Systems Security Group | Lancaster University

Welcome to SCC.363

Security and Risk

Who are we?

- Dr Antonios Gouglidis
  - Email: a.gouglidis@lancaster.ac.uk
  - Research domain and interests: Security of systems and models
  - Topics: Access control, Model checking , Cloud systems
- Dr Yang Lu
  - Email: y.lu44@lancaster.ac.uk
  - Research domain and interests: Cyber-physical systems privacy and security
  - Topics: Attack-resilient machine learning, Multi-agent control and optimization
- Prof. Neeraj Suri
  - Email: neeraj.suri@Lancaster.ac.uk
  - Research domain and interests: Design, analysis and assessment of trustworthy Cloud systems and software.

Online learning expectations

- Online tools will be used to facilitate some aspects of learning e.g. Moodle, Teams, etc.
- However, this is a reminder that the use of these is governed by *existing* policies that you are all currently bound by and have agreed to
- Academic malpractice and plagiarism still applies online
- Direct sharing of code, sharing solutions and/or partial solutions with other students, either privately or in an open chat, is **not acceptable**

SCC.363 Curriculum

- Cryptography
- Fundamentals of security
- AAA
- Operating system security
- Network Security
- Risk management
- Cyber threat intelligence
- Security economics
- Security metrics

+ Guest talks

Systems Security Group | Lancaster University

Structure of the module

Lectures 2 x 1 hour per week

- Topics of lectures are gradually released on Moodle https://modules.lancaster.ac.uk/course/view.php?id=37062#section-3

Labs 1 x 2 hour per week

- Join in your timetabled slot
- Coding exercises to develop practical experience on aspects of security and risk

Module assessment

- 2 x coding exercises coursework
  - Submit your code on Coderunner
  - Each having a weight of 30%
  - Deadlines are end of **Week 15** and **19**
- Coursework is submitted online and checked for plagiarism automatically!
- Exams 40%

More expectations…

- What we expect from you
  - Integrity (no plagiarism, no faking results) and effort (active learning)
  - Join the lectures
  - Plan your time and coursework carefully
- What you can expect from us
  - Make all material promptly available to you on Moodle
  - Arrange extra support if you have already tried the normal route (books, web, etc.)
  - Respond to emails

Communications

- Academic Queries – Questions about the course material and delivery
  - Moodle
  - Email

- All other queries – Difficulties with studies, systems, policies, access, deadlines etc.
  - Email Teaching Office Team: scc-teaching-office@lancaster.ac.uk

Communications

- Check your Lancaster University email every day.
  - This is how we communicate with you. And, Moodle can send email updates of forum posts, and other alerts.
  - Checking your email will be the expectation when you go into paid work or a postgraduate degree.
- If you make a request, explain as much as you can about your situation so we know how to help you without going back and forward multiple times.
- Be courteous and professional when you communicate with staff.

Questions?

Systems Security Group | Lancaster University

An Introduction to Cryptography

Learning Objectives

- Learn about cryptosystems

- Understand what cryptosystems offer and where to use

What is cryptography?

- Cryptography is the study of mathematical techniques related to aspects of information security
- A cryptosystem can provide
  - **Confidentiality**: Renders the information unintelligible except by authorised entities
  - **Data integrity**: Ensure that data has not been altered in an unauthorized manner since it was created, transmitted or stored
  - **Authentication**: Can verify the identity of the user or system that created the information
  - **Authorisation**: Upon providing identity information, the individual is then provided with the key or password that will allow access to a resource
  - **Non-repudiation**: Ensure that the sender cannot deny sending the message

Cryptosystem

- A cryptosystem is a five-tuple (*P,C,K,E,D*), where

  P: Finite message space - plain texts

  C: Finite crypto-text space - cipher texts

  K: Finite key space

  E: Encryption function $E_k : P \rightarrow C, k \in K$

  D: Decryption function $D_k : C \rightarrow P, k \in K$

- It holds: $\boxed{\forall e \in K \exists d \in K : \forall m \in P \Rightarrow D_d(E_e(m)) = m}$

  e: encryption key

  d: decryption key

https://www.cs.jhu.edu/~scheideler/courses/600.471_S05/lecture_6.pdf

What is a key?

- A key is used as an input to a cryptographic function.
- The security of the cryptosystem is based on the key secret.
  - Kerckhoff's principle: *'A cryptosystem should be secure even if everything about the system, except the key, is public knowledge'*.

One-time pad: Perfect encryption scheme

- XOR's message stream and keystream

Message stream: 1001010111

Keystream: 0011101010

Ciphertext stream: 1010111101

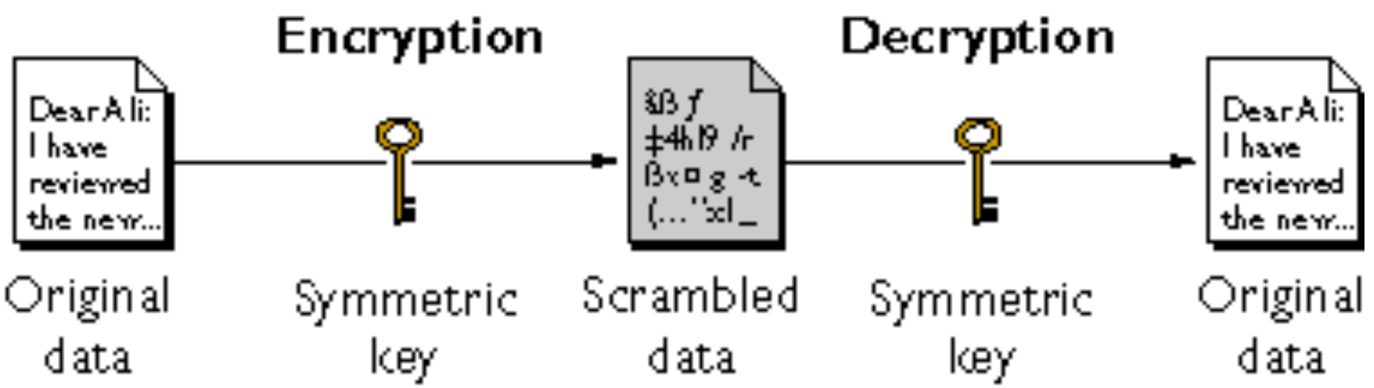
- Deemed unbreakable when:
  - The pad must be used only once
  - The pad must be as long as the message
  - The pad must be securely distributed and protected at its destination
  - The pad must be made up of truly random values

Building blocks of a security system

- Symmetric cryptography
- Asymmetric cryptography
- Message Authentication Codes Vs. Signatures Vs. Hashing

Symmetric cryptography

- A cryptosystem in called symmetric if $d=e$ or if d can at least be easily computed from e.
- Keys required for N parties: $\boxed{N(N-1)/2}$
- Need for exchanging *e (e.g., Diffie – Hellman)*.



Symmetric cryptography - Types

- Block-based ciphers
  - Encrypt blocks of information at a time
  - Stronger than stream ciphers, but slower.
- Two attributes to look after
  - Confusion (obscurity)
    - Relation of key-ciphertext should be complicated; key can't be determined from ciphertext
  - Diffusion
    - Output should depend in a complex way with the inputs; changing 1-bit should have a significant difference in the output

Symmetric cryptography - Types

- Stream-based ciphers
  - Work with one bit at a time
  - They mix plaintext with key stream
  - Good choice for real-time services
  - They are fast and easy to implement in hardware
  - Key is often combined with an initialization vector (IV)

Algorithms for symmetric cryptography

Systems Security Group | Lancaster University

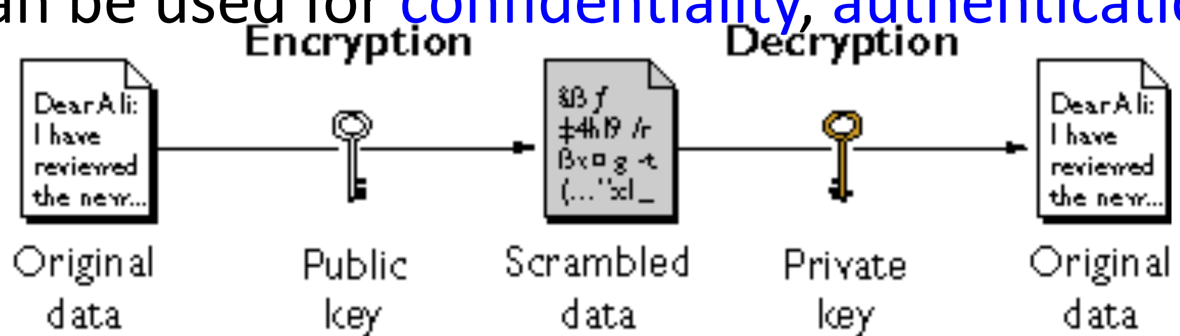
- Data Encryption Standard (DES)
  - DEA is the algorithm
  - DES is the standard
- Triple DES
- Advanced Encryption Standard (AES)
- International Data Encryption Algorithm (IDEA)
- Blowfish
- RC4, RC5

Algorithms for symmetric cryptography

- RC4
  - Stream cipher
  - Use in SSL
  - Improperly implemented in WEP
    - Initialisation vectors: Random values used with algorithms so patterns are not created during encryption
- RC5
  - Block cipher
  - Changeable key size, block size and number of rounds
- RC6
  - Speed improvements over RC5

Asymmetric cryptography

- A cryptographic scheme is called asymmetric if *d* <> *e and* it is computationally infeasible in practice to compute *d* out of *e*.
- In asymmetric cryptography *e* goes public and *d* is kept as a secret.
  - Anybody can use *e* to encrypt a plaintext and only the one that has *d* can decrypt it.
  - Public key cryptographic schemes.
  - Can be used for confidentiality, authentication or both



Algorithms for asymmetric cryptography

- RSA
- Rabin
- El Gamal
- Diffie – Hellman
- Elliptic Curve Cryptography
  - A 256-bit ECC key can be considered equivalent to a 3072-bit RSA key.
  - ECC keys are much smaller than RSA keys.
  - More efficient: computes logarithms of elliptic curves

Advantages and disadvantages

Symmetric

- Strengths
  - High speed encryption
  - Several algorithms use variable key length
- Weaknesses
  - Secure key exchange difficulty
  - Key management difficult

Asymmetric

- Strengths
  - Does not require secure key exchange
  - Provides a method for authentication and digital signatures
- Weaknesses
  - Slow encryption speeds

Cryptographic hash functions

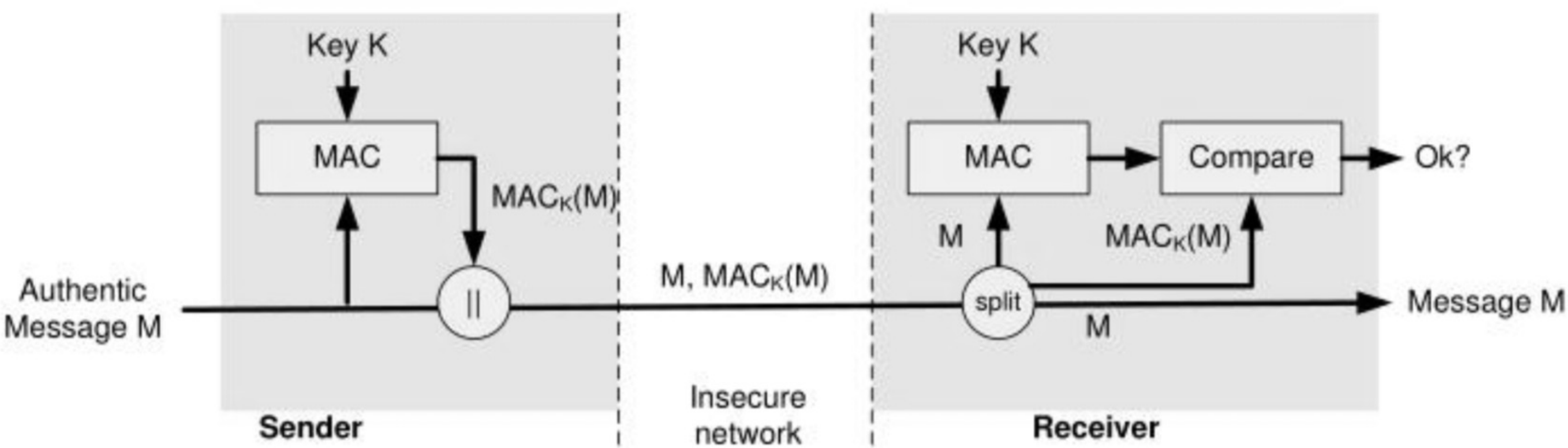
- A cryptographic hash function *H* must provide
  - Compression: e.g, H: $\{0,1\}^* \rightarrow \{0,1\}^{160}$
  - Efficiency: H: h(x) easy to compute for any x
  - One-way: given y it is infeasible to find x: h(x)=y (preimage resistance)
  - Weak collision resistance: for any given x, it should be difficult to find x' , $x' \neq x$ so that h(x')=h(x) (2nd preimage resistance)
  - Strong collision resistance: it should be difficult to find any pair (x, x') with $x \neq x'$ so that h(x)=h(x') (collision resistance)

Cryptographic hash functions

- MD4
  - 128 bit
  - Very fast
  - Has been shown to be broken
- MD5
  - 128 bits
  - Fast
  - Has been shown to have certain weaknesses (collisions can be found easily)
  - Widely used as checksum to verify the integrity of data
- SHA-1
  - 160 bits hash value
  - Standard for US Government
  - Has been shown to have weaknesses
  - Slower than MD5

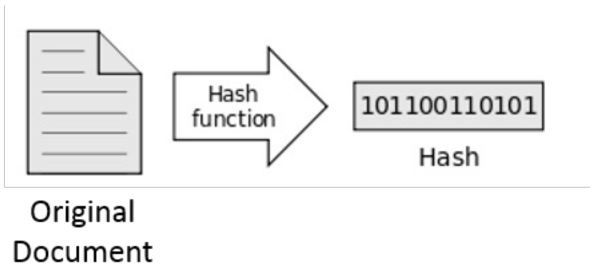
Message Authentication Codes (MAC)

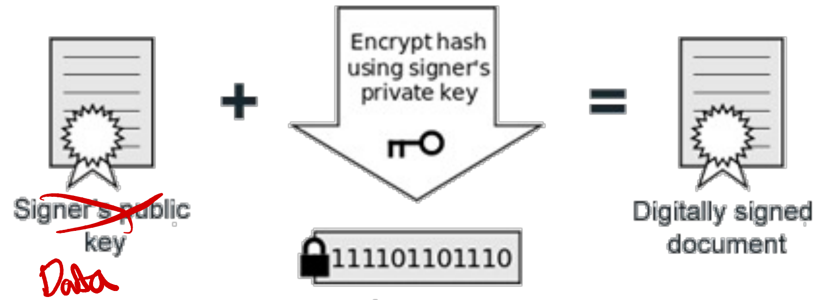
- MAC prevent tampering with messages
  - Encryption may prevent from reading messages, but doesn't prevent from manipulation

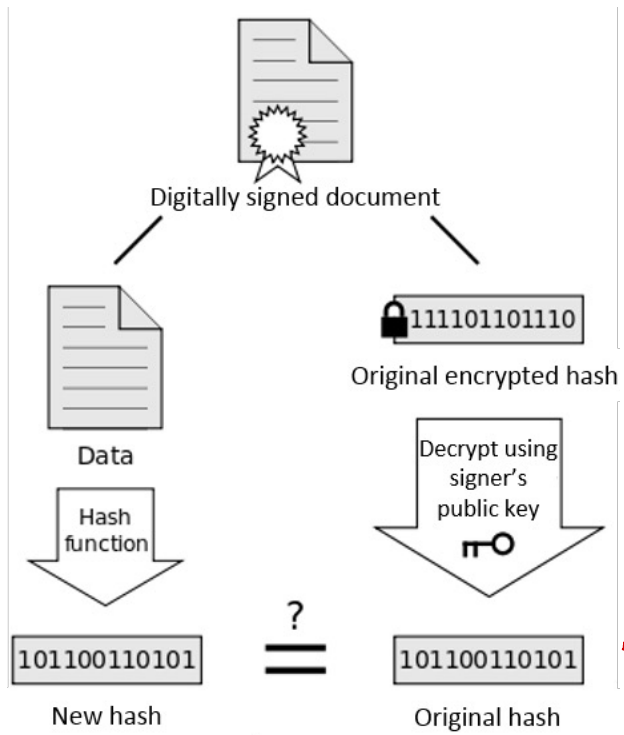


Systems Security Group | Lancaster University

Digital signatures







Ensure that the two hash values are the same to ensure the data sent is authentic

Hash vs MAC vs Digital signatures

	Hash	MAC	Digital Signature
Authentication	No	Yes	Yes
Integrity	Yes	Yes	Yes
Non-repudiation	No	No	Yes
Key type	N/A	Symmetric	Asymmetric

Security of cryptography

- Integer factorization
  - Result = *p* * *q*, *p*, *q* primes (Prime factorization)
  - Given Result find *p* and *q*
  - 6 = *p* * *q*? (easy) *p*=2, *q*=3
  - How about 49,098,013? And it can get really worse!
- Let a large b-bits number
  - No algorithm that can factor in polynomial time $O(b^k)$
- Not completely true!
  - Shor's algorithm can factor in $O(b^3)$ BUT can be run only on a quantum computer!
- Discrete logarithmic problem: find the unique integer $i \in [0, n-1]: \alpha^i = \beta, i = \log_a \beta,$
  p: prime, α,β nonzero integer mod p
  Find $x: a^x \equiv b \bmod p$

Cryptography in networks

- Protecting data while in transit
- Link encryption
  - Protects confidentiality of information within the communications channel only
  - Not prone to traffic analysis
- Network encryption
  - Transparent to users.
  - Independent of any other encryption process used
  - Data encrypted only while in transit.
- End – to – end encryption
  - Encrypts application layer data only.
  - Network devices doesn’t need to be aware.

Secure Socket Layer(SSL) Transport Layer Security (TLS)

- Provided services
  - Data encryption
  - Client and server authentication
- *'The differences between TLS 1.0 and SSL 3.0 are not dramatic, but they are significant enough that TLS 1.0 and SSL 3.0 do not interoperate ...'* [RFC 2246]

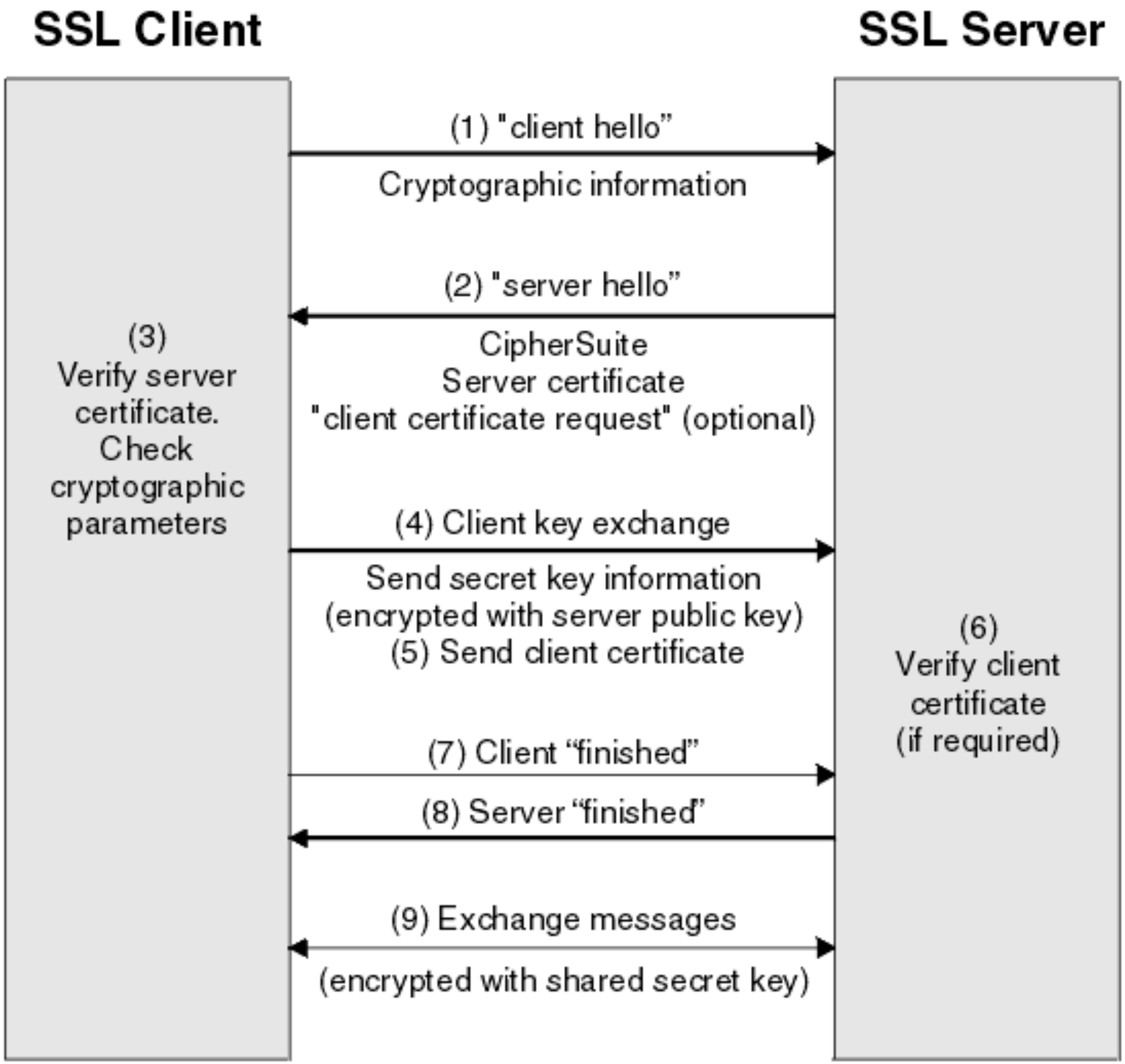
SSL

- Handshake protocol
  - Authentication of client and server
  - Set of encryption algorithms and symmetric keys (agreements)
- Data transfer
  - Encryption
  - Integrity checking

SSL architecture

- SSL session
  - Client and server association
  - Created by the handshake protocol
  - Defines a set of encryption parameters
  - It is possible to be shared among different connections
- SSL connection
  - Temporal, as peer-to-peer connection
  - Associated with one SSL session

SSL handshake



SSL vulnerability!

- Encryption in SSL
  - RC4: know to have biases
  - Cipher block chaining (CBC): currently used

https://www.us-cert.gov/ncas/alerts/TA14-290A
https://www.openssl.org/~bodo/ssl-poodle.pdf

TLS

- Latest version is TLS 1.3 (RFC 8446)
- Includes security and performance improvements
- Security
  - TLS 1.3 removes obsolete and insecure features from TLS 1.2 (e.g., RC4, DES, arbitrary DH-groups)
- Performance
  - TLS handshake requires only one round-trip and reduces the encryption latency in half

Cryptographic attacks

Systems Security Group | Lancaster University

Goal is to discover the key

- Cipher-only attack
  - Obtain ciphertext from several messages
  - Encrypted with the same encryption algorithm
- Known-plaintext attack
  - Attacker has plaintext and corresponding ciphertext of one or more messages
- Chosen-plaintext attacks
  - Attacker has the plaintext and ciphertext, but can choose the plain text that gets encrypted
- Chosen-ciphertext attacks
  - Choose ciphertext to be decrypted and study transformation to plain text

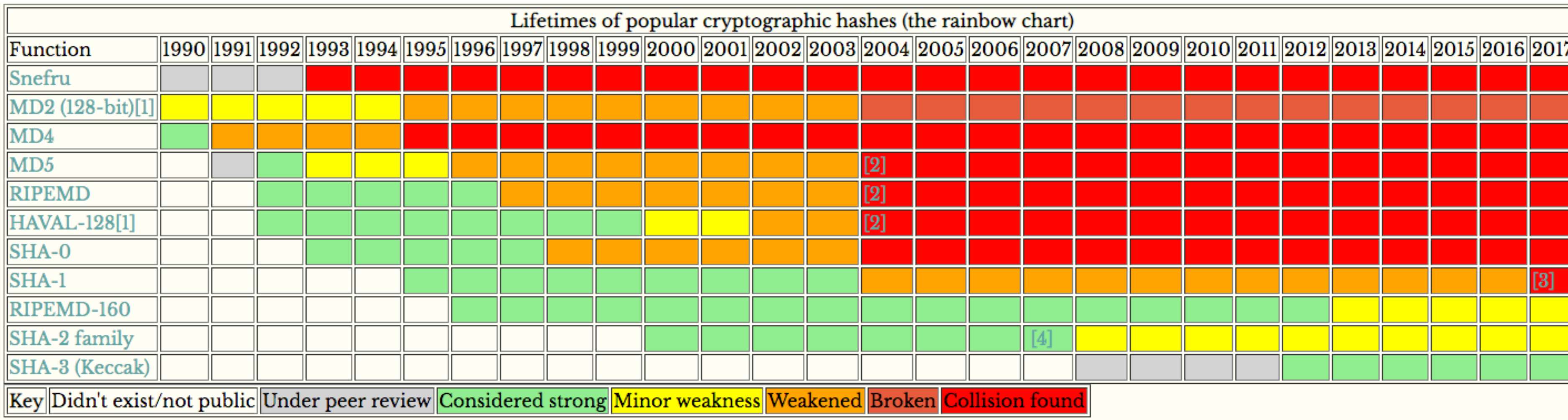
Cryptographic attacks

Systems Security Group | Lancaster University

- Differential cryptanalysis
  - Look at statistical differences when encrypting different messages with the same key
- Side-channel attack
  - Gathering ‘outside’ information
  - 1995 RSA private key uncovered by measuring the relative time of crypto operations
- Social engineering attacks
  - Non-technical attacks that are carried out on people

Systems Security Group | Lancaster University

Lifetime of cryptographic hash functions



Lifetimes of popular cryptographic hashes (the rainbow chart)

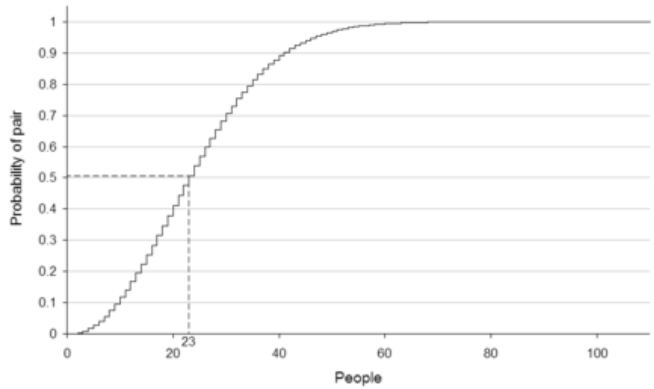
Function	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
Snefru																												
MD2 (128-bit)[1]																												
MD4																												
MD5															[2]													
RIPEMD															[2]													
HAVAL-128[1]															[2]													
SHA-0																												
SHA-1																												[3]
RIPEMD-160																												
SHA-2 family																		[4]										
SHA-3 (Keccak)																												

Key	Didn't exist/not public	Under peer review	Considered strong	Minor weakness	Weakened	Broken	Collision found

http://valerieaurora.org/hash.html

Birthday attack

- Refer to a class of brute-force attacks
- Based on the birthday problem in probability theory
  - The probability that 2 or more people in a group of 23 people to share the same birthday is 50%
  - Raising the group people to 70 increase the probability to 99.9%
- Birthday attacks often used to find collisions of hash functions



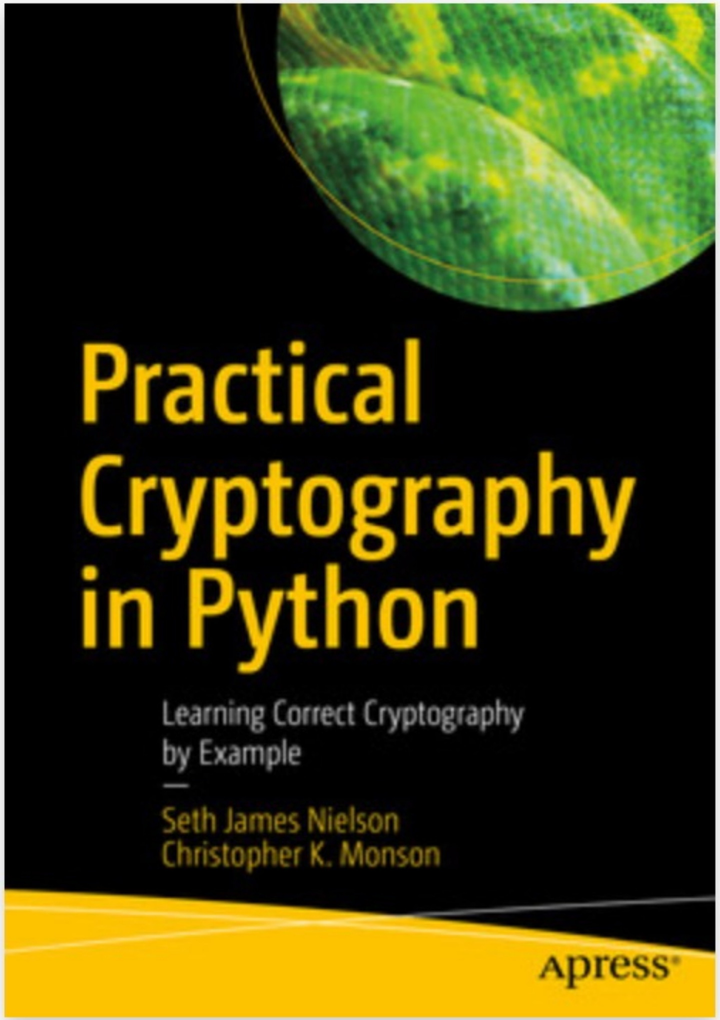
Questions?

References

[1] Shon Harris, All in One, CISSP Exam Guide, Chapter on Cryptography
[2] William Stallings, Cryptography and Network Security, Principles and Practise, 5th edition
[3] Birthday attack, https://www.sciencedirect.com/topics/computer-science/birthday-attack

Week 11 Basics

Recommended reading



The book is available to you via the library

Technology stack

- Python 3
  Link to a Python Cheat Sheet

- cryptography.io
  Link to the library

Topics

- Useful datatypes and conversions
- XORing
- Rotating ciphers

Recommended reading: Chapter 1 from the book of "Practical Cryptography in Python"

Systems Security Group | Lancaster University

String to/from Bytes

Assuming the following string

`str1 = “Hello World!”`

```
>>> print(str1)
Hello World!
>>> type(str1)
<class 'str'>
```

You can convert it to a bytes object using

bytes([source[, encoding[, errors]]]) e.g.

`str1_bytes = bytes(str1, ‘utf-8’)`

```
>>> print(str1_bytes)
b'Hello World!'
>>> type(str1_bytes)
<class 'bytes'>
```

String to/from Bytes (2)

Can't be changed

Byte objects are immutable

```
obj1 = bytes(2)
```

```
>>> obj1
b'\x00\x00'
>>> obj1[0] = 9
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'bytes' object does not support item assignment
```

If a mutable object is required use bytearrays instead

```
obj2 = bytearray(2)
```

```
>>> obj2
bytearray(b'\x00\x00')
>>> obj2[0] = 3
>>> obj2
bytearray(b'\x03\x00')
```

To convert a byte/bytearray to a string datatype use the `.decode()` member function.

Hex to/from bytes

Assume the byte literal

```
obj1 = b”Hello World!”
```

We can covert it to its hexadecimal value as follows

```
hex_obj1 = obj1.hex()
```

```
>>> hex_obj1
'48656c6c6f20576f726c6421’
>>> type(hex_obj1)
<class 'str'>
```

We can covert a hexadecimal value to a byte as follows

```
obj2 = bytes.fromhex(hex_obj1)
```

```
>>> obj2
b'Hello World!'
>>> type(obj2)
<class 'bytes'>
```

Other conversions

Convert a hexadecimal value to an integer

```
int1 = int(hex_obj1, 16)
```

```
>>> int1
22405534230753928650781647905
```

And for reverting it to a hex

```
hex_int1 = hex(int1)[2:]
```

removes the "0x" at the start

```
>>> hex_int1
'48656c6c6f20576f726c6421'
>>> type(hex_int1)
<class 'str'>
```

Other conversions (2)

Convert an integer to a binary

```
bin1 = bin(100)
```

```
>>> bin1
'0b1100100'
```

Informs us that it is a binary

And for reverting it to an integer

```
int1 = int(bin1, 2)
```

base for binary

```
>>> int1
100
```

XORing…

The XOR operator in python is ^

Assume 2 integers, e.g. 10 and 7

What is the value of 10 ^ 7 and why?

$$\begin{array}{lr} 10_{10} \rightarrow & 1010_2 \\ 7_{10} & 0111_2 \\ \hline & 1101_2 = 13_{10} \end{array}$$

XORing (2)

Let’s assume we want to XOR the following strings

“a” with “b”

How can we do this?

What is the result?

XORing (2) - solution

Assuming the string literals ”a” and “b”

```
>>> a = b"a"
>>> b = b”b”
>>> ha = a.hex()
>>> ia = int(ha, 16)
97
```

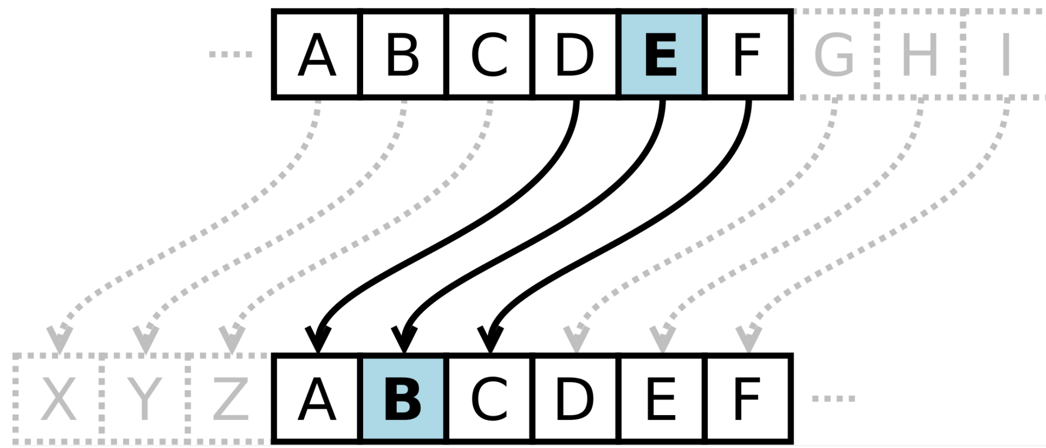
Similarly it’s 98 for “b”. And 97 ^ 98 = 3

Or we could XOR the bytes directly

```
>>> print(a[0]^b[0])
3
```

Rotating ciphers

- Assume an alphabet and a rotation



- A useful data structure to use: dictionaries

```
dict1 = {}
Dict1["E"] = "B"
```

Rotating ciphers

- What if your rotation is greater than your alphabet's size?
  - Modular arithmetic

$$\boxed{\frac{\alpha}{\beta} = q\ remainder\ r}$$

Where, α: dividend, β: divisor, q: quotient, r: remainder

  - The modulo operator in python is %

Behaviour of % with negative numbers

- The result depends on the programming language
- Python calculates the remainder as:

$$r = \alpha - (\beta * floor\left(\frac{\alpha}{\beta}\right))$$

Where floor is the `math.floor(x)` method which returns the floor of `x`, the largest integer less than or equal to `x`.

- What is the result for the following?
  - `math.floor(1.1) = …` 1
  - `math.floor(-1.1) = …` -2

Structure of your code…

Modules you want to import

```
import XYZ
```

List of functions you implement

```
def  myFunction():
     # TODO

     return # TODO
```

Have a main section to call your functions

```
if __name__ == "__main__":
     x = myFunction()
```