Hacktales

Password Cracking: Cryptography & Hashing

WEEK5

PASSWORDS & THEIR SIGNIFICANCE

WHAT ARE PASSWORDS

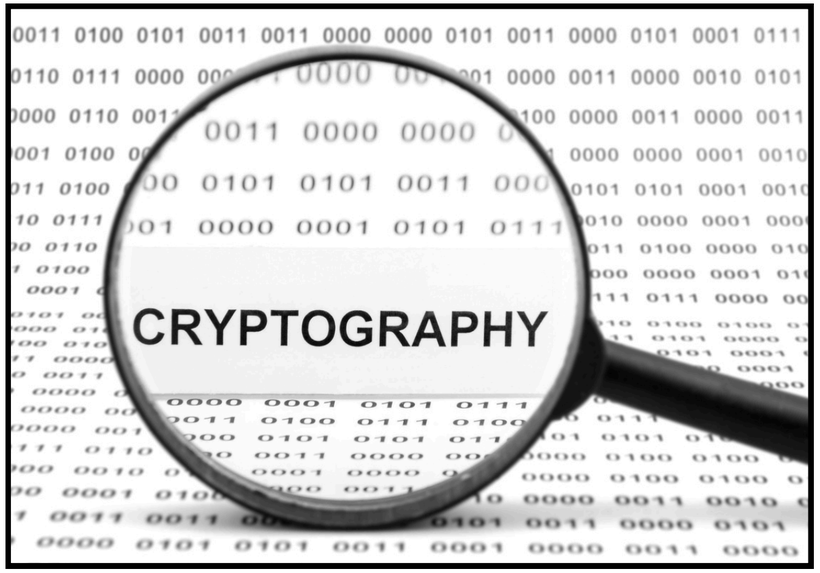
A password is a secret sequence of characters used to authenticate a user's identity and authorize access to computer systems, websites, mobile devices, or other digital resources. It is typically paired with a username to confirm the user's identity during the authentication process.

WHY PASSWORD SECURITY MATTER

80% of security breaches involve passwords, and weak passwords lead to data leaks and unauthorized access. Example breaches: LinkedIn (2012), RockYou (2009) Rockyou was storing passwords in plaintext along with connected account passwords such as facebook etc in plaintext, a hacker used a 10year old SQL injection vulnerability to compromise over 32million user passwords.

INTRODUCTION TO CRYPTOGRAPHY

OVERVIEW



CRYPTOGRAPHY

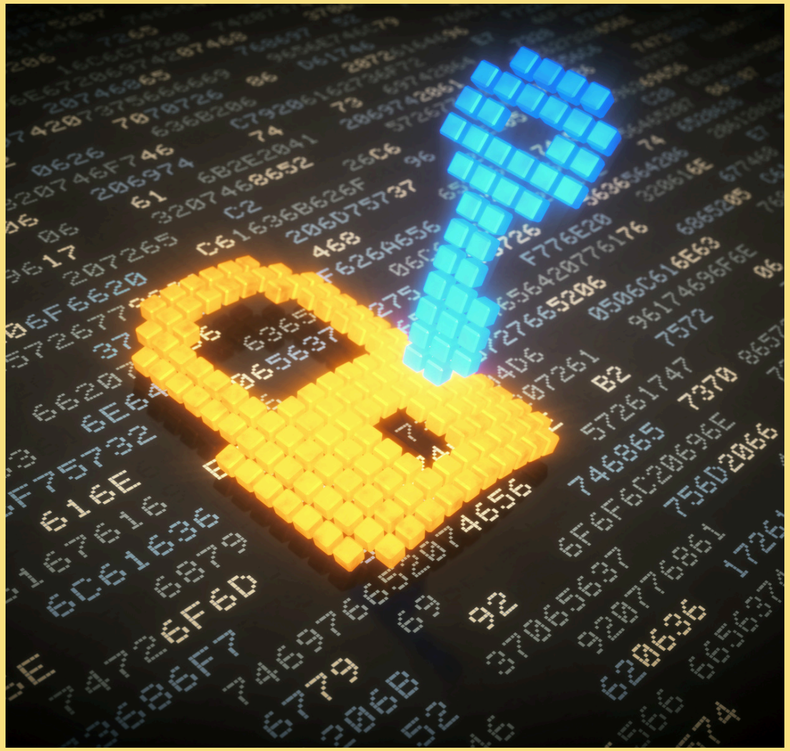
Cryptography ensures information confidentiality by converting it into a secure format known as a ciphertext. Cryptography is the practice of securing communication and data in the presence of adversaries. It involves techniques for transforming information into a form that is unreadable to unauthorized parties, while still allowing authorized recipients to recover the original data. There are 2 types of cryptography: Symmetric Encryption (Private Key Encryption) Asymmetric Encryption (Public Key Encryption)

SYMMETRIC ENCRYPTION

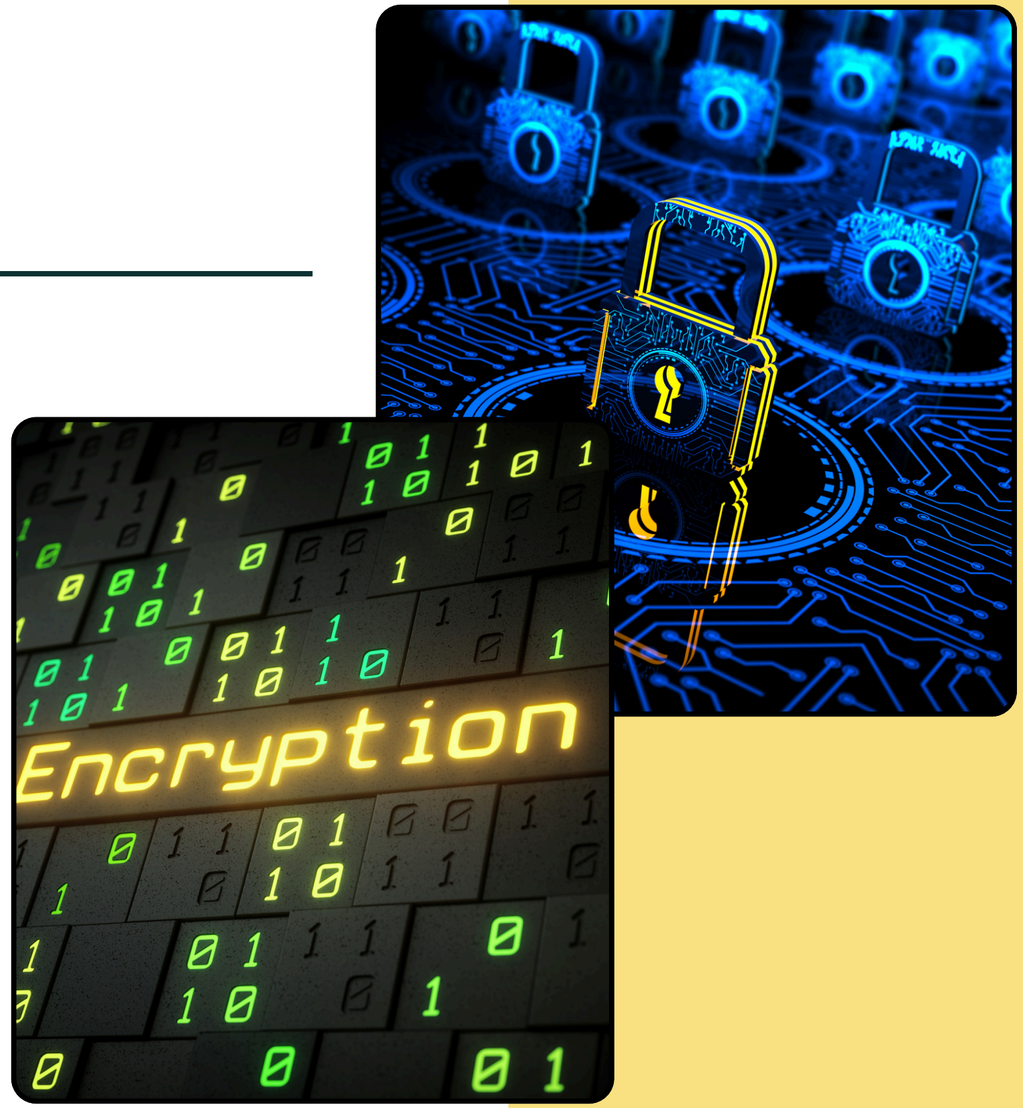
OVERVIEW

WHAT IS SYMMETRIC ENCRYPTION

Symmetric Cryptography is also known as Private Key Encryption, symmetric algorithms use a key or secret to encrypt the data and use the same key to decrypt it. A basic example of symmetric encryption is XOR, other common algorithms are AES (Advanced Encryption Standard), DES, 3DES, Blowfish.Its main use cases are in Encrypting files, securing VPNs, disk encryption and its main advantage is its speed and efficiency in encrypting large data with Key distribution as a major challenge



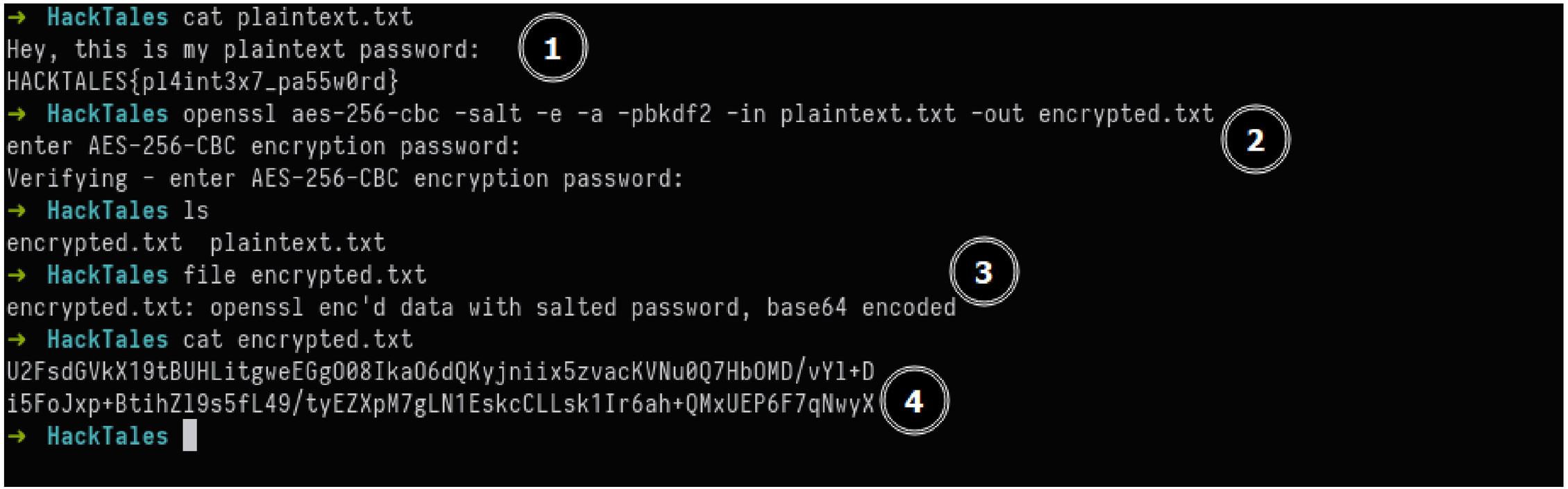
ASYMMETRIC ENCRYPTION



WHAT IS ASYMMETRIC ENCRYPTION

Assymmetric algorithms divide the key into two parts (i.e., public and private). The public key can be given to anyone who wishes to encrypt some information and pass it securely to the owner. The owner then uses their private key to decrypt the content. Some examples of asymmetric algorithms are RSA, ECDSA, and Diffie-Hellman.

Demo: Encrypting Files with Symmetric Encryption (AES)



Symmetric encryption on a file containing our password Here we are using the openssl library utility and the AES 256 CBC encryption algorithm alongside the password based key derivation function mode to reiterate the encryption and passing the file to the -in parameter and outputing the resulting encrypted content to a file called “encrypted.txt”

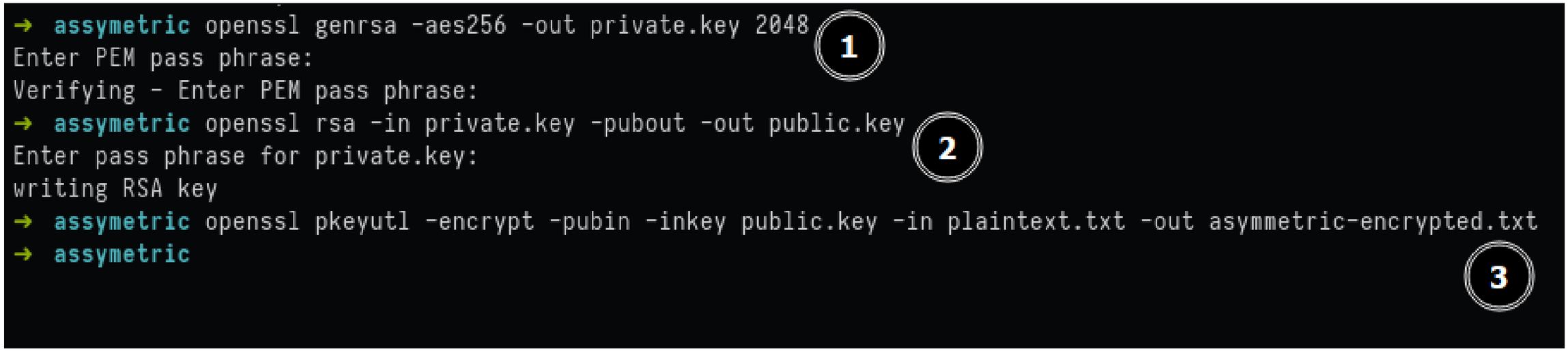
Decrypting Symmetrically Encrypted Files (AES)

```
➜ HackTales openssl aes-256-cbc -salt -d -a -pbkdf2 -in encrypted.txt -out unencrypted.txt
enter AES-256-CBC decryption password:
➜ HackTales cat unencrypted.txt
Hey, this is my plaintext password:
HACKTALES{pl4int3x7_pa55w0rd}
➜ HackTales
```

In the same light, we just need to specify that we want to decrypt the file with “-d” along with the encrypted file with “-in” and the resulting output with “-out” and then provide our decryption key at the password prompt

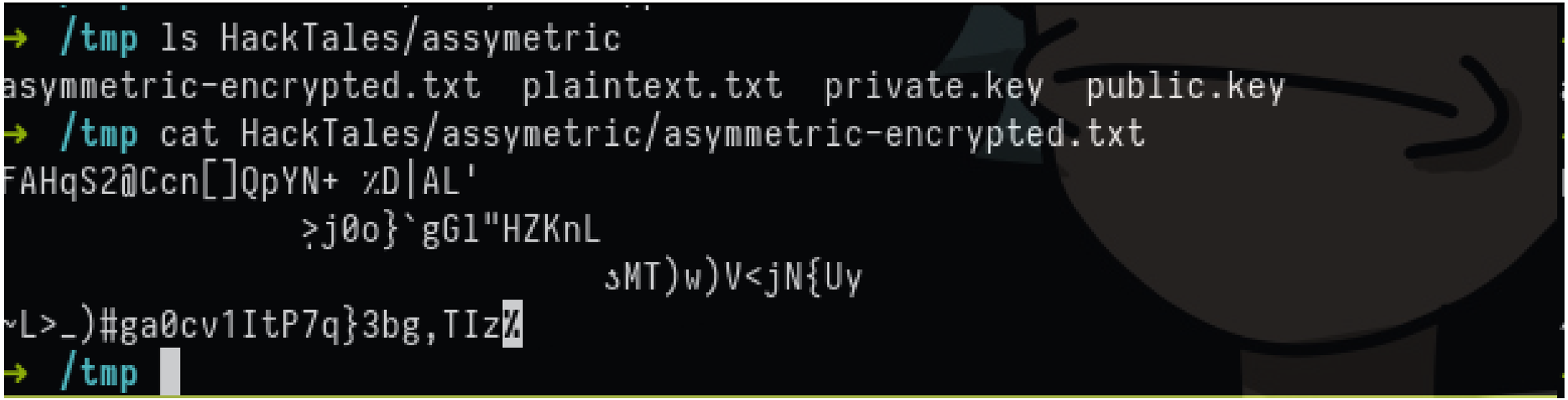
Demo: Encrypting files with Asymmetric Encryption (RSA)

For Asymmetric encryption you must first generate your private key(1) and extract the public key.(2)



Now we have our private and public keys, we can encrypt a file using the public key as shown in (3)

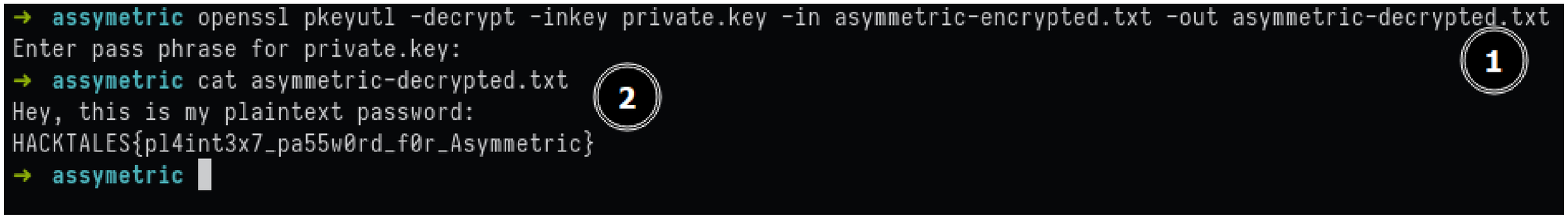
Content Of Encrypted Files



The content of encrypted files (ciphertext) appear unintelligible, unless we possess the decryption key, it will remain unreadable hence ensuring confidentiality of data

Decrypting Asymmetric encrypted files (RSA)

To decrypt, we will enter our passphrase in the prompt



The content of the decrypted file is now readable by the application of the private key on the encrypted file passed to the “-in” parameter and is saved to the “asymmetric-decrypted.txt”

Only people we share our private key with will be able to decrypt the content of our encrypted file.

HASHING

OVERVIEW

WHAT IS HASHING

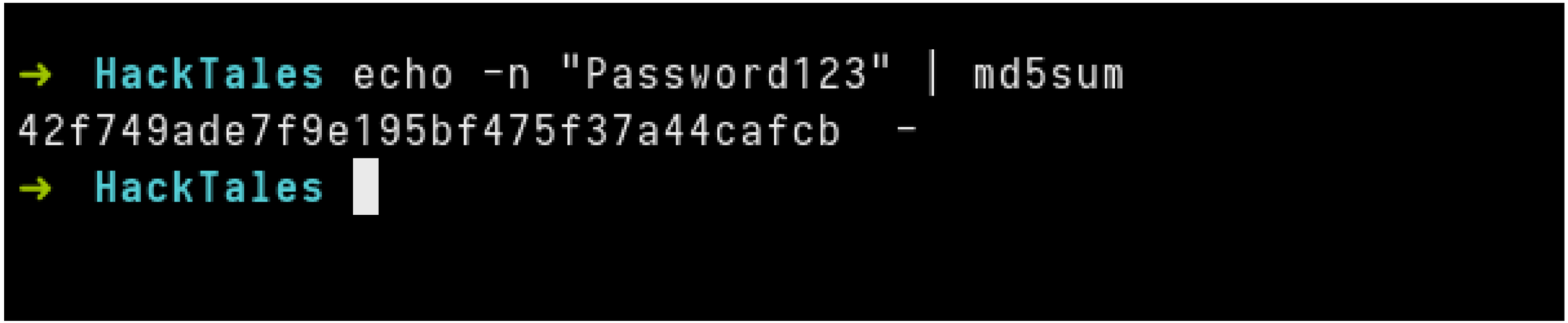
While hashing is not a type of cryptography in the traditional sense, it is closely related to cryptography Hashing is the use of a one-way cryptographic function that converts any input (like a file, password, or message) into a fixed-size string of characters, often represented as a hexadecimal number. The output is called a hash value, digest, or checksum.

Examples of hashing algorithms

MD5: 42f749ade7f9e195bf475f37a44cafcb
NTLM: 58a478135a93ac3bf058a5ea0e8fdb71
SHA1:
b2e98ad6f6eb8508dd6a14cfa704bad7f05f6fb1
SHA256:
008c70392e3abfbd0fa47bbc2ed96aa99bd49e1597
27fcba0f2e6abeb3a9d601
SHA512:
804f50ddbaab7f28c933a95c162d019acbf96afde56d
ba10e4c7dfcfe453dec4bacf5e78b1ddbdc1695a793b
cb5d7d409425db4cc3370e71c4965e4ef992e8c4

Demo: Crafting Hashing on Linux

Some hashes can be created with Linux utilities like the MD5 and SHA family, some with python libraries like the hashlib and binascii libraries, lets craft an MD5 hash of the password “Password123”



“-n” ensures that no whitespaces are printed after echoing the text, because even the slightest change in the input will result in a large difference in output, a completely different hash.
Only people we share our private key with will be able to decrypt the content of our encrypted file.

In the same way we can craft a sha256 hash

```
➜  HackTales echo -n "Password123" | sha256sum
008c70392e3abfbd0fa47bbc2ed96aa99bd49e159727fcba0f2e6abeb3a9d601  -
➜  HackTales
```

As well as a sha512 hash:

```
➜  HackTales echo -n "Password123" | sha512sum
804f50ddbaab7f28c933a95c162d019acbf96afde56dba10e4c7dfcfe453dec4bacf5e78b1ddbdc1695a793bcb5d7
d409425db4cc3370e71c4965e4ef992e8c4  -
➜  HackTales
```

Linux utilizes the SHA512crypt algorithm until recently a switch was made to use YesCrypt as it is a more secure hashing algorithm these hashes are stored in the /etc/shadow file and it looks like this:
yj9T$P9NwRV7M8BTTSBhatZPP61$PY5mtxgP8IwpLuaGhZg7kYkqODDgRAxXnfyVUtXXuL9

PURPOSE OF HASHES

Hashing is employed in secure password storage and ensuring data integrity. Some commonly used hashing algorithms include MD5, SHA256, SHA512, Bcrypt, NTLM, YesCrypt etc

Hashes are one way in the sense that they cannot be reversed, however hashed passwords can be “cracked” in a certain way, by comparing the resulting hash of a guessed password with the acquired password hash and if they are a match you have the plaintext of the digest.

PASSWORD CRACKING OVERVIEW

What is Password Cracking

Password cracking refers to the process of recovering or uncovering passwords stored in computer systems, either as plaintext or hashed values. Ethical hackers use this technique to test the strength of passwords and identify weak points in security systems before malicious attackers exploit them

Common Password Cracking Techniques

Brute Force Attack

Dictionary Attack

Rainbow Table Attack

Social Engineering

Hybrid Attacks

HASH CRACKING

CRACKING HASHES WITH JOHNTHERIPPER & HASHCAT

CRACKING HASHES WITH JOHNTHERIPPER

From our understanding of Hashes, they are one way, meaning they cant be reversed hence cant be “cracked” but cracked in this sense means we can effectively retrieve the plaintext of a hashed password if the password is weak by utilizing a wordlist of passwords which a tool such as hashcat or JohnTheRipper(JtR) will try to process and hash each word within and compare it with the hash that was submitted, the hash of a plaintext within the wordlist of passwords that matches is our recovered password.

Demo: Identifying Unknown Hashes

Sometimes we might acquire odd hashes which we have never come across before, and to crack them we might need to specify their format to john or hashcat, we can use tools like “hashid” to identify these hashes first like this:

```
→ /tmp hashid '42f749ade7f9e195bf475f37a44cafcb'
Analyzing '42f749ade7f9e195bf475f37a44cafcb'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5
[+] Skype
```

Demo: HASH CRACKING WITH JtR

CONTINUATION

JtR is a CPU based offline password cracking tool, in contrast to hydra which does, all JtR requires to crack a password is a wordlist of passwords (which Rockyou.txt is sufficient for) and the hash as well as the format/hash algo you which to crack. The name of the hash format we wish to crack can be gotten by running john --list=formats

```
→ hashing john -w=/opt/sc/rockyou.txt --format=raw-md5 md5.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 512/512 AVX512BW 16x3])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
Password123      (?)
1g 0:00:00:00 DONE (2025-04-11 13:17) 50.00g/s 1689Kp/s 1689Kc/s 1689KC/s 062089..redlips
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
→ hashing
```

Demo: Hash Cracking With Hashcat

Hashcat is a GPU/CPU based password recovery tool. we can crack a hash with hashcat by providing the attack mode, hash mode, the wordlist it should utilize and the hash to be cracked, a hash mode can be acquired by running the --help option and grepping for the hash we type we want to crack, the first column holding numbers is our hash mode, in this example, we wish to crack an md5 hash so the hash mode is “0”

```
➜ hashing hashcat --help | grep -i md5
      0 | MD5                                                  | Raw Hash
   5100 | Half MD5                                             | Raw Hash
     70 | md5(utf16le($pass))                                  | Raw Hash
     10 | md5($pass.$salt)                                     | Raw Hash salted and/or iterated
     20 | md5($salt.$pass)                                     | Raw Hash salted and/or iterated
```

Next we will crack the hash by specifying the attack mode which is 0 for a single wordlist. Combination mode involves the use of multiple wordlists and merging them into a single wordlist.

```
 #  | Mode
====+======
 0  | Straight
 1  | Combination
 3  | Brute-force
 6  | Hybrid Wordlist + Mask
 7  | Hybrid Mask + Wordlist
 9  | Association
```

Next we will proceed to crack the hash.

we successfully recover the plaintext of the hash

```
➜ hashing hashcat -a 0 -m 0 md5.txt /opt/sc/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian  Linux, None+Asserts, RELOC, LLVM 18.1.8, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
====================================================================================================================================
* Device #1: cpu-skylake-avx512-AMD Ryzen 9 8945HS w/ Radeon 780M Graphics, 1998/4061 MB (512 MB allocatable), 4MCU
```

1

```
42f749ade7f9e195bf475f37a44cafcb:Password123

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: 42f749ade7f9e195bf475f37a44cafcb
Time.Started.....: Fri Apr 11 13:51:40 2025 (0 secs)
Time.Estimated...: Fri Apr 11 13:51:40 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/opt/sc/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   504.0 kH/s (0.16ms) @ Accel:256 Loops:1 Thr:1 Vec:16
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 33792/14344385 (0.24%)
Rejected.........: 0/33792 (0.00%)
Restore.Point....: 32768/14344385 (0.23%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: dyesebel -> redlips
Hardware.Mon.#1..: Util: 26%
```


Q
&
A