A Layman’s Guide to Cryptography

SELF 2024
Charlotte, NC
2024-06-08

ZMN XMRE IJENTFID

What this talk isn’t about

- Cryptocurrency/memecoins
- Implement Rijndael
- Complicated math
- How to deploy a certificate authority

Actually in this talk

- What is a cipher, key, mode, hash?
- Extreme oversimplifications
- How does Linux FDE work?
- Why RSA isn’t named after its creator
- How can I use my bank’s website over WiFi?
- Am I getting pwned by Quantum computers?

The Speaker

- Cameron Conn
- Dev/Hacker/Linux User for ~15 years
- Sr. Software Engineer at `$DAYJOB`
- I have implemented everything in this talk

Crypto-what?

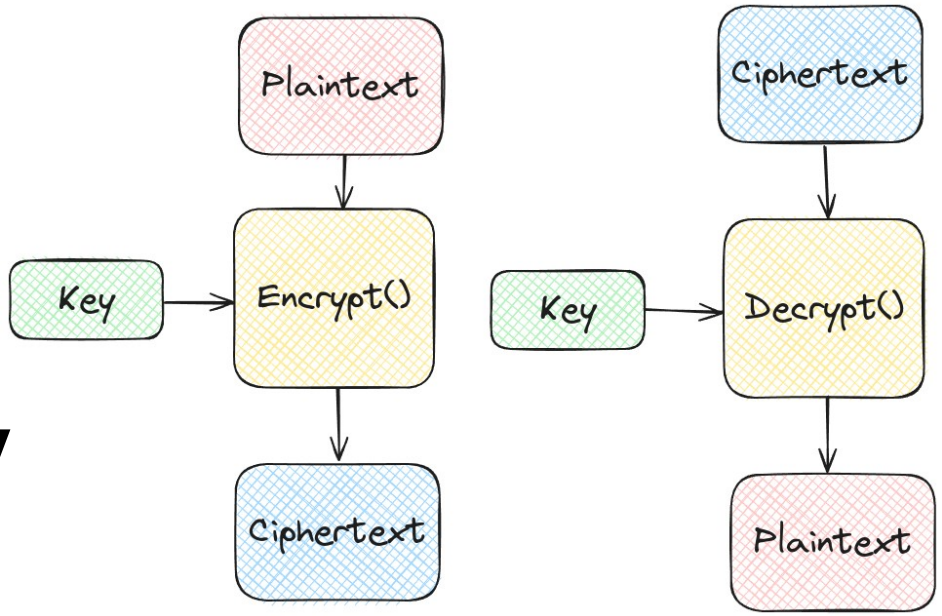
- What
  - Securely communicate around adversaries
- Why
  - Keep a secret
  - To prove authenticity
  - To uniquely identify something

Why it Works

- It’s hard to guess random numbers
- AES
  - 128 bits: 3.26 Bn Bn Years
  - 256 bits: 5.54 x 10^{29} Bn Bn Bn Yrs
- RSA
  - 2048 bits: 2.57 Mn Bn Yrs
  - 4096 bits: 152 Bn Bn Bn Yrs
- Age of the universe: 13.7 Bn Yrs

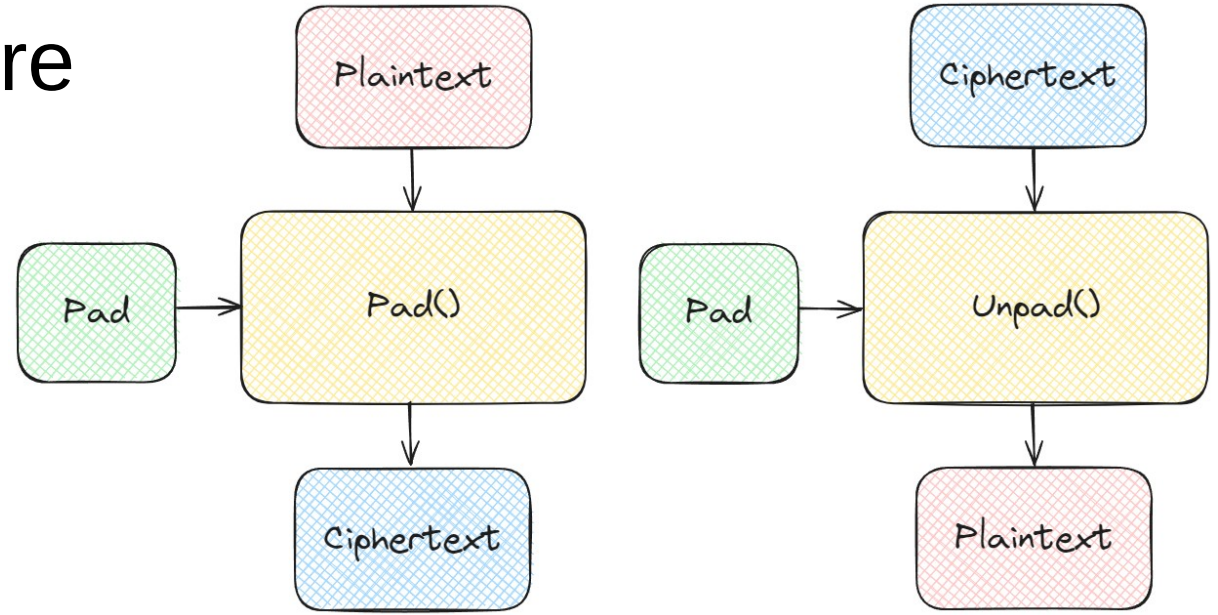
Symmetric Ciphers

- Have a **plaintext** message
- **Encipher** with a **key** to get **ciphertext**
- **Decipher** the with the **key** to get back **plaintext**

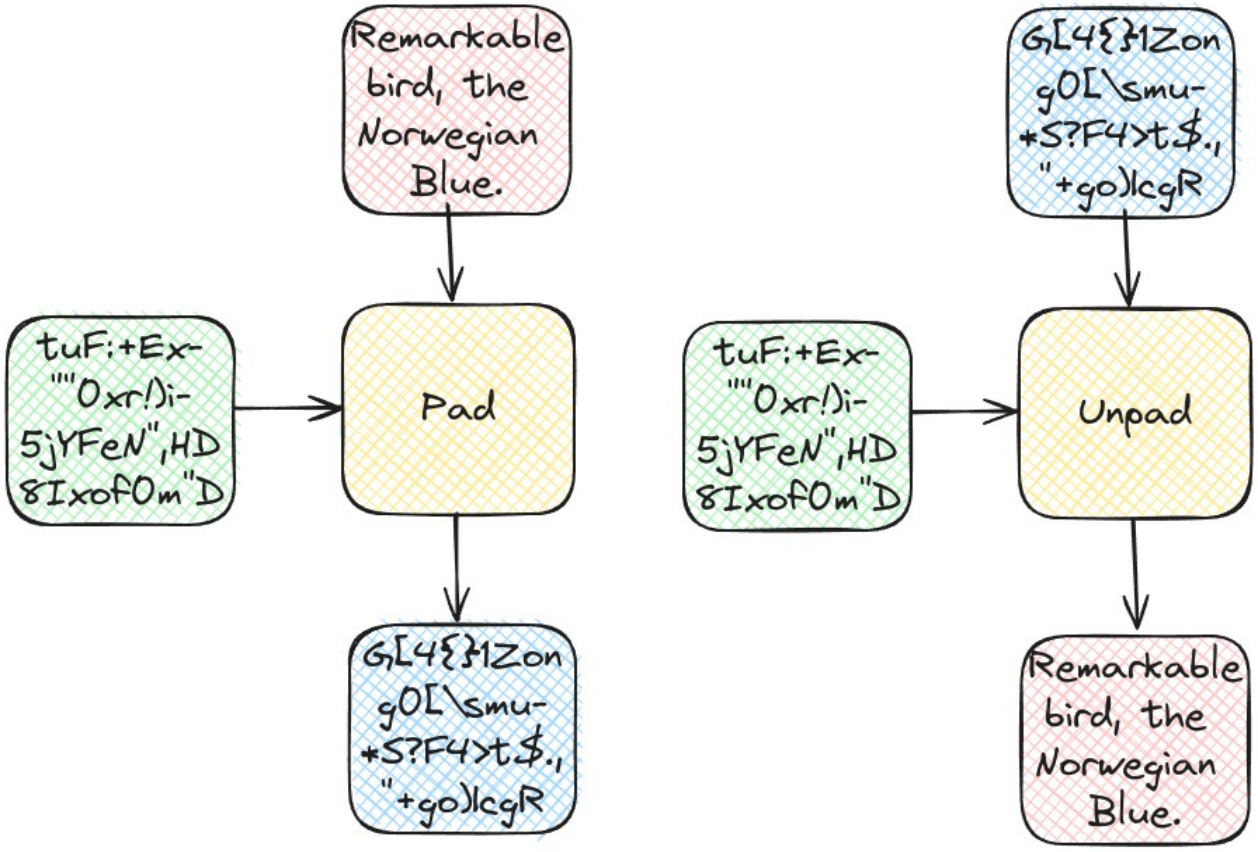


One Time Pad

- Theoretically Unbreakable
- Encrypt message with random **pad**
- Quantum secure



Example: One Time Pad



Issues: One Time Pad

- length(Key) = length(Message)
  - 1 GB Data + 1 GB Key = 2 GB Stored
- Re-using a key compromises secrecy
  - Allows reconstruction of message
  - Examples
    - Repeating key XOR
    - Venona Spy Cables

“Modern” Symmetric Ciphers

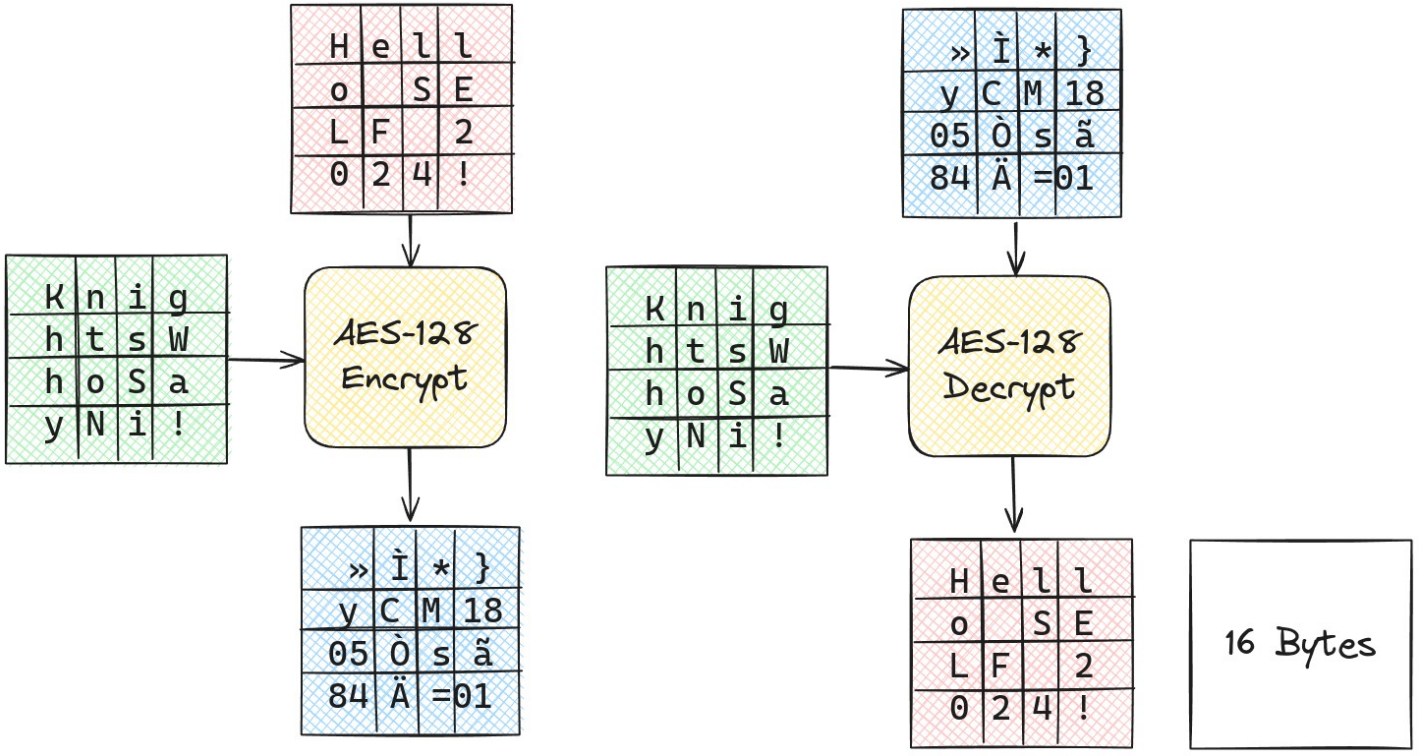
- Want to re-use a small key
  - Re-use is nice
  - But not too small to guess (≤ 8 bytes)
- Make ciphertext **indistinguishable** from a random message
- Types
  - Block
  - Stream

Block Ciphers

- Symmetric Cipher
- Fixed size **Key**
  - Generated with a **CSPRNG** (*not* rand(3))
- Fixed size Message, the **Block**
- e.g. AES, Twofish

Example: Block Ciphers

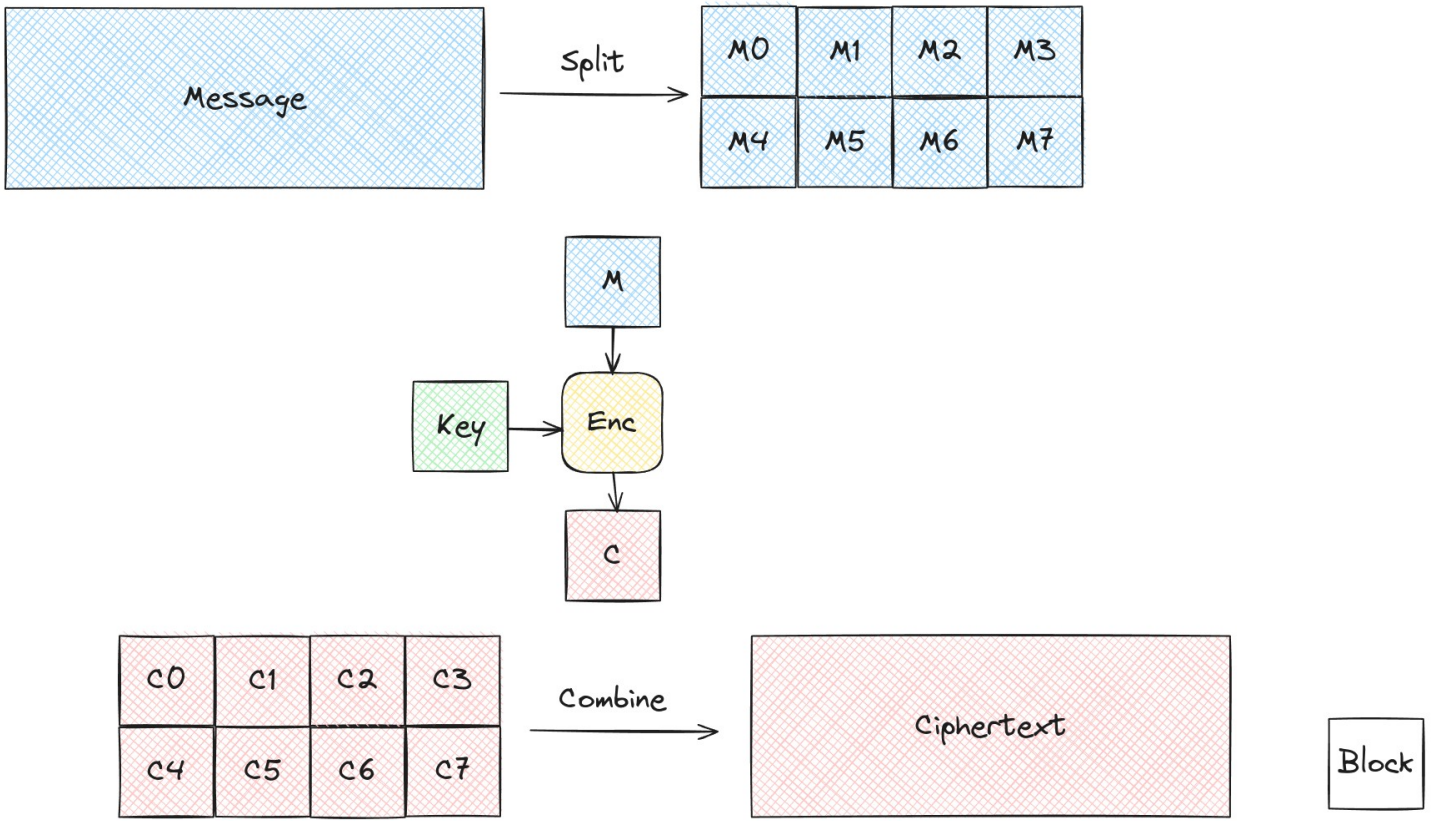
```
Message: Hello SELF 2024!
Key:     KnightsWhoSayNi!
```



Issues: Block Ciphers

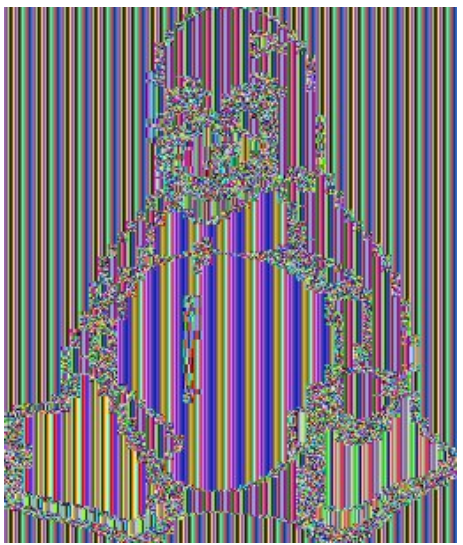
- Can’t handles message not divisible into blocks
  - Need **Padding** to a full block
- Block Ciphers only handle one block
- Need a special **Mode** of operation for > 1 block

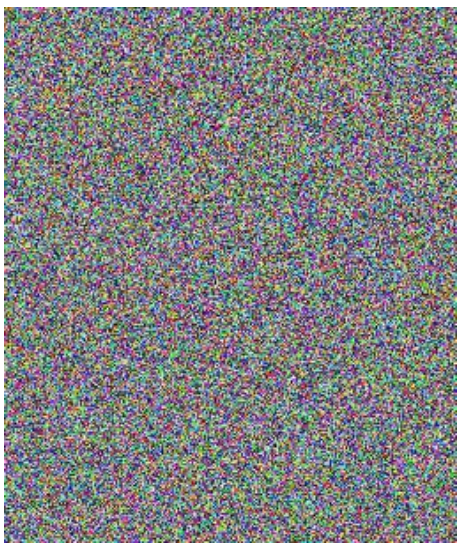
ECB: Electronic Codebook



ECB: Extra Cryptographically Broken

- Don’t use ECB
- You can “decrypt” ECB just by looking at it:





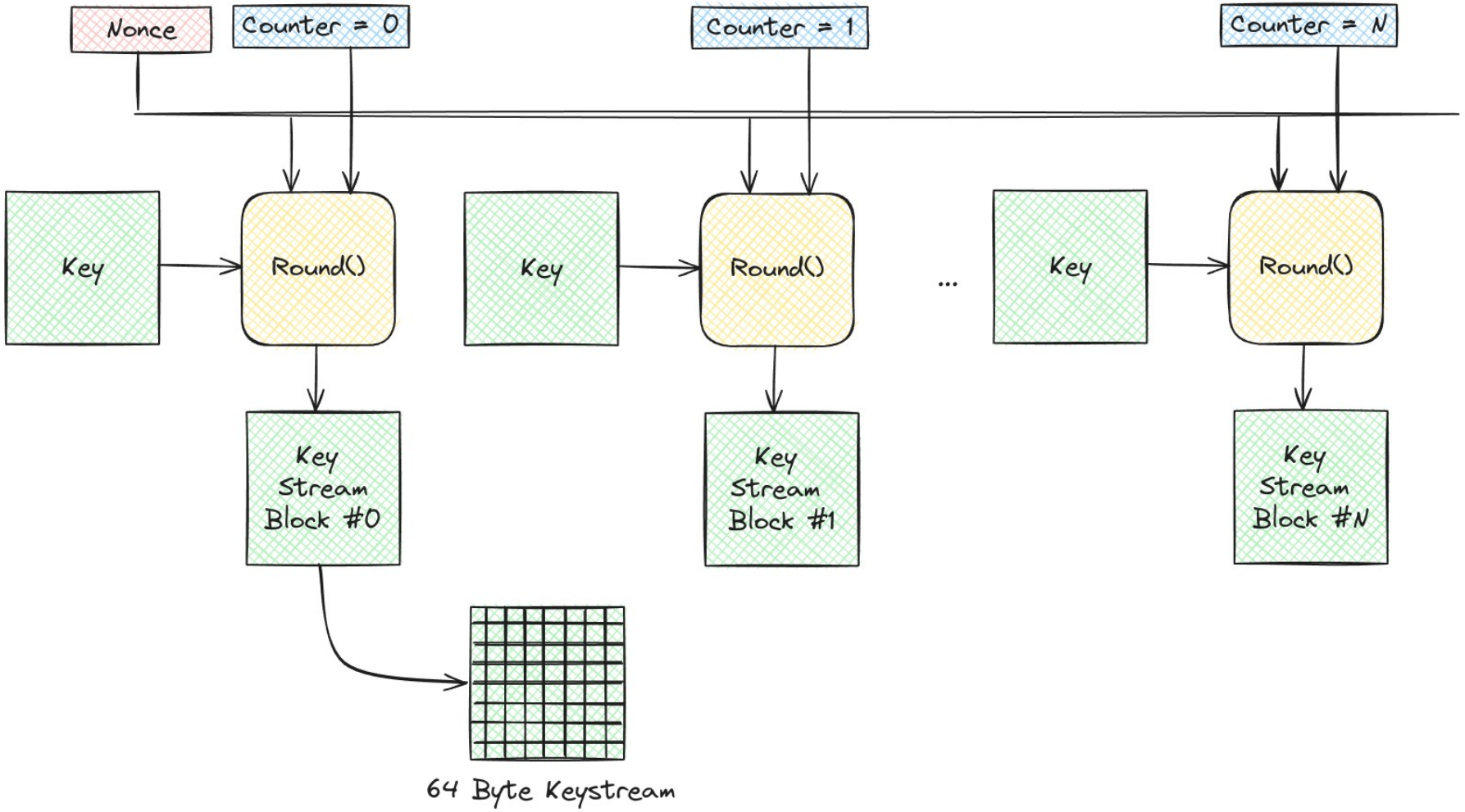
Modes

- The “simple” modes have issues (ECB, CBC)
- Use other ones for speed + security
  - CCM
  - GCM
  - CTR
  - XTS

Stream Ciphers

- Like OTP
  - Difference: Pad with a **Keystream** derived from a key
  - Difference: Only as secure as Keystream algorithm
- Flexible
  - No padding
  - No mode of operation
- Examples: ChaCha/Salsa20, HC-256, MICKEY

Salsa20 Stream Cipher



So Far

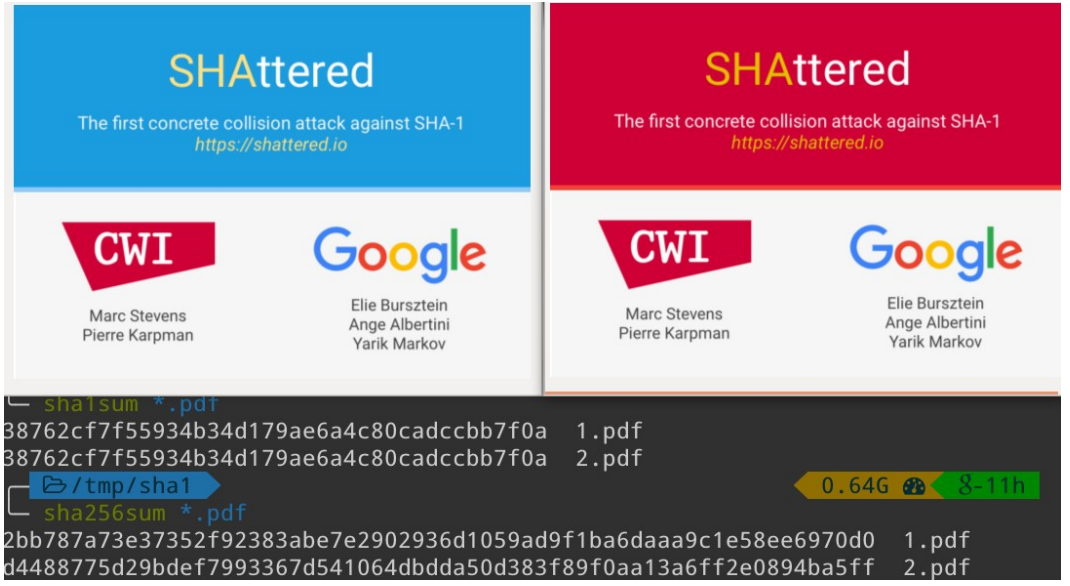
- Symmetric Ciphers
- Modes of Operation
- Block vs Stream Ciphers
- Things that are *intuitively* OK break security

Hashing

- A **hash function** is a one-way function that changes an arbitrary length input into a (usually) fixed length output (the **hash** or **digest**)
- We care about **cryptographic hashes**

Cryptographic Hashes

- It *should* be difficult to find the inputs when you *only* know the output
- It *should* be difficult to find two inputs with the same output (a **collision**)
- Made-up categories:
  - Fast Hashes
  - Password Hashes



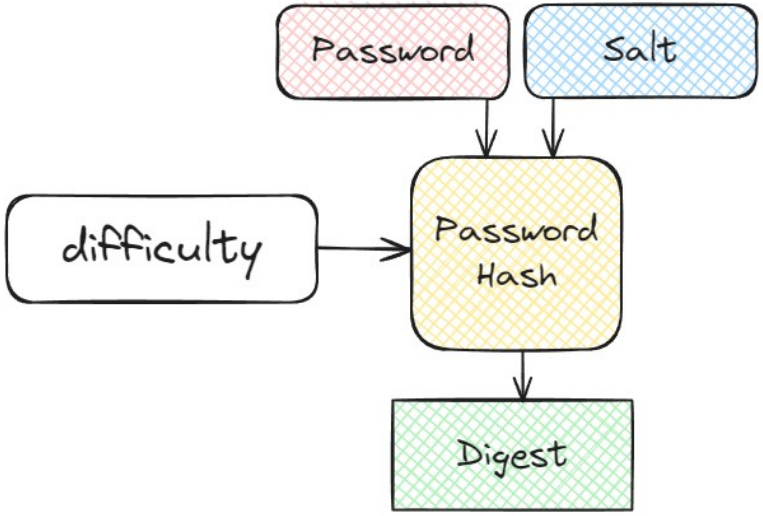
“Fast” Hashes

- Variable length of data goes in and a fixed-length digest comes out.
- e.g. SHA family, Blake3, MD5 (bad)

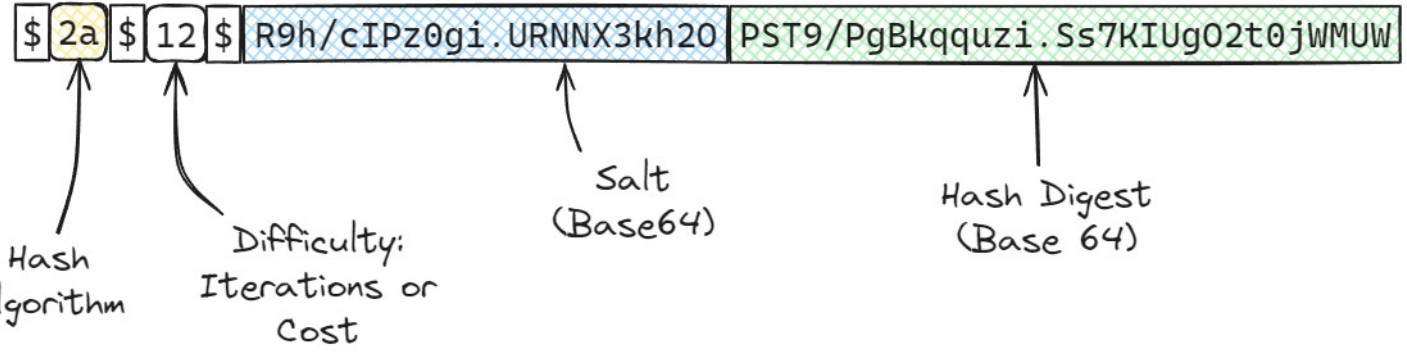
```
cam@BigBox:~/Documents/SELF-2024$ sha256sum Mona_Lisa.jpg
9f7c25a9260e754cfbffeeaf3a0add0f6e650697b91eb27285f267d00028f77f  Mona_Lisa.jpg
```

Password Hashes

- Called a **Key Derivation Function** (**KDF**)
  - Used to avoid storing user passwords on websites
  - Or user-defined passphrases (disk encryption)
- Designed to hamper brute force attacks
  - Accept a per-hash **salt**
  - Usually accept a difficulty
- e.g. Argon2, scrypt, bcrypt



BCrypt Hash Format

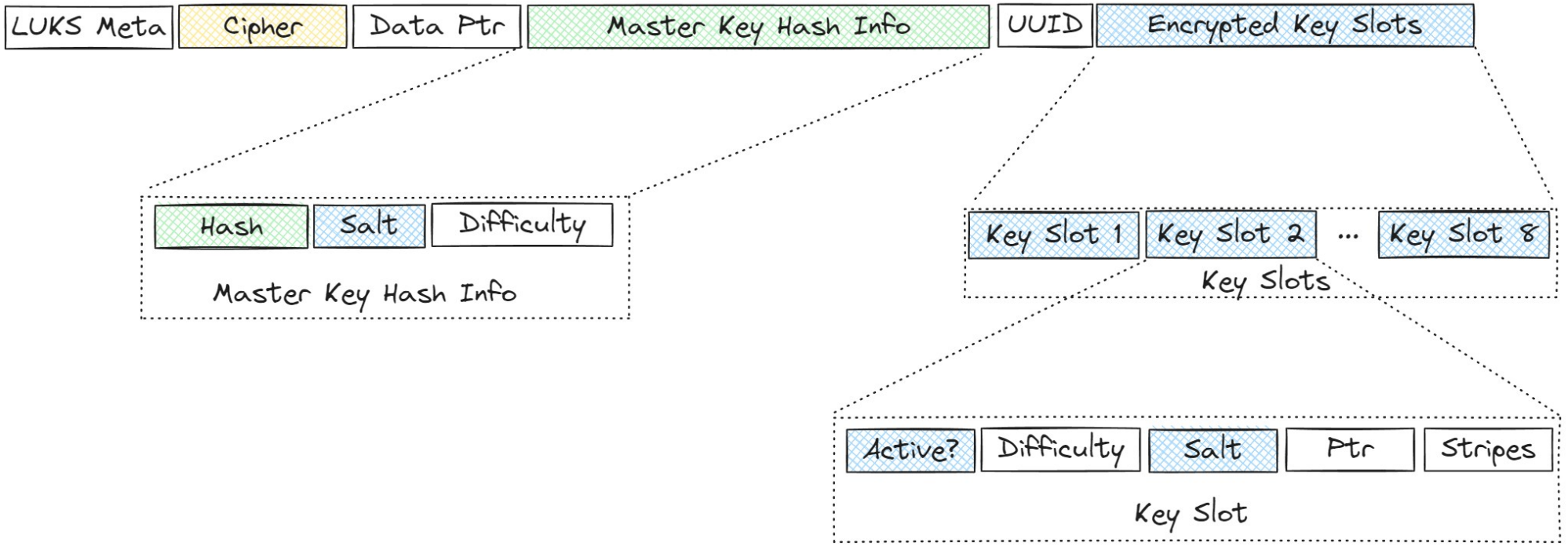


Aside: Full Disk Encryption (FDE)

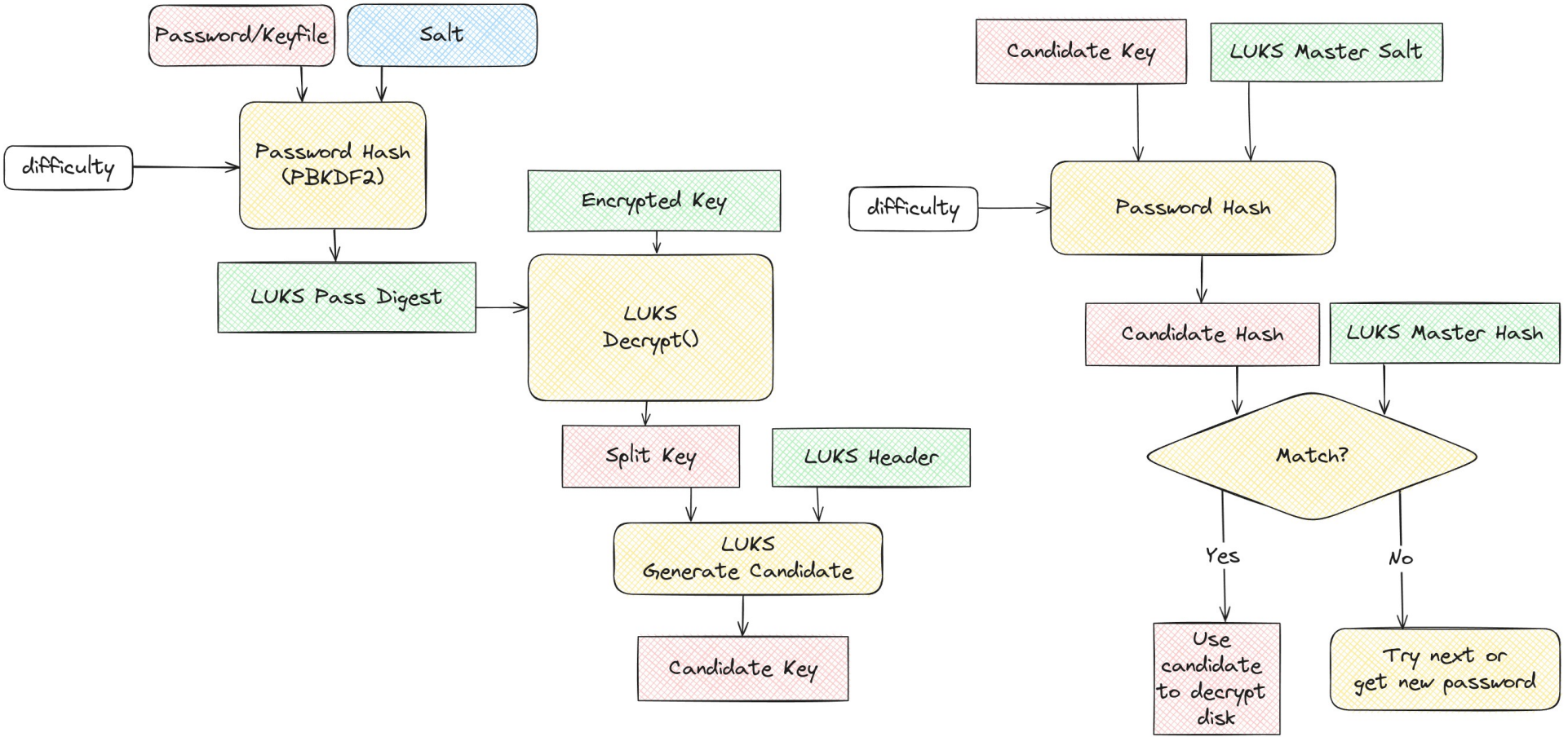
Aside: dm-crypt+LUKS

- UEFI/BIOS loads Linux kernel
- kernel calls `cryptsetup(8)`
  - Asks for password
  - Checks password hash against LUKS header
    - Called **Key Recovery**
  - If match, hand over key to dm-crypt
- Then keep booting Linux

LUKS Header Layout



LUKS Key Recovery

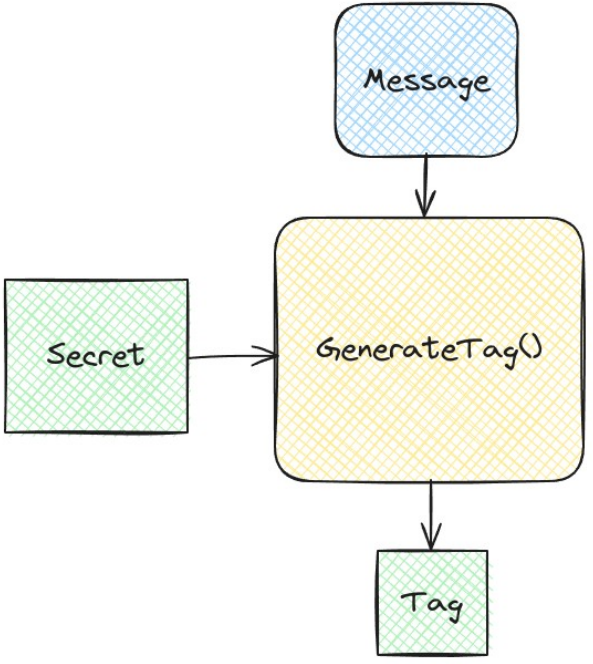


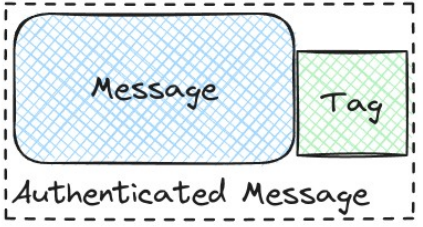
Now we can keep a message secret.

How do we prove the message wasn’t modified?

Message Authentication Codes

- **MAC**s are like a **checksum**
- Use a **secret** to **authenticate** a message.
  - Difficult to forge
  - Useless without the secret





Summary: Symmetric Crypto

- Ciphers: Keep a secret
- Hash: Identify something without storing it
- MAC: Verify authenticity

And now for something completely different

Asymmetric Crypto

- Generate a keypair of a **public** and **private key**
- Give out public, keep private secret
- You can now:
  - Sign
  - Encrypt

Asymmetric Crypto (cont)

- Things you can do
  - De/Encrypt messages
  - Sign/Verify messages
  - Establish a shared secret between two parties
  - Delegate authority (e.g. Web Certificates)
  - Use your bank’s website

Asymmetric Systems

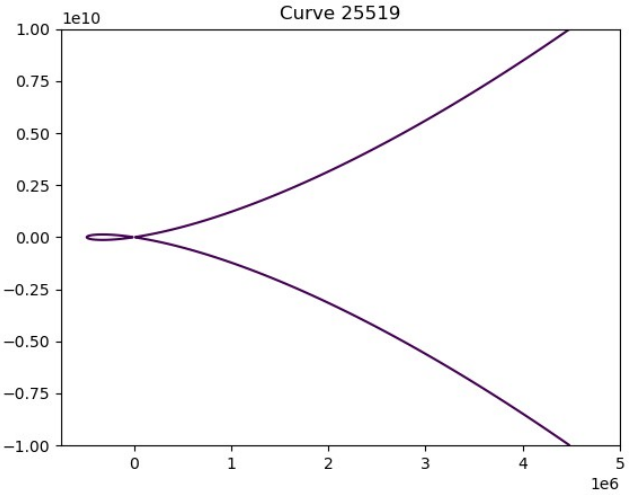
- Two common ones currently
  - RSA
  - Elliptic Curves (ECC)
- Post-Quantum coming down the pipe:
  - Kyber (KEM)
  - Dilithium (Signatures)

RSA

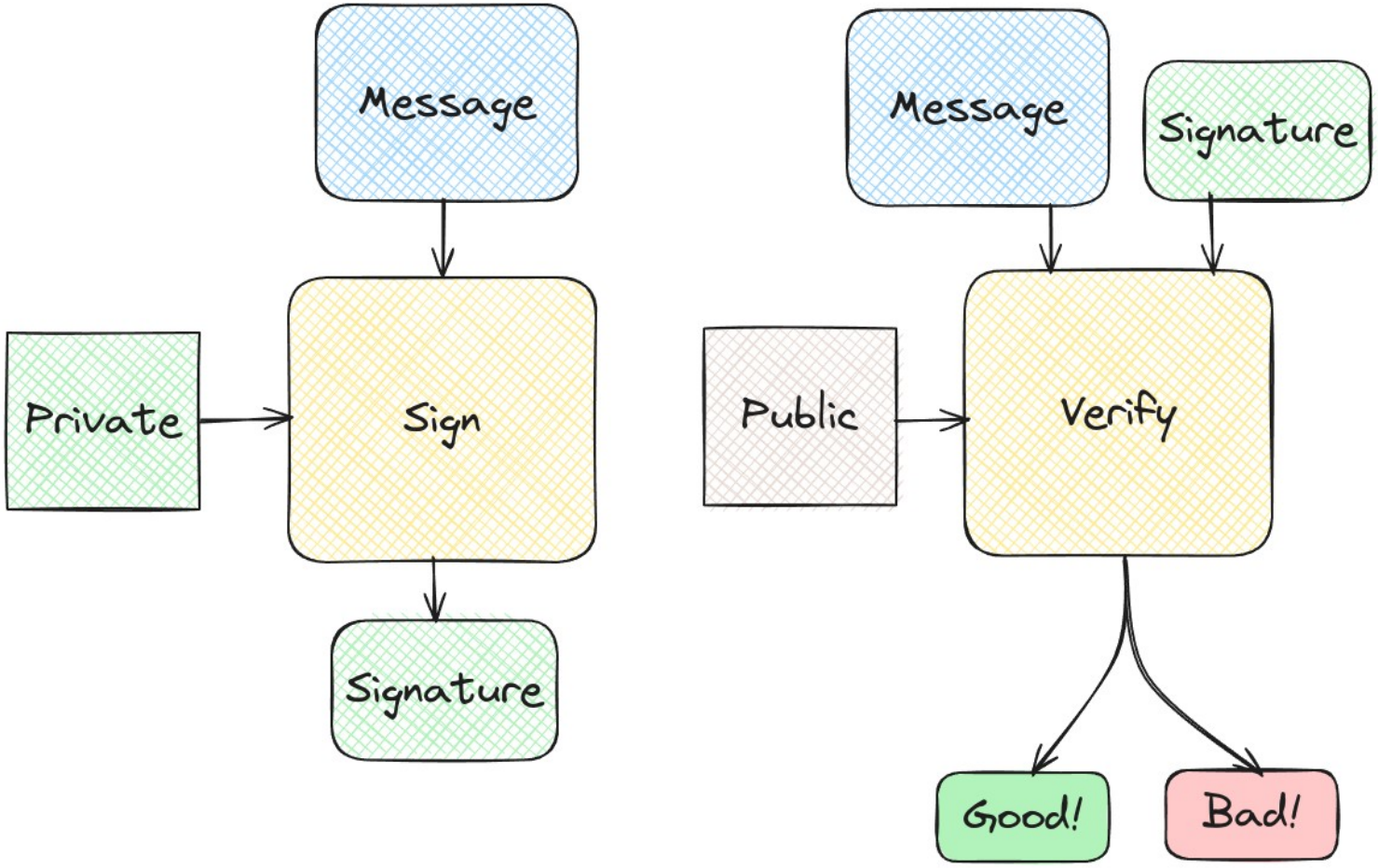
- Independently invented in the UK (1973) and USA (1977)
- Pick two very large secret large prime numbers (p & q)
- Calculate public key: N = p * q
- RSA-2048
  - len(p) == len(q) == 1024 bits
  - N = p*q
  - len(N) == 2048 bits
  - 256 bytes

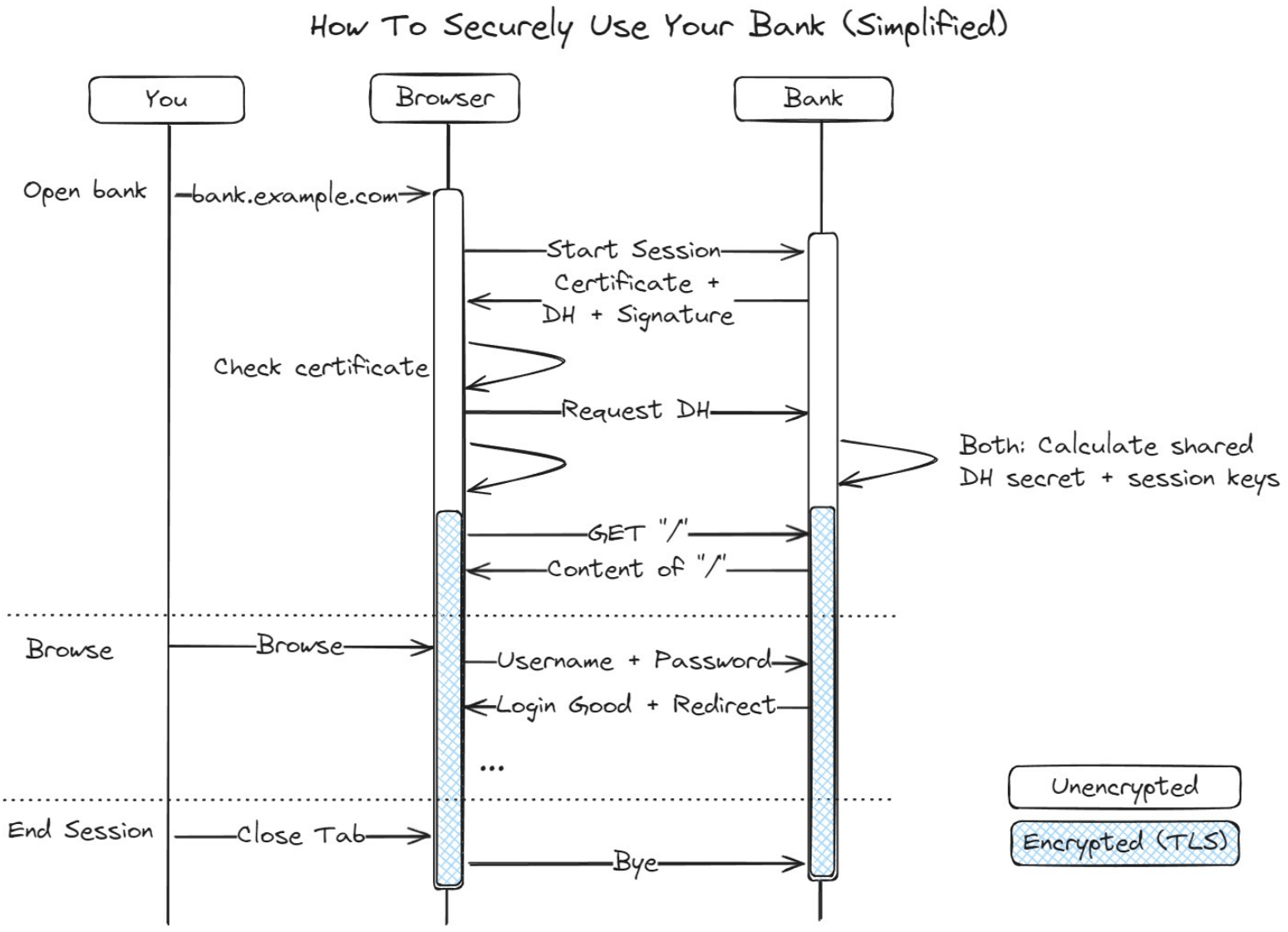
Elliptic Curve Cryptography (ECC)

- ECC uses point on a curve instead of scalar #s
- Split into signing + encryption algorithms
- Smaller keys
- Not actually a single system
  - You get to pick a curve as well
  - Many of these



Asymmetric Signing




How To Securely Use Your Bank (Simplified)
You
Browser
Bank
Open bank
bank.example.com
Start Session
Certificate +
DH + Signature
Check certificate
Request DH
Both: Calculate shared
DH secret + session keys
GET "/"
Content of "/"
Browse
Browse
Username + Password
Login Good + Redirect
...
End Session
Close Tab
Bye
Unencrypted
Encrypted (TLS)

Now what?

I can crypto now, right?

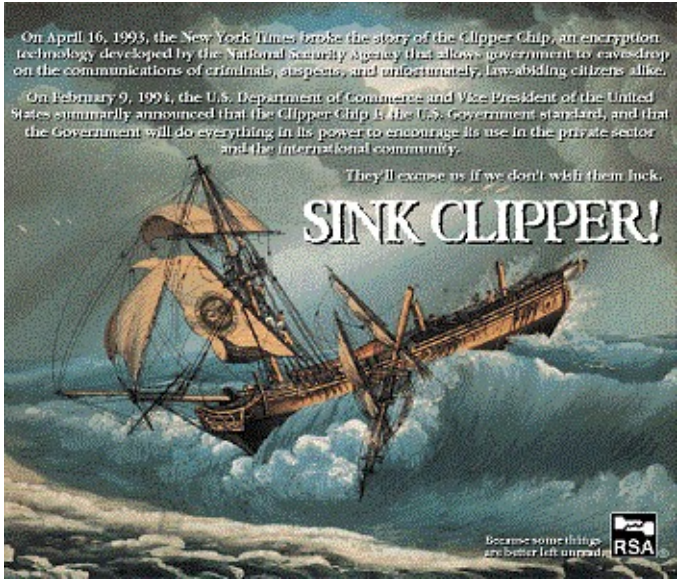
Internal Threats: Don’t FSCK it up

People don’t write crypto themselves because it’s easy to mess up

- Use a key after it “wears” out
- You use the wrong mode/padding
- You don’t verify the recipient’s key is well-formed
- ...And a whole lot more!

External Threats: Don’t Get Pwned

- Side Channels
  - Power
  - Timing
  - Acoustic
- Software vulnerabilities
- Gov-mandated backdoor
- A quantum computer breaks funny math



Pwned by Quantum Computers?

- Quantum Computing (QC) kind of breaks cryptography
- Asymmetric
  - RSA: Broken
  - ECC: Even more broken
- Symmetric + Hashes
  - Mostly OK
  - Grover’s Algorithm: ½’s key sizes

Post Quantum Cryptography (PQC)

- PQC is designed to be QC-secure
- Downsides
  - Immature algorithms may have flaws
    - e.g. SIKE, Rainbow
  - Way larger keys
  - Way slower than contemporary crypto
- Getting standards now

QC Recommendations

- Biggest threat is **Harvest now, decrypt later**
- Don’t Panic
  - Move to the exists in an orderly fashion
  - Start migrating to PQC immediately
  - Consider hybrid constructions
- Stay up to date
  - Keep software up to date for when support is added
  - Move to vendors that support PQC if yours don’t

Other Recommendations

- I want to wrap an existing protocol
  - OpenSSL
  - s2n
- I want to write write a new protocol
  - See above, or...
  - libsodium (or NaCl)

Recap

- Basics of common crypto
- Pitfalls of rolling your own
  - Throw it away and use someone else’s
  - Unless you have a degree in math
- The quantum threat

Would you like to know more?

- Cryptopals
- Books
  - *Real World Cryptography* by Wong
  - *Serious Cryptography* by Aumasson
    - 2nd edition scheduled for August 2024
  - *Handbook of Applied Cryptography* by Menezes, Oorschot, & Vanstone
  - Read the spec (AES/Rijndael, SHA, Curve25519, ChaCha20)
- Roll your own crypto
  - Not Safe for Sanity

Slides:
https://www.camconn.cc/SELF-2024
or scan: