Better living through cryptography

Secure Systems Engineering Spring 2024

🛡️ EE G7701

February 20, 2024

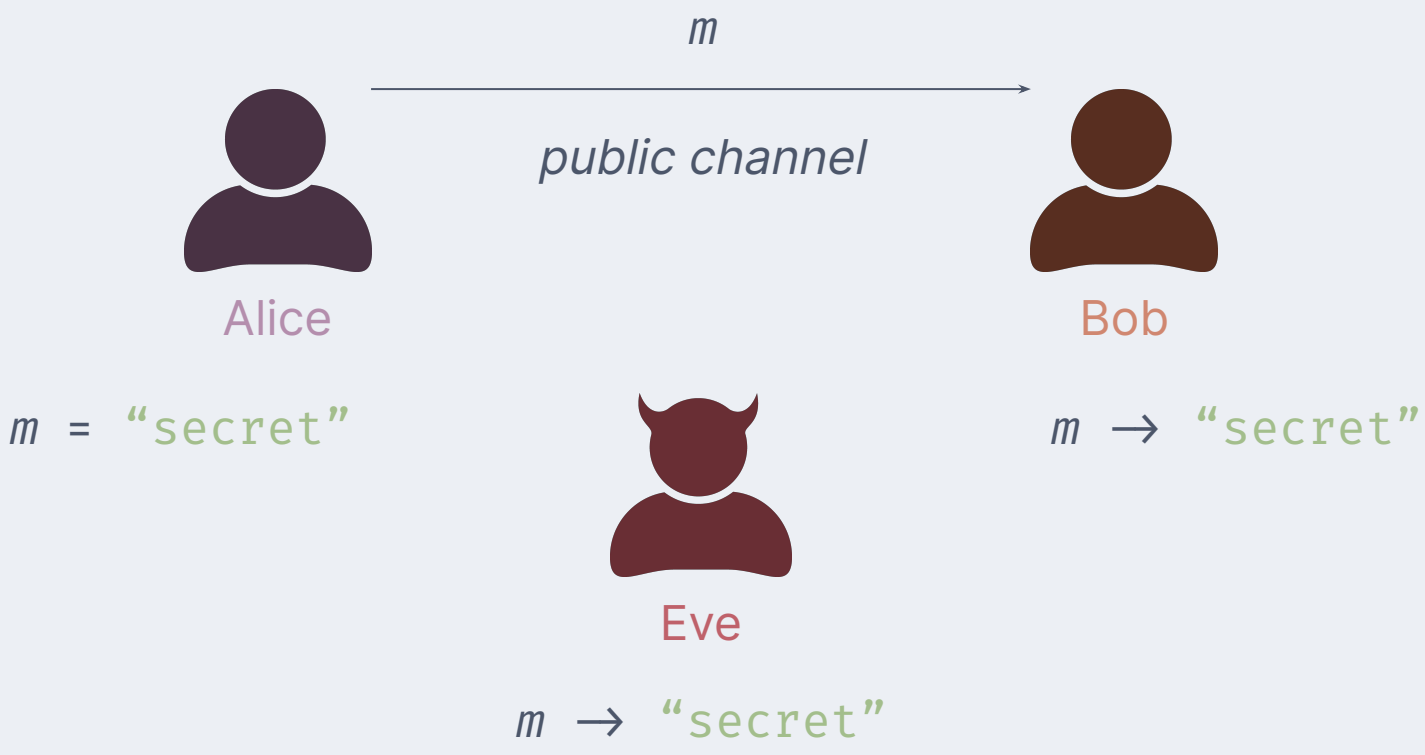
Tushar Jois

Recap

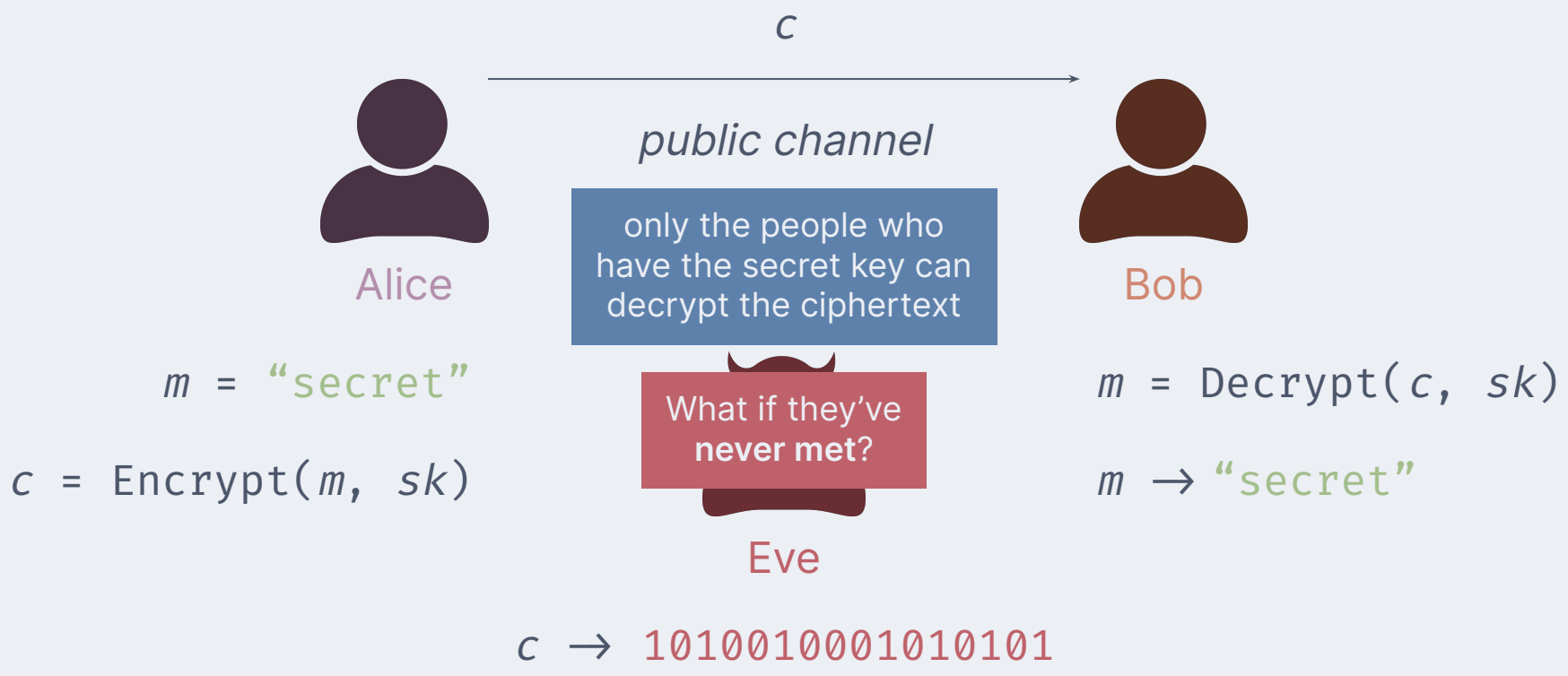
- Rust is a language designed to build reliable and efficient software
- Rust's syntax and features help when writing large codebases
- Ownership and borrowing rules help ensure safety at compile time

Lesson objectives

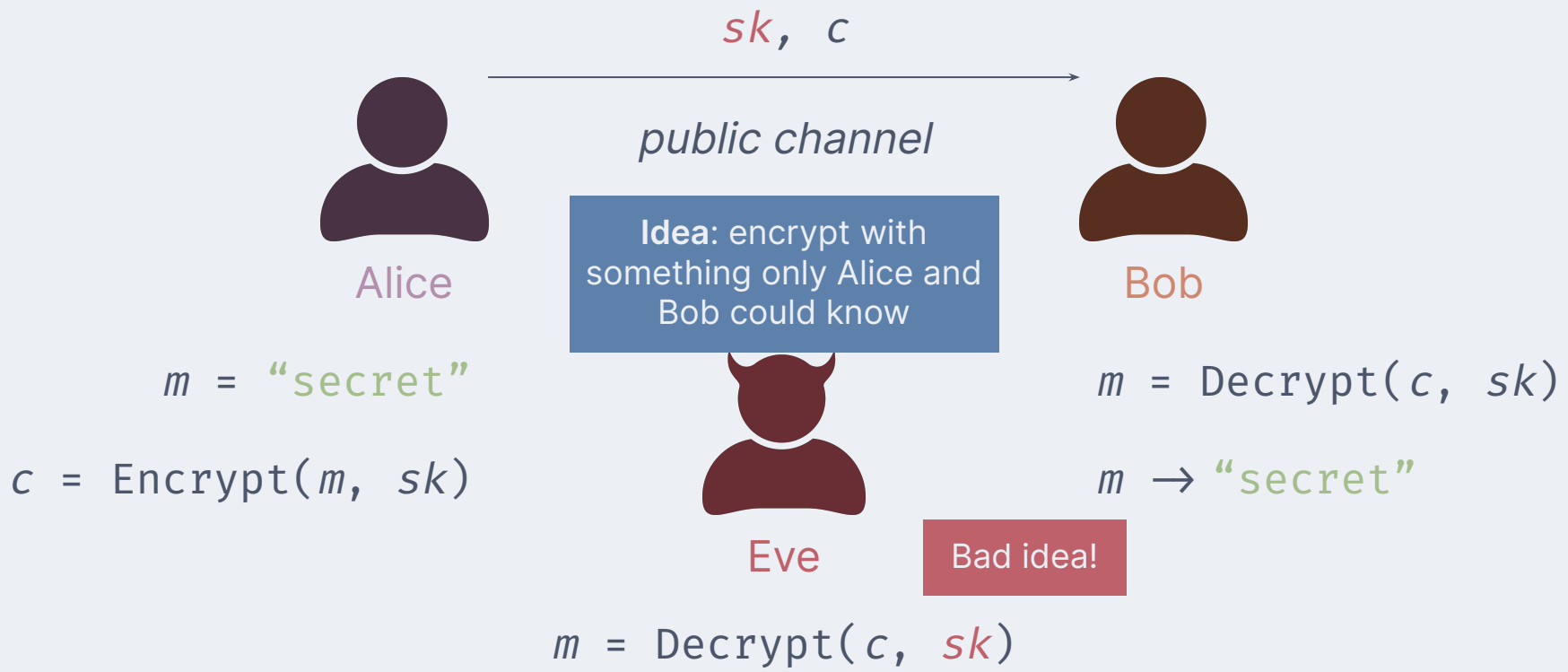
- Explain how symmetric encryption and digital signatures work
- Describe the Diffie-Hellman key exchange protocol
- Compose cryptographic primitives to build secure systems


m
public channel
Alice
Bob
m = “secret”
m → “secret”
Eve
m → “secret”

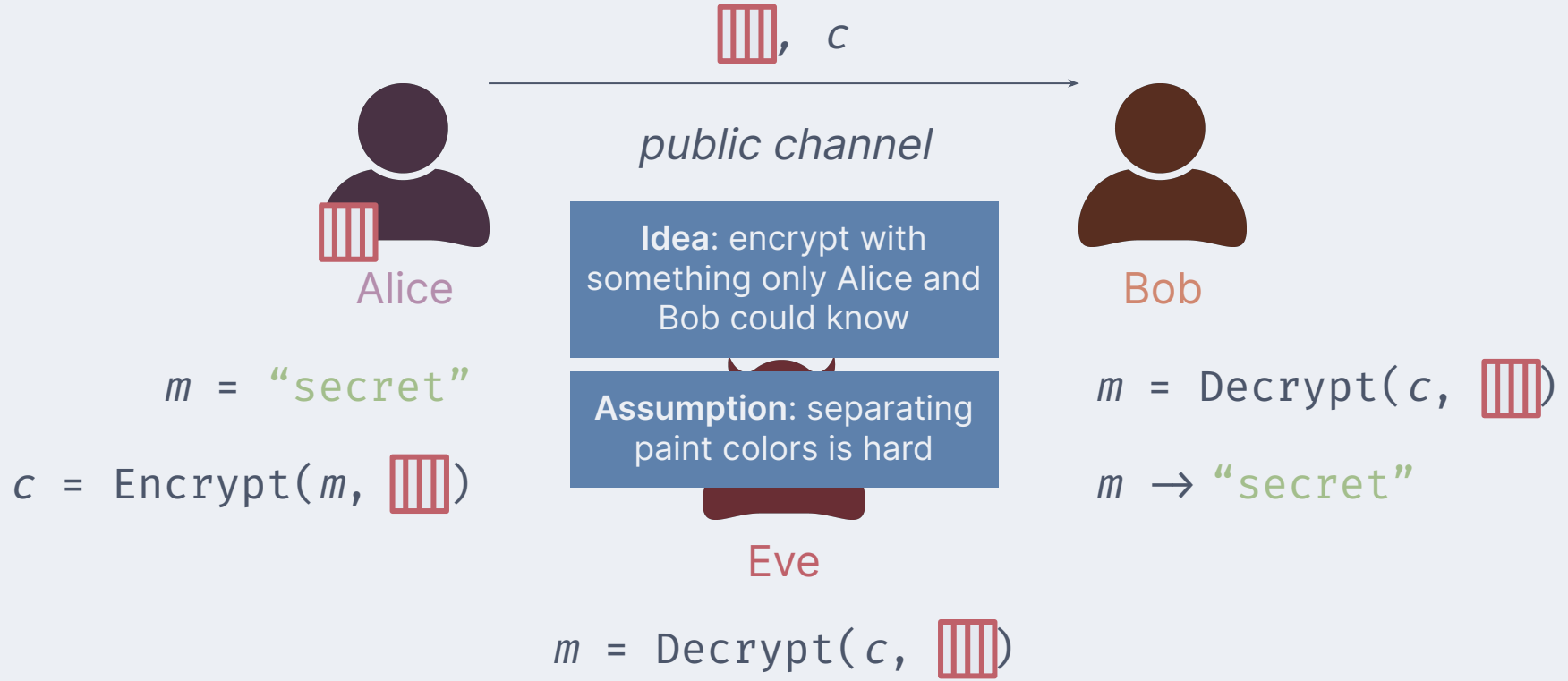
Symmetric encryption



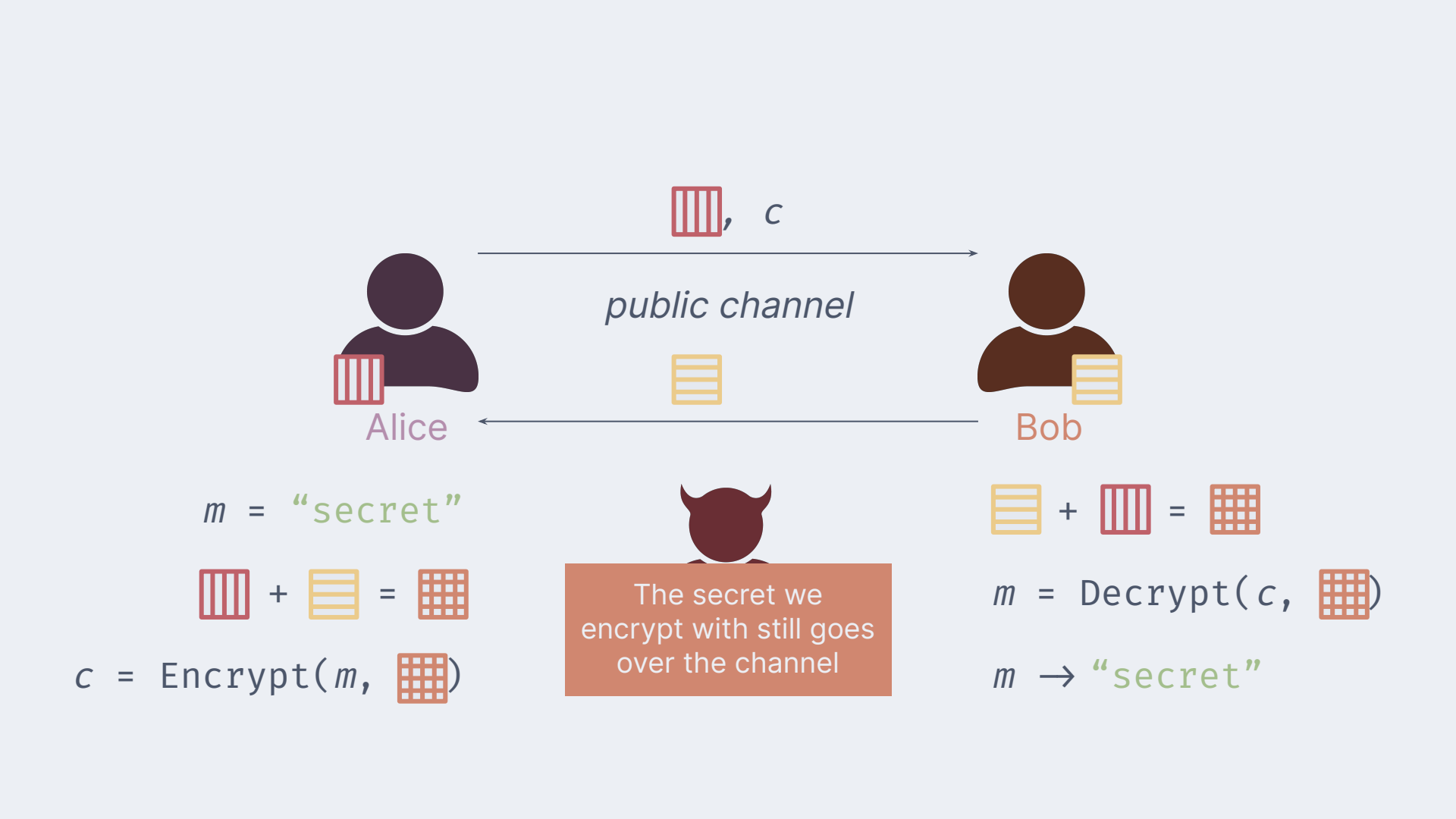
Symmetric encryption



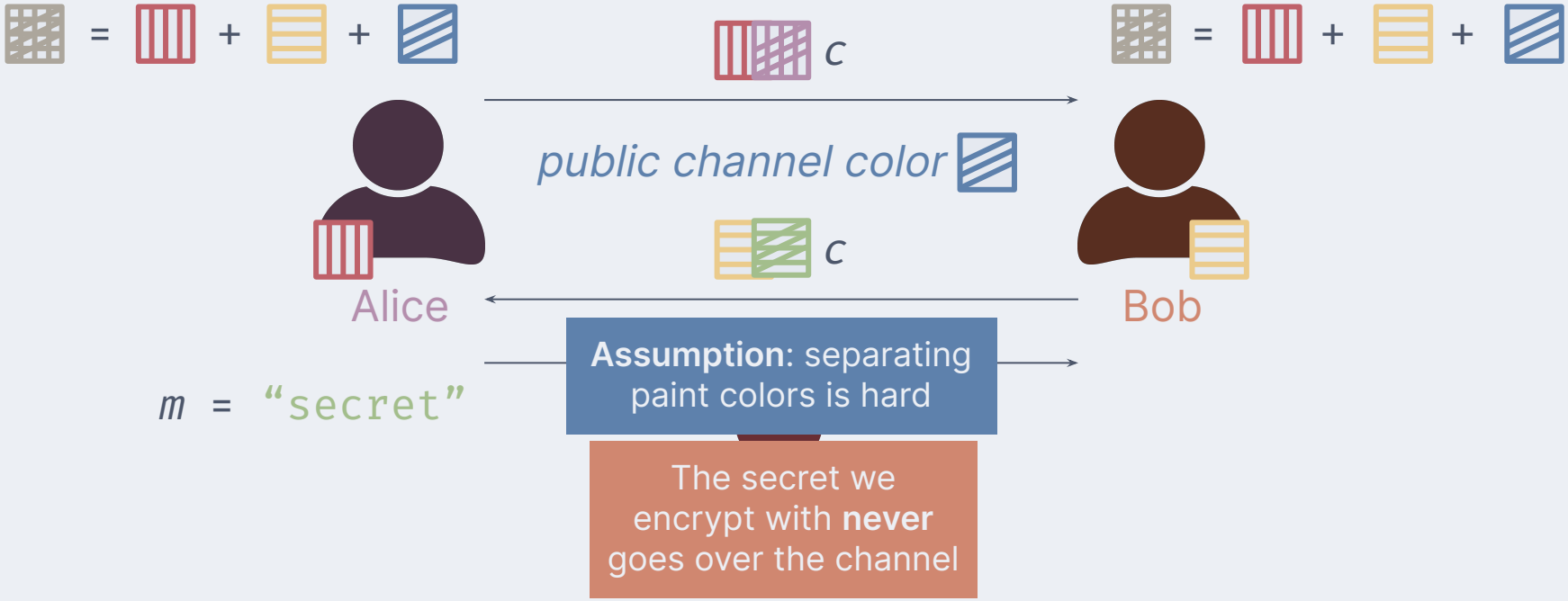
Symmetric encryption



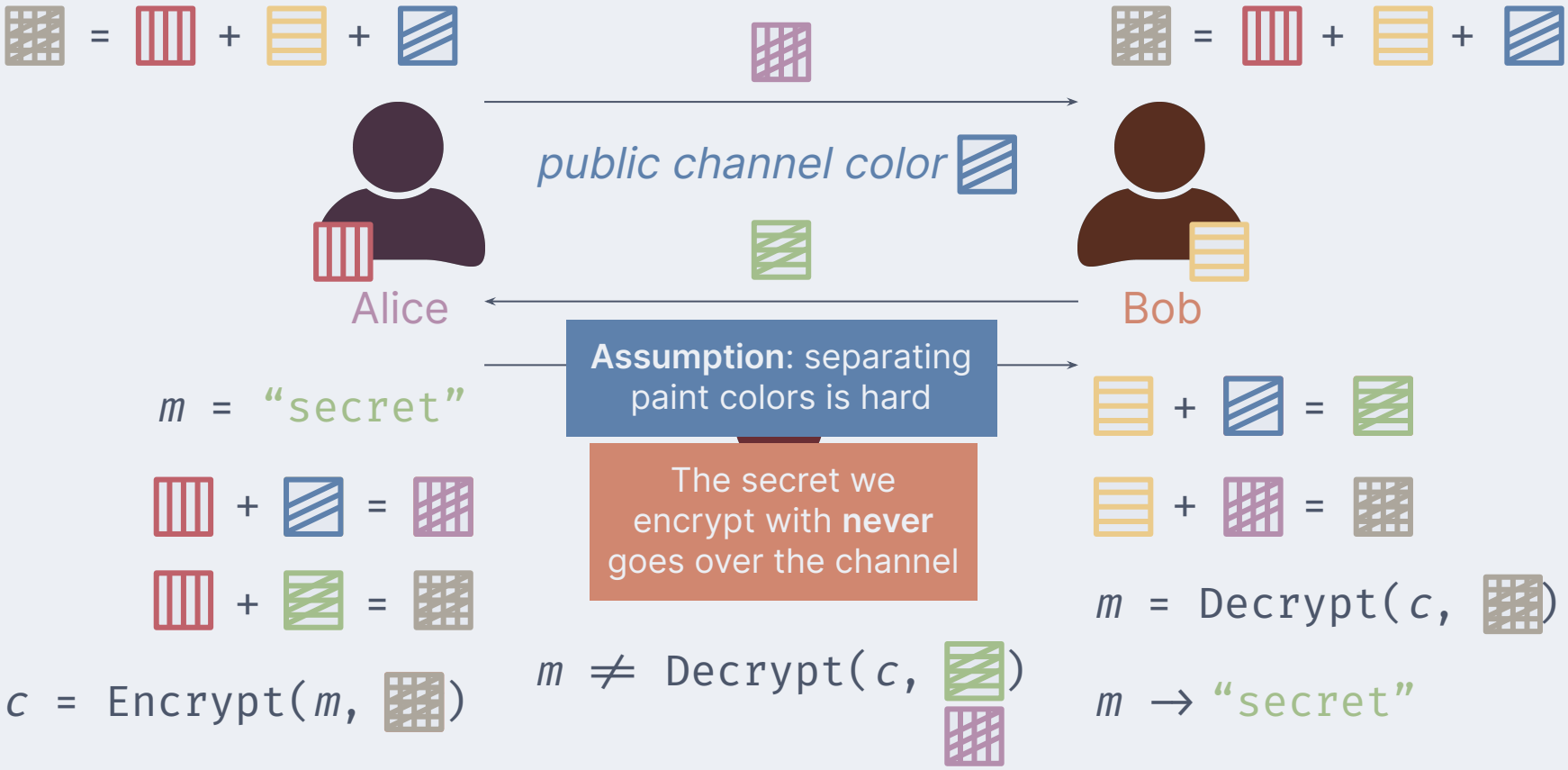

, c
public channel
Alice
Bob
m = “secret”
The secret we encrypt with still goes over the channel


, 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



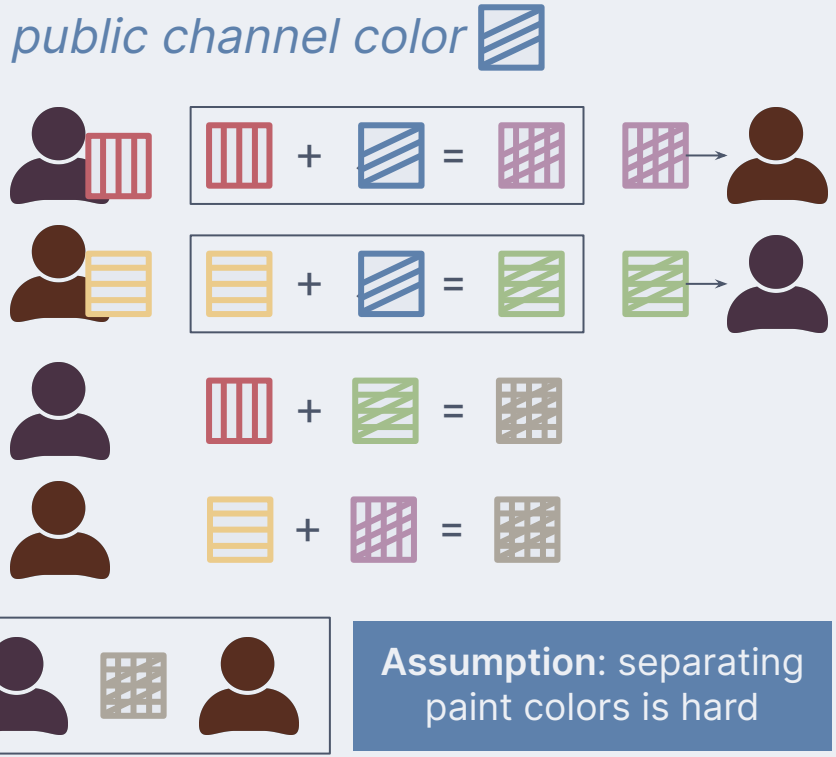
Diffie-Hellman Key Exchange



Diffie-Hellman Key Exchange

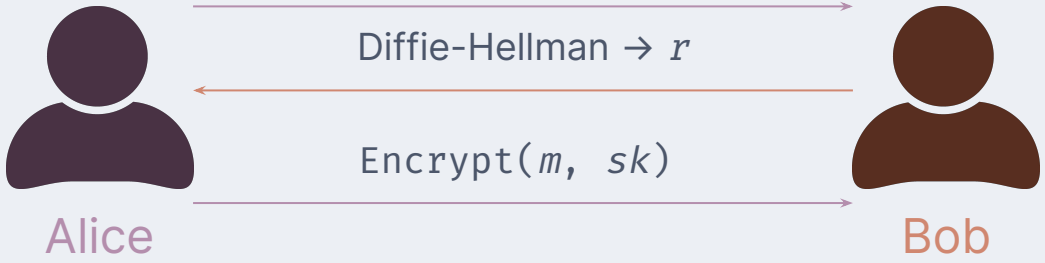
- Alice and Bob agree to use $p = 23$, $g = 5$
- Alice chooses a secret integer $a = 4$
- Alice sends Bob $A = g^a \bmod p$
  - $A = 5^4 \bmod 23 = 4$
- Bob chooses a secret integer $b = 3$
- Bob sends Alice $B = g^b \bmod p$
  - $B = 5^3 \bmod 23 = 10$
- Alice computes $s = B^a \bmod p$
  - $s = 10^4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



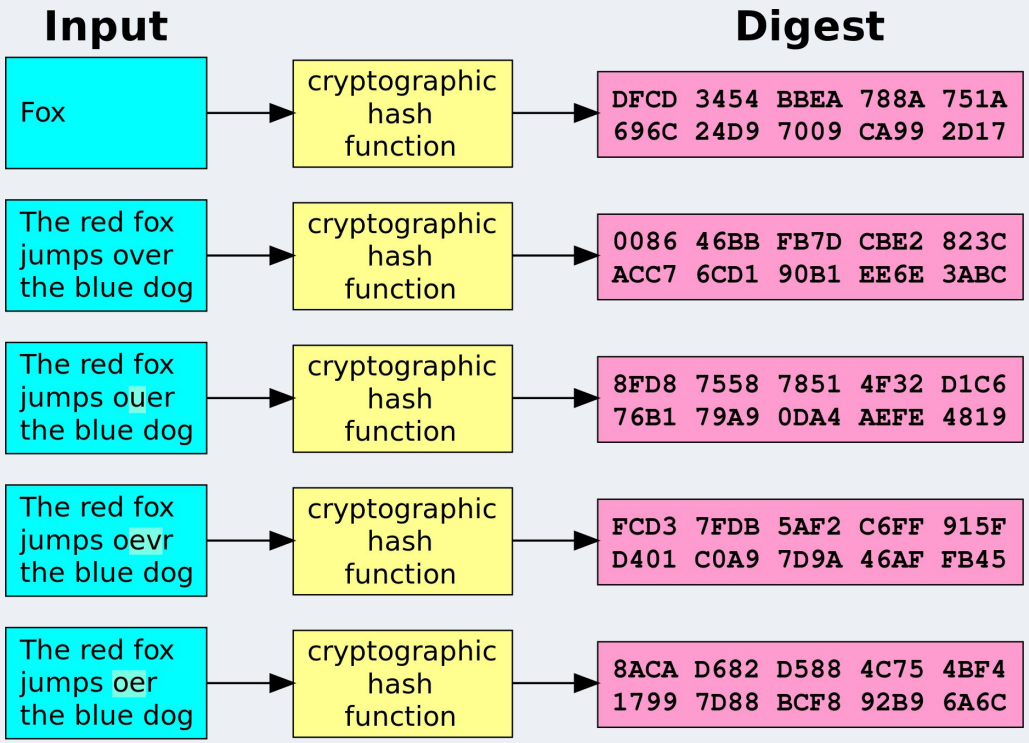
Need to convert Diffie-Hellman result r into encryption key sk

Cryptographic hash function

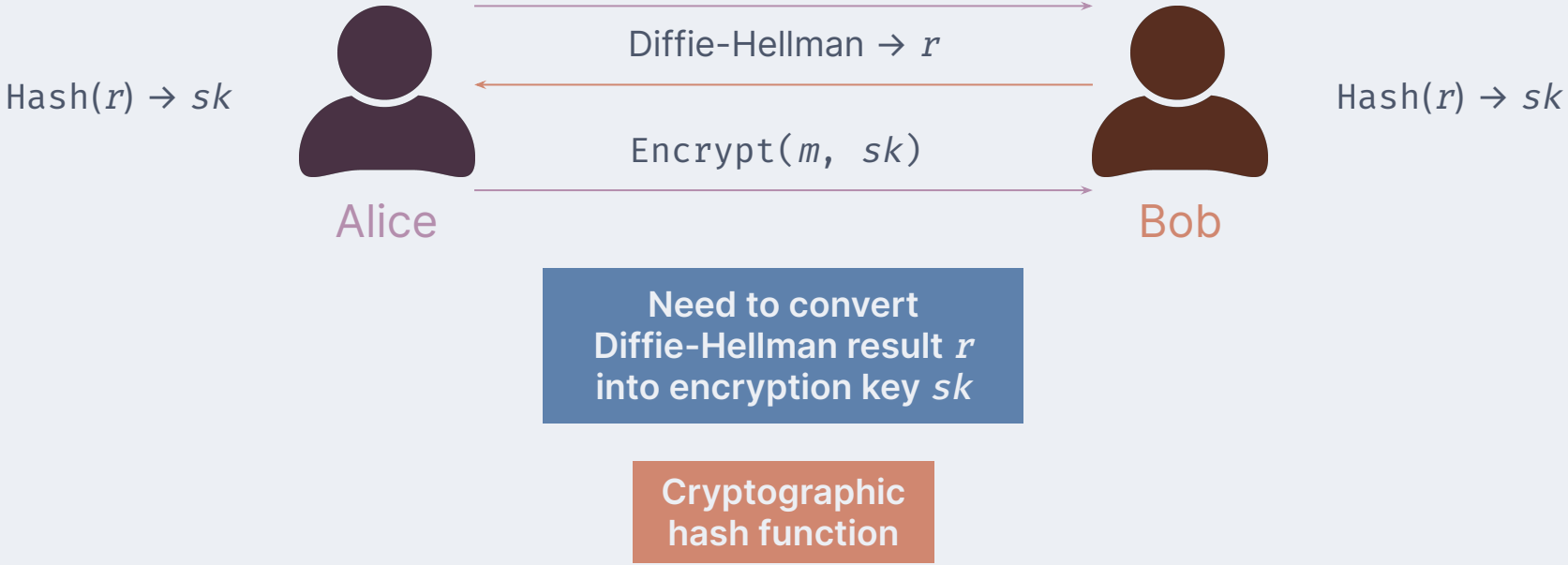
Cryptographic hash function

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

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



Basic file encryption



Digital signatures

Integrity

Encryption

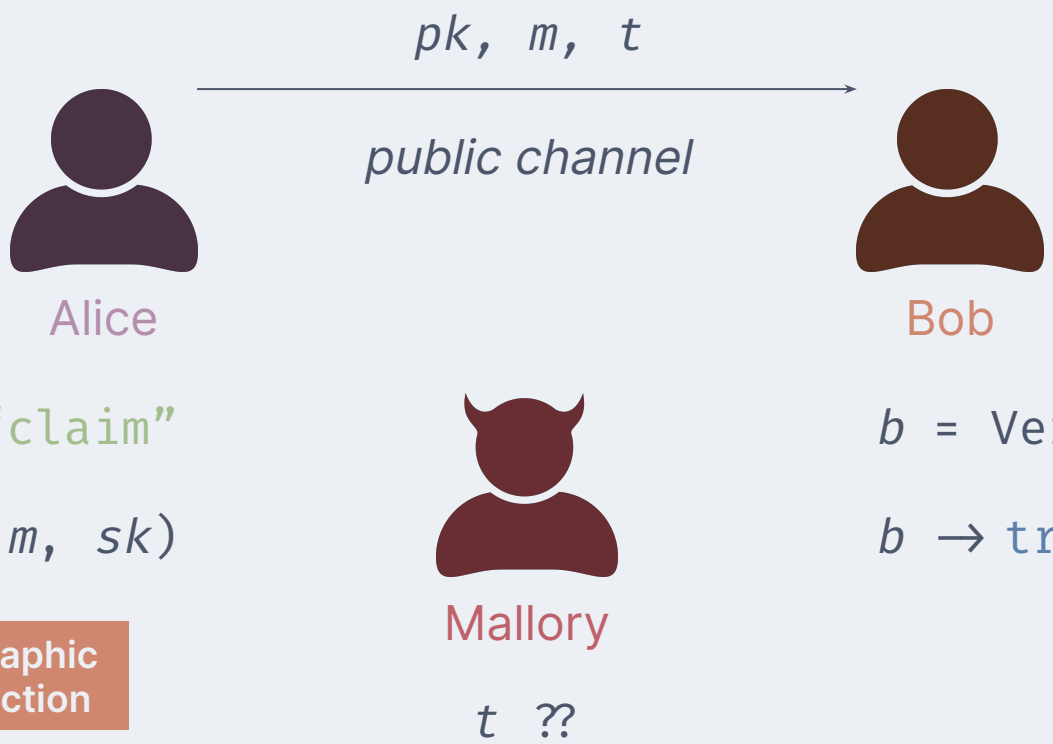
Confidentiality

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



Looking ahead

- Crypto is great, but also has a number of failure modes, as we'll see
- Assignment 2 is up and due **10p Mar 4**
  - Working through your Assignment 1 grades now
  - Office hours will be moving to **Thu 4:30p**
- Exam 1 is **Mar 5** (two weeks away!)
- Cybersecurity capture-the-flag competition team!
- **Today's activity**: more Rust practice
  - Pair programming with your Assignment 2 partner (different from Assignment 1)

Lesson objectives

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