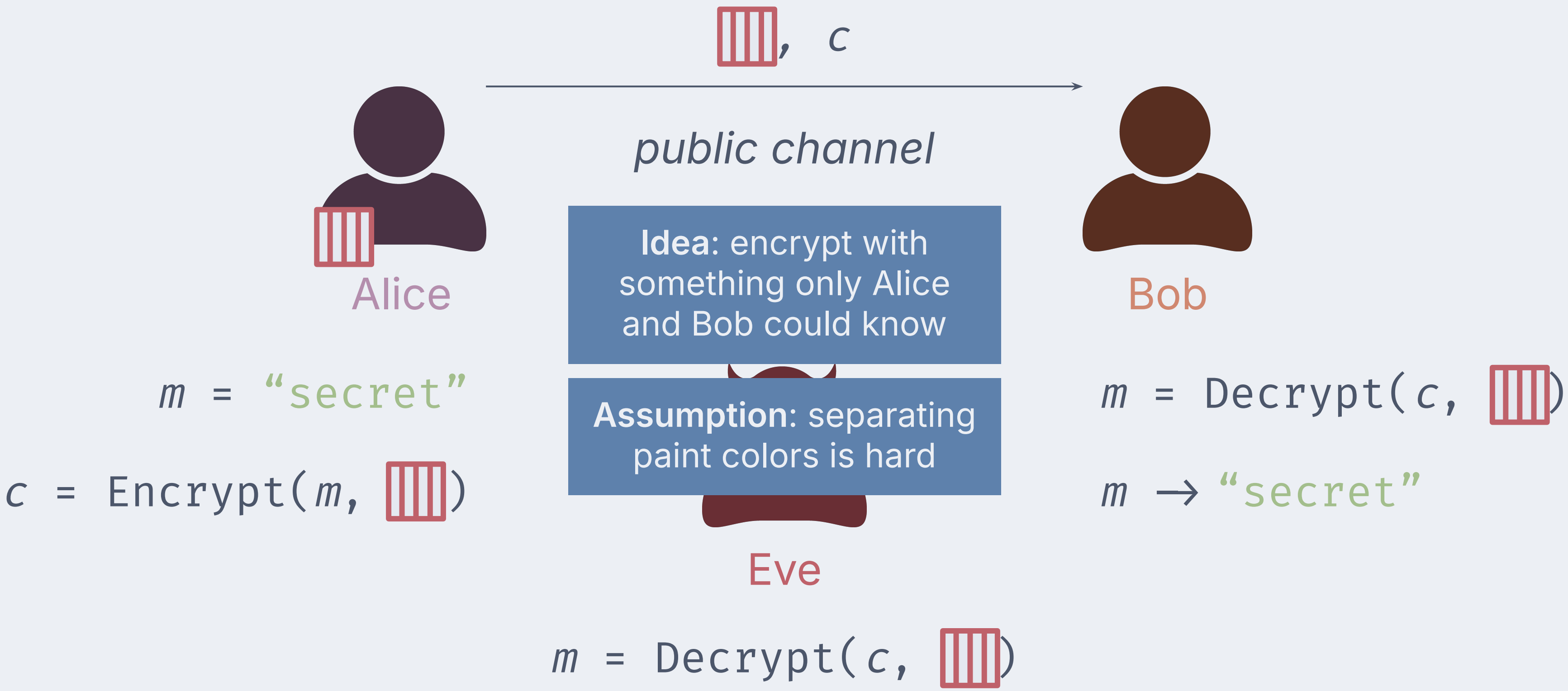
m = Decrypt(c, sk)

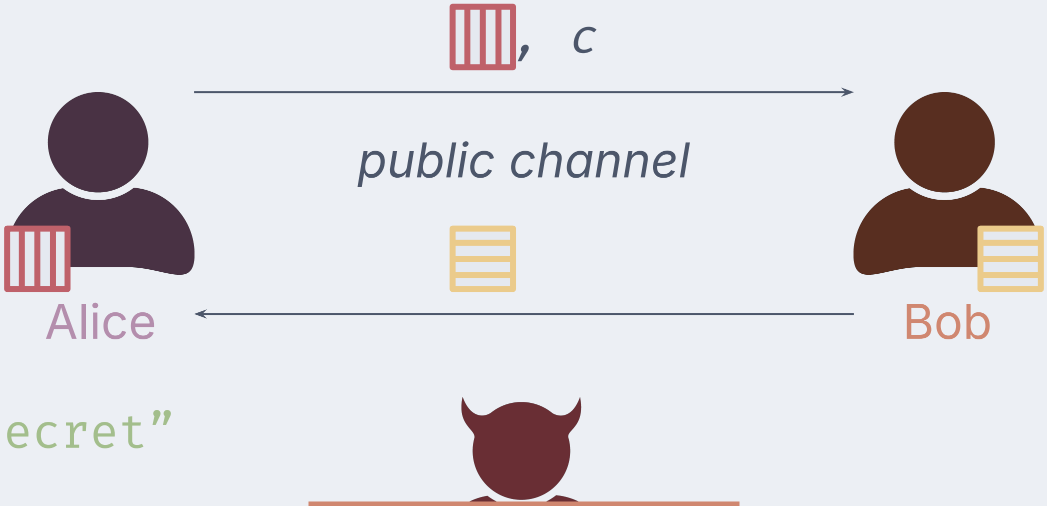
$m \rightarrow$ "secret"

Bad idea!

m = Decrypt(c, sk)

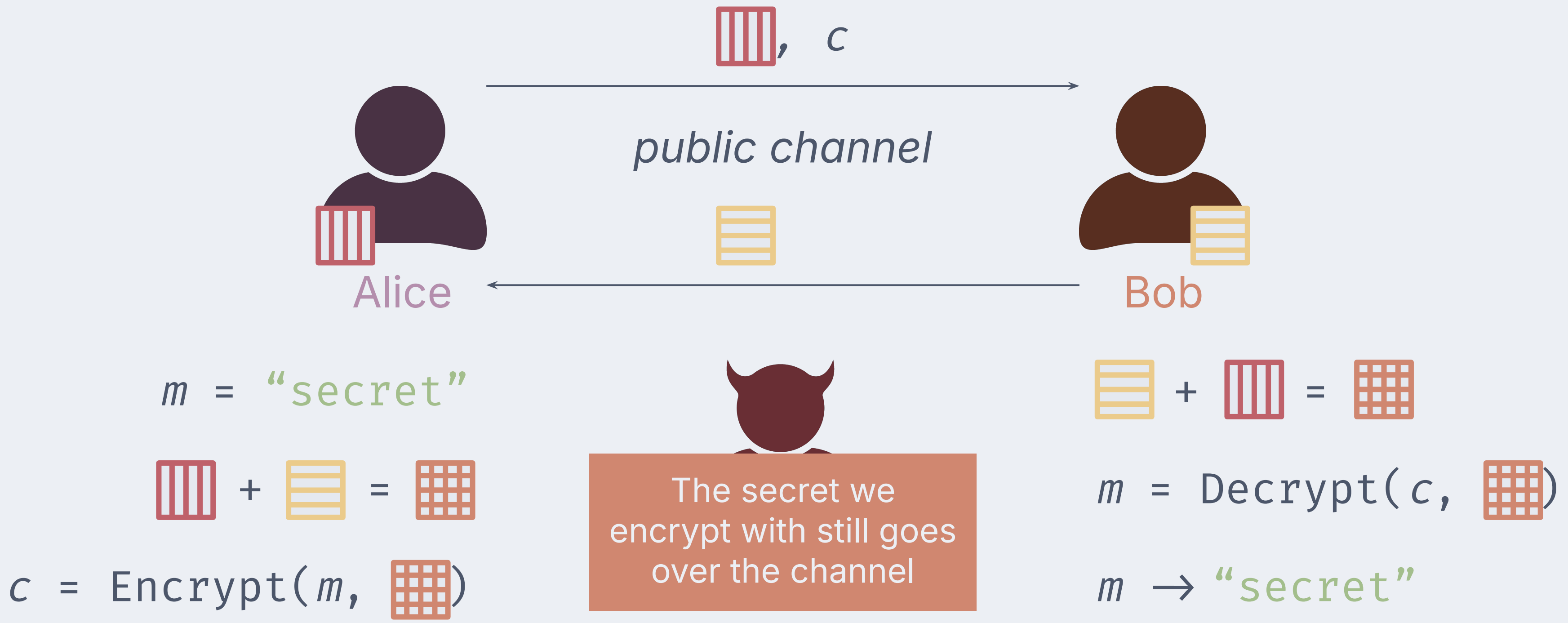
Symmetric encryption





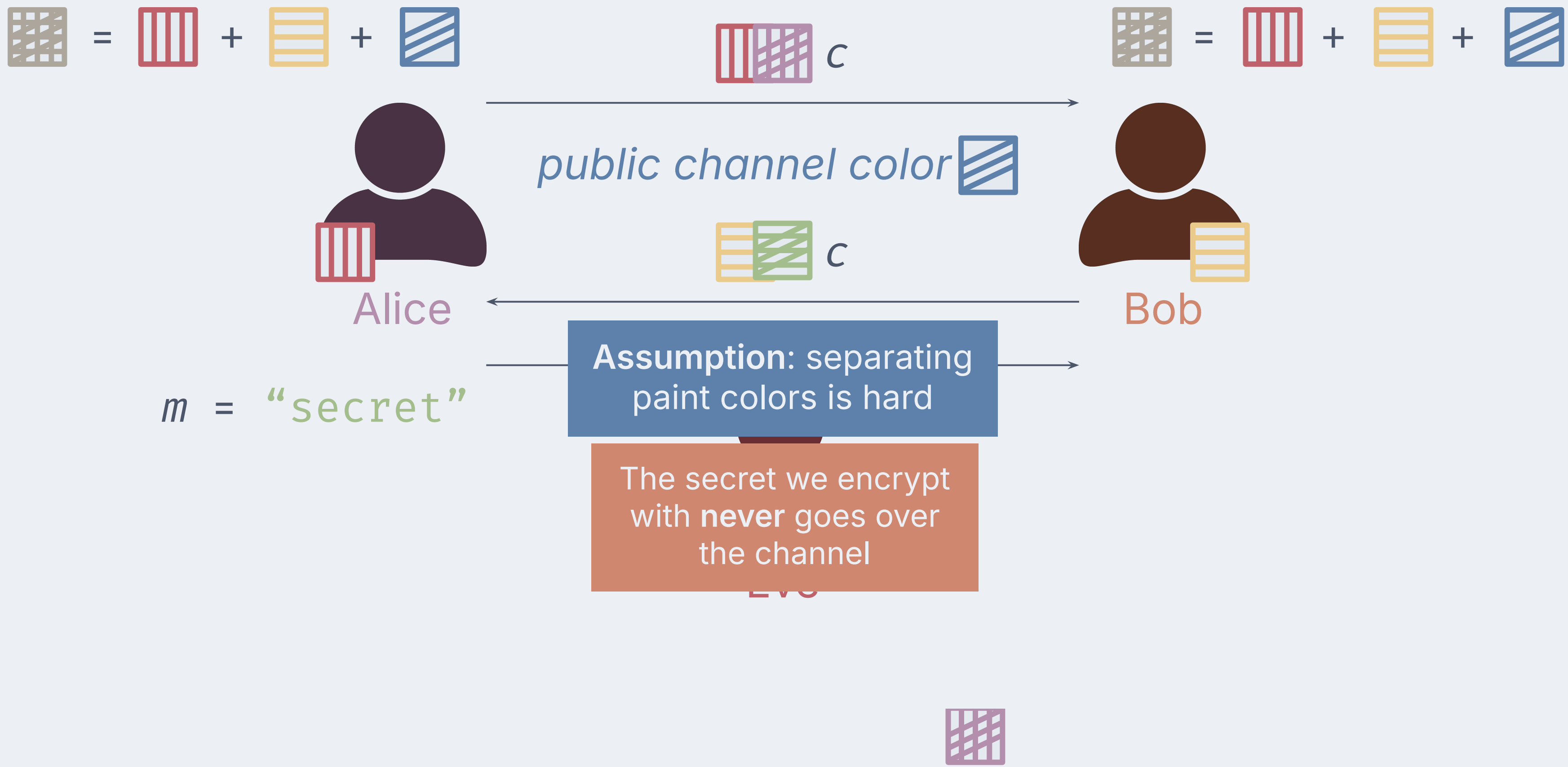
m = "secret"

The secret we encrypt with still goes over the channel

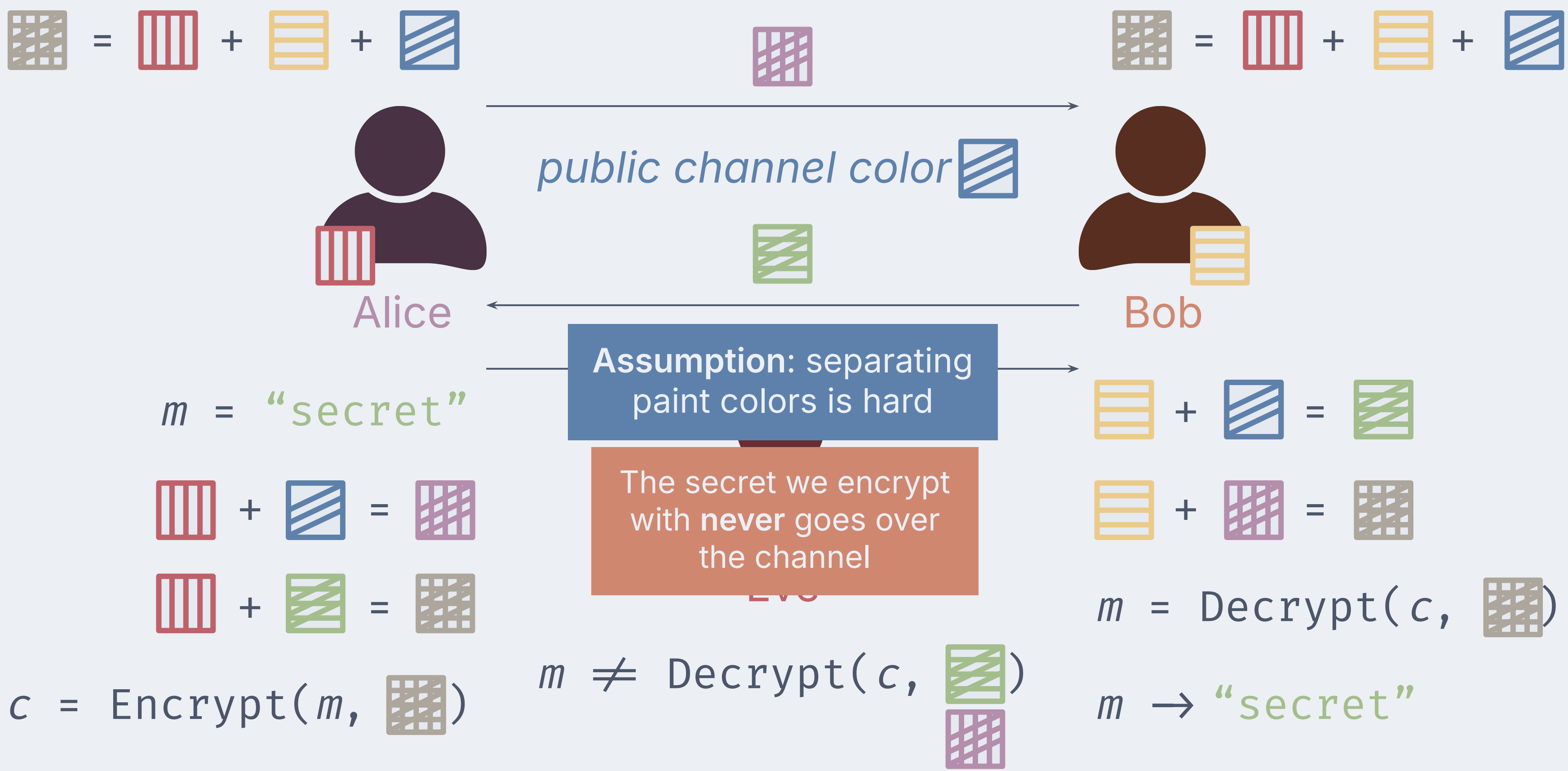


, c
public channel
Alice
Bob
m = “secret”
+ =
c = Encrypt(m,)
The secret we encrypt with still goes over the channel
+ =
m = Decrypt(c,)
m → “secret”

Diffie-Hellman Key Exchange



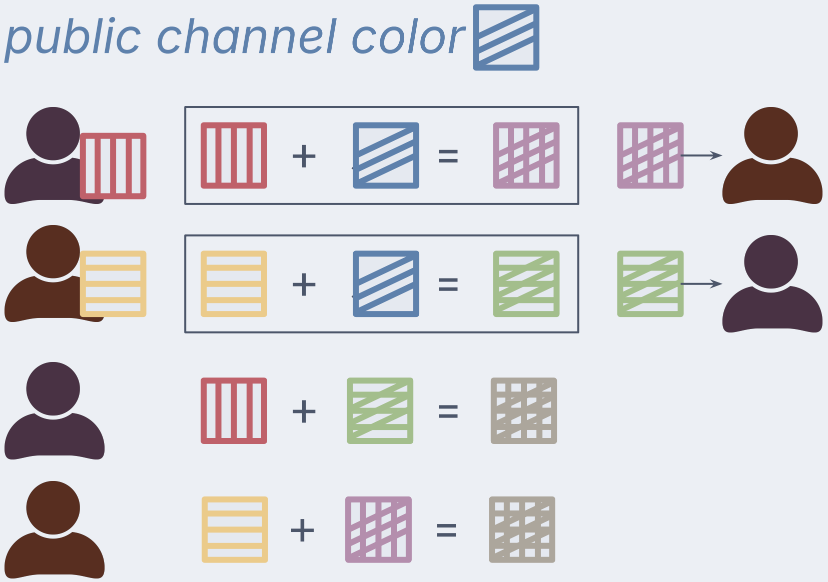
Diffie-Hellman Key Exchange

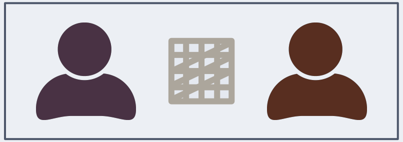


Diffie-Hellman Key Exchange

- Alice and Bob agree to use $p = 23$, $g = 5$
- Alice chooses a secret integer $a = 4$
- Alice sends Bob $A = g^a \bmod p$
  - $A = 5^4 \bmod 23 = 4$
- Bob chooses a secret integer $b = 3$
- Bob sends Alice $B = g^b \bmod p$
  - $B = 5^3 \bmod 23 = 10$
- Alice computes $s = B^a \bmod p$
  - $s = 10^4 \bmod 23 = 18$
- Bob computes $s = A^b \bmod p$
  - $s = 4^3 \bmod 23 = 18$
- Alice and Bob now share a secret 18

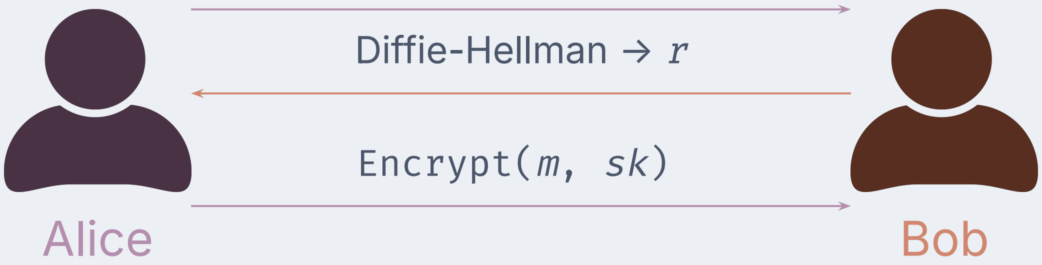
Assumption: the discrete logarithm problem is hard





Assumption: separating paint colors is hard

Basic file encryption



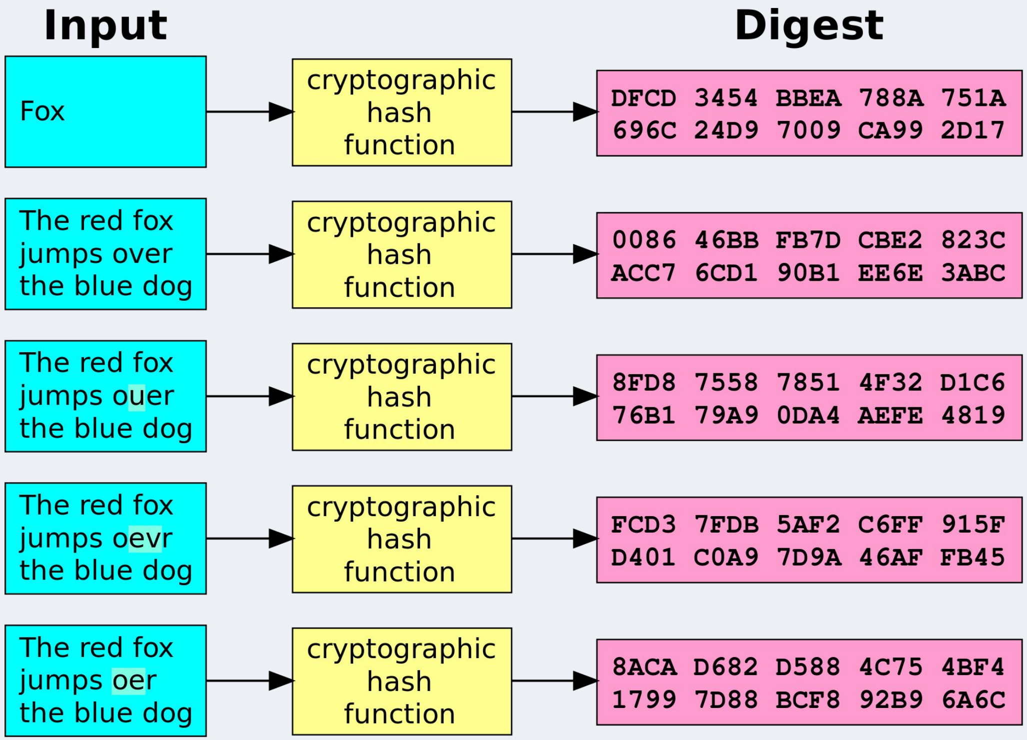
Need to convert Diffie-Hellman result r into encryption key sk

Cryptographic hash function

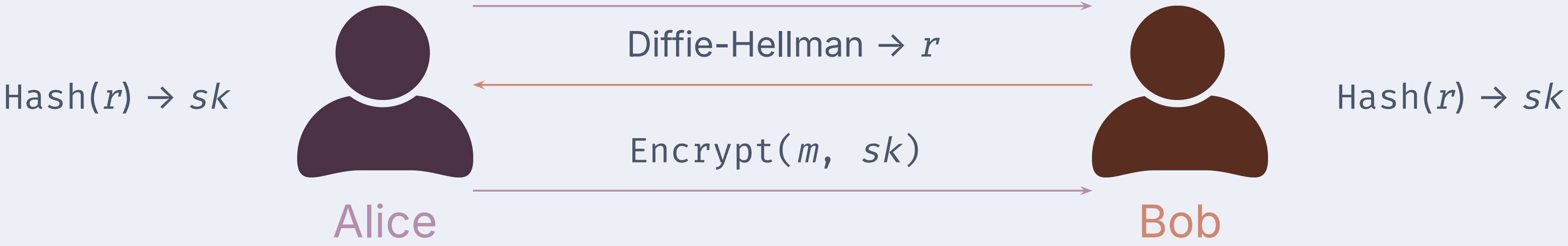
Cryptographic hash function

- **Pre-image resistance:** Given hash value h, it should be difficult to find any message m such that h = $\mathrm{Hash}(m)$
- **Second pre-image resistance:** Given input m_1 , it should be difficult to find a different input m_2 such that $\mathrm{Hash}(m_1) = \mathrm{Hash}(m_2)$
- **Collision resistance:** It should be difficult to find inputs m_1 , m_2 such that $\mathrm{Hash}(m_1) = \mathrm{Hash}(m_2)$

Ideally: the output of the cryptographic hash function **looks random**



Basic file encryption



Need to convert Diffie-Hellman result r into encryption key sk

Cryptographic hash function

Digital signatures

Integrity

Encryption

Confidentiality

Digital signatures

Integrity

- The party who wants to sign generates (sk, pk)
  - sk = “signing key”
  - pk = “verification key”
- The signer keeps sk secret, using it to sign messages
- The signer publishes pk
  - Anyone can use pk to validate the signature on a message
  - The only way that pk can validate a message is if sk signed it **-- unforgeability**

Digital signatures

pk, m, t

public channel

Alice

m = "claim"

t = Sign(m, sk)

Cryptographic hash function

Mallory

t ??

Bob

b = Verify(t, m, pk)

$b \rightarrow$ true || false

Looking ahead

- Crypto is great, but also has a number of failure modes, as we'll see
- Exam 1 is **Oct 9** (three weeks away!)
- Yet another seminar is happening next week! (Sep 26 11am)
- **Today's activity**: Lab 4, Cryptography in Rust

Lesson objectives

- Explain how symmetric encryption and digital signatures work
- Describe the Diffie-Hellman key exchange protocol
- Compose cryptographic primitives to build secure systems