Cyber Security

Applied cryptography

Chris G. Willcocks
Durham University

Introduction

1. Content in this lecture will be useful in the coursework
2. This is a **single lecture** on **applied cryptography** for computer security. there's dedicated codes and cryptography teaching:
    i. History of cryptography
    ii. RSA
    iii. Cryptographic hash functions
    iv. AES
    v. Elliptic Curves
    vi. Entropy
    vii. Error correction
    viii. Linear codes
3. For this reason I won't cover in detail theory, but will focus more on real-world usage & concepts.

Definition

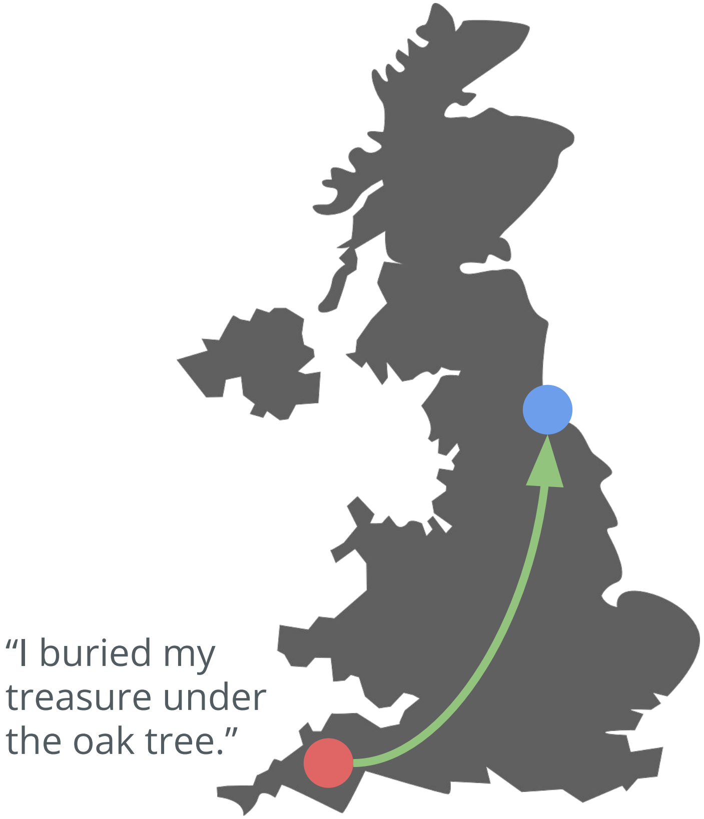
What is cryptography?

“The science of secret writing” - Gollmann.

“Cryptology is the science of communicating using secret codes. It is subdivided into cryptography, writing in codes, and cryptanalysis, deciphering codes.” - Richard R. Brooks.

“Cryptography or cryptology (from Greek κρυπτός kryptós, "hidden, secret"; and γράφειν graphein, "writing", or -λογία -logia, "study", respectively) is the practice and study of techniques for secure communication in the presence of third parties called adversaries.” - Wikipedia.

Encryption & Decryption



Clear text:
"I buried my treasure under the oak tree."

Encryption

Cipher text:
"V ohevrq zl gernfher haqre gur bnx gerr."

Decryption

Clear text:
"I buried my treasure under the oak tree."

Substitution Cyphers

- Replaces each letter of the alphabet with another letter, e.g. ROT13 is a popular basic example.
  - ROT*k* is easy to break, just iterate over all keys and fuzzy string search a word list.
- Lots of variants:
  - Monoalphabetic
    - Fixed substitution
  - Polyalphabetic
    - Change substitution rules in different parts of the message
  - Polygraphic
    - Substitute with groups of letters, e.g. just using pairs increases to $26^2 = 676$

Monoalphabetic simple substitution:
"treasure under the oak tree."
"gernfher haqre gur bnx gerr."

Polyalphabetic:

"treasure under the oak tree."
13 14 15 16 17 18 19

"gerngifs jcstg ixu esc lkxx."

Polygraphic:

"treasure under the oak tree."

"h(C%7]_"

- Variants throughout history: Vigenère, Enigma. Not used much anymore. Broken with frequency analysis (various divide-and-conquer approaches for more advanced poly-alphabetic ciphers)

Encryption & Decryption

- In practise we use algorithms that encrypt the message with a key.
- If both keys are the same, we call this a “symmetric key” cryptosystem.
- If both keys are different, we calls this “asymmetric key” or “public key cryptography”.

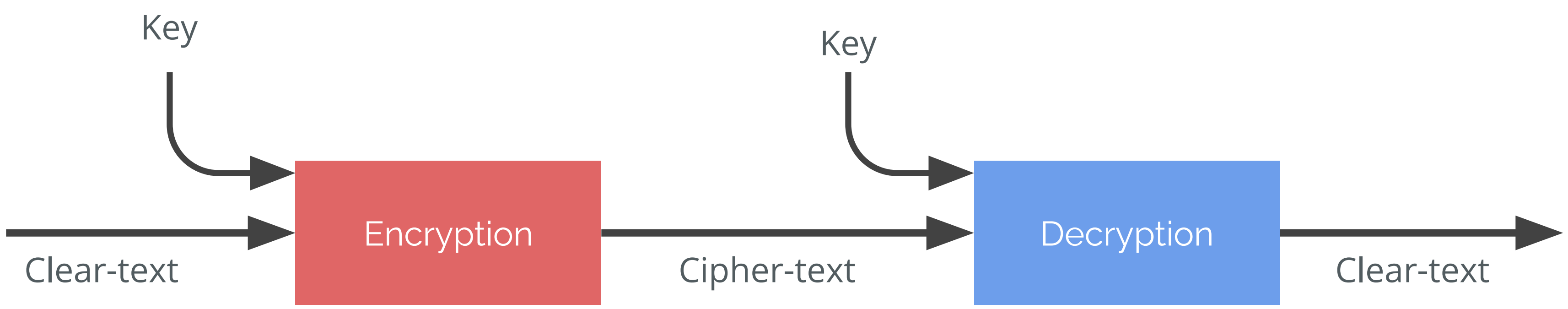


**diagram from Brooks*

SSH Example

```
 chris@chris-lab  ~  master ●  ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/chris/.ssh/id_rsa):
Created directory '/home/chris/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/chris/.ssh/id_rsa.
Your public key has been saved in /home/chris/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:AEo9eIZWH7wDe9ubVfy1oPiVJ0/7njiV4MRvpH8DiTU chris@chris-lab
The key's randomart image is:
+---[RSA 2048]----+
|   .=o..         |
|  .+.*o..        |
|  ..o +o.   ..   |
|     . +.    oE ..|
|      . +S ..*.X.o|
|       . ...o O.O |
|          +. . X .|
|         o  . ..=o|
|              ..o*|
+----[SHA256]-----+
 chris@chris-lab  ~  master ●  cd .ssh
 chris@chris-lab  ~/.ssh  master ●  ls
id_rsa  id_rsa.pub
 chris@chris-lab  ~/.ssh  master ●
```

1

Generate private & public key from a large random number (/dev/random)

Public Key

```
 chris@chris-lab  ~/.ssh  master ●  cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDNJ1om7pvRWr29voRpwQiKbDyA+St3sIJqq2vwNFgIsxZJst
k/se7VcoPpPVknfbS33jPwdDaoM38rJPCh747P1HKz9+yQrF0BxBHb5wd2jlipdRE7gneyiDGQBCJWBZfte/gX
yjjZSHVjXURHWMDqqnxiFQqp1Yy9ql5+jMjM7qCTF32q/SRpJXj7L68wO6AJq7WQV2WLR4Dx3T4yZXqu/eG3Wv
l/e98KfsALjsFEryxwL+qKBShxT04gmRh1QwBq7WZHFB2zz7tLlUSdzWU1m8aIoPrpqVoVxHQEfXR5IBJgMM8Y
PtBfkiFalgWsgmMOA6xtri3JBBKE0fwR chris@chris-lab
 chris@chris-lab  ~/.ssh  master ●  cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAzSdaJu6b0Vq9vb6EacEIimw8gPkrd7CCaqtr8DRYCLMWSbLX
ZP7Hu1XKD6T1ZJ320t94z8HQ2qDN/KyTwoe+Oz9Rys/fskKxdAcQR2+cHdo5YqXU
RO4J3sogxkAQiVgWX7Xv4F48o42Uh1Y11ER1jA6qp8YhUKqdWMvapefozIzO6gkx
d9qv0kaSV4+y+vMDugCau1kFdli0eA8d0+MmV6rv3ht1r/Jf3vfCn7AC47BRK8sc
C/qigUocU9OIJkYdUMAau1mRxQds8+7S5VEnc1lNZvGiKD66alaFcR0BH10eSASY
DDPGGT7QX5IhWpYFrIJjDgOsba4tyQQShNH8EQIDAQABAoIBACi1z2j8XoVL2W0m
ukX2HUjCs7rwNxT4PNkIDVJMQM/xIdIcICwVdvMQ0hkKpfuCcTFlUScBEf3gfR/P
6z5mC3TcWtHBMGXCYE5i/HMAjd1KqaqOt7G5/tFsENbmbaGjjkUN6H0Alk3qUXgN
vn7R2vWhD3JdmczVHoY5ZbkYbRTx3TSUL+qfVugJq0fqg9eHAsErMGQshH1ngyYp
yUApZzZudzh24XLrm6IzzZdIYAQpzOlHVbXIfpC4YvBXk0mS7e5ckPYFTXF4UnQT
hXHuXGUJeDOXFBjK+BiVVF13MmNzENGa/p6WaXpVWcE7R9ebhJUfsW2dx7vO5+lp
vRl2htUCgYEA7g0JeoghuWHrQXdKWxa7nxzCR4DZT28D/KFd0nGkLrqcnjWRbPyE
EGqZW3XvVcqif3P7L0pclsiIShSt2ru6HRy/gISK2XOnrbLwWQxIAT7zALjxo/uU
krW/kbHcDi+NONDKzm3i/1opBnmKdAwWeWB1o3+S1c29iWAIiOEDpt8CgYEA3J9R
ofnV3n6tBV+imD1zP66gTgFB6FgoCuUaYhTQIoKB7y1PjWZOE1dTPu0Bly2bw65l
5y7nKu/+UbXvN4+MxzrSuwljvnbHleh4557kDokqLXpropqGHVIZOguZiH0Zplff
Iorn4AbR1dDoVmGLxL2B14a9bL+owlMZmxikaw8CgYA+PG5T+K9Idk85SGTKN8FJ
sg0hAJDP8ahLLi0HTeqsxOEZg8vHgKVOXXBGtyihd17Qj5QGIxzd1wO6ynlBjH72
CYQPp5ddjp0yDmxD+6//k2OYtsGGK8oGKToybxunIct8JSpAmf4U4I+FP9VwzZA6
n3tiCYJsH2QEPHU+xAVeZQKBgQCm3HyiVFIUQJzd5pIUMM3cyVeHB95wO1S1wMdu
jS1KHIjne86iNFEywaY9foXocF9R5bI+4rOGPxOLe13dGN4xxOSi/5wH7tBPKg9f
p0X3VHipAuErj1GGFZTs2N8bYvqmW+lwt7xeLOpBnApNYuOSPn5Yoxjbii26I53H
NyELswKBgQCAAm3Ef2rFVWf9+8g+4xFYyN2AZ4PmIJH9+s1LWespTiI2mhprc/I7
guWT7mPQPXKap4b8weH7cikQFsjf2SRaPPh702BCuL8z85lYjWRsCJUm97f4AGXz
iR5TnZVs0lBZGbrxbRgq2NsVEze/ySKl5RVALNaOM9PUWmyNzxOJgg==
-----END RSA PRIVATE KEY-----
```

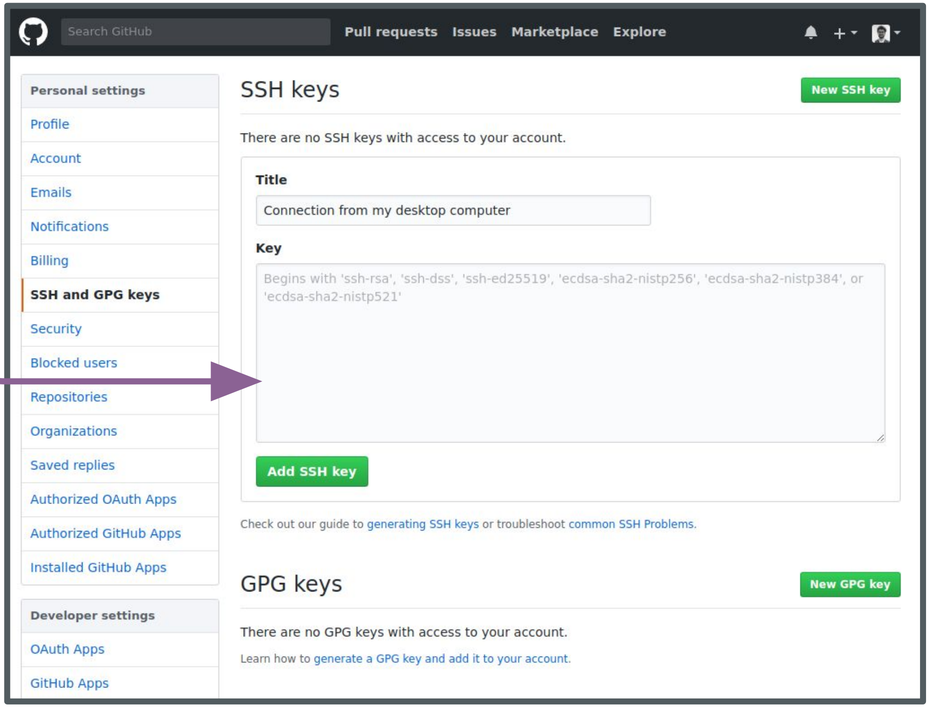
2

Private key, stays with you and is not distributed.

Distributing Public Keys

```
chris@chris-lab  ~/.ssh  master ●  cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDNJ1om7pvRWr29voRpwQiKbDyA+St3sIJqq2vwNFgIsxZJst
k/se7VcoPpPVknfbS33jPwdDaoM38rJPCh747P1HKz9+yQrF0BxBHb5wd2jlipdRE7gneyiDGQBCJWBZfte/gX
yjjZSHVjXURHWMDqqnxiFQqp1Yy9ql5+jMjM7qCTF32q/SRpJXj7L68wO6AJq7WQV2WLR4Dx3T4yZXqu/eG3Wv
l/e98KfsALjsFEryxwL+qKBShxT04gmRh1QwBq7WZHFB2zz7tLlUSdzWU1m8aIoPrpqVoVxHQEfXR5IBJgMM8Y
PtBfkiFalgWsgmMOA6xtri3JBBKE0fwR chris@chris-lab
```

Give your public key to other computers or applications, then you can connect and send messages to them securely without typing in your username/password each time



Other key pairs:

PGP key pairs:

- Can be set up in a similar fashion. Both can choose the underlying algorithm (RSA, DSA, etc).

TLS (replaced SSL) key pairs:

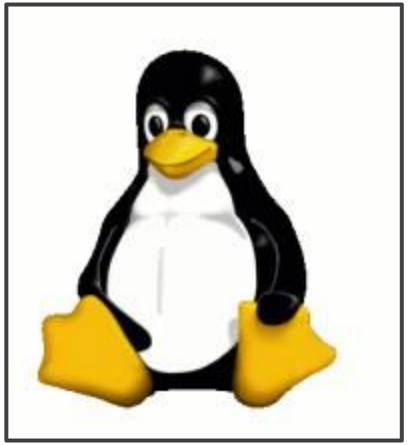
- Encrypt TCP/IP communications and secure browser-server connections (used for SSL Certificates)

Bitcoin wallets.

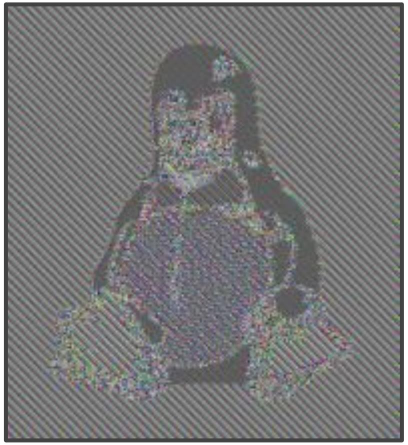
Block Ciphers

- **Symmetric key** encryption method typically used for files
- Encrypts blocks of text at a time, rather than bits of text (stream ciphers).
- DES encrypts 64-bit blocks at a time.
- AES encrypts 128-bit (or bigger) blocks at a time.
- Developed to eliminate the chance of encrypting identical data the same way: the ciphertext from the previous block is fed into the algorithm for computing the next block.
- Also uses an initialisation vector such that same message encrypted multiple times will be different.

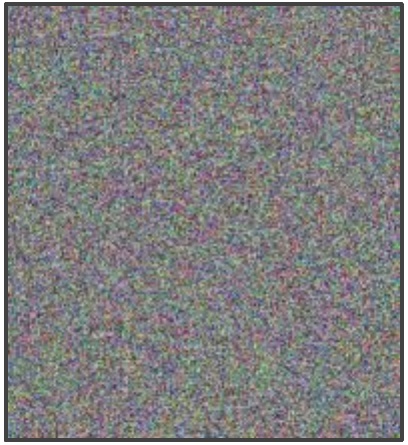
ECB vs Non-ECB modes



Image



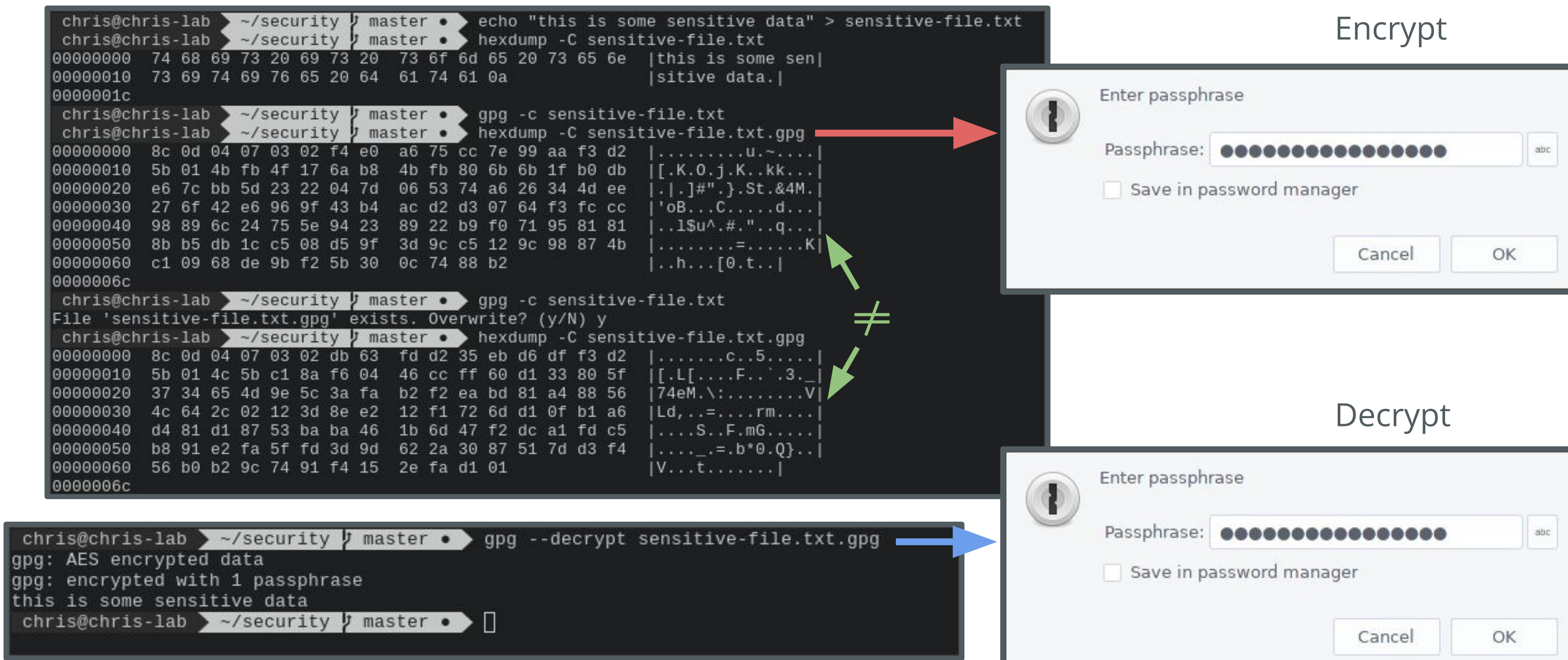
“ECB Penguin”



Non-ECB mode looks pseudo random

Further reading: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Block Cipher Example



Hacking AES-256

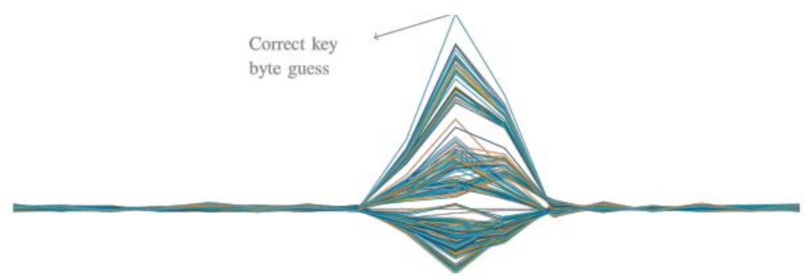
AES-256 is currently regarded as one of the most secure block cipher algorithms. To brute-force you would need 2^{256} =

115792089237316195423570985008687907853269984665640564039457584007913129639936

guesses, which would take longer than the age of the universe.
AES makes the system secure?

- Hacking AES-256 wifi passwords in 8,192 guesses
- Cache-timing attacks for AES

These are **side-channel attacks**.



Correlation traces for all key byte guesses (address Hamming distance model)

Storing Passwords

Storing passwords in plain text is not good.

- If someone obtains database of user IDs/passwords (e.g. database leak, inside job, hacked server, bad admin) then all users are exposed.
- We should design it that, even there is a flaw in the system security, the password should be hard to find.
- Q. Should we encrypt the passwords?
- We can hash the passwords and check the hashes match instead.

```
chris@chris-lab  ~  master ●  echo -n "mypassword" | sha256sum
89e01536ac207279409d4de1e5253e01f4a1769e696db0d6062ca9b8f56767c8  -
chris@chris-lab  ~  master ●  echo -n "mypassword" | sha256sum
89e01536ac207279409d4de1e5253e01f4a1769e696db0d6062ca9b8f56767c8  -
chris@chris-lab  ~  master ●
```

Hash Functions

- Any function that can map data of arbitrary size to a fixed size.
- Different applications require hash functions with different properties.
  - E.g. in graphics, you may want a spatial hash which maps from 3D space to 1D space while guaranteeing locality; points nearby in 3D are also nearby in 1D so you can retrieve objects from your game world quickly without cache misses.
- Cryptographic hash functions should guarantee these properties:
  - **Deterministic**
  - **One-way** function
  - **No collisions**
  - **Avalanche effect**
- Popular algorithms:
  - MD5 (no longer deemed secure)
  - SHA-1 (no longer deemed secure)
  - SHA-2, e.g. SHA-256 and SHA-512 - better but still susceptible to certain attacks.

Storing passwords (continued)

- Will storing our passwords as a list of hashes, which can’t be inverted, make us secure?
- Most passwords are
  - not random characters
  - not arbitrary length
  - have some structure to them
- For example, if we assume passwords are <= 8 characters
  - 1 character could have 94 possibilities (number of printable ascii characters)
  - 2 characters have 94^2
  - 8 characters have 94^8 possible values = 6,095,689,385,410,816 assuming 10^{10} hashes/second (e.g. MD5, others are much slower) would take **7 days**.
  - 9 characters would take **2 years**.
- This would be an **offline attack**.

Precomputed Hash Tables

What if we crowd-funded the precomputation of the hashes, and stored (sold) them on a hard drive?

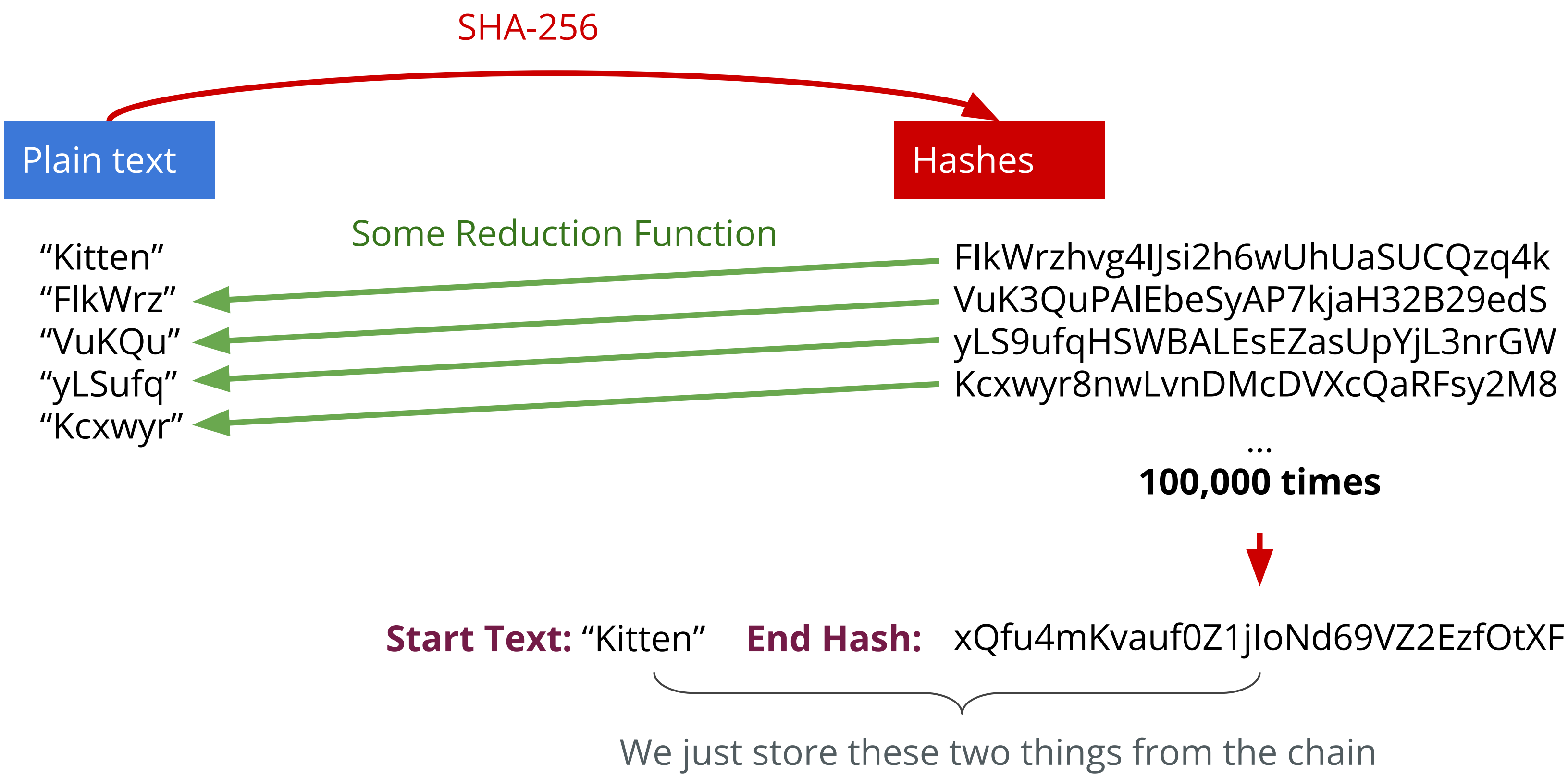
- They could just lookup in the order of minutes/hours.
- Calculating hashes is significantly slower than doing a lookup.

94^8 x (8 bytes **plain text** + 32 **byte hash**) = **244,000 TB** (too much storage for such an attack).

Is there a really good compression scheme which allows for fast querying?

- Rainbow Tables (get some here)

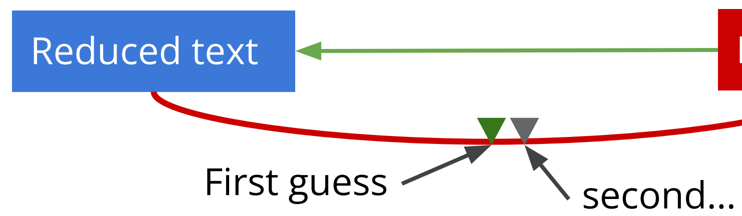
Rainbow Tables



Rainbow Tables

Start text	**End of chain after 100,000**
RiLpFt	c744b1716cbf8d4dd0ff4ce31a17715
NoEqki	xQfu4mKvauf0Z1jIoNd69VZ2EzfOtXF
VsTwNi	3cd696a8571a843cda453a229d7418
…	…
FsAilW	7ad7d6fa6bb4fd28ab98b3dd33261e8f

We sort table by end hashes allowing for fast search. Table “contains” all pre-computed hashes but is **100,000 times** smaller.



To query the rainbow table:

1. **Iterate 100,000 times.** Look for the hash in the sorted list of final hashes.
2. If not found **reduce the hash** into another plaintext, and **hash the new plaintext**.
3. If it is found, the chain for which the hash matches the final hash contains the original hash. You can now go from start of chain to recover secret plain text.

[Reading]

Salts

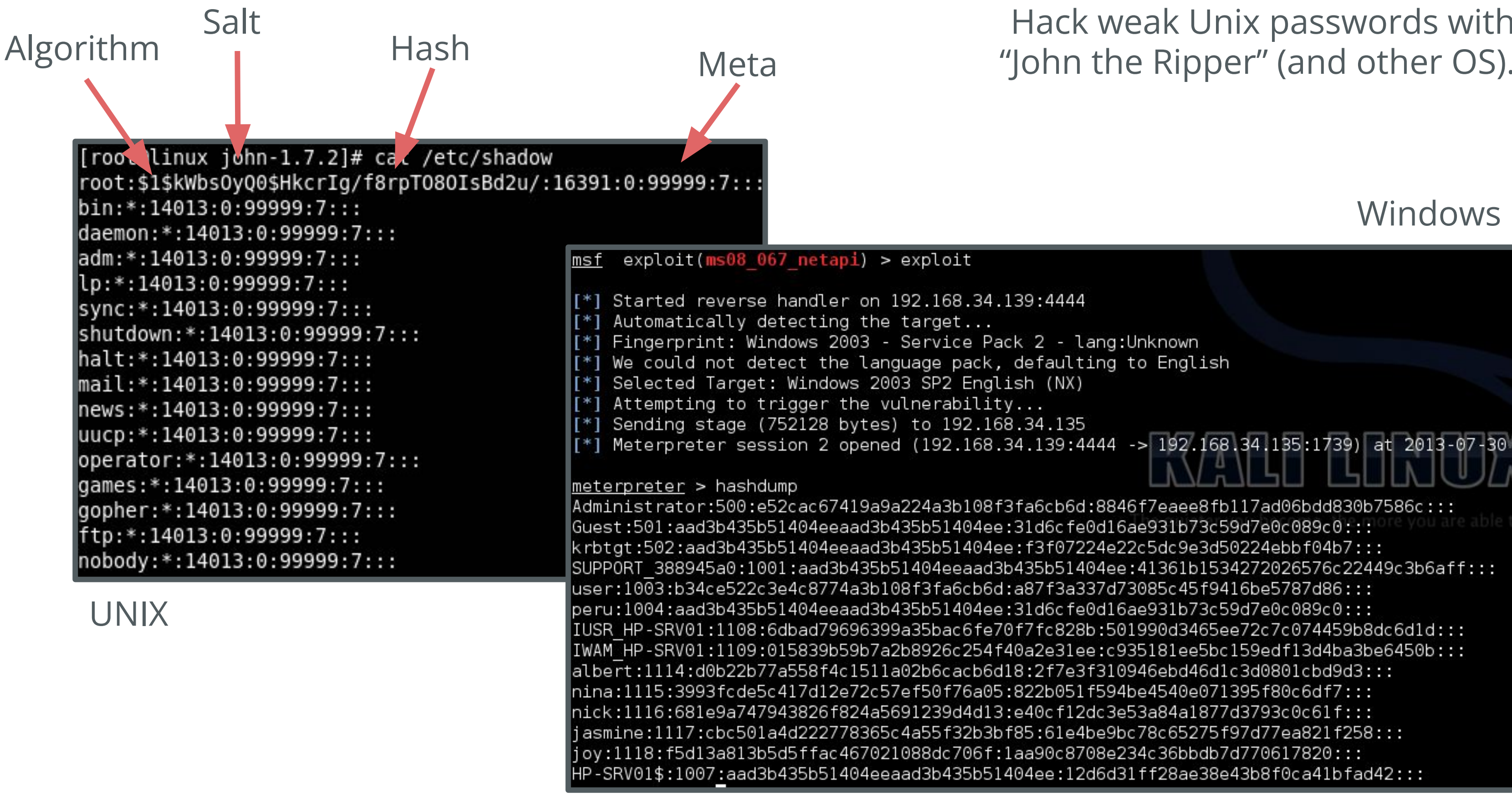
- Using rainbow tables we can recover passwords within minutes and retain reasonable storage requirements (e.g. **500 GB** for a rainbow table).
- So we introduce a "salt" which is random string "7sA9Fbf" stored as **plain text** alongside the **hash**, but we compute the hash by:

$$\text{hash} = H(\text{salt} + \text{password})$$

- Two users with the same password will now have different hashes, as they will have different randomly generated salts.
- For 32-bit salts, you would now need to pre-compute and query 2^{32} rainbow table databases (for each salt value) making such hacking approaches infeasible.

Storing Passwords Example

Hack weak Unix passwords with “John the Ripper” (and other OS).



Coursework

- Coursework will be on DUO next week.

This is unlike any other coursework you will get, as you will not have explicit instructions on how to hack the system. This is done on purpose to emulate the mindset of a hacker.

- **DON'T PANIC!**

We will cover more things that will be useful for the coursework in subsequent lectures and labs.