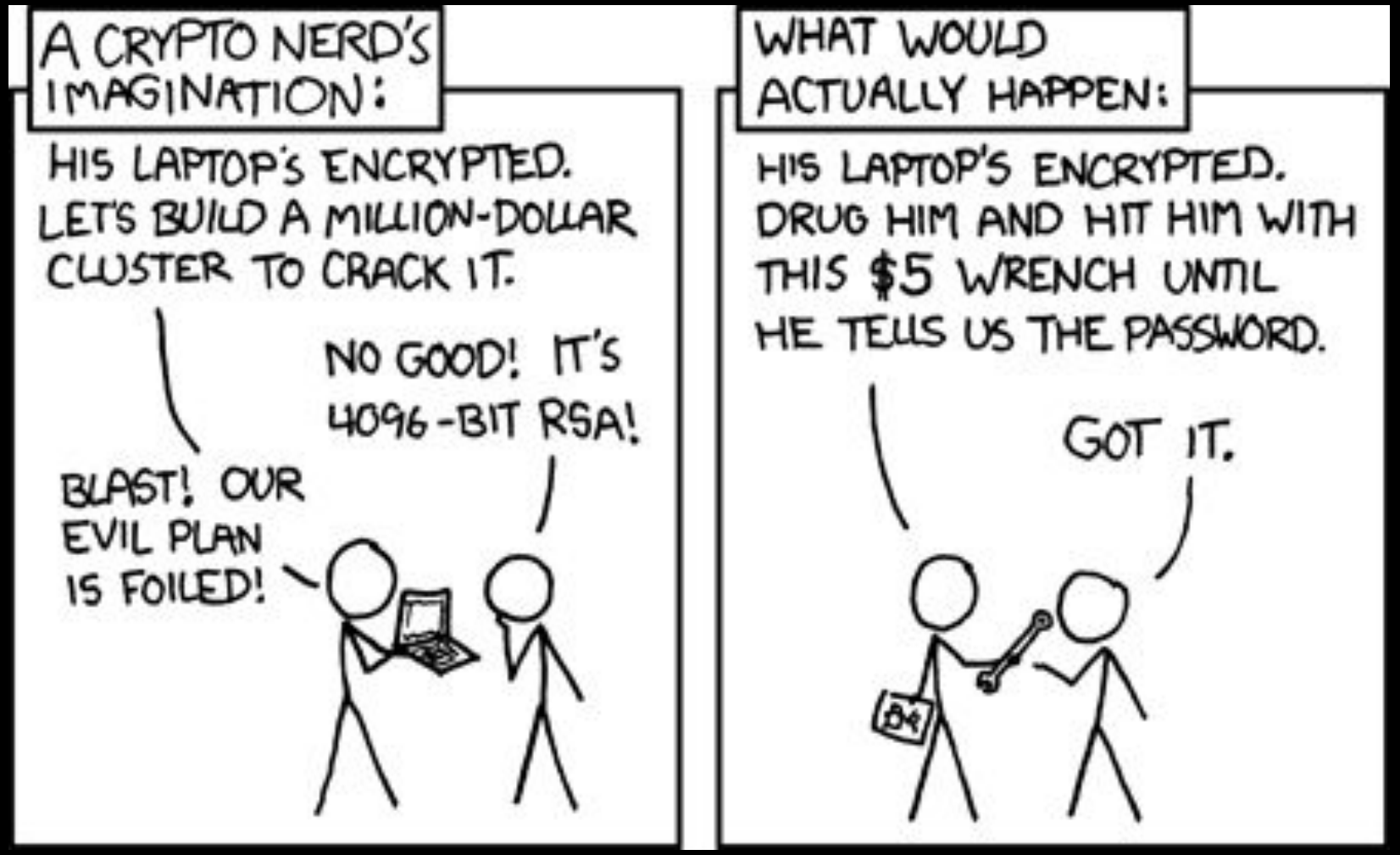
Week 03
Crypto I

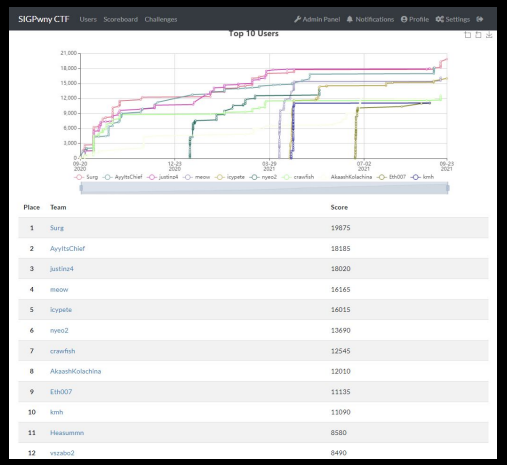
Husnain, Anakin, Hassam, Pranav, Nebu

sigpwny{rot13_twice_extra_secure}



Announcements

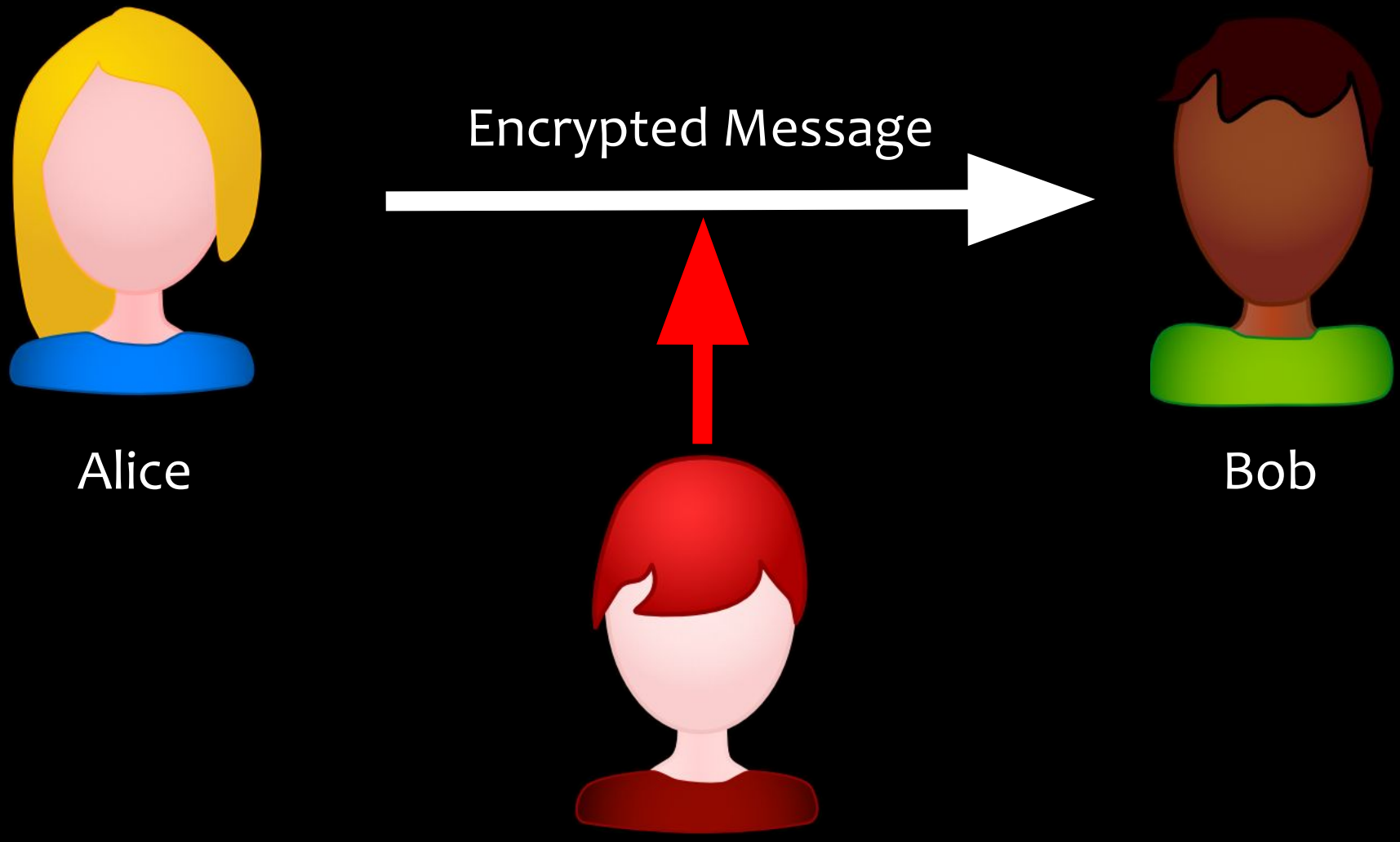
Scoreboard reset!



Weekend LAN Party

Feedback form!
https://forms.gle/z5kWDCcWcoKbixhT9 (scan QR Code ^^^)

What is crypto anyway?



Applications

https://

>_SSH

Encodings

- TL;DR - computers store things in binary (0s and 1s) , and we have different ways of representing this
- Tip: In Python, always work with bytestrings, never with the normal string types (get on modern python!!! 3.8+)

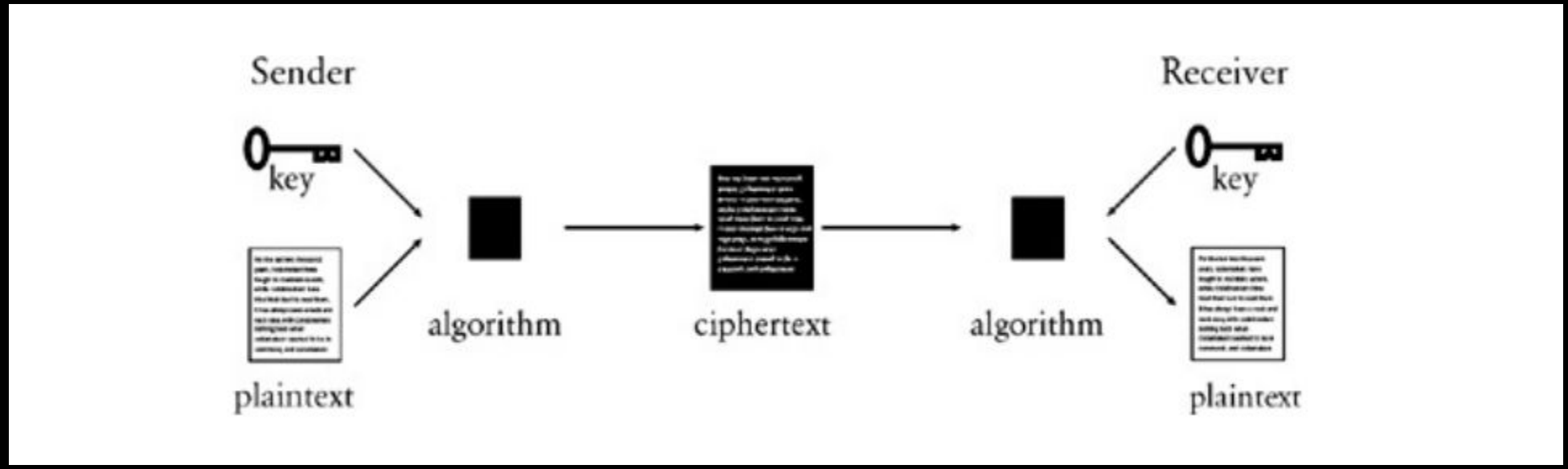
Format	Description	From Bytes	To Bytes
base64	uses printable letters to encode more complex binary	**base64.b64encode**	**base64.b64decode**
hex	uses symbols 0-9, A-F	**bytearray.hex()**, binascii.hexlify()	**bytes.fromhex()**, binascii.unhexlify()
integer	normal integers	**Crypto.Util.number.bytes_to_long (from PyCryptoDome)**, int.from_bytes	**Crypto.Util.number.long_to_bytes (from PyCryptoDome)**, int.to_bytes

“Ancient” Ciphers

- Caesar Cipher: Shifting letters over by some number (rot k):
  - a -> c, b -> d, … , y -> a, z -> b (rot 2)
- Substitution Cipher:
  - Create a table mapping each letter to another
  - To crack: use frequency analysis
- Vigenere Cipher:
  - like Caesar Cipher, but each character is shifted by a keyphrase, rather than just one number

These are **not secure**, can easily be solved or brute forced. In recent years, “easy” crypto challenges have moved past them.

Symmetric Encryption

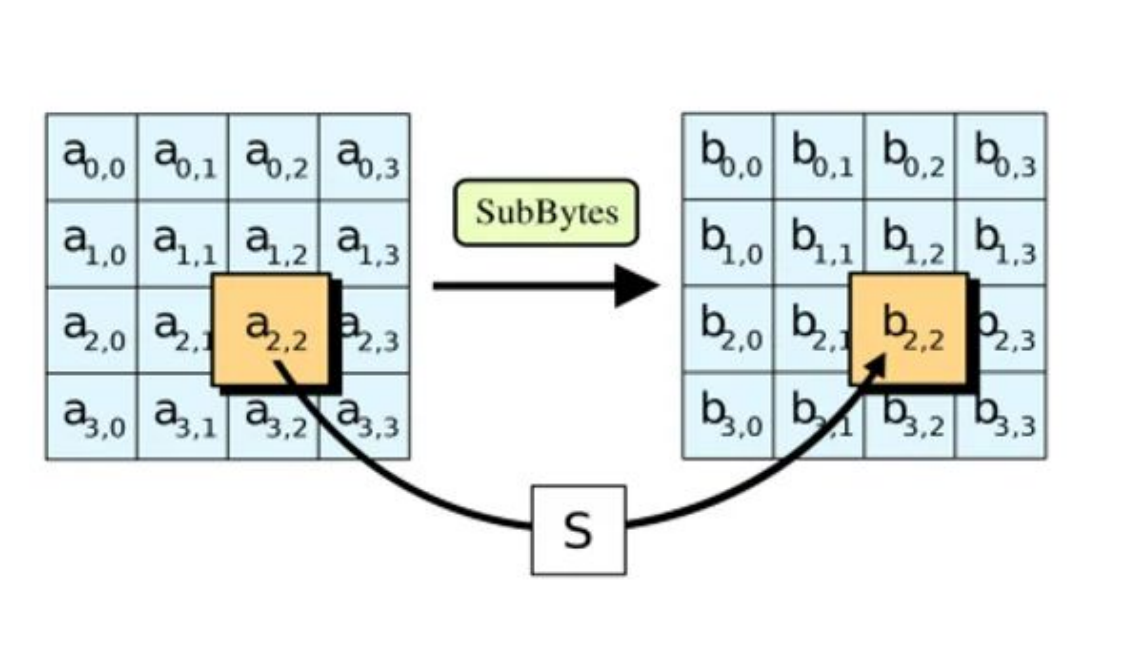


XOR Encryption

A	B	A XOR B
1	1	0
1	0	1
0	1	1
0	0	0

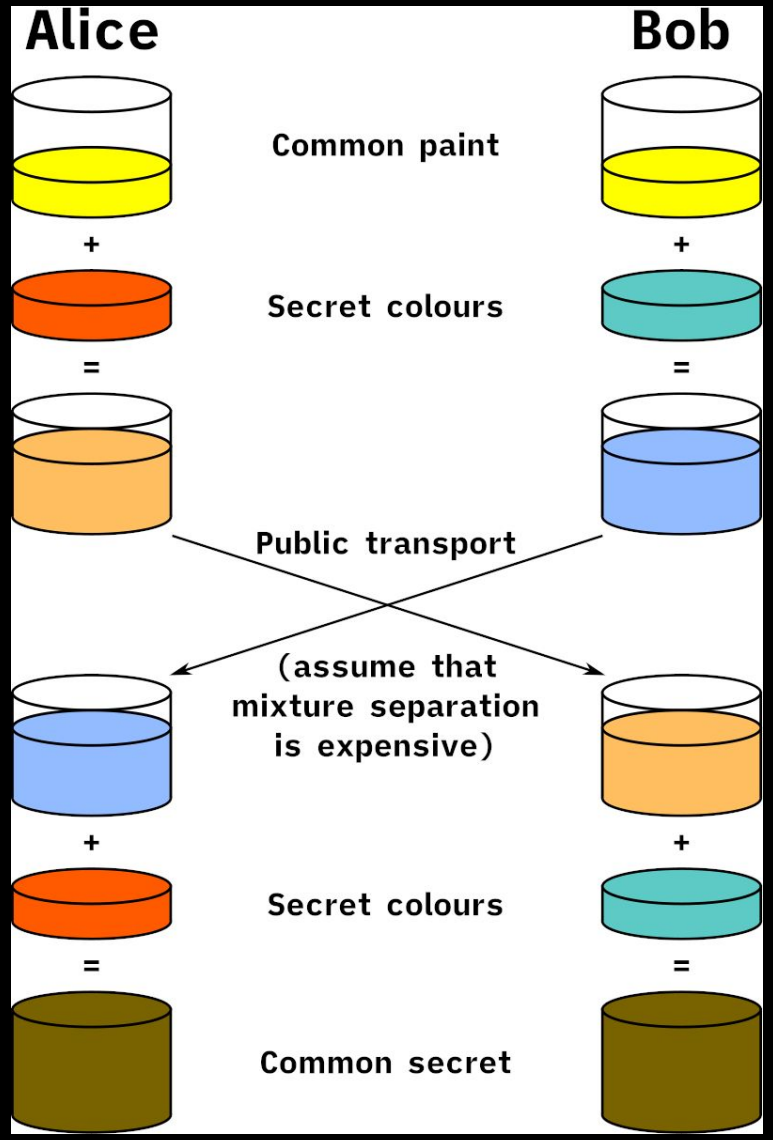
- XOR is the inverse of itself -> perfect for symmetric encryption
- Has nice probabilistic properties (ask us for short proof later)
- Fundamental operation in nearly all symmetric encryption

AES

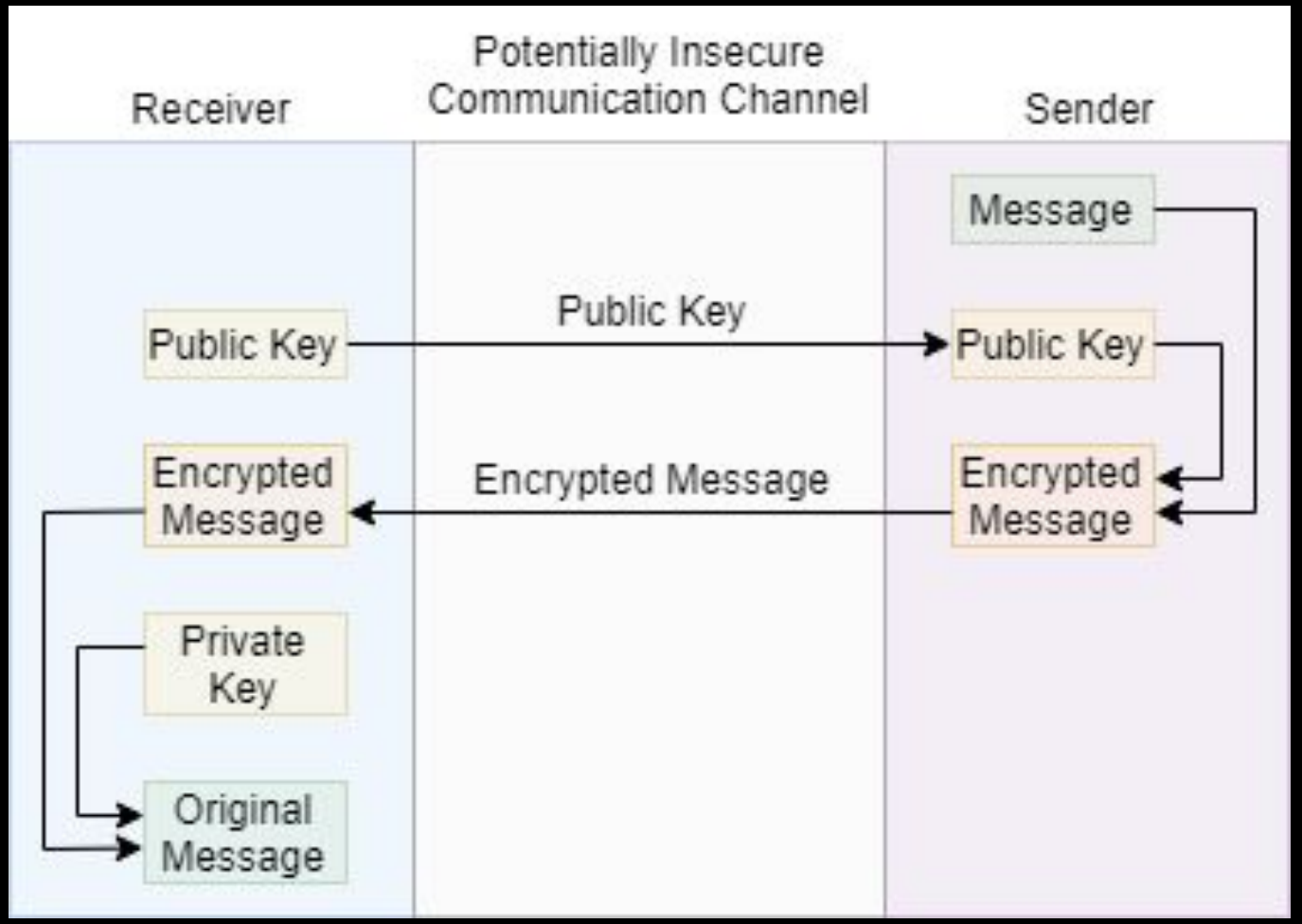


- XOR requires that your key is the same length as the message
- AES is a modern symmetric encryption standard that uses XOR internally

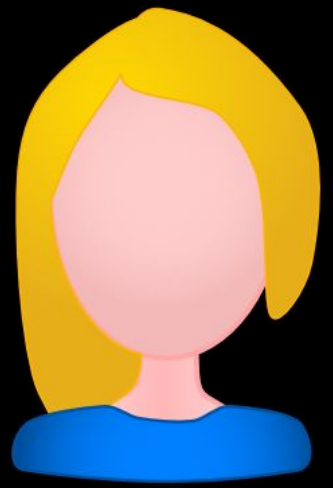
Diffie-Hellman Key Exchange



Asymmetric Encryption



RSA: Intuition



Alice

Bob

RSA

- Fundamentally based on asymmetry of the factoring problem:
  - multiplying two numbers is easy, factoring a number is hard
  - “Cracking” RSA means factoring a large product of primes p and q
- Bigger n = more secure and hard to find p and q
- Small n (<512 bits) can be brute forced
  - See: factordb, Wolfram Alpha, SageMath factorization algorithm
  - https://en.wikipedia.org/wiki/Texas_Instruments_signing_key_controversy
- Try challenge: Easy RSA -- Look at Wikipedia
- What to do when n is big?
  - ~~Cry~~ Come back Sunday for attacks on bigger n

RSA: Asymmetric Encryption

- $n = p*q$
- $\Phi(n) = (p-1)*(q-1)$
- e = usually 65537, coprime to Φ **(small e is usually breakable)**
- d = inverse mod(e, Φ(n))
  - In Python 3.8+: pow(e, -1, Φ(n))

- **Public key: (n, e)**
- **Private key: (p, q, e)**
- Encrypting: $c \equiv m^e \pmod{n}$
- Decrypting: $m \equiv c^d \pmod{n}$

Tools!

- SageMath is your friend, especially for CTF challenges (but not necessary for today)
- PwnTools
- PyCryptodome
- Google + StackOverflow (“how to crack RSA”)

- Installing these all is annoying: Get Docker + CryptoHack container

We will go over installation of common crypto libraries on Sunday!

Cryptohack!!

Next Meetings

Weekend Seminar: Crypto 2

- Frequency Analysis
- Block Cipher
- ECC (Elliptic Curve Cryptography)
- ????

Next Thursday: Intro to Binary Exploitation (PWN)

- Stack overflows
- Memory attacks
- The history of binary exploitation!
  - The back and forth

Demo for funsies