



# **Cryptanalysis and Crypto Research**

# Outline

- How hackers learn
- Cryptanalysis and the homework
- Bits matter in cybersecurity
- We've already covered the basics, these slides are just the lay of the land and things you can explore more on your own
  - We're going to go through these slides fast

# How hackers learn

- Hacker community has always been about helping each other and asking questions, since at least the 60's
  - My mom used to borrow source code on stacks of punch cards from programmers at competing companies
  - 2600 magazine, phrack, Chaos Communication Club, Homebrew Computer Club, DEFCON, USENET, BBSes, *etc.*

## But...

- It will serve you well (in this class, in your studies in general, in your future growth as a hacker) if you get into the habit of learning like a hacker:
  - Picking up a new language quickly via online tutorials
  - Reading source code (or binary code, or RFCs, or PCAPs, or whatever the case may be)
  - Reading manuals, man pages, online resources, Google results, *etc.*



...and...

- Hacking is all about reasoning about systems as they **actually** work, not how they're **supposed to** work
  - Means getting your hands dirty
  - Also means a healthy distrust of all authority (manuals, Professors, TAs, diagrams, textbooks, *etc.*)
  - Russian proverb: “Do not believe your brother, believe your own crooked eye.”



# Cryptanalysis and the homework

# Types of cryptanalysis...

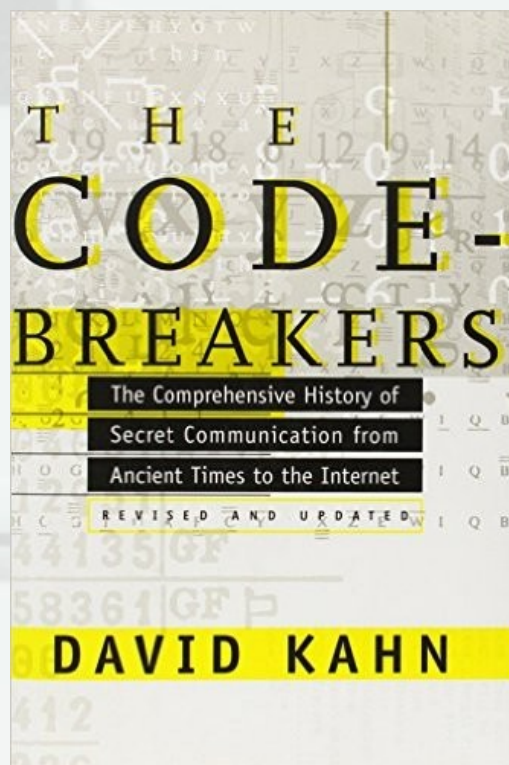
- Symmetric attack types according to outdated textbooks: Ciphertext-only, known plaintext (e.g., linear cryptanalysis), and chosen plaintext (e.g., differential cryptanalysis)
  - Often forget chosen ciphertext for, e.g., padding oracles
- Asymmetric desired properties: Indistinguishability under Chosen Plaintext (IND-CPA), Chosen Ciphertext (IND-CCA, IND-CCA2)
  - E.g., malleability of RSA (need something like OAEP)
- You should also read up on man-in-the-middle attacks, birthday attacks, attacks on hash functions, *etc.* for your own benefit

# Homework 1.1 and 1.2

- There's a gray area between ciphertext only and known plaintext, where you know something about the plaintext...



2000+ years of history...



# William and Elizebeth Friedman

- Met while analyzing Shakespeare ciphers at Riverbank Laboratories (“William Friedman wrote Shakespeare's plays”)
- Elizabeth solved ciphers of alcohol and drug smugglers, then German ambassadors in South America (three enigma machines)
- William led a team that solved PURPLE



# Zodiac cipher

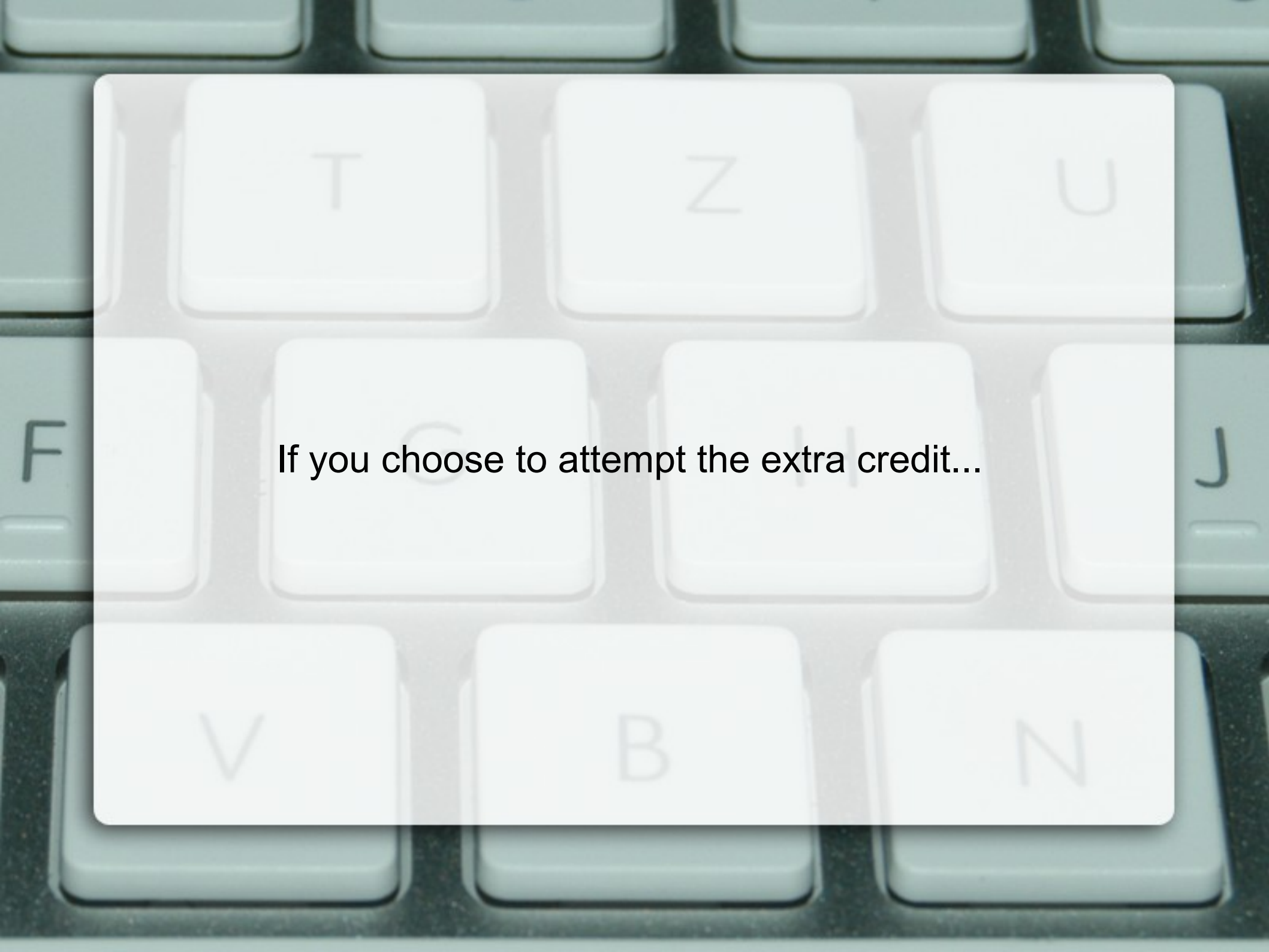


Image from wikia



# Bitwise XOR as a cipher itself

- Typically used by malware, 8 or 32 bits
  - WEP had similar problems
- $(B \text{ xor } K) \text{ xor } K = B$
- $(A \text{ xor } K) \text{ xor } (B \text{ xor } K) = A \text{ xor } B$
- $(0 \text{ xor } K) = K$
- $(K \text{ xor } K) = 0$
- Frequency analysis or brute force



If you choose to attempt the extra credit...



# Linear cryptanalysis (known plaintext)

- Block ciphers are made up of a limited variety of operations
  - XOR
    - addition modulo 2
  - Permutation
  - Substitution
    - Hard, need piling up lemma

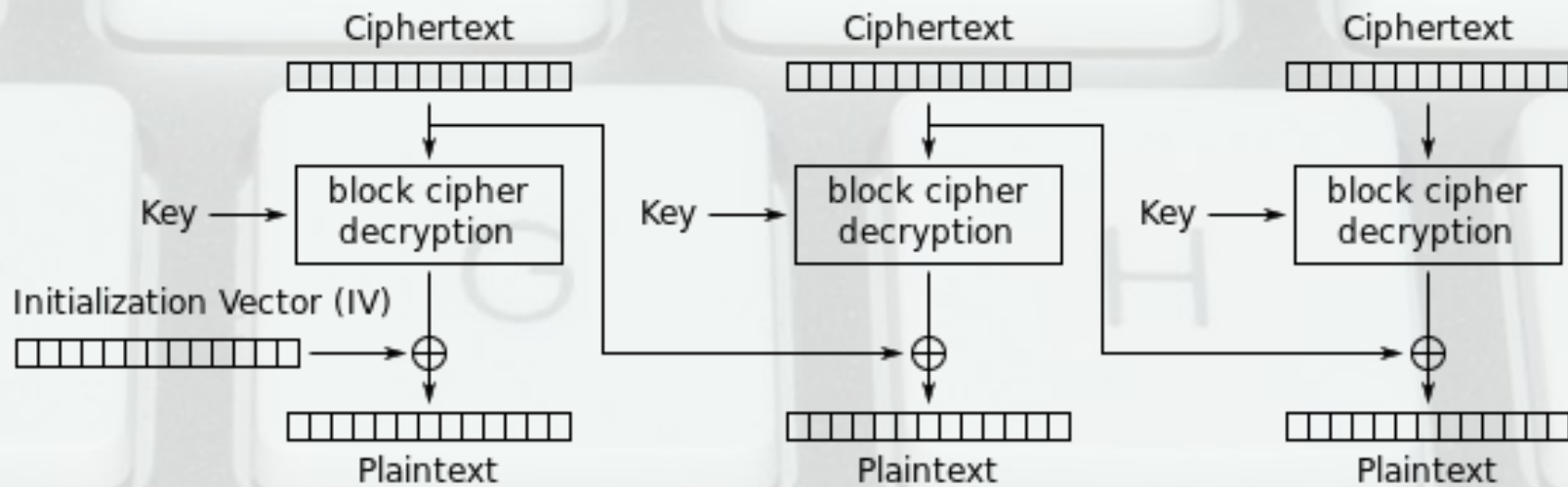
$$D = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

# Differential cryptanalysis (chosen plaintext)

- Choose plaintexts that differ in a small number of bits, e.g., 00110101 and 00100101
- Block ciphers are made up of a limited variety of operations
  - XOR (Bit difference is maintained)
  - Permutation (Bit difference is maintained)
  - Substitution
    - Hard

Chosen ciphertext...

# CBC decryption is malleable (chosen ciphertext attack)



Cipher Block Chaining (CBC) mode decryption

Image stolen from Wikipedia



Entering asymmetric crypto land...



In the food coloring or paint demos, it is assumed that mixing colors is cheap, but *un-mixing* them is prohibitively expensive.

# Modular arithmetic

$$5 + 7 = 2 \pmod{10}$$

$$7^2 = 9 \pmod{10}$$

$$8 + 8 = 6 \pmod{10}$$

# Modular arithmetic

$$8 + 9 = ? \pmod{10}$$

$$4^3 = ? \pmod{10}$$

$$1 + 1 = ? \pmod{10}$$

# Modular arithmetic

$$8 + 9 = 7 \pmod{10}$$

$$4^3 = 4 \pmod{10}$$

$$1 + 1 = 2 \pmod{10}$$

# RSA

Encryption:

$$c \equiv m^e \pmod{n}$$

Decryption:

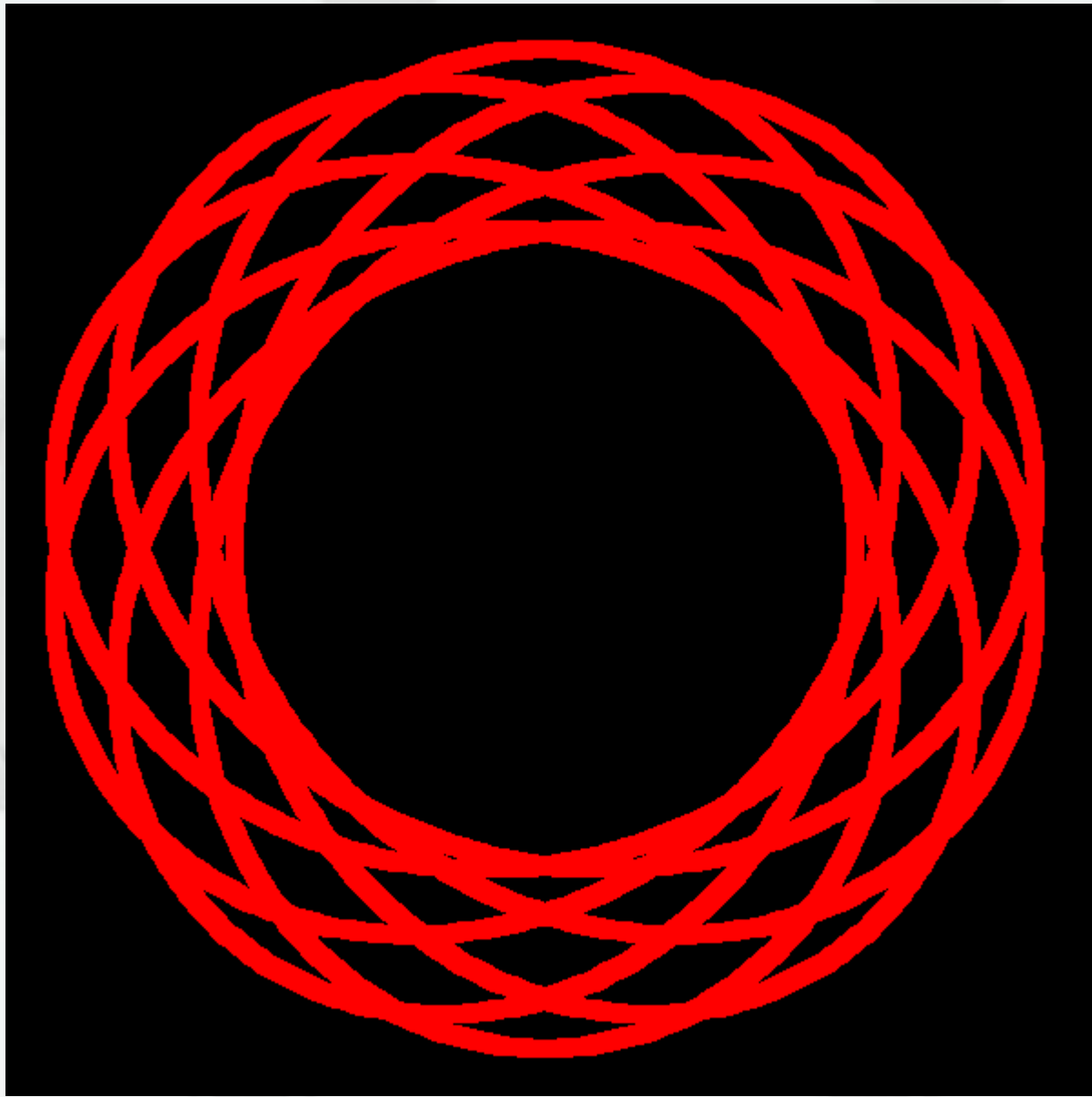
$$c^d \equiv (m^e)^d \pmod{n}$$

RSA provides encryption,  
authentication, and non-repudiation









# RSA

- Security is based on the hardness of integer factorization

$$n = pq$$

- $p$  and  $q$  are primes, suppose  $p = 61$ ,  $q = 53$
- $n = 3233$
- Euler's totient counts the positive integers up to  $n$  that are relatively prime to  $n$
- $\text{totient}(n) = \text{lcm}(p - 1, q - 1) = 780$ 
  - 52,104,156,208,260,312,364,416,468,520,572,624,676,728,780
  - 60,120,180,240,300,360,420,480,540,600,660,720,780
- Choose  $1 < e < 780$  coprime to 780, e.g.,  $e = 17$
- $d$  is the modular multiplicative inverse of  $e$ ,  $d = 413$
- $413 * 17 \bmod 780 = 1$

- Public key is  $(n = 3233, e = 17)$
- Private key is  $(n = 3233, d = 413)$
- Encryption:  $c(m = 65) = 65^{17} \bmod 3233 = 2790$
- Decryption:  $m = 2790^{413} \bmod 3233 = 65$
- Could also do...
  - Signature:  $s = 100^{413} \bmod 3233 = 1391$
  - Verification:  $100 = 1391^{17} \bmod 3233$
- Fast modular exponentiation is the trick
- Using RSA for key exchange or encryption is often a red flag, more commonly used for signatures

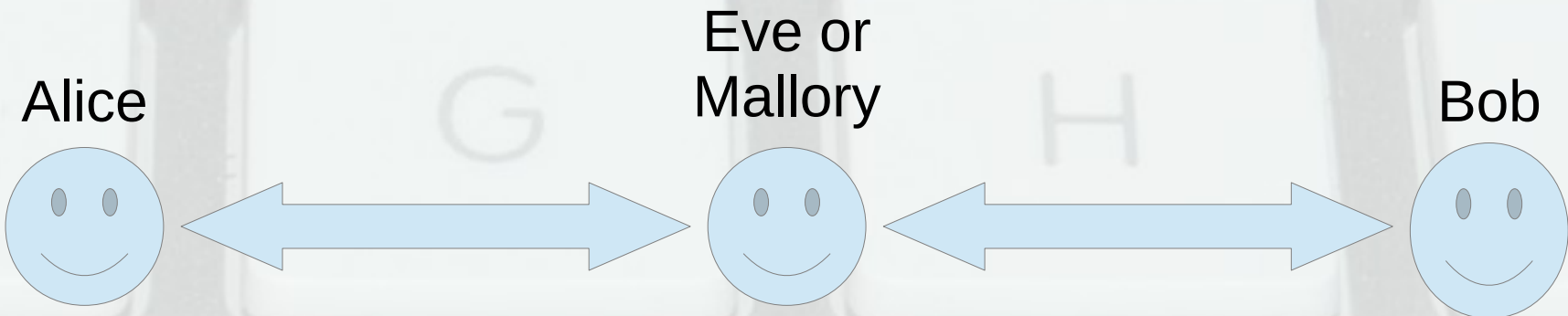


```
jedi@route66:~$ python3
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> for i in range (52, 781, 52):
...     for j in range (60, 781, 60):
...         if (i == j):
...             print(i)
...
780
>>> print((413 * 17) % 780)
1
>>> print(pow(2790, 413, 3233))
65
>>> print(pow(65, 17, 3233))
2790
>>> print(pow(100, 413, 3233))
1391
>>> print(pow(1391, 17, 3233))
100
>>> □
```

```
1
>>> print(pow(2790, 413, 3233))
65
>>> print(pow(65, 17, 3233))
2790
>>> print(pow(100, 413, 3233))
1391
>>> print(pow(1391, 17, 3233))
100
>>> print(pow(7, 17, 3233))
2369
>>> print((2369*2790) % 3233)
1258
>>> print(pow(1258, 413, 3233))
455
>>> print(7*65)
455
>>> print("{0:b}".format(78913))
10011010001000001
>>> print("{0:b}".format(78913*32))
1001101000100000100000
>>> print("{0:b}".format(78913<<5))
1001101000100000100000
>>> █
```

Homework 1.3...

# Man-in-the-middle attacks



## UCWeb

\* Led to discovery of active comms channel from [REDACTED]

(S/S/SHREL TO USA, FVEY) The CONVERGENCE team helped discover an active communication channel originating from [REDACTED] that is associated with the [REDACTED] [REDACTED] as they are known within the [REDACTED] hierarchy area of responsibility is for covert activities in Europe, North America, and South America. The customer [REDACTED] leveraged a Convergence Discovery capability that enabled the discovery of a covert channel associated with smart phone browser activity in passive collection. The covert channel originates from users who use UCBrowser (mobile phone compact web browser). The covert channel leaks the IMSI, MSISDN, Device Characteristics, and IMEI back to server(s) in [REDACTED]. Initial investigation has determined that perhaps malware can be associated when the covert channel is established. [REDACTED] covert exfil activity identifies SIGINT opportunity where potentially none may have existed before. Target offices that have access to X-KEYSCORE can search within this type of traffic based on their IMSI or IMEI to determine target existence.





# QQ Browser Fail #1 (not part of the homework)

- Keys should be 2048 or 4096 bits, at least
- 128 bits is not adequate
- 2454064175737408847100477458699650234  
63 =  
14119218591450688427 x  
17381019776996486069
- <https://citizenlab.org/2016/03/privacy-security-issues-q-q-browser/>

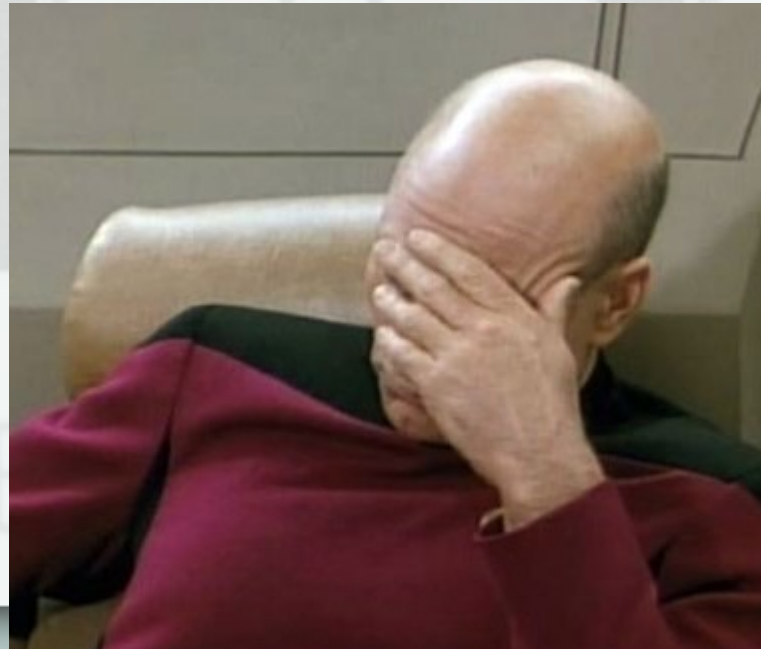
# QQ Browser Fail #2

(not part of the homework)

- AES session key generation

```
srand(currenttimeinmilliseconds)
```

```
key = rand()
```



## QQ Browser Fail #3

- RSA encrypt AES key (using public key of QQ's server) the AES session key and send it
  - Using 1024-bit textbook RSA encryption
- Textbook RSA is malleable...

- Public key is  $(n = 3233, e = 17)$
- Private key is  $(n = 3233, d = 413)$
- Encryption:  $c(m = 65) = 65^{17} \bmod 3233 = 2790$
- Decryption:  $m = 2790^{413} \bmod 3233 = 65$
- Eve wants a ciphertext that decrypts to  $m \cdot 7 = 455$ , without knowing  $m$ 
  - Eve...  $c(m=7) = 7^{17} \bmod 3233 = 2369$
  - $2369 * 2790 \bmod 3233 = 1258$
- $1258^{413} \bmod 3233 = 455$



Finally, the part about the homework...



Let  $C$  be the RSA encryption of 128-bit AES key  $k$  with RSA public key  $(n, e)$ . Thus, we have

$$C \equiv k^e \pmod{n}$$

Now let  $C_b$  be the RSA encryption of the AES key

$$k_b = 2^b k$$

*i.e.*,  $k$  bitshifted to the left by  $b$  bits. Thus, we have

$$C_b \equiv k_b^e \pmod{n}$$

We can compute  $C_b$  from only  $C$  and the public key, as

$$\begin{aligned} C_b &\equiv C(2^{be} \pmod{n}) \pmod{n} \\ &\equiv (k^e \pmod{n})(2^{be} \pmod{n}) \pmod{n} \\ &\equiv k^e 2^{be} \pmod{n} \\ &\equiv (2^b k)^e \pmod{n} \\ &\equiv k_b^e \pmod{n} \end{aligned}$$

Server chops off all but the lowest 128 bits

1. Record a session
2. Connect to the server with key shifted left 127 bits
3. Can you encrypt/decrypt with 1000000... or 0000000...?

(Just learned one bit of the key, repeat for left shift of 126 bits, 125 bits, etc. until you learn the key of the recorded session and can decrypt it)

This is a chosen ciphertext attack, and a padding oracle attack, but involves RSA padding rather than AES-CBC padding

Also called a Bleichenbacher-style attack.

Note:

QQ Browser used 1024-bit RSA for encrypting a 128-bit AES key, in the homework you'll see that the mock server being attacked uses 2048-bit RSA to encrypt a 256-bit AES key.

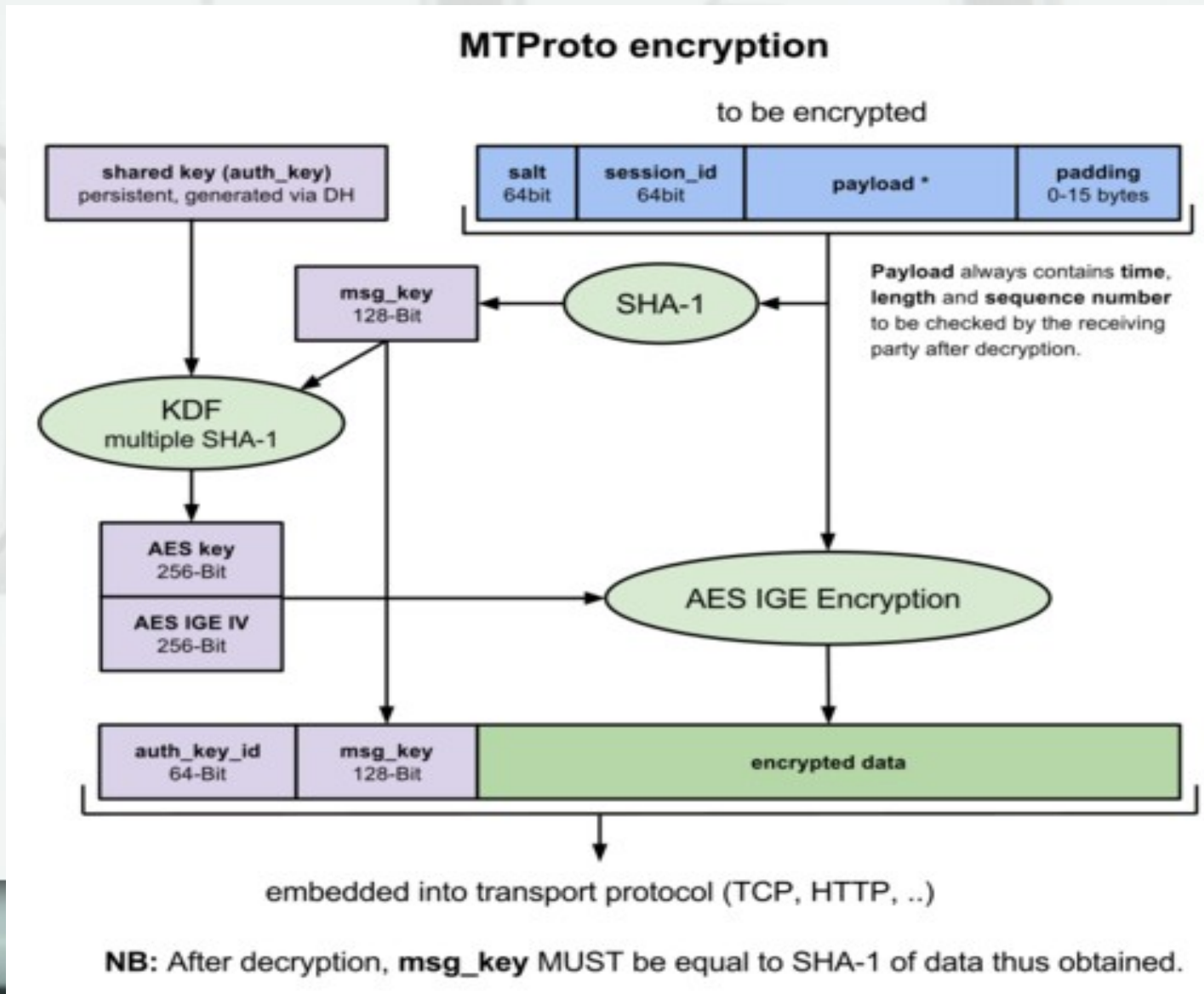
# Semantic security (*e.g.*, OAEP)

- Basic problem: we don't know the format of the plaintext
- Desirable properties
  - Indistinguishability under Chosen Plaintext Attack (IND-CPA)
  - Indistinguishability under Chosen Ciphertext Attack (IND-CCA)
  - Indistinguishability under Adaptive Chosen Ciphertext Attack (IND-CCA2)



# Telegram

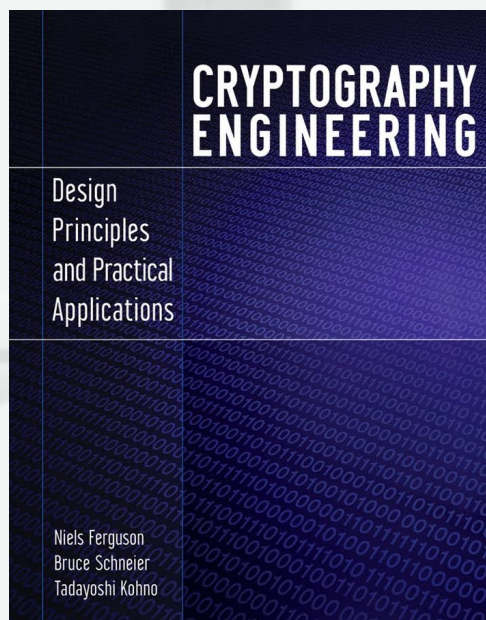
(<http://www.cryptofails.com/post/70546720222/telegrams-cryptanalysis-contest>)



Lay of the land  
(things to read about on your own if  
you're interested...)

Theme: crypto research is not  
“done”.

# *Cryptography Engineering by Ferguson et al.*



# Alternatives/complements to crypto

- Spread spectrum
- Steganography
- Wiretap Channels
- Quantum Key Distribution
- Randomized algorithms



# Hedy Lamarr



- Spread spectrum
  - Bluetooth
  - COFDM (WiFi)
  - CDMA
- Goal was torpedo guidance

# The future?

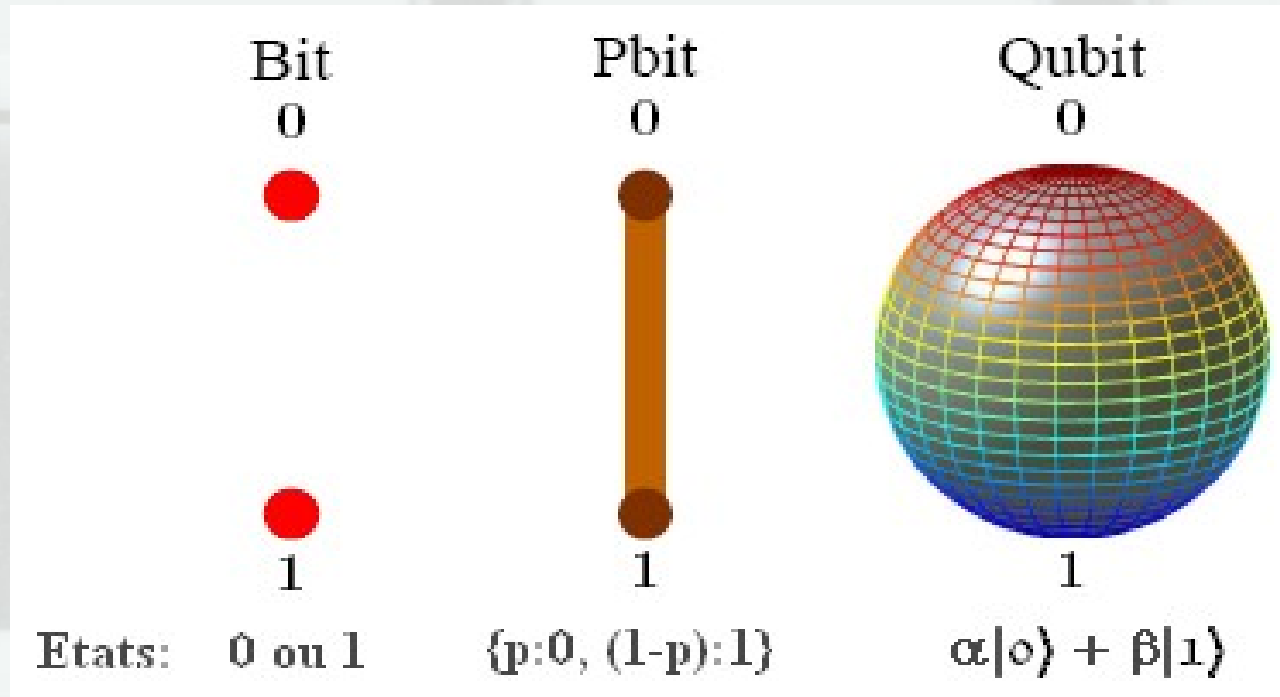
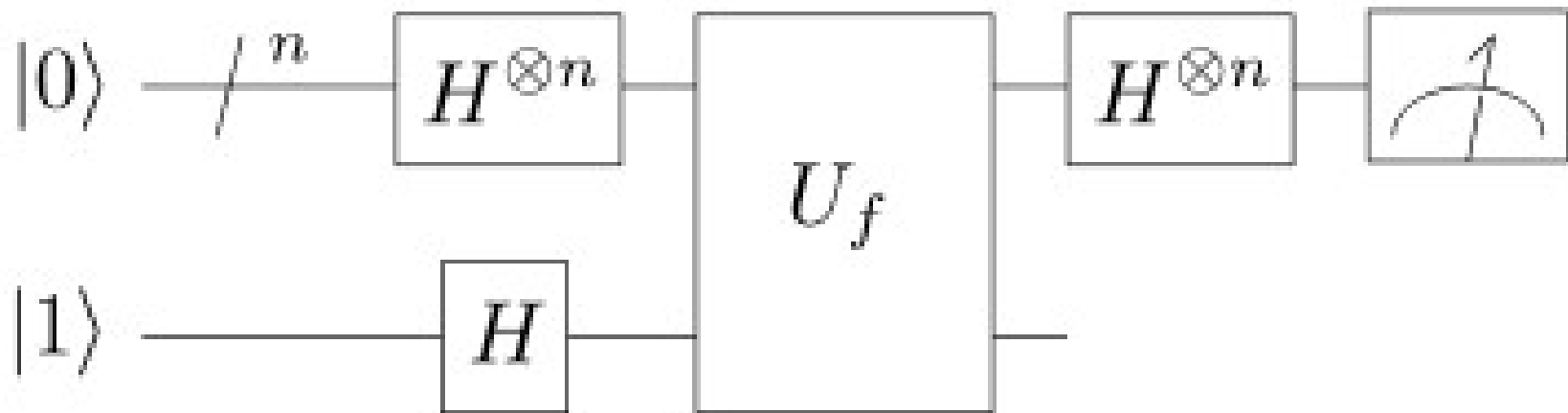


Image taken from <http://filipchsroom.blogspot.com/>

# Deutsch-Jozsa algorithm



By Skippydo - Own work, Public Domain,  
<https://commons.wikimedia.org/w/index.php?curid=2547135>

## Quantum computing example

$$f(x) = 10^x \pmod{15}$$

Shor's integer factorization algorithm involves a quantum Fourier transform.



# Asymmetric crypto is under threat

- Some newer algorithms can't (as far as we know) be broken by quantum computers
  - ***But*** RSA, Diffie-Hellman, elliptic curves, *etc.* all can
- Symmetric crypto is okay
  - Grover's algorithm finds the input corresponding to an output in  $O(\sqrt{N})$  time where  $N$  is the size of the function's domain



**Backup slides**

# Common symmetric algorithms

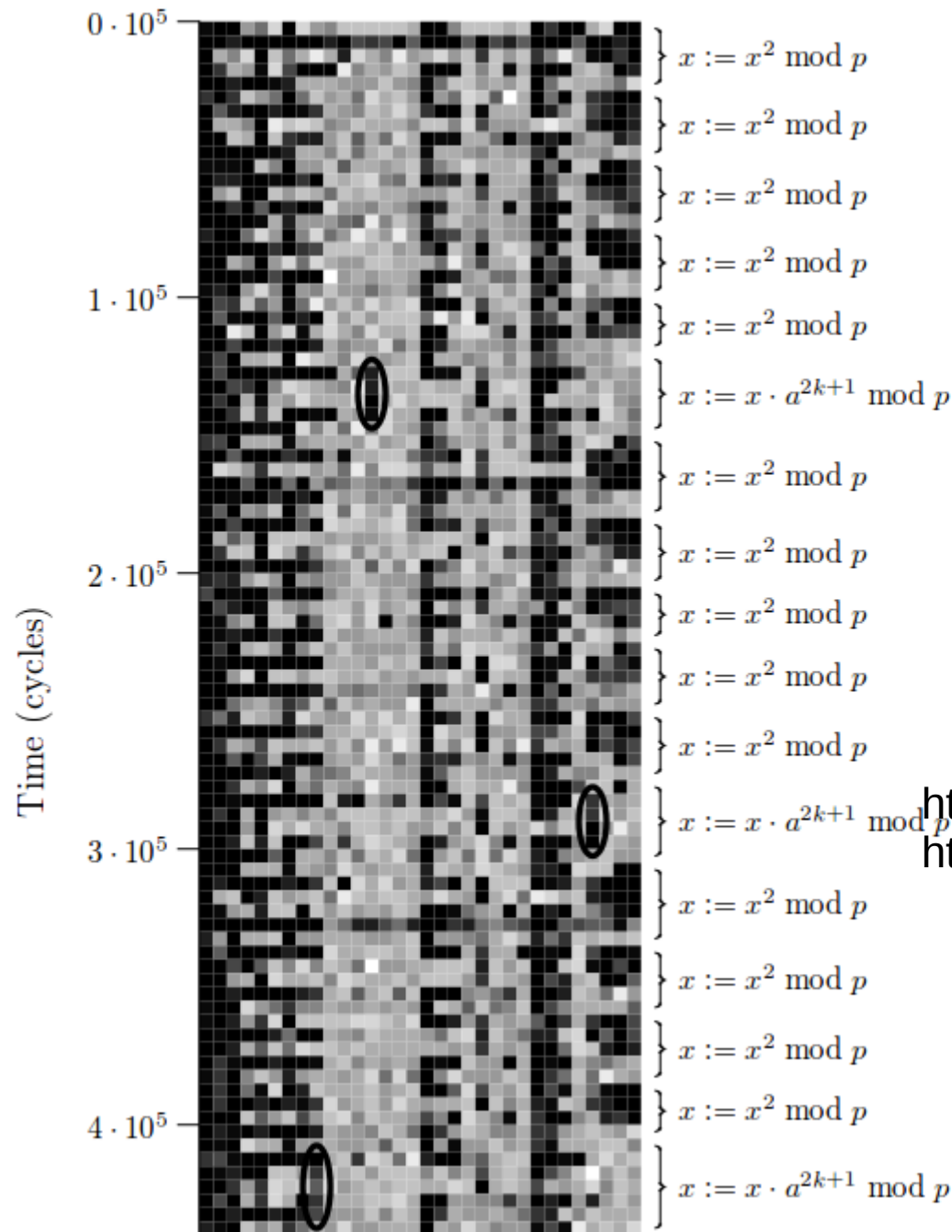
- DES (56-bit) and 3-DES (56, 112, or 168 bits)
  - DES is outdated, no good reason to use 3-DES that I know of
- AES (128, 192, or 256 bits)
  - Recognized standard
- Blowfish (32 to 448 bits, see also twofish and threefish)
  - Common, fairly good choice
- TEA (128 bits)
  - Simple to implement

# Common symmetric algorithms (continued...)

- RC4 (40-2048 bits)
  - Stream cipher, don't reuse key material
- IDEA (128 bits)
  - Cannot be expanded to larger key sizes
- Camellia (128, 192, or 256 bits)
  - Good alternative to AES
- Bitwise XOR (8 or 32 bits), ROT13 (e.g., WHAT→JUNG)
  - Terrible choices, but you'll see them...

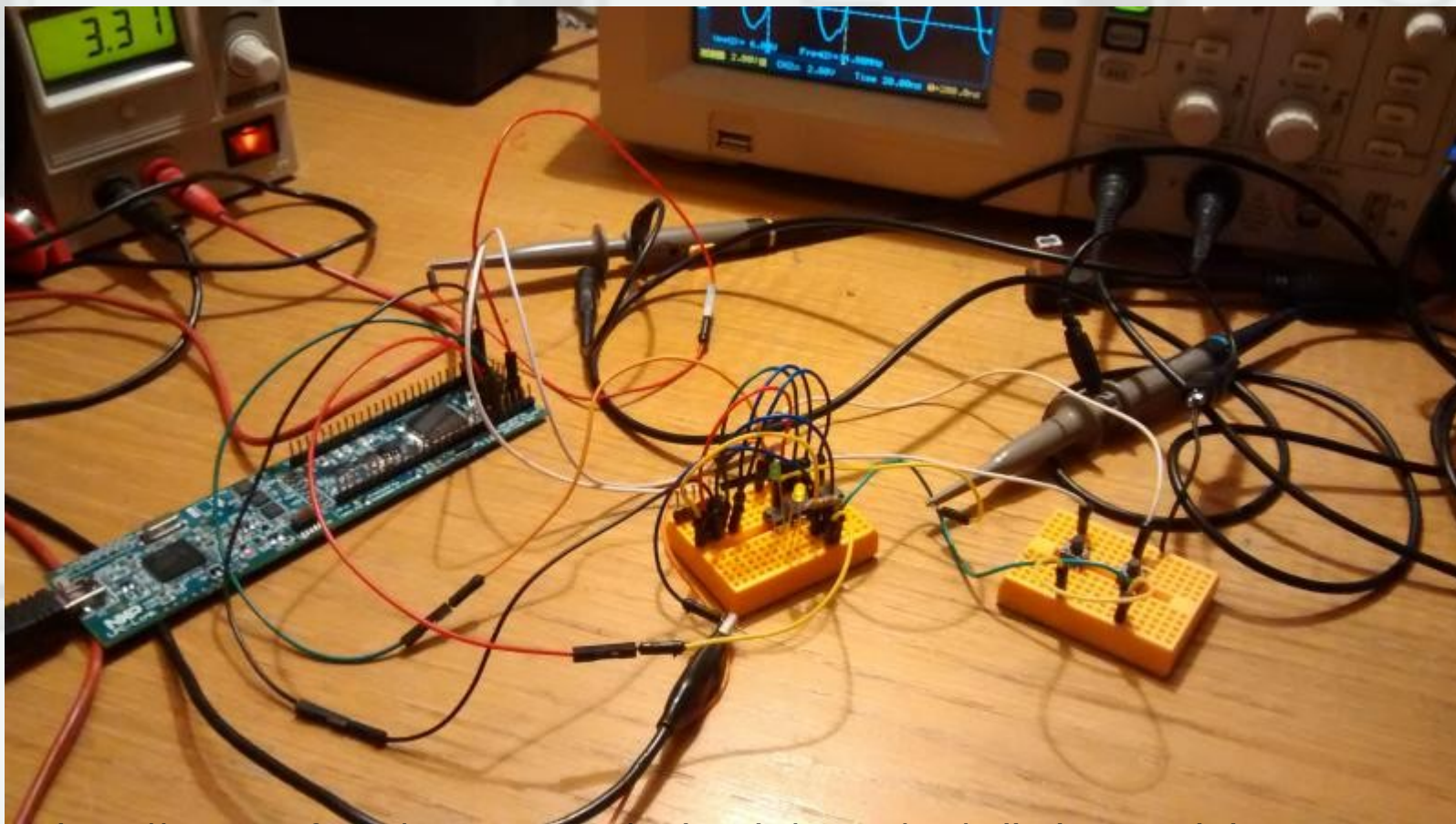


# Side channels



<http://www.daemonology.net/papers/htt.pdf>

# Fault injection attacks



<http://www.t4f.org/wp-content/uploads/2014/02/Glitch-Tutorial-setup.jpg>

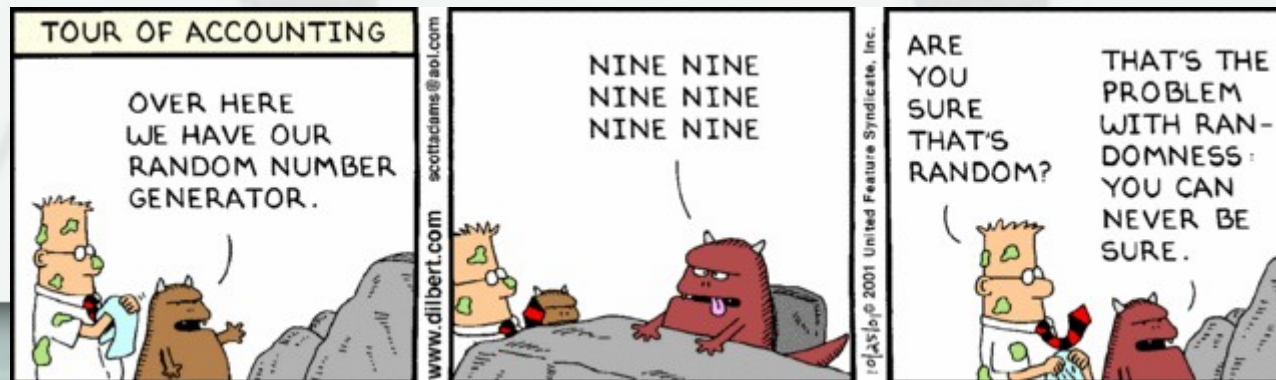


Random number generation



# Entropy needed for...

- Symmetric keys
- Asymmetric keys
- Initialization vectors
- Nonces
- Etc.





# Entropy pool

- `/dev/random` vs. `/dev/urandom`
  - Former blocks on read, latter doesn't
  - Entropy sources
    - Hardware support
    - Keyboard timings
    - Mouse activity
    - Hard drive activity



Some real goofs

# Cryptocat

- Array of random integers
  - {60278, 44571, 56801, 34115, 38861, 6386, 13716}
- As an escaped string
  - “\xeb\x76\xae\x1b”
- The above string in hex
  - 5c7865625c7837365c7861655c783162

See <https://tobtu.com/decryptocat-old.php>

# TomSkype

(<http://firstmonday.org/ojs/index.php/fm/article/view/4628/3727>)

**Algorithm 1:** Decrypting TOM-Skype 3.6–3.8 keyfiles

```
1: procedure DECRYPT( $C_{0..n}, P_{1..n}$ )
2: for  $i \leftarrow 1, n$  do
3:  $P_i = (C_i \oplus 0x68) - C_{i-1} \pmod{0xff}$ 
4: end for
5: end procedure
```

**Table 3: List of cryptographic algorithms and keys used for surveillance by TOM-Skype clients.**

Clients	Cryptography	Cryptographic key
TOM-Skype 3.6–4.2	DES+ECB (using only first 6 of 8 bytes of each plaintext block)	32bnx231
TOM-Skype 5.0	No surveillance	32bnx231
TOM-Skype 5.1–6.1 TOM-Skype Mobile Surveillance-only	DES+ECB (using only first 6 of 8 bytes of each plaintext block)	X7sRUjL\0



# Baidu

(report by Jeffrey Knockel, Sarah McKune, and Adam Senft)  
(<https://citizenlab.org/2016/02/privacy-security-issues-baidu-browser/>)

- Lots of custom stuff
  - Base64 substitution
  - Modified CBC
- ASCII encoded keys
  - *E.g.*, “vb%,J^d@2B1l’Abn”
- Other questionable decisions
  - TEA

# WordPress password hashes

- MD5(password)
- Don't Google this if you're offended by the f-word:  
“596a96cc7bf9108cd896f33c44aedc8a”
- How to do authentication properly is something we'll talk about later this semester, or maybe already talked about (salts would fix the above problem)

# Cryptovirology (1996)

- [Cryptovirology] by Young and Yung
- Ransomware (not counting AIDS trojan in 1989, started in 2005)
- Cryptocounters
- Cryptocurrency (Bitcoin in 2008)
- Mix networks (Tor paper presented in 2004)
- Private Information Retrieval (Chor *et al.*, 1995)
- Subliminal Channels (Gustavus Simmons in 1984)
- Salami slicing (Superman III in 1983, Office Space in 1999)
- RNG biasing

# Chat programs

- Things to read about on your own if you're interested:
  - Off the record messaging
  - Perfect forward secrecy
  - Signal's double ratchet algorithm



# References

- [Cryptography Engineering] *Cryptography Engineering: Design Principles and Applications*, by Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. Wiley Publishing, 2010.
- [Cryptovirology] *Malicious Cryptography: Exposing Cryptovirology*, by Adam Young and Moti Yung. Wiley Publishing, 2004.
- Lots of images and info plagiarized from Wikipedia