

Lecture 1: Course introduction

COSC362 Data and Network Security

Book 1: Chapter 1 – Book 2: Chapter 1

Spring Semester, 2021

Motivation

What is this course about?

- ▶ How does this course run?
- ▶ Why is cyber security important?
- ▶ What is in this course?

Outline

Introduction to the course

Why do we need cyber security?

Course outline

Outline

Introduction to the course

Why do we need cyber security?

Course outline

Administration

- ▶ Course coordinator and lecturer: Clémentine Gritti
 - ▶ `clementine.gritti@canterbury.ac.nz`
- ▶ Teaching assistant: Ryan Beaumont
 - ▶ `rbe72@uclive.ac.nz`
- ▶ Materials for lectures, labs, assessment items (quizzes and assignment) are on LEARN

Pre-requisites and textbook

- ▶ *Pre-requisites:* COSC264 or INFO333
- ▶ It is recommended that COSC362 and COSC364 are taken together.
- ▶ *Recommended textbooks:*
 - ▶ *Book 1: Cryptography and Network Security: Principles and Practice*, W. Stallings, (5th) 7th Edition
 - ▶ *Book 2: Computer Security: Principles and Practice*, W. Stallings and L. Brown, (3rd) 4th Edition
 - ▶ Useful to back up lectures and practice with exercises (useful for exam preparation!)
 - ▶ Syllabus for the examination mainly defined by the lecture slides, not by the textbooks
- ▶ Many useful resources online (some will be mentioned on LEARN)

Assessment

- ▶ Ongoing work during the semester:
 - ▶ 8 quizzes (20% in total)
 - ▶ 1 assignment (20%)
- ▶ Labs attendance and participation (10%)
- ▶ Final exam (50%)

Check the semester plan and timetable on LEARN:

- ▶ Submission dates
- ▶ Other useful details
- ▶ The timetable may sometimes be updated

Timetable

- ▶ Lecture times

- ▶ 19 July – 27 August and 13 September – 22 October
- ▶ Monday 10:00 – 11:00 in E5 Lecture Theatre
- ▶ Thursday 11:00 – 12:00 in E5 Lecture Theatre

- ▶ Lab times

- ▶ 26 July – 27 August and 13 September – 22 October
- ▶ Friday 16:00 – 18:00 (01) & Tuesday 16:00 – 18:00 (03) in Jack Erskine 136 Lab 4
- ▶ Wednesday 14:00 – 16:00 (02) in Jack Erskine 131 Lab 1

Check timetable on LEARN (the timetable may sometimes be updated).

Other information

- ▶ Online quizzes will give you direct feedback:
 - ▶ This kind of multiple-choice questions will be included in the final exam.
- ▶ The mid-term assignment will have similar questions to the ones encountered in the final exam.
- ▶ Labs:
 - ▶ Lab tasks/questions are first released on LEARN.
 - ▶ Solutions/answers are released later, say around two weeks after corresponding labs happened.
 - ▶ In case you missed one lab, you have one week to submit a report.

Outline

Introduction to the course

Why do we need cyber security?

Course outline

New Zealand – December 2017

Privacy “Agency obligations are defined in the 12 Information Privacy Principles that underpin the Privacy Act 1993.” – setting out how agencies may collect, store, use and disclose personal information

Security “Government agencies must consider the nature and value of the information they are managing and the measures needed to protect it.”

Risk management “Understanding, assessing and documenting the scope of [the] risk service delivery, reputation, legal exposure, security and integrity, customer confidentiality and investment.”

<https://www.digital.govt.nz/standards-and-guidance/privacy-security-and-risk/>

European Union (EU) – May 2018

The EU General Data Protection Regulation (GDPR) aims to “protect and empower all EU citizens data privacy”:

Google “We’re making these [privacy] updates as the GDPR takes effect across the EU.”

LinkedIn “Your privacy comes first in all of these updates. We now meet the high standard for data privacy introduced by the new European data protection law known as the GDPR.”

IBM cloud “Measures [are] in place to protect your data, including any personal information that may be subject to data protection regulations, including GDPR.”

[https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/
data-protection/2018-reform-eu-data-protection-rules_en](https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en)

Dark Hotel attack – 2012



- ▶ Targeted phishing attacks using spyware
- ▶ Infiltrating guests' computers through Wi-Fi networks in hotels
- ▶ Loss of confidentiality

<https://www.kaspersky.com/blog/darkhotel-apt/6613/>

Ashley Madison data breach – 2015



- ▶ Exposing over 30 GB of user data (real names, banking data, credit card transactions)
- ▶ Hacktivism: the hacking group decided to “punish” the company
- ▶ Loss of confidentiality

<https://digitalguardian.com/blog/timeline-ashley-madison-hack>

Hello Barbie attacked – 2016



- ▶ POODLE attack
(man-in-the-middle exploit)
- ▶ Communications
intercepted and decrypted
between Barbie and
servers
- ▶ Loss of confidentiality

[https://www.iflscience.com/technology/
barbie-doll-records-your-voice-has-hackable-security-flaw/](https://www.iflscience.com/technology/barbie-doll-records-your-voice-has-hackable-security-flaw/)

EncroChat used by criminals – 2020



GUARANTEE ANONYMITY
No way to associate device or SIM card to customer account.

CUSTOMIZED ANDROID PLATFORM
Fully encrypted from power-on. Focus on security and privacy. Simplified user settings.

DUAL OPERATING SYSTEM (IOS)
Subscribers can now launch either a standard Android OS or the EncroChat OS. Two distinctive Operating Systems packaged with each device.

OVER-THE-AIR (OTA) SERVICE
Enhancements, patches, and features can be securely added directly to the Android Operating system of a subscriber device.

INDUSTRY LEADING HARDWARE
Specially tailored to harder security. Removal of camera, microphone, GPS and USB data port.

GLOBAL SERVICE
Quad-band GSM, UMTS and CDMA all supported. Unlocked international SIM included (200 countries).

FIELD FACTORY RESTORE
A year can now securely wipe subscriber device and reflash it in the field.

ENRCROCHAT MESSAGING PROTOCOL
The electronic equivalent of a regular conversation between two people in an empty room.

SIMPLIFIED VERIFICATION
Using our history verification process vastly simplifies the complexities of encryption for end-users.

MESSAGES THAT SELF-DESTRUCT
With our advanced tools a user can force wipe their own messages from another user's device using a timer countdown.

PANIC WIPE
From screen lock a user can type in a PIN and instantly wipe device's data.

PASSWORD WIPE
After a set amount of password attempts on device all data is wiped.

SECURE BOOT
Upon boot, the device internally checks boot to ensure no one has tampered with the system files.

TAMPER PROOFING
Attack surfaces such as ADB connectivity and recovery mode have been removed.

HARDWARE CRYPTOGRAPHIC ENGINE (RIPPLE-2 CERTIFIED)
An EncroChat device can not be forced to mount the encrypted data partition. We generate an RSA public/private keypair with which the public key portion is combined with your disk encryption private key to create a unique key for each hardware backed keystore. You can't mount the encrypted data partition.

UPDATES & LIVE SUPPORT
Frequent application updates direct. Includes live support.

- ▶ A communications network and service provider allegedly used by gang members to plan a number of criminal activities
- ▶ Infiltrated by police in June and July 2020 during a Europe-wide investigation
- ▶ Operations were ceased due to the police operation
- ▶ Loss of confidentiality

Sony's PlayStation network attacked – 2011



- ▶ Attackers inject characters or lines of code into attacked website
- ▶ Structure Query Language (SQL) injection attack
- ▶ Loss of integrity

<https://news.softpedia.com/news/Sony-Pictures-Hacked-Millions-of-Accounts-Exposed-204036.shtml>

WannaCry ransomware – 2017



- ▶ Unpatched Windows systems
- ▶ Stolen government hacking tools
- ▶ Worm encrypting files on computers' hard drive, then demanding a payment in bitcoin to decrypt them
- ▶ Loss of availability

<https://www.cscoonline.com/article/3227906/what-is-wannacry-ransomware-how-does-it-infect-and-who-was-responsible.html>

Mirai botnet – 2016



- ▶ Botnet attacking IoT devices with default admin credentials
- ▶ Distributed Denial of Service (DDoS) attack
- ▶ Loss of availability

<https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/>

You, me – July 2021

- ▶ Have you been “pwned”?
 - ▶ Check whether your private information has been leaked or compromised.
 - ▶ Check your email address:
<https://haveibeenpwned.com/>
- ▶ Is your password strong enough?
 - ▶ Take a strength test.
 - ▶ Check your password's strength:
<https://www.my1login.com/resources/password-strength-test/>
- ▶ UC tips for better cyber security awareness:
 - ▶ <https://www.canterbury.ac.nz/its/cyber-security/>
 - ▶ <https://www.comparitech.com/blog/information-security/security-remote-working/>

Outline

Introduction to the course

Why do we need cyber security?

Course outline

Course focus

- ▶ Cryptography as a foundation for information security
- ▶ Applications of cryptography in network security
- ▶ Prominent internet security protocols

Some mathematics for cryptography are needed, but their usage is emphasized rather than proofs.

Course contents

- ▶ Historical cryptography
- ▶ Modern cryptography:
 - ▶ Block ciphers, stream ciphers
 - ▶ Public key cryptography
 - ▶ Hashing and MAC
- ▶ Some mathematics:
 - ▶ Modular arithmetic
 - ▶ Number theory
 - ▶ Elliptic curves
- ▶ Using all of the cryptography:
 - ▶ Public key infrastructure
 - ▶ Secure email
 - ▶ TLS (HTTPS)

Any questions?

Fell free to:

- ▶ ask during the lectures
- ▶ come to see me in my office (no specific office hours)
- ▶ send me an email
- ▶ post your questions onto the **anonymous** LEARN forum

Lecture 2: Course overview

COSC362 Data and Network Security

Book 1: Chapter 1 – Book 2: Chapter 1

Spring Semester, 2021

Motivation

What is this course about?

- ▶ What is cyber security?
- ▶ What is information security?

Outline

What is cyber security?

What is information security?

Outline

What is cyber security?

What is information security?

Defining cyber security

Definition from the NIST computer security handbook: “The protection afforded to an automated information system in order to attain the applicable objectives of preserving the *integrity*, *availability*, and *confidentiality* of information system resources (include hardware, software, firmware, information/data, and telecommunications).”

- ▶ Security literature might differentiate the concepts of *computer security* (concerned with the security of a single computer) and *cyber security* (concerned with the security of multiple computers).
- ▶ In this course, we do not make much of a distinction between the two terms.

Security terminology

A threat represents a potential security harm to an asset (system resource).

An attack is a threat that is carried out and, if successful, leads to an undesirable violation of security.

The threat agent carrying out the attack is referred to as an attacker.

A countermeasure is any means taken to deal with a security attack (e.g. prevention, detection/recovery).

A residual level of risk to the assets is represented by vulnerabilities possibly exploited by threat agents.

Key questions

- ▶ What assets (system resources) do we need to protect?
- ▶ How are those assets threatened?
- ▶ What can we do to counter those threats?

Assets

Hardware: computer systems and other data processing, data storage, and data communications devices

Software: operating system, system utilities, and applications

Data: files and databases, as well as security-related data (e.g. password files)

Communication facilities and networks: local and wide area network communication links, bridges, routers, etc.

Vulnerabilities

A computer system or network can be:

Leaky meaning that it gives access to information through the network while it should not (see Confidentiality).

Corrupted meaning that it does the wrong thing or gives wrong answers (see Integrity).

Unavailable meaning that it becomes impossible to use it or impractical (see Availability).

Passive attacks

- ▶ DO NOT alter information and resources in the system
- ▶ may be hard to detect but easy to prevent

Eavesdropping (interception): the attacker directly accesses sensitive data traveling between authorised source and destination.

Traffic analysis (inference): the attacker gains information from observing the amount of traffic between source and destination.

Active attacks

- ▶ DO alter information and/or resources in the system
- ▶ may be hard to prevent but easy to detect (and recover)

Masquerade: the attacker claims to be a different entity.

Modification of messages (falsification): the attacker changes messages during transmission.

Distributed denial of service (misappropriation): the attacker prevents legitimate users from accessing resources.

Inside attacks

- ▶ initiated by an entity INSIDE the security perimeter
- ▶ authorization to access system resources but use of them in a malicious way

Exposure: the attacker intentionally releases sensitive information to an outsider.

Falsification: the attacker alters or replaces valid data or introduces false data into a file or database.

Outside attacks

- ▶ initiated from OUTSIDE the perimeter, by an unauthorised or illegitimate user of the system

Obstruction: the attacker disables communication links or alters communication control information.

Intrusion: the attacker gains unauthorised access to sensitive data by overcoming the access control protections.

Security functional requirements

- ▶ Information security management requires to:
 1. Identify threats
 2. Classify all threats according to likelihood and severity
 3. Apply security controls based on cost benefit analysis
- ▶ Countermeasures to vulnerabilities and threats comprise:
 1. Computer security technical measures (e.g. access control, authentication, system protection)
 2. Management measures (e.g. awareness and training)
 3. Both (e.g. configuration management)

Outline

What is cyber security?

What is information security?

Defining information security

Definition from the ISO security architecture: “The term *security* is used in the sense of minimizing the vulnerabilities of assets and resources. An asset is anything of value. A *vulnerability* is any weakness that could be exploited to violate a system or the information it contains. A *threat* is a potential violation of security.”

- ▶ *Information security* can be defined as security where the assets and resources are information systems.
- ▶ This can include data, software and hardware, people and even buildings.

The CIA triad

Traditional definitions are based on 3 information security goals:

Confidentiality: preventing unauthorised disclosure of information (POODLE attack)

Integrity: preventing unauthorised (accidental or deliberate) modification or destruction of information (SQLI attack)

Availability: ensuring resources are accessible when required by an authorised user (DoS attack)

OSI Security Architecture X.800

- ▶ A bit dated now but still worth looking at:
 - ▶ Most definitions and terminology still apply.
- ▶ Defines *security threats* (attacks), *security services* and *security mechanisms* and how they are related.

Source: <https://www.itu.int/rec/T-REC-X.800/en>
Useful supplement: Internet Security Glossary, RFC 4949

Security services and mechanisms

Security service: a processing or communication service to give a specific kind of protection to system resources.

Security mechanism: a method of implementing one or more security services.

A security service, provided by a layer of communicating open systems, ensures adequate security of the systems or of data transfers as defined by ITU-T X.800.

Security services

- ▶ *Peer entity authentication* provides confirmation of the claimed identity of an entity.
- ▶ *Data origin authentication* provides confirmation of the claimed source (origin) of a data unit (message).
- ▶ *Access control* provides protection against unauthorized use of resources. Access control service is usually provided in combination with authentication and authorisation services.

Security services (continued)

- ▶ *Data confidentiality* protects data against unauthorised disclosure.
- ▶ *Traffic flow confidentiality* protects disclosure of data which can be derived from knowledge of traffic flows.
- ▶ *Data integrity* detects any modification, insertion, deletion or replay of data in a message or a stream of messages.

Security services (continued)

- ▶ ***Non-repudiation*** protects against any attempt by the creator of a message to falsely deny creating the data or its contents.
 - ▶ X.800 talks about non-repudiation of origin to protect against denial by the sender of a message, and non-repudiation of receipt to protect against denial by the recipient of a message.
- ▶ ***Availability*** service protects a system against denial of service.
 - ▶ It is not listed in X.800 as a separate service.

Security mechanisms

- ▶ *Encipherment* is the transformation of data in order to hide its information content.
 - ▶ Later in the course you will look at both public-key and symmetric-key encryption.
- ▶ *Digital signature* mechanisms are cryptographic algorithms which transform data using a signing key.
 - ▶ The essential property is that signed data can only be created with the signing key.
 - ▶ You will look at standard signature schemes.

Security mechanisms (continued)

- ▶ X.800 describes a variety of *access control* mechanisms including access control lists, passwords, or tokens, which may be used to indicate access rights.
- ▶ X.800 describes *data integrity* mechanisms as “corruption detection techniques“ which can be used with ”sequence information“.
 - ▶ You will look at the example of Message Authentication Codes (MACs).
- ▶ *Authentication exchange* mechanisms are protocols which exchange information to ensure identity of protocol participants.
 - ▶ You will study examples such as TLS later.

Security mechanisms (continued)

- ▶ *Traffic padding* is spurious traffic generated to protect against traffic analysis.
 - ▶ Traffic padding is typically used in combination with encipherment.
- ▶ *Routing control* mechanism is the use of specific secure routes.
- ▶ The *notarization* mechanism uses a trusted third party to assure the source or receipt of data.
 - ▶ The trusted third party is sometimes called a notary.

Relating security services to mechanisms

Mechanism	Encipherment	Digital signature	Access control	Data Integrity	Auth. exchange	Padding	Routing control	Notarization
Service								
Peer entity authentication	✓	✓		✓				
Data origin authentication	✓	✓						
Access control			✓					
Data Confidentiality	✓					✓		
Traffic Flow Confidentiality	✓				✓	✓		
Data Integrity	✓	✓		✓				
Non-repudiation	✓		✓				✓	
Availability			✓	✓				

From Stallings' book (Book 1), based on X.800.

✓ indicates the mechanism is relevant to provide the service.

Risk management

A key tool in information security management:

1. Identify threats
2. Classify all threats according to likelihood and severity
3. Apply security controls based on cost benefit analysis

For more details, see:

- ▶ NIST Special Publication 800-30: Guide for Conducting Risk Assessments
- ▶ ISO 27000 standards.

Lecture 3: Number Theory and Finite Fields (Discrete Mathematics)

COSC362 Data and Network Security

Book 1: Chapter 2

Spring Semester, 2021

Motivation

- ▶ Cryptology makes use of mathematics, computer science and engineering.
- ▶ Mostly discrete mathematics because cryptology deals with finite objects such as alphabets and blocks of characters.
- ▶ Looking at modular arithmetic which only deals with a finite number of values.
- ▶ Understanding the algebraic structure of finite objects helps to build useful cryptographic properties.

Outline

Basic Number Theory

Primes and Factorisation

GCD and the Euclidean Algorithm

Modular Arithmetic

Groups and Fields

Boolean Algebra

Outline

Basic Number Theory

Primes and Factorisation

GCD and the Euclidean Algorithm

Modular Arithmetic

Groups and Fields

Boolean Algebra

Factorisation

- ▶ $\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$ is the set of integers.
- ▶ Given $a, b \in \mathbb{Z}$, a divides b if there exists $k \in \mathbb{Z}$ s.t. $ak = b$.
 - ▶ a is a factor of b
 - ▶ $a|b$
- ▶ An integer $p > 1$ is a *prime* number if its only divisors are 1 and p :
 - ▶ Examples: 2, 3, 5, 11, 13, 17, 19, etc.
- ▶ Testing prime numbers p by trial numbers, up to the square root of p (i.e. \sqrt{p}).
- ▶ There are more efficient ways to check for primality (later in the course).

└ Basic Number Theory

└ Primes and Factorisation

Basic Properties of Factors

- ▶ If a divides b AND a divides c , then a divides $b + c$.
- ▶ If p is a prime and p divides ab , then p divides a OR b .

Example:

$$6|18 \text{ and } 6|24 \Rightarrow 6|42$$

$$7|42 \Rightarrow 7|3 \text{ or } 7|14$$

└ Basic Number Theory

└ Primes and Factorisation

Division Algorithm

Given $a, b \in \mathbb{Z}$, s.t. $a > b$, then there exists $q, r \in \mathbb{Z}$ s.t.

$$a = bq + r$$

where q is the *quotient* and $0 \leq r < b$ is the *remainder*. One can show that $r < a/2$.

Example:

$$17 = 5 \times 3 + 2$$

Greatest Common Divisor (GCD)

d is the GCD of a and b , written $\gcd(a, b) = d$, if:

- ▶ d divides a AND b
- ▶ if c divides a and b then c divides d
- ▶ $d > 0$

a and b are *relatively prime* when $\gcd(a, b) = 1$.

Euclidean Algorithm

Finding $d = \gcd(a, b)$ as follows:

$$a = bq_1 + r_1 \text{ for } 0 < r_1 < b$$

$$b = r_1 q_2 + r_2 \text{ for } 0 < r_2 < r_1$$

$$r_1 = r_2 q_3 + r_3 \text{ for } 0 < r_3 < r_2$$

⋮

$$r_{k-3} = r_{k-2} q_{k-1} + r_{k-1} \text{ for } 0 < r_{k-1} < r_{k-2}$$

$$r_{k-2} = r_{k-1} q_k + r_k \text{ for } 0 < r_k < r_{k-1}$$

$$r_{k-1} = r_k q_{k+1} \text{ with } r_{k+1} = 0$$

Hence, $d = r_k = \gcd(a, b)$.

└ Basic Number Theory

└ GCD and the Euclidean Algorithm

Euclidean Algorithm

Data: a, b **Result:** $\gcd(a, b)$ $r_{-1} \leftarrow a;$ $r_0 \leftarrow b;$ $k \leftarrow 0;$ **while** $r_k \neq 0$ **do**
$$q_k \leftarrow \left\lfloor \frac{r_{k-1}}{r_k} \right\rfloor;$$
$$r_{k+1} \leftarrow r_{k-1} - q_k r_k;$$
$$k \leftarrow k + 1;$$
end
$$k \leftarrow k - 1;$$
return r_k

└ Basic Number Theory

└ GCD and the Euclidean Algorithm

Back Substitution

Finding integers x, y in:

$$ax + by = d = r_k$$

by using back substitution in the Euclidean algorithm.

We have from the previous slide:

$$r_{k-3} = r_{k-2}q_{k-1} + r_{k-1}$$

$$r_{k-2} = r_{k-1}q_k + r_k$$

Rewriting:

$$r_{k-1} = r_{k-3} - r_{k-2}q_{k-1}$$

$$r_k = r_{k-2} - r_{k-1}q_k$$

└ Basic Number Theory

└ GCD and the Euclidean Algorithm

Back Substitution

Getting:

$$\begin{aligned} r_k &= r_{k-2} - (r_{k-3} - r_{k-2}q_{k-1})q_k \\ &= r_{k-2}(1 + q_{k-1}q_k) - r_{k-3}q_k \end{aligned}$$

by replacing r_{k-1} in the next line UP.

Back Substitution

Using $r_k = r_{k-2}(1 + q_{k-1}q_k) - r_{k-3}q_k$ in the next line UP.

Getting r_k as a multiple of a and a multiple of b by replacing r_1 by $r_1 = a - bq_1$.

An interesting case for us is when $r_k = d = 1$.

Example

- ▶ $\text{gcd}(17, 3) = ?$
- ▶ Euclidean algorithm:

$$\begin{aligned} 17 &= 3 \times 5 + 2 \\ 3 &= 2 \times 1 + 1 \\ 2 &= 1 \times 2 \end{aligned}$$

So $\text{gcd}(17, 3) = 1$

- ▶ Back substitution:

$$\begin{aligned} 1 &= 3 - 2 \times 1 \text{ from the second to last line} \\ &= 3 - (17 - 3 \times 5) \times 1 \text{ from } 2 = 17 - 3 \times 5 \\ &= 17 \times (-1) + 3 \times 6 \text{ by reordering} \end{aligned}$$

Outline

Basic Number Theory

Primes and Factorisation

GCD and the Euclidean Algorithm

Modular Arithmetic

Groups and Fields

Boolean Algebra

Modular Arithmetic

Definition:

b is a residue of $a \pmod n$ if $a - b = kn$ for some integer k :

$$a \equiv b \pmod n \iff a - b = kn$$

(One can also write $a = kn + b$ where n is seen as the quotient and b as the remainder.)

Given $a \equiv b \pmod n$ and $c \equiv d \pmod n$, then:

- ▶ $a + c \equiv b + d \pmod n$
- ▶ $ac \equiv bd \pmod n$
- ▶ $ka \equiv kb \pmod n$

N.B.: one can always reduce the inputs modulo n BEFORE performing multiplication and addition.

Notation: $a \bmod n$

$b \bmod n$ denotes the unique value a in the complete set $\{0, 1, \dots, n - 1\}$ of residues such that:

$$a \equiv b \pmod{n}$$

$b \bmod n$ is the remainder after dividing a by n .

Residue Class

Definition:

The set $\{r_0, r_1, \dots, r_{n-1}\}$ is a complete set of residues modulo n if for every integer a , then $a \equiv r_i \pmod{n}$ for EXACTLY one r_i :

- ▶ Numbers $0, 1, \dots, n - 1$ form a complete set of residues modulo n since $a = qn + r$ for any a (where $0 \leq r < n$).
- ▶ $\{0, 1, \dots, n - 1\}$ is denoted as \mathbb{Z}_n .

Outline

Basic Number Theory

Primes and Factorisation

GCD and the Euclidean Algorithm

Modular Arithmetic

Groups and Fields

Boolean Algebra

Groups

Definition:

A group \mathbb{G} is a set with *binary operation* \cdot and:

- ▶ *Closure*: $a \cdot b \in \mathbb{G}$ for $a, b \in \mathbb{G}$
- ▶ *Identity*: there is an element 1 s.t. $a \cdot 1 = 1 \cdot a = a$ for $a \in \mathbb{G}$
- ▶ *Inverse*: there is an element b s.t. $a \cdot b = 1$ for $a \in \mathbb{G}$
- ▶ *Associativity*: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for $a, b, c \in \mathbb{G}$

A group \mathbb{G} is said to be *abelian* when:

- ▶ *Commutativity*: $a \cdot b = b \cdot a$ for $a, b \in \mathbb{G}$

Cyclic Groups

- ▶ The order $|\mathbb{G}|$ of a group \mathbb{G} is the number of elements in \mathbb{G} .
- ▶ g^k denotes the repeated application of $g \in \mathbb{G}$ using the group operation \cdot .
Example: $g^3 = g \cdot g \cdot g$
- ▶ The order $|g|$ of $g \in \mathbb{G}$ is the smallest integer k s.t. $g^k = 1$.
- ▶ g is a generator for \mathbb{G} if $|g| = |\mathbb{G}|$.
- ▶ A group is *cyclic* if it has a generator.

Cyclic groups are important in cryptography: if we construct a group \mathbb{G} with a large order, then we can be sure that a generator g can also take on the same large number of values.

Computing Inverses modulo n

The inverse of a (if it exists!) is a value x s.t. $ax = 1 \pmod{n}$ and is written $a^{-1} \pmod{n}$.

In cryptosystems, finding inverses enables to decrypt (or undo) certain operations.

Theorem: Let $0 < a < n$, then a has an inverse modulo n IF AND ONLY IF $\gcd(a, n) = 1$.

Modular Inverses using the Euclidean Algorithm

Using the Euclidean algorithm (very efficient) to find the inverse of a .

Given a , we want to find the value x s.t.:

$$ax \equiv 1 \pmod{n}$$

Thus, there is an integer y s.t. $ax = 1 + yn$.

From $\gcd(a, n) = 1$, we write $ax + ny = 1$ and find the integers x, y using back substitution.

Group of prime modulus: \mathbb{Z}_p^*

$\mathbb{Z}_p^* = \{1, 2, \dots, p - 1\}$ is a complete set of residues modulo the prime p with the value 0 removed.

Properties:

- ▶ $|\mathbb{Z}_p^*| = p - 1$
- ▶ \mathbb{Z}_p^* is cyclic
- ▶ \mathbb{Z}_p^* has many generators (in general)

One can see \mathbb{Z}_p^* as the multiplicative group of integers $1, 2, \dots, p - 1$ which have inverses modulo p .

Example: \mathbb{Z}_5^*

- ▶ 5 is prime
- ▶ \mathbb{Z}_5^* is a group and all numbers less than 5 are in the group
- ▶ Obtaining $\mathbb{Z}_5^* = \{1, 2, 3, 4\}$
- ▶ There are indeed $5 - 1 = 4$ elements

Finding a Generator of \mathbb{Z}_p^*

- ▶ A generator of \mathbb{Z}_p^* is an element of order $p - 1$.
- ▶ **Lagrange theorem:** the order of any element must exactly divide $p - 1$.
- ▶ Finding a generator of \mathbb{Z}_p^* as follows:
 1. Compute all the distinct prime factors f_1, f_2, \dots, f_r of $p - 1$.
 2. g is a generator if and only if $g^{(p-1)/f_i} \neq 1 \pmod p$ for $i = 1, 2, \dots, r$.

Group of composite modulus: \mathbb{Z}_n^*

For any n (not prime), \mathbb{Z}_n^* is a group of residues which have an inverse under multiplication.

Properties:

- ▶ \mathbb{Z}_n^* is a group
- ▶ \mathbb{Z}_n^* is NOT cyclic in general
- ▶ Finding its order is difficult in general (when n is big)

Example: \mathbb{Z}_6^*

- ▶ Writing out the numbers less than 6
- ▶ Removing all of those which are not coprime to 6
- ▶ Obtaining $\mathbb{Z}_6^* = \{1, 5\}$

Fields

Definition:

A field \mathbb{F} is a set with *binary operations* $+$ and \cdot , and:

- ▶ \mathbb{F} is an *abelian group* under the operation $+$, with identity element 0
- ▶ $\mathbb{F} \setminus \{0\}$ is an *abelian group* under the operation \cdot , with identity element 1
- ▶ *Distributivity*: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ for $a, b, c \in \mathbb{F}$

Finite Fields

- ▶ Setting up secure communications requires fields with a finite number of elements.
- ▶ **(famous) Theorem:** finite fields exist of size p^n for any prime p and positive integer n , and no finite field exists of other sizes.
- ▶ **Interesting cases for us:** fields of size p for a prime p and fields of size 2^n for some integer n .

Finite Field $GF(p)$

- ▶ $GF(p) = \mathbb{Z}_p$
- ▶ Multiplication and addition done modulo p
- ▶ Its multiplicative group is exactly \mathbb{Z}_p^*
- ▶ Some public key encryption and digital signature schemes use $GF(p)$ (see later)

Finite Field $GF(2)$

- ▶ $GF(2)$ is the simplest field, with 2 elements 0 and 1
- ▶ **Addition modulo 2:** the same as the logical XOR (exclusive-OR) operation
- ▶ **Only one non-zero element:** trivial multiplicative group with the single element 1
- ▶ XOR is often used in crypto, written \oplus :
 - ▶ For bit strings a, b , $a \oplus b$
 - ▶ **Example:** $101 \oplus 011 = 110$

Finite Field $GF(2^8)$

- ▶ Field used for calculations in AES (block cipher)
- ▶ Arithmetic in this field considered as polynomial arithmetic where the field elements are polynomials with binary coefficients
 - ▶ Equating any 8-bit string with a polynomial in a natural way
 - ▶ **Example:** $00101101 \leftrightarrow x^5 + x^3 + x^2 + 1$
- ▶ Polynomial division can be done very efficiently in hardware using shift registers

Arithmetic in $GF(2^8)$

- ▶ Adding 2 strings by adding their coefficients modulo 2 (XOR)
- ▶ Multiplication done with respect to a generator polynomial
 - ▶ for AES, $m(x) = x^8 + x^4 + x^3 + x + 1$
- ▶ Multiplying 2 strings by multiplying them as polynomials and taking their remainder after dividing by $m(x)$

Outline

Basic Number Theory

Primes and Factorisation

GCD and the Euclidean Algorithm

Modular Arithmetic

Groups and Fields

Boolean Algebra

Boolean Values

- ▶ Boolean variable x takes values 0 or 1 (representing *false* and *true* respectively)
- ▶ Boolean function has its output in the set $\{0, 1\}$
- ▶ Truth table used to represent boolean function

Truth Tables

Logical AND (equivalent to multiplication modulo 2):

x_1	x_2	$x_1 \wedge x_2$
1	1	1
1	0	0
0	1	0
0	0	0

Negation:

x	$\neg x$
1	0
0	1

Logical OR:

x_1	x_2	$x_1 \vee x_2$
1	1	1
1	0	1
0	1	1
0	0	0

Lecture 4: CrypTool

COSC362 Data and Network Security

CrypTool website and Youtube channel

Spring Semester, 2021

Motivation

- ▶ CrypTool is an open-source project that focuses on the free e-learning software CrypTool illustrating cryptographic and cryptanalytic concepts.
- ▶ CrypTool is worldwide the most widespread e-learning software in the field of cryptology.

Outline

CrypTool project and CryptTool 2 software

What is CrypTool?

- ▶ The CrypTool project aims to raise awareness and interest in encryption techniques for everyone.
- ▶ CrypTool 2 (CT2) is a modern e-learning program for Windows, which visualizes cryptography and cryptanalysis:
 - ▶ It includes the encryption and cryptanalysis of ciphers, along with their basics and the whole spectrum of modern cryptography.
- ▶ We will use CT2 in our Labs 3, 4 and 7.

Downloading CrypTool

Disclaimer: We will use CT2 so please download that version!

<https://www.cryptool.org/en/ct2/downloads>

Using CrypTool

Disclaimer: This is the video you must watch for Lecture 4!

<https://www.youtube.com/watch?v=dELT2-Vgsr8>

Additional material

- ▶ **CrypTool portal:** <https://www.cryptool.org/en/>
- ▶ **Wikipedia:**
<https://en.wikipedia.org/wiki/CrypTool>
- ▶ **Youtube channel:** https://www.youtube.com/channel/UC8_FqvQWJfZYxcSoEJ5ob-Q
- ▶ **CT2 webpage:** <https://www.cryptool.org/en/ct2/>

Lecture 5: Classical Encryption Part 1

COSC362 Data and Network Security

Book 1: Chapter 3 – Book 2: Chapter 2

Spring Semester, 2021

Motivation

Studying historical ciphers in order to:

- ▶ Establish basic notation and terminology
- ▶ Introduce basic cryptographic operations still used as building blocks for modern cryptographic algorithms
- ▶ Explore typical attacks and adversary capabilities that cryptosystems should defend against

Some books:

- ▶ “The Codebreakers” by D. Kahn about history of cryptography
- ▶ “The Code Book” by S. Singh about classical and modern cryptography

Outline

Introduction

- Basic Definitions

- Cryptanalysis

- Statistics of Natural Language

Transposition Ciphers

Simple Substitution Ciphers

- Caesar Cipher

- Random Simple Substitution Cipher

Outline

Introduction

 Basic Definitions

 Cryptanalysis

 Statistics of Natural Language

Transposition Ciphers

Simple Substitution Ciphers

 Caesar Cipher

 Random Simple Substitution Cipher

Terminology

The science of cryptology has two facets:

- ▶ *Cryptography*: the study of designing cryptosystems
- ▶ *Cryptanalysis*: the study of breaking cryptosystems

These facets are generally studied together.

Another facet (not covered in this course) could be:

- ▶ *Steganography*: the study of concealing information

Confidentiality and Authentication

- ▶ Cryptography is the science of *secret writing*:
 - ▶ Transformations of data depending on a secret called the *key*.
- ▶ Cryptography used to provide *confidentiality* and *authentication* (or *integrity*):
 - ▶ Confidentiality: a key is needed to *read* the message.
 - ▶ Authentication: a key is needed to *write* the message.

Cryptosystems

A cryptosystem consists of:

- ▶ a set of *plaintexts* (holding the original message)
- ▶ a set of *ciphertexts* (holding the encrypted message)
- ▶ a set of *keys*
- ▶ a function, called *encryption* or *encipherment*, which transforms the plaintext into a ciphertext
- ▶ an inverse function, called *decryption* or *decipherment*, which transforms the ciphertext back into the plaintext

The ciphertext is sometimes called *cryptogram*.

Symmetric and Asymmetric Cryptography

- ▶ Symmetric key cipher (secret key cipher):
 - ▶ Encryption and decryption keys are known ONLY to the sender and receiver.
 - ▶ Secure channel for transmission of the keys.
- ▶ Asymmetric key cipher (public key cipher):
 - ▶ Each participant has a public key AND a private key.
 - ▶ Possibly working for both encryption of messages and creation of digital signatures.

Notation for Symmetric Encryption Algorithms

- ▶ Encryption function E
- ▶ Decryption function D
- ▶ Message or plaintext M
- ▶ Cryptogram or ciphertext C
- ▶ Shared secret key K

Encryption is denoted as $C = E(M, K)$

Decryption is denoted as $M = D(C, K)$

Methods of Cryptanalysis

An adversary has access to many methods to break a cryptosystem, such as:

- ▶ What are the resources available to the adversary?
Examples: computational capability, inputs/outputs of the system.
- ▶ What is the adversary aiming to achieve?
Examples: retrieving the whole secret key, distinguishing two messages (e.g. YES and NO). **Why is this important?**

Exhaustive Key Search

- ▶ **Basic method:** exhaustive key search (or brute force attack) where the adversary tries ALL possible keys.
- ▶ NO ONE can prevent such attack:
 - ▶ All cryptosystems must have ENOUGH keys to make exhaustive search too difficult computationally.
 - ▶ The adversary may find the key without trying exhaustive search!
 - ▶ The adversary may break the cryptosystem without finding the key!

Prevention of exhaustive key search is a *minimum* standard.

Attack Classification

- ▶ *Ciphertext Only Attack*: the attacker has access to ONLY intercepted ciphertexts.
- ▶ *Known Plaintext Attack*: the attacker knows a small amount of plaintexts and their corresponding ciphertexts.
- ▶ *Chosen Plaintext Attack*: the attacker can obtain the ciphertext from some plaintext that it has selected (the attacker has an “inside encryptor” available).
- ▶ *Chosen Ciphertext Attack*: the attacker can obtain the plaintext from some ciphertext that it has selected (the attacker has an “inside decryptor” available).

Which Attacks Should Be Prevented?

- ▶ A cryptosystem is seen as *highly insecure* if it can be practically attacked using only intercepted ciphertexts.
- ▶ A cryptosystem should be secure against chosen plaintext and chosen ciphertext attacks (modern standard).
- ▶ History shows that chosen ciphertext attacks are practical to set up for an attacker.

Kerckhoffs' Principle

Kerckhoffs' Principle: The attacker has complete knowledge of the cipher (i.e. the decryption key is the only item UNKNOWN to the attacker):

- ▶ History has shown that it is a reasonable assumption. **Can we think of any examples?**
- ▶ Using a secret, non-standard algorithm can cause severe problems:
 - ▶ This would be an example of *security through obscurity*.

Alphabets

- ▶ **Historical ciphers:** defining the alphabet for the plaintext and ciphertext (usually the same).
- ▶ **Roman alphabet:** A, B, C, \dots, Z .
Sometimes are included: space, upper and lower case, punctuation.
- ▶ Sometimes, alphabet is mapped to numbers:
 $A = 0, B = 1, C = 2, \dots, Z = 25$. And space is 26.
- ▶ Real-world attackers would need to work out the alphabet.

└ Introduction

 └ Statistics of Natural Language

Statistical Attacks

- ▶ Statistical attacks depend on using the redundancy of the alphabet. **Can you read this?** TDY S VRY CLD
- ▶ Information from distribution of single letters, digrams (double letters) and trigrams (triple letters) helps in the attack.
- ▶ Exact statistics of a language vary according to what sample is taken.

Sample Statistics for English

- ▶ The following statistics give a typical distribution of *English text* (calculated on a text passage of 143000 characters).
- ▶ **Simplifying the statistics:** the text is restricted to a plaintext alphabet of 27 characters:
 - ▶ ABCDEFGHIJKLMNOPQRSTUVWXYZ▽ with ▽ being space.
- ▶ Proportions shown are relative:
 - ▶ *Example:* ▽ accounts for 14.6% of all characters while E▽ accounts for 2.3% of all digrams.

Single Character Percentage Frequencies

▽ 14.6	A 7.0	H 2.6	V 1.3	Z 0.1
E 10.1	R 5.2	M 2.5	B 1.3	J 0.1
N 7.8	S 5.1	P 2.5	Y 0.8	Q 0.1
T 7.5	L 3.7	U 2.4	W 0.6	
I 7.1	C 3.5	G 1.7	K 0.2	
O 7.0	D 3.5	F 1.6	X 0.1	

Most Common Digram Percentage Frequencies

E∇ 2.3	D∇ 1.7	ES 1.3	RE 1.1
∇A 2.1	TI 1.7	AT 1.3	IO 1.1
ON 1.9	AN 1.6	ND 1.3	∇I 1.1
IN 1.9	EN 1.6	N∇ 1.3	ME 1.0
∇T 1.8	TH 1.6	AL 1.2	ER 0.9
S∇ 1.7	NT 1.4	HE 1.2	∇O 0.9

Outline

Introduction

Basic Definitions

Cryptanalysis

Statistics of Natural Language

Transposition Ciphers

Simple Substitution Ciphers

Caesar Cipher

Random Simple Substitution Cipher

Basic Cipher Operations

Historical ciphers combine two basic operations:

Transposition: characters in the plaintext are mixed up with each other (permuted).

Substitution: each character (resp. set of characters) is replaced by a different character (resp. set of characters).

Transposition Ciphers

Transposition Cipher
Columnar Transposition

- Plaintext:
MESSAGE FROM MARY STUART KILL THE QUEEN


4	9	1	7	5	3	2	8	6
M	E	S	S	A	G	E	F	R
O	M	M	A	R	Y	S	T	U
A	R	T	K	I	L	L	T	H
E	Q	U	E	E	N			

With 9 columns, we have $9! = 362,880$ possible keys
- Ciphertext:
SMTUESLGYLNMOAEARIERUHSAKEFTTEMRQ
SMTUE SLGYL NMOAE ARIER UHSAK EFTTE MRQ

- ▶ Permuting characters in a fixed period d and permutation f .
- ▶ Plaintext seen as a matrix of rows of length d .
- ▶ Permuting rows/columns and outputting in row/column order.
- ▶ Here, considering permutation of rows and outputting in column order.

Simple Transposition Cipher

- ▶ Key is (d, f)
- ▶ Each block of d characters is re-ordered using permutation f
- ▶ $d!$ permutations of length d :
 - ▶ $d! = d \times (d - 1) \times (d - 2) \times \cdots \times 2 \times 1$
- ▶ *Example:* $d = 10$ gives 3,628,800 possible keys

Cryptanalysis of a Transposition Cipher

- ▶ Frequency distribution of ciphertext characters = Frequency distribution of plaintext characters:
 - ▶ Helping to identify a transposition cipher.
- ▶ If d is small then transposition ciphers solved by hand using *anagramming*:
 - ▶ Restoring disarranged characters to their original positioning.
- ▶ Guessing value of d and writing the ciphertext in columns s.t. there are d rows.
- ▶ Knowledge of plaintext language digrams and trigrams to optimise trials.
- ▶ Automating this process.

Outline

Introduction

Basic Definitions

Cryptanalysis

Statistics of Natural Language

Transposition Ciphers

Simple Substitution Ciphers

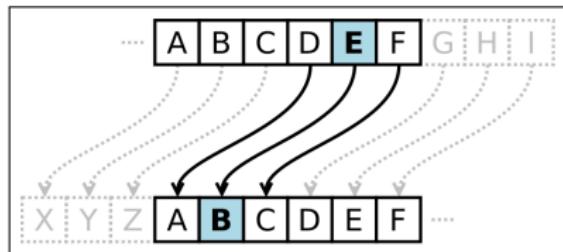
Caesar Cipher

Random Simple Substitution Cipher

Simple Substitution Ciphers

- ▶ Each character in the plaintext alphabet replaced by a character in the ciphertext alphabet following a substitution table.
- ▶ Also called *monoalphabetic* substitution ciphers.
- ▶ Transposition ciphers permutes PLAINTEXT characters while substitution ciphers permute ALPHABET characters.
- ▶ **Some special cases:**
 - ▶ Caesar cipher
 - ▶ Random simple substitution cipher

Caesar Cipher



- ▶ Moving the i th letter of an alphabet to $(i + j)$ th letter, s.t. the key is j .
- ▶ One can write the substitution table or simply:
 - ▶ **Encryption:** $C_i = (M_i + j) \bmod n$
 - ▶ **Decryption:** $M_i = (C_i - j) \bmod n$
 - ▶ either $n = 26$ or $n = 27$ (size of alphabet)
- ▶ **Example:** $j = 1$ then CIPHER → DJQIFS

Cryptanalysis of the Caesar Cipher

- ▶ Finding where one of the most frequent characters is shifted to.
- ▶ *Example:* given the ciphertext
PACGHJUHHCRICGRFWRUCRICPHGLFLQH
 - ▶ Counting the characters
 - ▶ Most common characters: H and C (frequency is 5 each)
- ▶ Let ∇ be in the alphabet ($n = 27$), finding where it is mapped to?
 - ▶ **Trial 1:** $\nabla \rightarrow H$, i.e. $j = 8$, thus HTV ∇ BM ∇ ∇ VJA ..., so incorrect
 - ▶ **Trial 2:** $\nabla \rightarrow C$, i.e. $j = 3$, thus
MY ∇ DEGREE ∇ OF ∇ DOCTOR ∇ OF ∇ MEDICINE

Random Simple Substitution Cipher

- ▶ Assigning a random character of the alphabet to another character of the alphabet.
- ▶ Encryption and decryption defined by substitution table that randomly permutes the alphabet.
- ▶ If the alphabet has 26 characters, then $26!$ keys. **Why?**
 - ▶ Greater than 10^{26}
 - ▶ Too many keys to search even with modern computers
- ▶ Caesar cipher is a special case.

Example

Message: THE DIVING AND THE MORNING

Substitution table (key):

A → S	J → G	S → M
B → J	K → C	T → O
C → V	L → F	U → Q
D → I	M → K	V → D
E → N	N → B	W → P
F → Y	O → U	X → ▽
G → W	P → H	Y → T
H → A	Q → L	Z → X
I → Z	R → R	▽ → E

Message substitution
(encryption):

T → O	N → B	D → I
H → A	I → Z	▽ → E
E → N	N → B	T → O
▽ → E	G → W	H → A
E → N	▽ → E	E → N
V → D	A → S	▽ → E
E → N	N → B	...

Cryptogram: OANENDNBZBWESBIEOANEKURBZBW

Cryptanalysis of a Random Substitution

- ▶ Using frequency analysis on alphabet characters
- ▶ Deciphering the following ciphertext:

FJLTXCFWKOVLHKJVKCBCOTEEVLPKCKJVJSTWTJYVKJVOJSTSBPLVITWCWPVDBIT
WICKTKQLVPHYTPRBJSTQLVYTKKJSCJETSCGTUHKJPTKYLFRTPETXCBTKJFXCJTJ
STGCZHTVOCGVZJCXTJTLJCJCSHWPLTPOLCWYKFOJSTCQQCLCJHKVQTLCJTKEFJSV
HJCQQLTYFCRZTETCLJSTCXVLJFATXTWJKSVHZPRTYCZYHZCJTPCJCGTLBZVEOFI
HLTKCBQTLYTWTJESFKZCLITFWYVWJFWHVHKVQTLCJFVWFJEVHZPQLVPHYTXVL
TJSCWYHRFYXTJTLKVOICKCBTCLKBCZJJZTZTKKJSCWVWTYTWTJFXTQTLYHRFYX
TJTLJSTYCHKJFYKVPCFKYVWKJCWJZBLTYHQTLCJTPCWPFKWTGTLPTKJLVBTPJST
KVZTQLVPHYJJSCJPFKCQQTCLKFKJSTPFKJFZZTPECJTLWVEVWTYHRFYXTJTLVOE
CJTLQLVPHYTKVLTJSCWYHRFYXTJTLKVOICKJSTTDQTWKTFWECJTLJSTWPVTKWV
JCXVHWJJVCYTWTJFXTQTLYHRFYXTJTLJSTILTCJOCYJVLVOJSTTDQTWKTLKFPTK
FWJSTTZTYJLFYTWTLIBJSTYVKJVOKHGTZZCWTTEFZZRTXFWFHXWCWPJSTITWT
LCZTDQWTKCPZFRFJHX . . .

Frequency Analysis of Ciphertext

No.	Character	%	Frequency
1	T	15.4	110
2	J	10.2	73
3	C	8.3	59
4	K	6.7	48
5	L	6.7	48
6	V	6.3	45

- ▶ E and T are the most frequent characters in English:
 - ▶ E → T and T → J in the substitution table
- ▶ Looking for English words such as THE and other common trigrams.

Using Cryptool

- ▶ Solving random substitution by hand is tedious and requires many trials and errors.
- ▶ Using Cryptool (freeware package):
 - ▶ Some utilities to help us, such as frequency counts.
 - ▶ Cryptool has a ciphertext-only tool to solve simple random substitution (needs a bit of help to get the full answer).

Substitution Table (Key)

Using Cryptool, the substitution table (key) is:

Plaintext	A	B	C	D	E	F	G	H	I
Ciphertext	C	R	Y	P	T	O	I	S	F
Plaintext	J	K	L	M	N	O	P	Q	R
Ciphertext	U	N	Z	X	W	V	Q	M	L
Plaintext	S	T	U	V	W	X	Y	Z	
Ciphertext	K	J	H	G	E	D	B	A	

The plaintext starts with: ITREMAINSFORUSTOSAY...

Lecture 6: Classical Encryption Part 2

COSC362 Data and Network Security

Book 1: Chapter 3 – Book 2: Chapter 2

Spring Semester, 2021

Motivation Reminder

Studying historical ciphers in order to:

- ▶ Establish basic notation and terminology
- ▶ Introduce basic cryptographic operations still used as building blocks for modern cryptographic algorithms
- ▶ Explore typical attacks and adversary capabilities that cryptosystems should defend against

Outline

Polyalphabetic Substitution

- Vigenère Cipher

- Other Polyalphabetic Ciphers

Hill Cipher

Outline

Polyalphabetic Substitution

 Vigenère Cipher

 Other Polyalphabetic Ciphers

Hill Cipher

Defining Polyalphabetic Substitution

- ▶ Using multiple mappings from plaintext to ciphertext.
- ▶ The effect with multiple alphabets is to smooth frequency distribution:
 - ▶ Direct frequency analysis should no longer be effective.
- ▶ Typical polyalphabetic ciphers are periodic substitution ciphers based on a period d .
- ▶ Given d ciphertext alphabets C_0, C_1, \dots, C_{d-1} , let $f_i : A \rightarrow C_i$ be a mapping from the plaintext alphabet A to the i th ciphertext alphabet C_i for $0 \leq i \leq d - 1$.

Encryption Process

A plaintext message

$$M = M_0 \cdots M_{d-1} M_d \cdots M_{2d-1} M_{2d} \cdots$$

is encrypted to

$$E(K, M) = f_0(M_0) \cdots f_{d-1}(M_{d-1}) f_0(M_d) \cdots f_{d-1}(M_{2d-1}) f_0(M_{2d}) \cdots$$

Special case with $d = 1$: the cipher is monoalphabetic (simple substitution cipher)

Random Polyalphabetic Substitution Cipher

► *Key Generation:*

- Select a block length d
- Generate d random simple substitution tables

► *Encryption:*

- Encrypting the i th character by using the substitution table number j such that $i \equiv j \pmod{d}$

► *Decryption:*

- Using the same substitution table as in encryption in order to reverse the simple substitution

Example

Let $d = 3$, thus there are 3 ciphertext alphabets.

Pltxt char.	ABC	DEF	GHI	JKL	MNO
C_1	UWY	SXΔ	TVZ	CEI	AFG
C_2	QLM	PJO	RKN	ΔXS	YUW
C_3	MLQ	RNQ	GFA	ZVT	YWU
Pltxt char.	PQR	STU	VWX	YZΔ	
C_1	BDH	KNR	JOP	LMQ	
C_2	ZVT	FGA	HDB	EIC	
C_3	POJ	HDB	IEC	ΔXS	

If the plaintext is $IT\Delta IS\Delta A\Delta BEAUTIFUL\Delta DAY$ then the ciphertext is $ZGSZFSUCLXQBNNKRSSSQ\Delta$.

Vigenère Cipher

- ▶ Popular form of periodic substitution ciphers based on *shifted* alphabets.
- ▶ The key K is a sequence of characters:
 - ▶ $K = K_0 K_1 \cdots K_{d-1}$
 - ▶ Let M be the plaintext character
 - ▶ For $0 \leq i \leq d - 1$, K_i gives the amount of shift in the i th alphabet, i.e. $f_i(M) = (M + K_i) \bmod n$
 - ▶ $n = 27$ when including space in the alphabet
- ▶ In the 19th century, it was believed to be unbreakable.

Example

Message M	AT ∇ T	HE ∇ T	IME ∇
Key K	LOCK	LOCK	LOCK
$E(K, M)$	LGBC	SSBC	T ∇ GJ

- ▶ Numbering the alphabet:
 $A = 0, B = 1, \dots, Z = 25, \nabla = 26$.
- ▶ In particular, $L = 11, O = 14, C = 2, K = 10$:
 - ▶ the 1st character of each 4-character group is shifted by 11,
 - ▶ the 2nd character is shifted by 14,
 - ▶ the 3rd character is shifted by 2,
 - ▶ the 4th character is shifted by 10.
- ▶ Shifting is computed modulo 27 (the alphabet “wraps around”).

Cryptanalysis of Vigenère Cipher

- ▶ Identify the period length

Different techniques such as:

- ▶ Kasiski method (illustrated below)
- ▶ Cryptool uses autocorrelation to estimate the period automatically

- ▶ Attack separately d substitution tables

Each substitution is just a shift (Caesar cipher):

- ▶ If there is sufficient ciphertext then it is straightforward

Identifying the Period Using Autocorrelation

- ▶ Method used to find the period length d of any periodic polyalphabetic cipher.
- ▶ Given a ciphertext C , computing the correlation between C and its shift C_i for all plausible values i of the period.
- ▶ English is non-random:
 - ▶ Better correlation between two texts with the same size shift than between two texts with different size shifts.
 - ▶ Seeing peaks in the value of C_i when i is a multiple of the period.
 - ▶ Plotting results on a histogram and then identifying the period.

└ Polyalphabetic Substitution

└ Vigenère Cipher

Example

The first characters of a ciphertext C are:

AUVHSGE**PELPEK**QTEDKSFNYJYATCTCCFKTSUTEFBVVHPNMFUHBFNPV
YFVRFVUSPVEVFNAOFLBFYJPFPMTMFFMFVBHFJAENEGVTIGHPWSFU
HPTTMAAGVESGIH**JPELPEK**JPTIGMPTNPGJUAUFOXPBFUIEGTIGFJTEIQ
WFXESYIUJTIGIOVEOVIPPOGCWBKTJPGIKMIQWFXESNOOIHFOIHJTCGIXC
SBNRFCDZFEFRRLZKNUGRFUTFFIOJITKNRWISAFPTTIQUHJIUYATUUSTOVP
DFFBZPOOGOGVHFIRJOAOFSUTAOIEGGAUWRFUWIKCIYESGATUODKAUG
DXKTIVHFVWPERJOETYHJEHJJAWGAMTEBFYSGCPTDFFSUCLMVHFPAUW
RFQFUJEDCSFCNEVHFGXBNTFFSUCTJQNPHHJUCMKEOGBXEJVADJASC
CUGRPHIUOXPIOFEFFAQCRUHRPOTIGNBVUSGOGVHFKNWGSUKGBVIP
PWIKCIOYGTIFPDICDPPHBPDUJESGWBUSPOEUJIOIIJITOATVESNYHTATR
OGCSJVUBVIPPAOFHJUKFGNJPJCJUIWGRFCSPPIO**WIKCIOAEGIUCPMGAT**
WRFVONGTPUTVFYIKSTASUGMPHWPTKBPDUQFPNLPYTIGQVKCLUUCVL
FOEUJOEUBZYHJEHIGDJUEOVAOILFFTIGMPUTJPEYVRJEACNENASUGRJ

Example Step 1

Identifying the period length d :

- ▶ Noting that sequences **PELPEK** and **WIKCIO** occur multiple times.
- ▶ Positions of some pairs of such strings are separated by 117 and 93 characters.
- ▶ Period is almost certain to be 1 or 3:
 - ▶ the only common divisors of 117 and 93 are 1 and 3.

This is the **Kasiski method**. One can also automate the process by plotting the autocorrelation (Cryptool).

Example Step 2

Attacking separately 3 different alphabets:

- ▶ Finding the shift for each alphabet as in Caesar cipher.
- ▶ Looking for character with the largest frequency, assuming this is shifted from E.
- ▶ Turning out that:
 - ▶ the first has key A (shift of 0)
 - ▶ the second has key B (shift of 1)
 - ▶ the third has key C (shift of 2)

The plaintext starts with:

ATTTHREEOCLOCKPRECISELYIWASATBAKERSTREET...

Other Ciphers Designed for Use by Hand

- ▶ **Beaufort cipher:** similar to Vigenère cipher but using the substitution $f_i(M) = (K_i - M) \bmod n$.
- ▶ **Autokey cipher:** starting off as Vigenère cipher but using the plaintext to define subsequent alphabets once the alphabets defined by the key have been used.
 - ▶ this cipher is NOT periodic
- ▶ **Running key cipher:** using a (practically) infinite set of alphabets from a shared key.
 - ▶ in practice, the shared key is an extract from a book called *book cipher*

Rotor Machines

► Early 20th century

Electromechanical machines developed for encryption using *rotors* as moving alphabets.

► World War II

The famous *Enigma* machine used by the Germans:

- ▶ each character is encrypted using a different alphabet
- ▶ the period is of about 17,000 so in practice it would never repeat the same message
- ▶ nice simulation in Cryptool

Outline

Polyalphabetic Substitution

Vigenère Cipher

Other Polyalphabetic Ciphers

Hill Cipher

Hill Cipher

- ▶ The American mathematician Lester S. Hill published his cipher in 1929.
- ▶ *Polygram cipher* (also polygraphic cipher):
 - ▶ simple substitution cipher on an extended alphabet consisting of multiple characters
 - ▶ *Example:* digram substitution in which the alphabet consists of all pairs of characters
- ▶ **Major weakness:** its linearity, hence known plaintext attacks are easy.

Definition

Performing a linear transformation on d plaintext characters to get d ciphertext characters:

- ▶ Encryption involves multiplying a $d \times d$ matrix K by the block of plaintext M .
- ▶ Decryption involves multiplying the matrix K^{-1} by the block of ciphertext C .

Encryption: $C = KM$

Decryption: $M = K^{-1}C$

Encryption Example

- ▶ Let $d = 2$ so encryption takes digrams as input and outputs blocks.
- ▶ Each plaintext pair is written as a column vector, letters are encoded as numbers.
- ▶ Suppose the 1st pair for encryption is EG:
 - ▶ E = 4 and G = 6 in our encoding
 - ▶ represented as $\begin{pmatrix} 4 \\ 6 \end{pmatrix}$
- ▶ If insufficient letters to fill a block then padding:
 - ▶ it can be done with uncommon letter such as Z
- ▶ Computations take place modulo 27.

Encryption and Decryption

$$d = 2, K = \begin{pmatrix} 4 & 6 \\ 1 & 7 \end{pmatrix}, K^{-1} = \begin{pmatrix} 4 & 12 \\ 11 & 10 \end{pmatrix}$$

One can check that $KK^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

Plaintext: $M = (\text{EG}) = \begin{pmatrix} 4 \\ 6 \end{pmatrix}$

Encryption: $C = KM = \begin{pmatrix} 4 & 6 \\ 1 & 7 \end{pmatrix} \begin{pmatrix} 4 \\ 6 \end{pmatrix} = \begin{pmatrix} 25 \\ 19 \end{pmatrix} = (\text{ZT})$

Decryption: $M = K^{-1}C = \begin{pmatrix} 4 & 12 \\ 11 & 10 \end{pmatrix} \begin{pmatrix} 25 \\ 19 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \end{pmatrix} = (\text{EG})$

Cryptanalysis of Hill Cipher

- ▶ Known plaintext attacks possible given d plaintext-ciphertext matching blocks.
- ▶ Given blocks (column vectors) M_i and C_i for $0 \leq i \leq d - 1$:
 - ▶ $C = [C_0 C_1 \cdots C_{d-1}]$
 - ▶ $M = [M_0 M_1 \cdots M_{d-1}]$
 - ▶ Solving $C = KM$ for K
 - ▶ $M = K^{-1}C$

Cryptanalysis Example

- ▶ Let $d = 2$ be known
- ▶ Ciphertext is PE▽TBEDLSTE▽HNFQTBR_LHIDB
- ▶ Known plaintext is the 2 first blocks FR and OM (the first word is FROM)

Example Step 1

Encoding the plaintext and ciphertext:

$$M_0 = (\text{FR}) = \begin{pmatrix} 5 \\ 17 \end{pmatrix}, M_1 = (\text{OM}) = \begin{pmatrix} 14 \\ 12 \end{pmatrix}$$

$$C_0 = (\text{PE}) = \begin{pmatrix} 15 \\ 4 \end{pmatrix}, C_1 = (\nabla\text{T}) = \begin{pmatrix} 26 \\ 19 \end{pmatrix}$$

Therefore $M = [M_0 \ M_1] = \begin{pmatrix} 5 & 14 \\ 17 & 12 \end{pmatrix}$ and $C = [C_0 \ C_1] = \begin{pmatrix} 15 & 26 \\ 4 & 19 \end{pmatrix}$

Example Step 2

Recovering encryption matrix K :

$$C = KM \text{ with } K = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$\text{Hence } \begin{pmatrix} 15 & 26 \\ 4 & 19 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 5 & 14 \\ 17 & 12 \end{pmatrix}$$

$$\text{And so } \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 15 & 26 \\ 4 & 19 \end{pmatrix} \begin{pmatrix} 5 & 14 \\ 17 & 12 \end{pmatrix}^{-1} = \begin{pmatrix} 13 & 5 \\ 2 & 6 \end{pmatrix}$$

Example Step 3

Computing K^{-1} and decrypting the ciphertext:

$$K = \begin{pmatrix} 13 & 5 \\ 2 & 6 \end{pmatrix} \text{ and thus } K^{-1} = \begin{pmatrix} 12 & 17 \\ 23 & 26 \end{pmatrix}$$

$$M = K^{-1}C \text{ with } C = \begin{pmatrix} B & D \\ E & L \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 4 & 11 \end{pmatrix} \text{ (3rd and 4th blocks)}$$

$$M = \begin{pmatrix} 12 & 17 \\ 23 & 26 \end{pmatrix} \begin{pmatrix} 1 & 3 \\ 4 & 11 \end{pmatrix} = \begin{pmatrix} 26 & 7 \\ 19 & 4 \end{pmatrix} = \begin{pmatrix} \nabla & H \\ T & E \end{pmatrix}$$

The plaintext is FROM∇THE∇REMINISCENCES∇O

Comments on Cryptanalysis of Hill Cipher

- ▶ In known plaintext attacks, equations may not be fully determined:
 - ▶ Step 2 will fail since matrix not invertible
 - ▶ Further plaintext/ciphertext characters can be examined
- ▶ Ciphertext only attacks follow known plaintext attacks with extra task of finding probable blocks of matching plaintext-ciphertext:
 - ▶ *Example:* when $d = 2$, frequency distribution of non-overlapping pairs of ciphertext characters can be compared with distribution of pairs of plaintext characters
- ▶ Automated cryptanalysis in Cryptool, assuming encoded alphabet with $A = 1, B = 2, \dots, Z = 26, \nabla = 27$.

Do we have some time left?

Yes, then let's start Lecture 7!

Lecture 7: Block Ciphers

COSC362 Data and Network Security

Book 1: Chapters 4 and 6 – Book 2: Chapters 2 and 20

Spring Semester, 2021

Motivation

- ▶ Block ciphers are the main bulk encryption algorithms used in commercial applications.
- ▶ Standardised block cipher AES and legacy cipher DES are widely deployed in real applications.
- ▶ AES algorithm validation list (by NIST) includes over 5000 implementations:
 - ▶ USB drives
 - ▶ door controllers
 - ▶ media server encryption
 - ▶ disk encryption
 - ▶ Bluetooth devices

Outline

Block Cipher Principles

- Product Cipher and Iterated Cipher

- Substitution-Permutation Network

- Feistel Cipher

- Standard Security Properties

DES

- History

- Algorithm

- Brute Force Attack

- Double and Triple DES

AES

- History

- Algorithm

- Comparisons between AES and DES

Conclusion

Outline

Block Cipher Principles

- Product Cipher and Iterated Cipher

- Substitution-Permutation Network

- Feistel Cipher

- Standard Security Properties

DES

- History

- Algorithm

- Brute Force Attack

- Double and Triple DES

AES

- History

- Algorithm

- Comparisons between AES and DES

- Conclusion

Block Ciphers

- ▶ Symmetric key ciphers where each block of plaintext is encrypted with the SAME key.
- ▶ A *block* is a set of plaintext symbols of a fixed size:
 - ▶ Typical block sizes for modern block ciphers between 64 and 256 bits.
- ▶ Used in certain configurations called *modes of operation* (next lecture).

Notations

- ▶ Plaintext block P (length is n bits)
- ▶ Ciphertext block C (length is n bits)
- ▶ Key K (length is k bits)
- ▶ *Encryption:* $C = E(P, K)$
- ▶ *Decryption:* $P = D(C, K)$

Criteria for Block Cipher Design

Claude Shannon discussed 2 important encryption techniques in 1940:

- ▶ *Confusion*: involving substitution to make the relationship between K and C as complex as possible.
- ▶ *Diffusion*: involving transformations to dissipate the statistical properties of P across C .

Shannon proposed to use these techniques repeatedly using the concept of *product cipher*.

Product Cipher

- ▶ Cryptosystem where encryption is formed by applying (also *composing*) several sub-encryption functions.
- ▶ Most block ciphers are composition of simple functions f_i , for $1 \leq i \leq r$, s.t. each f_i has its own key K_i :

$$C = E(P, K) = f_r(\dots(f_2(f_1(P, K_1), K_2)\dots), K_r)$$

Iterated Cipher

Most modern block ciphers are special product ciphers, called *iterated ciphers*:

- ▶ Encryption is divided into r similar *rounds*.
- ▶ Sub-encryption functions are all the same function g , called the *round function*.
- ▶ Key K_i is derived from the overall master key K :
 - ▶ K_i is called the *round key* or *subkey*
 - ▶ K_i is derived from K using a process called *key schedule*

Encryption in Iterated Ciphers

Given a plaintext block P , a round function g and round keys K_1, K_2, \dots, K_r , the ciphertext block C is derived through r rounds as follows:

$$W_0 = P$$

$$W_1 = g(W_0, K_1) = g(P, K_1)$$

$$W_2 = g(W_1, K_2)$$

...

$$W_r = g(W_{r-1}, K_r) = C$$

Decryption in Iterated Ciphers

There must be an inverse function g^{-1} s.t.

$g^{-1}(g(W, K_i), K_i) = W$ for all keys K_i and blocks W :

$$W_r = C$$

$$W_{r-1} = g^{-1}(W_r, K_r) = g^{-1}(C, K_r)$$

$$W_{r-2} = g^{-1}(W_{r-1}, K_{r-1})$$

...

$$W_0 = g^{-1}(W_1, K_1) = P$$

Decryption is thus the reverse of encryption.

Types of Iterated Cipher

- ▶ *Substitution-Permutation Network (SPN)*
 - ▶ Example: Advanced Encryption Standard (AES)
- ▶ *Feistel Cipher*
 - ▶ Example: Data Encryption Standard (DES)

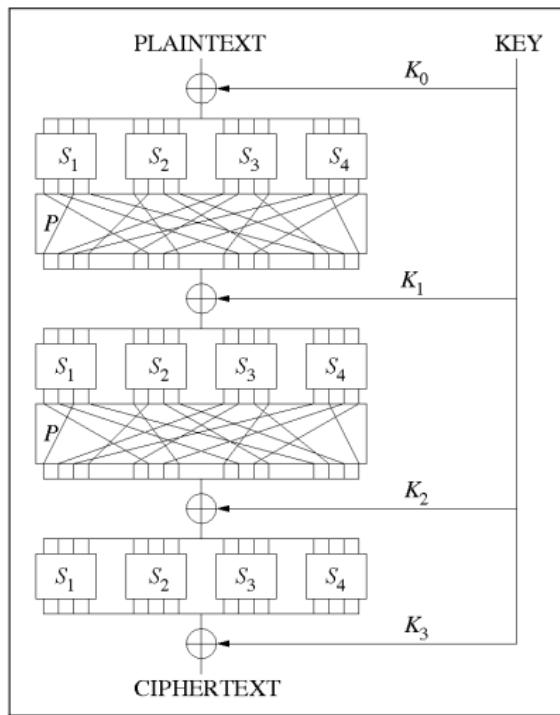
Substitution-Permutation Network

- ▶ Block length n must allow each block to be split into m sub-blocks of length l :
 - ▶ $n = l \times m$
- ▶ 2 operations:
 - ▶ Substitution π_S (called substitution box or simply S-box) operates on sub-blocks of length l bits:
$$\pi_S : \{0, 1\}^l \rightarrow \{0, 1\}^l$$
 - ▶ Permutation π_P (called permutation box or simply P-box) swaps the inputs from $\{1, \dots, n\}$, similarly to transposition ciphers:
$$\pi_P : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$$

Steps in the SPN Round Function

1. Round key K_i is XORed with the current state block W_i :
► $K_i \oplus W_i$
2. Each sub-block is substituted by applying π_S
3. The whole block is permuted using π_P

Illustration with 3 Rounds

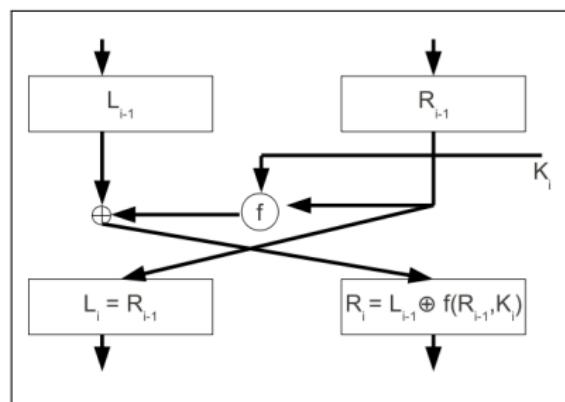


- ▶ Encrypting a plaintext block of n bits into a ciphertext block of n bits.
- ▶ 4 S-boxes S_i ($m = 4$)
- ▶ 1 P-box P
- ▶ 4 Round keys K_i

Feistel Cipher

- ▶ From Horst Feistel, a cryptographer at IBM who influenced the design of DES.
- ▶ Round function swaps the 2 halves of the block and forms a new right hand half.
- ▶ Process sometimes called *Feistel network*:
 - ▶ It can be seen as a network where the 2 halves of plaintext block travel through.

Encryption



1. Split plaintext block $P = W_0$ into 2 halves (L_0, R_0)
2. For each round, perform:
 - ▶ $L_i = R_{i-1}$
 - ▶ $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$
3. Output ciphertext block $C = W_r = (L_r, R_r)$

Decryption

1. Split ciphertext block C into 2 halves (L_r, R_r)
2. For each round, perform:
 - ▶ $L_{i-1} = R_i \oplus f(L_i, K_i)$
 - ▶ $R_{i-1} = L_i$
3. Output plaintext block $P = (L_0, R_0)$
 - ▶ No need to invert f :
 - ▶ Decrypt for ANY function f
 - ▶ Choice of f is critical for security:
 - ▶ f is the only non-linear part of the encryption

Differential and Linear Cryptanalysis

Differential cryptanalysis:

- ▶ First published in 1992.
- ▶ Chosen plaintext attack.
- ▶ Based on the idea that the difference between 2 input plaintexts can be correlated to the difference between 2 output ciphertexts.

Linear cryptanalysis:

- ▶ First published in 1993.
- ▶ Known plaintext attack.
- ▶ Theoretically used to break DES.

Modern block ciphers normally designed to be immune to both differential and linear cryptanalysis.

Avalanche Effects

Key avalanche:

- ▶ A SMALL change in the key (with the same plaintext) should result in a LARGE change in the ciphertext.
- ▶ Related to Shannon's notion of confusion.

Plaintext avalanche:

- ▶ A SMALL change in the plaintext should result in a LARGE change in the ciphertext.
- ▶ Changing 1 bit of plaintext should change each of the bits in the ciphertext with probability $1/2$.
- ▶ Related to Shannon's notion of diffusion.

Outline

Block Cipher Principles

Product Cipher and Iterated Cipher

Substitution-Permutation Network

Feistel Cipher

Standard Security Properties

DES

History

Algorithm

Brute Force Attack

Double and Triple DES

AES

History

Algorithm

Comparisons between AES and DES

Conclusion

Data Encryption Standard (DES)

- ▶ Designed by researchers from IBM.
- ▶ Submitted to the National Bureau of Standards (NBS) in US in a call for publicly available ciphers.
- ▶ Approved in 1977 as the US standard for encryption.
- ▶ Encryption and decryption definitions are public property.
- ▶ Security resides in difficulty of decryption without knowledge of key.
- ▶ 16-round Feistel cipher with key length of 56 bits and data block length of 64 bits.

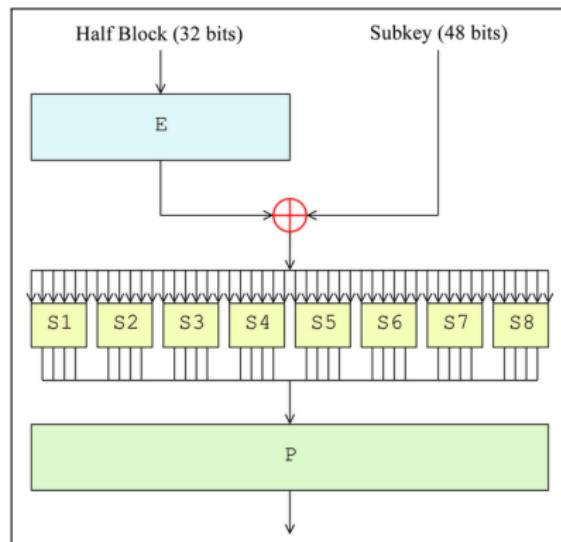
Encryption Steps

P is an input plaintext block of 64 bits:

1. ALL bits of P are permuted using an initial fixed permutation IP .
2. 16 rounds of Feistel operation are applied, denoted by function f :
 - ▶ A different 48-bit subkey is used for each round
3. A final fixed inverse permutation IP^{-1} is applied.

Output the ciphertext block C of 64 bits.

Feistel Operation



For each round:

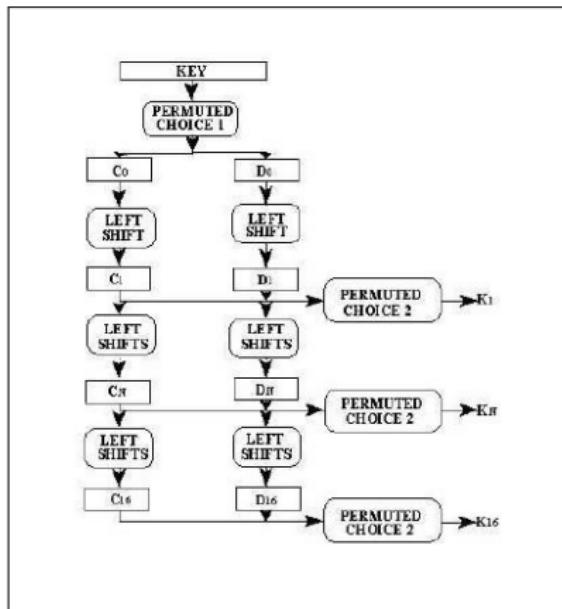
1. Expand 32 bits to 48 bits
2. XOR 48 bits to 48-bit subkey
3. Break 48 bits into 8 blocks of 6 bits
4. Put each block W_i into its substitution table S_i , resulting into blocks of length 4
5. Apply permutation to result into 32 ($= 4 \times 8$) bits.

S-box Example

Row No.	Column No.															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

- ▶ Let input block W be $x_1x_2x_3x_4x_5x_6$
- ▶ Digits x_1 and x_6 define row number between 0 and 3
- ▶ Digits x_2, x_3, x_4, x_5 define column number between 0 and 15
- ▶ **Example:** $W = 100101$
 - ▶ $x_1 = 1$ and $x_6 = 1$, thus 11, that is 3
 - ▶ $x_2 = 0, x_3 = 0, x_4 = 1, x_5 = 0$, thus 0010, that is 2

Key Schedule



- ▶ Each of the 16 rounds involves 48 bits of the 56-bit key
- ▶ Each 48-bit subkey is defined by a series of permutations and shifts on the full 56-bit key

Brute Force Attack

- ▶ Testing all possible 2^k keys in order to find the key K :
 - ▶ k is the size of K
- ▶ Key identified by using a small number of ciphertext blocks or by looking for low entropy in decrypted plaintext.
- ▶ 2^{56} DES keys to test:
 - ▶ On average, it would take $2^{56}/2 = 2^{55}$ trial samples to find the key:
 - ▶ Trying all keys with last bit equal to 0
- ▶ Short DES key size was criticised from the start.

Real World Attacks – Part 1

- ▶ 1997:
 - ▶ \$10,000 DES challenge in February (RSA)
 - ▶ Solved in June
 - ▶ Linked together thousands of computers over the Internet (parallel processing)
- ▶ 1998:
 - ▶ EFF DES cracker built, costing less than \$25,000
 - ▶ Less than 3 days to find 56-bit DES key
 - ▶ Searched 88 billion keys per second
- ▶ 1999:
 - ▶ EFF DES cracker plus distributed search
 - ▶ 22 hours and 15 minutes to find 56-bit DES key
 - ▶ Searched 245 billion keys per second
- ▶ 2007:
 - ▶ Parallel FGPA-based machine Copacobana built, costing \$10,000
 - ▶ Less than 1 week to find 56-bit DES key

Real World Attacks – Part 2

► 2016:

- ▶ Open source password cracking software *hashcat* added in DES brute force searching on general purpose GPUs
- ▶ Systems with 8 GTX 1080 Ti GPUs (each costing \$1,000) recover a key under 2 days

► 2017:

- ▶ Chosen plaintext attacks utilizing rainbow tables (precomputed tables for caching the output of cryptographic hash functions)
- ▶ Recovering the DES key for a single specific chosen plaintext 1122334455667788 in 25 seconds.

Double Encryption

- ▶ Let K_1 and K_2 be 2 block cipher keys.
- ▶ **Encryption:** $C = E(E(P, K_1), K_2)$.
- ▶ If both keys have length k , then exhaustive attacks require 2^{2k-1} trials on average. **Why?** (cf slide 27)
- ▶ Time-memory trade-off which reduces it using Meet-In-The-Middle (MITM) method.

MITM Attack Steps

Let (P, C) be a single plaintext-ciphertext pair:

1. For each key K , store $C' = E(P, K)$ in memory.
2. Check if $D(C, K') = C'$ for any key K' :
 - K from 1. is K_1 and K' from 2. is K_2
3. Check if key values in 2. work for other (P, C) pairs.

MITM Attack Applied to Double DES

It requires:

- ▶ Storage of 1 plaintext block for every key:
 - ▶ Storage of 2^{56} 64-bit blocks
- ▶ A single encryption for every key:
 - ▶ 2^{56} encryption operations
- ▶ A single decryption for every key:
 - ▶ 2^{56} decryption operations

Expensive but much easier than brute force search through $2^{2 \cdot 56 - 1} = 2^{111}$ keys.

Triple Encryption

- ▶ Much better security
- ▶ 3 keys K_1, K_2, K_3
- ▶ **Encryption:** $C = E(D(E(P, K_1), K_2), K_3)$.
- ▶ Secure against MITM attack. **Why?**

Standardised Options

Options for 1999 DES version:

- ▶ 3 independent keys K_1, K_2, K_3
 - ▶ the most secure
- ▶ 2 keys $K_1 = K_3$ and K_2
 - ▶ still secure enough
- ▶ 1 key $K_1 = K_2 = K_3$
 - ▶ backward compatible with single key DES (hence vulnerable to brute force search)

NIST SP 800-131A (2015):

- ▶ 2-key triple DES allowed ONLY for legacy use (decryption only).
- ▶ 3-key triple DES remains approved.

Current Usage

- ▶ OpenSSL does not include Triple DES by default since V1.1.0 (August 2016), considering it as "weak cipher".
- ▶ In December 2018, Microsoft announced the retirement of Triple DES throughout their Office 365 service.

Outline

Block Cipher Principles

Product Cipher and Iterated Cipher

Substitution-Permutation Network

Feistel Cipher

Standard Security Properties

DES

History

Algorithm

Brute Force Attack

Double and Triple DES

AES

History

Algorithm

Comparisons between AES and DES

Conclusion

Advanced Encryption Standard (AES)

- ▶ AES was designed in an open competition due to controversy over DES.
- ▶ Process over several years with much public debate.
- ▶ 15 original submissions.
- ▶ 5 finalists widely believed secure.
- ▶ Winner is “Rijndael” by Belgian cryptographers Vincent Rijmen and Joan Daeman.

Overview

- ▶ 128-bit data block
- ▶ 128-, 192- or 256-bit master key
- ▶ 10, 12 or 14 rounds (for 128-, 192- or 256-bit master key respectively)
- ▶ Byte-based design
- ▶ Substitution-permutation network (SPN):
 - ▶ Initial round key addition
 - ▶ 10, 12 or 14 (encryption/decryption) rounds w.r.t. to the length of the master key
 - ▶ Final round

State Matrix (byte-based)

16-byte data block size:

a_{00}	a_{01}	a_{02}	a_{03}
a_{10}	a_{11}	a_{12}	a_{13}
a_{20}	a_{21}	a_{22}	a_{23}
a_{30}	a_{31}	a_{32}	a_{33}

Mixture of finite field operations in $GF(2^8)$ and bit string operations.

Round Transformation

4 basic operations:

1. ByteSub (non-linear sustitution)
2. ShiftRow (permutation)
3. MixColumn (diffusion)
4. AddRoundKey

- ▶ Substitution-permutation network with block length $n = 128$ and sub-block length $l = 8$.
- ▶ S-box is a look-up table, mathematically defined in $GF(2^8)$.
- ▶ Cryptool has a nice animation of the encryption process.

Key Schedule

- ▶ Master key is 128 bits (resp. 192 and 256).
- ▶ Each of the 10 (resp. 12 and 14) rounds uses a 128-bit subkey.
- ▶ 1 subkey per round + 1 initial subkey:
 - ▶ 11 subkeys in total (resp. 13 and 15)
- ▶ Deriving the 128-bit subkeys from the master key.

Security

- ▶ Some cracks have appeared but no significant breaks.
- ▶ Attacks exist on reduced-round versions.
- ▶ **Related key attack:** requiring the attacker to obtain a ciphertext encrypted with a key related to the actual key in a specified way.
- ▶ Most serious real attacks so far reduce effective key size by around 2 bits.

Comparison

Data block size:

- ▶ DES: 64 bits
- ▶ AES: 128 bits

Key size:

- ▶ DES: 56 bits
- ▶ AES: 128, 192 or 256 bits

Design structure:

- ▶ Both are iterated ciphers
- ▶ DES has a Feistel structure while AES is SPN
- ▶ DES is bit-based and AES is byte-based
- ▶ AES is substantially faster in both hardware and software

Outline

Block Cipher Principles

Product Cipher and Iterated Cipher

Substitution-Permutation Network

Feistel Cipher

Standard Security Properties

DES

History

Algorithm

Brute Force Attack

Double and Triple DES

AES

History

Algorithm

Comparisons between AES and DES

Conclusion

Conclusion

- ▶ Block ciphers are the workhorses of secure communications.
- ▶ AES is the current choice, and Triple DES is still important.
- ▶ Designing good block ciphers is difficult and time-consuming.
- ▶ Block ciphers are used as building blocks for confidentiality and authentication.

Lecture 8: Block Cipher Modes of Operation

COSC362 Data and Network Security

Book 1: Chapters 7 and 12 – Book 2: Chapter 20

Spring Semester, 2021

Motivation

- ▶ Block ciphers encrypt SINGLE blocks of data.
- ▶ In applications, MANY blocks are encrypted sequentially.
- ▶ Breaking the plaintext into blocks and encrypting each separately are generally insecure.
- ▶ Many *modes of operation* standardised with different security and efficiency.
- ▶ Using block ciphers to provide authentication/integrity and/or confidentiality.

Outline

Important Features of Different Modes

Standards

Confidentiality Modes

Electronic Code Book (ECB) Mode

Cipher Block Chaining (CBC) Mode

Counter (CTR) Mode

Authentication Mode

Authenticated Encryption Mode

Outline

Important Features of Different Modes

Standards

Confidentiality Modes

Electronic Code Book (ECB) Mode

Cipher Block Chaining (CBC) Mode

Counter (CTR) Mode

Authentication Mode

Authenticated Encryption Mode

Why Different Modes?

- ▶ Designed to provide *confidentiality* for data OR *authentication* (and integrity) for data OR both:
 - ▶ Modes for confidentiality normally include *randomisation*.
- ▶ Different modes have:
 - ▶ Different efficiency properties
 - ▶ Different communication properties

Randomised Encryption

- ▶ **Problem:** the same plaintext block is encrypted to the same ciphertext block EVERY time:
 - ▶ Allowing patterns to be found in a long ciphertext.
- ▶ **Prevention:** randomising encryption schemes:
 - ▶ By using an *initialisation vector IV* which propagates through the entire ciphertext.
 - ▶ IV may require to be either unique or random.
- ▶ Another way to vary encryption:
 - ▶ By including a variable state which is updated with each block.

Efficiency

Features impacting on efficiency for practical usage (but not necessarily on security):

- ▶ **Parallel processing:**
 - ▶ Multiple plaintext blocks are encrypted in parallel.
 - ▶ Multiple ciphertext blocks are decrypted in parallel.
- ▶ **Error propagation:**
 - ▶ A bit error which occurs in the ciphertext results in multiple bit errors in the plaintext after decryption.

Padding

- ▶ Requiring the plaintext to consist of COMPLETE blocks.
- ▶ NIST suggests a padding method:
 - ▶ Append a single '1' bit to the data string.
 - ▶ Pad the resulting string by as few '0' bits, possibly none, as are necessary to complete the final block.
- ▶ Padding bits removed unambiguously if usage of this padding method is known:
 - ▶ Remove all trailing '0' bits after the last '1' bit.
 - ▶ Remove the single '1' bit.

Notations

- ▶ Plaintext message P (n blocks in length)
- ▶ t -th plaintext block P_t , for $1 \leq t \leq n$
- ▶ Ciphertext message C
- ▶ t -th ciphertext block C_t , for $1 \leq t \leq n$
- ▶ Key K
- ▶ Initialisation vector IV

All modes can be applied to any block cipher.

Example: AES when blocks are 128 bits in length.

NIST Standards

- ▶ ECB, CBC, CFB and OFB originally standardised for use with DES (1980).
- ▶ CTR mode initially added for use with AES (2001).
- ▶ SP 800-38A (2001): Confidentiality Modes
 - ▶ ECB, CBC, CFB and OFB.
 - ▶ An addendum defines Ciphertext Stealing.
- ▶ SP 800-38B (2005): CMAC Mode for Authentication.
- ▶ SP 800-38C (2004, updated 2007): CCM Mode.
- ▶ SP 800-38D (2007): Galois/Counter Mode (GCM).
- ▶ SP 800-38E (2010): XTS-AES Mode for Storage Devices.
- ▶ SP 800-38F (2012): Key Wrapping.
- ▶ SP 800-38G (2016): Format-Preserving Encryption.

Outline

Important Features of Different Modes

Standards

Confidentiality Modes

Electronic Code Book (ECB) Mode

Cipher Block Chaining (CBC) Mode

Counter (CTR) Mode

Authentication Mode

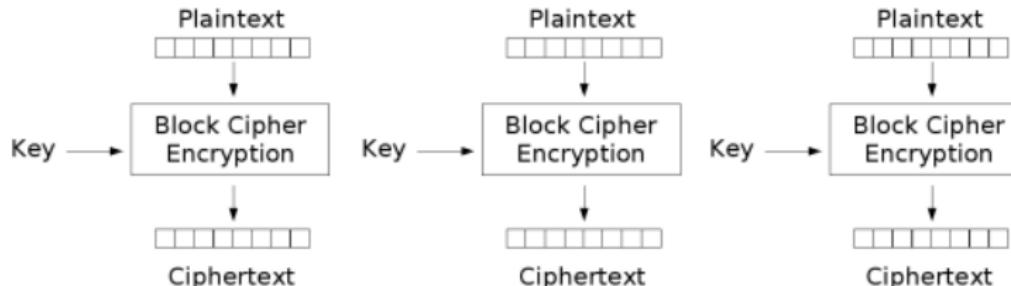
Authenticated Encryption Mode

ECB Mode Encryption

Basic mode of a block cipher.

Encryption:

- ▶ $C_t = E(P_t, K)$
- ▶ Plaintext block P_t is encrypted with key K to produce ciphertext block C_t

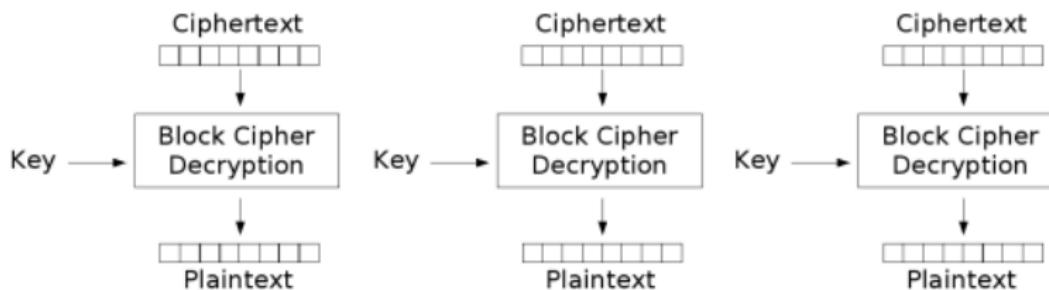


Electronic Codebook (ECB) mode encryption

ECB Mode Decryption

Decryption:

- ▶ $P_t = D(C_t, K)$
- ▶ Ciphertext block C_t is decrypted with key K to produce plaintext block P_t



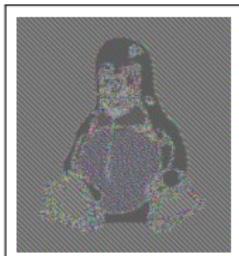
Electronic Codebook (ECB) mode decryption

└ Confidentiality Modes

└ Electronic Code Book (ECB) Mode

ECB Mode Properties

Randomised	✗
Padding	Required
Error propagation	Errors propagate within blocks
IV	None
Parallel encryption	✓
Parallel decryption	✓



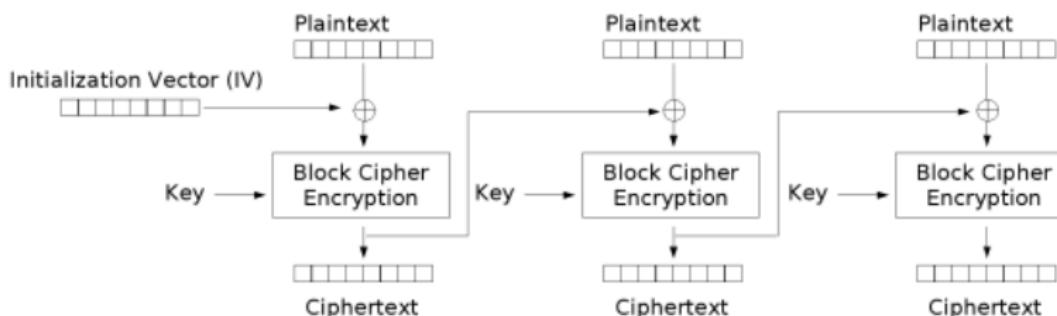
Deterministic ⇒ not normally
used for bulk encryption.

CBC Mode Encryption

“Chaining” blocks together.

Encryption:

- ▶ $C_t = E(P_t \oplus C_{t-1}, K)$ s.t. $C_0 = IV$
 - ▶ IV is chosen at random and sent together with ciphertext blocks
- ▶ P_t is XORed with previous ciphertext block C_{t-1} , and encrypted with key K to produce C_t

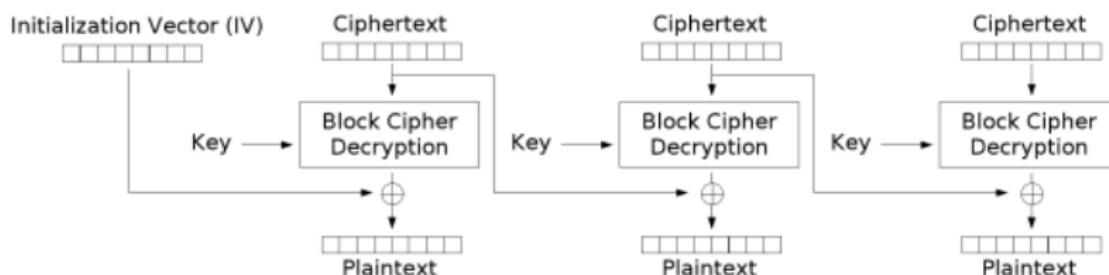


Cipher Block Chaining (CBC) mode encryption

CBC Mode Decryption

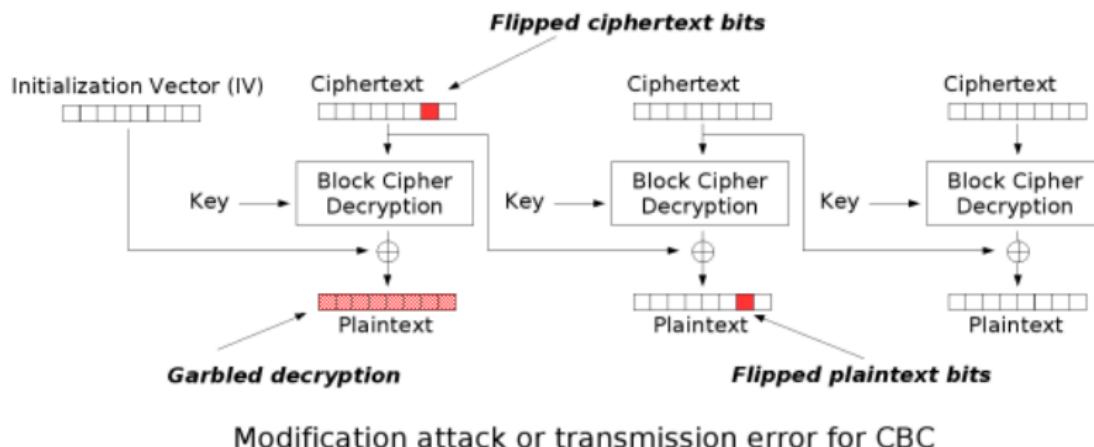
Decryption:

- ▶ $P_t = D(C_t, K) \oplus C_{t-1}$ s.t. $C_0 = IV$
- ▶ C_t is decrypted with key K , and XORed with previous ciphertext block C_{t-1} to produce P_t



Cipher Block Chaining (CBC) mode decryption

CBC Mode Error Propagation



└ Confidentiality Modes

└ Cipher Block Chaining (CBC) Mode

CBC Mode Properties

Randomised	✓
Padding	Required
Error propagation	Errors propagate within blocks and into specific bits of next block
IV	Must be random
Parallel encryption	✗
Parallel decryption	✓

- ▶ Commonly used for bulk encryption.
- ▶ Common choice for channel protection in TLS up to TLS 1.2.

CTR Mode

- ▶ *Synchronous stream cipher mode* (see later).
- ▶ A counter and a nonce are used, initialised using a randomly chosen value N :
 - ▶ $T_t = N||t$ is the concatenation of the nonce N and block number t
 - ▶ $O_t = E(T_t, K)$
 - ▶ This is XORed with the plaintext block P_t
- ▶ **Propagation of channel errors:** A one-bit change in the ciphertext produces a one-bit change in the plaintext at the SAME location.

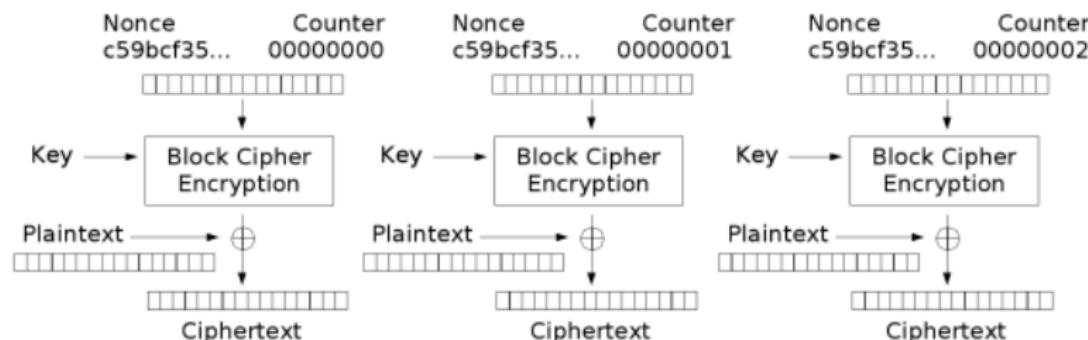
└ Confidentiality Modes

└ Counter (CTR) Mode

CTR Mode Encryption

Encryption:

- ▶ $C_t = O_t \oplus P_t$
- ▶ Plaintext block P_t is XORed with O_t

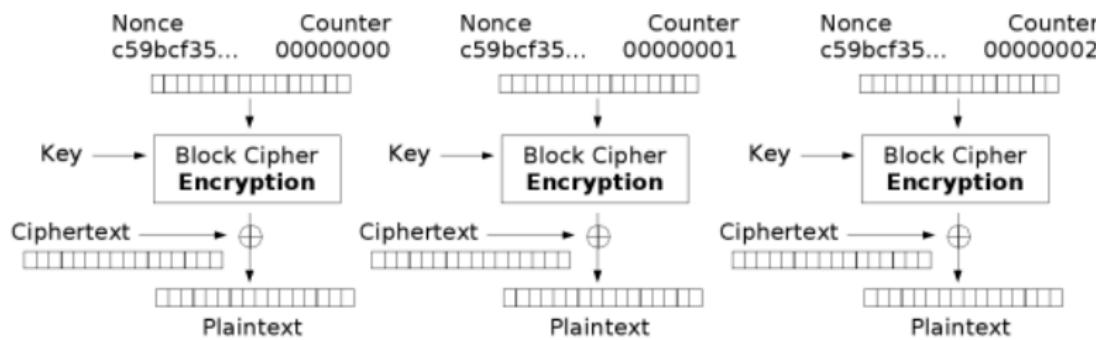


Counter (CTR) mode encryption

CTR Mode Decryption

Decryption:

- ▶ $P_t = O_t \oplus C_t$
- ▶ Ciphertext block C_t is XORed with O_t



Counter (CTR) mode decryption

CTR Mode Properties

Randomised	✓
Padding	Not required
Error propagation	Errors occur in specific bits of current block
IV	Nonce must be unique
Parallel encryption	✓
Parallel decryption	✓

- ▶ Good for access to specific plaintext blocks without decrypting the whole stream.
- ▶ Basis for authenticated encryption in TLS 1.2.

Outline

Important Features of Different Modes

Standards

Confidentiality Modes

Electronic Code Book (ECB) Mode

Cipher Block Chaining (CBC) Mode

Counter (CTR) Mode

Authentication Mode

Authenticated Encryption Mode

Message Integrity

- ▶ Ensuring that messages are not altered in transmission.
- ▶ Treating *message integrity* and *message authentication* as the same thing.
- ▶ Preventing an adversary from re-ordering, replacing, replicating and deleting message blocks in order to alter the received message.
- ▶ Proving message integrity is independent from using encryption for confidentiality.

Message Authentication Code (MAC)

- ▶ Cryptographic mechanism to ensure message integrity.
- ▶ $T = \text{MAC}(M, K)$:
 - ▶ **Inputs:** arbitrary-length message M and secret key K
 - ▶ **Output:** (short) fixed-length tag T
- ▶ Parties Alice and Bob share a common key K .
- ▶ Alice wants to send a message M to Bob:
 - ▶ Alice computes $T = \text{MAC}(M, K)$
 - ▶ Alice sends the message M and adjoins its tag T
 - ▶ Bob computes $T' = \text{MAC}(M', K)$ on received message M' and checks that $T' = T$

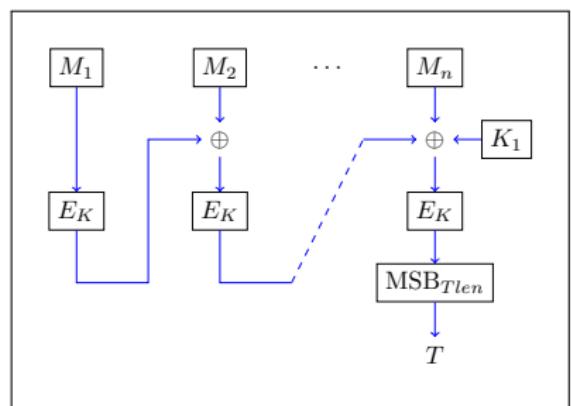
MAC Properties

- ▶ Providing sender authentication to the message:
 - ▶ Only Alice and Bob CAN produce T from M .
 - ▶ If $T' = T$ then Bob concludes that the message received M' was sent by Alice and has not been modified in transit (either intentionally or accidentally).
 - ▶ If $T' \neq T$ then Bob concludes that (M', T) was not sent by Alice.
- ▶ **Basic security property:** Unforgeability
 - ▶ Infeasible to produce M and T s.t. $T = \text{MAC}(M, K)$ without knowledge of K

Basic CBC-MAC

- ▶ Using a block cipher to create a MAC providing message integrity (but not confidentiality).
- ▶ IV must be fixed and public, and can be set to all zeros.
- ▶ CBC-MAC with random IV is NOT secure.
- ▶ P is the message with n blocks.
- ▶ $T = \text{CBC-MAC}(P, K)$ as follows:
 - ▶ $C_t = E(P_t \oplus C_{t-1}, K)$ for $0 \leq t \leq n$, s.t. $C_0 = IV$
 - ▶ $T = C_n$
- ▶ Unforgeable as long as the message length is fixed.

Cipher-based MAC (CMAC)



- ▶ Standardised: NIST secure version of CBC-MAC.
- ▶ 2 keys K_1, K_2 are derived from the original key K .
- ▶ K_1 or K_2 XORed with M_n (padding as necessary).
- ▶ IV set to be the all 0 block.
- ▶ CBC encryption on the message M .
- ▶ T is some number of MSB bits of final block.

CMAC

- ▶ NIST standard allows any number of bit $Tlen$ to be chosen for tag T :
 - ▶ Recommendation of 64 bits to avoid guessing attacks.
- ▶ Standard recommends MAC tag T to be of length at least $\log_2(lim/R)$ with:
 - ▶ lim is a limit on how many invalid messages are detected before K is changed.
 - ▶ R is the acceptable probability (risk) that a false message is accepted.

Outline

Important Features of Different Modes

Standards

Confidentiality Modes

Electronic Code Book (ECB) Mode

Cipher Block Chaining (CBC) Mode

Counter (CTR) Mode

Authentication Mode

Authenticated Encryption Mode

Authenticated Encryption

- ▶ 2 types of input data:
 - ▶ **Payload:** both encrypted and authenticated.
 - ▶ **Associated data:** only authenticated.
- ▶ 2 modes specified in:
 - ▶ NIST SP-800-38C in 2004 for Counter with CBC-MAC (CCM) Mode
 - ▶ NIST SP-800-38D in 2007 for Galois/Counter (GCM) Mode
- ▶ Both modes use CTR mode for confidentiality but add integrity in different ways.
- ▶ Both used in TLS 1.2 and TLS 1.3 (latest versions).

Counter with CBC-MAC (CCM) Mode

Combining CBC-MAC for authentication of ALL data (payload and associated data) and CTR mode encryption for the payload:

- ▶ **Inputs:** nonce N for CTR mode, payload P of P_{len} bits, and associated data A .
- ▶ Format N, A, P to produce a set of blocks.
- ▶ Compute CBC-MAC tag T for these blocks with length T_{len} .
- ▶ Use CTR mode to compute blocks of key stream S_0, S_1, \dots, S_m where $m = \lceil P_{len}/128 \rceil$.
- ▶ **Output:** $C = (P \oplus MSB_{P_{len}}(S))||(T \oplus MSB_{T_{len}}(S_0))$ where $S = S_1, \dots, S_m$.

CCM Mode Format

- ▶ Complex format with restrictions w.r.t. different standards.
- ▶ Lengths of N , P are included in the 1st block.
- ▶ If A is non-zero then formatted from the 2nd block onwards including its length.
- ▶ **Example:** TLS 1.2
 - ▶ Tag T is 8 bytes
 - ▶ CTR mode nonce N is 12 octets
 - ▶ Max payload size is $2^{24} - 1$ bytes

Lecture 9: Pseudorandom Numbers and Stream Ciphers

COSC362 Data and Network Security

Book 1: Chapter 8 – Book 2: Chapter 20

Spring Semester, 2021

Motivation

- ▶ Random values required in many cases in cryptography.
- ▶ For practical reasons, pseudorandom deterministic algorithms are used.
- ▶ Stream ciphers constructed from (pseudo)random number generators.
- ▶ Examples of stream ciphers widely deployed:
 - ▶ A5 cipher used in GSM mobile phones
 - ▶ AES in counter (CTR) mode

Outline

Random Numbers

DRBG

CTR_DRBG

Dual_EC_DRBG

Stream Ciphers

One Time Pad

Visual Cryptography

Prominent Stream Ciphers

A5 Cipher

RC4 Cipher

ChaCha Cipher

Conclusion

Outline

Random Numbers

DRBG

CTR_DRBG

Dual_EC_DRBG

Stream Ciphers

One Time Pad

Visual Cryptography

Prominent Stream Ciphers

A5 Cipher

RC4 Cipher

ChaCha Cipher

Conclusion

Randomness

- ▶ Defining randomness is difficult.
- ▶ **What we want:** any specific string of bits is exactly as random as any other string.
- ▶ *Generators* of random strings:
 - ▶ *True random number generator* (TRNG) is a physical process which outputs each valid string independently with equal probability.
 - ▶ *Pseudorandom number generator* (PRNG) is a deterministic algorithm which approximates a TRNG.
- ▶ Using a TRNG to provide a *seed* for a PRNG.

True Random Number Generator (TRNG)

- ▶ NIST Special Publication 800-90B (Jan. 2016):
 - ▶ Framework for design and validation of TRNG algorithms, called *entropy sources*.
 - ▶ Specification of statistical tests for validating the suitability of entropy sources.
- ▶ The entropy source includes:
 - ▶ A physical noise source
 - ▶ A digitization process
 - ▶ Post-processing stages
- ▶ The output of the entropy source is any requested number of bits.
- ▶ Periodic *health test* to ensure continuing reliable operation.
- ▶ Intel introduced TRNG into Ivy Bridge processors in 2012.

Pseudorandom Number Generator (PRNG)

- ▶ NIST Special Publication 800-90A (June 2015):
 - ▶ Recommendation of specific PRNG algorithms, named *deterministic random bit generator* (DRBG)
 - ▶ DRBG is based on hash functions, a specific MAC (known as HMAC) and block ciphers in counter mode.
- ▶ Each generator takes a seed as input.
- ▶ It outputs a bit string before updating its state.
- ▶ The seed should be updated after some number of calls.
- ▶ The seed can be obtained from a TRNG.

Functions

- ▶ *Instantiate*: setting the initial state of the DRBG using a seed.
- ▶ *Generate*: providing an output bit string for each request.
- ▶ *Reseed*: inputting a new random seed and updating the state.
- ▶ *Test*: checking correct operation of the other functions.
- ▶ *Uninstantiate*: deleting (zeroising) the state of the DRBG.

Security

Security w.r.t. the ability of an attacker to distinguish reliably between its output and a truly random string:

- ▶ ***Backtracking resistance***: an attacker who obtains the current state of the DRBG should not be able to distinguish between the output of *earlier calls* to the function Generate and random strings.
- ▶ ***Forward prediction resistance***: an attacker who obtains the current state of the DRBG should not be able to distinguish between the output of *later calls* to the function Generate and random strings.

CTR_DRBG

- ▶ Using a block cipher in counter (CTR) mode:
 - ▶ **Recommendation:** AES with 128-bit keys
- ▶ DRBG initialised with a seed whose length is equal to the key length PLUS the block length:
 - ▶ $128 + 128 = 256$ for AES with 128-bit master keys
- ▶ Seed defines a key K and a counter value ctr :
 - ▶ No separate nonce as in a normal CTR mode
- ▶ CTR mode encryption is run iteratively, with no plaintext added.
- ▶ The output blocks form the CTR_DRBG output.

Update Function

- ▶ Each request to DRBG generates up to 2^{19} bits.
- ▶ From the function Generate:
 - ▶ (K, ctr) 's state must be updated after each request by generating 2 blocks using the current key to obtain the new key and a counter.
- ▶ Updating provides backtracking resistance.
- ▶ **Restriction:** up to 2^{48} requests to the function Generate before requiring re-seeding.
- ▶ Each re-seed provides forward prediction and backtracking resistance.

└ Random Numbers

└ Dual_EC_DRBG

Dual_EC_DRBG

- ▶ From an older standard (Dec. 2012).
- ▶ Based on elliptic curve discrete logarithm problem:
 - ▶ But no security proof exists
 - ▶ And many flaws:

<https://blog.cryptographyengineering.com/2013/09/18/the-many-flaws-of-dualecdrbg/>
- ▶ Much slower than other DRBGs in the standard.
- ▶ Press (Reuters) reported a secret 10 million dollar deal between NSA and RSA Security company to use Dual_EC_DRBG as the default PRNG in its software suite (Dec. 2013).

Outline

Random Numbers

DRBG

CTR_DRBG

Dual_EC_DRBG

Stream Ciphers

One Time Pad

Visual Cryptography

Prominent Stream Ciphers

A5 Cipher

RC4 Cipher

ChaCha Cipher

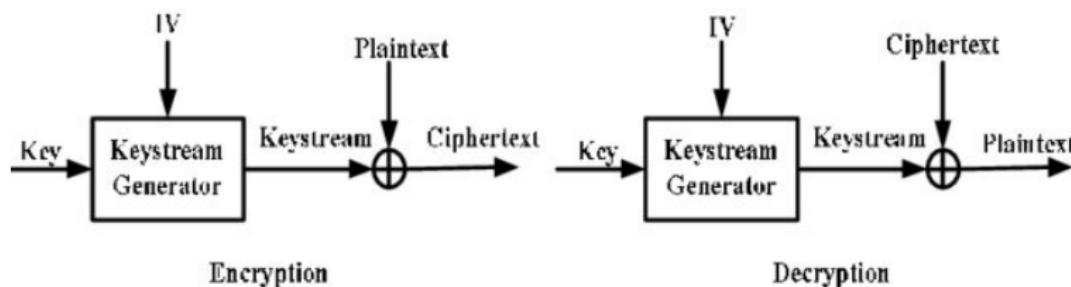
Conclusion

Stream Ciphers

- ▶ Characterised by the generation of a *keystream* using a short key and an initialisation value $/V$.
- ▶ Each element of the keystream is used successively to encrypt 1 or more ciphertext characters.
- ▶ Usually symmetric key ciphers:
 - ▶ The sender and receiver share the same key.
 - ▶ They can generate the same keystream given the same $/V$.

Synchronous Stream Ciphers

- ▶ The keystream is generated *independently* of the plaintext.
- ▶ Both sender and receiver need to generate the same keystream and synchronise on its usage.
- ▶ Vigenère cipher seen as a (periodic) synchronous stream cipher where each shift is defined by a key letter.
- ▶ CTR mode of operation for a block cipher is one method to generate a keystream.



Binary Synchronous Stream Ciphers

For each time interval t :

- ▶ Binary sequence $s(t)$, that is the keystream
- ▶ Binary plaintext $p(t)$
- ▶ Binary ciphertext $c(t)$

Encryption: $c(t) = p(t) \oplus s(t)$

Decryption: $p(t) = c(t) \oplus s(t)$

Outline

Random Numbers

DRBG

CTR_DRBG

Dual_EC_DRBG

Stream Ciphers

One Time Pad

Visual Cryptography

Prominent Stream Ciphers

A5 Cipher

RC4 Cipher

ChaCha Cipher

Conclusion

One Time Pad

- ▶ Often attributed to Vernam who made a one-time pad machine using teletype machinery in 1917 (earlier historical uses are known).
- ▶ Key is a random sequence of characters s.t. all of them are independently generated.
- ▶ Each character in the key is used ONE TIME ONLY.
- ▶ Alphabet of any length but usually:
 - ▶ A natural language alphabet
 - ▶ The binary alphabet $\{0, 1\}$
- ▶ **Example:** (non-periodic) binary synchronous stream cipher.
- ▶ Providing *perfect secrecy*.

Perfect Secrecy

Shannon's definition:

- ▶ Message set $\{M_1, \dots, M_k\}$.
- ▶ Ciphertext set $\{C_1, \dots, C_l\}$.
- ▶ $\Pr(M_i|C_j)$ is the probability that M_i is encrypted given that C_j is observed.
- ▶ In most cases, the messages M_i are NOT be equally likely.
- ▶ For all messages M_i and ciphertexts C_j :

$$\Pr(M_i|C_j) = \Pr(M_i)$$

One Time Pad Using Roman Alphabet

- ▶ Plaintext characters: p_1, \dots, p_r
- ▶ Ciphertext characters: c_1, \dots, c_r
- ▶ Keystream: random characters k_1, \dots, k_r
- ▶ **Encryption:** $c_i = (p_i + k_i) \bmod 26$
- ▶ **Decryption:** $p_i = (c_i - k_i) \bmod 26$
- ▶ Ciphertext is the addition of plaintext and keystream characters, modulo 26.

One Time Pad Perfect Secrecy

- ▶ Let a ciphertext C_j be observed.
- ▶ Any message could have been sent, depending on the keystream.
- ▶ The probability that M_i is sent given that C_j is observed = the probability that M_i is chosen, weighted by the probability that the right keystream is chosen.
- ▶ Each key is chosen with equal probability.
- ▶ Conditional probability is thus $\Pr(M_i|C_j) = \Pr(M_i)$

Example

- ▶ Plaintext: HELLO
- ▶ Keystream: EZABD
- ▶ Ciphertext: LDLMR
- ▶ Given the ciphertext, the plaintext can be ANY 5-letter message.

Vernam Binary One Time Pad

- ▶ Plaintext: binary sequence b_1, \dots, b_r
- ▶ Ciphertext: binary sequence c_1, \dots, c_r
- ▶ Keystream: random binary sequence k_1, \dots, k_r
- ▶ **Encryption:** $c_i \equiv p_i \oplus k_i$
- ▶ **Decryption:** $p_i \equiv c_i \oplus k_i$
- ▶ Keystream is SAME length as plaintext.
- ▶ Providing perfect secrecy since any ciphertext is equally possible given the plaintext.
- ▶ Encryption and decryption are identical processes.

Properties

- ▶ Shannon showed that any cipher with perfect secrecy MUST have as many keys as there are messages.
- ▶ One time pad is the ONLY unbreakable cipher.
- ▶ Practical usage is possible for pre-assigned communications between fixed parties.
- ▶ **Problem:** how to deal with key management of completely random keys?
 - ▶ Key generation, key transportation, key synchronization, key destruction are ALL problematic since the keys are SO large.

Outline

Random Numbers

DRBG

CTR_DRBG

Dual_EC_DRBG

Stream Ciphers

One Time Pad

Visual Cryptography

Prominent Stream Ciphers

A5 Cipher

RC4 Cipher

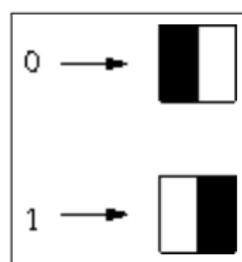
ChaCha Cipher

Conclusion

Visual Cryptography

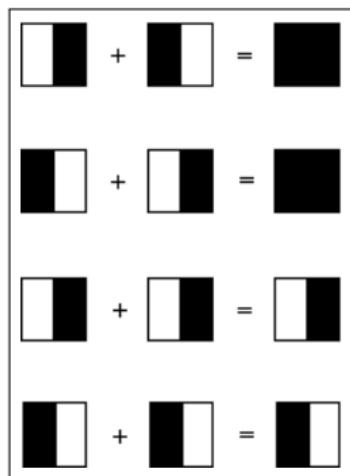
- ▶ **Application of one time pad:** visual cryptography splits an image into 2 shares.
- ▶ Decryption works by *overlaying* the 2 shared images.
- ▶ First proposed by Naor and Shamir in 1994.
- ▶ **Simple case:** monochrome images with black and white pixels.
- ▶ Many generalisations are possible.
- ▶ Each pixel is shared in a random way, similar to splitting a bit in the one time pad.
- ▶ Each share reveals NO information about the image:
 - ▶ Unconditional security as one time pad.

Encryption



- ▶ Generate a one time pad P (random bit string) with length equal to the number of pixels for the image I
- ▶ Generate a share $S_{I,1}$ by replacing each bit in P using the sub-pixel patterns shown on the left
- ▶ Generate the other share $S_{I,2}$ s.t.:
 - ▶ the same as $S_{I,1}$ for all the white pixels of I
 - ▶ the opposite of $S_{I,1}$ for all black pixels of I

Decryption



- ▶ To reveal the hidden image I , $S_{I,1}$ and $S_{I,2}$ are overlayed
- ▶ Each black pixel of I is black in the overlay
- ▶ Each white pixel of I is half white in the overlay

Outline

Random Numbers

DRBG

CTR_DRBG

Dual_EC_DRBG

Stream Ciphers

One Time Pad

Visual Cryptography

Prominent Stream Ciphers

A5 Cipher

RC4 Cipher

ChaCha Cipher

Conclusion

└ Prominent Stream Ciphers

└ A5 Cipher

A5 Cipher

- ▶ Binary synchronous stream cipher applied in most GSM mobile telephones.
- ▶ 3 variants:
 - ▶ A5/1 is the original algorithm defined in 1987.
 - ▶ A5/2 is a *weakened* version of A5/1, originally intended for deployment outside Europe, but no longer allowed under GSM standards.
 - ▶ A5/3, also known as KASUMI, is an algorithm for deployment in 3G mobile systems.
- ▶ Design was originally kept confidential by its designers but became public in 1994.

A5/1 Design

- ▶ A5/1 algorithm uses 3 linear feedback shift registers (LFSRs) whose output is combined.
- ▶ The 3 LFSRs are *irregularly clocked*:
 - ▶ The overall output is non-linear.
 - ▶ 64-bit keystream s.t. 10 bits fixed at zero.
 - ▶ The effective key length is thus 54 bits.
- ▶ Many successful attacks:
 - ▶ In 2008, Gendrullis, Novotny and Rupp reported an attack which broke A5/1 in practice in 7 hours given.

└ Prominent Stream Ciphers

└ RC4 Cipher

RC4 Cipher

- ▶ World-based stream cipher designed by Ron Rivest in the 80s: “Ron’s code #4”.
- ▶ Simple, efficient for software implementation.
- ▶ Originally proprietary owned by RSA Security, but leaked in 1994.
- ▶ Widely deployed in TLS before 2013.
- ▶ Practical attacks:
 - ▶ When used in TLS protocol and in wireless WPA-TKIP due to bias in its keystream output.
- ▶ Widely believed to be too weak to use in new systems.

└ Prominent Stream Ciphers

└ ChaCha Cipher

ChaCha Algorithm

- ▶ Available in TLS ciphersuites (RFC 7905) as a possible replacement for RC4.
- ▶ Designed by D. J. Bernstein in 2008.
- ▶ Faster than AES:
 - ▶ As little as 4 cycles per byte on x86 processors.
- ▶ Combining XOR, addition modulo 2^{32} and rotation operations over 20 rounds to produce 512 bits of keystream:
 - ▶ Example: add-rotate-xor (ARX) cipher.
- ▶ Using 256-bit key.

Outline

Random Numbers

DRBG

CTR_DRBG

Dual_EC_DRBG

Stream Ciphers

One Time Pad

Visual Cryptography

Prominent Stream Ciphers

A5 Cipher

RC4 Cipher

ChaCha Cipher

Conclusion

Conclusion

- ▶ TRNG constructed from physical devices, used as seeds for PRNG.
- ▶ PRNG constructed from other primitives including block ciphers.
- ▶ TRNG used to make unbreakable encryption via one time pad.
- ▶ PRNG used as practical synchronous stream cipher.

Lecture 10: Number Theory for Public Key Cryptography

COSC362 Data and Network Security

Book 1: Chapter 2

Spring Semester, 2021

Motivation

- ▶ Number theory problems used in public key cryptography.
- ▶ Need of efficient ways to generate large prime numbers in order to use such problems.
- ▶ Definitions of hard computational problems to base cryptosystems on.

Outline

Chinese Remainder Theorem

Euler Function

Primality Tests

Fermat Test

Miller-Rabin Test

Basic Complexity Theory

Factorisation Problem

Discrete Logarithm Problem

Outline

Chinese Remainder Theorem

Euler Function

Primality Tests

Fermat Test

Miller-Rabin Test

Basic Complexity Theory

Factorisation Problem

Discrete Logarithm Problem

Chinese Remainder Theorem (CRT)

Theorem: (this is a special case!)

- ▶ Let p, q be relatively prime.
- ▶ Let $n = p \times q$ be the modulus.
- ▶ Given integers c_1 and c_2 , there exists a unique integer x , $0 \leq x < n$, s.t.:

$$\begin{aligned} x &\equiv c_1 \pmod{p} \\ x &\equiv c_2 \pmod{q} \end{aligned}$$

- ▶ $x \equiv \frac{n}{p}y_1c_1 + \frac{n}{q}y_2c_2 \pmod{n}$ where:
 - ▶ $y_1 \equiv (\frac{n}{p})^{-1} \pmod{p} = q^{-1} \pmod{p}$
 (rewriting as $qy_1 \equiv 1 \pmod{p}$)
 - ▶ $y_2 \equiv (\frac{n}{q})^{-1} \pmod{q} = p^{-1} \pmod{q}$
 (rewriting as $py_2 \equiv 1 \pmod{q}$)

Example

Solve $x \equiv 5 \pmod{6}$ and $x \equiv 33 \pmod{35}$:

- ▶ $c_1 = 5$ and $c_2 = 33$
- ▶ $p = 6$ and $q = 35$ are relatively prime, so CRT can be used
- ▶ $n = 6 \times 35 = 210$

$$\begin{array}{c|c} \frac{210}{6}y_1 \equiv 1 \pmod{6} & \frac{210}{35}y_2 \equiv 1 \pmod{35} \\ 35y_1 \equiv 1 \pmod{6} & 6y_2 \equiv 1 \pmod{35} \\ y_1 \equiv 5 \pmod{6} & y_2 \equiv 6 \pmod{35} \end{array}$$

$$\begin{aligned} x &\equiv \frac{n}{p}y_1c_1 + \frac{n}{q}y_2c_2 \pmod{n} \\ &\equiv (35 \times 5 \times 5) + (6 \times 6 \times 33) \pmod{210} \\ &\equiv 175 \times 5 + 36 \times 33 \pmod{210} \equiv 173 \pmod{210} \end{aligned}$$

Outline

Chinese Remainder Theorem

Euler Function

Primality Tests

Fermat Test

Miller-Rabin Test

Basic Complexity Theory

Factorisation Problem

Discrete Logarithm Problem

Euler Function

Definition: given a positive integer n , the Euler function $\phi(n)$ denotes the number of *positive integers less than n and relatively prime to n* .

Example: $\phi(10) = 4$ since 1, 3, 7, 9 are each relatively prime to 10.

The set of positive integers less than $n = 10$ and relatively prime to $n = 10$ is thus $\mathbb{Z}_{10}^* = \{1, 3, 7, 9\}$.

Properties

- ▶ $\phi(p) = p - 1$ where p is prime.
- ▶ $\phi(pq) = (p - 1)(q - 1)$ where p, q are distinct primes.
- ▶ $n = p_1^{e_1} \cdots p_t^{e_t}$ where p_i are distinct primes, then:

$$\phi(n) = \prod_{i=1}^t p_i^{e_i-1} (p_i - 1)$$

Examples:

$$\begin{aligned}\phi(15) &= (3 - 1)(5 - 1) = 2 \times 4 = 8 \text{ since } 15 = 3 \times 5 \\ \phi(24) &= 2^2(2 - 1) \times 3^0(3 - 1) = 8 \text{ since } 24 = 2^3 \times 3\end{aligned}$$

Important Theorems

Fermat's theorem: Let p be a prime, then for any integer a s.t.
 $1 < a \leq p - 1$:

$$a^{p-1} \mod p = 1$$

Euler's theorem: If $\gcd(a, n) = 1$ then:

$$a^{\phi(n)} \mod n = 1$$

When p is prime then $\phi(p) = p - 1$, so Fermat's theorem is a special case of Euler's theorem.

Outline

Chinese Remainder Theorem

Euler Function

PRIMALITY TESTS

Fermat Test

Miller-Rabin Test

Basic Complexity Theory

Factorisation Problem

Discrete Logarithm Problem

Primality Tests

- ▶ Testing for primality by trial division is not practical (except for very small numbers).
- ▶ Many *probabilistic* methods:
 - ▶ Requiring random inputs
 - ▶ Possible failure in exceptional circumstances
- ▶ Indian mathematicians Agrawal, Saxena and Kayal found a polynomial time deterministic primality test (2002):
 - ▶ Huge theoretical breakthrough
- ▶ Probabilistic methods are still used in practice.

Fermat Primality Test

- ▶ *Fermat's theorem:* If p is prime then $a^{p-1} \bmod p = 1$ for all a s.t. $\gcd(a, p) = 1$.
- ▶ If a number n s.t. $a^{n-1} \bmod n \neq 1$, then n is NOT prime.
- ▶ *Test idea:* If a number satisfies Fermat's equation then we ASSUME that it is prime.
- ▶ Failure is possible but very unlikely in practice.
- ▶ Reducing the failure probability by repeating the test with different base values a .

└ Primality Tests

└ Fermat Test

Fermat Primality Test

- ▶ *Inputs:*
 - ▶ a value n to test for primality
 - ▶ a parameter k that determines the number of times the test is run
- ▶ *Output:* composite if n is composite; probable prime otherwise.
- ▶ *Algorithm:*
 - ▶ Pick a at random s.t. $1 < a < n - 1$.
 - ▶ If $a^{n-1} \bmod n \neq 1$ then return composite; otherwise return probable prime.

└ PrIMALITY TESTS

└ Fermat Test

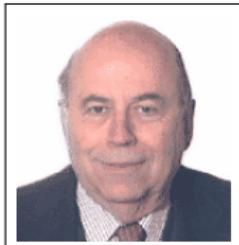
Effectiveness

- ▶ If the test outputs `composite` then n is definitely composite.
- ▶ The test can output `probable prime` if n is composite:
 - ▶ n is called *pseudoprime*
- ▶ The test ALWAYS outputs `probable prime` for some composite n :
 - ▶ n is called *Carmichael number*
 - ▶ First few Carmichael numbers: 561, 1105, 1729, 2465, etc.

└ Primality Tests

└ Miller-Rabin Test

Miller-Rabin Test



- ▶ Similar idea to Fermat test
- ▶ Guaranteeing to detect composite numbers if test run sufficiently many times
- ▶ Most widely used test for generating large prime numbers

└ PrIMALITY TESTS

└ Miller-Rabin Test

Square Roots of 1

- ▶ A *modular square root of 1* is a number x s.t. $x^2 \pmod{n} = 1$.
- ▶ There are 4 square roots of 1 when $n = pq$:
 - ▶ 2 are 1 and $-1 \pmod{n}$
 - ▶ 2 are called *non-trivial* square roots of 1
- ▶ If x is a non-trivial square root of 1 then $\gcd(x - 1, n)$ is a non-trivial factor of n :
 - ▶ Hence, if a non-trivial square root of 1 exists then n is composite

└ PrIMALITY TESTS

└ Miller-Rabin Test

Miller-Rabin Algorithm

Let n and u be odd, and v s.t. $n - 1 = 2^v u$:

1. Pick a at random s.t. $1 < a < n - 1$
2. Set $b = a^u \pmod{n}$
3. If $b = 1$ then return probable prime
4. For $j = 0$ to $v - 1$:
 - ▶ If $b = -1$ then return probable prime
 - ▶ Else set $b = b^2 \pmod{n}$
5. Return composite

Note: when an output is returned, the algorithm halts.

└ PrIMALITY TESTS

└ Miller-Rabin Test

Effectiveness

- ▶ If test returns composite then n is composite.
- ▶ If test returns probable prime then n MAY be composite.
- ▶ If n is composite then test returns probable prime with probability at most $1/4$:
 - ▶ Algorithm is run k times
 - ▶ Repeat while the output is probable prime
 - ▶ Output probable prime when n is composite with probability no more than $(1/4)^k$
- ▶ In practice, error probability is smaller:
 - ▶ *Example:* no composites less than 341,550,071,728,321 which pass the test for 7 base values
 $a = 2, 3, 5, 7, 11, 13, 17.$

Why Does Miller-Rabin Test Work?

- ▶ Random a s.t. $0 < a < n - 1$, and $n - 1 = 2^v u$.
- ▶ Sequence $a^u, a^{2u}, \dots, a^{2^{v-1}u}, a^{2^v u} \pmod{n}$.
- ▶ Each number in the sequence (after the 1st) is the square of the previous number.
- ▶ If n is prime then $a^{2^v u} \pmod{n} = 1$ (Fermat's theorem), and then:
 - ▶ Either $a^u \pmod{n} = 1$
 - ▶ Or there is a square root of 1 somewhere in the sequence, and the value must be -1
- ▶ If a non-trivial square root of 1 is found then n is composite.

└ Primality Tests

└ Miller-Rabin Test

Example

- ▶ Let $n = 1729$ be a Charmichael number.
 - ▶ $n - 1 = 1728 = 64 \times 27 = 2^6 \times 27$.
 - ▶ Hence $v = 6$ and $u = 27$.
1. Choose $a = 2$
 2. $b = 2^{27} \pmod{1729} = 645$
 3. Since $b \neq 1$, continue:
 - ▶ $b = 645^2 \pmod{n} = 1065$
 - ▶ $b = 1065^2 \pmod{n} = 1$
 - ▶ Thus $b = -1$ will never occur
 4. Return composite

Note: 1065 is a non-trivial square root of 1 modulo 1729, since $\gcd(1729, 1064) = 133$ is a factor of 1729.

Generation of Large Primes

Miller-Rabin test used to generate large primes:

1. Choose a random odd integer r of the same number of bits as the required prime.
2. Test if r is divisible by any of a list of small primes.
3. Apply Miller-Rabin test with 5 random (or fixed) base values a .
4. If r fails any test, then set $r = r + 2$ and return to Step 2.

Note: this *incremental* method does not produce completely random primes.

Instead, start from Step 1 if r is found to be composite. Both options are seen in practice.

Outline

Chinese Remainder Theorem

Euler Function

Primality Tests

Fermat Test

Miller-Rabin Test

Basic Complexity Theory

Factorisation Problem

Discrete Logarithm Problem

Complexity Theory in Cryptology

- ▶ Computational complexity provides a foundation for:
 - ▶ Analysing computational requirements of cryptanalytic techniques.
 - ▶ Studying the difficulty of breaking ciphers.
- ▶ 2 aspects:
 - ▶ *Algorithm complexity*: how long does it take to run a particular algorithm?
 - ▶ *Problem complexity*: what is the best (known) algorithm to solve a particular problem?

Algorithm Complexity

- ▶ Computational complexity of an algorithm is measured by its time and space requirements, measured as functions of the size of the input m .
- ▶ “big O” notation:
 - ▶ $f(m), g(m)$ are 2 positive functions
 - ▶ $f(m)$ is expressed as an “order of magnitude” of the form $O(g(m))$
 - ▶ $f(m) = O(g(m))$ if there are constants $c > 0$ and m_0 s.t.
 $f(m) \leq c \cdot g(m)$ for $m \geq m_0$

Polynomial and Exponential Functions

- ▶ **Polynomial time function:** function $f(m) = O(m^t)$ for some positive integer t :
 - ▶ Polynomial time function is seen as *efficient* in cryptography.
- ▶ **Exponential time function:** function $f(m) = O(b^m)$ for some number $b > 1$:
 - ▶ A problem whose the best solution is an exponential time function is seen as *hard* in cryptography.
- ▶ Brute force key search is *exponential* as a function of the key length:
 - ▶ An m -bit key length allows 2^m keys.

Examples of Algorithm Complexity

1. Let $f(m) = 17m + 10$, then $f(m) = O(m)$:
 - ▶ $17m + 10 \leq 18m$ for $m \geq 10$
2. Let $f(m) = a_0 + a_1m + \dots + a_tm^t$ be a polynomial, then $f(m) = O(m^t)$

Problem Complexity

- ▶ A problem is classified according to the minimum time and space needed to solve its hardest instances on a deterministic computer.
- ▶ *Examples:* polynomial time problems
 - ▶ Multiplication of 2 $m \times m$ matrices, with fixed size entries, using the obvious algorithm is $O(m^3)$.
 - ▶ Sorting a set of integers into ascending order is $O(m \cdot \log_2 m)$ with algorithms such as Quicksort.

Hard Problems

- ▶ *Integer factorisation:* given an integer of m bits, find its prime factors.
- ▶ *Discrete logarithm problem (with base 2):* given a prime p of m bits and an integer y s.t. $0 < y < p$, find x s.t. $y = 2^x \pmod{p}$.
- ▶ There are no known polynomial algorithms to solve these problems.
- ▶ The best known algorithms are *sub-exponential*:
 - ▶ Slower than any polynomial algorithm but faster than any exponential algorithm.

Outline

Chinese Remainder Theorem

Euler Function

Primality Tests

Fermat Test

Miller-Rabin Test

Basic Complexity Theory

Factorisation Problem

Discrete Logarithm Problem

Integer Factorisation:

- ▶ Factorisation by trial division is an exponential time algorithm:
 - ▶ Hopeless for numbers of few hundred bits.
- ▶ Several methods exist:
 - ▶ They apply if the integer to be factorised has SPECIAL properties.
- ▶ The best current general method is *number field sieve*:
 - ▶ Sub-exponential algorithm.

Factorisation Records

Decimal digits	Bits	Date	CPU years
140	467	Feb. 1999	?
155	512	Aug. 1999	?
160	533	Mar. 2003	2.7
174	576	Dec. 2003	13.2
200	667	May. 2005	121
232	768	Dec. 2009	3300

- ▶ All records used number field sieve method.
- ▶ Assuming 1 GHz CPU.

http://en.wikipedia.org/wiki/RSA_numbers

Comparing Brute Force Search and Factorisation

Symmetric key length	Length of $n = pq$
80	1024
112	2048
128	3072
192	7680
256	15360

- ▶ *Example:* Brute force key search of 128-bit keys for AES takes roughly the same computational effort as factorisation of a 3072-bit number with 2 factors of roughly equal size.
- ▶ NIST SP 800-57 Part 1: Recommendations for Key Management (2016).

Outline

Chinese Remainder Theorem

Euler Function

Primality Tests

Fermat Test

Miller-Rabin Test

Basic Complexity Theory

Factorisation Problem

Discrete Logarithm Problem

Discrete Logarithm Problem

- ▶ g is a generator of \mathbb{Z}_p^* for a prime p .
- ▶ Discrete logarithm problem over \mathbb{Z}_p^* is:
 - ▶ Given $y \in \mathbb{Z}_p^*$, find x s.t. $y = g^x \pmod p$.
- ▶ If p is large enough, then the problem is believed to be hard.
- ▶ The best known algorithm is a variant of the number field sieve.
- ▶ Length of modulus p should be chosen of same length as RSA modulus for the same security level (at least 2048 bits).

Example

$g^x \bmod p$	x	$g^x \bmod p$	x
1	18	10	17
2	1	11	12
3	13	12	15
4	2	13	5
5	16	14	7
6	14	15	11
7	6	16	4
8	3	17	10
9	8	18	9

- Set of non-zero integers modulo 19 is \mathbb{Z}_{19}^*
- Generator is $g = 2$
- When $y = g^x \bmod p$ then $\log_g(y) = x$
- Example:
 $\log_2(3) = 13$

Comparing Brute Force Search, Factorisation and DL

Symmetric key length	Length of $n = pq$	Length of discrete log modulus p
80	1024	1024
112	2048	2048
128	3072	3072
192	7680	7680
256	15360	15360

- ▶ *Example:* brute force key search of 128-bit keys for AES takes roughly the same computational effort as factorisation of a 3072-bit number with 2 factors of roughly equal size, or finding discrete logs with a 3072-bit modulus.
- ▶ NIST SP 800-57 Part 1 (2016).

Lecture 11: Hash Functions and MACs

COSC362 Data and Network Security

Book 1: Chapters 11 and 12 – Book 2: Chapters 2 and 21

Spring Semester, 2021

Motivation

- ▶ Examples of message authentication codes (MACs) built from block ciphers (previously seen).
- ▶ However, these MACs are not commonly used in TLS.
- ▶ Another MAC, called HMAC, is widely used in TLS.
- ▶ Authenticated encryption mode GCM is also widely used in TLS (previously mentioned).
- ▶ Hash functions are typical building blocks in cryptography for MACs and digital signatures.

Outline

Hash Functions

- Security Properties

- Iterated Hash Functions

- Standardized Hash Functions

- Using Hash Functions

Message Authentication Code (MAC)

- MAC from Hash Function (HMAC)

Authenticated Encryption

- Combining Encryption and MAC

- Galois Counter Mode (GCM)

Outline

Hash Functions

 Security Properties

 Iterated Hash Functions

 Standardized Hash Functions

 Using Hash Functions

Message Authentication Code (MAC)

 MAC from Hash Function (HMAC)

Authenticated Encryption

 Combining Encryption and MAC

 Galois Counter Mode (GCM)

Hash Functions

A *hash function* H is a PUBLIC function s.t.:

- ▶ H is simple and fast to compute
- ▶ H takes as input a message m of ARBITRARY length and outputs a message *digest* $H(m)$ of FIXED length

Security Properties

- ▶ *Collision resistant:*
 - ▶ It should be infeasible to find any 2 different values x_1, x_2 s.t. $H(x_1) = H(x_2)$.
- ▶ *Second-preimage resistant:*
 - ▶ Given a value x_1 , it should be infeasible to find a different value x_2 s.t. $H(x_1) = H(x_2)$.
- ▶ *Preimage resistant (one-way):*
 - ▶ Given a value y , it should be infeasible to find any input x such that $H(x) = y$.

An attacker who can break second-preimage resistance can break collision resistance.

Birthday Paradox

- ▶ Let a group of 23 randomly chosen people:
 - ▶ The probability that at least 2 have the same birthday is over 0.5.
- ▶ If choosing around $\sqrt{|S|}$ values from a set S , then the probability of getting 2 values the same is around 0.5.
- ▶ Let H be a hash function with output size of k bits:
 - ▶ Let H be seen as a random function.
 - ▶ Then $\sqrt{2^k} = 2^{k/2}$ trials are enough to find a collision with probability around 0.5.
- ▶ Today, 2^{128} trials would be considered infeasible:
 - ▶ Hash functions should have output of at least 256 bit to satisfy collision resistance.

Example with $|S| = 100$

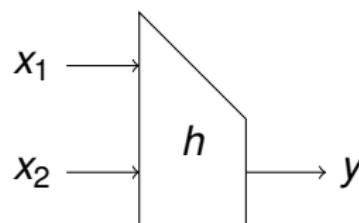
# trials	Collision prob.	# trials	Collision prob.
1	0	13	.55727
2	.01000	14	.61483
3	.02980	15	.66876
4	.05891	16	.71845
5	.09656	17	.76350
6	.14174	18	.80371
7	.19324	19	.83905
8	.24972	20	.86964
9	.30975	21	.89572
10	.37188	22	.91762
11	.43470	23	.93575
12	.49689	24	.95053

Iterated Hash Functions

- ▶ Cryptographic hash functions need to:
 - ▶ take arbitrary-sized inputs
 - ▶ produce a fixed-sized output
- ▶ From block ciphers, arbitrary-sized data can be processed by:
 - ▶ having a function processing fixed-sized data
 - ▶ using it repeatedly
- ▶ An *iterated hash function* splits the input blocks of fixed size and operates on each block sequentially using the same function with fixed-sized inputs.
- ▶ *Merkle-Damgård*: using a *compression function* h taking fixed-sized inputs and applied to multiple blocks of the message.

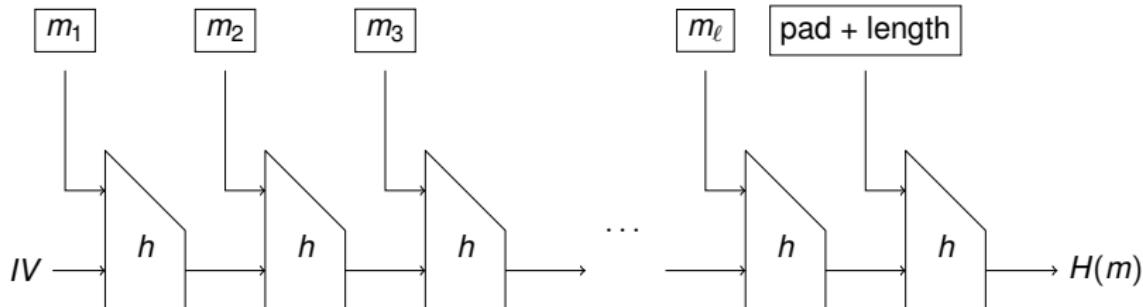
Compression Function h

h takes 2 n -bit input strings x_1 and x_2 and produces an n -bit output string y :



Merkle-Damgård Construction

1. Break message m into n -bit blocks $m_1 \parallel m_2 \parallel \dots \parallel m_\ell$.
2. Add padding and an encoding of the length of m :
 - ▶ This process may, or may not, add one block.
3. Input each block into compression function h along with chained output:
 - ▶ Use IV to get started.



Using Merkle-Damgård Construction

- ▶ **Security:** if compression function h is collision-resistant then hash function H is collision-resistant.
- ▶ But security weaknesses:
 - ▶ *Length extension attacks*: once there is one collision, easy to find more.
 - ▶ *Second-preimage attacks*: not as hard as they should be.
 - ▶ *Collisions for multiple messages*: found without much more difficulty than collisions for 2 messages.
- ▶ **Examples:** MD5, SHA-1, SHA-2 family.

MDx Family

- ▶ Proposed by Ron Rivest and widely used in the 90s.
- ▶ Deployed family members: MD2, MD4 and MD5.
- ▶ 128-bit output.
- ▶ All are broken:
 - ▶ Real collisions have been found.
 - ▶ MD5 collisions can be found in 1 minute on a PC (2006).

Secure Hash Algorithm (SHA)

- ▶ Based on MDx family design:
 - ▶ More complex design.
 - ▶ Larger output of 160 bits.
- ▶ Published by US standard agency NIST (previously called NBS) in 1993:
 - ▶ Called SHA-0.
 - ▶ Replaced by SHA-1 with very small changes to algorithm (1995).
- ▶ Both have been broken:
 - ▶ SHA-0: collisions found in 2004.
 - ▶ SHA-1: collisions found in 2017 s.t. attack 100,000 faster than brute force search.

SHA-2 Family

- ▶ Developed in response to (real and theoretical) attacks on MD5 and SHA-1.
- ▶ **Standard:** FIPS PUB 180-4 (Aug. 2015).
- ▶ Known as SHA-2.

Name	Hash size	Block size	Security match
SHA-224	224 bits	512 bits	2 key 3DES
SHA-512/224	224 bits	1024 bits	2 key 3DES
SHA-256	256 bits	512 bits	AES-128
SHA-512/256	256 bits	1024 bits	AES-128
SHA-384	384 bits	1024 bits	AES-192
SHA-512	512 bits	1024 bits	AES-256

Padding in SHA-2 Family

- ▶ *Message length field:*
 - ▶ 64 bits when block length is 512 bits.
 - ▶ 128 bits when block length is 1024 bits.
- ▶ Always at least one bit of padding.
- ▶ There is an exact number of complete blocks:
 - ▶ After the 1st bit “1”, enough bits “0” are added.
 - ▶ Length field is then added.
- ▶ Adding the padding and length field sometimes add an extra block, and sometimes does not.

SHA-3

- ▶ Crisis in hash function design late 2000s:
 - ▶ MDx and SHA families all based on same basic design.
 - ▶ Unexpected attacks against them in recent years.
- ▶ NIST announced a competition for new hash standard SHA-3 (Nov. 2007):
 - ▶ Entries closed in Oct. 2008 with 64 original submissions.
 - ▶ 14 went through Round 2.
 - ▶ 5 finalists announced in Dec. 2010.
 - ▶ Keccak selected as winner in Oct. 2012.
- ▶ Keccak does not use compression function:
 - ▶ Instead, a *sponge function*.
- ▶ **Standard:** FIPS PUB 202 (Aug. 2015).

Using Hash Functions

- ▶ Applying a hash function is NOT encryption:
 - ▶ Hash computation does NOT depend on a key.
 - ▶ Not possible to go backwards to find the input in general.
- ▶ Helping to provide data authentication:
 - ▶ But not providing it alone!
 - ▶ Authenticating the hash of a message to authenticate the message.
 - ▶ Building block for MACs.
 - ▶ Building block for signatures.

Storing Passwords for Login

- ▶ Storing user passwords on servers using hash functions.
- ▶ Storing salted hashes of passwords:
 1. Pick at random *salt*
 2. Compute $h = H(pw, salt)$
 3. Store $(salt, h)$
- ▶ Easy to check entered password pw' : $h = H(pw', salt)$?
- ▶ Hard to recover pw from h assuming that H is preimage resistant.
- ▶ The attacker needs to store a different dictionary for EACH *salt*.
- ▶ Using a *slower* hash function slows down password guessing.

Outline

Hash Functions

 Security Properties

 Iterated Hash Functions

 Standardized Hash Functions

 Using Hash Functions

Message Authentication Code (MAC)

 MAC from Hash Function (HMAC)

Authenticated Encryption

 Combining Encryption and MAC

 Galois Counter Mode (GCM)

Message Authentication Code (MAC)

- ▶ Message authentication code (MAC) is a cryptographic mechanism to ensure message integrity:
 - ▶ **Inputs:** message M of arbitrary length, secret key K
 - ▶ **Output:** (short) fixed-sized tag $T = \text{MAC}(M, K)$
- ▶ Alice, the sender, appends the tag T to the message M (in the clear).
- ▶ Bob, the recipient, computes $T' = \text{MAC}(M', K)$ with the received message M' , and checks whether $T = T'$.

Properties

- ▶ ***Unforgeability:*** it is not feasible to produce a valid pair (M, T) s.t. $T = \text{MAC}(M, K)$ without knowledge of K .
- ▶ ***Unforgeability under chosen message attack:***
 - ▶ The attacker has access to a *forging oracle* s.t. on input any message M of the attacker's choice, the oracle outputs the tag $T = \text{MAC}(M, K)$.
 - ▶ The attacker should not be able to produce a valid forgery that was not asked to the oracle.

MAC from Hash Function (HMAC)

- ▶ Proposed by Bellare, Canetti and Krawczyk in 1996.
- ▶ Built from ANY iterated hash function H :
 - ▶ *Examples:* MD5, SHA-1, SHA-256, ...
- ▶ **Standard:** FIPS PUB 198-1 (July 2008).
- ▶ Used in many applications such as TLS and IPsec.

Construction

- ▶ H : iterated cryptographic hash function
- ▶ M : message to be authenticated
- ▶ K : key padded with zeros to be of block size of H
- ▶ opad: fixed string 0x5c5c5c...5c
- ▶ ipad: fixed string 0x363636...36
- ▶ \parallel : concatenation of bit strings

$$\text{HMAC}(M, K) = H((K \oplus \text{opad}) \parallel H((K \oplus \text{ipad}) \parallel M))$$

Security

- ▶ HMAC is secure if:
 - ▶ either H is collision resistant
 - ▶ or H is a pseudorandom function
- ▶ Designed to resist length extension attacks:
 - ▶ Even if H is a Merkle-Damgård hash function.
- ▶ Often used as a *pseudorandom function* for deriving keys in cryptographic protocols.

Outline

Hash Functions

 Security Properties

 Iterated Hash Functions

 Standardized Hash Functions

 Using Hash Functions

Message Authentication Code (MAC)

 MAC from Hash Function (HMAC)

Authenticated Encryption

 Combining Encryption and MAC

 Galois Counter Mode (GCM)

Authenticated Encryption

- ▶ Let Alice and Bob share a key K .
- ▶ Alice wants to send to Bob a message M with *confidentiality* and *authenticity/integrity*.
- ▶ 2 options:
 - ▶ Split K into 2 parts K_1, K_2 , encrypt with K_1 (confidentiality) and use K_2 with a MAC (authenticity/integrity).
 - ▶ Use the *authenticated encryption* algorithm providing both confidentiality and authenticity/integrity.

Combining Encryption and MAC

3 options:

- ▶ Encrypt and MAC: encrypt M , apply MAC to M , and send the ciphertext C and the tag T .
- ▶ MAC then encrypt: apply MAC to M , then encrypt $M||T$, and send the ciphertext C .
- ▶ Encrypt then MAC: encrypt M , apply MAC to the ciphertext C , and send C and the tag T .

Encrypt-then-MAC option is the safest approach:

1. $C = \text{Enc}(M, K_1)$
2. $T = \text{MAC}(C, K_2)$
3. Send $C||T$

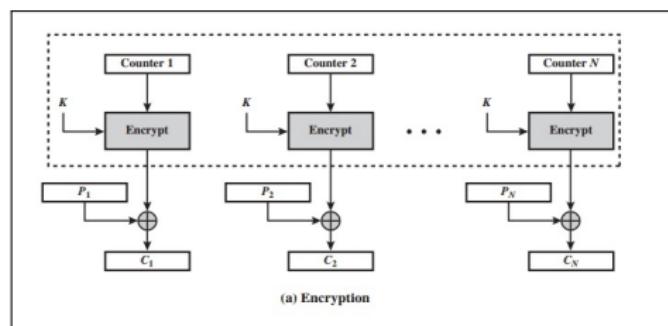
Authenticated Encryption Mode

- ▶ MAC-then-encrypt construction used in older versions of TLS (SSL 3, TLS 1.0 and 1.1):
 - ▶ Several security vulnerabilities.
- ▶ Combined authentication encryption modes in recent versions (TLS 1.2 and 1.3):
 - ▶ Supporting CCM and GCM modes of operation for block ciphers.
 - ▶ Allowing data to be only authenticated (not encrypted) with *authenticated encryption with associated data* (AEAD).

CTR Mode for Block Ciphers

CTR is a synchronous stream cipher:

- ▶ A counter is initialised using a randomly chosen nonce N .
- ▶ Keystream generated by encrypting successive values of the counter:
 - ▶ $O_t = E(T_t, K)$ where $T_t = N||t$ is the concatenation of N and block number t .



$$\text{Encryption: } C_t = O_t \oplus P_t$$

$$\text{Decryption: } P_t = O_t \oplus C_t$$

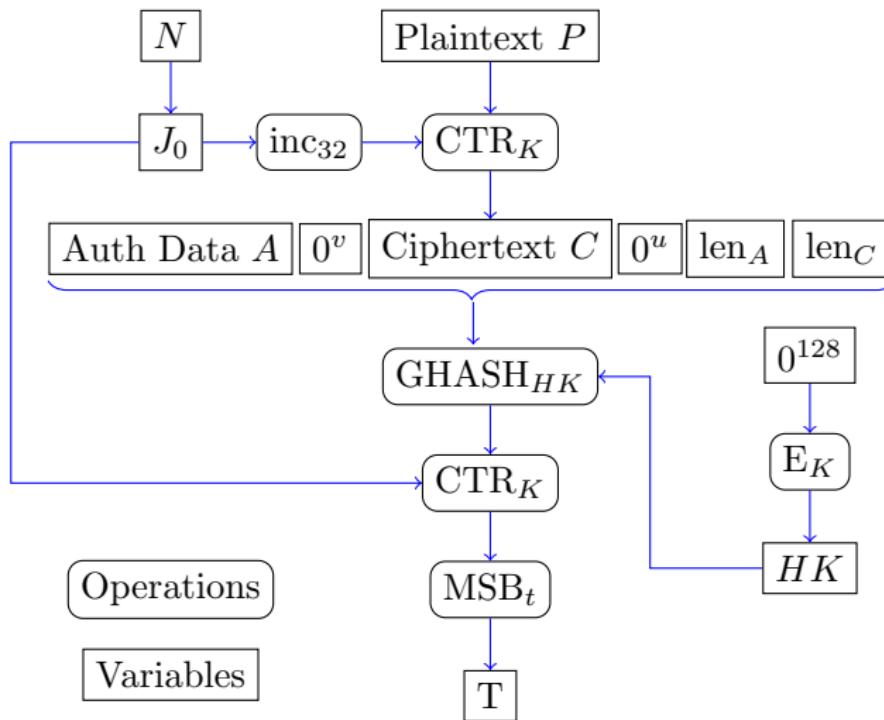
Galois Counter Mode (GCM)

- ▶ CCM mode (Lecture 8) is NOT suitable for processing of streaming data:
 - ▶ Formatting function for N, A, P requires knowledge of length of A and P .
- ▶ Galois counter mode (GCM) overcomes such limitation.
- ▶ **Standard:** NIST SP-800 38D.
- ▶ AES with GCM faster than AES with HMAC:
 - ▶ Hardware support of AES and carry-less addition in modern Intel chips.

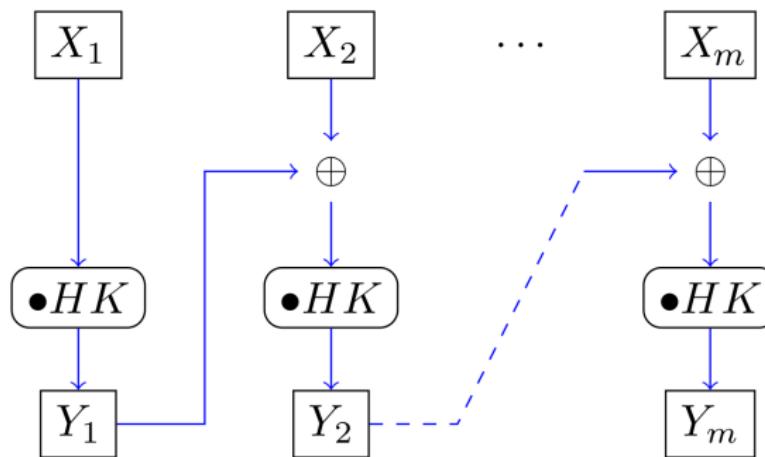
Algorithm

- ▶ Combining CTR mode on block cipher E (e.g. AES) with a special keyed hash function GHASH:
 - ▶ GHASH uses multiplication in finite field $GF(2^{128})$.
- ▶ **Inputs:** plaintext P , authenticated data A , nonce N .
- ▶ **Outputs:** ciphertext C and tag T .
- ▶ Length len_A of A and length len_C of C are 64-bit values:
 - ▶ u and v are minimum numbers of zeros required to expand A and C to complete blocks, respectively.
- ▶ Length t of T is 128 bits and length of N is 96 bits.
- ▶ Initial block input is $J_0 = N||0^{31}||1$.
- ▶ Function inc_{32} increments the 32 MSB of input string by 1 modulo 2^{32} .

Algorithm



GHASH



- ▶ Output is $Y_m = \text{GHASH}_{HK}(X_1, \dots, X_m)$
- ▶ Operation \bullet is multiplication in the finite field $GF(2^{128})$
- ▶ $HK = E(0^{128}, K)$ is the hash subkey.

Decryption

- ▶ Elements transmitted to Bob, the recipient:
 - ▶ ciphertext C , nonce N , tag T , authenticated data A
- ▶ Bob computes the tag T' using the shared key K and received C, N, A .
- ▶ Bob compares T' with received T :
 - ▶ If $T' \neq T$ then output “invalid”.
 - ▶ If $T' = T$ then the plaintext P is computed by generating the same keystream from CTR mode as for encryption.

Lecture 12: Public Key Cryptography Part 1

COSC362 Data and Network Security

Book 1: Chapters 9 and 10 – Book 2: Chapters 2 and 21

Spring Semester, 2021

Motivation

- ▶ Public key cryptography (PKC) has features that symmetric key cryptography does not have.
- ▶ Applied for key management in protocols such as TLS and IPsec.
- ▶ RSA is one of the best known public key cryptosystems, widely deployed in practice.
- ▶ Alternatives include discrete log based ciphers, also widely deployed and standardised.

Outline

Public Key Cryptography

RSA Algorithms

RSA Implementation

RSA Security

Outline

Public Key Cryptography

RSA Algorithms

RSA Implementation

RSA Security

One-Way Functions

- ▶ A function f is *one-way* if $f(x) = y$ is easily computed given x , but $f^{-1}(y) = x$ is (computationally) hard to compute given y .
- ▶ **Open problem:** Do one-way functions actually exist?
- ▶ **Examples of functions believed to be one-way:**
 - ▶ Multiplication of large primes: the inverse function is integer factorisation.
 - ▶ Exponentiation: the inverse function takes discrete logarithms.

Trapdoor One-Way Functions

- ▶ A *trapdoor one-way* function f is a one-way function s.t. $f^{-1}(y)$ is easily computed given additional information, called *trapdoor*.
- ▶ Example:
 - ▶ Modular squaring: given $n = pq$ where p, q are 2 large primes, $f(x) = x^2 \bmod n$.
 - ▶ If an algorithm takes square roots (i.e. computes f^{-1}) then it can be used to factorise n .
 - ▶ The trapdoor is the factorisation of n .
 - ▶ If the trapdoor is known then an efficient algorithm finds square roots.

Ciphers Based on Computationally Hard Problems



- ▶ Diffie and Hellman published *New Directions in Cryptography* (1976).
- ▶ Computational complexity applied in design of encryption algorithms.
- ▶ A public key cryptosystem designed by using a trapdoor one-way function.
- ▶ Trapdoor is the decryption key.

Also Known as Asymmetric Cryptography

- ▶ **Asymmetry:** encryption and decryption keys are different.
- ▶ Encryption key is a *public* key, known to anybody.
- ▶ Decryption key is a *private* key, known ONLY to its owner.
- ▶ Finding the private key from the knowledge of the public key MUST be a hard computational problem.

Why Public Key Cryptography?

Advantages (in comparison to shared key/symmetric cryptography):

- ▶ Key management is simplified:
 - ▶ keys do not need to be transported confidentially
- ▶ Digital signatures can be obtained.

In Practice

- ▶ In a public cipher, encryption keys can be made public.
- ▶ Alice stores her public key in a public directory:
 - ▶ Anyone can obtain her public key and use it to form an encrypted message to Alice.
 - ▶ Since Alice has the private key (associated with her public key), she can decrypt and recover the message.

Outline

Public Key Cryptography

RSA Algorithms

RSA Implementation

RSA Security

Introduction



- ▶ Rivest, Shamir and Adleman from MIT in 1977.
- ▶ Public key cryptosystem and digital signature scheme.
- ▶ Based on integer factorisation problem.
- ▶ RSA patent expired in 2000.

Key Generation

Key Generation:

- ▶ Randomly choose 2 distinct primes p, q from the set of all primes of a certain size.
- ▶ Compute $n = pq$.
- ▶ Randomly choose e s.t. $\gcd(e, \phi(n)) = 1$:
 - ▶ ϕ is the Euler function.
 - ▶ Here, $\phi(n) = \phi(pq) = (p - 1)(q - 1)$.
- ▶ Compute $d = e^{-1} \pmod{\phi(n)}$.
- ▶ Set the public key K_E as (n, e) .
- ▶ Set the private key K_D as (p, q, d) .

Encryption and Decryption

Encryption:

- ▶ Public encryption key is $K_E = (n, e)$.
- ▶ Input is a value M s.t. $0 < M < n$.
- ▶ Compute $C = Enc(M, K_E) = M^e \bmod n$.

Decryption:

- ▶ Private decryption key is $K_D = (p, q, d)$:
 - ▶ Note that p, q are not used here.
- ▶ Compute $Dec(C, K_D) = C^d \bmod n = M$.

Any message requires to be pre-processed to become M :

- ▶ Coding it as a number
- ▶ Adding randomness

Numerical Example

Key generation:

- ▶ Let $p = 43$ and $q = 59$:
 - ▶ $n = pq = 2537$
 - ▶ $\phi(n) = (p - 1)(q - 1) = 2436$
- ▶ Let $e = 5$:
 - ▶ $d = e^{-1} \bmod \phi(n) = 5^{-1} \bmod 2436 = 1949$
 - ▶ Solving $ed + k'\phi(n) = 1$ using the Euclidean algorithm
(unknowns are d and the integer k')

Encryption:

- ▶ $M = 50$, thus $C = M^e \bmod n = 50^5 \bmod 2537 = 2488$.

Decryption:

- ▶ $C^d \bmod n = 2488^{1949} \bmod 2537 = 50 = M$.

Encryption Correctness

Does encryption followed by decryption get back where we started from?

$$(M^e)^d \mod n = M ?$$

- ▶ $d = e^{-1} \mod \phi(n)$, thus $ed \mod \phi(n) = 1$:
 - ▶ there is some integer k s.t. $ed = 1 + k\phi(n)$
- ▶ $(M^e)^d \mod n = M^{ed} \mod n = M^{1+k\phi(n)} \mod n.$

To complete the proof, we need to show:

$$M^{1+k\phi(n)} \mod n = M \quad (1)$$

Proving Equation (1)

Case 1: assuming $\gcd(M, n) = 1$.

Applying Euler's theorem directly to get:

► $M^{\phi(n)} \bmod n = 1$

$$\begin{aligned} M^{1+k\phi(n)} \bmod n &= M \times (M^{\phi(n)})^k \bmod n \\ &= M \times (1)^k \bmod n \\ &= M \end{aligned}$$

Proving Equation (1)

Case 2: assuming $\gcd(M, n) \neq 1$.

Remember that $n = pq$ where p, q are primes, and $M < n$:

- ▶ Thus either $\gcd(M, p) = 1$ or $\gcd(M, q) = 1$.

Supposing $\gcd(M, p) = 1$ (and the other case is similar):

- ▶ $\gcd(M, q) = q$, thus there exists some integer l s.t. $M = lq$

Applying Fermat's theorem to get:

- ▶ $M^{\phi(p)} \bmod p = M^{p-1} \bmod p = 1$

$$\begin{aligned}
 M^{1+k\phi(n)} \bmod p &= M \times (M^{\phi(n)})^k \bmod p \\
 &= M \times (M^{p-1})^{(q-1)k} \bmod p \\
 &= M \times (1)^{(q-1)k} \bmod p \\
 &= M \bmod p \quad (2)
 \end{aligned}$$

Proving Equation (1)

Case 2 (continued):

Since $M = lq$, it follows that $M^{1+k\phi(n)} \bmod q = 0$ (3).

Applying the Chinese Remainder Theorem (CRT):

- ▶ It is possible since $n = pq$ for p, q primes.
- ▶ There is a unique solution $x = M^{1+k\phi(n)} \bmod n$ to equations (2) and (3).
- ▶ The solution $x = M$ satisfies (2) and (3), and it is the unique solution for $M^{1+k\phi(n)} \bmod n$:
 - ▶ $M = M^{1+k\phi(n)} \bmod p$
 - ▶ $M = M^{1+k\phi(n)} \bmod q (= 0)$
- ▶ Equation (1) is satisfied too.

Applications

- ▶ Message encryption
- ▶ Digital signature
- ▶ Distribution of a shared key for symmetric key encryption (hybrid encryption)
- ▶ User authentication by proving knowledge of the private key corresponding to an authenticated public key

Outline

Public Key Cryptography

RSA Algorithms

RSA Implementation

RSA Security

Evolution

Optimisations in RSA implementation have been widely studied:

- ▶ **Key generation:**
 - ▶ Generating large primes p, q
 - ▶ Choice of e
- ▶ **Encryption and decryption:**
 - ▶ Fast exponentiation
 - ▶ Faster decryption using CRT
- ▶ **Data formatting:**
 - ▶ Padding

Generating Large Primes

- ▶ Primes p, q should be random of a chosen length:
 - ▶ Today, the recommended one is at least 1024 bits.
- ▶ Simple algorithm:
 1. Select a random odd number r of the required length.
 2. Check whether r is prime:
 - ▶ If so, then output r and halt.
 - ▶ Otherwise, increment r by 2 and go to Step 2.
- ▶ Fast way to check for primality (e.g. Miller-Rabin test).

Choice of e

- ▶ Public exponent e should be chosen at random for best security.
- ▶ A small value is often used in practice:
 - ▶ It has a large effect on efficiency.
 - ▶ $e = 3$ is the smallest possible value and sometimes used (but security problems!).
 - ▶ $e = 2^{16} + 1$ is a popular choice.
- ▶ A smaller than average value for private exponent d is also possible:
 - ▶ But at least \sqrt{n} to avoid known attacks.

Fast Exponentiation

- ▶ Using *square-and-multiply* modular exponentiation algorithm for encryption and decryption.
- ▶ e in binary representation:
 - ▶ $e = e_0 2^0 + e_1 2^1 + \dots + e_k 2^k$, where e_i are bits
- ▶ Let M be the message to encrypt:
 - ▶ $M^e = M^{e_0} \times (M^2)^{e_1} \times \dots \times (M^{2^k})^{e_k}$

Square-and-multiply Algorithm

Data: $M, n, e = e_k \dots e_1 e_0$

Result: $M^e \bmod n$

$z \leftarrow 1;$

for $i = 0$ to k **do**

if $e_i = 1$ **then**

$| z \leftarrow z * M \bmod n;$

end

if $i < k$ **then**

$| M \leftarrow M^2 \bmod n;$

end

end

return z

Cost

- ▶ If $2^k \leq e < 2^{k+1}$, then the algorithm uses k squarings:
 - ▶ If b of e_i bits are '1', then the algorithm uses $b - 1$ multiplications.
 - ▶ 1st computation $z \leftarrow z * M$ is not counted because $z = 1$.
- ▶ n is a 2048-bit modulus and so e is of at most 2048 bits.
- ▶ Computing $M^e \bmod n$ requires at most:
 - ▶ 2048 modular squarings
 - ▶ 2048 modular multiplications
- ▶ On average, only half of bits e_i are '1':
 - ▶ Only 1024 multiplications
- ▶ Reducing modulo n after every operation!

Faster Decryption Using CRT

Using CRT to decrypt C w.r.t. p, q separately:

- ▶ Compute $M_p = C^{d \bmod (p-1)} \bmod p$ and
 $M_q = C^{d \bmod (q-1)} \bmod q$.
- ▶ Solve $M \bmod n$ using CRT:
 - ▶ $d = (d \bmod (p-1)) + k(p-1)$ for some k :

$$\begin{aligned} M \bmod p &= C^{d \bmod n} \bmod p = C^{d \bmod p} \\ &= C^{d \bmod (p-1)} C^{k(p-1)} \bmod p = C^{d \bmod (p-1)} \\ &= M_p \end{aligned}$$

Thus $M \equiv M_p \bmod p$.

- ▶ Similarly, $M \equiv M_q \bmod q$.
- ▶ Then, output $M = q \times (q^{-1} \bmod p) \times M_p + p \times (p^{-1} \bmod q) \times M_q \bmod n$ (see slide 5 of Lecture 10).

Example

See previous example:

- ▶ $p = 43$, $q = 59$, and so modulus $n = 43 \times 59 = 2537$
- ▶ Ciphertext $C = 2488$ and private exponent $d = 1949$
- ▶ $d \bmod (p - 1) = 1949 \bmod 42 = 17$
- ▶ $d \bmod (q - 1) = 1949 \bmod 58 = 35$
- ▶ $M_p = 2488^{17} \bmod 43 = 37^{17} \bmod 43 = 7$
- ▶ $M_q = 2488^{35} \bmod 59 = 16^{35} \bmod 59 = 50$
- ▶ Using CRT:

$$\begin{aligned} M &= q \times (q^{-1} \bmod p) \times M_p + \\ &\quad p \times (p^{-1} \bmod q) \times M_q \bmod n \\ &= 59 \times (59^{-1} \bmod 43) \times 7 + \\ &\quad 43 \times (43^{-1} \bmod 59) \times 50 \bmod 2537 \\ &= 50 \bmod 2537 \end{aligned}$$

How faster is Decryption with CRT?

- ▶ Exponents $d \bmod (p - 1)$ and $d \bmod (q - 1)$ are about half the length of d .
- ▶ Complexity of exponentiation (with square-and-multiply) increases with the cube of the input length:
 - ▶ Computing M_p and M_q each uses $1/2^3 = 1/8$ of computation for $M = C^d \bmod n$.
- ▶ About 4 times less computation:
 - ▶ If M_p and M_q can be computed in parallel, then the time is up to 8 times faster.
- ▶ Good reason to store p, q with d .

Padding

- ▶ Encryption directly on message encoded as a number is a weak cryptosystem, vulnerable to attacks such as:
 - ▶ Building up a dictionary of known plaintexts.
 - ▶ Guessing the plaintext and checking if it encrypts to the ciphertext.
 - ▶ Håstad's attack.
- ▶ Padding mechanism must be used to prepare message for encryption:
 - ▶ It must include redundancy and randomness.

Håstad's Attack

- ▶ The SAME message M is encrypted without padding to 3 different ciphertexts C_1, C_2, C_3 .
- ▶ Public exponent $e = 3$ used by ALL recipients.
- ▶ Cryptanalysis:

$$C_1 \equiv M^3 \pmod{n_1}$$

$$C_2 \equiv M^3 \pmod{n_2}$$

$$C_3 \equiv M^3 \pmod{n_3}$$

Equations solved using CRT to obtain M^3 in the ordinary (non-modular) integers.

- ▶ M found by taking a cube root.

Padding Types

- ▶ **PKCS #1:** simple, ad-hoc design for encryption and digital signature.
- ▶ **Optimal asymmetric encryption padding (OAEP):**
 - ▶ Designed by Bellare and Rogaway (1994).
 - ▶ Security proof in a suitable model.
 - ▶ **Standard:** IEEE P1363 Standard specifications for public key cryptography.

Outline

Public Key Cryptography

RSA Algorithms

RSA Implementation

RSA Security

Attacks

Most of existing attacks avoided by using standardised padding mechanisms.

- ▶ Factorisation of the modulus n :
 - ▶ Factorisation is believed to be a hard problem.
 - ▶ Factorisation can be prevented by choosing n large enough.
- ▶ Finding d from n and e :
 - ▶ Finding d is as hard for the adversary as factorising the modulus n .

Equivalence with Factorisation Problem

- ▶ An attacker factorises n into its prime factors p, q , and thus recover d :
 - ▶ Breaking RSA is not harder than the factorisation problem.
- ▶ Breaking RSA is shown to be as hard as the RSA problem:
 - ▶ It is unknown if RSA problem is as hard as the factorisation problem.
 - ▶ It is also unknown if factorisation is really computationally hard!

Finding d without factorising the modulus n ? **No!**

Miller's theorem: determining d from e, n is as hard as factorising n .

Other Attacks

- ▶ **Quantum computers:** not existing yet (at least commercially):
 - ▶ Shor's theoretical algorithm can factorise n in polynomial time.
- ▶ **Timing analysis:** using timing of decryption process to obtain information about d :
 - ▶ Demonstrated in practice for RSA in smart cards.
 - ▶ Avoided by randomising decryption process.

Practical Problems with Key Generation

- ▶ Implementation of OpenSSL in Debian-based Linux system used massively reduced randomness for RSA key generation (2008).
- ▶ Lenstra and others published a study of over 6 million RSA keys deployed on the Internet (2012):
 - ▶ 270,000 keys (4%) were identical.
 - ▶ 12,934 keys (0.2%) provide no security because sharing one prime factor with each other.
 - ▶ Certainly due to poor random number generation.

Lecture 13: Public Key Cryptography Part 2

COSC362 Data and Network Security

Book 1: Chapters 9 and 10 – Book 2: Chapters 2 and 21

Spring Semester, 2021

Motivation Reminder

- ▶ Public key cryptography (PKC) has features that symmetric key cryptography does not have.
- ▶ Applied for key management in protocols such as TLS and IPsec.
- ▶ RSA is one of the best known public key cryptosystems, widely deployed in practice.
- ▶ Alternatives include discrete log based ciphers, also widely deployed and standardised.

Outline

Diffie-Hellman Key Exchange

Protocol

Properties

Elgamal Cryptosystem

Algorithms

Security

Elliptic Curves

Recent Developments

Outline

Diffie-Hellman Key Exchange

Protocol

Properties

Elgamal Cryptosystem

Algorithms

Security

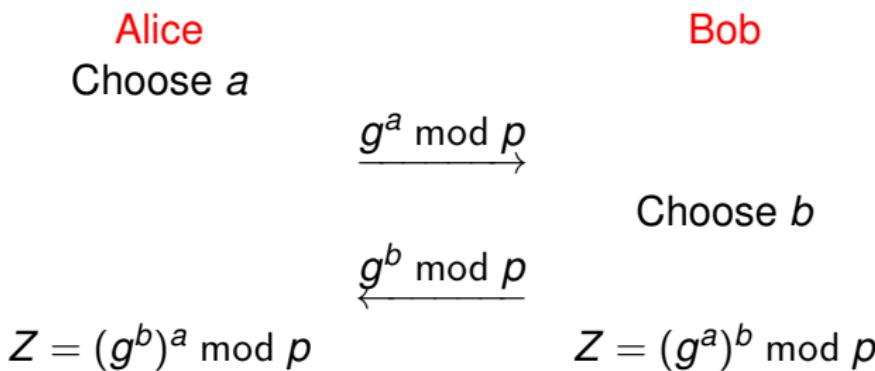
Elliptic Curves

Recent Developments

Diffie-Hellman Key Exchange

- ▶ 2 users, Alice and Bob, share a secret using only public communication.
- ▶ **Public elements:**
 - ▶ Large prime p
 - ▶ Generator $g \in \mathbb{Z}_p^*$
- ▶ Alice and Bob each selects random values a and b respectively, where $1 < a, b < p$:
 - ▶ Alice sends g^a to Bob over an *insecure* channel.
 - ▶ Bob sends g^b to Alice over an *insecure* channel.
- ▶ Alice and Bob both compute the secret key $Z = g^{ab} \bmod p$.

Protocol



Z can be used to compute a key (e.g. AES) by using a *key derivation function* based on a public hash function.

Example

Public elements are $p = 181$ and $g = 2$.

► Selecting private keys:

- Alice selects $a = 50$
- Bob selects $b = 33$

► Sharing public keys:

- Alice sends $g^a \bmod p = 2^{50} \bmod 181 \equiv 116$ to Bob
- Bob sends $g^b \bmod p = 2^{33} \bmod 181 \equiv 30$ to Alice

► Computing the shared key:

- Alice computes $Z = (g^b)^a \bmod p \equiv 30^{50} \bmod 181$
- Bob computes $Z = (g^a)^b \bmod p \equiv 116^{33} \bmod 181$

The common secret is $Z = 49$.

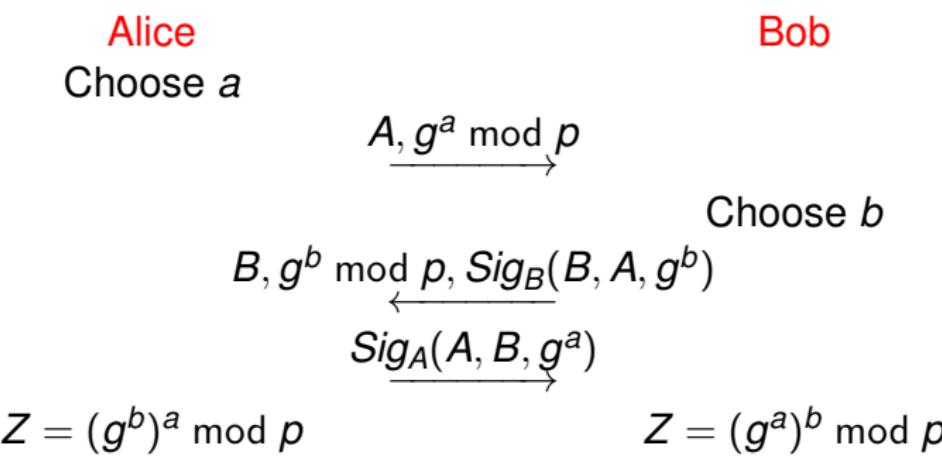
Security

- ▶ An attacker who finds discrete logarithms breaks the protocol:
 - ▶ Intercepting $g^a \bmod p$ and taking the discrete log to get a .
 - ▶ Computing $(g^b)^a$ in the same way as Bob.
- ▶ No better way known for a passive adversary than by taking discrete logs:
 - ▶ It is unknown if there is a better way.

Authenticated Diffie-Hellman

- ▶ In the basic protocol:
 - ▶ Messages between Alice and Bob are not authenticated.
- ▶ In a network, Alice/Bob do not know how Z is shared, unless messages are authenticated.
- ▶ **Man-in-the-middle attack:** the adversary sets up 2 keys, 1 with Alice and 1 with Bob, and relays messages between the 2.
- ▶ **Authentication feature:**
 - ▶ Authentication can be added by using digital signatures.

Authenticated Diffie-Hellman



- ▶ Signature $\text{Sig}_A(m)$ on message m by Alice
- ▶ Signature $\text{Sig}_B(m)$ on message m by Bob
- ▶ Both parties know each other's public signature verification key.

Static and Ephemeral Diffie-Hellman

- ▶ The above protocol uses *ephemeral keys*:
 - ▶ Key used once and then discarded.
- ▶ In the *static* protocol:
 - ▶ Alice chooses a long-term private key x_A and public key $y_A = g^{x_A} \bmod p$.
 - ▶ Bob chooses a long-term private key x_B and public key $y_B = g^{x_B} \bmod p$.
- ▶ Alice and Bob find a shared secret $S = g^{x_A x_B} \bmod p$, that is static:
 - ▶ S stays the same until Alice and Bob change their public keys.

Outline

Diffie-Hellman Key Exchange

Protocol

Properties

Elgamal Cryptosystem

Algorithms

Security

Elliptic Curves

Recent Developments

Elgamal Cryptosystem



- ▶ Proposed by Taher Elgamal in 1985.
- ▶ Diffie-Hellman protocol turned into a cryptosystem.
- ▶ For encryption and for signature.
- ▶ Alice combines her ephemeral private key with Bob's long-term public key.

Key Generation

Key generation:

- ▶ Select a prime p and a generator $g \in \mathbb{Z}_p^*$.
- ▶ Select a long-term private key $K_D = x$, where $1 < x < p$.
- ▶ Compute $y = g^x \bmod p$.
- ▶ Set the long-term public key as $K_E = (p, g, y)$.

Encryption and Decryption

Encryption:

- ▶ $K_E = (p, g, y)$ is the public key for encryption.
- ▶ Select a message M where $0 < M < p$.
- ▶ Choose at random an ephemeral private key k .
- ▶ Compute $g^k \pmod{p}$ and $My^k \pmod{p}$.
- ▶ Set the ciphertext as:

$$C = (C_1, C_2) = Enc(M, K_E) = (g^k \pmod{p}, My^k \pmod{p})$$

Decryption:

- ▶ $K_D = x$ is the private key for decryption.
- ▶ $C = (C_1, C_2)$ is the ciphertext.
- ▶ Compute $C_1^x \pmod{p}$.
- ▶ $Dec(C, K_D) = C_2 \cdot (C_1^x)^{-1} \pmod{p} = M$.

Correctness

- ▶ Alice knows the ephemeral private key k .
- ▶ Bob knows the static/long-term private key $K_D = x$.
- ▶ Both Alice and Bob compute the Diffie-Hellman value for the 2 public keys:
 - ▶ $C_1 = g^k \pmod{p}$
 - ▶ $y = g^x \pmod{p}$
- ▶ Diffie-Hellman value $y^k \pmod{p} = C_1^x \pmod{p}$ used as a mask for the message M .

Example

Key generation:

- ▶ Choose prime $p = 181$ and generator $g = 2$.
- ▶ Bob's private key is $x = 50$.
- ▶ Compute $y = g^x \bmod p = 116$.
- ▶ Bob's public key is $(181, 2, 116)$.

Encryption:

- ▶ Alice wants to send $M = 97$.
- ▶ Alice chooses at random $k = 31$.
- ▶ Ciphertext is $C = (C_1, C_2) = (98, 173)$.

Decryption:

- ▶ Bob receives $C = (C_1, C_2)$.
- ▶ Bob computes $C_1^x \bmod p = 98^{50} \bmod 181 = 138$.
- ▶ Bob recovers $M = C_2 \times (C_1^x)^{-1} \bmod p = 173 \times 138^{-1} \bmod 181 = 97$.

Security

- ▶ An attacker who solves the discrete log problem breaks Elgamal cryptosystem by determining the private key x from $g^x \bmod p$.
- ▶ Possible for many users to share the same p and g .
- ▶ No need for any padding as in RSA:
 - ▶ Each ciphertext is already randomised, thanks to the ephemeral key k .
- ▶ Security proof in a suitable model, subject to the difficulty of *decisional Diffie-Hellman problem*.

Outline

Diffie-Hellman Key Exchange

Protocol

Properties

Elgamal Cryptosystem

Algorithms

Security

Elliptic Curves

Recent Developments

Elliptic Curves

- ▶ Algebraic structures formed from cubic equations.
- ▶ Curves defined over any field.
- ▶ Example:
 - ▶ Set of all (x, y) pairs which satisfy $y^2 = x^3 + ax + b \pmod{p}$.
 - ▶ Curve over field \mathbb{Z}_p .
- ▶ Add an identity element, and then define a binary operation on the points (e.g. multiplication):
 - ▶ Form a group over the elliptic curve points, called *elliptic curve group*.

Choosing Elliptic Curves

- ▶ Generate a new elliptic curve at any time:
 - ▶ But applications usually use standardised curves.
 - ▶ **Standard:** FIPS 186-4 (NIST curves, 2013).
- ▶ Standardised curves generated in a verifiably random way:
 - ▶ Difficult to generate curves with any hidden special properties.

Example

NIST curve P-192:

- ▶ Curve of n points over \mathbb{Z}_p with generator (G_x, G_y) and equation $y^2 = x^3 - 3x + b \bmod p$.
- ▶ p and n are 192 bits long.
- ▶ s is the seed for random generation.
- ▶ c is the output of a SHA-1 hash generated from s .

$p = 6277101735386680763835789423207666416083908700390324961279$

$n = 6277101735386680763835789423176059013767194773182842284081$

$s = 3045ae6fc8422f64ed579528d38120eae12196d5$

$c = 3099d2bbbfc2538542dcd5fb078b6ef5f3d6fe2c745de65$

$b = 64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1$

$G_x = 188da80eb03090f67cbf20eb43a18800f4ff0af82ff1012$

$G_y = 07192b95ffc8da78631011ed6b24cdd573f977a11e794811$

Discrete Logarithm

- ▶ Discrete log defined on elliptic curve groups:
 - ▶ If elliptic curve operation denoted as multiplication, then definition same as in \mathbb{Z}_p^* .
- ▶ Best known algorithms for solving discrete log problem are *exponential* in the length of parameters.
- ▶ Elliptic curve implementations use smaller keys.
- ▶ Comparison with RSA:
 - ▶ Relative advantage of elliptic curve cryptography increases at higher security levels.

Security Comparison

Sym. key length	RSA modulus length	EC element length
80	1024	160
128	3072	256
192	7680	384
256	15360	512

Example: brute force search of 128-bit key for AES takes roughly same computational effort as factorisation of 3072-bit RSA modulus or for taking discrete logarithms in an elliptic curve with elements of size 256 bits.

Standard: NIST SP 800-57 Part 1 Recommendations for Key Management (revised 2016).

Elliptic Curve Cryptography

- ▶ Most cryptosystems based on discrete log constructed with elliptic curves as well as in \mathbb{Z}_p^* .
- ▶ Examples of cryptosystems run on elliptic curves:
 - ▶ Diffie-Hellman key exchange
 - ▶ Elgamal encryption
- ▶ Canadian company Certicom holds ECC-based patents:
 - ▶ <https://www.certicom.com/content/certicom/en/about.html>

Outline

Diffie-Hellman Key Exchange

Protocol

Properties

Elgamal Cryptosystem

Algorithms

Security

Elliptic Curves

Recent Developments

Identity-Based Cryptography

- ▶ Proposed by Shamir (1982).
- ▶ Extensively researched in the 2000s, with the use of elliptic curve *pairings*.
- ▶ Public keys and certificates are not needed:
 - ▶ Identity of the key owner replaces the public key.
 - ▶ Message encryption using public parameters and recipient's identity.
- ▶ **Limitation:**
 - ▶ Need of a trusted key generation process.
- ▶ **Generalisation with functional cryptography:**
 - ▶ General access policies used to define who may decrypt the ciphertext.

Post-Quantum Cryptography

- ▶ Most current public key cryptography will be broken if quantum computers become available:
 - ▶ Shor's algorithm enabling factorisation.
 - ▶ Shor's algorithm also enabling to find discrete logarithms.
- ▶ **Concerns:** building cryptographic primitives still secure if this happens!
- ▶ Symmetric key cryptography can be used but with double-length keys:
 - ▶ Grover's algorithm allowing searching.
- ▶ Post-quantum cryptosystems based on different problems:
 - ▶ Lattice problems, coding theory, multi-variable polynomial resolution.
- ▶ NIST process to standardise is under way:
 - ▶ <https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization>

Lecture 14: Digital Signatures

COSC362 Data and Network Security

Book 1: Chapter 13 – Book 2: Chapter 2

Spring Semester, 2021

Motivation

- ▶ Digital signatures are one of the main benefits of public cryptosystems.
- ▶ In some countries, digital signatures are legally binding in the same way as handwritten signatures.

Outline

Properties

RSA Signatures

Discrete Logarithm Signatures

Elgamal Signatures

Digital Signature Standard

Outline

Properties

RSA Signatures

Discrete Logarithm Signatures

Elgamal Signatures

Digital Signature Standard

Confidentiality and Authentication

- ▶ Message authentication codes (MACs) only allow an entity with shared secret to generate a valid tag:
 - ▶ Providing data integrity and data authentication.
- ▶ Digital signatures use public key cryptography to provide properties of a MAC and more:
 - ▶ Only the owner of the private signing key can generate a valid digital signature.
- ▶ **Security service:**
 - ▶ Non-repudiation
 - ▶ A judge can decide which party has formed the signature

Comparing Physical and Digital Signatures

Physical signatures	Digital signatures
Produced by a human Same on all documents Easy to recognize	Produced by a machine Function of the message Requiring a computer to check

Both signature types need to be difficult to forge.

Algorithms

- ▶ **Algorithms:**
 - ▶ Key generation
 - ▶ Signature generation
 - ▶ Signature verification
- ▶ The key generation algorithm outputs 2 keys:
 - ▶ A private *signing* key K_S
 - ▶ A public *verification* key K_V

Signature Generation Algorithm

Alice wants to generate a signature on a message M :

► Inputs:

- Alice's private signing key K_S
- Message M

► Output:

- Signature $s = \text{Sig}(M, K_S)$
- Only Alice, the owner of K_S , should be able to generate a valid signature.
- The message should be any bit string.
- The set of all signatures is usually a set of fixed size.

Signature Verification Algorithm

Bob wants to verify a claimed signature s on message M :

- ▶ Inputs:
 - ▶ Alice's public verification key K_V
 - ▶ Message M
 - ▶ Claimed signature s
- ▶ Output:
 - ▶ Boolean value $\text{Ver}(M, s, K_V) = \text{true/false}$
 - ▶ Anyone should be able to verify the signature.

Properties

- ▶ **Correctness:**
 - ▶ If $s = \text{Sig}(M, K_S)$ then $\text{Ver}(M, s, K_V) = \text{true}$ for any matching K_S and K_V .
- ▶ **Unforgeability:**
 - ▶ It is computationally infeasible for anyone without K_S to construct the pair (M, s) s.t. $\text{Ver}(M, s, K_V) = \text{true}$.
- ▶ The signing algorithm Sig may be randomized:
 - ▶ There are many possible signatures for a single message.
- ▶ **Stronger security definition:**
 - ▶ An attacker has access to a *chosen message oracle*.
 - ▶ Forging a new signature should be difficult even if the attacker can obtain signatures on messages of her choice.

Security Goals

- ▶ **Key recovery:**
 - ▶ The attacker attempts to recover the private signing key K_S from the public verification key K_S and some known signatures.
- ▶ **Selective forgery:**
 - ▶ The attacker chooses a message and attempts to obtain a signature on that message.
- ▶ **Existential forgery:**
 - ▶ The attacker attempts to forge a signature on any message not previously signed.
 - ▶ It could be a meaningless message.
- ▶ Modern digital signatures are seen secure if they can resist *existential forgery under a chosen message attack*.

Outline

Properties

RSA Signatures

Discrete Logarithm Signatures

Elgamal Signatures

Digital Signature Standard

Key Generation

RSA signature keys are generated in the same way as RSA encryption keys:

- ▶ **Public verification key:** n, e where $n = pq$ for large primes p, q .
- ▶ **Private signing key:** p, q, d s.t. $ed \bmod \phi(n) = 1$.

A hash function h is also required, as a fixed public parameter. It can be a standard hash function (e.g. SHA-256).

Signature Generation and Verification

Signature generation:

- ▶ Inputs are message M , modulus n and private exponent d .
- ▶ Compute $s = h(M)^d \bmod n$.
- ▶ Output (M, s) as the signature.

Signature verification:

- ▶ Inputs are claimed signature (M, s) , modulus n and public exponent e .
- ▶ Compute $h' = h(M)$.
- ▶ Check if $s^e \bmod n = h'$? If so, then output true; otherwise, output false:
 - ▶ Check Lecture 12 for correctness.

Outline

Properties

RSA Signatures

Discrete Logarithm Signatures

Elgamal Signatures

Digital Signature Standard

Discrete Logarithm Signatures

- ▶ Security relying on difficulty of discrete logarithm problem.
- ▶ 3 versions:
 1. Original Elgamal signatures in \mathbb{Z}_p^* (1985).
 2. Digital signature algorithm (DSA) standardised by NIST:
 - ▶ an optimized version of Elgamal signatures
 3. DSA based on elliptic curve groups, known as ECDSA.

Elgamal Elements in \mathbb{Z}_p^*

- ▶ p is a large prime.
- ▶ g is generator of \mathbb{Z}_p^* .
- ▶ x is the private signing key s.t. $0 < x < p - 1$.
- ▶ p, g, y form the public verification key where $y = g^x \pmod{p}$.
- ▶ Alice wants to sign a value M where $0 < M < p - 1$.

Elgamal Operations in \mathbb{Z}_p^*

Signature generation:

1. Alice selects a random k s.t. $\gcd(k, p - 1) = 1$ and computes

$$r = g^k \pmod{p}$$

2. Alice solves $M = xr + ks \pmod{p - 1}$ for s by computing

$$s = k^{-1}(M - xr) \pmod{p - 1}$$

3. Alice outputs the tuple (M, r, s) .

Signature verification:

- Bob checks if $g^M \equiv y^r r^s \pmod{p}$ ($= (g^x)^r (g^k)^s$).

Digital Signature Algorithm (DSA)

- ▶ First published by NIST in 1994.
- ▶ **Standard:** FIPS PUB 186-4 (2013).
- ▶ Based on Elgamal signatures.
- ▶ Simpler calculations and shorter signatures:
 - ▶ Calculations done in a *subgroup* of \mathbb{Z}_p^* or an elliptic curve group.
- ▶ Used with SHA family of hash functions.
- ▶ Preventing attacks that Elgamal signatures may be vulnerable to.

Idea

- ▶ Prime p chosen s.t. $p - 1$ has a prime divisor q of much smaller size (224 or 256 bits).
- ▶ Generator g used in Elgamal signatures replaced by $g = h^{\frac{p-1}{q}} \pmod{p}$ where h is a generator:
 - ▶ g has order q since $g^q \pmod{p} = 1$:
 - ▶ $g^q \pmod{p} = (h^{\frac{p-1}{q}})^q \pmod{p} = h^{p-1} \pmod{p} = 1$ (Fermat's theorem).
 - ▶ All exponents can be thus reduced modulo q before exponentiation.

Comparison

Differences with Elgamal signatures:

- ▶ Message is hashed using standard SHA hash algorithm.
- ▶ g chosen to be of order q , which is much smaller than p .
- ▶ Verification equation becomes¹:

$$(g^{H(M)})^{s^{-1}}(y^{-r})^{s^{-1}} \equiv r \pmod{p}$$

Both sides of the equation are then reduced modulo q .

¹from $g^{H(M)} \equiv y^r r^s \pmod{p}$

Parameters

- ▶ p is a prime modulus, of L bits.
- ▶ q is a prime divisor of $p - 1$, of N bits.
- ▶ Valid combinations:
 - ▶ $L = 1024, N = 160$
 - ▶ $L = 2048, N = 224$
 - ▶ $L = 2048, N = 256$
 - ▶ $L = 3072, N = 256$
- ▶ $g = h^{\frac{p-1}{q}} \pmod{p}$ is the generator where $1 < h < p - 1$.
- ▶ H is the hash function from SHA family variant s.t. the output is an N -bit digest.

Key and Signature Generations

Key generation:

- ▶ Choose a random integer x s.t. $0 < x < q$.
- ▶ Compute $y = g^x \pmod{p}$.
- ▶ Set x as the secret key and y as the public key.

Signature generation:

- ▶ Let M be a message.
- ▶ Choose k at random s.t. $0 < k < q$.
- ▶ Compute $r = (g^k \pmod{p}) \pmod{q}$.
- ▶ Compute $s = k^{-1}(H(M) - xr) \pmod{q}$.
- ▶ Set (M, r, s) as the signature.

Signature Verification

Signature verification:

- ▶ (M, r, s) is the claimed signature.
- ▶ Check if $0 < r < q$ and $0 < s < q$.
- ▶ Compute $w = s^{-1} \pmod{q}$.
- ▶ Compute $u_1 = H(M)w \pmod{q}$.
- ▶ Compute $u_2 = rw \pmod{q}$.
- ▶ Check if $(g^{u_1}y^{-u_2} \pmod{p}) \pmod{q} = r$.

Comparison

Differences with Elgamal signatures:

- ▶ Verification equation is the same, except that all exponents and final result are reduced modulo q .
- ▶ Signature generation mainly requires one exponentiation with a short exponent (224 or 256 bits).
- ▶ Signature verification requires 2 short exponentiations.
- ▶ Signature size is only $2N$ bits:
 - ▶ 448 bits when $N = 224$
 - ▶ 512 bits when $N = 256$

Parameter Values

Key lengths defined in the 2013 standard version:

Version no.	$ p $	$ q $	Hash function
1	1024 bits	160 bits	SHA-1
2	2048 bits	224 bits	SHA-224
3	2048 bits	256 bits	SHA-256
4	3072 bits	256 bits	SHA-256

NIST special publication SP 800-57 does NOT approve Version no. 1.

Elliptic Curve DSA (ECDSA)

- ▶ **Standard:** FIPS PUB 186-4 (2013).
- ▶ Parameters chosen from NIST approved curves.
- ▶ Signature generation and verification are the same, except that:
 - ▶ q becomes the order of the elliptic curve group.
 - ▶ Multiplication modulo p is replaced by the elliptic curve group operation.
 - ▶ After operations on group elements, only the x coordinate is kept (from the pair (x, y)).

Comparison

ECDSA versus DSA:

- ▶ ECDSA signatures are generally not shorter than DSA signatures for the same security level.
- ▶ ECDSA signature size varies with the underlying curve:
 - ▶ Between 326 bits and 1142 bits from approved curves.
- ▶ ECDSA public keys are shorter than DSA public keys.

Lecture 15: PKI and Certificates

COSC362 Data and Network Security

Book 1: Chapter 14 – Book 2: Chapters 21 and 23

Spring Semester, 2021

Motivation

- ▶ Public key infrastructures imply the use of public digital certificates.
- ▶ Digital signatures provide these certificates.
- ▶ X.509 certificates are standardised and used in most network security applications.

Outline

Public Key Infrastructure (PKI)

Digital Certificates
Trust of Certificates

PKI Examples

Outline

Public Key Infrastructure (PKI)

Digital Certificates

Trust of Certificates

PKI Examples

Definition

- ▶ **NIST definition:**
 - ▶ “A public key infrastructure is the key management environment for public key information of a public key cryptographic system.”
- ▶ Key management concerned with *lifecycle* of cryptographic keys:
 - ▶ Generation, distribution, storage and destruction of keys.
- ▶ Various legal or business (trusted) entities may be involved:
 - ▶ **Registration authorities (RAs):** vouching for the identity of an user.
 - ▶ **Validation authorities (VAs):** verifying that identity.
 - ▶ **Certification authorities (CAs):** issuing digital certificates (certifying the public key of the user).

Outline

Public Key Infrastructure (PKI)

Digital Certificates
Trust of Certificates

PKI Examples

Digital Certificates

- ▶ How to be confident of the correct binding between a public key and its owner?
 - ▶ When using a public key to encrypt a message or to verify a digital signature.
- ▶ Achieved through the use of *digital certificates*:
 - ▶ They contain the public key and owner identity.
 - ▶ There is other information such as signature algorithm and validity period.
- ▶ Certificate digitally signed by a *certification authority* (CA):
 - ▶ CA should be trusted by the certificate verifier.
- ▶ Certificates play a central role in key management for PKIs.

Certification Authority (CA)

- ▶ A CA creates, issues and revokes certificates for subscribers and other CAs.
- ▶ A CA has a *certification practice statement* (CPS).
- ▶ A CPS covers issues such as:
 - ▶ Checks performed before certificate issue
 - ▶ Physical, personnel and procedural security controls for the CA
 - ▶ Technical and key pair protection and management controls
 - ▶ Certificate revocation management procedures
 - ▶ Accreditation information
 - ▶ Legal and privacy issues and liability limitations

X.509 Standard

- ▶ Most widely used standard:
 - ▶ Originally ITU standard.
 - ▶ Now RFC 5280.
 - ▶ Current version (number 3) allows flexible extensions.
- ▶ Important fields in X.509 certificates:
 - ▶ Version number
 - ▶ Serial number (set by the CA)
 - ▶ Signature algorithm identifier (algorithm used to digitally sign)
 - ▶ Issuer name (of the CA)
 - ▶ Subject name (of the user to which the certificate is issued)
 - ▶ Public key information
 - ▶ Validity period
 - ▶ Digital signature (of the certificate, generated by the CA)

Example

 **www.ssl.com**
Issued by: SSL.com EV SSL Intermediate CA RSA R3
Expires: Saturday, April 17, 2021 at 5:15:06 PM Central Daylight Time
 This certificate is valid

▼ Details

Subject Name	
Country or Region	US
State/Province	Texas
Locality	Houston
Organization	SSL Corp
Serial Number	NV20081614243
Common Name	www.ssl.com
Postal Code	77098
Business Category	Private Organization
Street Address	3100 Richmond Ave
Inc. State/Province	Nevada
Inc. Country/Region	US

- ▶ Information about the subject (the user to which the certificate is issued)

Issuer Name
Country or Region US
State/Province Texas
Locality Houston
Organization SSL Corp
Common Name SSL.com EV SSL Intermediate CA RSA R3

- ▶ Information about the issuer (the CA)

Example

- ▶ Serial number
 - ▶ X.509 version (3)
 - ▶ Signature algorithm
 - ▶ Period validity
 - ▶ Information about the public key
 - ▶ Information about the signature

Using a Certificate

- ▶ **Verifying a certificate:**

- ▶ By checking that the CA's signature is valid.
 - ▶ By checking that any conditions set in the certificate are correct.

- ▶ **In order to verify a certificate:**

- ▶ The user of the certificate must have the correct public key of the CA.
 - ▶ It does not matter how the user obtains the certificate.
 - ▶ Public directories may store certificates:
 - ▶ Often, the owner of the public key sends the certificate to the user.

Certification Paths

- ▶ Suppose that the public key of the CA ca_0 is not already known and trusted.
- ▶ Then, ca_0 's public key can be certified by another CA ca_1 .
- ▶ In turn, ca_1 's public key can be certified by another CA ca_2 .
- ▶ Thus, a *chain of trust* is set up, known as a *certification path*:

$$ca_n \rightarrow \dots \rightarrow ca_2 \rightarrow ca_1 \rightarrow ca_0$$

- ▶ Suppose that an entity has a trusted copy of ca_n 's public key.
- ▶ The chain of trust is used with certificates for all the *intermediate CAs* to obtain a trusted copy of ca_0 's public key.

Phishing Attack

- ▶ The victim connects securely to a bogus site with the wrong certificate.
- ▶ The attacker makes the URL similar and the interface identical to a genuine site.



- ▶ If the website uses a certificate, then the *padlock* indicator still shows:
 - ▶ But the secure connection is to the attacker's site.

- ▶ Not always easy to tell if a certificate is one for a genuine site.

Extended Validation Certificates



- ▶ Browser indication:
 - ▶ A color in the address bar to indicate that the certificate has been issued at a specified level.
- ▶ Agreement between browser developers and CAs:
 - ▶ No technical difference in the certificate.
 - ▶ Just signed by a specific intermediate CA.
- ▶ Surveys have shown that extended validation certificates are mostly ignored by users.

Revocation

- ▶ Declaring a certificate invalid even though its validity period is current.
- ▶ The user must check which certificates have been revoked.
- ▶ **Certificate revocation list (CRL):**
 - ▶ Each CA periodically issues a list of revoked certificates which can be downloaded and then checked by clients before using a certificate.
- ▶ **Online certificate status protocol (OCSP):**
 - ▶ A server maintains a current list of revoked certificates and responds to requests about specific certificates.

Public Key Pinning

- ▶ Allowing browsers to fix for a certain time the public key used to verify certificates for certain sites.
- ▶ Preventing attacks due to compromised CAs.
- ▶ Supported by Firefox and other browsers.
- ▶ Previously supported by Chrome, but Google announced to remove it (Oct. 2017).

Outline

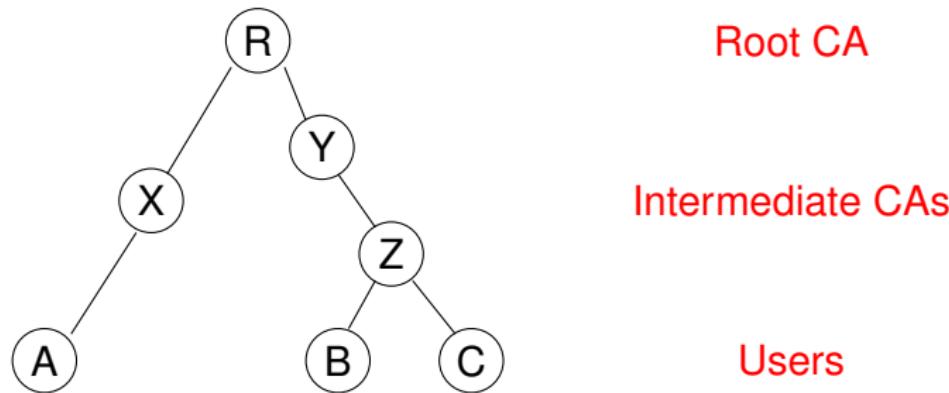
Public Key Infrastructure (PKI)

Digital Certificates

Trust of Certificates

PKI Examples

Hierarchical PKI



- ▶ A CA certifies the public key of the entity below.
- ▶ In a non-hierarchical PKI, certification done between any CAs:
 - ▶ X can certify Y 's public key, or Z can certify Y 's public key.

Browser PKI

- ▶ Multiple hierarchies with preloaded public keys as root CAs.
- ▶ Intermediate CAs can be added.
- ▶ Users can also add their own certificates.
- ▶ Most servers send their public key and certificate to the browser at the start of a secure communication using TLS protocol.

OpenPGP PKI

- ▶ Used in PGP email security.
- ▶ Certificate includes ID, public key, validity period and *self-signature*.
- ▶ There is NO certification authorities:
 - ▶ Keys signed by anyone.
- ▶ Various key servers store keys:
 - ▶ Example: <http://pgp.mit.edu>
- ▶ Often known as *web of trust*.

Do we have some time left?

Yes, then let's start Lecture 16!

Lecture 16: Key Establishment

COSC362 Data and Network Security

Book 1: Chapters 14 and 15 – Book 2: Chapters 21 and 23

Spring Semester, 2021

Motivation

- ▶ Distribution of cryptographic keys to protect subsequent communications sessions.
- ▶ Key establishment in TLS uses public keys to allow clients and servers to share a new communication key.
- ▶ Kerberos is a widely used system for secure communications which achieves key establishment without using public keys.

Outline

Key Establishment

 Key Pre-Distribution

 Session Key Distribution using Symmetric Keys

 Session Key Distribution using Asymmetric Keys

Example of Session Key Distribution using Asymmetric Keys

 Signed Diffie-Hellman

Examples of Session Key Distribution using Symmetric Keys

 Needham-Schroeder Protocol

 Kerberos

Outline

Key Establishment

 Key Pre-Distribution

 Session Key Distribution using Symmetric Keys

 Session Key Distribution using Asymmetric Keys

Example of Session Key Distribution using Asymmetric Keys

 Signed Diffie-Hellman

Examples of Session Key Distribution using Symmetric Keys

 Needham-Schroeder Protocol

 Kerberos

Key Management

- ▶ Critical aspect of any cryptographic system.
- ▶ Phases:
 - ▶ **Key generation:** keys should be generated s.t. they are equally likely to occur.
 - ▶ **Key distribution:** keys should be distributed in a secure fashion.
 - ▶ **Key protection:** keys should be accessible for use in relevant cryptographic algorithms, but not accessible to unauthorised parties.
 - ▶ **Key destruction:** once a key has performed its function, it should be destroyed s.t. it is of no value to an attacker.

Key Types

Keys are often organized in a hierarchy.

A simple 2-level hierarchy is common:

- ▶ Long-term keys:

- ▶ Also called *static keys*.
- ▶ Intended to be used for a long time.
- ▶ Depending upon the application, from few hours to few years.
- ▶ Used to protect distribution of session keys.

- ▶ Short-term keys:

- ▶ Also called *session keys*.
- ▶ Intended to be used over a short period.
- ▶ Depending upon the application, from few seconds to few hours.
- ▶ Used to protect communications in a session (e.g. with authenticated encryption).

Key Establishment

- ▶ In practice, session keys are symmetric keys used with ciphers (e.g. AES, MAC):
 - ▶ Due to their greater efficiency over public key algorithms.
- ▶ Long-term keys can be either symmetric or asymmetric keys, depending on how they are used.
- ▶ How to *establish* secret session keys among communicating parties using the long-term keys.
- ▶ Common approaches:
 - ▶ Key pre-distribution.
 - ▶ Using an online server with symmetric long-term keys.
 - ▶ Using asymmetric long-term keys.

Key Distribution Security Goals

2 properties:

- ▶ **Authentication:** if Alice completes the protocol and believes that the key is shared with Bob, then it should not be the case that the key is actually shared with another party Carol.
- ▶ **Confidentiality:** the adversary is unable to obtain the session key accepted by a particular party.

In formal models, the protocol is seen as *broken* if the adversary can distinguish the session key from a random string.

Mutual and Unilateral Authentication

- ▶ If both parties achieve the authentication goal, then the protocol provides *mutual authentication*.
- ▶ If only one party achieves it, then the protocol provides *unilateral authentication*.
- ▶ Many real-world key establishment protocols achieve only unilateral authentication:
 - ▶ Typically, clients can authenticate servers.
 - ▶ Client authentication often happens later, protected with the established key.

Adversary Capabilities

Let a strong adversary know the details of the cryptographic algorithms involved and be able to:

- ▶ *Eavesdrop* on all messages sent in a protocol.
- ▶ *Alter* all messages sent in a protocol using any information available to him/her.
- ▶ *Re-route* any messages (including new ones) to any other party.
- ▶ *Obtain* the value of the session key used in any previous run of the protocol.

Distribution of Pre-Shared Keys

- ▶ A trusted authority (TA) generates and distributes long-term keys to all users when they join the system.
- ▶ **Simple schemes:**
 - ▶ Assigning a secret key for each pair of users.
 - ▶ The number of keys thus grows quadratically.
- ▶ The TA only operates in the pre-distribution phase:
 - ▶ It does not need to be online afterwards.
- ▶ Poor scalability.
- ▶ **Probabilistic schemes:**
 - ▶ Reducing key material at each party.
 - ▶ But only guaranteeing a secure channel between any 2 users with some (high) probability.
 - ▶ Suitable for sensor networks.

Key Distribution using Symmetric Keys

- ▶ Key distribution with an online server.
- ▶ The TA shares a long-term shared key with each user.
- ▶ An online TA generates and distributes session keys to users when requested:
 - ▶ In a secure fashion using the long-term keys.
- ▶ The TA is highly trusted and is a single point of attack:
 - ▶ The security of the whole network depends on it.
- ▶ Scalability can be a problem.

Key Distribution using Asymmetric Cryptography

- ▶ No online TA is required.
- ▶ Public keys used for authentication.
- ▶ Public keys managed by PKI (certificates and CAs).
- ▶ Users are trusted to generate good session keys:
 - ▶ A good pseudo-random number generator required at each party.
- ▶ Types:
 - ▶ *key transport*
 - ▶ *key agreement*

Forward Secrecy

What happens when a long-term key is compromised?

- ▶ The attacker can now act as the owner of the long-term key.
- ▶ Previous session keys may also be compromised:
 - ▶ This is the case with key transport!
 - ▶ This can be prevented with key agreement.

A protocol provides *(perfect) forward secrecy* if compromise of long-term secret keys does NOT reveal session keys previously agreed using those long-term keys.

Key Transport

- ▶ User chooses key material and sends it encrypted to another party:
 - ▶ Sometimes, the message is also signed by the sender.
- ▶ TLS includes options for key transport.
- ▶ Not providing *forward secrecy*.

Key Agreement

- ▶ 2 parties each provide input to the key material.
- ▶ Providing authentication with public keys:
 - ▶ By signing the exchanged messages.
- ▶ **Example:** Diffie-Hellman protocol (widely used).
- ▶ TLS includes options for key agreement.
- ▶ Providing *forward secrecy*.

Outline

Key Establishment

 Key Pre-Distribution

 Session Key Distribution using Symmetric Keys

 Session Key Distribution using Asymmetric Keys

Example of Session Key Distribution using Asymmetric Keys

 Signed Diffie-Hellman

Examples of Session Key Distribution using Symmetric Keys

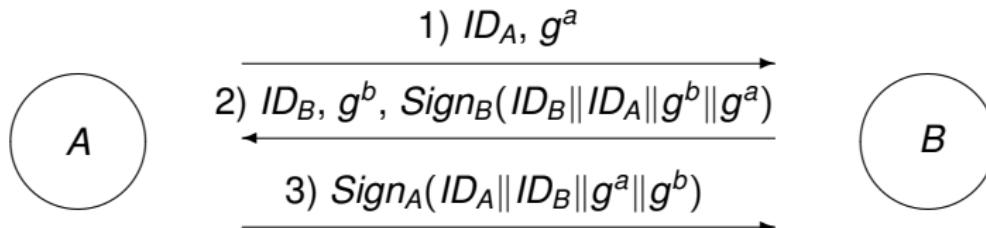
 Needham-Schroeder Protocol

 Kerberos

Notation

- ▶ Alice and Bob want to share a secret key.
- ▶ Computations done in \mathbb{Z}_p^* with a large prime p :
 - ▶ Generator g .
 - ▶ Random values a, b chosen by Alice and Bob, where $1 \leq a, b \leq p - 1$.
 - ▶ $Sign_A(m)$ is a signature on message m from Alice.
 - ▶ $Sign_B(m)$ is a signature on message m from Bob.
- ▶ Both Alice and Bob know each other's public verification key.
- ▶ Forward secrecy since long-term signing keys are only used for authentication.

Protocol



- ▶ Alice checks the signature in flow 2:
 - ▶ If invalid then Alice aborts.
 - ▶ Otherwise, Alice computes the session key as

$$K_{AB} = (g^b)^a = g^{ab}$$

- ▶ Bob checks the signature in flow 3:
 - ▶ If invalid then Bob aborts.
 - ▶ Otherwise, Bob computes the session key as

$$K_{AB} = (g^a)^b = g^{ab}$$

Outline

Key Establishment

 Key Pre-Distribution

 Session Key Distribution using Symmetric Keys

 Session Key Distribution using Asymmetric Keys

Example of Session Key Distribution using Asymmetric Keys

 Signed Diffie-Hellman

Examples of Session Key Distribution using Symmetric Keys

 Needham-Schroeder Protocol

 Kerberos

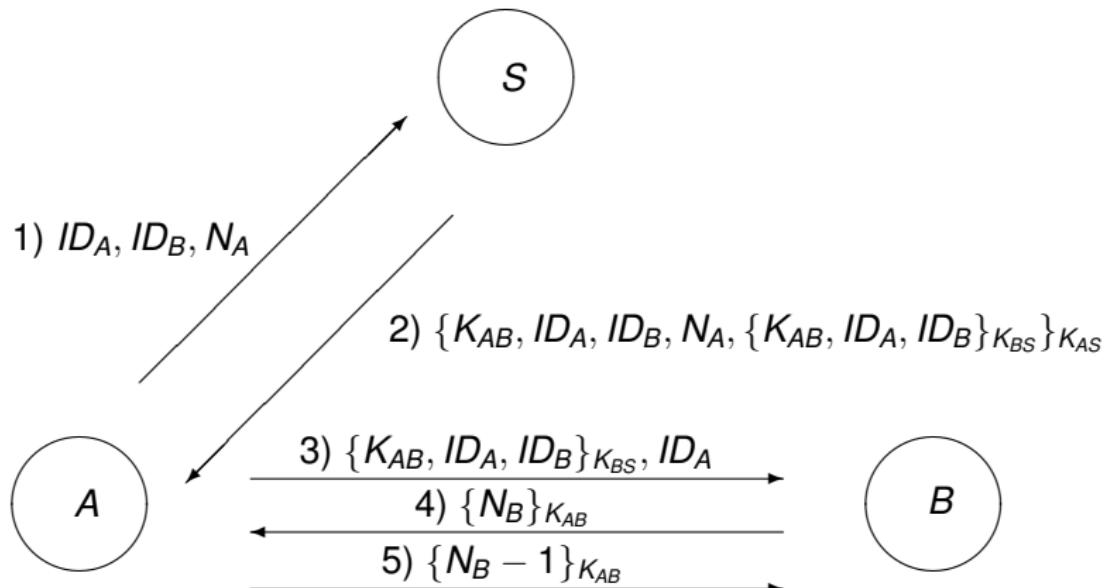
Description

- ▶ Published by Needham and Schroeder in 1978.
- ▶ A widely known key establishment protocol.
- ▶ Basis for many related protocols:
 - ▶ Example: Kerberos
- ▶ Vulnerable to *replay attacks* found by Denning and Sacco in 1981:
 - ▶ An attacker can replay old protocol messages s.t. an honest party will accept an old session key.

Notations

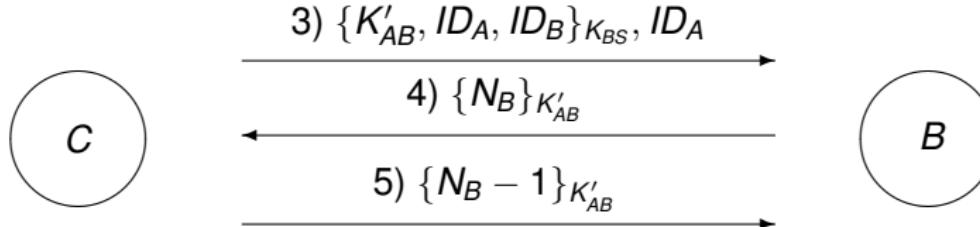
- ▶ **Parties:**
 - ▶ 2 parties A and B want to establish a shared secret key.
 - ▶ S is the TA.
- ▶ **Shared secret keys:**
 - ▶ A and S share the long-term key K_{AS} .
 - ▶ B and S share the long-term key K_{BS} .
 - ▶ New session key K_{AB} generated by S .
- ▶ **Nonces:**
 - ▶ N_A, N_B are randomly generated for one-time use.
 - ▶ $S \rightarrow A : M$ means that S sends a message M to A .
 - ▶ $\{M\}_K$ denotes the authenticated encryption of message M using the key K .

Protocol



Replay Attacks

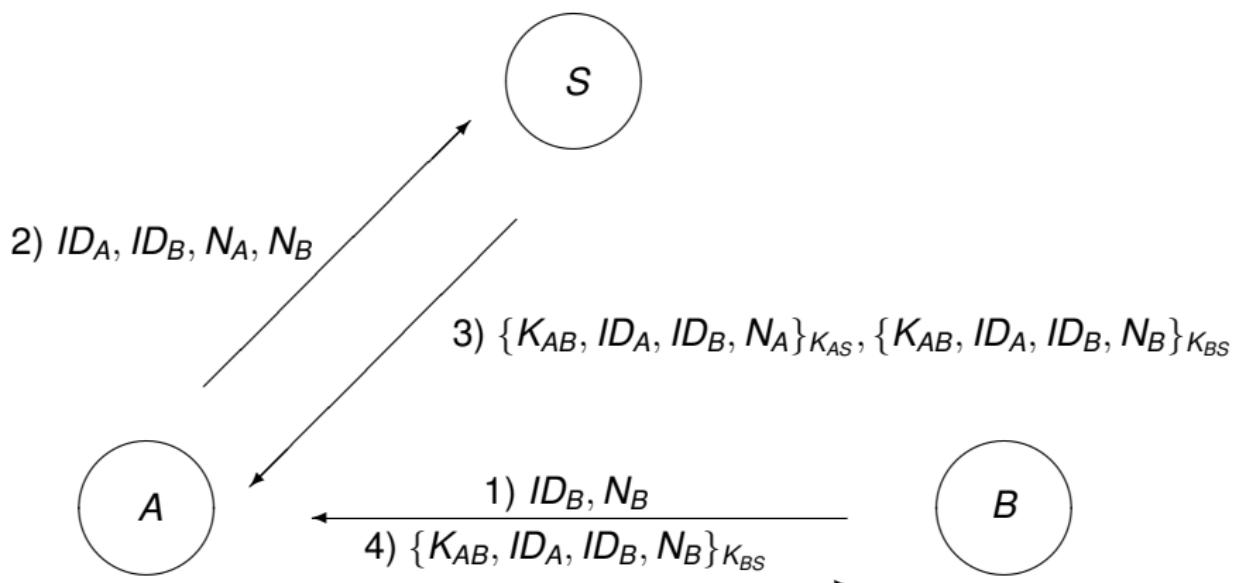
- ▶ Let an attacker C get a session key K'_{AB} previously established between A and B .
- ▶ C masquerades as A , and persuades B to use the old key K'_{AB} .



Freshness

- ▶ To defend against replay attacks, established key must be *fresh* (new) for each session.
- ▶ Mechanisms:
 - ▶ Random challenges (nonces).
 - ▶ Timestamps (string on the current time).
 - ▶ Counters (increased for each new message).
- ▶ Repaired protocol uses random challenges:
 - ▶ It can be adapted to use timestamps and counters.

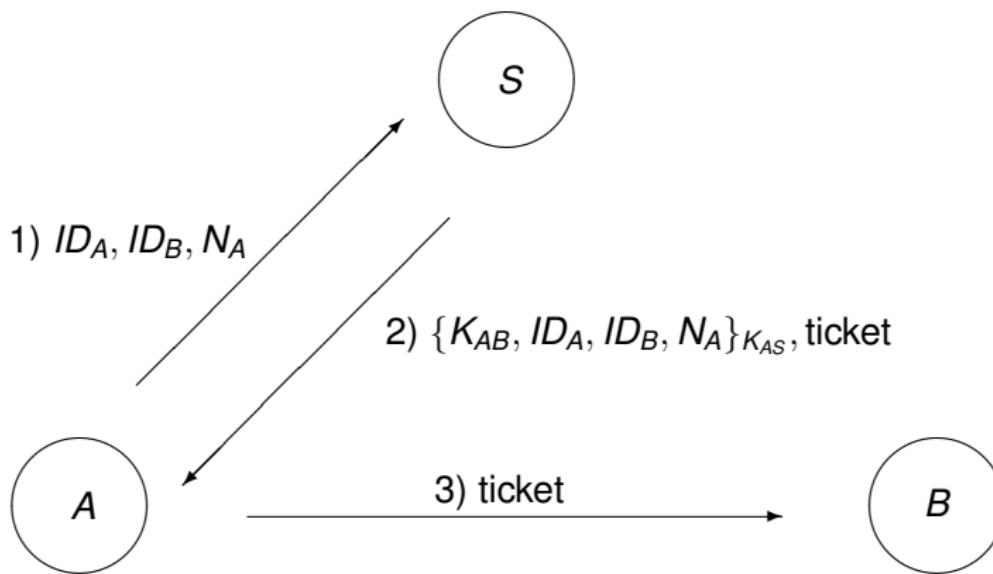
Repaired Protocol using Random Challenges



Tickets

- ▶ Another way to fix Needham-Schroeder protocol.
- ▶ The client A wishes to get access to the server B :
 - ▶ The authentication server S issues a *ticket* to allow A to obtain access.
- ▶ Ticket is $\{K_{AB}, ID_A, ID_B, T_B\}_{K_{BS}}$ where T_B is a timestamp (e.g. validity period).
- ▶ A gets ticket and uses it at any time while T_B is valid.

Repaired Protocol using Tickets



where $\text{ticket} = \{K_{AB}, ID_A, ID_B, T_B\}_{K_{BS}}$ for some validity period T_B

Description

- ▶ Developed at MIT as part of Project Athena.
- ▶ Several versions since its beginning in early 80s.
- ▶ Latest version (V5) released in 1995.
- ▶ **Standard:** RFC 4120 (2005).
- ▶ Default Windows domain authentication method from Windows 2000.

Goals

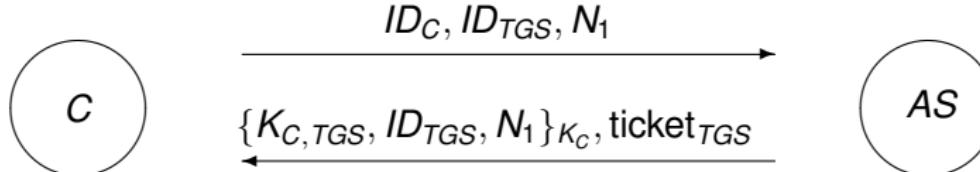
- ▶ Secure network authentication service in an insecure network environment.
- ▶ **Single sign-on (SSO) solution:**
 - ▶ Users only need to enter usernames and passwords once for a session.
- ▶ Providing access selectively for a number of different online services, using individual tickets.
- ▶ Establishing session keys to deliver confidentiality and integrity services for each service access.

3-Level Protocol

- ▶ **Level 1:** client C interacts with authentication server AS in order to obtain a ticket-granting ticket:
 - ▶ Happening once for a session (e.g. one day long).
 - ▶ C only authenticates once at the start of the session.
- ▶ **Level 2:** C interacts with ticket-granting server TGS in order to obtain a service-granting ticket:
 - ▶ Happening once for each server during the session.
- ▶ **Level 3:** C interacts with application server V in order to obtain a service:
 - ▶ Happening once for each time C requires service during the session.

Level 1

Interaction with the authentication server AS



where $ticket_{TGS} = \{K_{C,TGS}, ID_C, T_1\}_{K_{TGS}}$ for some validity period T_1

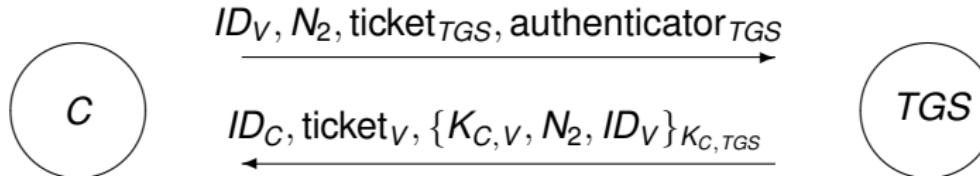
Result: C has a ticket-granting ticket that can be used to obtain different service-granting tickets.

Notes for Level 1

- ▶ K_C :
 - ▶ Symmetric key shared between AS and C.
 - ▶ Typically generated by the workstation of C from a password entered by C at logon time.
- ▶ $K_{C,TGS}$:
 - ▶ New symmetric key generated by AS and shared between TGS and C.
- ▶ N_1 :
 - ▶ Nonce used by C to check that key $K_{C,TGS}$ is fresh.
- ▶ K_{TGS} :
 - ▶ Long-term key shared between AS and TGS.

Level 2

Interaction with the ticket-granting server TGS



where $\text{ticket}_V = \{K_{C,V}, ID_C, T_2\}_{K_V}$ for some validity period T_2 and $\text{authenticator}_{TGS} = \{ID_C, TS_1\}_{K_{C,TGS}}$ for some timestamp TS_1

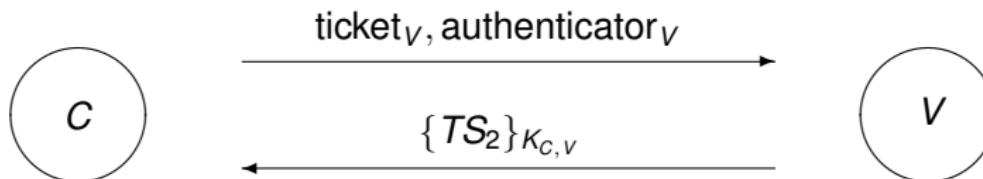
Result: C has a service-granting ticket that can be used to obtain access to a specific server.

Notes for Level 2

- ▶ ticket_{TGS} is the same than the one sent in Level 1.
- ▶ $K_{C,V}$:
 - ▶ Session key shared between V and C .
- ▶ N_2 :
 - ▶ Nonce used by C to check that key $K_{C,V}$ is fresh.
- ▶ TGS first gets $K_{C,TGS}$ from ticket_{TGS} , and then checks the fields in authenticator $_{TGS}$ are valid:
 - ▶ Checking that TS_1 is recent.
 - ▶ Checking that C is authorized to access V .
- ▶ In practice, both AS and TGS are the same machine.

Level 3

Interaction with the application server V



where $\text{authenticator}_V = \{ID_C, TS_2\}_{K_{C,V}}$ for some timestamp TS_2

Result: C has secure access to a specific server V.

Notes for Level 3

- ▶ ticket_V is the same than the one sent in Level 2.
- ▶ $K_{C,V}$, contained in ticket_V , is the same than the one sent in Level 2.
- ▶ Reply from V intended to provide mutual authentication:
 - ▶ C can check that it is using the right application server V .

Miscellaneous

- ▶ Above descriptions are simplified.
- ▶ **Timestamp:**
 - ▶ includes start and end times.
 - ▶ can be suggested by C in the last version of Kerberos (v5).
- ▶ **Realm:** a domain over which an authentication server has the authority to authenticate a user.
- ▶ **Flag:** used in tickets to indicate when and how tickets should be used.
- ▶ **Sequence number:** optional, initiated during the client-server exchange.
- ▶ **Subkey:** derived from the key $K_{C,V}$.

Limitations

- ▶ **Limited scalability:**
 - ▶ Even though different realms are supported, one realm needs to share a key with each other realm.
 - ▶ Kerberos best suited for corporate environments with shared trust.
 - ▶ Public-key variants exist.
- ▶ **Attack:**
 - ▶ Offline password guessing.
 - ▶ When the key K_C derived from a human memorable password.
- ▶ **Standard:**
 - ▶ Does not specify how to use the session key once it is established.

Lecture 17: Transport Layer Security Protocol Part 1

COSC362 Data and Network Security

Book 1: Chapter 17 – Book 2: Chapter 22

Spring Semester, 2021

Motivation

- ▶ TLS is the most widely used security protocol.
- ▶ TLS is used to secure communications with banks, online shops, email providers, etc.
- ▶ TLS uses most of the mainstream cryptographic algorithms.
- ▶ TLS is a very complex protocol.
- ▶ TLS has been subject of many attacks, and subsequent repairs.

Outline

History and Overview

TLS Record Protocol

TLS Handshake Protocol

Outline

History and Overview

TLS Record Protocol

TLS Handshake Protocol

History

- ▶ **1994:** Netscape Communications developed Secure Sockets Layer (SSL) 2.0:
 - ▶ It should no longer be used!
- ▶ **1995:** Netscape released SSL 3.0:
 - ▶ It should no longer be used!
- ▶ **1999:** RFC 2246 issued by IETF, documenting Transport Layer Security (TLS) 1.0, similar to SSL 3.0:
 - ▶ It should no longer be used!
- ▶ **2006:** RFC 4346 documenting TLS 1.1:
 - ▶ Fixing problems with non-random IVs and exploitation of padding error messages.
- ▶ **2008:** RFC 5246 documenting TLS 1.2:
 - ▶ Allowing the use of standard authenticated encryption rather than separating encryption and MAC.
- ▶ **2018:** RFC 8446 documenting TLS 1.3:
 - ▶ Separating key agreement and authentication algorithms for cipher suites.

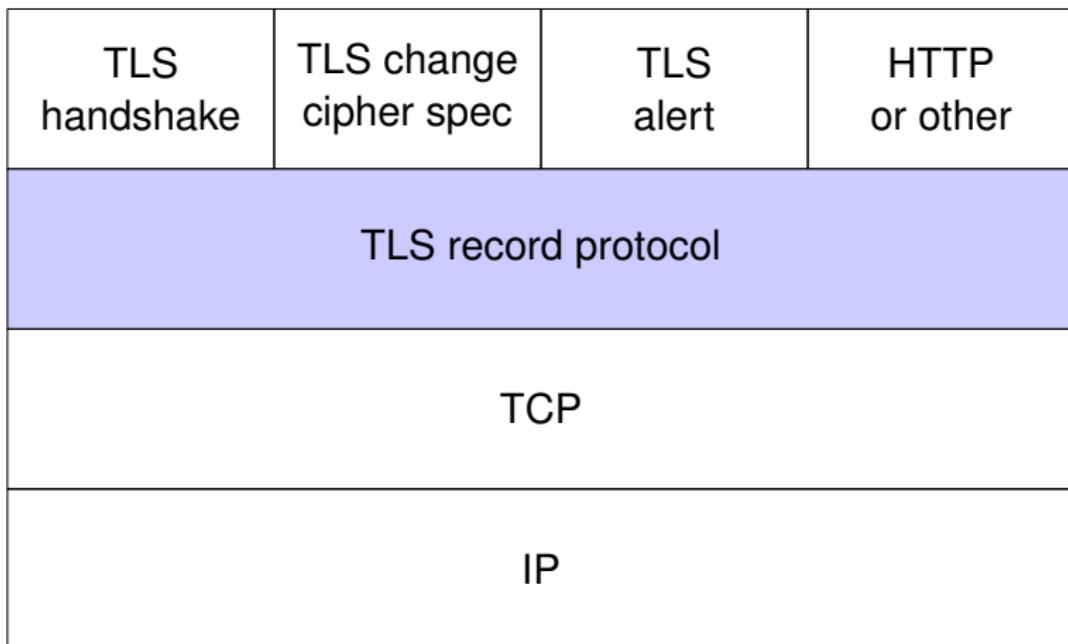
Applications

- ▶ Cryptographic services protocol based upon PKI and commonly used on the Internet.
- ▶ Often used to allow browsers to establish secure sessions with Web servers.
- ▶ Many other application areas.
- ▶ TLS runs primarily over TCP:
 - ▶ Variant DTLS runs over datagram protocols.

Architecture Overview

- ▶ Designed to secure reliable end-to-end services over TCP.
- ▶ 3 higher level protocols:
 - ▶ *TLS handshake protocol* to set up sessions.
 - ▶ *TLS alert protocol* to signal events, such as failures.
 - ▶ *TLS change cipher spec protocol* to change the cryptographic algorithms.
- ▶ TLS record protocol provides basic services to various higher level protocols.

Protocol Stack



Outline

History and Overview

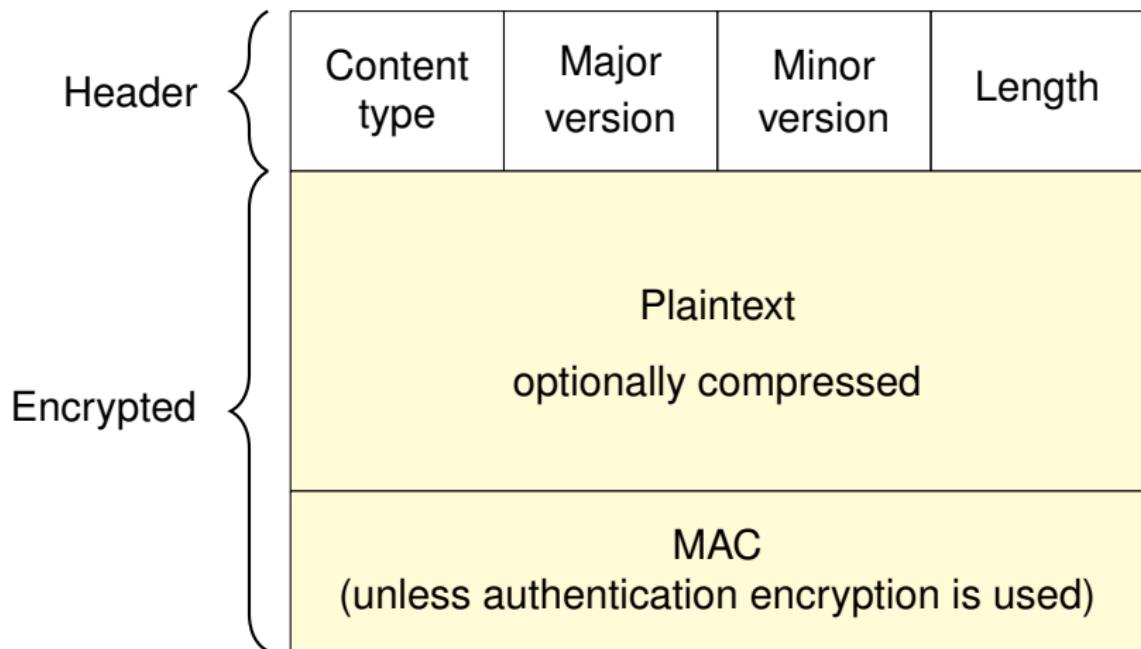
TLS Record Protocol

TLS Handshake Protocol

Overview

- ▶ TLS connection services:
 - ▶ **Message confidentiality:** ensuring that the message contents cannot be read in transit.
 - ▶ **Message integrity:** ensuring that the receiver can detect if a message is modified in transmission.
- ▶ Services possibly provided by a symmetric encryption algorithm and a MAC.
- ▶ From TLS 1.2, services provided with authenticated encryption modes (CCM, GCM).
- ▶ Handshake protocol establishes symmetric session keys to use with these mechanisms.

Format



Header

- ▶ Content type:
 - ▶ change-cipher-spec
 - ▶ alert
 - ▶ handshake
 - ▶ application-data
- ▶ Protocol version:
 - ▶ Major version: 3 for TLS
 - ▶ Minor version:
 - ▶ 1 for TLS 1.0
 - ▶ 2 for TLS 1.1
 - ▶ 3 for TLS 1.2
 - ▶ 4 for TLS 1.3
- ▶ Length: of the data, in octets.

Operation

- ▶ **Fragmentation:** each application layer message is fragmented into blocks of 2^{14} bytes or less.
- ▶ **Compression:**
 - ▶ Default compression algorithm is null in TLS 1.2 (thus optionally applied).
 - ▶ Removed in TLS 1.3.
- ▶ **Authenticated data:** consisting of the (compressed) data, header and an implicit record sequence number.
- ▶ **Plaintext:** compressed data and MAC (if present).
- ▶ **Session keys:** computed during handshake protocol, for either MAC and encryption algorithms, or authenticated encryption algorithm.
- ▶ **Specification:** encryption and MAC algorithms are specified in the negotiated cipher suite.

MAC Algorithm

- ▶ HMAC in all TLS versions using a negotiated hash function.
- ▶ SHA-2 allowed only from TLS 1.2.
- ▶ MD5 and SHA-1 discarded from TLS 1.3.

Encryption Algorithm

- ▶ Either a negotiated block cipher in CBC mode or a stream cipher.
- ▶ Most common block cipher is AES.
- ▶ 3DES and RC4 discarded in TLS 1.3.
- ▶ For block ciphers, padding is applied after MAC to make a multiple of the cipher block size.

Authenticated Encryption Algorithm

- ▶ Allowed instead of encryption and MAC from TLS 1.2.
- ▶ Only AES with either CCM or GCM modes in TLS 1.3.
- ▶ Authenticated additional data in the header and implicit record sequence number.

Outline

History and Overview

TLS Record Protocol

TLS Handshake Protocol

Purposes

- ▶ Negotiating the TLS version and cryptographic algorithms to be used.
- ▶ Establishing a shared session key for use in the record protocol.
- ▶ Authenticating the server, and optionally authenticating the client.
- ▶ Completing the session establishment.
- ▶ Variations with:
 - ▶ RSA
 - ▶ Diffie-Hellman
 - ▶ Pre-shared keys
 - ▶ Mutual authentication
 - ▶ Server-only (unilateral) authentication
- ▶ Simplified in TLS 1.3 (see later).

Phases

- ▶ **Phase 1:** initiating the logical connection and establishing its security capabilities.
- ▶ **Phases 2 and 3:** performing key exchange.
 - ▶ Messages and their contents depend on the handshake variant negotiated in Phase 1.
- ▶ **Phase 4:** completing the setting up of a secure connection.

Cipher Suites

- ▶ Specifying:
 - ▶ Public key algorithms used for key establishment.
 - ▶ Symmetric algorithms used for providing authenticated encryption and key computation.
- ▶ Over 300 standardised cipher suites:
 - ▶ Many are weak.
 - ▶ Many have been discarded in TLS 1.3.
- ▶ Big change in TLS 1.3:
 - ▶ All supported cipher suites must be Authenticated Encryption with Associated Data (AEAD).

Cipher Suite Example

TLS_RSA_WITH_3DES_EDE_CBC_SHA¹

- ▶ Mandatory in TLS 1.0 and 1.1.
- ▶ RSA used for key exchange, to encrypt a secret chosen by the client.
- ▶ 3DES (Encrypt Decrypt Encrypt) in CBC mode used for encryption, for data confidentiality.
- ▶ SHA-1 used for HMAC, for data integrity.

¹https://ciphersuite.info/cs/TLS_RSA_WITH_3DES_EDE_CBC_SHA/

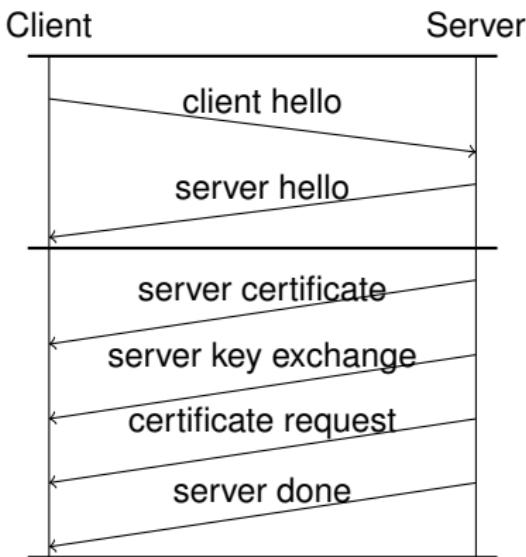
Handshake Algorithms

Algorithm	Description	TLS versions
DHE-DSS	DHE with Digital Signature Standard	1.2
DHE-RSA	Ephemeral Diffie-Hellman with RSA signatures	1.2 and 1.3
ECDHE-RSA	Elliptic curve DHE with RSA signatures	1.2 and 1.3
ECDHE-ECDSA	Elliptic curve DHE with elliptic curve Digital Signature Algorithm	1.2 and 1.3

Record Algorithms

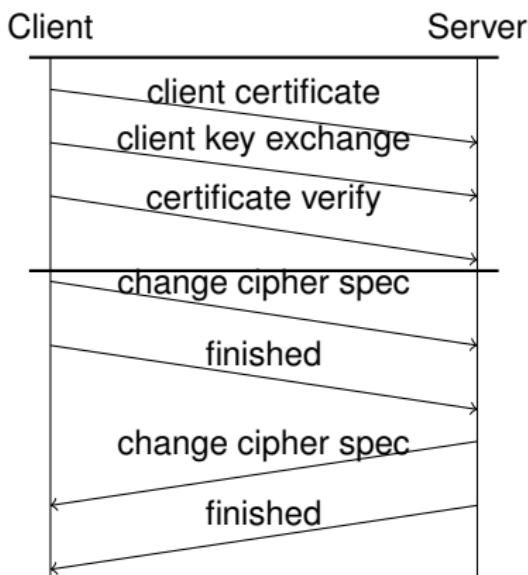
Algorithm	Description	TLS versions
AES-CBC-SHA256	AES in CBC mode with HMAC from SHA256	1.2
AES-GCM	AES with GCM mode	1.2 and 1.3
CHACHA20 -POLY1305	ChaCha stream cipher with Poly1305 MAC	1.2 and 1.3

Phases 1 and 2



- ▶ **Phase 1:** client and server negotiate version, cipher suite and compression, and exchange nonces.
- ▶ **Phase 2:** server sends certificate and key exchange message (if needed).

Phases 3 and 4



- ▶ **Phase 3:** client sends certificate and key exchange message.
- ▶ **Phase 4:** client and server start secure communications.
Finished messages include a check value (pseudorandom function) of all the previous messages.

Handshake Messages

- ▶ **Client hello:**
 - ▶ Stating the highest TLS version available.
 - ▶ Advertising cipher suites available to the client.
 - ▶ Sending the client's nonce N_C .
- ▶ **Server hello:**
 - ▶ Returning the selected version and cipher suite.
 - ▶ Sending the server's nonce N_S .
- ▶ **Server key exchange:** server's inputs to key exchange.
- ▶ **Client key exchange:** client's inputs to key exchange.
- ▶ **Change cipher suite:** switching to newly negotiated cipher suite for record layer.

Ephemeral Diffie-Hellman Handshake Variant

- ▶ **Server key exchange:** inputs are the Diffie-Hellman generator and group parameters, along with the server's ephemeral Diffie-Hellman value, all signed by the server.
- ▶ **Client key exchange:** inputs are client's ephemeral Diffie-Hellman value:
 - ▶ Optionally signed by the client if the client's certificate is used.
- ▶ Pre-master secret *pms* is the shared Diffie-Hellman secret (from key agreement).

RSA Handshake Variant

- ▶ **Server key exchange:** not required.
- ▶ **Client key exchange:** key transport of pre-master secret pms :
 - ▶ The client randomly selects the pre-master secret pms .
 - ▶ The client encrypts pms with the server's public key and sends the ciphertext to the server.
 - ▶ The server decrypts using its secret key to recover pms .

Session Key Generation

- ▶ Master secret ms defined using the pre-master secret pms (and a pseudorandom function):

$$ms = PRF(pms, \text{"master secret"}, N_C \| N_S)$$

- ▶ Key material generated as much as required by agreed cipher suite:

$$k = PRF(ms, \text{"key expansion"}, N_S \| N_C)$$

- ▶ Independent session keys are partitioned from k in each direction:

- ▶ A write key and a read key on each side.

- ▶ Depending on the agreed cipher suite, key material includes:

- ▶ Encryption key
 - ▶ MAC key
 - ▶ IV

Pseudorandom Function

- ▶ PRF built from HMAC with a specified hash function.
 - ▶ TLS 1.0 and 1.1: based on a combination of MD5 and SHA-1.
 - ▶ TLS 1.2: based on SHA-2.
- ▶ Example in TLS 1.2:

$$\begin{aligned} \text{PRF}(\text{key}, \text{label}, \text{nonce}) = & \quad \text{HMAC}(\text{key}, A(1) \parallel \text{label} \parallel \text{nonce}) \parallel \\ & \quad \text{HMAC}(\text{key}, A(2) \parallel \text{label} \parallel \text{nonce}) \parallel \\ & \quad \dots \end{aligned}$$

where $A(0) = \text{nonce}$, $A(i) = \text{HMAC}(\text{key}, A(i - 1))$.

HMAC uses a specified SHA-2 variant (e.g. SHA256) as its hash function.

Other Handshake Variants

- ▶ **Diffie-Hellman:** client and server use static/fixed Diffie-Hellman with certified keys:
 - ▶ When the client does not have a certificate (usual on the Internet), she uses an ephemeral Diffie-Hellman key.
- ▶ **Anonymous Diffie-Hellman:** the ephemeral Diffie-Hellman keys are not signed at all:
 - ▶ It only protects against passive eavesdropping.

Alert Protocol

- ▶ Handling connection by sending an alert message of various degrees of severity.
- ▶ Types:
 - ▶ Warning alerts
 - ▶ close_notify alerts
 - ▶ Fatal alerts
- ▶ Improperly handling alert messages leads to truncation attacks.

Forward Secrecy

- ▶ **Reminder:** compromise of a long-term key should not lead to compromise of session keys established before the long-term key is compromised.
- ▶ Diffie-Hellman key exchange achieves forward secrecy:
 - ▶ Exchange is authenticated using signatures from the long-term keys.
 - ▶ Diffie-Hellman-based cipher suites provide forward secrecy.
- ▶ RSA-based handshake does not offer forward secrecy but is currently used in many cipher suites:
 - ▶ TLS 1.3 does not allow static RSA.

Summary

- ▶ TLS consists of 3 protocols:
 - ▶ Handshake protocol
 - ▶ Record layer protocol
 - ▶ Alert protocol
- ▶ New 1.3 version has been rolled out as understanding of cryptography and potential attacks increase.
- ▶ TLS assumes reliable message delivery, provided by TCP.

Some time left?

Yes, then we keep on with Lecture 18!

Lecture 18: Transport Layer Security Protocol Part 2

COSC362 Data and Network Security

Book 1: Chapter 17 – Book 2: Chapter 22

Spring Semester, 2021

Motivation Reminder

- ▶ TLS is the most widely used security protocol.
- ▶ TLS is used to secure communications with banks, online shops, email providers, etc.
- ▶ TLS uses most of the mainstream cryptographic algorithms.
- ▶ TLS is a very complex protocol.
- ▶ TLS has been subject of many attacks, and subsequent repairs.

Outline

Summary of Lecture 17

Attacks on TLS

TLS 1.3

TLS Protocols

1. Handshake protocol
2. Record protocol
3. Alert protocol

Handshake Protocol

Process to start a communication session between a server and a client:

- ▶ Specify which version of TLS they will use (mostly TLS 1.2 or 1.3).
- ▶ Decide on which cipher suites they will use.
- ▶ Authenticate the identity of the server via the server's public key and the certificate authority's digital signature.
- ▶ Generate session keys in order to use symmetric encryption after the handshake is complete.

Steps with RSA Key Exchange

1. ***“client hello” message:*** TLS version and cipher suites supported by the client + N_C .
2. ***“server hello” message:*** certificate + chosen cipher suite + N_S .
3. ***Authentication:*** client checks certificate.
4. ***Premaster secret using key transport:***
 - ▶ chosen by client and encrypted using server's public key.
 - ▶ decrypted using server's private key.
5. ***Session keys:*** computed using PRF, on each side.
6. ***“client finished” message:*** encrypted with a session key.
7. ***“server finished” message:*** encrypted with a session key.

The handshake is completed and communication continues using the session keys.

Steps with Diffie-Hellman Key Exchange

1. ***“client hello” message:*** TLS version and cipher suites supported by the client + N_C .
2. ***“server hello” message:*** certificate + chosen cipher suite + N_S .
3. ***server’s signature:*** on N_C , N_S and server’s Diffie-Hellman parameters using server’s private key.
4. ***Signature verification:*** client checks signature + sends client’s Diffie-Hellman parameters.
5. ***Premaster secret using key agreement:*** using exchanged Diffie-Hellman parameters.
6. ***Session keys:*** computed using PRF, on each side.
7. ***“client finished” message:*** encrypted with a session key.
8. ***“server finished” message:*** encrypted with a session key.

The handshake is completed and communication continues using the session keys.

Record Protocol

Guarantee confidentiality and integrity of application data using the session keys created during the handshake:

- ▶ Divide outgoing messages into manageable blocks and re-assemble incoming messages.
- ▶ (optional) Compress outgoing blocks and decompress incoming blocks.
- ▶ Apply a MAC to outgoing messages and verify incoming messages using the MAC.
- ▶ Encrypt outgoing messages and decrypting incoming messages.

When the Record Protocol is complete, the outgoing encrypted data is passed down to the TCP layer for transport.

Outline

Summary of Lecture 17

Attacks on TLS

TLS 1.3

Backward Compatibility

Backward compatibility is a problem:

- ▶ SSL 3.0 was deprecated in 2015.
- ▶ End of life for TLS 1.0 and 1.1 only in 2020.
- ▶ TLS 1.2 is still the most widely supported:
 - ▶ Supported by 99.5% of websites (August 2021).
- ▶ TLS 1.3 is slowly adopted:
 - ▶ Supported by 47.8% of websites (August 2021).

Limitations

- ▶ Many practical attacks on TLS in the past few years.
- ▶ Many servers:
 - ▶ do not support the latest TLS versions.
 - ▶ are not protected against known attacks.
- ▶ Example:
 - ▶ Recent attacks show that RC4 is vulnerable.
 - ▶ RC4 is offered in TLS 1.2.
 - ▶ TLS 1.3 has discarded it, but not widely supported.
- ▶ SSL Pulse gives an up-to-date picture:
 - ▶ <https://www.ssllabs.com/ssl-pulse/>
- ▶ Good coverage of attacks is given on Matt Green's blog:
 - ▶ <http://blog.cryptographyengineering.com>

BEAST Attack

- ▶ Browser Exploit Against SSL/TLS (BEAST).
- ▶ Exploiting non-standard use of IV in CBC mode encryption:
 - ▶ IVs are chained from the previous ciphertexts.
 - ▶ Allowing the attacker to recover the plaintext byte by byte.
- ▶ Stages:
 - ▶ 2002: theoretical weakness
 - ▶ 2011: practical weakness
 - ▶ Only random IV from TLS 1.1
 - ▶ No longer considered as a realistic threat
- ▶ Most browsers implement a mitigation strategy:
 - ▶ Splitting the plaintext into first byte and remainder to force a randomized IV including a MAC computation.

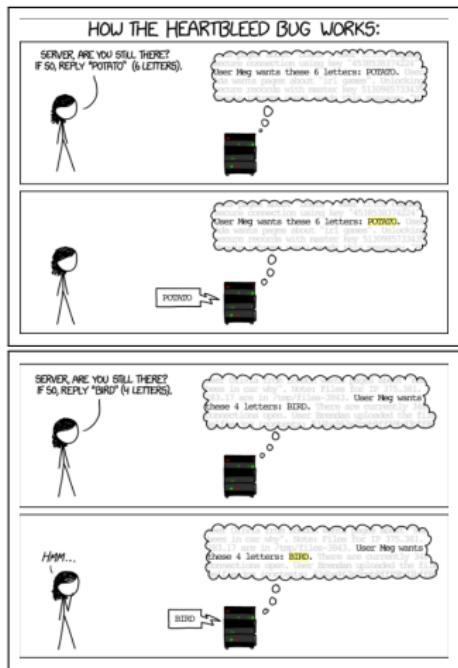
CRIME and BREACH Attacks

- ▶ Compression Ratio Info-leak Made Easy (CRIME).
- ▶ Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext (BREACH).
- ▶ Side channel attacks based on *compression*:
 - ▶ Different inputs result in different amounts of compression.
 - ▶ CRIME exploits compression in TLS.
 - ▶ BREACH exploits compression in HTTP.
- ▶ 2002: idea of the attack.
- ▶ Commonly recommended to switch off compression in TLS:
 - ▶ Compression not available in TLS 1.3.
- ▶ Switching off in HTTP results in big performance hit.

POODLE Attack

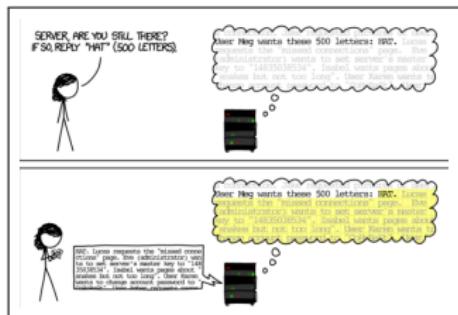
- ▶ **Padding oracle:** enabling an attacker to know if a message in a ciphertext was correctly padded.
- ▶ **2002:** theoretical idea.
 - ▶ Encryption in CBC mode can provide a padding oracle due to its error propagation properties.
 - ▶ Applied to TLS in a variety of attacks.
- ▶ **Main mitigation:** having a uniform error response, so that the attacker cannot distinguish padding errors from MAC errors.
- ▶ Padding Oracle On Downgraded Legacy Encryption (POODLE):
 - ▶ **2014:** attack is published.
 - ▶ Forcing downgrade to SSL 3.0, and then running padding oracle attack.

Heartbleed Bug



- ▶ Implementation error in toolkit OpenSSL.
- ▶ Result from improper input validation based on missing bounds check in *heartbeat* messages:
 - ▶ Allowing memory leakage from the server which is likely to include session keys and long-term keys.

Heartbleed Bug

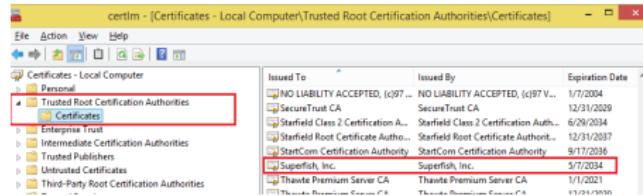


- ▶ 2014: error is discovered.
- ▶ Required updating of many server keys after the bug was fixed.
- ▶ Is it *reasonable* that big companies use a *free* software for securing important transactions?

<https://www.vox.com/2014/6/19/18076318/heartbleed>

MITM Attack

- ▶ Man-In-The-Middle (MITM):
 - ▶ 2015: attack is found.
 - ▶ Attack relying on issuing a new certificate and installing a root certificate in the browser.
- ▶ Superfish is a media company whose software was bundled with some Lenovo computers:
 - ▶ Users expressed concerns about scans of SSL-encrypted web traffic pre-installed on Lenovo machines.
 - ▶ US department of Homeland Security warned users to remove the root certificate.
- ▶ May 2015: Superfish closed.



<https://pkic.org/2015/02/20/lenovo-enables-man-in-the-middle-attacks-via-superfish-adware/>

Other Attacks

- ▶ STARTTLS command injection attack
- ▶ Sweet32 attack
- ▶ Triple Handshake attack
- ▶ RC4 attacks
- ▶ Lucky Thirteen attack (padding oracle attack)
- ▶ Renegotiation attack

Outline

Summary of Lecture 17

Attacks on TLS

TLS 1.3

History and Overview

- ▶ 2014: first draft version.
- ▶ January 2018: Internet draft version.
- ▶ August 2018: RFC 8446 is published.
- ▶ Browser support by default:
 - ▶ Draft version since Chrome 65 and final version (for outgoing connections) since Chrome 70.
 - ▶ Draft version in Firefox 52 and above (including Quantum) and final version since Firefox 63.
 - ▶ Since Microsoft Edge version 76, and Safari 12.1 on macOS 10.14.4.

Big Removals

The following items were suppressed in TLS 1.3:

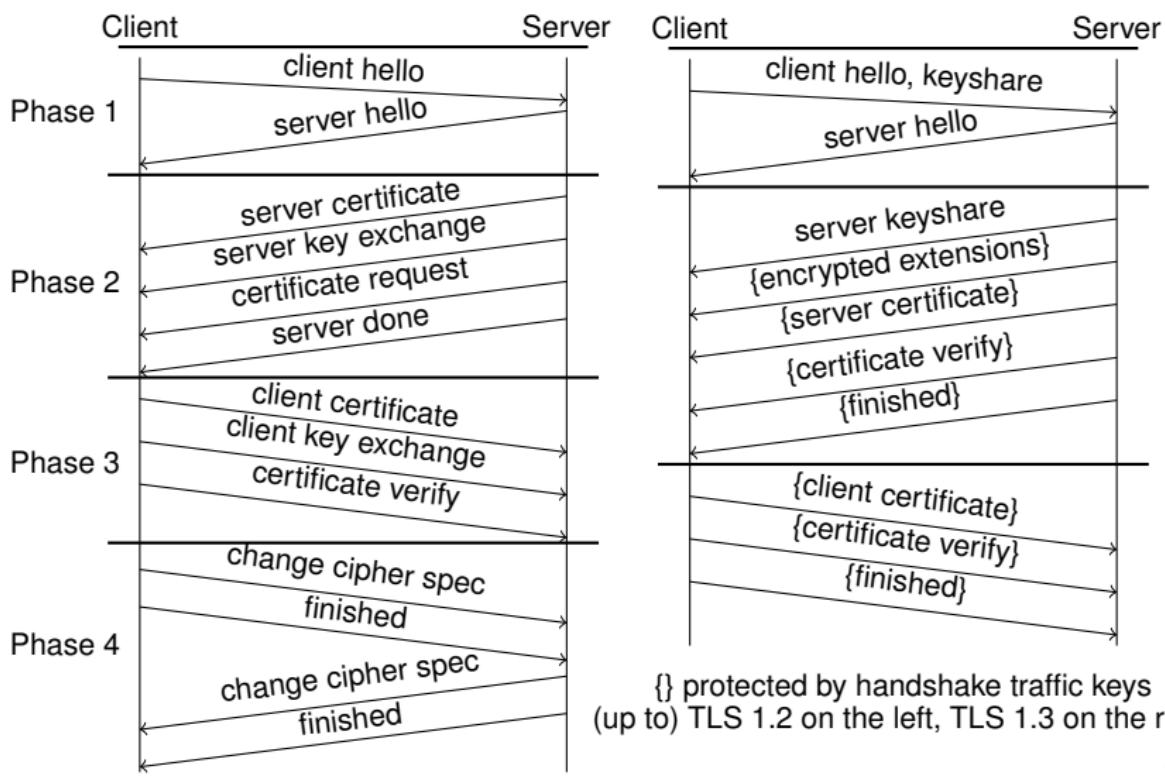
- ▶ Static RSA and Diffie-Hellman key exchange
- ▶ Renegotiation
- ▶ SSL negotiation
- ▶ DSA
- ▶ Data compression
- ▶ Non-AEAD cipher suites
- ▶ MD5 and SHA-224 hash functions
- ▶ Change Cipher Spec protocol

Big Supplements

The following properties/items were added in TLS 1.3:

- ▶ Only authenticated encryption with associated data (AEAD) cipher suites.
- ▶ Separating key agreement and authentication algorithms from cipher suites.
- ▶ Mandating perfect forward secrecy:
 - ▶ Using ephemeral keys during (EC) Diffie-Hellman key agreement.
- ▶ Encrypting the content type.
- ▶ Introducing 0-RTT mode (from pre-shared key).
- ▶ Introducing post-handshake client authentication.
- ▶ ChaCha20 stream cipher with Poly1305 MAC.

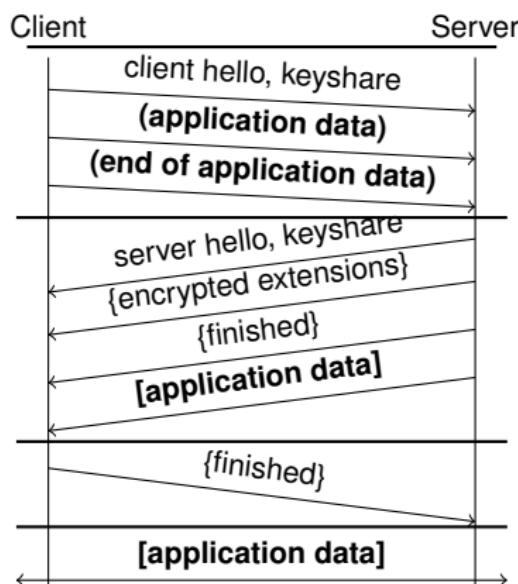
Handshake Comparison



0-RTT Overview

- ▶ 0-RTT based on a pre-shared key (resumption master secret):
 - ▶ Obtained when server and client complete a handshake for the first time.
 - ▶ Using that key when establishing a connection again at a later time:
 - ▶ So avoiding to perform the handshake a second time.
- ▶ Is this really 0-RTT (zero round trip time)?
 - ▶ TLS 1.3 handshake is 1-RTT instead of 2-RTT.
- ▶ **Limitations:**
 - ▶ Resumption data require no interaction from the server.
 - ▶ An attacker can capture encrypted 0-RTT data and re-send them to the server.
 - ▶ If the server is misconfigured, then it may accept replayed requests as valid:
 - ▶ Allowing the attacker to perform unsanctioned actions.

0-RTT



() protected by early data keys
{ } protected by handshake traffic keys
[] protected by further traffic keys

1-RTT and 0-RTT Comparison

- ▶ A user visits a website *not* for the first time:
 - ▶ Already visited it *recently*.
 - ▶ Resuming a previous connection (which was established using TLS).
- ▶ Full handshake in TLS1.3 using ephemeral DH key exchange (resulting in 1-RTT):
 1. $C \rightarrow S$: "client hello"
 2. $S \rightarrow C$: "server hello" + DH key share + encrypted extensions + certificate + key to verify the certificate + "finished"
 3. $C \rightarrow S$: DH key share + certificate + verification data + "finished"
 - ▶ Subsequent steps: C and S exchange encrypted application data using the session key obtained from the previous steps.

1-RTT and 0-RTT Comparison

- ▶ Session resumption process exists in TLS (mostly 1.2) but still 1-RTT:
 1. $C \rightarrow S$: "client hello" + DH key share + pre-shared key (obtained from a previous connection after a handshake was completed)
 2. $S \rightarrow C$: "server hello" + DH key share + pre-shared key + encrypted extensions + "finished" (no certificate and verification key anymore)
 3. $C \rightarrow S$: "finished" (no certificate and verification data anymore)
- ▶ Subsequent steps: C and S exchange encrypted application data using the session key obtained from the previous steps.

1-RTT and 0-RTT Comparison

- ▶ With 0-RTT, the process is shortened:
 1. $C \rightarrow S$: "client hello" + DH key share + pre-shared key (obtained from a previous connection after a handshake was completed) + early data + "end of early data" (that is an alert)
 2. $S \rightarrow C$: "server hello" + DH key share + pre-shared key + encrypted extensions + application data + "finished" (no certificate and verification data anymore)
 3. $C \rightarrow S$: application data + "finished"
- ▶ Subsequent steps: C and S exchange *more* encrypted application data using the session key obtained from the previous steps.

Example

- ▶ C and S share a pre-shared key psk from a previous session/connection/handshake.
- ▶ C uses psk to encrypt everything after "client hello" in 0-RTT.
- ▶ While psk was created from a previous handshake, it was not created specifically for that previous handshake:
 - ▶ It was rather created for future use, when C would resume a connection.
- ▶ S has also psk and can thus decrypt what C sent.
- ▶ S replies by using the key used for the previous handshake.
- ▶ The DH key shares will serve to create a fresh session key for the rest of the connection (encrypting application data).

Security Summary

- ▶ Different kinds of attacks:
 - ▶ Implementation errors
 - ▶ Poor choice of cryptographic primitives
 - ▶ Flaws in protocol
- ▶ Backward compatibility is a problem:
 - ▶ Downgrade attacks
- ▶ Several examples of the principle that “attacks only get better” over time.
- ▶ Complexity is a major problem:
 - ▶ TLS 1.3 removes many cipher suites and protocol options.
 - ▶ TLS 1.3 simplifies the handshake protocol.
 - ▶ TLS 1.3 adds new features (e.g. 0-RTT mode) which present new challenges.

Lecture 19: IPsec and VPN

COSC362 Data and Network Security

Book 1: Chapter 20 – Book 2: Chapters 9 and 22

Spring Semester, 2021

Motivation

- ▶ IP security (IPsec) is a framework for ensuring secure communications over IP (internet protocol) networks.
- ▶ Security services similar as TLS, but at a lower layer in the communication protocol stack.
- ▶ Security added to IPv4 and IPv6.
- ▶ Virtual private networks (VPNs) extend a private network across a public network and enable secure communication over the latter.

Outline

- IP Layer Security
 - Architectures
 - Protocols
 - Modes

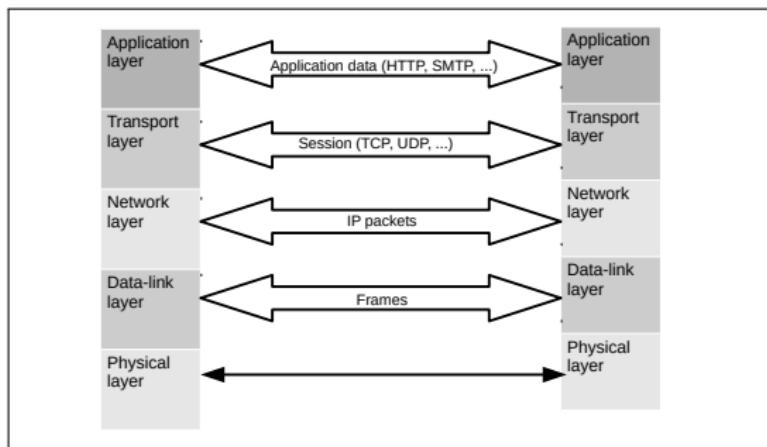
- Virtual Private Networks
 - IPsec Virtual Private Networks

Outline

IP Layer Security
Architectures
Protocols
Modes

Virtual Private Networks
IPsec Virtual Private Networks

Cryptography in the TCP/IP Stack



- ▶ **Application layer security:** SSH, S-MIME, PGP, etc.
- ▶ **Transport layer security:** SSL, TLS.
- ▶ **Network layer security:** IPsec.
- ▶ **Data-link layer security:** WEP, WPA, WPA2, etc.

Introduction

- ▶ **Standard:** RFCs 4301-4305 (2005) with cryptographic algorithms updated in subsequent RFCs.
- ▶ Providing protection for any higher layer protocol, including arbitrary TCP and UDP sessions.
- ▶ Using encryption, authentication and key management algorithms.
- ▶ Commonly used to provide virtual private networks (VPNs).
- ▶ Providing a security architecture for both:
 - ▶ **IPv4:** RFC 791 (1981)
 - ▶ **IPv6:** RFC 8200 (2017)

Security Services

- ▶ ***Message confidentiality:*** Protecting against unauthorized data disclosure:
 - ▶ By using encryption mechanisms.
- ▶ ***Message integrity:*** Determining if data has been changed (either intentionally or unintentionally):
 - ▶ By using message authentication codes (MACs).
- ▶ ***Limited traffic analysis protection:*** Possibly difficult to know which parties are communicating, how often, or how much data is being sent when monitoring network traffic:
 - ▶ By concealing IP datagram details such as source and destination addresses.

Security Services

- ▶ ***Message replay protection:*** Data not delivered multiple times, and not delivered badly out of order.
- ▶ ***Peer authentication:*** Ensuring network traffic to be sent from the expected host:
 - ▶ Each IPsec endpoint confirms its identity of the other IPsec endpoint with which it wishes to communicate.

Gateway-to-Gateway Architecture

- ▶ Providing secure communications between 2 networks.
- ▶ Network traffic routed through IPsec connection, protecting it appropriately.
- ▶ Only protecting data between the 2 gateways.
- ▶ Often used when connecting 2 secured networks:
 - ▶ **Example:** Linking a branch office to headquarters over the Internet.
- ▶ Less costly than private wide area network (WAN) circuits.

Host-to-Gateway Architecture

- ▶ Commonly used to provide secure remote access:
 - ▶ **Example:** An organization deploys a VPN gateway onto its network.
- ▶ Each remote access user establishes a VPN connection between the local computer (host) and the gateway.
- ▶ The VPN gateway may be either a dedicated device or part of another network device.
- ▶ Often used when connecting hosts on unsecured networks to resources on secured networks.

Host-to-Host Architecture

- ▶ Typically used for special purpose needs:
 - ▶ **Example:** System administrators performing remote management of a single server.
- ▶ Providing protection for data throughout its transit (end-to-end).
- ▶ Resource-intensive to implement and maintain in terms of user and host management.
- ▶ All user systems and servers participating in VPNs need to have VPN software installed and/or configured.
- ▶ Key management through a manual process.

Types

- ▶ *Encapsulating security payload (ESP)*: Providing confidentiality, authentication, integrity and replay protection.
- ▶ *Authentication header (AH)*: Providing authentication, integrity and replay protection, but NOT confidentiality:
 - ▶ AH is now deprecated.
- ▶ *Internet key exchange (IKE)*: Negotiating, creating and managing session keys in *security associations* (SAs).

IPsec Connection Setup

- ▶ Key exchange using IKEv2 protocol:
 - ▶ Standard: RFC 7296 (2014).
- ▶ IKEv2 uses a Diffie-Hellman protocol authenticated using signatures with public keys in X.509 certificates.
- ▶ Including *cookies* to mitigate denial-of-service (DoS) attacks:
 - ▶ Providing *Proof of Reachability* before any expensive cryptographic processing is completed.

Using cookies

RFC 7296 Section 2.6:

- ▶ Mechanism to mitigate the DoS attack called *stateless cookie*.
- ▶ When the server is under load, the initial request is responded with a calculated *stateless cookie*:
 - ▶ a value that can be re-calculated based on values in the initial request *without* storing responder-side state.
- ▶ The initial request is then expected to repeat, this time including the stateless cookie.

RFC 7296 Section 3:

- ▶ Addition of a *Proof of Work*:
 - ▶ by calculating a pre-image for a partial hash value.
- ▶ Setting an upper bound determined by the attacker's CPU to the number of negotiations it can initiate in a unit of time.

Security Associations

- ▶ Containing information needed by an IPsec endpoint to support an IPsec connection.
- ▶ Possibly including cryptographic keys and algorithms, key lifetimes, security parameter index (SPI), security protocol identifier (ESP and/or AH).
- ▶ SPI included in IPsec header to associate a packet with the appropriate SA.
- ▶ Telling the endpoint how to process inbound IPsec packets and/or how to generate outbound packets.
- ▶ **Unidirectional:** One SA for each direction of connection.
- ▶ IKEv2 to establish keys used in SAs.

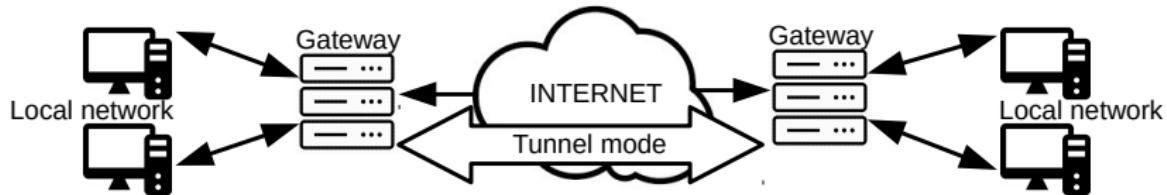
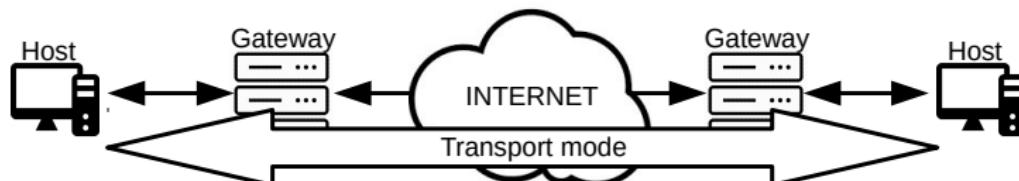
Cryptographic Suites

- ▶ Similar to TLS cipher suites:
 - ▶ Several standardised cryptographic suites, incorporating both public key and symmetric key algorithms.
 - ▶ Specific groups are available for Diffie-Hellman (in finite fields and on elliptic curves).
 - ▶ 3DES and AES used for encryption, either in CBC or GCM mode.
 - ▶ HMAC or CMAC (variant) used for integrity if GCM mode is not used.

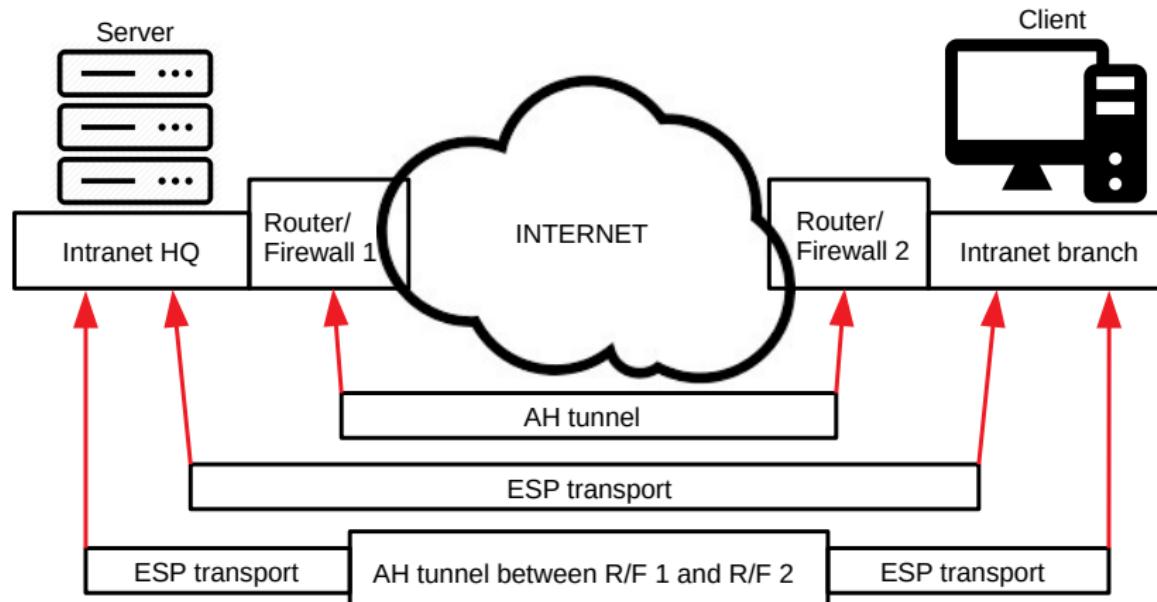
Modes of Operation

- ▶ Each protocol (either ESH or AH) can operate in either *transport* or *tunnel* mode.
- ▶ *Transport mode*: Maintaining IP header of the original packet and protecting the payload:
 - ▶ Generally used in host-to-host architectures.
- ▶ *Tunnel mode*: Encapsulating the original packet into a new one, and letting the payload be the original packet:
 - ▶ Generally used in gateway-to-gateway and host-to-gateway architectures.

Mode Overview



Example

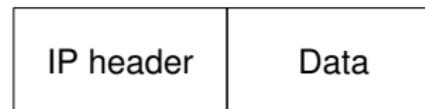


Components

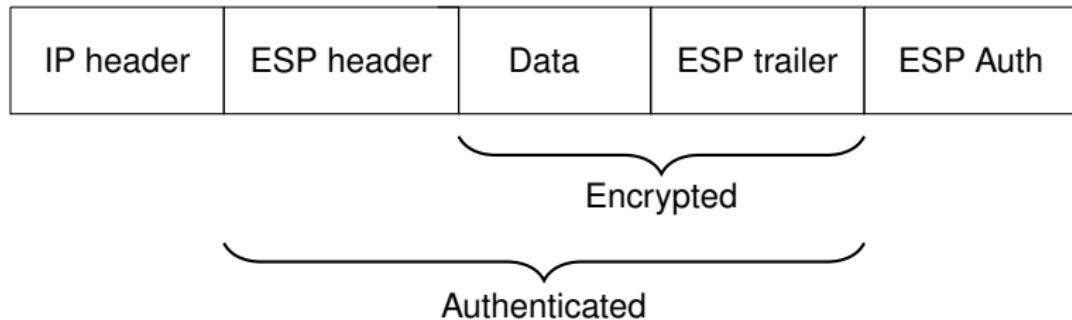
- ▶ ***ESP header:*** Containing the SPI identifying the SA and sequence numbers.
- ▶ ***ESP trailer:*** Containing padding and its length, and possibly including extra padding to enhance traffic flow confidentiality.
- ▶ ***ESP auth:*** Containing MAC of the encrypted data and ESP header:
 - ▶ Possibly not required if an authenticated encryption mode is used.

Transport Mode ESP

- ▶ Original IP packet:



- ▶ IP packet protected by transport mode ESP:



Pictures for IPv4 (slight differences for IPv6)

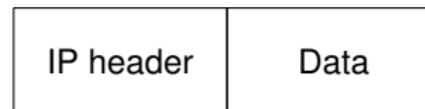
Transport Mode ESP

Outbound packet processing:

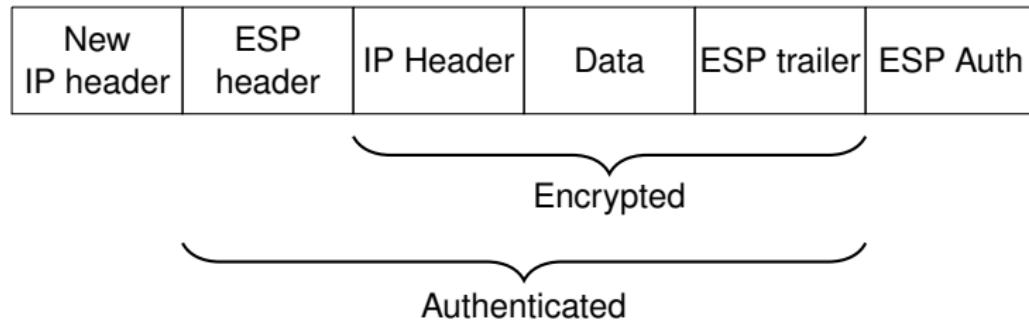
- ▶ Data after original IP header:
 - ▶ Padded by adding an *ESP trailer*.
 - ▶ Encrypted using symmetric cipher and key agreed in the SA.
- ▶ *ESP header* prepended.
- ▶ If SA uses the authentication service, then:
 - ▶ ESP MAC calculated over the data prepared so far and appended.
- ▶ Original IP header prepended BUT some fields must be changed:
 - ▶ Protocol field changed from TCP to ESP.
 - ▶ Total length field changed to reflect the addition of ESP header.
 - ▶ Checksums recalculated.

Tunnel Mode ESP

- ▶ Original IP packet:



- ▶ IP packet protected by tunnel mode ESP:



Pictures for IPv4 (slight differences for IPv6)

Tunnel Mode ESP

Outbound packet processing:

- ▶ Entire original packet:
 - ▶ Padded by adding an *ESP trailer*.
 - ▶ Encrypted using symmetric cipher and key agreed in the SA.
- ▶ *ESP header* prepended.
- ▶ If SA uses the authentication service, then:
 - ▶ ESP MAC calculated over the data prepared so far and appended.
- ▶ A new outer IP header prepended:
 - ▶ Inner IP header of original IP packet carrying the *ultimate* source and destination addresses.
 - ▶ Outer IP header may contain distinct IP addresses (e.g. addresses of security gateways).
 - ▶ Outer IP header protocol field set to ESP.

Security

- ▶ Active attacks exist for *encryption-only* mode of ESP protocol:
 - ▶ Providing encryption without integrity is known to be insecure.
 - ▶ Unlike earlier IPsec versions, the 2005 version does not require implementations to support encryption-only modes, but still allows it.
- ▶ Attacks due to MAC-then-encrypt configurations:
 - ▶ AH applies encryption *after* MAC (*MAC-then-encrypt*).
 - ▶ ESP applies encryption *before* MAC (*encrypt-then-MAC*).

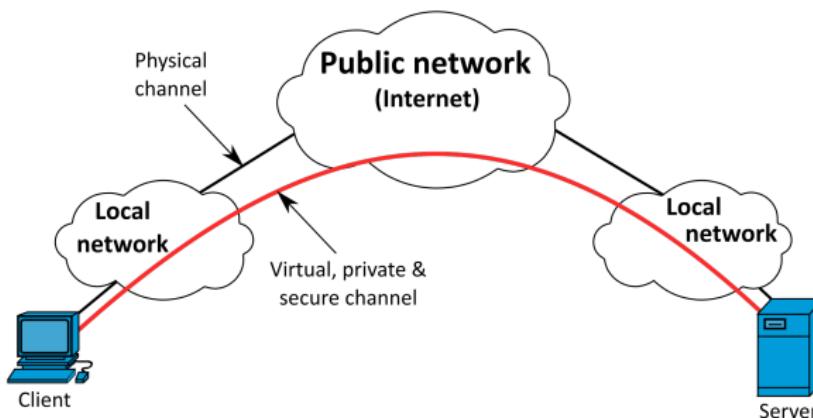
Outline

IP Layer Security
Architectures
Protocols
Modes

Virtual Private Networks
IPsec Virtual Private Networks

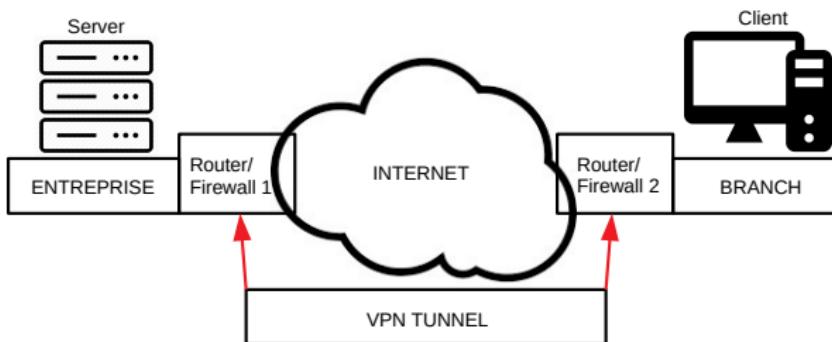
Introduction

- ▶ Providing a secure distributed network.
- ▶ Creating secure channels over the insecure Internet.
- ▶ **Types:**
 - ▶ Branch office interconnect (Intranet VPN)
 - ▶ Supplier/business partner access (Extranet VPN)
 - ▶ Remote access



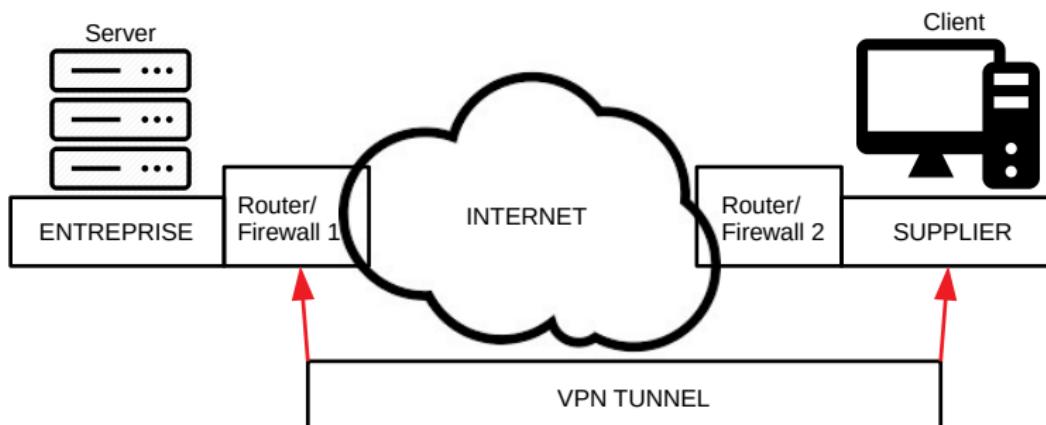
Branch Office Interconnect

- ▶ Establishing a VPN tunnel between router/firewall 1 and router/firewall 2:
 - ▶ Using AH to authenticate data from tunnel endpoints (routers/firewalls).
 - ▶ Using ESP to encrypt data over the Internet.
- ▶ Only routers/firewalls need to support IPsec:
 - ▶ No change to Intranet resources.



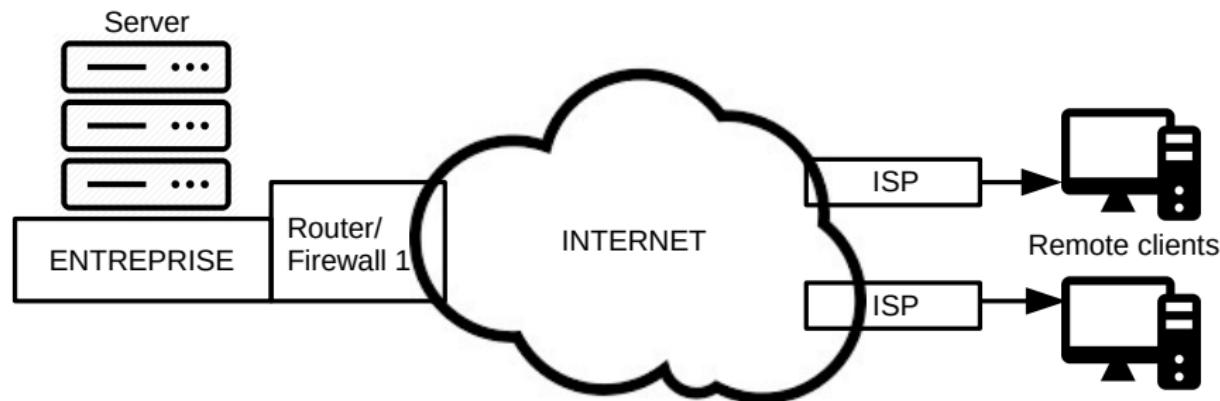
Supplier Network

- ▶ Supplier may not be part of the entreprise:
 - ▶ VPN extended to operate between router/firewall 1 and individual parts of supplier network.



Remote Access

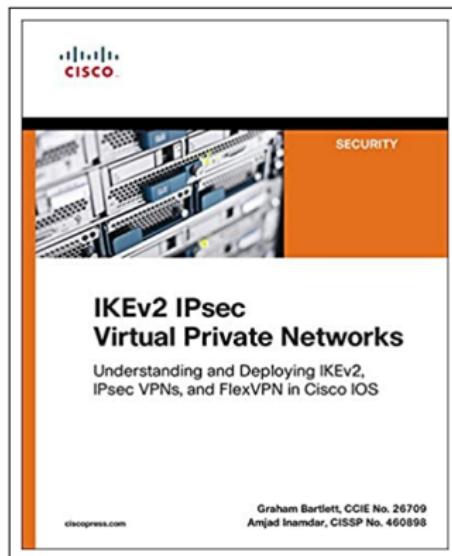
- ▶ Internet service providers (ISPs) can provide VPN services across the untrusted Internet.



└ Virtual Private Networks

 └ IPsec Virtual Private Networks

IKEv2 VPN



- ▶ Most recent and now very common in commercial equipment.
- ▶ Simple configuration and use with modern Windows and Linux systems, as well as mobile phones.

Lecture 20: Email Security

COSC362 Data and Network Security

Book 1: Chapter 19 – Book 2: Chapters 9 and 22

Spring Semester, 2021

Motivation

- ▶ Although TLS is the most widely used security protocol on the Internet, there are other important examples.
- ▶ Emails remain one of the most widely used forms of electronic communication, but are often sent without any security.

Outline

Email Security Requirements

Link Security

DomainKeys Identified Mail (DKIM)

STARTTLS

End-to-End Security

PGP

Secure/Multipurpose Internet Mail Extension (S/MIME)

Outline

Email Security Requirements

Link Security

DomainKeys Identified Mail (DKIM)

STARTTLS

End-to-End Security

PGP

Secure/Multipurpose Internet Mail Extension (S/MIME)

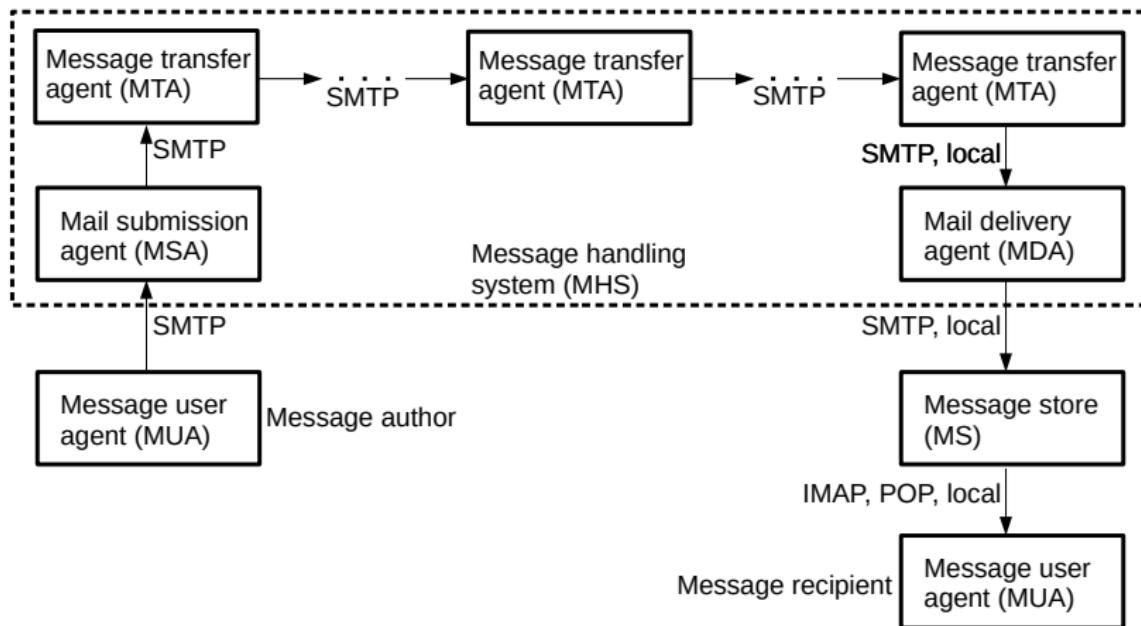
Protocols

- ▶ Single message transfer protocol (SMTP) is a mail transmission protocol to send an email from a source to a destination:
 - ▶ Standard: RFC 5321
- ▶ POP and IMAP are mail access protocols to allow a *message user agent* (MUA) to download an email from a *message transfer agent* (MTA).

Entities

- ▶ The *message user agent* (MUA) connects a client to a mail system:
 - ▶ Using SMTP to send a mail to a *message submission agent* (MSA).
 - ▶ Using POP or IMAP to retrieve the mail from the *message store* (MS).
- ▶ The *message handling system* (MHS) transfers a message from MSA to MS via one or more *message transfer agents* (MTAs).

Architecture



Security Threats

- ▶ Considering threats in the usual CIA categories:
 - ▶ Email content may require confidentiality and/or authentication.
 - ▶ Email service availability may be threatened.
- ▶ Metadata in the header is a significant source of information for an attacker.

Spam

- ▶ Unsolicited bulk email (UBE).
- ▶ A cheap form of advertising?
- ▶ Common vector for phishing attacks.
- ▶ Countermeasures typically use email filtering.
- ▶ Proposals to implement *proof of work*:
 - ▶ The email sender must solve a moderately hard puzzle in order to have the mail accepted into MHS.
 - ▶ **Example:** Hashcash.

Link Security and End-to-End Security

- ▶ Security may be provided between different agents in the mail system on a *link-by-link* basis, using protocols such as STARTTLS and DKIM.
- ▶ Alternatively, security may be provided from client to client (*end-to-end*), using protocols such as PGP and S/MIME.
- ▶ Both have their pros and cons.
- ▶ Ideally, both are used.

Outline

Email Security Requirements

Link Security

DomainKeys Identified Mail (DKIM)

STARTTLS

End-to-End Security

PGP

Secure/Multipurpose Internet Mail Extension (S/MIME)

└ Link Security

└ DomainKeys Identified Mail (DKIM)

Overview

- ▶ **Standard:** RFC 6376 (2011).
- ▶ Allowing the *sending mail domain* to sign an outgoing mail using RSA signatures.
- ▶ The *receiving domain* can verify the origin of mail.
- ▶ Widely used by prominent email providers, including Gmail.
- ▶ Helping to prevent email spoofing, and so to reduce spams and phishing.
- ▶ Public key of the sending domain retrieved using a domain name system (DNS).

└ Link Security

└ DomainKeys Identified Mail (DKIM)

Example

2048-bit RSA signature on a message, coded in base 64:

```
v=1; // Version
a=rsa-sha256; // Algorithm
c=relaxed/relaxed; // Header/body canonicalization (format)
d=gmail.com; // Domain claiming origin
s=20120113; // Selector subdividing namespace
h=mime-version:date:message-id:subject:from:to;
// Signed header fields
bh=NJjTF6QE7tvCE3fjCqEDuriGtvA2alydEz7wt4mn4EA=;
// Hash of the body part
b=h7aimB9ROItSF8RWWmd5MmJBQBR3gUo+w5L41UsMBSoDCjdqxmZQKyAhv
F7Cx E5+PzFLwQceVCYk3CzYuexyXkRNwuVw7A81NeJdDxA4b1SbFy8MuY5v
c+b2MPYQcP9v2iTli0m5N2AejzwSLyGvGU CtNSC7xQWHm0fTDC2LRY9b/xT
QzO6/608LSE59HW1gIf+AkWQae/ew41fyamu1QBoGFkgWy6ZMeOF+tzKtSy
RSc4FIcU1kcDuHkvQP jmw3hQN0gz+x4zfkb2wD9kyliWjw1tH3MM5FTwKzm
tAT/gDCtpCI7/HW6jeyx6HcevCjeFK+bkMy0nVa6oQc69o0MA==
// Signature
```

Overview

- ▶ Extending mail protocols SMTP, POP and IMAP to run over TLS connections.
- ▶ Providing link-by-link security, *but not end-to-end security*.
- ▶ *Opportunistic* use of TLS encryption security:
 - ▶ Using it if possible.
- ▶ **Standards:**
 - ▶ RFC 2595 for IMAP and POP3.
 - ▶ RFC 3207 for SMTP.
- ▶ Widely used by prominent email providers, including Gmail and Microsoft Outlook.
- ▶ Vulnerable to STRIPTLS attacks:
 - ▶ An attacker interrupts TLS negotiation, and connection falls back to plaintext transmission.

DKIM and STARTTLS Uptake

- ▶ **Recent survey:** Noticeable increase in uptake of DKIM and STARTTLS:
 - ▶ **Biblio:** Z. Durumeric et al., *Neither Snow nor Rain nor MITM... An Empirical Analysis of Email Delivery Security*, 2015.
- ▶ Gmail able to use STARTTLS for around 90% of both outgoing and incoming mails (Oct. 2021):
 - ▶ <https://transparencyreport.google.com/safer-email/overview>
- ▶ Gmail authentication using DKIM covered around 80% of incoming mails (2015).

Outline

Email Security Requirements

Link Security

DomainKeys Identified Mail (DKIM)

STARTTLS

End-to-End Security

PGP

Secure/Multipurpose Internet Mail Extension (S/MIME)

History



- ▶ Originally the product of one person, Phil Zimmermann.
- ▶ Subject to widely reported export restriction controversy.
- ▶ OpenPGP standard (RFC 4880) allows for interoperable implementations.
- ▶ GnuPG (GPG) is an open implementation.
- ▶ PGP corporation acquired by Symantec (2010).

Email Processing

- ▶ Protecting the mail message contents.
- ▶ **Hybrid encryption:**
 - ▶ A new random “session key” is generated for each message.
 - ▶ The session key is encrypted with the long-term public key of the recipient.
- ▶ Signing with either RSA or DSA.
- ▶ Compressing with Zip.
- ▶ Coding using radix-64 to ensure that binary strings can be sent in the mail body.

Encryption

- ▶ Session keys encrypted using asymmetric encryption:
 - ▶ OpenPGP requires the support of Elgamal encryption and recommends the support of RSA encryption.
- ▶ Message content encrypted using symmetric encryption:
 - ▶ OpenPGP requires the support of 3DES with 3 keys (168 bits in total) and recommends the support of AES-128 and CAST5 (other algorithms are also defined).
- ▶ Compression applied before encryption.
- ▶ Encryption can be applied independently of signing:
 - ▶ No requirement for authenticated encryption.

Signature

- ▶ Plaintext message is *optionally* signed with the sender's private key:
 - ▶ OpenPGP standard requires the support of RSA signatures and recommends the support of DSA signatures.
- ▶ RSA-signed messages are hashed with SHA1 (in the standard) or with SHA2 hash functions.

Web of Trust

- ▶ Users generate their own public/private key pairs.
- ▶ Public keys available on distributed key servers.
- ▶ Any PGP user can sign another user's public key, indicating their level of trust.
- ▶ Users can revoke their own key by signing a revocation certificate with the revoked key:
 - ▶ Users can also decide on the key expiration date when generating it.

Usability and Security

- ▶ Can we expect the average user to understand public key cryptography?
- ▶ Is it possible to design an interface that helps users to operate PGP correctly and safely?
 - ▶ <https://www.whitehatsec.com/blog/pgp-still-hard-to-use-after-16-years/>
 - ▶ A. Witten and J. D. Tygar, *Why Johnny can't encrypt: A Usability Evaluation of PGP 5.0*, Usenix Security Symposium, 1999
- ▶ Follow-up studies show that newer PGP versions are still hard to use.
- ▶ **Vulnerability:** EFail (2018).
 - ▶ Using a piece of HTML code to trick email users to reveal encrypted messages.

PGP Uptake

- ▶ Plugins available for many popular mail clients and for webmail interfaces:
 - ▶ Example: Mailvelope, a browser extension that enables the exchange of encrypted emails following the OpenPGP encryption standard.
- ▶ Around 100 keys added per day on SKS keyserver pool:
 - ▶ https://sks-keyservers.net/status/key_development.php
 - ▶ DNS records no longer maintained (due to GDPR).
- ▶ Growth rate remains linear over past several years.
- ▶ Around 60 000 keys in the *strong set* of keys with a trust path between any pair of keys.

OpenPGP Criticisms

- ▶ Outdated cryptographic algorithms still used:
 - ▶ SHA1, CAST5, etc.
- ▶ No support of SHA3 and authenticated encryption (e.g. GCM).
- ▶ Lots of metadata available to an eavesdropper:
 - ▶ File length
 - ▶ Used encryption algorithms
 - ▶ Recipient key identity

Overview

- ▶ Similar features to PGP:
 - ▶ Authentication, integrity, non-repudiation (signature) and confidentiality (encryption) of the message body carried in SMTP messages.
 - ▶ Different, not interoperable message format.
 - ▶ Sender's public key included with each message:
 - ▶ Used to verify the message.
 - ▶ X.509 certificates issued by CAs instead of Web of Trust:
 - ▶ NIST recommends S/MIME rather than PGP because of greater confidence in CA system.
 - ▶ Supported by most popular mail clients.

Authentication

1. The sender S creates a message m .
2. S generates a message digest $h(m)$ using SHA-256.
3. S signs $h(m)$ with her RSA private key, resulting into a signature s .
4. S appends s and m together and forwards them to the receiver R .
5. R verifies s (and gets $h(m)$) with the RSA public key of S .
6. R calculates a new digest for m and checks if it matches $h(m)$:
 - ▶ If there is a match then m accepted as authentic.

Guarantees

- ▶ RSA guarantee:
 - ▶ R assured that only the owner of the private key can generate s .
- ▶ SHA-256 guarantee:
 - ▶ R assured that no one else could generate a new digest that matches that $h(m)$, and a signature of m .

Confidentiality

1. The sender S creates a message m and a random 128-bit *content-encryption key* k for m only.
2. S encrypts m using k and AES-128 with CBC mode.
3. S encrypts k using RSA public key of the receiver R .
4. S sends both encrypted m and k to R .
5. R decrypts the encrypted k using her RSA private key.
6. R decrypts the encrypted m using k .

Guarantees

- ▶ Combining symmetric cryptography and public key cryptography allows to reduce encryption time.
- ▶ **Public key encryption:**
 - ▶ No session (content-encryption) key distribution needed.
 - ▶ Only R can recover k .
- ▶ **One-time mechanism:**
 - ▶ Symmetric encryption approach is strengthened.

Lecture 21: Malware and Cyber Attacks

COSC362 Data and Network Security

Book 2: Chapter 6

Spring Semester, 2021

Motivation

- ▶ Attacks have been recurrent.
- ▶ Scale extensively varies:
 - ▶ From one computer to million computers worldwide.
- ▶ Many attempts to expose, alter, disable, destroy, steal or gain unauthorized access to or make unauthorized use of an asset.
- ▶ Goals broadly vary:
 - ▶ Disabling the target computer or knocking it offline.
 - ▶ Getting access to the target computer's data and perhaps gaining admin privileges on it.

Outline

Attack Classification

Attack Methods

Social Engineering

Hacking and Cracking

Viruses and Worms

Trojan Horses

Denial of Service (DoS) Attacks

Rootkits

Blended Threats

Zero-Day Attacks

Bots and Botnets

Buffer Overflow

Outline

Attack Classification

Attack Methods

Social Engineering

Hacking and Cracking

Viruses and Worms

Trojan Horses

Denial of Service (DoS) Attacks

Rootkits

Blended Threats

Zero-Day Attacks

Bots and Botnets

Buffer Overflow

Methods

- ▶ **Social engineering:** Persuading somebody to do something.
- ▶ **Hacking or cracking:** Guessing, corrupting or stealing information.
- ▶ **Viruses and worms (malware):**
 - ▶ A virus propagates by inserting a copy of itself into and becoming part of another program.
 - ▶ Melissa, CryptoMix.
 - ▶ A worm replicates functional copies of itself but does not require a host program's help to propagate.
 - ▶ WannaCry, Code-Red, Nimda, Slammer.
- ▶ **Trojan horses:** A harmful piece of software that looks legitimate.
 - ▶ Backdoor trojan, downloader trojan, ransom trojan.

Methods

- ▶ *Network layer attacks:* IP spoofing (masquerading), sequence number prediction, TCP hijacking.
- ▶ *Web-based attacks:* Cross-site scripting, cooking poisoning, SQL injection.
- ▶ *Denial of service (DoS):*
 - ▶ Operating system attacks: Ping of Death, Tear Drop, Land, Snork.
 - ▶ Network attacks: SYN flood, TCP fin/rst, Smurf, Coke.
 - ▶ Distributed DoS: Cayosin, TCP Flood, Reflection.
- ▶ *Others...*

Outline

Attack Classification

Attack Methods

Social Engineering

Hacking and Cracking

Viruses and Worms

Trojan Horses

Denial of Service (DoS) Attacks

Rootkits

Blended Threats

Zero-Day Attacks

Bots and Botnets

Buffer Overflow

└ Attack Methods

 └ Social Engineering

Overview

- ▶ Persuading someone to disclose sensitive information:
 - ▶ Example: phishing attacks on bank customers.
- ▶ Persuading someone to run/install malicious or subverted software.
- ▶ Inviting someone to log into a bogus website:
 - ▶ Example: spoofed bank website.
- ▶ Impersonating a new employee who has forgotten user ID and/or password.
- ▶ Impersonating a technical support staff member and requesting a user login to “check” accounts.

Spear Phishing Attacks

- ▶ Generally an email appearing to be from an individual or business that users know.
- ▶ Looking for credit card and bank account numbers, passwords, and other financial information.
- ▶ Example: Westpac Australia (2004):
 - ▶ "Dear client of the Westpac Bank,
The recent cases of fraudulent use of clients accounts forced the Technical Services of the bank to update the software. We regret to acknowledge that some data on users accounts could be lost. The administration kindly asks you to follow the reference given below and to sign in to your online banking account:
<https://oIb.westpac.com.au/ib/default.asp>.
We are grateful for your cooperation."

└ Attack Methods

 └ Hacking and Cracking

Overview

- ▶ Password discovery: default passwords (“guest”, etc.).
- ▶ Password cracking tools, readily available from the Internet for a wide range of password protected systems:
 - ▶ UNIX password files, Word documents, ZIP files, Windows password files, etc.
- ▶ 15 GB of passwords:
[https://crackstation.net/
buy-crackstation-wordlist-password-cracking-dictionary.
htm](https://crackstation.net/buy-crackstation-wordlist-password-cracking-dictionary.htm)

└ Attack Methods

 └ Hacking and Cracking

Password Attacks

- ▶ Brute force attacks: few characters.
- ▶ Dictionary attacks: real-word passwords.
- ▶ Tools:
 - ▶ CRACK: www.pwcrack.com
 - ▶ L0phtcrack: www.l0phtcrack.com
 - ▶ John the Ripper: www.openwall.com/john
- ▶ One-time passwords (OTPs) are valuable!

Viruses

- ▶ Executable piece of code.
- ▶ Travelling and spreading by attaching itself to legitimate executable programs.
- ▶ Causing some unexpected and usually undesirable behaviour.
- ▶ Automatically spreading to other computer users:
 - ▶ **Example:** By transferring infected files via email attachments.

└ Attack Methods

 └ Viruses and Worms

Worms

- ▶ Computer program run independently.
- ▶ Propagating a complete working version of itself onto other hosts on a network:
 - ▶ Usually by exploiting software vulnerabilities in the target systems.

Overview

- ▶ Unlike viruses and worms, not infecting files and not propagating.
- ▶ Installing a trojan horse allows an attacker to access a user's machine remotely via the Internet.
- ▶ Components:
 - ▶ Client application run on the attacker's computer.
 - ▶ Server application run on the victim's computer.
- ▶ A program pretending to be benign but containing a malicious code.
- ▶ Normally waiting to be downloaded or installed by a user, and then executing attack:
 - ▶ Example: email attachment.
- ▶ Computers on a network are then scanned to locate any with a trojan installed, creating a *botnet*.

└ Attack Methods

 └ Trojan Horses

Zeus

- ▶ Stealing banking information by keystroke logging.
- ▶ Spreading through drive-by downloads and phishing schemes.
- ▶ Compromising thousands of accounts on websites of companies:
 - ▶ Examples: Bank of America, NASA, Oracle, Cisco, Amazon.
- ▶ Current botnet estimated to include million of compromised computers (3.6 million in USA).
- ▶ Currently the largest botnet on the Internet.

└ Attack Methods

└ Denial of Service (DoS) Attacks

Overview

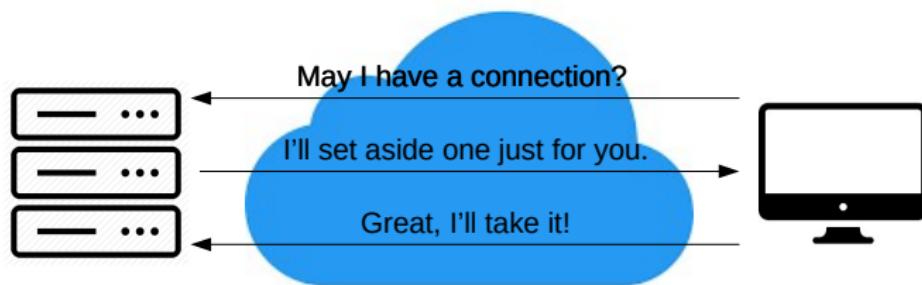
- ▶ **Intention:** Making network services unavailable to users (rather than gaining illegal access).
- ▶ Flooding attacks overload servers.
- ▶ **Examples:** Ping o' Death, SYN flood, ICMP redirect messages.
- ▶ Financial incentive and extortion.
- ▶ No magic solution:
 - ▶ Sharing services across different servers.
 - ▶ Using a properly configured firewall.

└ Attack Methods

└ Denial of Service (DoS) Attacks

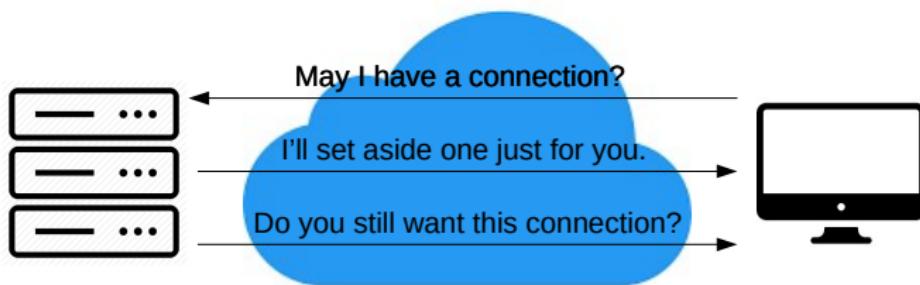
Normal TCP Connection Setup

TCP SYN-ACK Sequence:



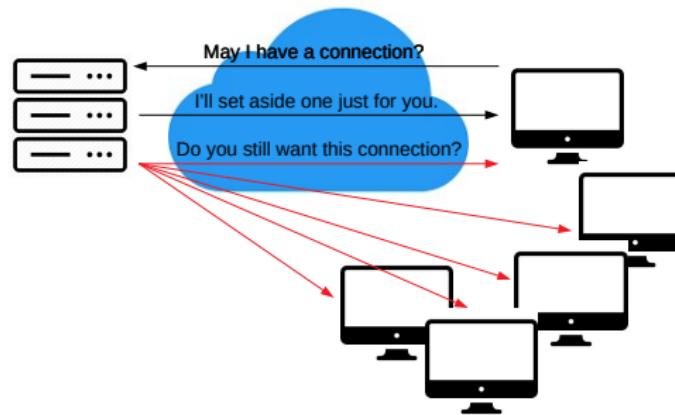
Abnormal TCP Connection Setup

TCP SYN-ACK Sequence:



Organised DoS Attack

TCP SYN-ACK Sequence:



- ▶ Over time, other requests will not be serviced.
- ▶ System locks up, does not really die (just impaired).

Overview

- ▶ Collection of programs that hackers use to mask intrusion and obtain admin access.
- ▶ An intruder installs a rootkit after obtaining user-level access:
 - ▶ By exploiting known vulnerability or cracking password.
- ▶ Collecting user IDs and passwords to other machines on the network:
 - ▶ Thus giving the hacker root/privilege access.

└ Attack Methods

 └ Rootkits

Utilities

- ▶ Monitoring traffic and keystrokes.
- ▶ Creating a “backdoor” into the system for hacker’s use.
- ▶ Altering log files.
- ▶ Attacking other machines on the network.
- ▶ Altering existing system tools to circumvent detection.

Remarks:

- ▶ Available for a number of operating systems.
- ▶ Increasingly difficult to detect on any network.

Overview

Software exploit that involves a combination of attacks against different vulnerabilities:

- ▶ Worms dropping parasitic viruses.
- ▶ Destructive trojan horses.
- ▶ Password stealers.
- ▶ Remote access trojans (RATs).
- ▶ Trojanised applications replacing legitimate system tools.
- ▶ Multiplatform attacks:
 - ▶ Payloads affecting multiple platforms.
 - ▶ Linux worms with drop.exe trojans.
- ▶ Advanced persistent threats (APTs).

└ Attack Methods

 └ Blended Threats

Remote Access Trojans (RATs)

- ▶ Malware threat previously used in attacks against energy sectors.
- ▶ Now aimed at organizations using/developing industrial applications and machines.
- ▶ Distributed new version of a RAT, called *Havex*:
 - ▶ Discovered in 2013 by F-Secure.
 - ▶ Hacking into websites of industrial control system (ICS) manufacturers and poisoning their software downloads.

Advanced Persistent Threats (APTs)

- ▶ Set of stealthy and continuous computer hacking processes:
 - ▶ Involving humans in real-time.
- ▶ Targeting organizations for business motives and nations for political motives.
- ▶ Requiring a high degree of covertness over a long period of time.
- ▶ Sophisticated techniques using malware to exploit vulnerabilities in systems.
- ▶ External command and control, continuously monitoring and extracting data off a specific target.
- ▶ **Examples:** Stuxnet, Duqu, Sandworm, BlackEnergy.

└ Attack Methods

 └ Zero-Day Attacks

Overview

- ▶ Taking advantage of software vulnerabilities for which there is no available fix.
- ▶ Taking advantage of flaws before software makers can fix them.
- ▶ Emphasizing the importance of safe configuration policies and good incident reporting systems.

└ Attack Methods

 └ Zero-Day Attacks

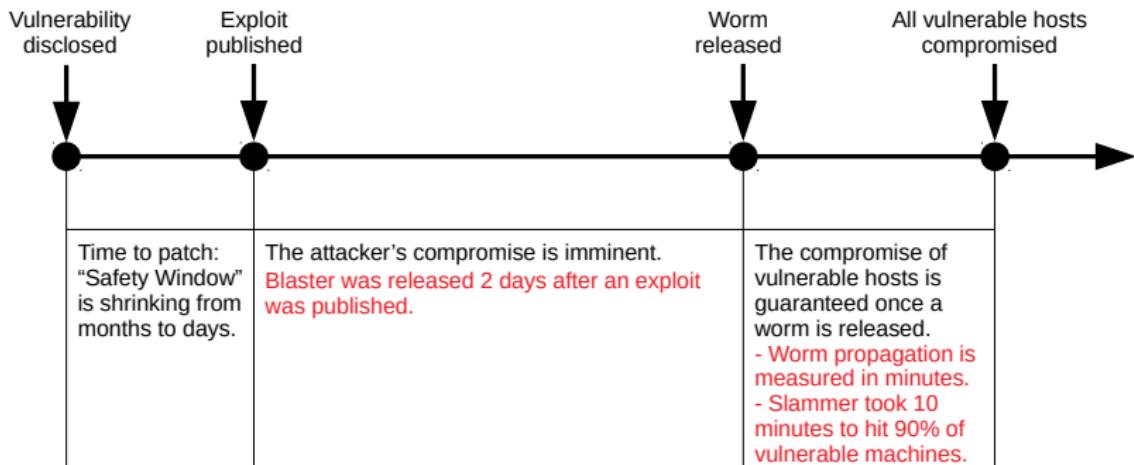
Examples

- ▶ Malicious hackers are getting faster at exploiting flaws.
- ▶ Blaster worm:
 - ▶ One of the most virulent ever.
 - ▶ Hitting the Internet barely one month after Microsoft released a patch for the flaw it exploited.
- ▶ Nachi worm:
 - ▶ A variant of Blaster worm.
 - ▶ Carrying a dangerous payload.
 - ▶ Hitting users less than a week later.
- ▶ Timelines are collapsing:
 - ▶ Only a matter of time before users see attacks against flaws not yet discovered or for which no patches are available.

└ Attack Methods

└ Zero-Day Attacks

Getting Closer



Overview

► Bot:

- ▶ Derived from the word “robot”.
- ▶ Also called *webcrawler*.
- ▶ Software agent interacting with other network services intended for people as if it were a real person.
- ▶ Typical use is gathering information.

► Botnet:

- ▶ Collection of software bots, running autonomously.
- ▶ Usually a collection of compromised machines running worms, trojans or backdoors.

Overview

- ▶ Technique used to gain remote execution on host.
- ▶ Taking advantage of inadequate buffer boundary checking in applications/services.
- ▶ Often involving overwriting return addresses on the stack.
- ▶ Involving sending executable code as binary data within the attack data stream:
 - ▶ Usually carefully crafted to be located at specific position within a buffer.
- ▶ **Example:** Heartbleed bug:
 - ▶ Bug in the OpenSSL's implementation of the SSL/TLS heartbeat extension.
 - ▶ When exploited, it leads to the leak of memory contents from the server to the client and from the client to the server.

Heartbleed Buffer Overflow

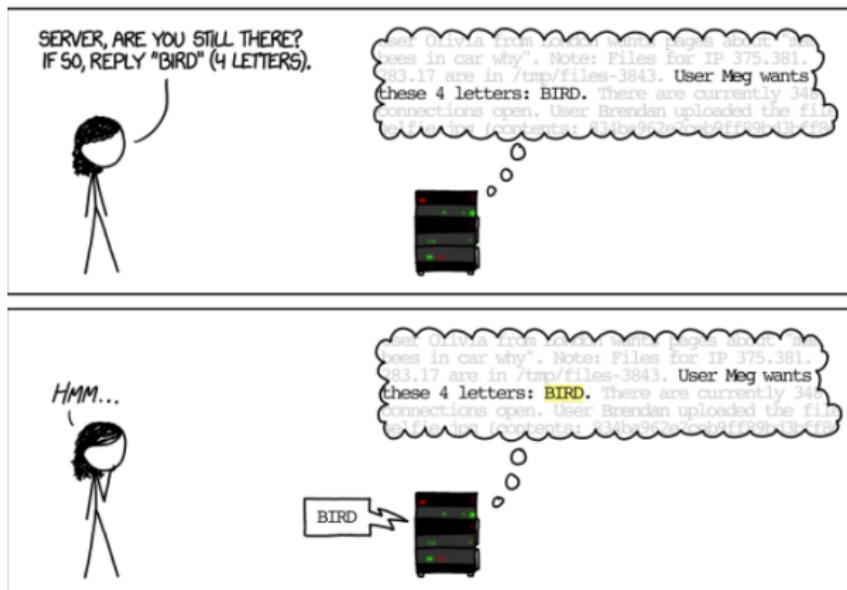
HOW THE HEARTBLEED BUG WORKS:



└ Attack Methods

└ Buffer Overflow

Heartbleed Buffer Overflow



└ Attack Methods

└ Buffer Overflow

Heartbleed Buffer Overflow



Heartbleed Attack Scale

- ▶ Well-known bug in SSL/TLS.
- ▶ Vulnerability exploited to access memory:
 - ▶ Secret cryptographic keys.
 - ▶ User names, passwords, their contents.
- ▶ The bug is public knowledge:
 - ▶ Supposed to exist at least 2 years before discovery.
- ▶ The highest volume of attacks:
 - ▶ Occurred when there were more than 300 000 attacks in one day.

└ Attack Methods

└ Buffer Overflow

In Real World

US hospital hack “exploited Heartbleed flaw”.



- ▶ www.bbc.com/news/technology-28867113
- ▶ 4.5 million healthcare patient data stolen because of delays in patching the 6 vulnerable SSL engines.
- ▶ Time between zero-day (i.e. Heartbleed release) and patch day was too long!