jung gur shpx vf
pelcgbtencul

what the fuck is cryptography

by @wtfsper

If you are here for cryptocurrency

wrong room

also, you doing ok?

Let’s talk about what’s not cryptography

pretty logical way to start……

Binary

How to count

0,1, 2,...,9, 10

How to count (Binary)

0,1,10,11,100

Hexadecimal

How to count (Hexadecimal)

0...9,A,B,C,D,E,F,10

Hexadecimal need to knows

Usually will start with 0x

Can represent any byte (0x00-0xFF)

Base64

Bas64 is an encoding

Not secure
Easily reversible
Case sensitive
Binary-to-Ascii encoding

How to spot Base64

All in ascii
Usually ends in =
Perfect for representing binary data
(ssh keys, data out of ascii range, etc…)

Example Base64

d2hhdCB0aGUgZnVjayBjdWNrcwo=

How to encode/decode base64

```
echo “Asper is the best officer” | base64

> QXNwZXIgaXMgdGhlIGJlc3Qgb2ZmaWNlcgo=

echo “QXNwZXIgaXMgdGhlIGJlc3Qgb2ZmaWNlcgo=” | base64 -D

> Asper is the best officer
```

Hashing

What is hashing

One-way function and not reversible

Hashing Algorithms

- MD5 💩 💩 💩 💩 💩
  - 16 bytes
  - incredibly broken: rainbow table or GPU bruteforce
- SHA-1 💩 💩 💩
  - 20 bytes
  - "broken": rainbow table or GPU bruteforce
- Any other hash function is safe and shouldn’t be your point of attack unless the challenge is mean
  - SHA-256/512, Whirlpool, SHA-3, etc…
  - Can attack with *length extension attacks* but out of scope

Lets start simple

Caesar Cipher

Rating: 💩 💩 💩 💩 💩

How it works

```
plain_text = “hello world”

key = 5

cipher_text = “”

for c in plain_text:
    cipher_text += chr(((ord(c)- 97) + key % 26) + 97)

print(cipher_text)

> mjqqt btwqi
```

Methods of Attack

try every shift ⭐⭐⭐⭐⭐

always work...

Substitution Cipher

Rating: 💩 💩 💩

How it works

```
plain_alphabet = “abcdefghijklmnopqrstuvwxyz”

cipher_alphabet = “udekyjhmltswpqgrozicvxnabf”

plain_text = “acm is a shadow organization”

cipher_text = encrypt(plain_alphabet, cipher_alphabet, plain_text)

print(cipher_text)

> uep li u imukgn gzhuqlfuclgq
```

Methods of Attack

Letter statistics ★★½

Pretty old cool, but can work

Word statistics ★★★½

Little bit harder to do, but a lot better

Fancier Algorithms

Genetics ★★★★½

Neural Networks ★★★★★

One-time Pad

Rating (in theory): ⭐⭐⭐⭐

Rating (in practice): 💩

Crib Dragging

- Remember when I said garbage in practice, here's one example:

```
message1 = "Hello World"
message2 = "the program"
key = "supersecret"

cipher-text1: "3B101C091D53320C000910"
cipher-text2: "071D154502010A04000419"
```

The english language isn't perfectly random, so we can actually guess words that we think are in the messages. "the" is a perfect example.

```
       3C0D094C1F523808000D09
XOR    746865 (the)
-------------------------------
       48656C (Hel)
```

Possible words:

- Help
- Hell
- Hello

```
       3C0D094C1F523808000D09
XOR    48656C6C6F (Hello)
-------------------------------
       7468652070 (the p)
```

Possible words:

- the person
- the petition
- the program

Rinse and Repeat

One-Time Pad weaknesses

- The key must be longer or equal to the length of the message
  - This is usually unlikely and you can build a frequency table
- The key must also be completely and truly random
  - This is unlikely as well and there are more sophisticated attacks against

RNG

Random number generation

Rating: ????

Linear congruential generator

Common, but not cryptographically secure:

$$X_{n+1} = (aX_n + C) \pmod{m}$$

Official C implementation:

```
static unsigned int seed = 1;

void srand (int newseed) {
    seed = (unsigned)newseed & 0x7fffffffU;
}

int rand (void) {
    seed = (seed * 1103515245U + 12345U) & 0x7fffffffU;
    return (int)seed;
}
```

a = 1103515245
c = 12345
m = 2^{31}
X_0 = 1

Methods of Attack

Solve for seed ⭐⭐⭐⭐⭐

Kind of mathematically hard to understand, but tools already exist

Read more here:

Predicting the next Math.random() in Java by Franklin Ta (2014)

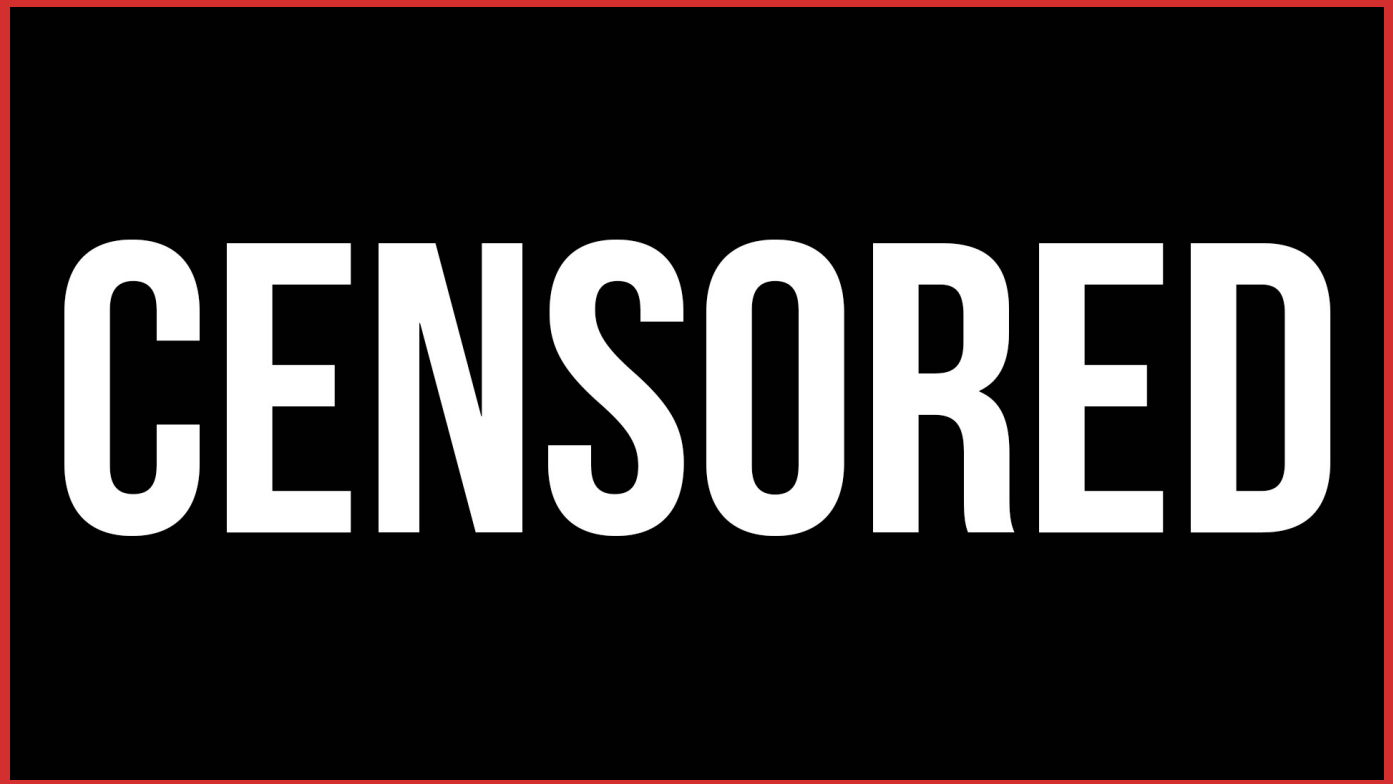
https://bit.ly/2NlokDC – Article

https://bit.ly/2EsSLyj – Automated Tool

https://bit.ly/2CGxbEq – Archive

Mersenne Twister

More random, more common, but not cryptographically secure



Methods of Attack

Solve for seed ⭐⭐⭐

wait isn't this the same thing

yes...but no. You need a LOT more numbers (624) to predict a mersenne twister number

Fortunately tools exist!

- nh2/AntiMersenne
- kmyk/mersenne-twister-predictor

RC4

ARCFOUR

Rating: 💩 💩

RC4

- It's a stream cipher
- Designed by the people who made RSA
- Pretty fast

How it works

```
key = "Secret"

*keystream = KSA(key)

plain_text = "attack at dawn"

cipher_text = encrypt(keystream, plain_text)

print(cipher_text)

> 45A01F645FC35B383552544B9BF5
```

Methods of Attack

Bruteforce ⭐⭐⭐

You don't need to understand much, but ARCFOUR has not been popular due to it's ease at bruteforcing if the entropy isn't high enough (which is quite common since the algorith doesn't support high entropy)

For bruteforcing, you can find tools online or just bruteforce yourself

RSA

Rating: ⭐⭐⭐⭐

How it works

```
q = generatePrime() # prime factor 1
p = generatePrime() # prime factor 2
n = pq #modulus
phi = ((p-1)*(q-1)) # just a helping variable
e = findCoPrime(phi) # public exponent
d = inverse(e, phi) # private exponent
private_key = (d, n)
public_key = (e, n)
plaintext = "william sucks"
cipher = [(ord(char) ** d) % n for char in plaintext]
```

RSA Attacks...

Weak public key factorization

If n is small, you can find p and q, which lets you get d.

`n = 10142789312725007`, `e = 5` (we always know e)

Then, take the `sqrt(n) = 100711415`.

Keep trying odd numbers below the sqrt. You'll hit a factor, here we hit `100711409` very quickly. Therefore, `p = 100711409` and `q = n/p = 100711423`. Finally, we can find d with `d = e^-1 mod (p-1)*(q-1) = 811423128904174`1

Wiener's Attack

If the private exponent is small enough $d < \frac{1}{3} N^{\frac{1}{4}}$, you can actually find it using Wiener's attack.

The math of the attack is pretty out of scope, but there are automated resources to check and crack these types of mistakes

Low Public Exponent Attack

If e is small, there are many ways to recover the plain text, but not break RSA (this means no recovery of d)

Small e attacks:

- Coppersmith's attack
- Hastad's attack
- Franklin-Reiter Related Message Attack

There are so many attacks

Multikey attack:

If someone encrypts something with lots of keys, you can use those keys to factor each other

Fermat's attack:

If p and q are too close to each other, you can recover

Bad Primes:

Mersenne Primes, primes on FactorDB, p/q is a small fraction

Less common in CTFs

Discrete Logarithm Problem

$$g^x = h \pmod{p}$$

Diffie-Hellman Key Exchange
Elgamal Public Key Cryptosystem
Digital Signatures
Ellipitical Curves (kinda)

Methods of Attack

Simple Bruteforce ⭐

You can brute force, but most likely will not be able to crack

Baby-step Giant-step ⭐⭐

This is essentially a more efficient algorithm for brute forcing

Pollard's kangaroo algorithm ⭐⭐⭐

Will work with any finite cyclic group

Homebrew Garbage

First, never roll your own crypto

Second, NEVER roll your own crypto

If you ever encounter some custom encryption problem, it's usually more likely a reversing engineering challenge, but the implementation is more likely to be wrong.

I say this with personal experience grading AES implementations...

AES

If we're being honest, AES is pretty safe, except for one attack you may encounter.

There are two modes, ECB and CBC. CBC essentiallys adds a "random" element into the algorithm, while ECB can be fought against with known plaintext.

If you know for a fact that it's ECB AES, and you have a collection of plaintext and outputs, you can figure out the key with some cryptoanalysis (tools exist)

Thank you for listening