SIGPwny

FA2024 Week 08 • 2024-10-24

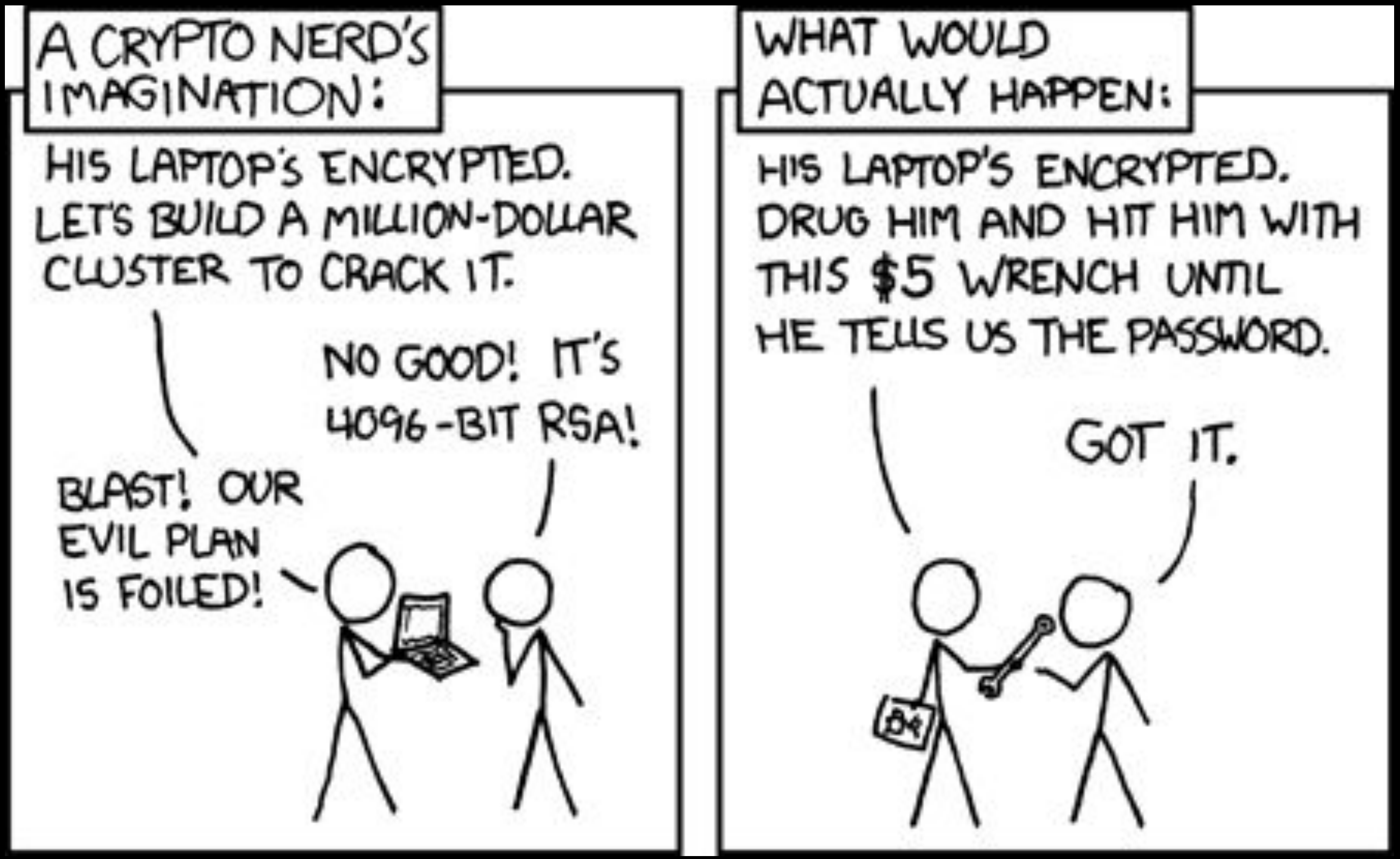
Cryptography I

Nikhil and George

Announcements

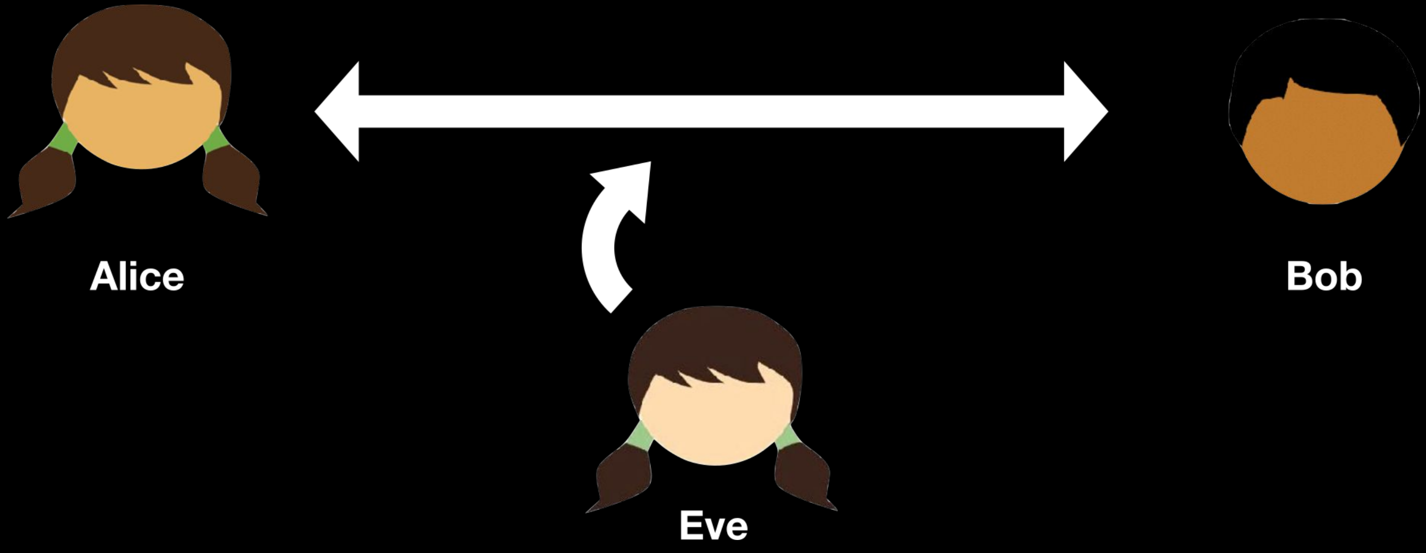
ctf.sigpwny.com

sigpwny{r3v3r51n6_bu7_m47h}


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

What is Cryptography all about?

- Secure communication between 2+ parties (Alice, Bob)



Consequences of bad cryptography

- Mary Queen of Scots executed for conspiring to kill Queen Elizabeth I (Babbington Plot)
- Vulnerabilities in OpenSSH (e.g. CVE-2008-0166) give an attacker a free shell on your system

Then vs. now

- Cryptanalysis done manually by spymasters, generally very targeted (e.g. military use)
  - Schemes were secure until they weren’t
- Current day: your computer send millions of encrypted packets to tens of thousands of hosts
- We need schemes predicated on computational hardness assumptions (if these assumptions hold, this scheme is secure to these categories of attacks)

XOR

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

A.k.a. addition mod 2

Associative, commutative, self-inverse

Data Representation

```
>>> from Crypto.Util.number import long_to_bytes
>>> long_to_bytes(0xdeadbeef) # integer
b'\xde\xad\xbe\xef'
>>> base64.b64decode(b'3q2+7w==') # base64
b'\xde\xad\xbe\xef'
>>> bytes.fromhex("deadbeef") # hex string
b'\xde\xad\xbe\xef'
```

Substitution ciphers

- Caesar Cipher (a.k.a. `rot13`, hint for Vim users: `:h g?`)
    - Add 13 to every letter in the alphabet (with wraparound)
    - Ex. CAESAR -> PNRFNE
- Generally, any function that maps each letter to another letter
- **Insecure!!** Why?
- Cryptanalysis
    - Frequency analysis
    - Known plaintext (cribs): “Keine besonderen Ereignisse” (nothing to report)

The one-time pad

```
>>> plain = b"Test"
>>> cipher = bytes.fromhex("cafebabe")
>>> bytes([i ^ j for i, j in zip(cipher, plain)])
b'\x9e\x9b\xc9\xca'
```

The one-time pad

- Achieves “perfect secrecy”! 🥳
    - …but at what cost?
- Requires a completely random bitstring the same length of your plaintext
    - Not only does this double the message size, but how do you agree on this shared secret?
    - Pseudorandom generators can “stretch” a little bit of randomness into a lot of randomness
    - Stay tuned for AES in crypto III…

Symmetric Encryption

(Asymmetric encryption, e.g. RSA, in Cryptography II)

Dec(Enc(plaintext, key), key) == plaintext

vs.

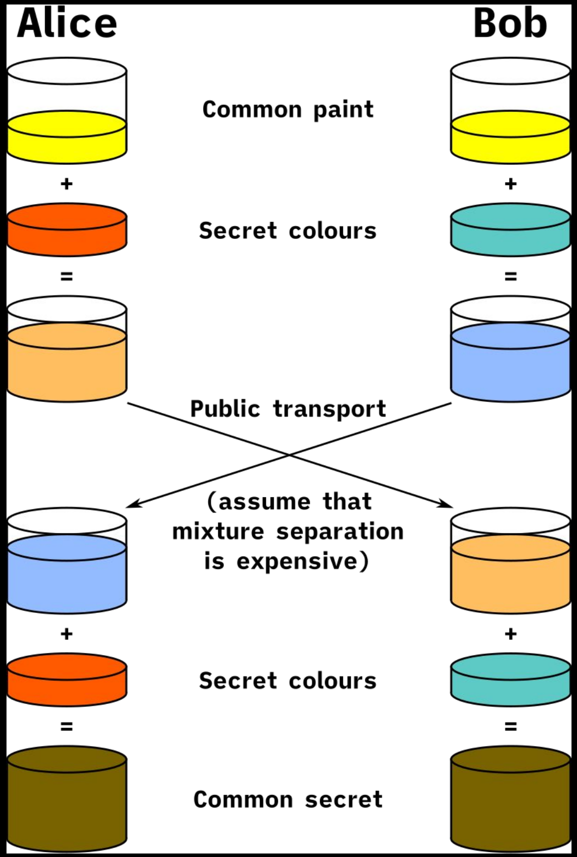
Dec(Enc(ciphertext, public key), private key) == plaintext

Computational hardness

- We cannot actually prove that these are hard, but they are strongly believed to be hard
  - This assumption turns out to be false for quantum computers, which is why people want to build quantum computers
- Discrete log/factoring problem
  - $a^b \equiv X \bmod p$
  - Exponentiation is easy, logarithms are hard

Diffie-Hellman

- Alice and Bob arrive at a shared secret using their private secrets
- All communication happens over a public channel
- Modern implementations perform computations over elliptic curves (ECDH)



Tools

- Pen and paper
- Wikipedia
- Stack Exchange
- SageMath, PyCryptodome, pwntools

```python
from sage.all import *
from pwn import *

conn = remote('localhost', 1337)

a = int(conn.recvline()[3:].decode('utf-8'))
b = int(conn.recvline()[3:].decode('utf-8'))
sol = a.powermod(b, p)

conn.recvuntil(b'c = ')
conn.sendline(str(int(sol)).encode('utf-8'))
print(conn.recvline())
```

Food for thought

- How to establish a shared secret? (RSA)
- How does Alice know she’s really talking to Bob? (digital certificates, web of trust)
- If you take one thing away from this meeting: **never roll your own crypto!**

CryptoHack

Learn with fantastic lessons and challenges, and earn points on PwnyCTF while you're at it!

ctf.sigpwny.com/challenges#Meetings/CryptoHack

Challenges

- Start with First XOR, flag_format (both XOR-based) and Vigenère Visionary
- Diffie-Hellman god has you do the Diffie-Hellman shared secret computation (look at Wikipedia for implementation details)
- First AES and Add One are based on the “Advanced Encryption Standard (AES)” block cipher
- Totient Turmoil and Easy RSA involve RSA (will be covered this Sunday)

Next Meetings

2024-10-27 • This Sunday

- Cryptography II (RSA) with Richard and Emma

2024-10-31 • Next Thursday

- Halloween 👻

2024-10-31 • Next Sunday

- Pwn II (format string attacks, control flow hijacking) with Sam ctf.sigpwny.com

sigpwny{TODO}

Meeting content can be found at sigpwny.com/meetings.

SIGPwny