Cryptography

Intro to Cryptography

Basic Ciphers

Hashing

Password Security

DISCLAIMER

- All of the material in this school can be used for good (testing, research, educating), but also for bad
- We use our skills and knowledge responsibly and ethically
- We recommend you do the same
- We are not responsible for anything you do as a result of these lessons

What is Cryptography

Cryptography is the practice and study of techniques for secure communication in the presence of third parties called adversaries.

The “*secure communication*” notion can be extended, and modern cryptography also encompasses fields such as electronic voting, decentralized cryptocurrencies etc.

Cryptography is pretty much omnipresent in computer systems nowadays.

It’s been around for a while

Cryptography was used as a means of secure communication since the ancient times, mainly for military purposes.

Spartan scytale, a *transposition cipher*:

Now cryptography is even more important. Since our life is organized around the communication of computer systems, the data transmitted should be confidential and untamperable.

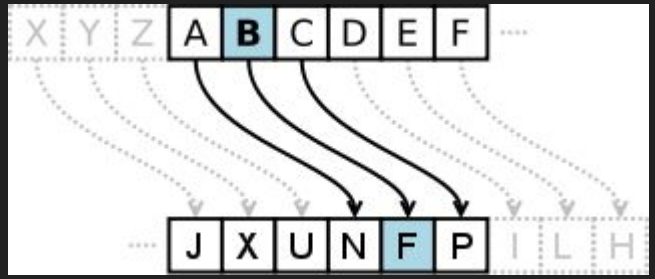
And has greatly evolved

Cryptography has evolved from simple substitution and rotation ciphers, which can be broken via the analysis of letter frequency.

To methods such as Enigma which was used in the world war (which can also be broken)

And nowadays we use encryption methods such as hashing, AES, RSA and many more.

Simple Ciphers

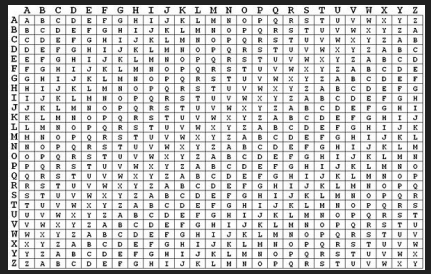


Substitution - replaces each letter of the alphabet by a different character.

Caesar - a substitution cipher where the letters of the alphabet get shifted

Rot13 - type of caesar cipher where all the letters in the alphabet get shifted by 13 characters.

Vigenere - each letter is shifted by a different amount, which is usually determined by a key



Hashing

Hashing is a method commonly used nowadays.

Sometimes we need to shrink a large message to an “equivalent” but smaller version. Think of it like taking a message’s “fingerprint”.

Other times we need to commit (tell everyone that we know) a message, but not immediately publish the message itself.

We may also want to use functions that cannot be easily inverted, or create a scheme that verifies that a message comes from a trusted source (authentication).

Hashing

Hashes (or hash functions) are functions that map data of arbitrary size to a fixed size.

For example, they can map one 1GB file to a 256-bit number.

Their properties:

- They are *deterministic* (Same input message means same output hash every time).
- Given a hash, it should be hard/impossible to compute the message.
- Two almost-identical messages should produce completely different hashes.
- Finding two messages with the same hash value should be computationally hard.

Example hashes

Here are some example hash functions. MD5 and SHA1 can be broken, it's better to use hashes from the SHA2 or SHA3 family (like SHA256, which is a SHA2 hash).

Given these hashes, one should not be able to find the preimage (the original message) without trying all the possible messages.

```
bash-4.4$ echo -n "123456" | md5sum
e10adc3949ba59abbe56e057f20f883e  -
bash-4.4$
bash-4.4$ echo -n "123456" | sha1sum
7c4a8d09ca3762af61e59520943dc26494f8941b  -
bash-4.4$
bash-4.4$ echo -n "123456" | sha256sum
8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92  -
bash-4.4$
```

Example hashes

A small change in the message (fox -> fpx) results in a completely different hash.

```
bash-4.4$ echo -n "The quick brown fox jumps over the lazy dog" | sha256sum
d7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592  -
bash-4.4$
bash-4.4$ echo -n "The quick brown fpx jumps over the lazy dog" | sha256sum
f72e2e8db512f2474c9d2a53ec55eb5c9ab728b80e65e3fc2a63cd95cb559c25  -
```

Hashes can be calculated easily in all major programming languages.

```
In [1]: import hashlib

In [2]: message="The quick brown fox jumps over the lazy dog"

In [3]: print(hashlib.sha256(message).hexdigest())
d7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592
```

Where will we meet hashes?

- Used to check file integrity. A file is transmitted correctly from A to B, if the hash that A calculates is the same as the one that B calculates.
- Used to provide authentication (in HMACs). Only parties who know a specific key are able to calculate a hash.
- For secure password storage
- For so many other things. They have been called “the workhorses of modern cryptography”.

Secure password storage

Passwords in databases should NEVER be stored in plaintext. A database leak will compromise everyone (and the other accounts in which they have the same password).

1st idea: We should store the hash of the password. And at the time of login, we should check the hashes for equality.

But it's not expensive to *brute force* each password, modern graphics cards can calculate millions of hashes per second.

Secure password storage

2nd idea: The password hashes should be the result of the many iterations of the hash function on the original password. SHA256(SHA256(....SHA256(password)...)) .

But what if someone precomputes all possible 8-character passwords (it might take months but that's ok), and then simply compares the leaked hashes with the ones that he has found? Also, what if two users have the same password?

3rd idea: Use *Salts*, i.e. random values (different for every password stored) that are included in the hash. SHA256(SHA256(.... SHA256 (salt || password) …)).

Modern programming languages provide such an option, eg PHP's password_hash() function.

But how do we break them to win the CTFs?

John The Ripper (aimed for CPU-based attacks)

Hashcat (aimed for GPU-based attacks)

```
root@kali:/tmp# echo -n "rusty100" |md5sum | tr -d '-' > unknown_md5
root@kali:/tmp# john --wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-MD5 unknown_md5
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
rusty100         (?)
1g 0:00:00:00 DONE (2018-09-27 10:21) 7.692g/s 10213Kp/s 10213Kc/s 10213KC/s rusty129..rusty-
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Challenges

https://school.sigint.mx

Useful links:

Crackstation <- online cracked hashes

CyberChef

https://www.openwall.com/john/

https://countuponsecurity.files.wordpress.com/2016/09/jtr-cheat-sheet.pdf

https://hashcat.net/hashcat/