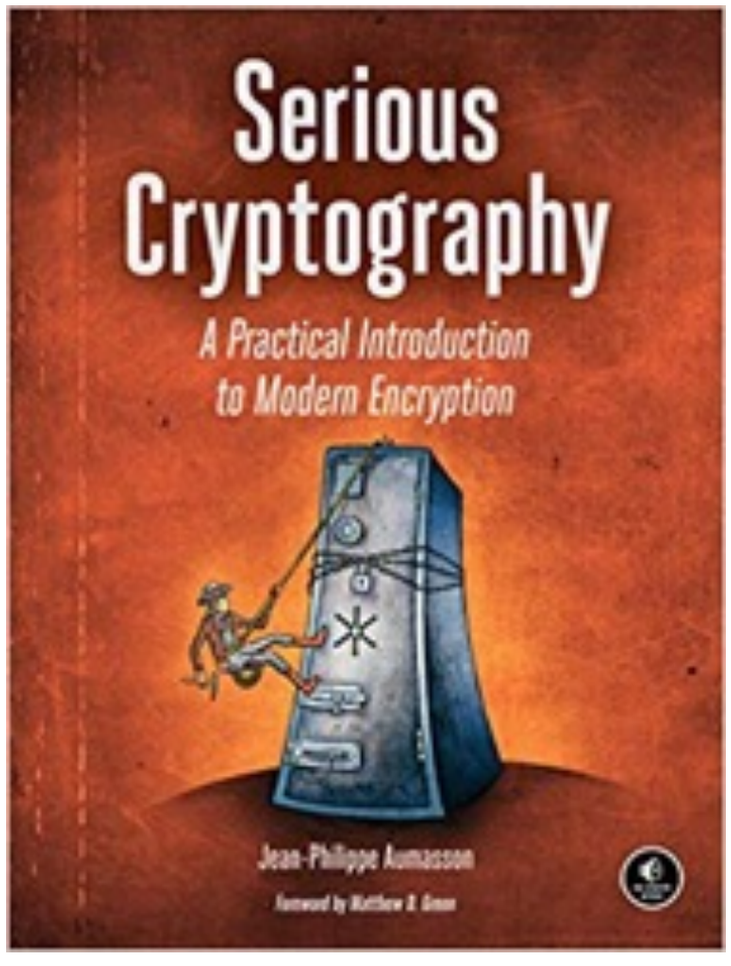
CNIT 141

Cryptography for Computer Networks



3. Cryptographic Security

Updated 9-7-22

Topics

- Defining the Impossible
- Quantifying Security
- Achieving Security
- Generating Keys
- How Things Can Go Wrong

Defining the Impossible

Two Types of Security

- Informational security
  - Theoretical impossibility
- Computational security
  - Practical impossibility

Informational Security

- Even an attacker with unlimited computation time and memory cannot crack it
- If an attack works, even if it requires trillions of years
  - That cipher is **informationally insecure**
- The one-time pad is **informationally secure**
  - Brute-force attack finds all possible messages, but there's no way to tell which is correct

Security in Practice: Computational Security

- A cipher cannot be broken with a reasonable amount of time, memory, hardware, power, money, etc.
  - Ex: AES-128 is currently **computationally secure**
    - Because no current system can try 2^128 keys
  - But quantum computers can test 2^128 keys with only 2^64 operations
    - So when or if they are developed, AES-128 will cease being **computationally secure**

Quantifying Security

- A cipher is (***t***, ε)-secure if cracking it requres

  t -- calculations an attacker will perform

  ε -- probability of success

- Ex: if no flaw is found in AES-128, it is
  - (***t***, ***t*/2^128)** - secure, for any **t** between 1 and 2^128

128-Bit Security

- If ***t*** is 1, attacker tries one key
  - Probability of success is **1/2^128**
- If ***t*** is 2^64, attacker tries 2^64 keys
  - Probability of success is **1/2^64**
- If ***t*** is 2^128, attacker tries 2^128 keys
  - Probability of success is **1**

Quantifying Security

Measuring Security in Bits

- Assume we want success probability of 1
- An attack requires ***t*** operations
- **n** bits of security means it will take 2^**n** operations to crack
- A perfect cipher with a 128-bit key provides 128 bits of security
  - Does not consider the speed of a single operation; a rough measure

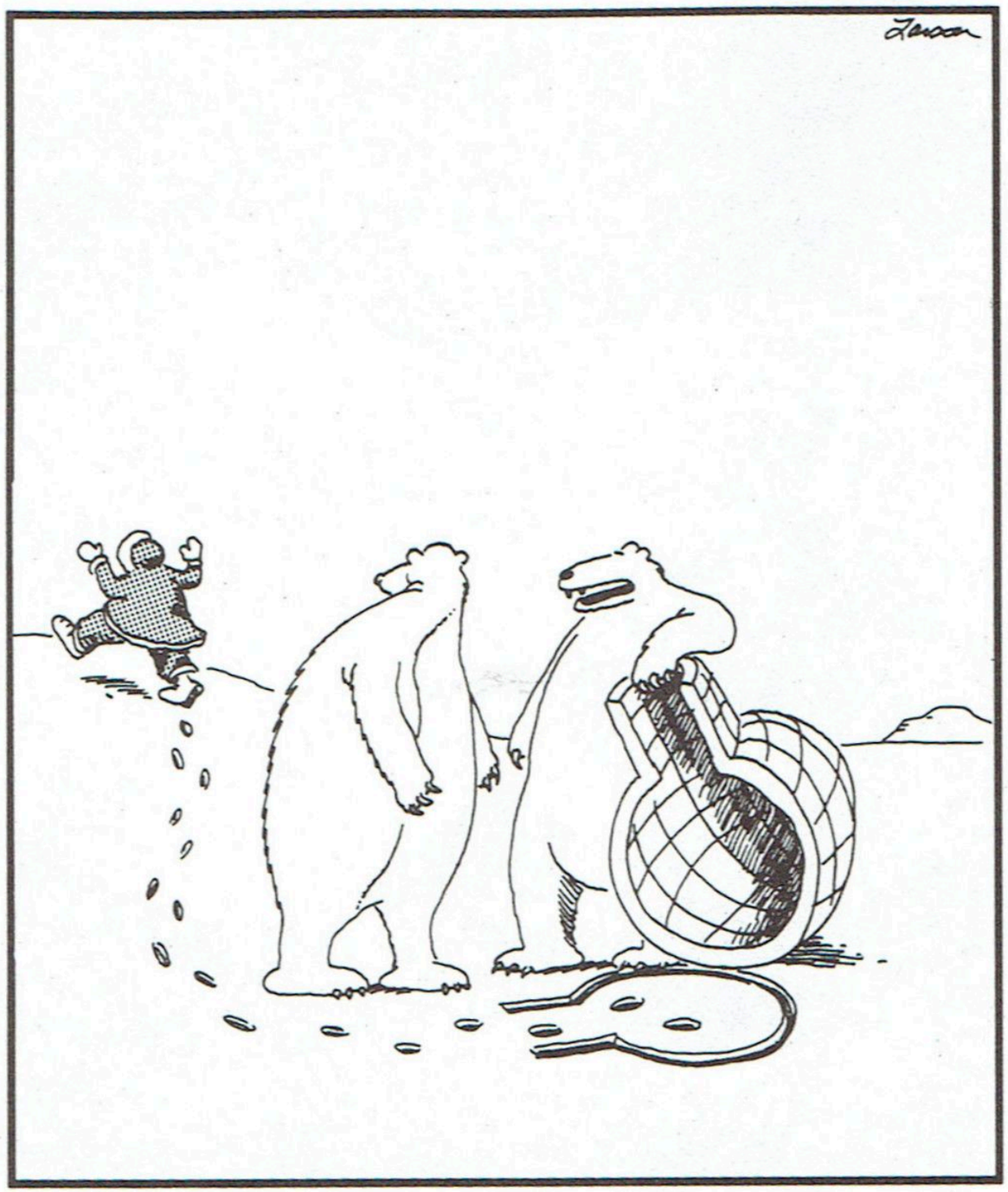
Example: WEP

- Stronger version had a 104-bit key
  - Should require 2^104 calculations to crack
  - In 2001, the FMS attack cracked it in 1.5 million operations (2^20.5)
  - In 2007, the PTW attack worked in 40,000 operations (2^15.3)
    - Link Ch 3a

Example: Substitution Cipher

- Keyspace is
  - 26 x 25 x 24 x ... x 3 x 2 x 1 = 26!
  - = 4 x 10^26 = 2^88
- But children crack it with no computers for fun with chosen-ciphertext guessing attacks

"QDHH, ALLI BDWPDKT! C FWK'Z XDHCDPD ULY JDK. ... C'PD ALZ TLJD SLMD!"



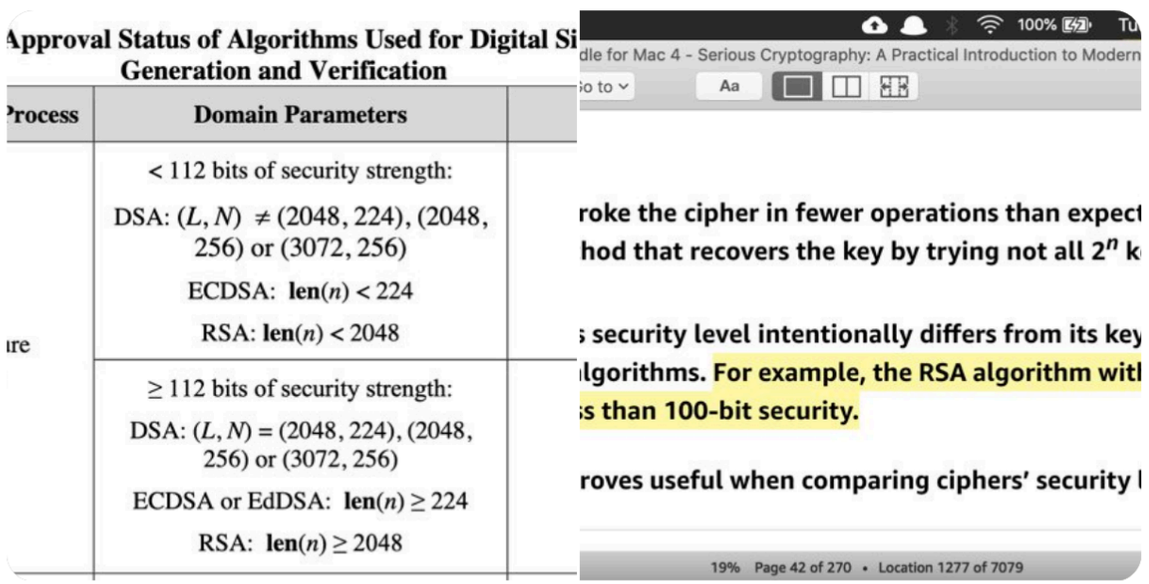
G KGBP, WNQ CZFI...UFY PEFP DNMDASP HQYP F KGPPKA PNN DNLSKAV, DFZK?

Example: RSA-2048

- Key is 2048 bits long, but provides 112 bits of security
- Because key is not simply a random number, but the product of two primes
- Textbook says "< 100 bits", which is a more conservative estimate

Sam Bowne @sambowne · 12m

@veorq Thanks for "Serious Cryptography" -- my class and I love it! Could you please explain how secure RSA-2048 is? Is it <100 bits or 112 bits?
csrc.nist.gov/publications/d… nostarch.com/seriouscrypto#…



Approval Status of Algorithms Used for Digital Si
Generation and Verification

Process	Domain Parameters	
	< 112 bits of security strength: DSA: $(L, N) \neq (2048, 224), (2048, 256)$ or $(3072, 256)$ ECDSA: $\text{len}(n) < 224$ RSA: $\text{len}(n) < 2048$	
	≥ 112 bits of security strength: DSA: $(L, N) = (2048, 224), (2048, 256)$ or $(3072, 256)$ ECDSA or EdDSA: $\text{len}(n) \geq 224$ RSA: $\text{len}(n) \geq 2048$	

ıre

dle for Mac 4 - Serious Cryptography: A Practical Introduction to Modern
Go to
Aa

roke the cipher in fewer operations than expect
hod that recovers the key by trying not all 2^n k

s security level intentionally differs from its key
lgorithms. For example, the RSA algorithm witl
s than 100-bit security.

roves useful when comparing ciphers' security l

19% Page 42 of 270 • Location 1277 of 7079

2

mjos\dwez @mjos_crypto · 2m

Replying to @sambowne @veorq

The O factor in factorization algorithms is difficult to gauge and not all steps of the algorithm are wonderfully parallelizable. NIST clearly put the round number 2048 at the 3DES security level because those just happened to be the two minimum security things that they had.

JP Aumasson @veorq · 2m

Replying to @sambowne

Thanks! No simple answer here, hard to quantify, different sources/people will give different answers, so I preferred to be conservative based on the information I had and the experts I talked with :)

NIST SP 800-57

Security Strength	RSA key length
<= 80	1024
112	2048
128	3072
192	7680
256	15360

Security Strength	Through 2030	2031 and beyond
< 112	Disallowed	Disallowed
112	Acceptabe	Disallowed
128	Acceptable	Acceptable
192	Acceptable	Acceptable
256	Acceptable	Acceptable

- Link 3g

Full Attack Cost

- Parallelism
- Memory
- Precomputation
- Number of Targets

Parallelism

- 2^56 sequential steps are much slower than
- 2^56 independent steps
- If you can use many processors

Memory

- Some algorithms take a lot of memory
- Memory is much slower than registers
- Ethereum mining is *memory-hard* (link Ch 3i)

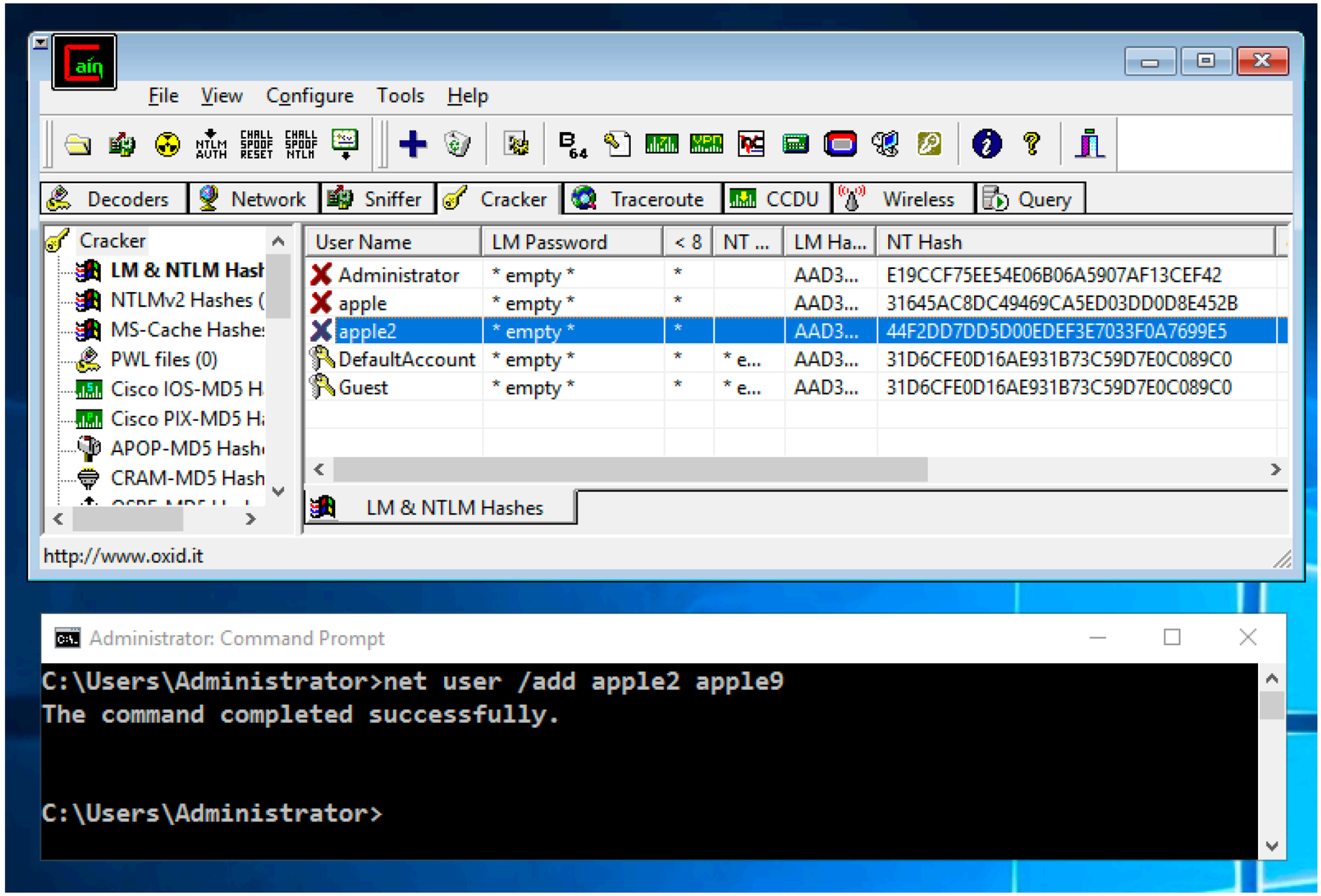
April 28th, 2017 / By Vijay Pradeep / Categories: Cryptocurrency, Tech
14 Comments

Ethereum's Memory Hardness Explained, and the Road to Mining It with Custom Hardware

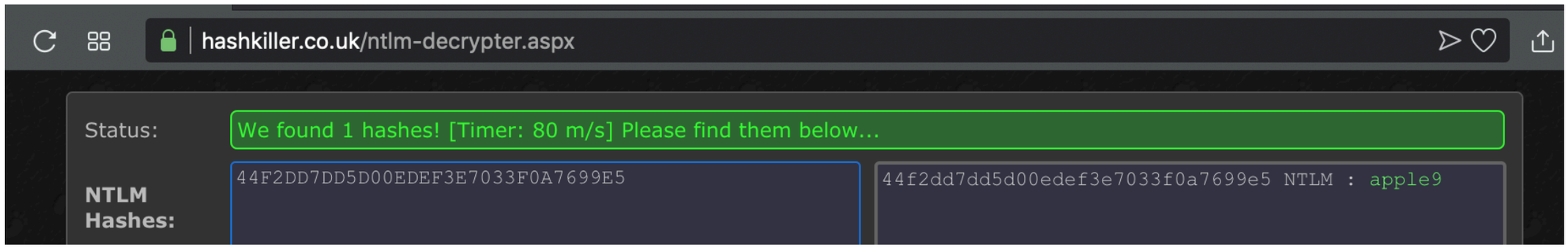
Precomputation

- Some operations only need to be performed once
- Can be reused
- Time-memory trade-off attack
  - Also called **rainbow tables**
  - After tables are calculated, further attacks are fast

Example: Windows Password Hashes



Example: Windows Password Hashes



- Fast, free cracker
- Link Ch 3j

Number of Targets

- Example: you have 1000 password hashes to crack
- You will find one of them in 1/1000 the time it takes to crack them all

Choosing and Evaluating Security Levels

- Usually only two choices
- **128 bits** for normal use
- **256 bits** for high security


Kahoot!

Achieving Security

Building Confidence

- **Provable security**
  - Mathematical proof
- **Heuristic security**
  - Evidence of failed attempts to break it

Provable Security

- Proof that breaking the crypto scheme is at least as hard as a known hard problem
  - RSA requires factoring a number into its component primes
  - A famously hard problem, studied for thousands of years
- *Note: there's no proof that "hard" math problems are actually hard. It could be just that no fast solution has been found yet.*

RSA Algorithm

- Choose two random, large prime numbers **p** and **q**
- Calculate **n** = **p** x **q**
- Choose an exponent **e** (usually 65537)
- **Public Key** is (**n**, **e)**

RSA Algorithm

- To encrypt a plaintext message **x**

  y = x^e mod n

- To decrypt a ciphertext **y**

  x = y^d mod n

- **d** is the decryption key, and is calculated from **p** and **q**
- **n** is public, but no one can find **p** or **q**
  - Because factoring numbers is a hard problem

Proofs Relative to Another Crypto Problem

- Compare one crypto scheme to another one
- Prove you can only break the new one if you can break the old one
- This is used to derive new crypto schemes from old ones
- Such as developing symmetric algorithms, random bit generators, and hash functions from the same permutation algorithm

Caveats

- The underlying system might be broken
- The proof might overlook some other attack against a real implementation
- The proof might contain an error

Examples

- RSA is easy to crack if **p** and **q** aren't sufficiently random
- Knapsack, by Merkle and Hellman, was later totally broken by lattice reduction
- Side-channel attacks may reveal the key
  - Without breaking the math
  - By measuring radio waves, power consumption, timing, etc.

Examples

- Implementation may use the cryptography incorrectly
- True of most Android apps I've studied

Password Stored with Reversible Encryption

Home Depot

Kroger

Safeway

Walgreens

Heuristic Security

- Most symmetric ciphers don't have a security proof
- Including AES
- But ***"skilled people have tried to break it and failed"***

Security Margin

- Cryptanalysts often attack ***simplified versions*** of a cipher, with fewer rounds
  - They find how many rounds they can crack
  - The difference between the actual number of rounds in the full cipher and the number of rounds they can crack is the **security margin**

AES Margin of Security

AES is the best known and most widely used block cipher. Its three versions (AES-128, AES-192, and AES-256) differ in their key sizes (128 bits, 192 bits and 256 bits) and in their number of rounds (10, 12, and 14, respectively). In the case of AES-128, there is no known attack which is faster than the 2^{128} complexity of exhaustive search. However, AES-192 and AES-256 were recently shown to be breakable by attacks which require 2^{176} and 2^{119} time, respectively. While these complexities are much faster than exhaustive search, they are completely non-practical, and do not seem to pose any real threat to the security of AES-based systems.

The attack only breaks 11 rounds of AES-256. Full AES-256 has 14 rounds.

- Link Ch 3k

Generating Keys

Three Ways to Generate Keys

- **Randomly** using a pseudorandom number generator
- **From a password** using a Key Derivation Function
- **Key agreement protocol** using a series of messages between parties to establish a shared key

Generating Symmetric Keys

- They are simple random numbers
- The same length as the security they provide
- For 128-bit security, generate 16-byte number

```
root@kali:~# openssl rand -hex 16
7c7888fa3524aa41e7d3998c4ec0c42b
root@kali:~#
```

Generating Asymmetric Keys

- Feed random numbers into a ***key generation algorithm***
- It's complex. For RSA
  - Must make many guesses to find a prime number **p**
  - Repeat for **q**

Demonstration

- In Python
- RSA key generation takes a long time

```
>>> from Crypto.PublicKey import RSA
>>> key = RSA.generate(2048)
>>> key = RSA.generate(4096)
>>> key = RSA.generate(8192)
```

Protecting Keys

- Key wrapping
  - Encrypting the key with a second key
  - Second key often generated from a password
- On-the-fly key generation from a password
  - No need to store the key at all
- Storing keys on a hardware token
  - Like a smartcard or USB dongle
  - Password-protected

How Things Can Go Wrong

Incorrect Security Proof

- RSA-OAEP was "proven"
- Proof shown to be incorrect
- But the actual implementation was secure by accident (link Ch 3I)

Short Keys for Legacy Support

- "Export" cryptography was deliberately weakened by US regulations in the 1990's
  - Including 512-bit RSA
- Many servers continued to support these weak protocols in 2015
  - Link Ch 3m


Kahoot!