

# **02239 Data Security**

# Christian Damsgaard Jensen

Office: 322/225

Email: cdje@dtu.dk

Sebastian A. Mödersheim

Office: 321/018

Email: samo@dtu.dk

DTU Compute

Department of Applied Mathematics and Computer Science

A vibrant collage of mathematical symbols and numbers. It features a large purple integral symbol with 'a' and 'b' as limits, a red summation symbol with a factorial (!) at its base, a blue infinity symbol, a green square root symbol with '17', a red delta symbol, a blue theta symbol, a red plus sign, a blue omega symbol, a red equals sign, a blue brace, a red double arrow, a blue sigma symbol, a red exclamation mark, and a blue comma. The background is a light grey with faint, semi-transparent mathematical drawings like a grid and a circle.

# Course Objectives

*The objective of this course is to provide a **holistic introduction** to the **basic concepts** of computer security, which will provide the necessary basis for **ongoing scholarly studies** and **starting professional activities** in the area of computer security*

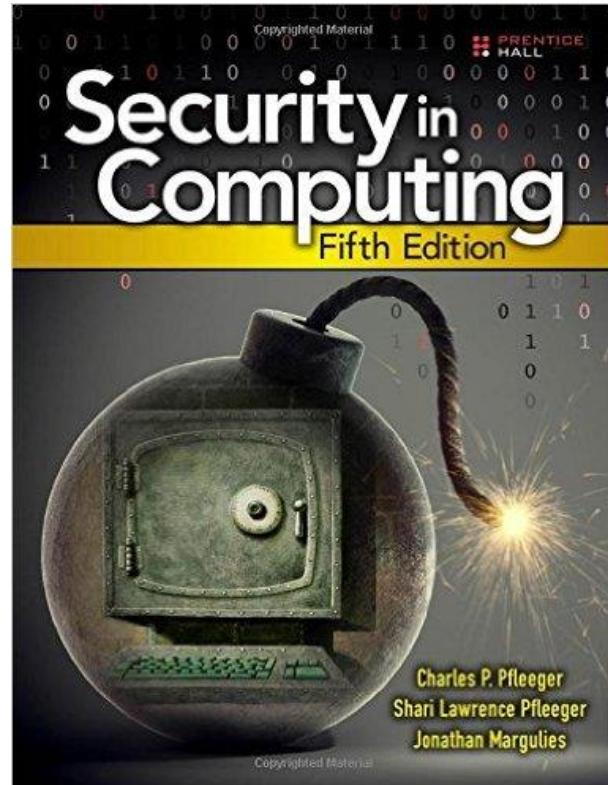
- No chain is stronger than its weakest link so all aspects that impact on security must be examined
- Our intention is to cover topics broadly, but not in great detail
  - Coverage should be sufficient to allow ongoing scholarly studies in computer security (possible courses listed on the next slide)
  - Coverage should be sufficient to allow a motivated engineer to learn more about computer security from self-study and on the job learning
- A few important topics are not covered at all
  - Some will be covered in later courses for those who continue

# Course Overview

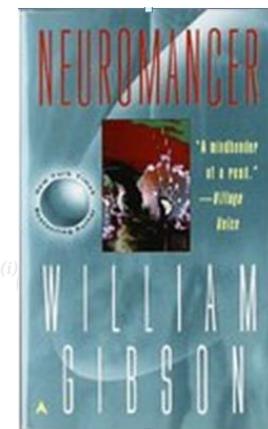
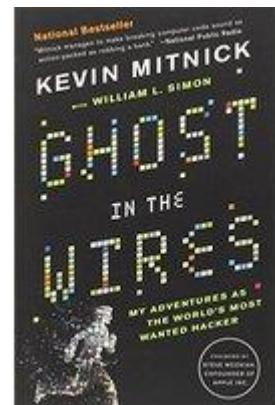
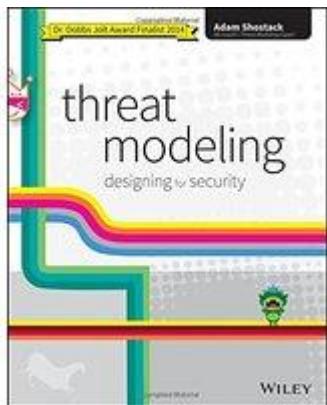
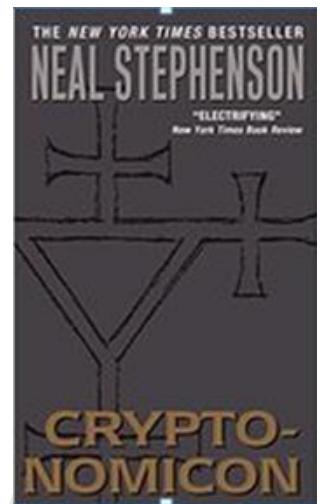
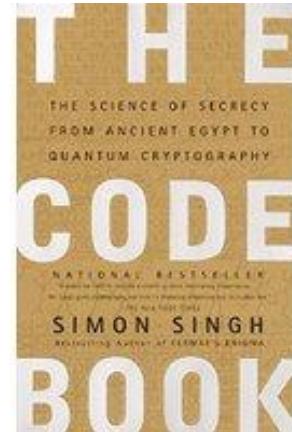
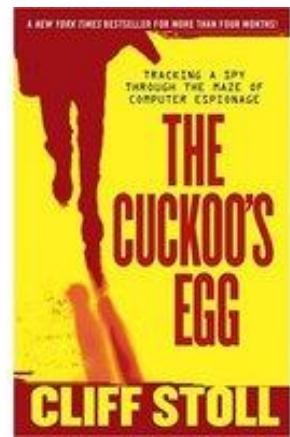
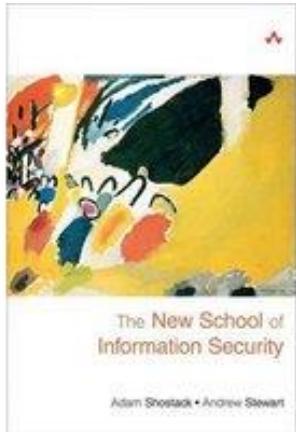
- Lectures (B303A/A042)
- Labs (B308/databars: 001,017,101,109,117,127)
  - Labs must be documented by a short report
    - *4 mandatory assignments*
- Examination
  - Evaluation of reports from 4 mandatory assignments
  - Separate report on chosen topic from the course
    - *Developing a multiple choice question*
  - Final exam (multiple choice)
    - *10% of questions on the final exam will be selected among the student contributed questions, small "obfuscations" are possible*

# Textbook

- Pfleeger & Pfleeger: Security in Computing (5th Edition)



# Extra Curricular Security Reading



# Course Schedule

Date	Topic
1 Sep	Introduction
8 Sep	Protocol Security I
15 Sep	Cryptography I
22 Sep	Protocol Security II
29 Sep	Cryptography II
6 Oct	Security in Networks
13 Oct	Authentication and Identity Management
20 Oct	Autumn Holiday
27 Oct	Protection/Access Control
3 Nov	Privacy and Privacy Enhancing Technologies
10 Nov	Security Management
17 Nov	Software Security I
24 Nov	Software Security II
1 Dec	Legal Issues

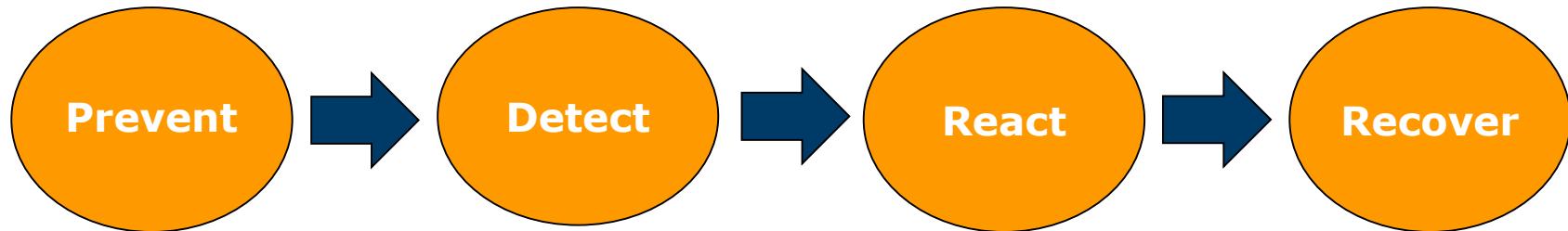
# Security courses at DTU Compute

- Three research groups that focus on security at DTU Compute
  - Embedded Systems Engineering
    - *Investigates security in embedded systems incl. Wireless Sensor Networks and Internet of Things*
  - Formal Methods
    - *Focuses on modelling and analysis of information systems using language-based techniques and tools*
  - Cyber Security
    - *Focuses on models, policies and (cryptographic) mechanisms to develop and support secure computing systems*
- Security courses at DTU Compute
  - 02239 Data Security (fall)
  - 02232 Applied Cryptography (fall)
  - 02255 Practical Cryptology (fall)
  - 02234 Current Topics in Systems Security(fall)
  - 02238 Biometric Systems (June)
  - 02242 Program Analysis (fall)
  - 02233 Network Security (spring)
  - 02244 Language Based Security (spring)
  - 02191 Computer Security Forensics (January)
  - 02192 Computer Security Incident Response (spring)
  - 02193 Ethical Hacking (june)

# Computer Security Study Line

- One the following courses is mandatory
  - 41633 Innovation and Product Development
  - 42435 Knowledge based Entrepreneurship
  - 42490 Technology, Economics, Management and Organization (TEMO)
- Selective courses (30 ECTS must be selected from this list)
  - 02220 Distributed Systems
  - 02232 Applied Cryptography
  - 02233 Network Security
  - 02234 Current Topics in System Security
  - 02238 Biometric Systems
  - 02239 Data Security
  - 02242 Program Analysis
  - 02244 Logic for Security
  - 02255 Modern Cryptology
  - 02291 System Integration

# Specialist Program in Cybersecurity



**Data Security**  
Applied Cryptology  
Current Topics ...  
**Network Security**  
Biometric Systems  
Program analysis  
Language Based ...  
System Integration

**Ethical Hacking**

**Incident Response**

**Security Forensics**

# Join DTUHax

- Meets most Wednesdays 17-19 in the Hackerlab (322/233)



# Event at Børsen (in Danish)

## ***Strategisk cybersikkerhed i 2021***

- Invitation to an event on strategic challenges in cybersecurity
  - Arranged by Dansk Erhverv and Women in Technology
  - Friday 3 September, 13-16:30
  - Børssalen
- You can see the program here:

<https://www.danskerhverv.dk/kurser-og-events/strategisk-cybersikkerhed-i-2021/>

- DTU demonstrate a few examples of hands-on hacking, so we have received a code for free registration: **Womenintech2021**

**Registration deadline is today 1 September**

## An Overview of Computer Security



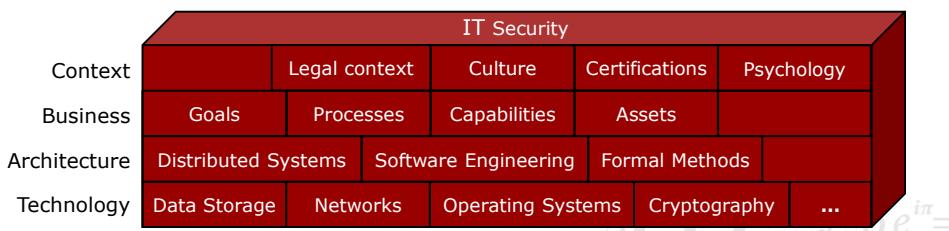
DTU Compute

Department of Applied Mathematics and Computer Science

$$\begin{aligned} \Theta^{\sqrt{17}} &+ \Omega^f \delta e^{i\pi} = \\ \infty^{\infty} &= \{2.7182818284 \\ \Sigma! &\gg , \end{aligned}$$

## Elements of Cybersecurity

- Cybersecurity must consider aspects from many domains of human activity, in addition to the theory and technologies normally attributed to the domain



$$f(x+\Delta x) = \sum_{n=0}^{\infty} \frac{(\Delta x)^n}{n!} f^{(n)}(x)$$

*... this makes it challenging, exciting and rewarding to work with!*

## The Basic Components

- Primary Security Goals (CIA-properties)
  - Confidentiality
  - Integrity
  - Availability
- Other goals frequently listed
  - Accountability
    - Actions can be traced back to a single entity
      - People can be made responsible for their actions
      - Principle known as non-repudiation in cryptography
  - Privacy (e.g. privacy families defined by Common Criteria)
    - Pseudonymity, unlinkability, anonymity, unobservability
    - There is an inherent conflict between accountability and privacy
  - Authenticity
    - Requests or information are authentic and authenticated
    - Resources (both hardware and software) are genuine

## Confidentiality

- Preventing unauthorised observation of information or resources (keeping secrets secret)
  - War-plans, business strategies, client confidentiality (priest/lawyers), ...
- Particularly important in military information security
  - Security models, policies and mechanisms developed to enforce the need-to-know principle
- Confidentiality can be ensured with cryptography
  - A cryptographic key is used to scramble (encrypt) data so that unauthorised entities cannot read it
  - Authorised entities have access to a cryptographic key so that they can restore (decrypt) data to its original form
- Access control mechanisms protect data from unauthorised access
- Confidentiality may extend to protect knowledge about the existence of information or resources

## Integrity

- Preventing unauthorised modification of information or resources
  - Data integrity pertains to the content of the information
  - Origin integrity pertains to the source of the information
    - *Origin integrity implies authentication of the source of the information, which is part of authenticity*
- Two classes of integrity mechanisms:
  - Prevention mechanisms
    - *Prevents data from being modified in unauthorised ways*  
*Prevents the bank's janitor from modifying my bank account, but does not prevent the bank manager from moving all my money to his own account*
  - Detection mechanisms
    - *Detects unauthorised modification of data after the fact*  
*Prevents neither of the scenarios above, but allows both to be detected and corrected*
- Integrity is often more important than confidentiality in commercial information systems

## Availability

- Availability means that the systems information and resources are available to authorised users when they need them
- Attacks against availability is known as *Denial-of-Service* (DoS)
  - Many spectacular DoS attacks reported in the press
- Availability is devilishly difficult and most security research has focused on confidentiality and integrity
  - It is easy to ensure confidentiality and integrity, simply unplug the computer and store it in a bank vault
- Difficulties in ensuring availability include:
  - Difficult to distinguish between high load and DoS ( ./-phenomena)
  - Influenced by factors outside the security model
    - *A backhoe may be used to cut power or communication supplies*

## Risks

- Security is concerned with management of risk
  - Eliminating or reducing harm to assets
- Material Harm
  - Theft of property (e.g. computers, peripherals, ...) or money
  - Harm to people or property (e.g. health, vandalism, ...)
- Immaterial Harm
  - Theft of Intellectual Property (incl. copyright violations)
  - Harm to Intellectual Property (e.g. disclosure of trade secrets)
  - Harm to reputation (e.g., website defacement, bad mouthing, ...)
- Risk Management
  - Risk Identification
  - Risk Analysis/Assessment
  - Risk Treatment
  - Monitoring/Review

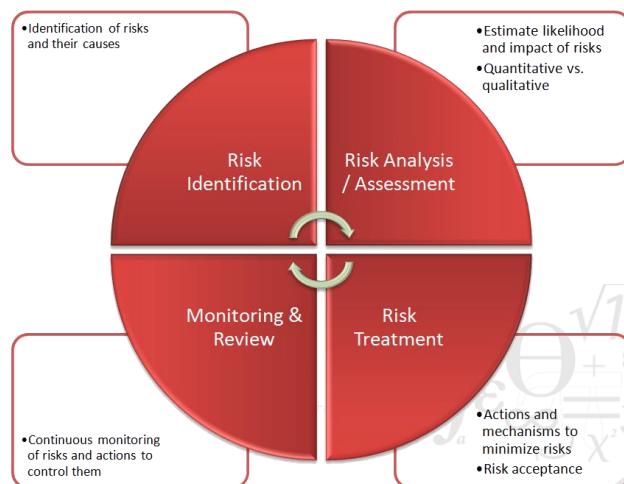


7

DTU Compute Technical University of Denmark

02239 – Data Security

## Risk Management Cycle



8

DTU Compute Technical University of Denmark

02239 – Data Security

## Threats

- A threat is a potential violation of security
  - Often a four step process
    - threat → vulnerability → opportunity → attack (exploit)
- Four major classes of threats:
  - Disclosure (unauthorised access to information)
  - Deception (acceptance of false data)
  - Disruption (interruption or prevention of correct operation)
  - Usurpation (unauthorised control of (part of) the system)
- Five ways to deal with the effects of exploits:
  - Prevention (remove all vulnerabilities)
  - Deterrence (making exploits difficult – *but not impossible*)
  - Deflection (make other targets relatively more attractive)
  - Detection (as they happen or after the fact – *forensics*)
  - Recovery (restore the system to a usable state )

## Vulnerabilities

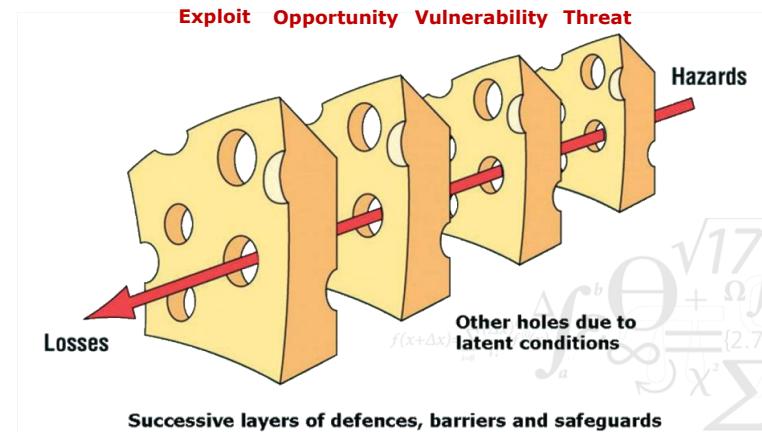
- Weaknesses in the Security Architecture
  - Weak assumptions
    - *Security requirements not specified or poorly understood*
  - Weak architecture
    - *Security requirements not properly identified*
    - *Security architecture does not cover all security requirements*
    - *Security Architecture not up to date (outdated requirements)*
  - Weak components
    - *Poor specification of components of the security architecture*
    - *Poor implementation of components of the security architecture*
    - *Components do not compose securely*
- Weak operation
  - Poor recruitment processes
  - Poor security awareness



## Threats, Vulnerabilities and Attacks

### Putting it All Together

- The "Swiss Cheese" Model

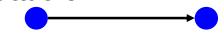


## Pause



## Network Attacks

### Active attacks



*Normal communication*



*Deletion*

*Modification*

### Passive attacks



*Eavesdropping*



*Fabrication*



*Traffic analysis*

## Possible Attackers

- Insiders ( $>50\%$ )
  - Disgruntled employees
  - Guests, consultants, contract workers ...
- Crackers (*hackers*)
  - Technically knowledgeable programmers
- Script-Kiddies (*cracker wannabes*)
  - Tools provided by others
- Spies (*industrial and military*)
  - Technical knowledge, technical means, many resources
- Criminals (*thieves, organized crime*)
  - Technical knowledge, technical means, many resources
- Hacktivists and Terrorists
  - Technical knowledge and means, disproportionate allocation of resources
- We need to consider: *means (method), motives and opportunity*

## Means of attackers

- Insiders
  - Knowledge of system configuration, network topologies, processes,...
  - Only computing resources provided by organisation
- Crackers (*hackers*)
  - Able to adapt tools to configuration of target
  - Able to write new tools/exploits
  - Few computing resources (apart from bot-nets)
- Script-Kiddies (*cracker wannabes*)
  - Can only use tools provided by others (already known attacks)
- Spies (*industrial and military*)
  - Technical knowledge, rich computing resources, other resources
- Criminals (*thieves, organized crime*)
  - Technical knowledge, technical means, many resources
- Terrorists
  - Probably between spies and script-kiddies, but nothing is really known

## Motivation for Attackers

- Curiosity about how the system works
  - The challenge of hacking the system
  - "Ethical hacking" (expose vulnerabilities and warn owners)
    - *SIEM cannot tell difference between white-hat and black-hat hackers, so defenders must always react*
- Fame
  - Recognition for their achievements
- Financial Gains
  - Fraud, theft
  - Industrial Espionage
- Ideology
  - Hactivism: disrupt but do not cause serious damage
  - Cyberterrorism: disrupt/destroy important services

## Opportunities for Attacks

- Insiders
  - Daily access to the network, relatively easy to launch attacks
- Crackers
  - Difficult to access internal network
  - Have to crack the firewall first
- Script-Kiddies
  - Can only exploit holes in the firewall (and possibly wireless access)
- Spies
  - Combination of insiders and crackers
  - 3 most important means of gathering information are:
    - *Beatings, Bribery and Blackmail (3 Bs)* -- Robert Morris Sr.
- Criminals
  - Ability to infiltrate, 3Bs, otherwise difficult
- Terrorists
  - Similar to criminals

## Policy and Mechanisms

- It is always important to distinguish between the policy and the mechanism that enforces the policy
  - **Policy:** statement of what is, and what is not, allowed
  - **Mechanism:** method, tool or process for enforcing a security policy
- Security Policies may be defined in different ways
  - Different levels of detail from very general (users must not copy other users' files) to very specific (user A must not copy user B's files)
  - Different levels of accuracy from general statements in English to precise mathematical formalisms
    - *Formal statements are more precise when they are formulated correctly*
- When multiple organizations collaborate the composed entity often has a security policy based on the individual security policies.
  - This raises the problem of policy composition which is inherently difficult

## Goals of Security Mechanisms

- Security mechanisms are put in place to *prevent* attacks, *detect* attacks and *recover* from attacks
- Prevention
  - Cryptography and access control are often used to prevent attacks
- Detection
  - Intrusion detection systems are often used to detect attacks during or after the attack
- Recovery
  - React to the attack
    - *Common reactions are to stop the attack in progress or allow it to continue with extensive logging (to allow the attacker to be traced)*
  - Repair the damage caused by the attack
    - *Identify the vulnerability, identify appropriate prevention mechanism (e.g., patch the system), determine damage caused by the attack, repair damage caused by the attack (e.g., roll-back of data bases to before the attack)*

## Assumptions and Trust

- Security policies are always based on some assumptions about the behaviour of components and entities in the system
- Two assumptions are generally made:
  - Security policy unambiguously partitions the system state into *secure* and *nonsecure* states
  - Security mechanism will guarantee that a system in the *secure* states will never become *nonsecure*
  - If either assumption fails the system is insecure
- A security mechanism can be characterised as:
  - Secure: it does not allow the system to enter *nonsecure* states
  - Precise: it allows the system to enter all *secure* states
  - Broad: it allows the system to enter states that are *nonsecure*
  - In practise, most security mechanisms are broad

## Trust Assumptions

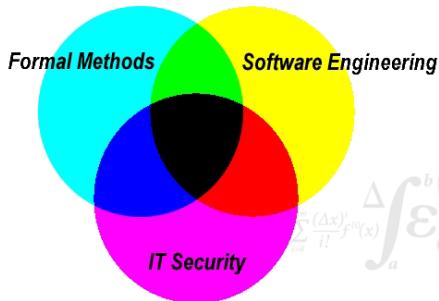
- Trusting that mechanisms work requires several assumptions
  - Each mechanism is designed to implement one or more elements of the security policy
  - The union of security mechanisms implements all aspects of the security policy
  - The mechanisms are implemented correctly
  - The mechanisms are installed and administered correctly
- So what is required to trust encrypted data from the network
  - Encryption algorithm must be strong (design)
  - Encryption algorithm must be implemented correctly (implementation)
  - Encryption software must be installed correctly (operation)
  - Cryptographic key must be secret (administration)

## Assurance

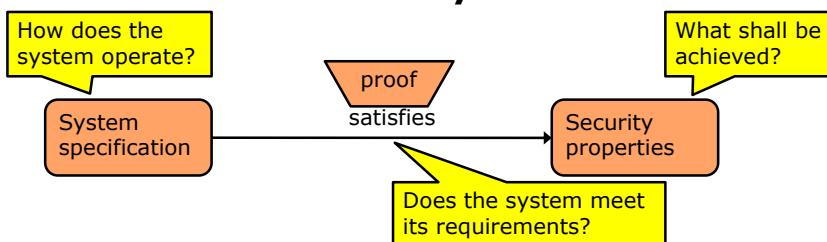
- Assurance attempts to quantify some of the assumptions about trust in the security mechanism
- Assurance requires a specification of the behaviour of the system
  - A system is said to *satisfy* a specification if the specification correctly states how the system will function
- Assurance considers all aspects of software development
  - *Specification* of the system must be correct and unambiguous
  - *Design* of the system translates the specification into components that will implement them
  - *Implementation* creates a system that satisfies the design
    - A program is correct if its implementation performs as specified
  - *Testing* verifies *a posteriori* if the implementation is correct
    - NB! testing cannot prove correctness, only incorrectness
- Stronger assurance requires formal proofs of correctness

## Formal Methods for Security

- Systems can be understood as mathematical objects.
- Formal methods based on mathematics and logic should be used to model, analyze, and construct them.
- Doing so can substantially improve the security



## Formal Methods for Security



- Even the mere attempt to formalize the security properties/goals of a system in a mathematically precise way can be revealing!
- FM group focuses on automatic methods to find either a proof or a counter example – given a specification of the system and the security properties
- Tools based on model-checking, static analysis, abstract interpretation...
- Many attacks have been detected – and fixed -- using such tools:
- H.530, Google-Apps SSO, Kerberos PKInit

## Operational Issues

- Security policies and mechanisms must be effective
  - **Defender:** the cost of an effective attack must be higher than the design, implementation and operation of the security infrastructure
  - **Attacker:** the cost of an effective attack must be lower than the benefits gained through the attack
- Risk analysis determines what attacks are plausible and determines the cost of these attacks
  - Credit card companies accept small amounts of fraud, because it is cheaper than implementing all mechanisms to ensure "perfect" security
- Cost benefit analysis establish whether the benefits of a particular countermeasure exceeds the cost of implementing and operating it
- Laws and customs in the local environment plays an important role
  - Different laws on data protection are defined in different jurisdictions
  - Policies and mechanisms that are legal, but unacceptable, in the local environment should be avoided

## Organisational Issues

- Implementing security in a large organisation is difficult and technology is often the easy bit (although we often get it wrong)
  - Benefits of security are not directly visible *on the bottom line*
  - Some security mechanisms may actually be costly
    - *Protocol overhead slows down communications*
    - *Requiring passwords to be entered frequently tends to lower productivity*
- Responsibility for computer security is not always obvious
  - IT department has responsibility for servers, hardware and software
  - Individual users are responsible for choosing good passwords (and keeping them secret)
  - If an attack exploits a weak password then who is to blame?
- Social Engineering (e.g. phishing) is surprisingly effective
- Security incidents are not always attacks
  - An employee who reads and sends private emails during work hours is generally breaching policy, but the breach is normally accepted

## Human Issues

- Social Engineering is surprisingly effective
  - Main approach used by Kevin Mitnick (FBI Most Wanted for many years)

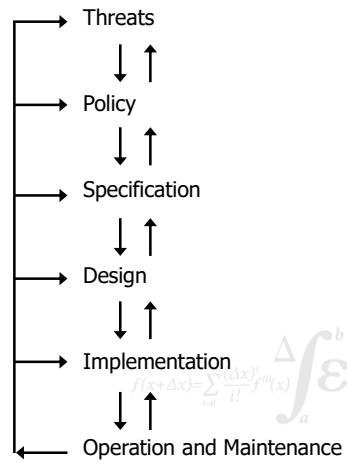


## Security Usability

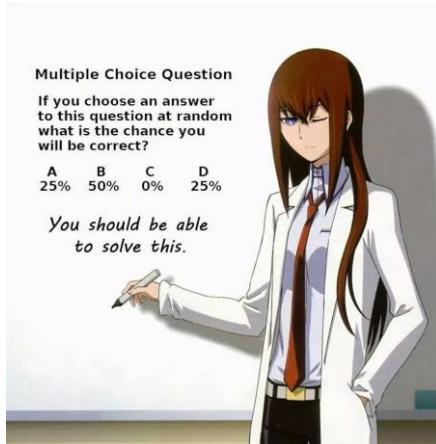
- Security mechanisms must be comprehensible and acceptable



## Security Life Cycle



## Writing Good Multiple Choice Questions



### Multiple Choice Question

If you choose an answer to this question at random what is the chance you will be correct?

A 25%    B 50%    C 0%    D 25%

You should be able to solve this.

DTU Compute

Department of Applied Mathematics and Computer Science

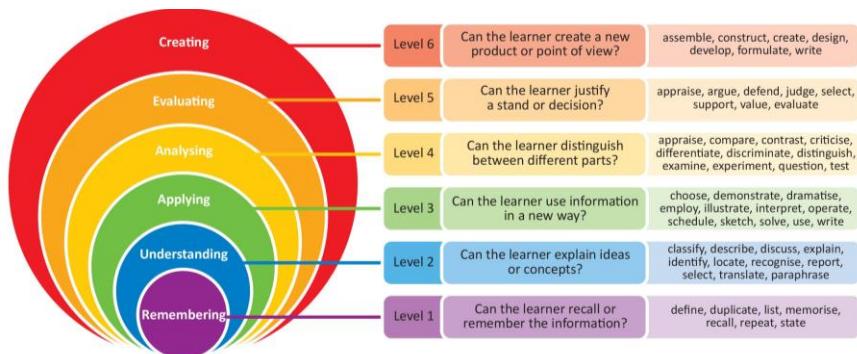
$$\Delta \int_a^b \varepsilon \Theta^{\sqrt{17}} + \Omega \int \delta e^{i\pi} = \\ \infty \approx \{2.7182818284 \\ \sum x^2 \gg , !$$

## Multiple Choice Question Assignment

- Develop a multiple choice question for the exam in 02239
- Document the development of the question in a short report
  - Topic examined in the question
    - *theory behind the question*
    - *learning objectives being tested*
    - *cognitive level addressed by the question*
  - Design of the question
    - *define the correct answer*
    - *make sure that the one correct answer is unambiguous*
  - Design of the wrong answers (distractors)
    - *plausibility of distractors*
    - *ensuring that distractors are not trick questions*
- Report should be no longer than 3-5 pages

## Bloom's Taxonomy of Knowledge

- Classification of educational learning objectives into levels of complexity and specificity



## Multiple Choice Questions

- Multiple choice question has 3 components
  - The question (stem)
  - The correct answer (key)
  - The incorrect answers (distractors)

### Parts of a Multiple-choice Question

Who sent the first internet email using the "@" symbol? > Stem

- a. Al Gore
- b. Douglas Engelbart
- c. Ray Tomlinson
- d. Vint Cerf

Answer

"Distractors"

## Procedural Rules

- Use either the best answer or the correct answer format
  - Best answer format refers to a list of options that can all be correct in the sense that each has an advantage, but one of them is the best
  - Correct answer format refers to one and only one right answer
  - Mark all matching refers to a list of options with several correct answers
- Allow time for editing and other types of item revisions
- Use good grammar, punctuation, and spelling consistently
- Minimize the time required to read each item
- Avoid trick questions
- Use the active voice
- Have your questions peer-reviewed.
- Avoid giving unintended cues – such as making the correct answer longer in length than the distractors

## Content-related Rules

- Base each item on an educational or instructional objective of the course, not trivial information
- Test for important or significant information
- Focus on a single problem or idea for each test item.
- Use the author's examples as a basis for developing your items
- Avoid overly specific knowledge when developing items
- Avoid textbook, verbatim phrasing when developing the items
- Avoid items based on opinions
- Use multiple-choice to measure higher level thinking
- Be sensitive to cultural and gender issues
- Use case-based questions that use a common text to which a set of questions refers

## Stem Construction Rules

- State the stem in either question form or completion form
- Ensure that the directions in the stem are clear, and that wording lets the examinee know exactly what is being asked
- Avoid window dressing (excessive verbiage) in the stem
- Word the stem positively; avoid negative phrasing such as "not" or "except." If this cannot be avoided, the negative words should always be highlighted by underlining or capitalization:  
Which of the following is NOT an example .....
- Include the central idea and most of the phrasing in the stem.
- Avoid giving clues such as linking the stem to the answer  
(.... Is an example of *an*: test-wise students will know the correct answer should start with a vowel)

## General Option Development Rules

- Place options in logical or numerical order
- Keep options independent; options should not be overlapping
- Keep all options homogeneous in content
- Keep the length of options fairly consistent.
- Phrase options positively, not negatively
- Avoid distractors that can clue test-wise examinees; for example, absurd options, formal prompts, or semantic (overly specific or overly general) clues
- Avoid giving clues through the use of faulty grammatical construction.
- Avoid specific determinates, such as *never* and *always*

## Distractor Development Rules

- Use plausible distractors
- Avoid technically phrased distractors
- Use familiar yet incorrect phrases as distractors
- Use true statements that do not correctly answer the item
- Avoid the use of humour when developing options



## Ideas for Good Multiple Choice Questions

- Present practical or real-world situations
- Present a diagram of system and ask for application, analysis or evaluation
- Present actual quotations taken from newspapers or other published sources and ask for the interpretation or evaluation of these quotations
- Use pictorial materials that require students to apply principles and concepts
- Use charts, tables or figures that require interpretation

## Scenario-Based Problem Solving Item Set

- Good for testing higher level cognitive skills
- Present a scenario and ask questions that:
  - require understanding of the theory
  - require application of theory and techniques to specific scenario
  - may require calculations on scratch paper to answer



# 02239 Data Security Introduction to Protocol Security

Sebastian Mödersheim



September 8, 2021



```

$ file chall
chall: ELF 32-bit LSB executable, Intel 80386,, statically linked, stripped
$ readelf -s chall
readelf: Error: Unable to read in 0x28 bytes of section headers
readelf: Error: Unable to read in 0x488 bytes of section headers
readelf: Error: Unable to read in 0x100 bytes of program headers
$ hexdump -C chall | head -2
00000000  7f 45 4c 46 01 01 01 13  37 13 37 13 37 13 37 00  |.ELF....7.7.7.7.| 
00000010  02 00 03 00 01 13 37 00  90 84 04 08 34 13 37 00  |.....7.....4.7.| 
$ hexdump -C /bin/ls | head -2
00000000  7f 45 4c 46 01 01 00  00 00 00 00 00 00 00 00 00  |.ELF.....| 
00000010  02 00 03 00 01 00 00  b4 c1 04 08 34 00 00 00 00  |.....4...| 
$ cp chall wip
$ sed -i 's/\x13\x37\x00\x00/g' wip
$ readelf -s wip

```

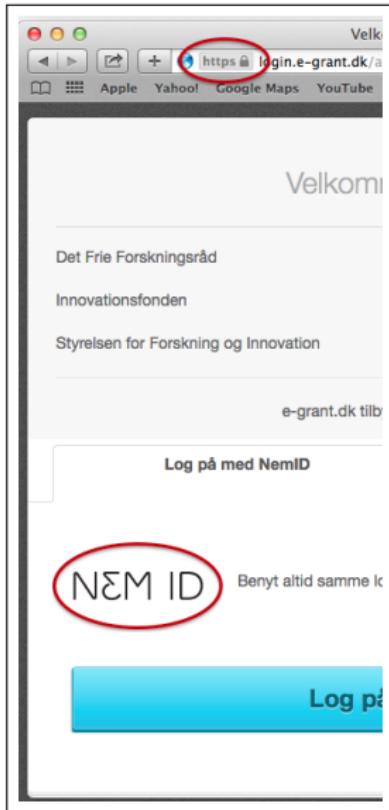
Symbol table '.dynsym' contains 12 entries:

Num:	Value	Size	Type	Bind	Vis	Ndx	Name
0:	00000000	0	NOTYPE	LOCAL	DEFAULT	UND	
1:	00000000	0	FUNC	GLOBAL	DEFAULT	UND	printf@GLIBC_2.0 (2)
2:	00000000	0	FUNC	GLOBAL	DEFAULT	UND	free@GLIBC_2.0 (2)
3:	00000000	0	FUNC	GLOBAL	DEFAULT	UND	fgets@GLIBC_2.0 (2)
4:	00000000	0	FUNC	GLOBAL	DEFAULT	UND	malloc@GLIBC_2.0 (2)
5:	00000000	0	FUNC	GLOBAL	DEFAULT	UND	puts@GLIBC_2.0 (2)
6:	00000000	0	NOTYPE	WEAK	DEFAULT	UND	__gmon_start__
7:	00000000	0	FUNC	GLOBAL	DEFAULT	UND	exit@GLIBC_2.0 (2)
8:	00000000	0	FUNC	GLOBAL	DEFAULT	UND	strlen@GLIBC_2.0 (2)
9:	00000000	0	FUNC	GLOBAL	DEFAULT	UND	__libc_start_main@GLIBC_2.0 (2)
10:	0804889c	4	OBJECT	GLOBAL	DEFAULT	16	_IO_stdin_used
11:	08049aa0	4	OBJECT	GLOBAL	DEFAULT	26	stdin@GLIBC_2.0 (2)

\$ ./wip

Ready to become 1337?

# Security Protocols



- TLS/SSL
- NemID
- IPSec,  
IKE/IKEv2
- Kerberos
- WEP/WPA
- Single Sign On
- Mobile IP
- SIP, SOAP,  
geopriv.



# Today's Program

① Introduction

② Needham-Schroeder

③ Development of a Key-Exchange Protocol in AnB

④ Diffie-Hellman

# Cryptographic Building Blocks (I)

- Symmetric Cryptography:
  - ★ two (or more) agents share a secret key  $K$
  - ★  $\{M\}_K$  denotes symmetric encryption of message  $M$  with key  $K$
  - ★ one can decrypt  $\{M\}_K$  only when knowing  $K$

# Cryptographic Building Blocks (I)

- Symmetric Cryptography:
  - ★ two (or more) agents share a secret key  $K$
  - ★  $\{M\}_K$  denotes symmetric encryption of message  $M$  with key  $K$
  - ★ one can decrypt  $\{M\}_K$  only when knowing  $K$
- Asymmetric Encryption (Public-key Encryption):
  - ★ every agent  $A$  has a key-pair  $(K, \text{inv}(K))$  consisting of
    - ▶ the public key  $K$  that everybody knows
    - ▶ the private key  $\text{inv}(K)$  that only  $A$  knows
  - ★  $\{M\}_K$  denotes asymmetric encryption of  $M$  with public key  $K$ .
  - ★ one can decrypt  $\{M\}_K$  only when knowing  $\text{inv}(K)$

# Cryptographic Building Blocks (I)

- Symmetric Cryptography:
  - ★ two (or more) agents share a secret key  $K$
  - ★  $\{M\}_K$  denotes symmetric encryption of message  $M$  with key  $K$
  - ★ one can decrypt  $\{M\}_K$  only when knowing  $K$
- Asymmetric Encryption (Public-key Encryption):
  - ★ every agent  $A$  has a key-pair  $(K, \text{inv}(K))$  consisting of
    - ▶ the public key  $K$  that everybody knows
    - ▶ the private key  $\text{inv}(K)$  that only  $A$  knows
  - ★  $\{M\}_K$  denotes asymmetric encryption of  $M$  with public key  $K$ .
  - ★ one can decrypt  $\{M\}_K$  only when knowing  $\text{inv}(K)$
- Digital Signatures
  - ★ Signing is “encryption” with a private key
  - ★ Signature checking is “decryption” with a public key
  - ★ Thus, if  $(K, \text{inv}(K))$  is the key pair of  $A$  then
    - ▶  $\{M\}_{\text{inv}(K)}$  can only be produced by  $A$
    - ▶ but everybody can read  $M$  and check that it comes from  $A$ .

# Cryptographic Building Blocks (II)

- Nonces (From [Number once](#)): random numbers that is used only once for a [challenge-response](#)

# Cryptographic Building Blocks (II)

- Nonces (From [Number once](#)): random numbers that is used only once for a [challenge-response](#)
- Hashes
  - ★  $h(M)$  denotes the [cryptographic hash](#) of message  $M$ .
  - ★ Hard to [invert](#): given  $h(M)$ , find  $M$ .
  - ★ Hard to [find collisions](#): find  $M$  and  $M'$  with  $h(M) = h(M')$ .
  - ★ Variant: [Message Authentication Code](#) (MAC, keyed hash)  
 $h(K, M)$  additionally has a symmetric key  $K$ .

# Cryptographic Building Blocks (II)

- Nonces (From [Number once](#)): random numbers that is used only once for a [challenge-response](#)
- Hashes
  - ★  $h(M)$  denotes the [cryptographic hash](#) of message  $M$ .
  - ★ Hard to [invert](#): given  $h(M)$ , find  $M$ .
  - ★ Hard to [find collisions](#): find  $M$  and  $M'$  with  $h(M) = h(M')$ .
  - ★ Variant: [Message Authentication Code](#) (MAC, keyed hash)  
 $h(K, M)$  additionally has a symmetric key  $K$ .
- Timestamps

# Cryptographic Building Blocks (II)

- Nonces (From Number once): random numbers that is used only once for a challenge-response
  - Hashes
    - ★  $h(M)$  denotes the cryptographic hash of message  $M$ .
    - ★ Hard to invert: given  $h(M)$ , find  $M$ .
    - ★ Hard to find collisions: find  $M$  and  $M'$  with  $h(M) = h(M')$ .
    - ★ Variant: Message Authentication Code (MAC, keyed hash)  
 $h(K, M)$  additionally has a symmetric key  $K$ .
  - Timestamps
  - Concatenation, i.e., a sequence of messages
    - ★ written as  $M_1, M_2, M_3$  for simplicity
    - ★ reality: quite complex encodings and source of mistakes
- see also: Mödersheim & Katsoris. *A sound abstraction of the parsing problem*, CSF 2014.

# Alice and Bob notation

(aka Message Sequence Charts aka Narrations)

A typical protocol description combines prose, data type specifications, various kinds of diagrams, ad hoc notations, and message sequences like

1.  $A \rightarrow B : \{NA, A\}_{pk(B)}$
2.  $B \rightarrow A : \{NA, NB\}_{pk(A)}$
3.  $A \rightarrow B : \{NB\}_{pk(B)}$

They often include informal statements concerning the properties of the protocol and why they should hold.

There are formal languages based on this notation, e.g. [AnB](#).

## Protocols (cont.)

What does a message  $A \rightarrow B : M$  actually mean?

*We assume that an intruder can interpose a computer in all communication paths, and thus can alter or copy parts of messages, replay messages, or emit false material.*

*We also assume that each principal has a secure environment in which to compute such as is provided by a personal computer...*

*Needham and Schroeder*

We will be more precise later..

# Today's Program

① Introduction

② Needham-Schroeder

③ Development of a Key-Exchange Protocol in AnB

④ Diffie-Hellman

# An authentication protocol

The Needham-Schroeder Public Key protocol (NSPK, 1978):

1.  $A \rightarrow B : \{NA, A\}_{pk(B)}$
2.  $B \rightarrow A : \{NA, NB\}_{pk(A)}$
3.  $A \rightarrow B : \{NB\}_{pk(B)}$

## How the protocol is executed

1.  $A \rightarrow B : \{NA, A\}_{pk(B)}$
2.  $B \rightarrow A : \{NA, NB\}_{pk(A)}$
3.  $A \rightarrow B : \{NB\}_{pk(B)}$

Role A:

1. Generate nonce  $NA$ , compose  $\{NA, A\}_{pk(B)}$  and send to  $B$ .
2. Receive some message  $M$ .  
Decrypt  $M$  it with  $\text{inv}(pk(A))$ .  
Split the result into two nonces  $NA'$  and  $NB$ .  
Check that  $NA = NA'$   
If any of this fails, reject  $M$ .
3. Compose  $\{NB\}_{pk(B)}$  and send to  $B$ .

# Problem with protocols

- Goal: mutual (entity) authentication.
- Correctness argument (informal):

1.  $A \rightarrow B : \{NA, A\}_{pk(B)}$  “This is Alice and I have chosen a nonce  $NA$ .”

2.  $B \rightarrow A : \{NA, NB\}_{pk(A)}$  “Here is your Nonce  $NA$ . Since I could read it, I must be Bob. I also have a challenge  $NB$  for you.”

3.  $A \rightarrow B : \{NB\}_{pk(B)}$  “You sent me  $NB$ . Since only Alice can read this and I sent it back, I must be Alice.”

# Problem with protocols

- Goal: mutual (entity) authentication.
- Correctness argument (informal):

1.  $A \rightarrow B : \{NA, A\}_{pk(B)}$  “This is Alice and I have chosen a nonce  $NA$ .”

2.  $B \rightarrow A : \{NA, NB\}_{pk(A)}$  “Here is your Nonce  $NA$ . Since I could read it, I must be Bob. I also have a challenge  $NB$  for you.”

3.  $A \rightarrow B : \{NB\}_{pk(B)}$  “You sent me  $NB$ . Since only Alice can read this and I sent it back, I must be Alice.”

Protocols are typically small and convincing...

# Problem with protocols

- Goal: mutual (entity) authentication.
- Correctness argument (informal):

1.  $A \rightarrow B : \{NA, A\}_{pk(B)}$  “This is Alice and I have chosen a nonce  $NA$ .”

2.  $B \rightarrow A : \{NA, NB\}_{pk(A)}$  “Here is your Nonce  $NA$ . Since I could read it, I must be Bob. I also have a challenge  $NB$  for you.”

3.  $A \rightarrow B : \{NB\}_{pk(B)}$  “You sent me  $NB$ . Since only Alice can read this and I sent it back, I must be Alice.”

Protocols are typically small and convincing... **and wrong!**

# Man-in-the-middle attack

NSPK (1978)

$$\begin{aligned} A \rightarrow B : & \{NA, A\}_{pk(B)} \\ B \rightarrow A : & \{NA, NB\}_{pk(A)} \\ A \rightarrow B : & \{NB\}_{pk(B)} \end{aligned}$$

# Man-in-the-middle attack

NSPK (1978)

$$\begin{aligned} A \rightarrow B &: \{NA, A\}_{pk(B)} \\ B \rightarrow A &: \{NA, NB\}_{pk(A)} \\ A \rightarrow B &: \{NB\}_{pk(B)} \end{aligned}$$

Attack (Lowe 1996):

1.  $a \rightarrow i : \{na, a\}_{pk(i)}$

# Man-in-the-middle attack

NSPK (1978)

$$\begin{aligned} A \rightarrow B &: \{NA, A\}_{pk(B)} \\ B \rightarrow A &: \{NA, NB\}_{pk(A)} \\ A \rightarrow B &: \{NB\}_{pk(B)} \end{aligned}$$

Attack (Lowe 1996):

$$1. \quad a \rightarrow i : \{na, a\}_{pk(i)}$$

$$1.' \quad i(a) \rightarrow b : \{na, a\}_{pk(b)}$$

# Man-in-the-middle attack

NSPK (1978)

$$\begin{aligned} A \rightarrow B &: \{NA, A\}_{pk(B)} \\ B \rightarrow A &: \{NA, NB\}_{pk(A)} \\ A \rightarrow B &: \{NB\}_{pk(B)} \end{aligned}$$

Attack (Lowe 1996):

$$1. \quad a \rightarrow i : \{na, a\}_{pk(i)}$$

$$1.' \quad i(a) \rightarrow b : \{na, a\}_{pk(b)}$$

$$2.' \quad b \rightarrow i(a) : \{na, nb\}_{pk(a)}$$

# Man-in-the-middle attack

NSPK (1978)

$$\begin{aligned} A \rightarrow B &: \{NA, A\}_{pk(B)} \\ B \rightarrow A &: \{NA, NB\}_{pk(A)} \\ A \rightarrow B &: \{NB\}_{pk(B)} \end{aligned}$$

Attack (Lowe 1996):

1.  $a \rightarrow i : \{na, a\}_{pk(i)}$

1.'  $i(a) \rightarrow b : \{na, a\}_{pk(b)}$

2.'  $b \rightarrow i(a) : \{na, nb\}_{pk(a)}$

2.  $i \rightarrow a : \{na, nb\}_{pk(a)}$

# Man-in-the-middle attack

NSPK (1978)

$$\begin{aligned} A \rightarrow B &: \{NA, A\}_{pk(B)} \\ B \rightarrow A &: \{NA, NB\}_{pk(A)} \\ A \rightarrow B &: \{NB\}_{pk(B)} \end{aligned}$$

Attack (Lowe 1996):

1.  $a \rightarrow i : \{na, a\}_{pk(i)}$

1.'  $i(a) \rightarrow b : \{na, a\}_{pk(b)}$

2.'  $b \rightarrow i(a) : \{na, nb\}_{pk(a)}$

2.  $i \rightarrow a : \{na, nb\}_{pk(a)}$

3.  $a \rightarrow i : \{nb\}_{pk(i)}$

# Man-in-the-middle attack

NSPK (1978)

$$\begin{aligned} A \rightarrow B &: \{NA, A\}_{pk(B)} \\ B \rightarrow A &: \{NA, NB\}_{pk(A)} \\ A \rightarrow B &: \{NB\}_{pk(B)} \end{aligned}$$

Attack (Lowe 1996):

1.  $a \rightarrow i : \{na, a\}_{pk(i)}$

1.'  $i(a) \rightarrow b : \{na, a\}_{pk(b)}$

2.'  $b \rightarrow i(a) : \{na, nb\}_{pk(a)}$

2.  $i \rightarrow a : \{na, nb\}_{pk(a)}$

3.  $a \rightarrow i : \{nb\}_{pk(i)}$

3.'  $i(a) \rightarrow b : \{nb\}_{pk(b)}$

# Man-in-the-middle attack

NSPK (1978)

$$\begin{aligned} A \rightarrow B : & \{NA, A\}_{pk(B)} \\ B \rightarrow A : & \{NA, NB\}_{pk(A)} \\ A \rightarrow B : & \{NB\}_{pk(B)} \end{aligned}$$

Attack (Lowe 1996):

1.  $a \rightarrow i : \{na, a\}_{pk(i)}$

1.'  $i(a) \rightarrow b : \{na, a\}_{pk(b)}$

2.'  $b \rightarrow i(a) : \{na, nb\}_{pk(a)}$

2.  $i \rightarrow a : \{na, nb\}_{pk(a)}$

3.  $a \rightarrow i : \{nb\}_{pk(i)}$

3.'  $i(a) \rightarrow b : \{nb\}_{pk(b)}$

**What went wrong?**

# What went wrong?

NSPK (1978)

$$\begin{aligned} A \rightarrow B &: \{NA, A\}_{pk(B)} \\ B \rightarrow A &: \{NA, NB\}_{pk(A)} \\ A \rightarrow B &: \{NB\}_{pk(B)} \end{aligned}$$

Attack (Lowe 1996):

1.  $a \rightarrow i : \{na, a\}_{pk(i)}$

1.'  $i(a) \rightarrow b : \{na, a\}_{pk(b)}$

2.'  $b \rightarrow i(a) : \{na, nb\}_{pk(a)}$

2.  $i \rightarrow a : \{na, nb\}_{pk(a)}$

3.  $a \rightarrow i : \{nb\}_{pk(i)}$

3.'  $i(a) \rightarrow b : \{nb\}_{pk(b)}$

# What went wrong?

NSPK (1978)

$$\begin{aligned} A \rightarrow B : & \{NA, A\}_{pk(B)} \\ B \rightarrow A : & \{NA, NB\}_{pk(A)} \\ A \rightarrow B : & \{NB\}_{pk(B)} \end{aligned}$$

Attack (Lowe 1996):

1.  $a \rightarrow i : \{na, a\}_{pk(i)}$

1.'  $i(a) \rightarrow b : \{na, a\}_{pk(b)}$

2.'  $b \rightarrow i(a) : \{na, nb\}_{pk(a)}$

2.  $i \rightarrow a : \{na, nb\}_{pk(a)}$

3.  $a \rightarrow i : \{nb\}_{pk(i)}$

3.'  $i(a) \rightarrow b : \{nb\}_{pk(b)}$

Nothing in second message indicates who created it.

# What went wrong?

NSPK (1978)

$$\begin{aligned} A \rightarrow B : & \{NA, A\}_{pk(B)} \\ B \rightarrow A : & \{NA, NB\}_{pk(A)} \\ A \rightarrow B : & \{NB\}_{pk(B)} \end{aligned}$$

Attack (Lowe 1996):

1.  $a \rightarrow i : \{na, a\}_{pk(i)}$

1.'  $i(a) \rightarrow b : \{na, a\}_{pk(b)}$

2.'  $b \rightarrow i(a) : \{na, nb\}_{pk(a)}$

2.  $i \rightarrow a : \{na, nb\}_{pk(a)}$

3.  $a \rightarrow i : \{nb\}_{pk(i)}$

3.'  $i(a) \rightarrow b : \{nb\}_{pk(b)}$

Nothing in second message indicates who created it.

The protocol **wrongly assumes** that the second message can only be created by the person to whom  $A$  had sent  $NA$  in the first step.

## What went wrong?

- Problem in step 2.

$$B \rightarrow A : \{NA, NB\}_{pk(A)}$$

Agent  $B$  should also give his name:  $\{NA, NB, \textcolor{green}{B}\}_{pk(A)}$ .

- Known as [Lowe's Fix](#).
- Is the improved version now correct?

# Today's Program

- ① Introduction
- ② Needham-Schroeder
- ③ Development of a Key-Exchange Protocol in AnB
- ④ Diffie-Hellman

# AnB and OFMC

- AnB: Formal language based on [Alice and Bob](#) notation for security protocols
- OFMC: Open-Source Fixedpoint Model-Checker
  - ★ Searching for attacks against an AnB protocol
  - ★ Verification for a bounded number of protocol sessions

- Demo/Tutorial for AnB.
- The AnB source codes of the protocol will be put on campusnet
- A similar development is found in the folder *AnB Tutorial* in the OFMC package.

# Today's Program

- ① Introduction
- ② Needham-Schroeder
- ③ Development of a Key-Exchange Protocol in AnB
- ④ Diffie-Hellman

# Cyclic Group

For a **large** prime  $p$  consider:

$$\mathbb{Z}_p^* = \{1, \dots, p - 1\}$$

- multiplication **modulo  $p$** ,  
e.g.  $5 * 3 \bmod 11 = 15 \bmod 11 = 4$ .
- exponentiation also modulo  $p$ ,  
e.g.  $5^2 \bmod 11 = 25 \bmod 11 = 3$ .

# Cyclic Group

Consider exponentiations of the form  $g^X \pmod{p}$  with  $g, X \in \mathbb{Z}_p^*$ .

E.g.  $p = 7$ :

X	1	2	3	4	5	6
g=1	1	1	1	1	1	1
g=2	2	4	1	2	4	1
g=3	3	2	6	4	5	1
g=4	4	2	1	4	2	1
g=5	5	4	6	2	3	1
g=6	6	1	6	1	6	1

# Cyclic Group

Consider exponentiations of the form  $g^X \pmod{p}$  with  $g, X \in \mathbb{Z}_p^*$ .

E.g.  $p = 7$ :

X	1	2	3	4	5	6
g=1	1	1	1	1	1	1
g=2	2	4	1	2	4	1
g=3	3	2	6	4	5	1
g=4	4	2	1	4	2	1
g=5	5	4	6	2	3	1
g=6	6	1	6	1	6	1

- $g$  is called a **generator** for  $\mathbb{Z}_p^*$  if  $\mathbb{Z}_p^* = \{g^X \pmod{p} \mid X \in \mathbb{Z}_p^*\}$ .

# Cyclic Group

Consider exponentiations of the form  $g^X \pmod{p}$  with  $g, X \in \mathbb{Z}_p^*$ .

E.g.  $p = 7$ :

X	1	2	3	4	5	6
g=1	1	1	1	1	1	1
g=2	2	4	1	2	4	1
g=3	3	2	6	4	5	1
g=4	4	2	1	4	2	1
g=5	5	4	6	2	3	1
g=6	6	1	6	1	6	1

- $g$  is called a **generator** for  $\mathbb{Z}_p^*$  if  $\mathbb{Z}_p^* = \{g^X \pmod{p} \mid X \in \mathbb{Z}_p^*\}$ .
- In the example  $p = 7$ ,  $g = 3$  and  $g = 5$  are generators.

# Cyclic Group

Consider exponentiations of the form  $g^X \pmod{p}$  with  $g, X \in \mathbb{Z}_p^*$ .

E.g.  $p = 7$ :

X	1	2	3	4	5	6
g=1	1	1	1	1	1	1
g=2	2	4	1	2	4	1
g=3	3	2	6	4	5	1
g=4	4	2	1	4	2	1
g=5	5	4	6	2	3	1
g=6	6	1	6	1	6	1

- $g$  is called a **generator** for  $\mathbb{Z}_p^*$  if  $\mathbb{Z}_p^* = \{g^X \pmod{p} \mid X \in \mathbb{Z}_p^*\}$ .
- In the example  $p = 7$ ,  $g = 3$  and  $g = 5$  are generators.
- Given  $g, p, X$  it is **easy** to compute  $g^X \pmod{p}$ .

# Cyclic Group

Consider exponentiations of the form  $g^X \pmod{p}$  with  $g, X \in \mathbb{Z}_p^*$ .

E.g.  $p = 7$ :

X	1	2	3	4	5	6
g=1	1	1	1	1	1	1
g=2	2	4	1	2	4	1
g=3	3	2	6	4	5	1
g=4	4	2	1	4	2	1
g=5	5	4	6	2	3	1
g=6	6	1	6	1	6	1

- $g$  is called a **generator** for  $\mathbb{Z}_p^*$  if  $\mathbb{Z}_p^* = \{g^X \pmod{p} \mid X \in \mathbb{Z}_p^*\}$ .
- In the example  $p = 7$ ,  $g = 3$  and  $g = 5$  are generators.
- Given  $g, p, X$  it is **easy** to compute  $g^X \pmod{p}$ .
- Given  $g, p, g^X \pmod{p}$  it is **believed hard** to compute  $X$ .

Kivinen &amp; Kojo

RFC 3526

5. 4096-bit MODP Group

Standards Track

MODP Diffie-Hellman groups for IKE

[Page 4]

May 2003

This group is assigned id 16.

This prime is:  $2^{4096} - 2^{4032} - 1 + 2^{64} * \{ [2^{3966} p_1] + 240904 \}$ 

Its hexadecimal value is:

```

FFFFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
EE386FBF 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B
E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9
DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510
15728E5A 8AAC42D AD33170D 04507A33 A85521AB DF1CBA64
ECFB8504 58DBEFOA 8AEA7157 5D060C7D B3970F85 A6E1E4C7
ABF5AE8C DB0933D7 1E8C94E0 4A25619D CEE3D226 1AD2EE6B
F12FFA06 D98A0864 D8760273 3EC86A64 521F2B18 177B200C
BBE11757 7A615D6C 770988C0 BAD946E2 08E24FA0 74E5AB31
43DB5BFC EOFD108E 4B82D120 A9210801 1A723C12 A787E6D7
88719A10 BDBA5B26 99C32718 6AF4E23C 1A946834 B6150BDA
2583E9CA 2AD44CE8 DBBBC2DB 04DE8EF9 2E8EFC14 1FBECAA6
287C5947 4E6BC05D 99B2964F A090C3A2 233BA186 515BE7ED
1F612970 CEE2D7AF B81BDD76 2170481C D0069127 D5B05AA9
93B4EA98 8D8FDDC1 86FFB7DC 90A6C08F 4DF435C9 34063199
FFFFFFFFFF FFFFFFFF

```

The generator is: 2.

# Unauthenticated Diffie-Hellman

Protocol : *Diffie-Hellman*

Types :

Agent  $A, B$ ;

Number  $g, X, Y, \text{Msg}$ ;

Function  $\text{pk}$ ;

Knowledge :

$A : A, B, g, \text{pk}, \text{inv}(\text{pk}(A))$ ;

$B : B, g, \text{pk}, \text{inv}(\text{pk}(B))$ ;

Actions :

$A \rightarrow B : \exp(g, X)$

$B \rightarrow A : \exp(g, Y)$

$A \rightarrow B : \{\|A, \text{Msg}\|\}_{\exp(\exp(g, X), Y)}$

Goals :

...

Relies on the algebraic property

$\exp(\exp(g, X), Y) \approx \exp(\exp(g, Y), X)$ .

## Diffie-Hellman: man-in-the-middle attack

- Diffie-Hellman (without authentication of the half-keys) can be attacked:

$$1. \quad a \rightarrow i(b) : \exp(g, x)$$

$$1.' \quad i(a) \rightarrow b : \exp(g, z)$$

$$2.' \quad b \rightarrow i(a) : \exp(g, y)$$

$$2. \quad i(b) \rightarrow a : \exp(g, z)$$

- $a$  believes to share key  $\exp(\exp(g, x), z)$  with  $b$ .  $b$  believes ...
- The intruder knows both keys ....
- Prevention: authenticate the half-keys, e.g. with digital signatures:

$$1. \quad A \rightarrow B : \{\exp(g, X)\}_{\text{inv}(\text{pk}(A))}$$

$$2. \quad B \rightarrow A : \{\exp(g, Y)\}_{\text{inv}(\text{pk}(B))}$$

- In general, using Diffie-Hellman for key-exchange is a good idea!

# Bibliography

- Matt Bishop. *Computer Security (Art and Science)*. Pearson, 2003.
- Kaufman, Perlman, Speciner. *Network Security: Private Communication in a Public World*, Prentice Hall, 2002.
- Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, 1996.
- Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
- John Clark and Jeremy Jacob: A survey of authentication protocol literature, 1997. <http://www.cs.york.ac.uk/~jac/>
- Martín Abadi and Roger Needham: Prudent Engineering Practice for Cryptographic Protocols. *IEEE Transactions on Software Engineering*, 22(1):2-15, 1996.
- Sebastian Mödersheim and Luca Viganò: The Open-source Fixed-point Model Checker for Symbolic Analysis of Security Protocols. *Fosad 2007-2008-2009, LNCS 5705*, Springer, 2009.

# 02239 Data Security Security Protocols II

Sebastian Mödersheim



September 22, 2021

# Outline

- ① Assumptions and Goals
- ② Channels
- ③ TLS
- ④ Single Sign-On
- ⑤ Password Guessing Attacks

# Outline

- ① Assumptions and Goals
- ② Channels
- ③ TLS
- ④ Single Sign-On
- ⑤ Password Guessing Attacks

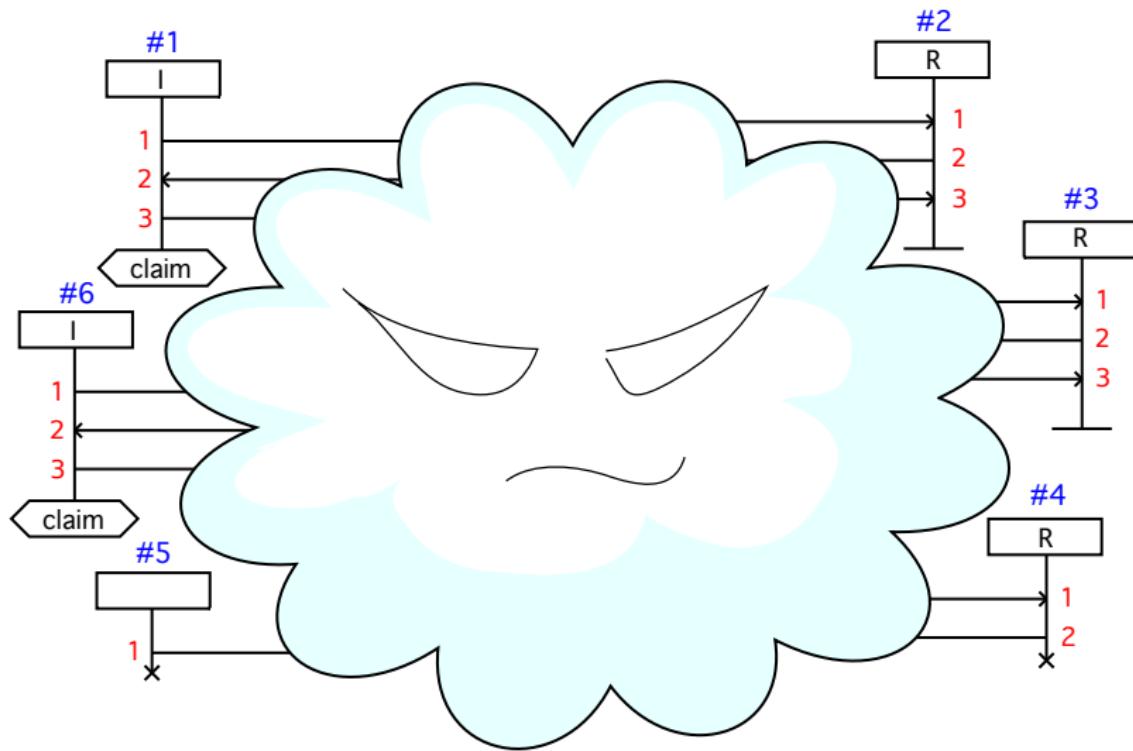
# Examples of Intruder Models



- He knows the protocol but cannot break cryptography.  
(Standard: **perfect encryption.**)
- He is **passive** but overhears all communications.
- He is **active** and can intercept and generate messages.  
“Transfer 100 Kr to Dorrit” ↵ “Transfer 10000 Kr to Charlie”  
Worst-case: the intruder controls the entire network
- He might even be one of the principals running the protocol!  
We should expect this unless a role is explicitly assumed to be trusted/honest.

*A friend's just an enemy in disguise. You can't trust nobody.*  
(Charles Dickens, *Oliver Twist*)

# Deployment in a Hostile Environment



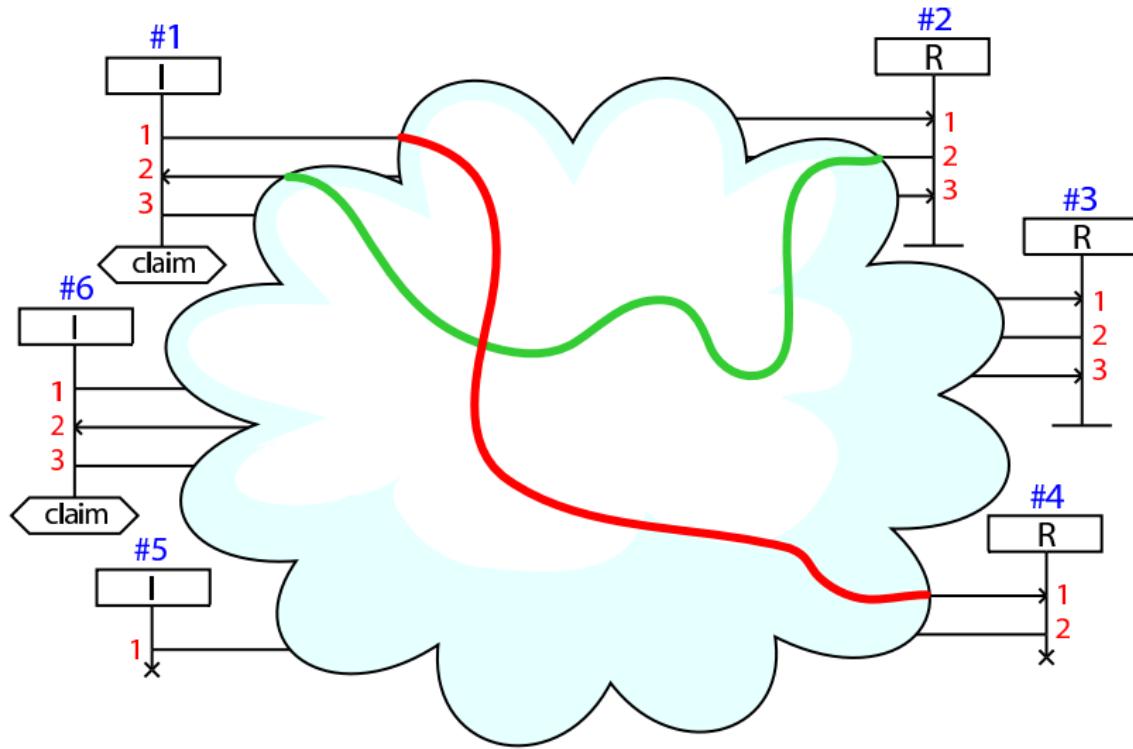
*Cloud illustrations from Cas Cremers.*

Sebastian Mödersheim

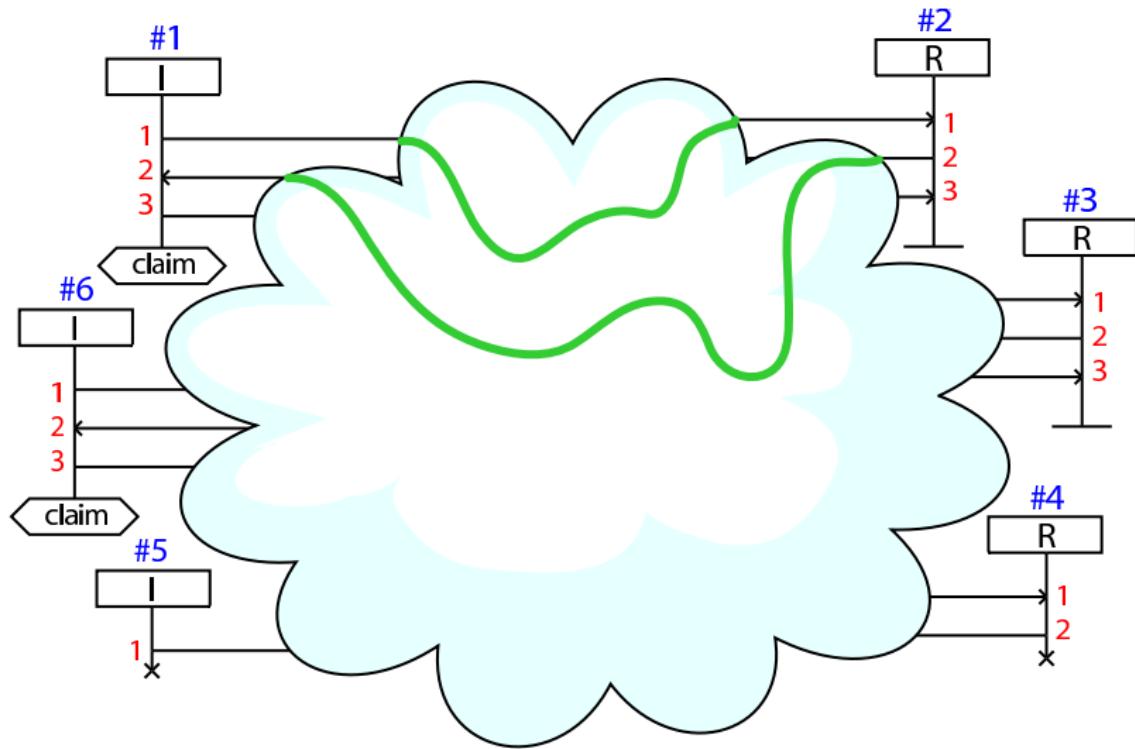
September 22, 2021

5 of 32

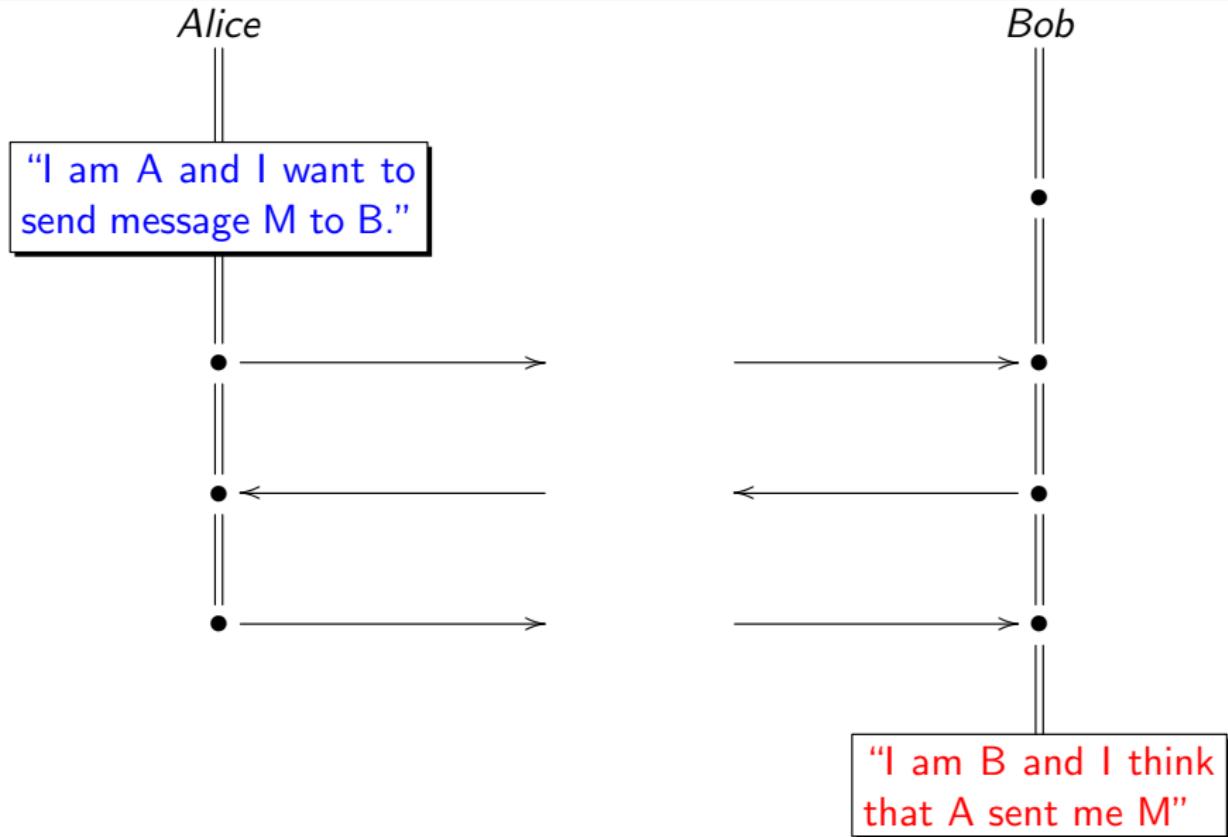
# Failed (Weak) Authentication



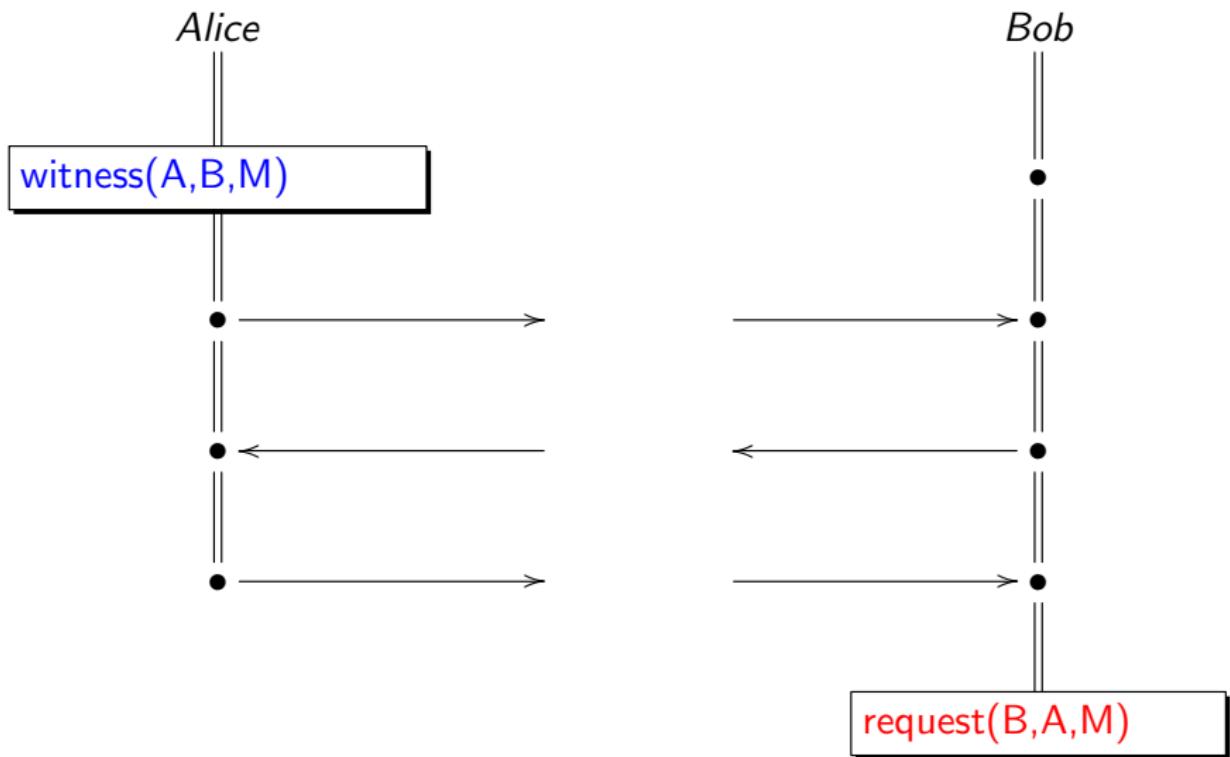
# Successful (Weak) Authentication



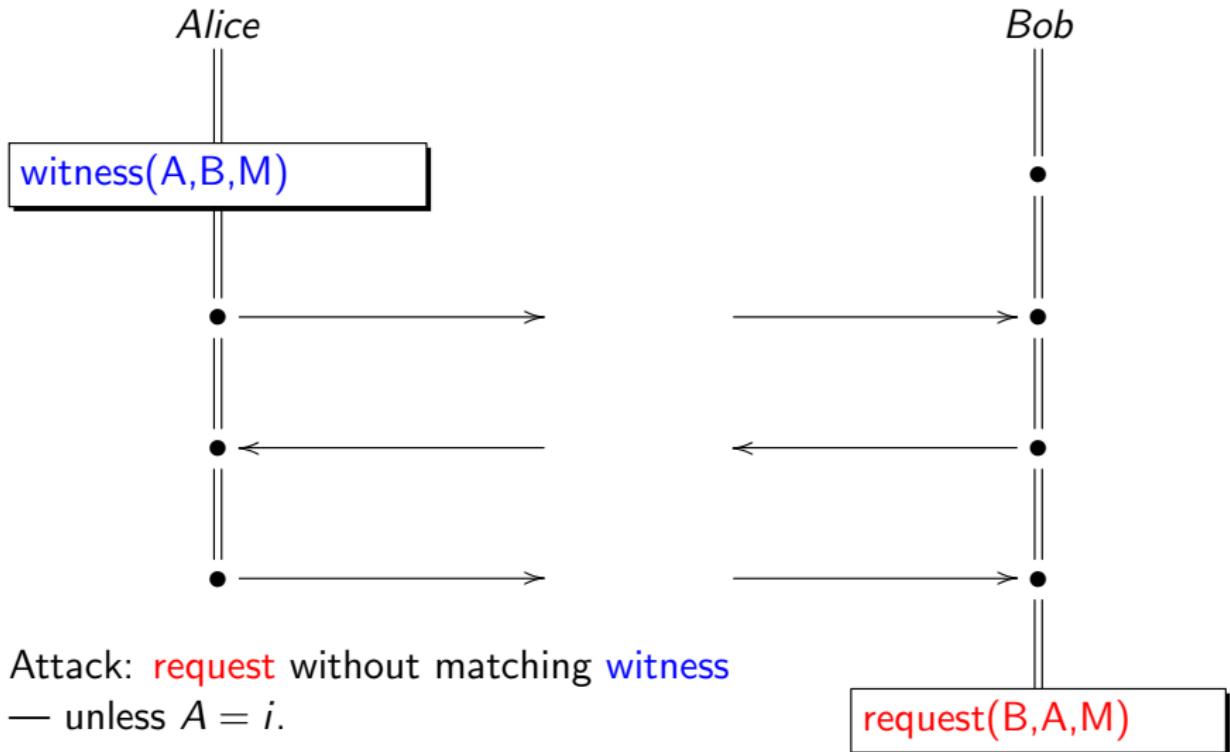
# Formalization: (Weak) Authentication



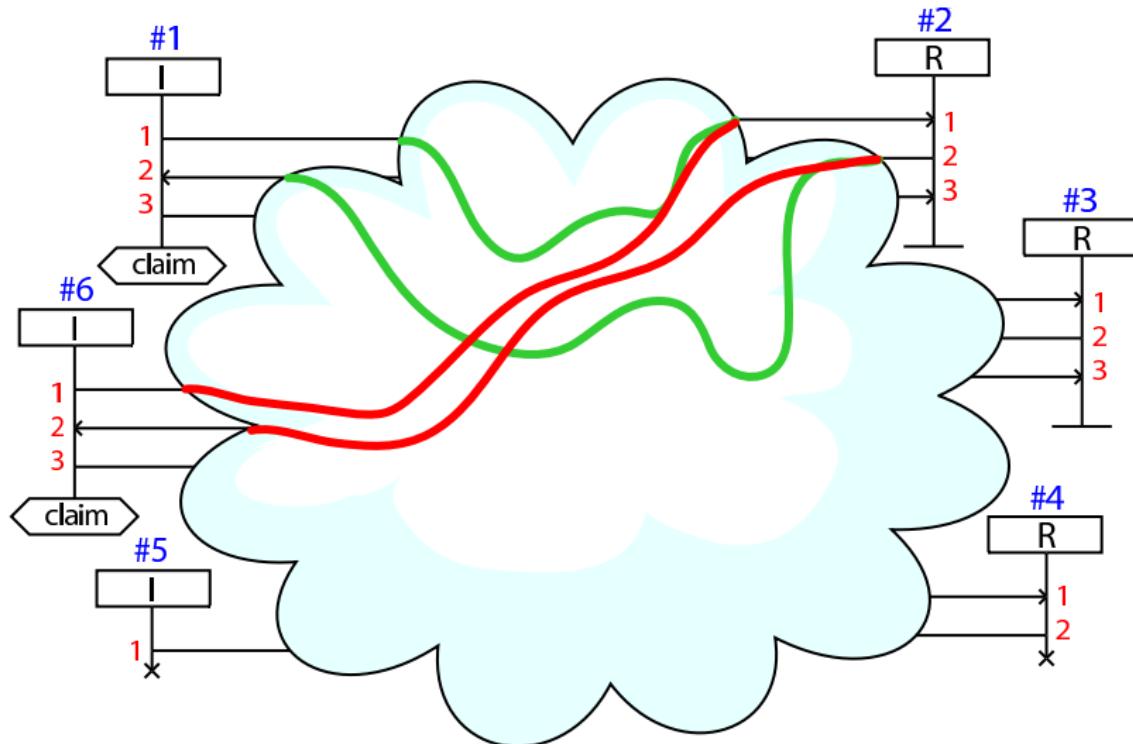
# Formalization: (Weak) Authentication



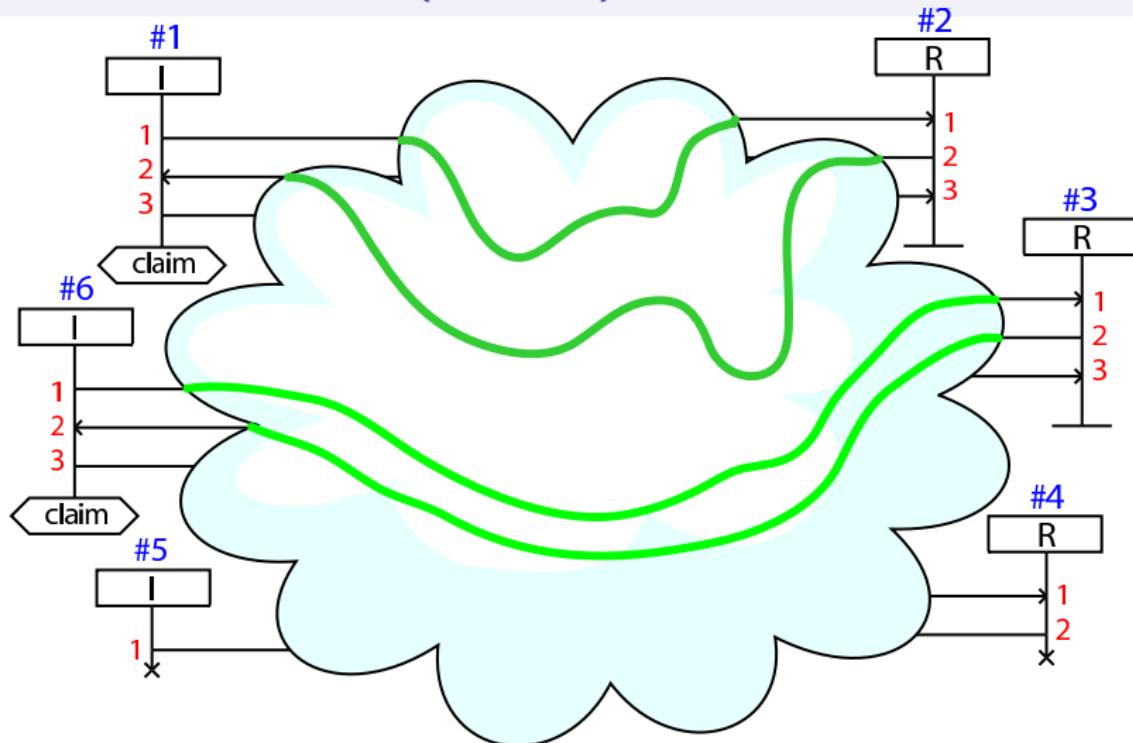
# Formalization: (Weak) Authentication



# Failed (Strong) Authentication



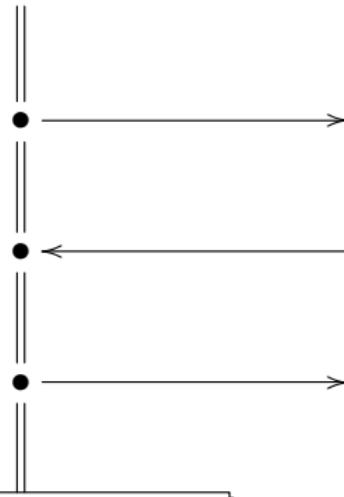
# Successful (Strong) Authentication



Attack: more **requests** than **witnesses**.

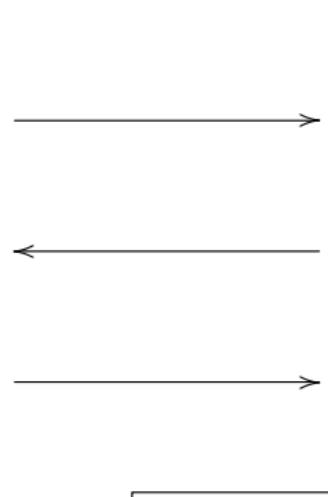
## Secrecy: M secret between A,B,C

Alice



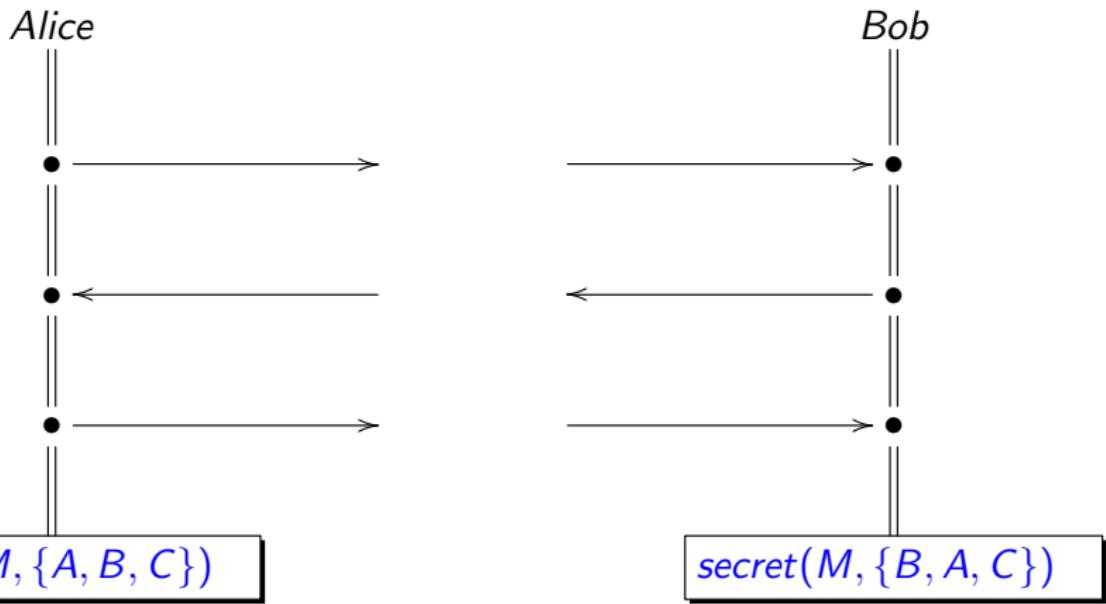
"I am A and I want M  
to be secret with B,C."

Bob



"I am B and I want M  
to be secret with A,C."

## Secrecy: M secret between A,B,C



Attack:  $\text{secret}(M, \text{Set})$ , intruder knows  $M$  and is not a member of the Set.

# Outline

- ① Assumptions and Goals
- ② Channels
- ③ TLS
- ④ Single Sign-On
- ⑤ Password Guessing Attacks

# Motivation

## Example (NSL)

$$A \rightarrow B : \{NA, A\}_{pk(B)}$$
$$B \rightarrow A : \{NA, NB, B\}_{pk(A)}$$
$$A \rightarrow B : \{NB\}_{pk(B)}$$

- Public-key cryptography is used here to ensure **confidential transmission** of messages.

# Motivation

## Example (NSL)

$$\begin{array}{ll} A \xrightarrow{\bullet} B : NA, A \\ B \xrightarrow{\bullet} A : NA, NB, B \\ A \xrightarrow{\bullet} B : NB \end{array}$$

- Public-key cryptography is used here to ensure **confidential transmission** of messages.
- Abstraction: use a **confidential channel**.

# Channel Notation

The Diffie-Hellman assumes an **authentic exchange** of the half-keys:

$$\begin{array}{lll} A & \xrightarrow{\bullet} & B : \exp(g, X) \\ B & \xrightarrow{\bullet} & A : \exp(g, Y) \end{array}$$

- How this exchange is authenticated is not relevant for Diffie-Hellman!
- Many protocols use Diffie-Hellman, e.g. Station2Station, IKE/IKEv2/JFK, Kerberos, TLS, device-pairing. . . .
- Many different ways to authenticate the key-exchange:  
**Cryptographically** Digital signatures, symmetric/asymmetric encryption, MACs.  
**Non-Cryptographically** using a trusted third party, meeting face to face, using additional channels (SMS etc.)
- Using an authentic channel abstracts from the realization.

# Channel Notation

- Channels can be both **assumptions** and **goals** of a protocol:

## Example

$$\begin{array}{lll} A & \xrightarrow{\bullet} & B : \exp(g, X) \\ B & \xrightarrow{\bullet} & A : \exp(g, Y) \\ A & \xrightarrow{} & B : \{ \textit{Payload} \}_{\exp(\exp(g, X), Y)} \\ \hline A & \xrightarrow{\bullet \rightarrow \bullet} & B : \textit{Payload} \end{array}$$

“Diffie-Hellman creates a secure channel from authentic channels.”

- Actually, public-key cryptography could be defined in a broad sense as a mechanism to obtain secure channels from authentic channels.
- Very general way to see Diffie-Hellman.
- Good for system design and verification: reason about small components with a **well-defined interface**.

# Outline

- ① Assumptions and Goals
- ② Channels
- ③ TLS
- ④ Single Sign-On
- ⑤ Password Guessing Attacks

# Transport Layer Security

TLS consists of basically two phases:

- **Handshake**: A client (typically webbrowser) and a server establish a “secure channel”: a **pair of symmetric keys** for communicating
- **Transport**: the parties exchange messages over the secure channel (encrypting with the symmetric keys).

Actually, there are some additions like re-establishing a session which we do not discuss.

## TLS 1.3 (simplified)

A->B:  $A, \exp(g, X)$

B->A:  $\exp(g, Y),$

let  $k_1 = \text{clientK}(\exp(\exp(g, X), Y))$

let  $k_2 = \text{serverK}(\exp(\exp(g, X), Y))$

{|  $\{B, \text{pk}(B)\}^{-1}(\text{pk}(s))$  |}  $k_2,$

{|  $\{h(\exp(g, X), \exp(g, Y))\}^{-1}(\text{pk}(B))$  |}  $k_2$

A->B: {|  $h(\exp(g, X), \exp(g, Y))$  |}  $k_1,$

{| data, DATA\_A |}  $k_1$

B->A: {| data, DATA\_B |}  $k_2$

where

- $h$ ,  $\text{clientK}$ ,  $\text{serverK}$  are one-way functions
- $\{B, \text{pk}(B)\}^{-1}(\text{pk}(s))$  is a key certificate issued for  $B$  by a trusted third party  $s$ .
- data is just a tag to distinguish transport messages.
- DATA\_A and DATA\_B represent payload messages transmitted on the channel.

## TLS 1.3 (simplified)

A->B:  $A, \exp(g, X)$

B->A:  $\exp(g, Y),$

let  $k1 = \text{clientK}(\exp(\exp(g, X), Y))$

let  $k2 = \text{serverK}(\exp(\exp(g, X), Y))$

{|  $\{B, \text{pk}(B)\}^{-1}(\text{inv}(\text{pk}(s)))$  |}  $k2,$

{|  $\{h(\exp(g, X), \exp(g, Y))\}^{-1}(\text{inv}(\text{pk}(B)))$  |}  $k2$

A->B:  $\{| h(\exp(g, X), \exp(g, Y)) | \} k1,$

{|  $\text{data}, \text{DATA\_A}$  |}  $k1$

B->A:  $\{| \text{data}, \text{DATA\_B} | \} k2$

- Note: only B has a certificate, but not A.
  - ★ TLS can also be deployed when both sides have a certificate.
  - ★ Typically A is a user/webbrowser that does not have a certificate.

## TLS 1.3 (simplified)

A->B:  $A, \exp(g, X)$

B->A:  $\exp(g, Y),$

let  $k1 = \text{clientK}(\exp(\exp(g, X), Y))$

let  $k2 = \text{serverK}(\exp(\exp(g, X), Y))$

{|  $\{B, \text{pk}(B)\}^{-1}(\text{inv}(\text{pk}(s)))$  |}  $k2,$

{|  $\{h(\exp(g, X), \exp(g, Y))\}^{-1}(\text{inv}(\text{pk}(B)))$  |}  $k2$

A->B:  $\{| h(\exp(g, X), \exp(g, Y)) | \} k1,$

{|  $\text{data}, \text{DATA\_A}$  |}  $k1$

B->A:  $\{| \text{data}, \text{DATA\_B} | \} k2$

- Note: only B has a certificate, but not A.
  - ★ TLS can also be deployed when both sides have a certificate.
  - ★ Typically A is a user/webbrowser that does not have a certificate.
- What kind of channel do we get without the client certificate?

## TLS 1.3 (simplified)

A->B:  $A, \exp(g, X)$

B->A:  $\exp(g, Y),$

let  $k1 = \text{clientK}(\exp(\exp(g, X), Y))$

let  $k2 = \text{serverK}(\exp(\exp(g, X), Y))$

{|  $\{B, \text{pk}(B)\}^{-1}(\text{inv}(\text{pk}(s)))$  |}  $k2,$

{|  $\{h(\exp(g, X), \exp(g, Y))\}^{-1}(\text{inv}(\text{pk}(B)))$  |}  $k2$

A->B:  $\{| h(\exp(g, X), \exp(g, Y)) | \} k1,$

{|  $\text{data}, \text{DATA\_A}$  |}  $k1$

B->A:  $\{| \text{data}, \text{DATA\_B} | \} k2$

- Note: only B has a certificate, but not A.
  - ★ TLS can also be deployed when both sides have a certificate.
  - ★ Typically A is a user/webbrowser that does not have a certificate.
- What kind of channel do we get without the client certificate?
- The intruder can impersonate the client A towards B.

## TLS 1.3 (simplified)

A->B:  $A, \exp(g, X)$

B->A:  $\exp(g, Y),$

```
let k1 = clientK(exp(exp(g,X),Y))
```

```
let k2 = serverK(exp(exp(g,X),Y))
```

```
{| {B,pk(B)}inv(pk(s)) |}k2,
```

```
{| {h(exp(g,X),exp(g,Y))}inv(pk(B)) |}k2
```

A->B:  $\{| h(\exp(g,X),\exp(g,Y)) | \}k1,$

```
{| data,DATA_A |}k1
```

B->A:  $\{| data,DATA_B | \}k2$

- Note: only B has a certificate, but not A.
  - ★ TLS can also be deployed when both sides have a certificate.
  - ★ Typically A is a user/webbrowser that does not have a certificate.
- What kind of channel do we get without the client certificate?
- The intruder can impersonate the client A towards B.
- But B has a secure channel with whoever created the secret X!
  - ★ A can be sure who B is.
  - ★ B cannot be sure who A is.
  - ★ the intruder cannot enter the connection between honest A and B.

# Secure Pseudonymous Channels

- Consider the  $\exp(g, X)$  of agent  $A$  as a **pseudonym** of  $A$ .
- The link between  $A$  and  $\exp(g, X)$  could be achieved by a certificate, but it is not available here.
- The pseudonym  $\exp(g, X)$  **cannot be stolen/hijacked** because “ownership” is the knowledge of  $x$ .
- This kind of channel is good enough for many applications such as transmitting credit card data or a login.
- We write often for this kind of channel:

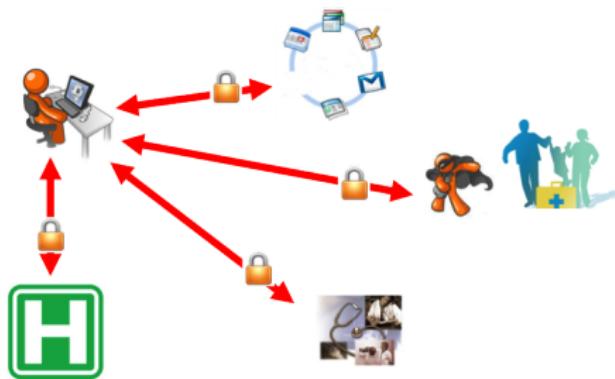
$$[A] \bullet\rightarrow\bullet B : M$$

# Outline

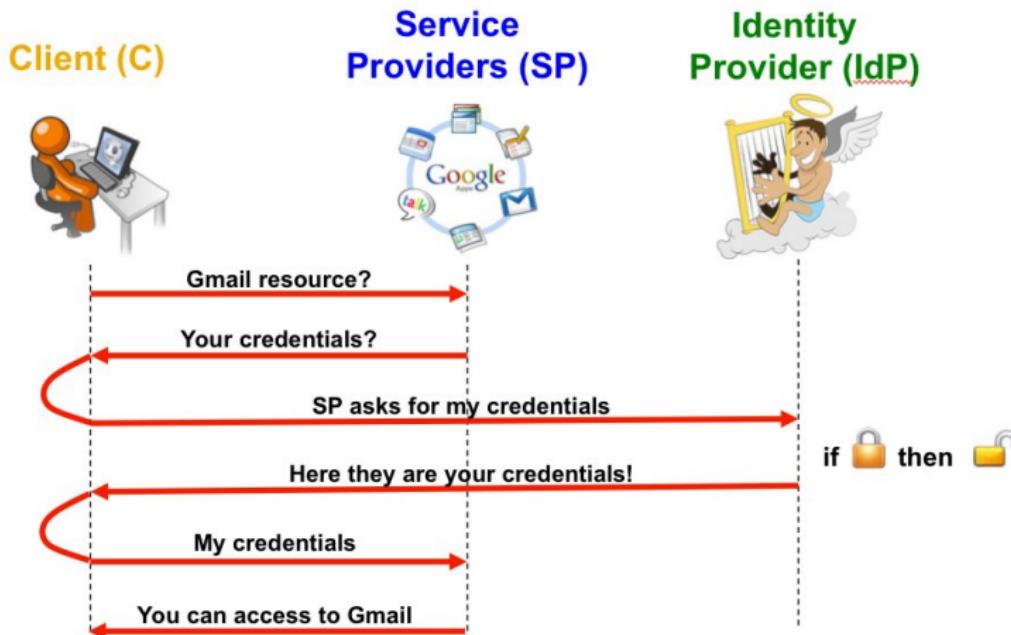
- ① Assumptions and Goals
- ② Channels
- ③ TLS
- ④ Single Sign-On
- ⑤ Password Guessing Attacks

# Single Sign-On (SSO)

- Username/passwords for different websites/accounts:
  - ★ Hassle and security problem
- Avoid this by **identity federation**:
  - ★ user has trusted **identity provider (idp)**
  - ★ user signs up only **once** at idp
  - ★ idp **vouches** for them to any website (that trusts idp)



# SSO with web-browser



# Google's SSO

Knowledge: C: C,idp,SP,pk(idp);  
idp: C,idp,pk(idp),inv(pk(idp));  
SP: idp,SP,pk(idp)

Actions:

[C] \*->\* SP : C,SP,URI  
SP \*->\* [C] : C,idp,SP,URI  
  
C \*->\* idp : C,idp,SP,URI  
idp \*->\* C : {C,idp}inv(pk(idp)),URI

[C] \*->\* SP : {C,idp}inv(pk(idp)),URI  
SP \*->\* [C] : Data

Goals:

SP authenticates C on URI  
C authenticates SP on Data  
Data secret between SP,C

# Attack on Google's SSO

The attack found by OFMC in nice notation:

1. [a] \*->\* i: a.i.URI(1)

1.' [i] \*->\* b: a.b.x306

2.' b \*->\* [i]: a.idp.b.ID(2).x306

2. i \*->\* [a]: a.idp.i.x505.URI(1)

3. a \*->\* idp: a.idp.i.x505.URI(1)

4. idp \*->\* a: {a.idp}\_inv(pk(idp)).URI(1)

5. [a] \*->\* i: {a.idp}\_inv(pk(idp)).URI(1)

5.' [i] \*->\* b: {a.idp}\_inv(pk(idp)).x306

6.' b \*->\* [i]: Data(6)

# The Problem

The authentication assertion from the ipd:

$$\{ID, C, ipd, SP\}_{\text{inv}(\text{pk}(ipd))}$$

- Google had omitted **some parts** that were suggested but not required by the standard.
- This allows a dishonest *SP* to re-use the authentication assertion and log in to other sites as *C*.

Again, this is a problem of a dishonest participant!

# Outline

- ① Assumptions and Goals
- ② Channels
- ③ TLS
- ④ Single Sign-On
- ⑤ Password Guessing Attacks

# Guessing Attacks

Example: a Variant of Microsoft-ChapV2

Protocol: PW

Types: Agent A, B;

Number NB;

Function pw, h;

Knowledge:

A: A, B, pw(A, B), h;

B: A, B, pw(A, B), h;

Actions:

B->A: NB

A->B: h(pw(A, B), NB)

Goals:

B authenticates A on NB

What if pw(A, B) has low entropy?

# Low Entropy Passwords

- Low entropy passwords
  - ★ e.g. natural language words
  - ★ short

# Low Entropy Passwords

- Low entropy passwords
  - ★ e.g. natural language words
  - ★ short
- The intruder compiles a **dictionary  $D$**  of passwords
  - ★ Relatively small space of passwords (a few million)
  - ★ Contains many weak passwords

# Low Entropy Passwords

- Low entropy passwords
  - ★ e.g. natural language words
  - ★ short
- The intruder compiles a **dictionary  $D$**  of passwords
  - ★ Relatively small space of passwords (a few million)
  - ★ Contains many weak passwords
- Use  $D$  for a **brute-force** attack on an observed message:

Observed message	$nb, h(\text{pw}(a, b), nb)$
Construct	$nb, h(\text{guess}, nb)$ for every $\text{guess} \in D$

# Low Entropy Passwords

- Low entropy passwords
  - ★ e.g. natural language words
  - ★ short
- The intruder compiles a **dictionary  $D$**  of passwords
  - ★ Relatively small space of passwords (a few million)
  - ★ Contains many weak passwords
- Use  $D$  for a **brute-force** attack on an observed message:

Observed message	$nb, h(\text{pw}(a, b), nb)$
Construct	$nb, h(\text{guess}, nb)$ for every $\text{guess} \in D$
- When  $\text{pw}(a, b) = \text{guess}$  for some guess, the constructed and the observed message are identical, and the intruder has found out  $\text{pw}(a, b)$ .

# Are weak passwords of any use?

## Knowledge:

A: A ,B ,pw(A ,B ) ,pk(B );

B: A ,B ,pw(A ,B ) ,pk(B ) ,inv(pk(B ));

## Actions :

A->B: { A ,pw(A ,B ) ,K }pk(B)

B->A: { | NB | }K

## Goals :

B authenticates A on K

A authenticates B on NB

K secret between A ,B

- This is similar to what happens in TLS-based login protocols.

# Are weak passwords of any use?

Knowledge:

A: A, B,  $\text{pw}(A, B)$ ,  $\text{pk}(B)$ ;

B: A, B,  $\text{pw}(A, B)$ ,  $\text{pk}(B)$ ,  $\text{inv}(\text{pk}(B))$ ;

Actions:

A  $\rightarrow$  B: { A,  $\text{pw}(A, B)$ , K } $\text{pk}(B)$

B  $\rightarrow$  A: { | NB | }K

Goals:

B authenticates A on K

A authenticates B on NB

K secret between A, B

- This is similar to what happens in TLS-based login protocols.
- Guessing not possible – despite low-entropy  $\text{pw}(A, B)$ 
  - ★ The intruder has to create {A,  $\text{guess}$ , K} $\text{pk}(B)$  for every guess.
    - ▶ That requires guessing K, too!

# Are weak passwords of any use?

**Knowledge:**

A: A, B,  $\text{pw}(A, B)$ ,  $\text{pk}(B)$ ;

B: A, B,  $\text{pw}(A, B)$ ,  $\text{pk}(B)$ ,  $\text{inv}(\text{pk}(B))$ ;

**Actions:**

A  $\rightarrow$  B: { A,  $\text{pw}(A, B)$ , K } $\text{pk}(B)$

B  $\rightarrow$  A: { | NB | }K

**Goals:**

B authenticates A on K

A authenticates B on NB

K secret between A, B

- This is similar to what happens in TLS-based login protocols.
- Guessing not possible – despite low-entropy  $\text{pw}(A, B)$ 
  - ★ The intruder has to create {A, *guess*, K} $\text{pk}(B)$  for every guess.
    - ▶ That requires guessing K, too!
- This is also why any reasonable implementation of asymmetric encryption contains randomization!
  - ★ {*guessable message*} $\text{pk}(B)$

# Guessing Attacks in OFMC

B → A : NB

A → B :  $h(pw(A, B), NB)$

Goals:

B authenticates A on NB

$pw(A, B)$  guessable secret between A, B

Gives attack:

GOAL:

guesswhat

...

ATTACK TRACE:

i → (x20, 1) : x205

(x20, 1) → i :  $h(pw(x20, x25), x205)$

i → (i, 17) :  $h(guessPW, x205)$

i → (i, 17) :  $h(guessPW, x205)$

# Guessing Attacks in OFMC

Implementation in OFMC:

- Say  $pw(A, B)$  is specified as a guessable secret between  $A$  and  $B$ .
- Let  $guessPW$  be a constant known to the intruder.
- Check every message sent by an honest agent if it contains  $pw(A, B)$ .
  - ★ Example: outgoing message  $h(pw(A, B), NA)$
  - ★ Then declare  $h(guessPW, NA)$  as a secret between  $A$  and  $B$ .
  - ★ If the intruder is able to violate this secrecy goal, then he can make a guessing attack.
- Additionally, if the password is used for symmetric encryption, then guessing the key is sufficient:
  - ★ Example:  $\{m\}_{h(pw(A,B),NA)}$
  - ★ Then also  $h(guessPW, NA)$  is a secret between  $A$  and  $B$ .

## Cryptography I



DTU Compute

Department of Applied Mathematics and Computer Science

$$\int_a^b \varepsilon^{f^{(n)}(x)} \Theta^{\sqrt{17}} + \Omega^{\int \delta e^{i\pi} =} = \{2.7182818284 \dots\}$$

## What Cryptography can do

- Cryptography is just one of the tools in the security tool box
  - But a very versatile and powerful tool
- Cryptography can help solve important problems:
  - Keeping secrets
    - *Messages on the network*
    - *Data stored on disks, memory cards, USB sticks, ...*
  - Ensuring integrity
    - *Messages on the network (Message Authentication Codes)*
    - *Data stored on disk, ...*
  - Authentication
    - *User authentication*
      - Protecting shared secrets during communication
    - *Message authentication*
      - Sender authentication
      - Message authentication

## Keeping Secrets

### Steganography, Codes and Ciphers

- Steganography

- Hide the existence of a secret message
- Inconspicuous message (invisible ink, micro dots)
- If the message is found, then the secret is revealed



- Codes

- Replace symbols in the message with "codes"
- Transformation must be agreed in advance
- Inconspicuous if code is well defined



- Ciphers

- Hide the meaning of the secret message
- Scrambling according to an agreed algorithm
- Conspicuous message (obviously a secret)



3

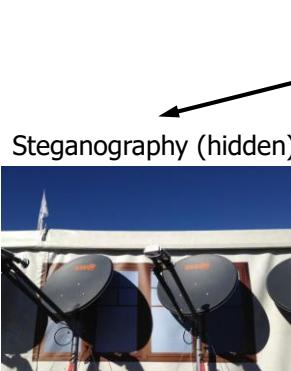
DTU Compute Technical University of Denmark

Master of Cybersecurity

28/8/2021

## Cryptography

### Fundamental principles



Code (replace "words")

Cipher (replace "letters")

4

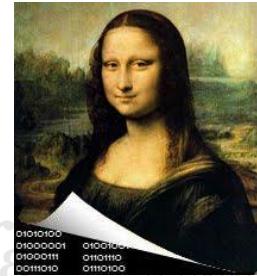
DTU Compute Technical University of Denmark

Master of Cybersecurity

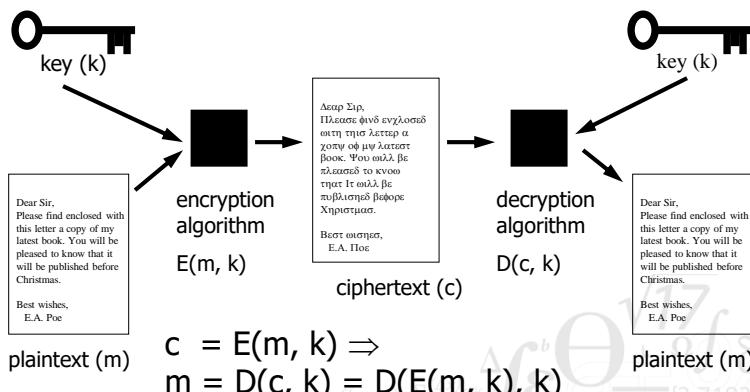
28/8/2021

## Steganography

- Classic example from antiquity
- Xerxes' invasion of Greece 480 BC
  - Persian build-up observed by Greek citizen Demaratos
  - Sent a warning to Athens on wax-plates
    - *Scraped of the wax*
    - *Engraved the warning in the wood*
    - *Covered the wood with new wax, so the plates looked unused*
    - *Warning safely reached Athens*
  - Xerxes' navy was defeated at Salamis
- Current methods for steganography now often uses images or sound
  - Message encoded in least significant bit of pixel or sample
  - Cover message (e.g. image) can be uploaded to public forum



## Cipher = Algorithm + Key



No cipher should rely on the secrecy of the algorithm!

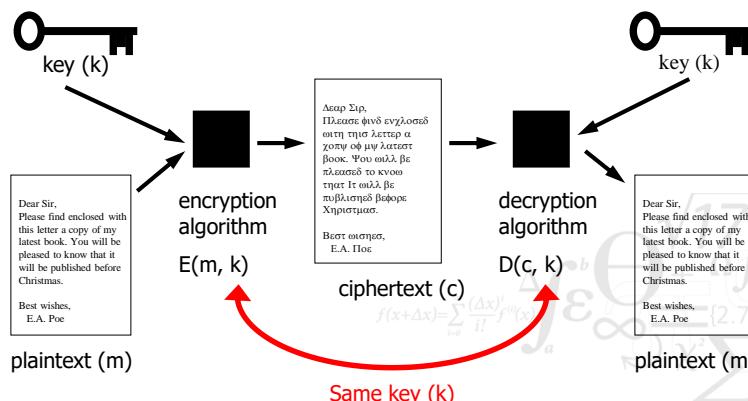
A. Kerckoffs, "La Cryptographie Militaire", 1883

## Basic Building Blocks of Cryptography

- Symmetric ciphers (1 key is used)
  - Same key is used for encryption and decryption
- Asymmetric ciphers (2 keys are used)
  - One key is used for encryption
  - Another key is used for decryption
    - *It is not computationally feasible to derive one key from the other*
- Cryptographic Hash Functions (no keys are used)
  - Scrambles input in a unique way
    - Generates fixed length output from variable length input
    - Scrambles input ("decryption" is infeasible)
- Digital signatures
  - One key signs data (private key)
  - Another key is used to verify signature (public key)
- Advanced algorithms, protocols and constructs not covered here

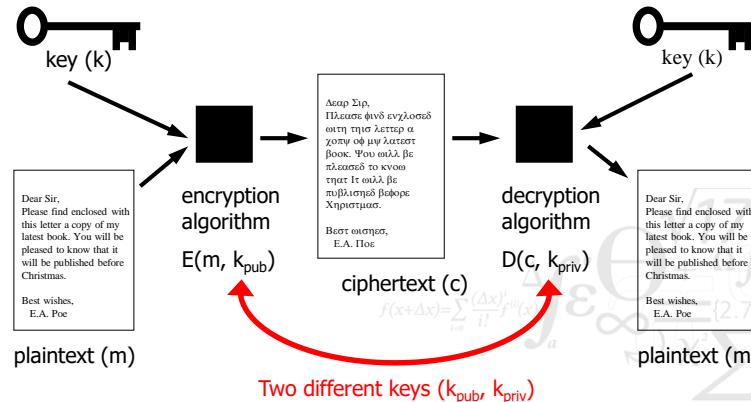
## Symmetric Cryptography

- Decryption-key is identical to Encryption-key (or easily derived)



## Asymmetric Cryptography

- Decryption-key cannot be derived from Encryption-key



9

DTU Compute Technical University of Denmark

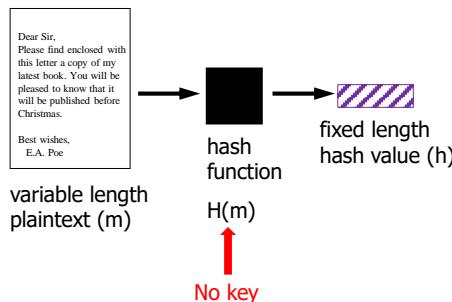
Master of Cybersecurity 28/8/2021

## Cryptographic Hash Functions

- Cryptographic hash functions must also satisfy:

- Given  $M$ , computation of  $h$  is easy
- Given  $h$ , it is intractable to compute  $M$  such that  $H(M) = h$
- It is intractable to find  $M$  and  $M'$  such that  $H(M') = H(M)$

- Hash value  $h$  is often called a “fingerprint” or “digest” of  $M$



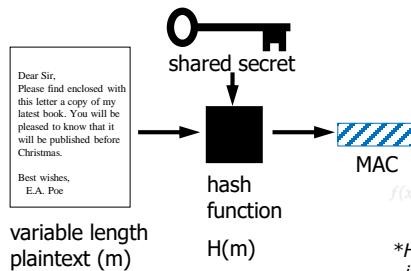
10

DTU Compute Technical University of Denmark

Master of Cybersecurity 28/8/2021

## Message Authentication Codes (MAC)

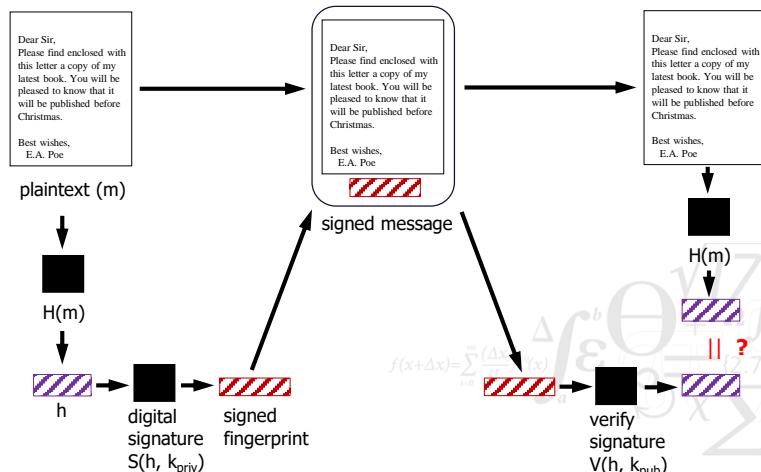
- Message Authentication Codes protect integrity of messages
- Hash-based MAC uses a cryptographic hash function and a shared secret
  - Shared secret (K) is prepended to the message (M) before applying H
  - $\text{MAC}(K, M) = H(K \parallel M \parallel K)^*$



\*HMAC is a frequently used MAC function; it is defined in RFC 2104

## Digital Signatures

- Operation is expensive, so we normally sign the fingerprint



## Security Properties

- Symmetric cryptography
  - Confidentiality of messages
  - Both parties know the shared key (may be more than two)
- Asymmetric cryptography
  - Confidentiality of messages
  - Only one party knows the secret key (only party who can decrypt  $M$ )
- Hash functions
  - Hash value corresponds to given  $M$  (with very high probability)
- Message Authentication Codes
  - Integrity of message (hash function)
  - Authenticity of message (shared secret)
- Digital signatures
  - Integrity and Authenticity (as with MAC)
  - Non-repudiation of message (assuming security of private-key)

## Protection Goals

- Before considering a cryptographic solution, get clear about its protection goals:
  - Confidentiality
  - Integrity
  - Authenticity
  - Non-Repudiation
- People often say: "Our system provides secure communication" or "we need a secure communication channel"  
What does this really mean?
- Cryptography is expensive, so we should only use the building blocks that we need

## Characteristics of Good Ciphers

- Defined by Claude Shannon (1949)
  - The father of Information Theory



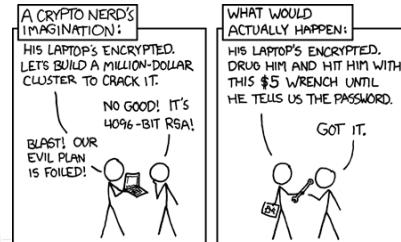
1. The amount of secrecy needed should determine the amount of labour appropriate for the encryption and decryption
2. The set of keys and the enciphering algorithm should be free from complexity
3. The implementation of the process should be as simple as possible
4. Errors in ciphering should not propagate and cause corruption of further information in the message
5. The size of the enciphered text should be no larger than the text of the original message

## Security of Cryptographic Solutions

- There are 3 classes of attacks on crypto-systems
  - Attacking the cipher (algorithm and mode)
  - Attacking the key (key space, key generation, key management)
  - Attacking the cryptographic protocol
- Most crypto-systems are "computationally secure"
  - Security often measured in number of operations required by the best known attack (this is known as the "workload" of an attack)
    - *Strong algorithm, brute force attack*  $\Rightarrow$  workload = key-length/2

## Cryptanalysis attacking ciphers

- Cryptanalysis attempts to recover plaintext without access to key, but with full knowledge of algorithm
- There are four general types of cryptanalytic attacks:
  - Ciphertext-only attack
  - Known-plaintext attack
  - Chosen-plaintext attack
  - Adaptive chosen-plaintext attack
- Other attacks are equally effective
  - Rubber-hose cryptanalysis aka. purchase-key attack



f(x+Δx)=  
<https://xkcd.com/538>

## Cryptanalysis attacking keys

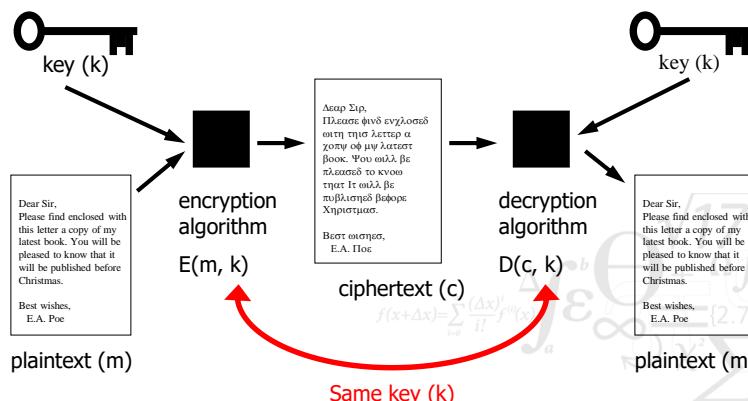
- Key space attacks
  - Exhaustive key search (brute force) attacks are possible with short keys
    - DES has fixed 56 bit keys, which are cracked very quickly
    - AES uses 128, 192 or 256 bit keys (algorithm allows longer keys)
  - Asymmetric algorithm keys are an order of magnitude longer
    - Elliptic Curve crypto. achieves equivalent security with shorter keys
- Key generation attacks
  - 128 bits = 16 random bytes are hard to remember
  - Key derivation functions (KDF) take a password and generates a key
    - Guessing the password ⇒ knowing the key
- Key management
  - Keys must be stored, shared, distributed, ...
    - All these steps may be attacked

## Coffee Break



## Symmetric Cryptography

- Decryption-key is identical to Encryption-key (or easily derived)



## One Time Pads

- A One Time Pad consists of a large non-repeating sequence of *truly random* characters
- Encryption xor each letter in the plaintext with the corresponding letter from the one time pad

$$C = P \oplus K$$

- Decryption xor each letter in the ciphertext with the corresponding letter from the one time pad

$$P = C \oplus K$$

- One time pads produce perfectly secure encryption

- Known as information-theoretic security, cryptosystem cannot be broken by attacker with unlimited resources

- *Derived from the work by Claude Shannon*

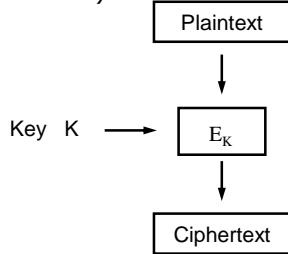


## Symmetric Cryptography

- Two main classes of symmetric cryptography:
  - Block Ciphers
  - Streams Ciphers
- Well known Block Ciphers include:
  - DES (badly broken, but found in older textbooks)
    - *Keys are too short to protect against brute force*
  - Triple DES (3DES, still in use, but will be retired by 2023)
    - $C = E_{K3}(D_{K2}(E_{K1}(P)))$
  - AES (current standard adopted by NIST)
- Well known Stream Ciphers include:
  - A5/1 (used in GSM networks, essentially broken)
  - RC4 (difficult to use securely)
  - Block ciphers can be used to construct stream ciphers
    - *This is often the better option*

## Block Cipher Algorithms

- Block ciphers operate on blocks of plaintext and ciphertext (usually 32, 64 or 128 bits)



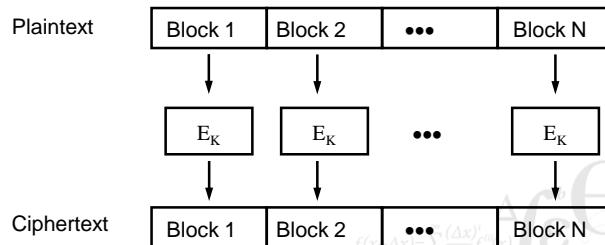
- Algorithm describes how to transform one block of a certain size
  - What happens when plaintext is shorter than block size of algorithm?
  - What happens when plaintext is longer than block size of algorithm?
- Introduces encryption schemes/modes

## Algorithms and Schemes/Modes

- Cryptographic algorithms
  - Describe transformation of plaintext/ciphertext to ciphertext/plaintext
  - Do not describe how cryptography is used in systems
- Algorithms operate on fixed sized data
  - Blocks, stream units (bits/bytes)
- What happens when message size does not fit algorithm size?
  - Message is padded to fit algorithm size
- Cryptographic schemes/modes define:
  - How to encrypt plaintexts of arbitrary lengths
  - How to use initialisation vectors to make each encryption unique

## Encrypting Long Plaintexts

- Divide the plaintext into blocks of the defined block size
  - Add padding to the end if necessary



## Electronic Codebook Mode

- ECB is the simplest application of a symmetric block cipher
  - plaintext blocks are separately encrypted into ciphertext blocks
    - *Figure on the previous slide*
  - the same plaintext block is always encrypted into the same ciphertext block (with the same key)
    - *Substitution of one "letter" for another "letter" => cipher*
      - Size of AES alphabet is  $2^{128}$
- Properties of ECB encryption
  - blocks can be en-/decrypted in random order
    - *ECB used in encrypted file systems means that the "seek" operation works*
      - Does require block alignment, but disk blocks are typically multiples of 32, 64 or 128 bits (e.g., 512B, 1024B or 4096B)
    - blocks can be en-/decrypted in parallel
      - *Useful in high speed network communication*

## Problems with ECB

- Structure in the plaintext is reflected in the ciphertext



Tux



ECB encoding



Non-ECB encoding

- Messages often have stereotyped beginnings (Dear Foo) and endings (Best regards, Bar)
- Cryptanalyst who learn the encryption of a plaintext block can decrypt the corresponding ciphertext block in all messages encrypted with the same key
- ECB is also vulnerable to *block replay attacks*

## Block Replay Attack

- Mallory has access to the network
- He deposits money several times in the Receiving bank (and looks for duplicates)
- He deduces the sending banks encoding of his name, account no. and the amount
- He can now modify other people's money transfers

Block number

1	2	3	4	5	6	7	8	9	10	11
Time-stamp	Sending bank	Receiving bank	Account holder's name			Account number	Amount			

Field

NB! You don't need the key to modify the message  
Encryption does not protect integrity

## Cipher Block Chaining Mode

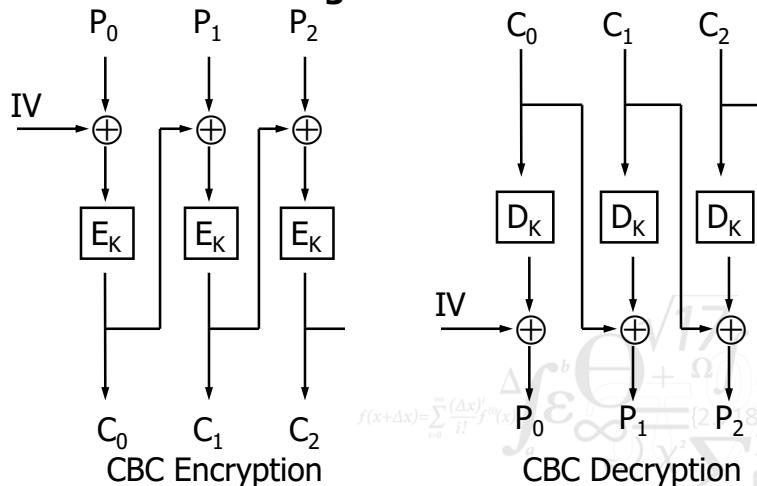
- Introduce a feedback mechanism
  - include the previous ciphertext block in the encryption of the current plaintext block

$$C_i = E_k(P_i \oplus C_{i-1})$$

$$P_i = C_{i-1} \oplus D_k(C_i)$$

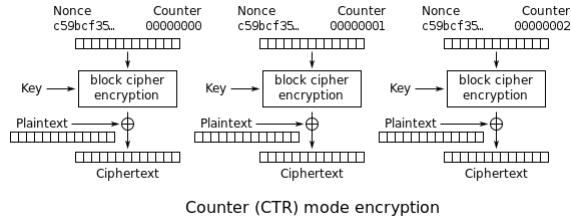
- An initialization vector (IV) is used to start the process (the IV need not be secret, but must not be reused)
  - Nonce, sequence number, date, ...
  - IV may be public, i.e., known to the attacker

## Cipher Block Chaining Mode



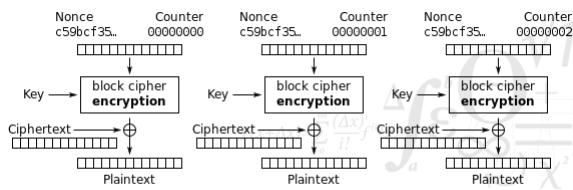
## Counter Mode

- Encryption



Counter (CTR) mode encryption

- Decryption

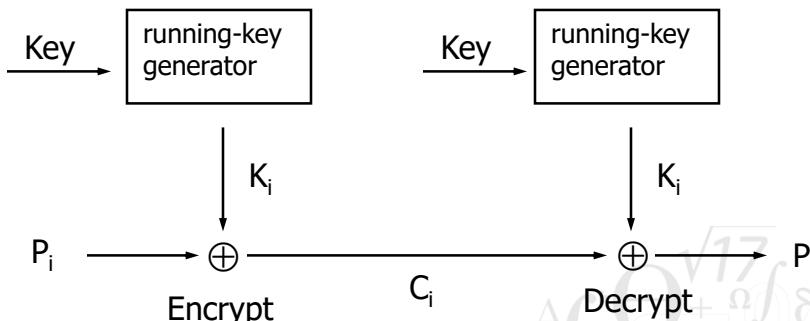


Counter (CTR) mode decryption

## Stream Ciphers Algorithms

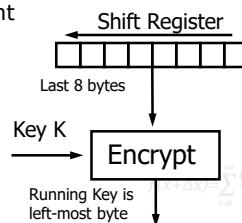
- Stream ciphers work on 1 bit/byte at the time
- A running-key generator generates a pseudo-random string of bits used for encryption
  - same running-key every time means that cryptanalysis is trivial
  - completely random running-key is equivalent to a one time pad (complete security, but not possible)
  - the key is used to seed the running-key generator
- Stream ciphers do not require an entire block to work
  - Good for character based applications

## Stream Ciphers II



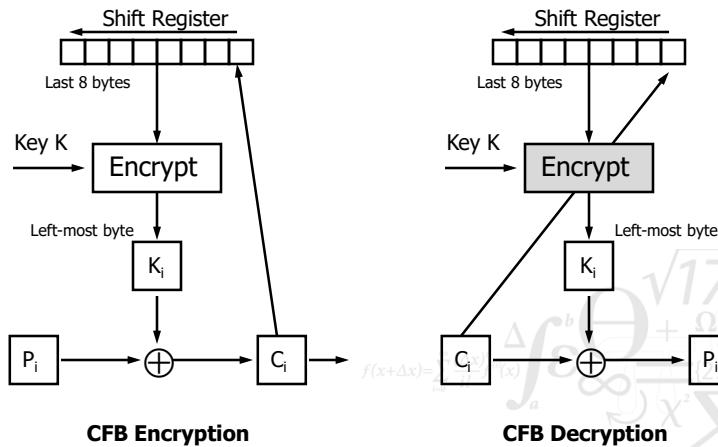
## Creating Stream Ciphers from Block Ciphers

- Block ciphers can be used to implement stream ciphers
- Example: a 64bit block cipher is used to implement a byte stream cipher
  - A "shift register" (64bit) is encrypted and the leftmost byte is XORed with the plaintext byte
  - the shift register is shifted 1 byte to the left and the ciphertext byte is added to the right

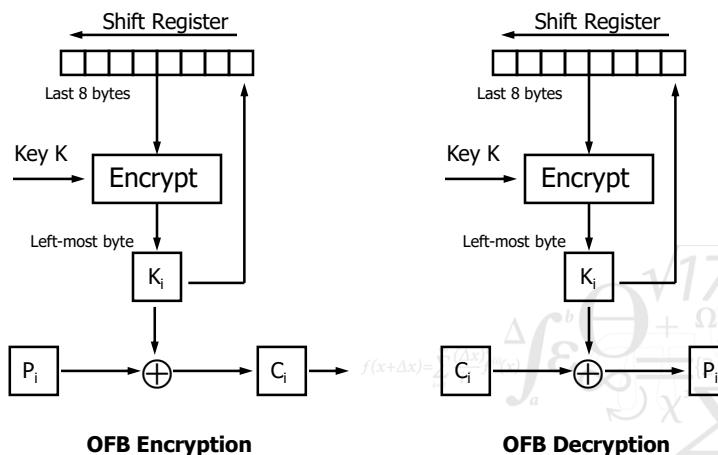


- The IV must be unique, e.g., a serial-number

## Cipher Feedback Mode

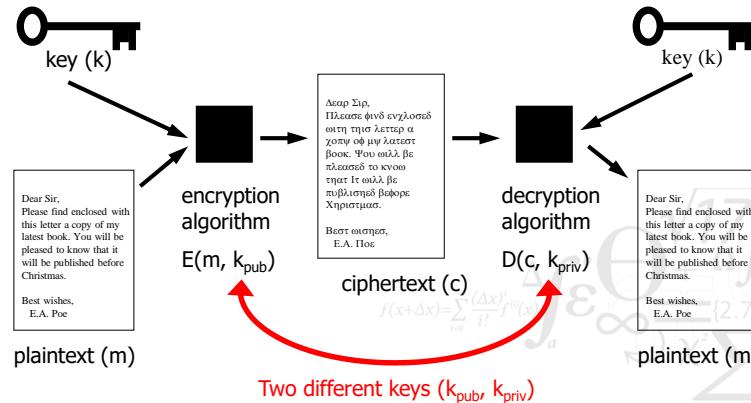


## Output Feedback Mode



## Asymmetric Cryptography

- Decryption-key cannot be derived from Encryption-key



## Limitations

- Compared to symmetric cryptography, asymmetric cryptography is inefficient (slow, long keys)
  - In practice it is not used to encrypt large amounts of data
    - Building block for key exchange*
    - Building block for other protocols*

## Sample Asymmetric Encryption Algorithms

- Asymmetric encryption is based on computationally hard problems
  - Problems that we cannot solve efficiently on computers
- Important groups of asymmetric encryption algorithms
  - RSA (prime factorization of large numbers)
  - Diffie-Hellman (modular arithmetic)
  - Diffie-Hellman (elliptic curves)
- Asymmetric encryption schemes define how algorithms are used in practice, i.e., arbitrary length messages, padding, etc.
- Particularly important: Semantic Security  
Encryption scheme has to randomize the message, since otherwise, the following attacks become possible
  - Guess the message's content (guess  $M$  that corresponds to given  $C$ )
  - Use (known) public key to check whether the guess is correct

## RSA

- Most popular public-key cryptosystem
- 1977: Rivest, Shamir, Adleman (RSA)
- Both encryption and authentication
- Survived years of attacks (cryptanalysis)
  - Probably secure
- Part of many official standards worldwide
  - Interoperability with existing code base



## RSA overview

- Based on the difficulty of factorization
- Pick two random large primes: p and q
- Calculate  $n = pq$
- Choose random encryption key  $e$ 
  - $e$  and  $(p-1)(q-1)$  must be relatively prime
- Compute decryption key  $d$  (extended Euclidian algorithm), such that
$$ed \equiv 1 \pmod{(p-1)(q-1)}$$
- Public key =  $(e, n)$  Private key =  $d$

## Using RSA

- Encryption of a plaintext  $M$ 
  - divide  $M$  into numerical blocks  $m_i < n$
  - $c_i = m_i^e \pmod{n}$
- Decryption of ciphertext block  $c_i$ 
$$m_i = c_i^d \pmod{n}$$
because
$$c_i^d = (m_i^e)^d = m_i^{ed} = m_i^1 \pmod{n}$$

## ElGamal

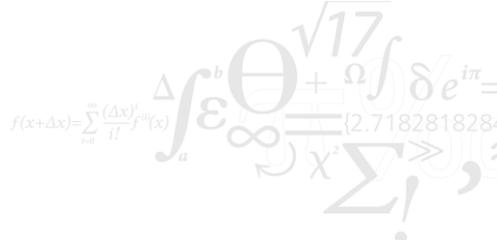
- Based on the difficulty of calculating discrete logarithms in a finite field

$$y = g^x \bmod p$$

- $p$  is prime
- $g$  and  $x$  are random numbers less than  $p$

- Public key =  $(y, g, p)$

- Private key =  $x$



## ElGamal Encryption

### Encryption of message $M$

- Select random  $k$  relatively prime to  $p - 1$

$$a = g^k \bmod p$$

$$b = y^k M \bmod p$$

- The pair  $(a, b)$  is the ciphertext

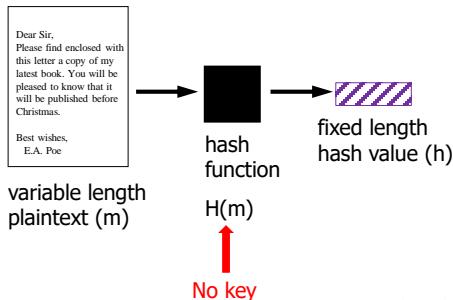
### Decryption of $a$ and $b$

$$M = b/a^x \bmod p$$

because

$$a^x \equiv g^{kx} \pmod{p} \wedge b/a^x \equiv y^k M/a^x \equiv g^{kx} M/g^{kx} \equiv M \quad (\text{all is mod } p)$$

## Cryptographic Hash Functions



- Cryptographic hash functions must also satisfy:

1. Given  $M$ , computation of  $h$  is easy
2. Given  $h$ , it is intractable to compute  $M$  such that  $H(M) = h$
3. It is intractable to find  $M$  and  $M'$  such that  $H(M') = H(M)$

- Hash value  $h$  is often called a “fingerprint” or “digest” of  $M$

## Hash Functions

- The “work horses” of cryptography
- Main uses include:
  - Condensing long strings into short strings (collision resistant hash function)
  - Making an irreversible transformation without a key (one-way function)
- Prominent Examples:
  - MD4, MD5, SHA-0 (badly broken)
  - SHA-1 (still found in standards, but practically broken since 2020)
  - SHA-2 (current standard)
  - SHA-3 (newest standard – since 2015)

## Collision Resistance

- Intractable to find random  $M$  and  $M'$  such that

$$H(M) = H(M')$$

- Collisions invalidates the use of hash functions in digital signatures
  - Alice creates two contracts  $M$  and  $M'$ , where  $M$  is fair, but  $M'$  is favourable to the Alice
  - Bob signs  $M$ :  $\text{Sign}(H(M), \text{Bob}_{\text{Priv}})$
  - Since  $H(M) = H(M')$  Alice can present  $M'$  as if it was signed by Bob

## Birthday Paradox

- Standard statistics problem

How many people do you need in a room to have better than 50% chance of two persons with the same birthday?

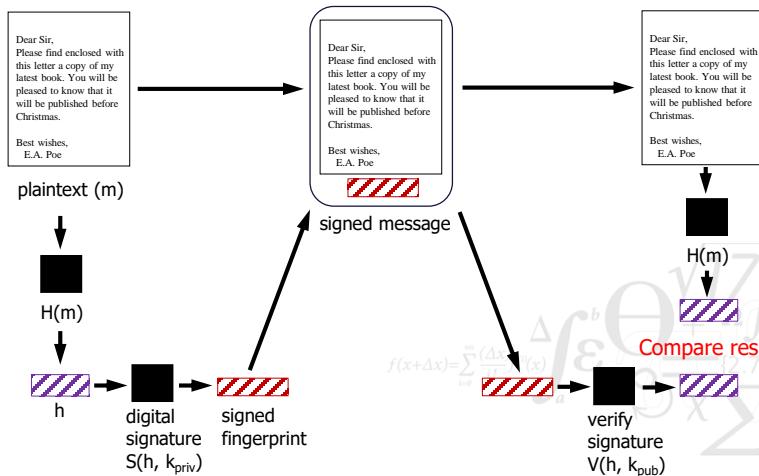
- someone with your birthday 253
- any two with the same birthday 23
- Any two born of the same day of the month 10

- $m$  bit hash function

- $2^m$  random hashes needed to find a particular  $h$
- $2^{m/2}$  random hashes needed to find two messages with the same hash value

## Digital Signatures

- Operation is expensive, so we normally sign the fingerprint



## Security Level

- If the best known attack is equivalent to running the cryptographic algorithm  $2^n$  times, then we have a security level of  $n$  bit
- Typical 80 bit (too low) 128 bit (decent), 256 bit (high)
- Often, the security level is equal to the key length
- Important exceptions:
  - Hash functions (hash size  $\geq 2 \times$  security level)
  - Asymmetric cryptography (e.g. RSA: 1024/2048/4096 bit)

## Summary on Building Blocks

- Please remember the following
  - Clarify your security goals
  - Don't design your own cryptographic primitives and protocols
    - Always rely on well known (and analysed) standards
  - Make sure that you understand the primitives and protocols you use:
    - Security goals
    - Security level
    - How to apply them
    - Known problems and pitfalls

$$f(x+\Delta x) = \sum_{l=0}^{\infty} \frac{(\Delta x)^l}{l!} f^{(l)}(x)$$
$$\Delta \int_a^b \Theta^{17} + \Omega \int \delta e^{i\pi} =$$
$$\mathcal{E}_{\infty} = [2.718281828]$$
$$\sum x^2 \gg !,$$



## Cryptography II



DTU Compute

Department of Applied Mathematics and Computer Science

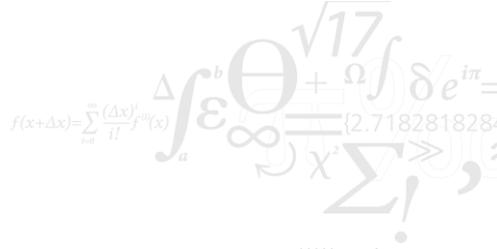
$$\Delta \int_a^b \varepsilon^{\sqrt{17}} \Theta + \Omega \int \delta e^{i\pi} = f^{(0)}(x) \infty = \{2.71828182845904523536028747135266249 \dots\}$$

## Encrypted Communication

- A protocol defines a sequence of steps involving several parties
- Characteristics of a protocol
  - Everyone knows the protocol in advance
  - Everyone agrees to follow the protocol
  - The protocol must be unambiguous
  - The protocol must be complete
- Typical “actors” in description of cryptographic protocols
  - Alice and Bob (parties who wish to communicate securely)
    - *Charlie and Dave are often added for multiparty protocols*
  - Eve (a malicious agent who wishes to eavesdrop on communication)
  - Mallory (a malicious attacker of any kind)
  - Trent (a trusted arbitrator – Trusted Third Party (TTP))

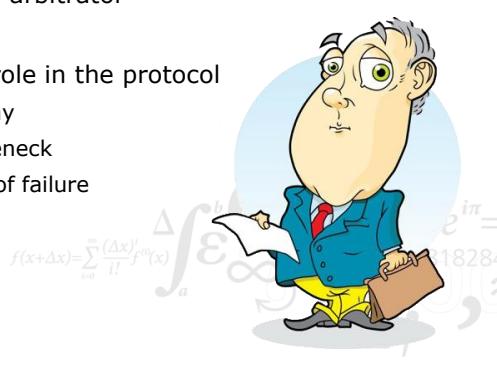
## Cryptographic Protocols

- Cryptographic protocols
  - Describe how multiple parties employ cryptography together
    - *How each party uses the algorithms*
    - *How cryptographic keys are managed*
      - Key generation (not always covered in the protocol description)
      - Key distribution
- Three types of protocols
  - Arbitrated protocols
  - Adjudicated protocols
  - Self-enforcing protocols



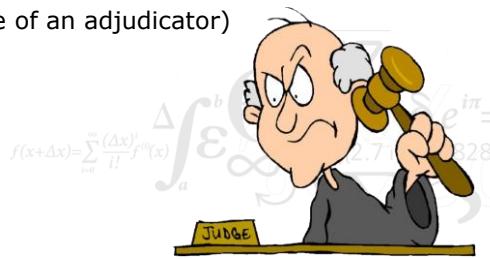
## Arbitrated Protocols

- An arbitrator is an uninterested trusted third party (e.g., lawyers, notaries, banks)
- All parties must trust the same arbitrator
- The arbitrator plays an active role in the protocol
  - the arbitrator introduces a delay
  - the arbitrator becomes a bottleneck
  - the arbitrator is a single point of failure



## Adjudicated Protocols

- The costs of arbitration can be reduced by subdividing the protocol in two
  - 1) A non-arbitrated protocol to do transactions
  - 2) An arbitrated protocol to resolve disputes
- The non-arbitrated protocol must provide non-disputable evidence of the transaction
- The adjudicator is only invoked in case of disputes (a judge in the legal system is a good example of an adjudicator)



5 DTU Compute Technical University of Denmark

02239 Data Security

## Self-Enforcing Protocols

- The protocol itself guarantees fairness
- It must be designed so that there cannot be any disputes
- Attempts to cheat must be detected before damage is done and the protocol stopped
- Self-enforcing protocols are preferred whenever possible
- Do you know examples of self-enforcing protocols?

## The Enforcer



6 DTU Compute Technical University of Denmark

02239 Data Security

## Encrypted Communication in Practice

- Handshake (taking the initial steps)
  - Agree on cryptographic protocol, including:
    - Algorithms and Modes
    - Cryptographic session keys
  - Authentication of Communicating Parties
    - Single-side authentication (typically server)
    - Mutual authentication (both parties are authenticated)
- Communication (steady state operation)
  - Send and receive messages securely
    - Confidentiality protected through encryption
    - Integrity protected through MAC or digital signatures
- Connection termination
  - Delete session specific state (keys, cookies, ...)

## Usage of Cryptographic Keys

- The following is a categorisation of cryptographic keys according to what they are used for:
  - **Data key:** Directly used for cryptographic purposes, e.g. encryption or authentication
  - **Key-encryption key:** Used to encrypt other keys, e.g. in key exchange or key storage
  - **Master key:** Used to generate other keys, e.g. in key derivation function (KDF).

Example:  $\text{Session\_Key} := \text{KDF}(\text{Master\_Key}, \text{Session\_Number})$

## Cryptographic Key Management

- Key Generation
- Key Distribution
  - Key Exchange
    - *One party generates the key*
    - Requires key transport to all other parties
  - Key Agreement
    - *All parties influence the generation of the key*
    - Diffie-Hellman algorithm popular for two parties
- Perfect Forward Secrecy (PFS)
  - Ensures that a session key derived from a set of long-term keys cannot be compromised if one of the long-term keys is compromised in the future

## Key Generation

- Cryptographic keys must be intractable to guess
  - All possible cryptographic keys should be equally likely
- Typical key generation methods include:
  - Password Based Key Derivation Function
    - *generates a cryptographic key from a readable string*
  - Random number generator (RNG)
    - *Statistical RNG, not cryptographically secure*  
*Never use this for Cryptographic purposes!*
    - *Pseudo-random number generator (PRNG)*  
*Must be seeded correctly, so use with care*
    - *True random number generator*  
*Uses measurements of physical processes to generate "real"*  
*randomness - too expensive for most applications*

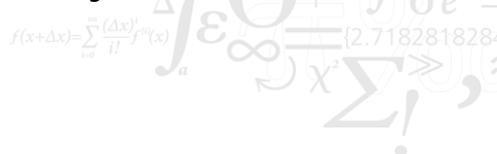
True random numbers are available online: <https://www.random.org>

## Key Exchange Requirements

- In addition to being generated, the key must also be distributed to all legitimate parties
  - How to prevent others from seeing the key?
  - How to authenticate the legitimate parties (sender and receiver)?
  - How to distribute the key to the legitimate parties?
  - How to ensure that the key is fresh?
  - How to verify that the legitimate parties received the key?

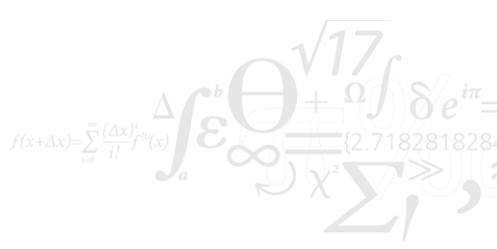
If done remotely: Use cryptography (many different solutions)

Sometimes easier: Personal key exchange

$$f(x+\Delta x) = \sum_{n=0}^{\infty} \frac{(\Delta x)^n}{n!} f^{(n)}(x)$$


## Key Exchange Techniques

- For key exchange we have the following options:
  - Generated by all parties working together (key agreement)
  - Generated by one party, then published in a public place Public Key Infrastructure (PKI)
  - Generated by one party, then sent to the other (key transport)
  - Generated by a trusted third party and sent to all parties TTP key transport

$$f(x+\Delta x) = \sum_{n=0}^{\infty} \frac{(\Delta x)^n}{n!} f^{(n)}(x)$$


## Key Expiration

- Keys can (in fact: should) expire sometime. Problems include:
  - How to keep track of key expiration?
  - How to inform all users of key expiration?
  - How to set up a new key?
  - What happens after expiration?
    - *Archive old key material? How?*
    - *Delete old key material? How? Remember to delete all copies!*

## Key Compromise

- Worst case: Key has been compromised because
  1. An attacker has potentially had access to the key
  2. The corresponding cryptographic algorithm was broken
- What do we have to do?
  - Key must no longer be used in the future
    - *Key expiration (see above)*
  - All concerned parties have to be informed!
    - *Key Revocation*
  - Old data has to be protected
    - *Re-encryption? Re-signing?*
    - *Destruction of all copies of old data*

## Symmetric-Key Cryptography Protocol

- The following steps are required to setup communication using SKC
  - 1) Alice and Bob agree on a cryptosystem
  - 2) Alice and Bob agree on a key
  - 3) Alice encrypts her plaintext with the key
  - 4) Alice sends the ciphertext to Bob
  - 5) Bob decrypts the ciphertext with the key

*How can these steps be compromised?*

## Problems of SKC

- Keys must be distributed in advance
- Keys must be kept secret from non-participants in the protocol
- A compromised key reveals all information encrypted with that key
  - Keys can be compromised:
    - Weak keys (e.g. PRNG or KDF is weak) can be guessed
    - Man In The Middle (MITM)
    - Compromising hosts
      - Probability of a host is compromised is  $p$ , probability of their shared key is compromised is  $2p$
- Both parties have the same key
  - A compromised key can be used to masquerade as either party
  - SKC cannot be used directly in adjudicated protocols
- Different keys are needed for every pair of communicating parties
  - $n$  users require  $n(n - 1)/2$  keys

## Key-Exchange Protocols

- Assume a Key Distribution Center (KDC)
- Assume all participants share a key with KDC
- Shared-Key (symmetric) Protocols
  - Alice → KDC, request session key to talk with Bob
  - KDC generates Key, returns  $K_A(K_{AB})$ ,  $K_B(K_{AB})$  to Alice
  - Alice decrypts  $K_A(K_{AB})$  and sends  $K_B(K_{AB})$  to Bob
  - Bob decrypts  $K_B(K_{AB})$ , both now know  $K_{AB}$
- Public-Key (asymmetric) Protocols
  - Alice gets Bob's public-key from KDC
  - Alice generates random session-key K and sends  $K_{pub\_B}(K)$  to Bob
  - Bob decrypts  $K_{priv\_B}(K)$ , both now know K

**Both protocols are vulnerable to "man in the middle" attacks**

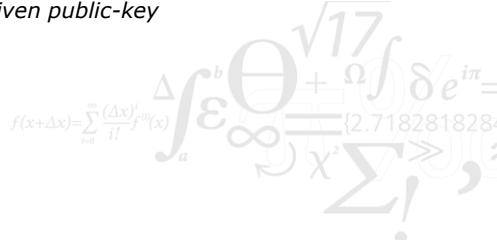
## Public-Key Cryptography Protocol

- The following steps are required to setup communication with PKC
  - 1) Alice and Bob agree on cryptosystem
  - 2) Bob sends Alice his public key
  - 3) Alice encrypts plaintext with Bobs public key
  - 4) Alice sends ciphertext to Bob
  - 5) Bob decrypts ciphertext using his secret key

*How can these steps be compromised?*

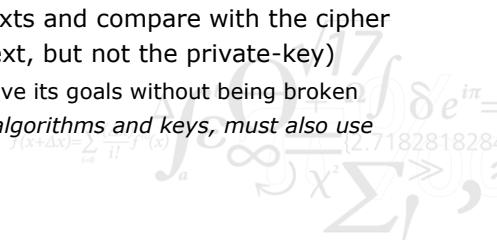
## Public-Key Distribution

- Public-key distribution must address:
  - Authenticity (Certification) of public-keys
    - *Linking public-keys to named identities*
  - Distribution of public-keys
    - *Obtaining somebody else's public-key*
    - *Distribute own public-key*
  - Revocation of public-keys
    - *Revoking published keys*
    - *Determining validity of a given public-key*



## Hybrid Cryptosystems

- Asymmetric cryptography is 3 orders of a magnitude slower than symmetric
- Asymmetric cryptography is vulnerable to chosen plaintext attacks
- If  $C = E(P)$ , where  $P$  is plaintext with entropy  $n$ , a cryptanalyst can encrypt all possible plaintexts and compare with the cipher text (thus he learns the plaintext, but not the private-key)
  - The cryptosystem fails to achieve its goals without being broken
    - *Not enough to use strong algorithms and keys, must also use them in a meaningful way*



## Hybrid Cryptosystems II

- Asymmetric cryptography can be used to encrypt a randomly generated symmetric key (session key)
- Advantages
  - Randomly chosen symmetric key has entropy close to the key-size
  - The symmetric key is short (compared to the encryption key)
    - *Encrypting symmetric key is shorter than encrypting message*
  - The encrypted symmetric key reveals little about the asymmetric key

## Hybrid Cryptosystems III

- The following steps are required to setup communication with hybrid cryptography
  - 1) Bob sends Alice his public key
  - 2) Alice generates a random session key,  $K$
  - 3) Alice encrypts  $K$  with Bobs public key
  - 4) Alice sends the encrypted key to Bob
  - 5) Bob decrypts  $K$  with his secret key
  - 6) Alice and Bob exchanges messages encrypted with the session key

*These steps still require authenticity of public-key and authentication of end points*

## Coffee Break



23 DTU Compute Technical University of Denmark

02239 Data Security

## Public Key Distribution

- Public-key cryptography simplifies key distribution
  - secrecy of the encryption key is not required
- Authentication of Bob's public-key is required
  - In-Band (online)
    - *Public Key Infrastructures*
      - Key Distribution Centers (KDC)/Certificate Authorities (CA)
  - Out-of-Band
    - *Build into product*
    - *Published in Sunday Newspaper every week*

24 DTU Compute Technical University of Denmark

02239 Data Security

## Certification Authorities

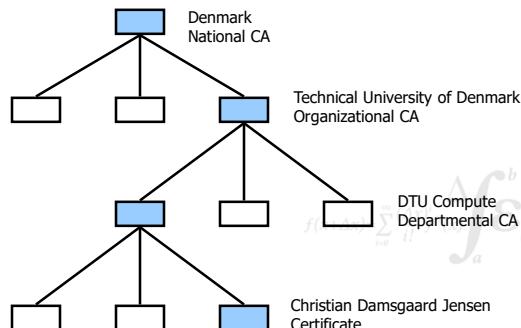
- A certification authority (CA) guarantees that the key belongs to the named principal
- A principal can be:
  - A user
  - An attribute of the user (e.g., her role in within an organisation)
  - An organisation (e.g., a company or another CA)
  - A pseudonym
  - A piece of hardware/software
- Some CAs only allow certain types of principals

## Obtaining a Certificate from a CA

- Alice wishes to obtain a certificate from Charlie the CA
- 1. Alice generates a public-/private-key pair and signs the public-key and identification information with the private-key
  - *Proves that Alice knows the private-key*
  - *Protects public-key and identification information in transit to CA*
- 2. CA verifies Alice's signature on the public-key and her ID
  - *Verification may be done out-of-band*
    - Email/phone callback
    - Business/Credit bureau records, in-house records
- 3. CA signs Alice's public-key and ID with the CA private-key
  - *Creating a certificate that binds Alice's public-key and her ID*
- 4. Alice verifies the public-key, ID and CA's signature
  - *Proves that CA didn't substitute Alice's public-key*
  - *Protection of the certificate in transit from CA*
- 5. Alice and/or CA publishes the certificate

## PKI

- Certificate Authorities organized in a hierarchy
  - Only top-level CA's certificate needs to be known by everyone
  - Intermediate CAs have certificates signed by "superiors"
  - Certificate verification is done by verifying certificates from the principal towards the top-level CA (aka. "Root CA")

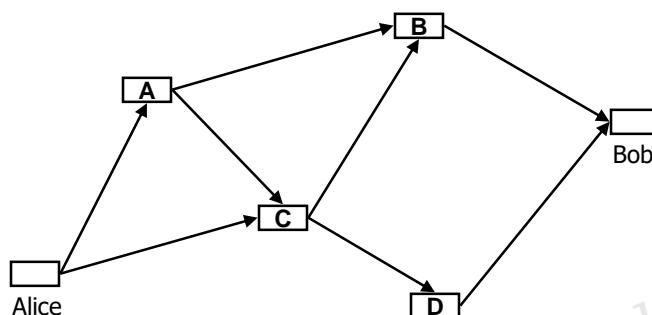


27

DTU Compute Technical University of Denmark

02239 Data Security

## Alternatives Trust Hierarchies



- Bob knows B and D who know A and C who know Alice  $\Rightarrow$  Bob knows the key came from Alice
- Web of trust closer to human notions of trust
  - This is the method used in PGP (and GPG)

28

DTU Compute Technical University of Denmark

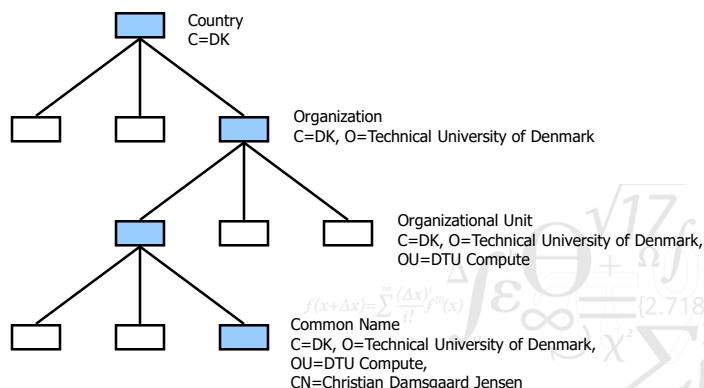
02239 Data Security

## What's in a Name

- Alice uses a PKI to find Bob's certificate
  - Which PKI?
  - Which Bob?
- Names are context dependant
  - Bob is personally known by everyone in the village
    - *People have many names: Robert Johnson, Big Bob the sheriff, ...*
  - There are many people called Bob (or Robert) in the town
    - *People may not know that Robert Johnson = Big Bob*
  - The ASCII string "Bob" loses meaning in the big city
- Qualifiers can be added to names
  - Passport number
  - Central Person Register (CPR) number
- Are we looking for Bob123 or Bob421?

## X.500 Naming

- X.500 defines Distinguished Names (DN) that uniquely names everything on earth



## X.500 Distinguished Names

- Typical DN components
  - Country (C)
  - State or Province (SP)
  - Locality (L)
  - Organisation (O)
  - Organisational Unit (OU)
  - Common Name (CN)
- Problems with X.500 Distinguished Names
  - No rules for how the naming hierarchy should be organised
  - Hierarchical naming only works in clearly hierarchical contexts
    - *Governments, national telecoms providers, ...*
    - *Cannot accommodate nomadic people, stateless people, people with dual citizenship, ...*

## What's in a Certificate

- A typical certificate contains
  - Public-key (e.g., 2048bit RSA key)
  - Identification (e.g., X.500 DN)
  - Validity
    - *Not valid before, not valid after*
  - Issuer identification
    - *Used to establish certification path back to root CA*
  - Issuer signature
- Extensions may qualify the certificate
  - Certificate can only be used for certain purposes
    - *Especially important for CA certificates*
    - Establish the domain of authority for the CA

## Authority of a CA

- What is the root of authority for a CA?
  - TLS certificate binds a public-key to a business' web-server
    - CA has no authority to register businesses
    - CA has no authority to register domain names
- How do we get the CA root certificate?
  - Built into most browsers
  - Firefox comes with around 100 root-certificates pre-installed, incl.:
    - Deutsche Telekom AG
    - Digicert
    - Entrust, Inc.
    - TDC OCES CA
    - TÜRKTRUST Bilgi İletişim ve Bilişim
    - Verisign, Inc.



33 DTU Compute Technical University of Denmark

02239 Data Security

## Trust in PKI

- Root CA can compromise the security of everyone
  - Root CA must be infallible
  - No single authority in the world is trusted by everyone
- Trust in root CA is diluted by the certification path
  - Problem exposed by Uli Maurer's model
  - 90% in CA gives 60% trust in certificate with 5 levels of CAs
- Adding attributes magnifies this problem
  - Credential containing permissions to access a particular resource lends authority from a root CA who knows neither principal nor resource
  - Further down the credential chain permissions become more specific, but the authority of the root CA more diluted



34 DTU Compute Technical University of Denmark

02239 Data Security

## Certificate Revocation

- Certificates must be revoked whenever (*not if*) a private-key is compromised
- Revocation is measured by:
  - Speed of revocation: maximum delay from the compromise is discovered and the last use of the certificate?
  - Reliability of revocation: is it acceptable that someone sometimes may not have learned about the revocation?
  - Number of revocations: how many revocations can the system handle at the time?
- Revocation is either
  - Automatic, e.g., using short expiration times
  - Manual, e.g., using Certificate Revocation Lists (CRL)

## Applications of Asymmetric Cryptography

- Asymmetric cryptography has many interesting applications
  - Blind signatures
  - Zero-Knowledge (ZK) protocols
  - Electronic voting systems
- We'll examine some of these in the following

## Digital Signatures with RSA

- Some public-key crypto-systems allow either key to be used for encryption where the other key is used for decryption (e.g. RSA)
- The basic protocol is:
  - 1) Alice encrypts the message with her private key, this effectively signs the message  
 $\text{sign}(m) = m^d \pmod{n} = m' \text{ (all mod } n\text{)}$
  - 2) Alice sends the encrypted message to Bob
  - 3) Bob decrypts the message with Alice's public key, this verifies Alice's signature  
 $\text{verify}(m') = m'^e = (m^d)^e = m^{ed} = m \text{ (all mod } n\text{)}$
- Asymmetric encryption is slow
  - Normally sign a hash of the message instead of the message itself

## Completely Blind Signatures

- Notaries don't always need to know what they sign
- Alice has a document ( $M$ ) that she wishes Bob to sign using RSA
  - Bob should not learn the content of  $M$
- 1) Alice picks a random value  $X$  and multiplies the document by  $X^e$   
( $e$  is Bob's public-key and  $X$  is called a blinding factor)  
$$M' = M \times X^e$$
- 2) Alice sends the blinded document to Bob
- 3) Bob signs the blinded document  
$$M'' = M'^d = (M \times X^e)^d = M^d \times X^{ed} = M^d \times X$$
- 4) Alice divides out the blinding factor ( $X$ ), leaving the original document signed by Bob ( $M^d$ )

This requires signature and multiplication to be commutative

## Completely Blind Signatures II

- If Alice chooses a true random value, Bob cannot learn anything about the document
- There is no correlation between the two signed documents in step 3 and 4
- Even if Bob records all the blinded documents that he signs, he will be unable to correlate the two
- The signature is valid and can be verified in the normal way

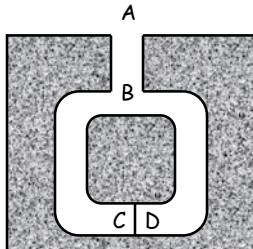
## Blind Signatures

- Completely blind signatures are of limited use
- Blind signatures reassures the signer that he does not sign something bad, without revealing the precise content
  - Alice wish Bob to sign one of  $n$  "synonymous" documents
  - Alice choose  $n$  different random blinding factors
  - Alice blinds the documents and sends all  $n$  documents to Bob
  - Bob asks Alice for  $n-1$  of the blinding factors
  - Bob verifies  $n-1$  copies and signs the  $n^{\text{th}}$  copy
- In order to cheat Alice must guess which blinded copy will be signed
- This "cut and choose" principle is also used to generate "zero-knowledge" proofs
  - A proves to B that she knows secret  $s$  without revealing  $s$

## Zero Knowledge Proofs

### the cave allegory

- How to prove that you know a secret without telling



How can Alice prove that she knows the secret password?

Alice goes into the cave, Bob moves to point B,  
Bob calls what passage Alice should exit from,  
Alice goes through the door (if necessary)  
Alice exits from the correct passage

## Zero Knowledge Proofs

### basic protocol

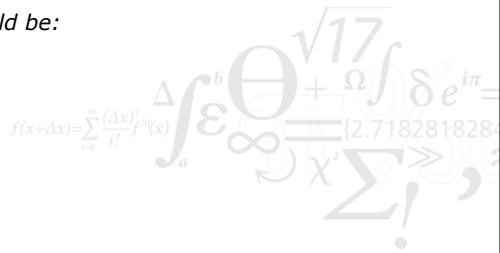
- Assume Alice knows the solution to a hard problem
  - 1) Alice uses a random number to transform her problem into another hard problem, isomorphic to the first problem, she is able to solve the new problem using the solution to the original problem
  - 2) Alice commits to the new problem
  - 3) Alice reveals the new problem to Bob
  - 4) Bob asks Alice to :
    - a) prove that the two problems are isomorphic
    - b) reveal the solution (from 2)
  - 5) Alice complies to the request
  - 6) Alice and Bob repeats the protocol until Bob is satisfied Alice knows the secret
- This is essentially the “cut and choose” principle often used to divide a cake between two siblings

## Secure Elections

- Requirements for elections on the Internet
  - 1) Only authorized voters can vote
  - 2) No one can vote more than once
  - 3) No one can determine for whom someone else voted
  - 4) No one can duplicate anyone else's vote
  - 5) No one can change anyone else's vote
  - 6) Every voter can make sure that his vote is counted

*an additional requirement could be:*

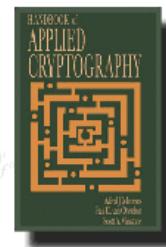
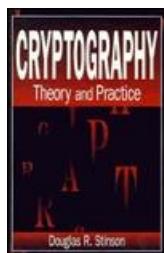
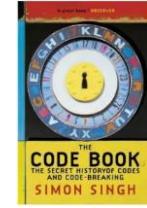
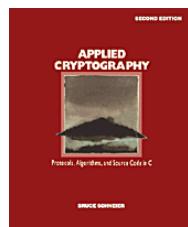
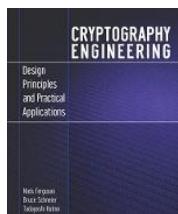
- 7) Everyone knows who voted

$$f(x+\Delta x) = \sum_{l=0}^{\infty} \frac{(\Delta x)^l}{l!} f^{(l)}(x)$$
A faint watermark in the background of the slide contains various mathematical symbols and equations, including  $\sqrt{17}$ ,  $f$ ,  $\Theta$ ,  $\Omega$ ,  $\delta e^{i\pi}$ ,  $\Sigma$ ,  $\infty$ ,  $x^2$ , and exclamation marks.

## Voting with Blind Signatures

- 1) The voter creates 10 sets of votes, each set contains one vote for every possible outcome (e.g., two votes for a referendum) and a large random number
- 2) The voter blinds all sets of votes to the polling station
- 3) The polling station verifies that the voter hasn't voted before, requests the blinding factor for 9 of the 10 sets, opens and verifies the 9 sets and signs the 10<sup>th</sup> set of votes, returning it to the voter
- 4) The voter unblinds the votes and selects the appropriate vote
- 5) The voter encrypts his unblinded vote with the public key of the voting station
- 6) The voter sends the encrypted vote to the polling station
- 7) The polling station decrypts the vote, checks both the signature and that the serial number has not been received before (i.e., that only one vote from the set has been received) and registers the vote, the serial is made public so the voter can know that his vote is counted

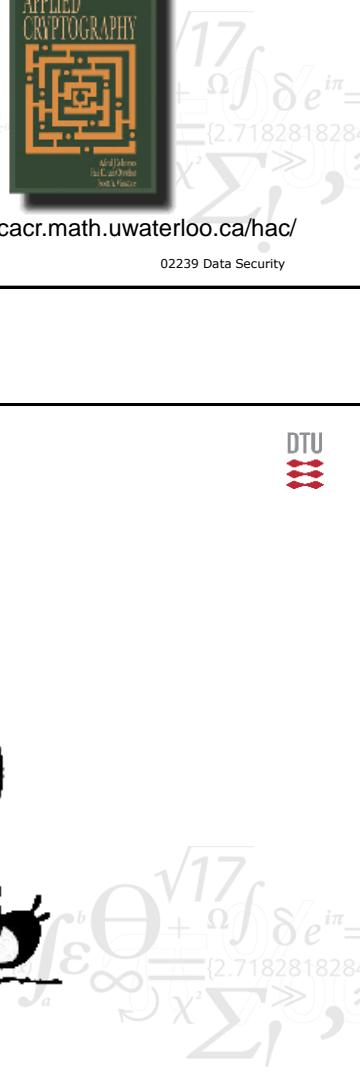
## To learn more



<http://www.cacr.math.uwaterloo.ca/hac/>

45 DTU Compute Technical University of Denmark

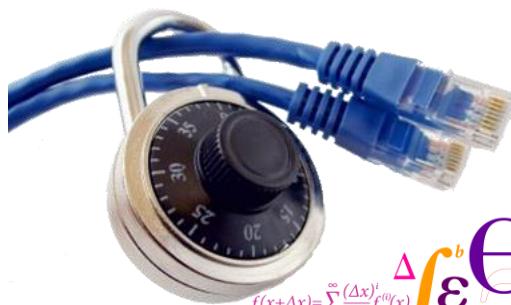
02239 Data Security



46 DTU Compute Technical University of Denmark

02239 Data Security

## Security in Networks

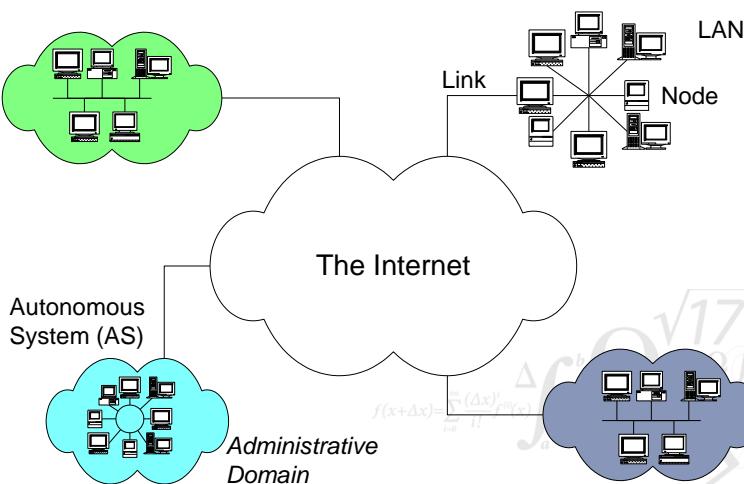


$$\begin{aligned} f(x+\Delta x) &= \sum_{n=0}^{\infty} \frac{(\Delta x)^n}{n!} f^{(n)}(x) \\ \int_a^b \mathcal{E}^{\Theta} \Omega^{\sqrt{17}} \delta e^{i\pi} &= \{2.7182818284 \\ \infty \cup x^2 \gg , \sum ! \end{aligned}$$

DTU Compute

Department of Applied Mathematics and Computer Science

## Network Definitions



## Network Characteristics

- Networks are defined by:
  - Boundary  
Distinguishes nodes inside the network from nodes outside the network. It is theoretically possible to make a list of all components belonging to the network
  - Ownership (Autonomous System/Administrative Domain)  
Nodes belonging to the same owner and managed by the same administrators. Ownership is sometimes difficult to determine, e.g., who owns the Internet?
  - Control (physical security)  
Not all nodes belonging to the network are under control of the administrator, e.g., your own laptop connected to DTU's wireless network, Bring Your Own Device (BYOD) in many companies.

## Types of Networks

- 2 types of networks:
  - Shared medium (Bus, Token Ring, Mesh, ...)
  - Point-to-point networks
- Classification based on diameter:

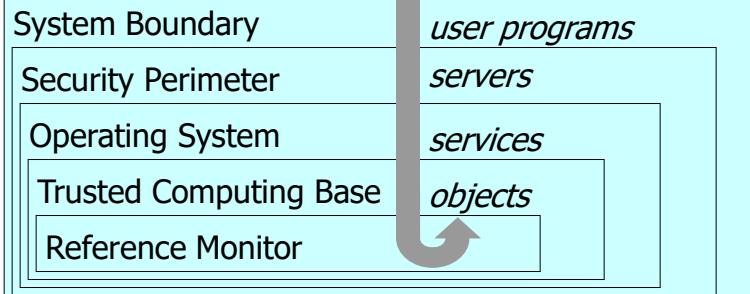
1 m	System
10 m	Room
100 m	Building
1 km	Campus
10 km	City
100 km	Country
1,000 km	Continent
10,000 km	Planet

Multi-processor  
Device connection, Body Area Networks  
LAN  
Metropolitan Area Network (MAN)  
WAN  
The Internet

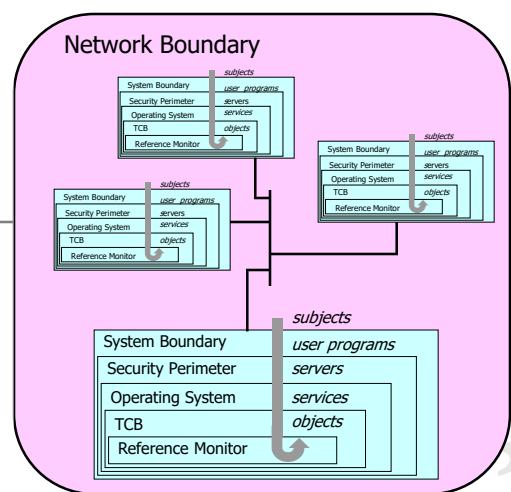
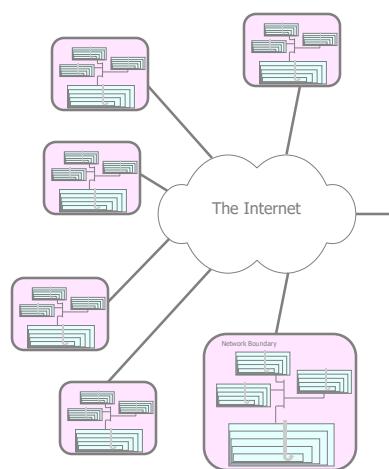
## Stand Alone Systems

*users, input devices, output devices,...*

*subjects*



## Networked Systems

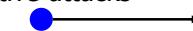


## Network Vulnerabilities

- Programming mistakes
  - Public services must handle active and passive data from untrusted sources
    - *Buffer overflows, malicious applets, ...*
- Protocol Flaws
  - Internet was designed for a few hundred trusted computers
    - *This is evident in the lack of security in many protocols (SMTP)*
  - Protocols are often complex and specifications are ambiguous
- Misconfiguration of software and systems
  - Complexity of systems and focus on functional requirements (short deadlines) leads to misconfigured systems
- Unpatched software
  - Vendors normally issue patches when vulnerabilities are exposed, responsibility of customers to apply patches

## Network Attacks

### Active attacks



*Normal communication*



*Deletion*

### Modification



*Modification*

### Passive attacks



*Eaves dropping*

### Fabrication



*Fabrication*

### Traffic analysis

$f(x+\Delta x)=\sum_{i=0}^n \frac{(\Delta x)^i}{i!} f^{(i)}(x)$

## Prevent – Detect – React (– Recover)

- Network attack prevention
  - Firewalls - *perimeter security*
  - Identity and Access Management (IAM) – *network access authorisation*
  - Segmentation - *compartmentalisation + security in depth*
  - Virtual Private Networks (VPN) – *secure communication*
- Network attack detection
  - Intrusion Detection Systems
  - Honey Pots
  - Security Information and Event Management (SIEM)
- Network attack reaction
  - Systems-/Network Operation Centres (SOC/NOC)
  - Contingency-/Disaster plans
    - *Procedures described in playbooks or runbooks*

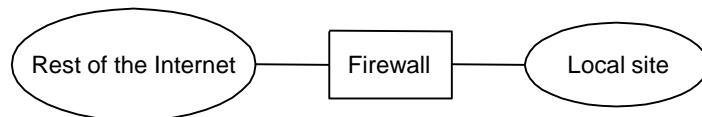
## Perimeter Security

- Defending the Network Boundary
  - Keep unwanted outsiders away from valuable resources on the network
  - Don't want outsiders coming in (except for special web-services )
    - *Implies filtering of traffic*
  - Allow insiders to go out
    - *Network Address Translation (NAT) may suffice*
- Filtering traffic at the network boundary (corporate firewalls)
  - Network wide filtering
  - Configured/managed by system administrators
  - Transparent to individual nodes/users
- Filtering traffic at each individual node (personal firewalls)
  - Host-based policy (locally or globally defined)
  - May allow local policy override
  - May require intervention by local users

## Firewalls

- A firewall filters information that is sent and received from outside the network
  - It is software that resides on the network's gateway
- A firewall can filter
  - data packets, examining the source & destination IP
  - ports and applications, in order to deny access
    - *May include time of day in decision*
  - Content (payload), such as macros, inappropriate web content, ...

## Filtering Firewalls

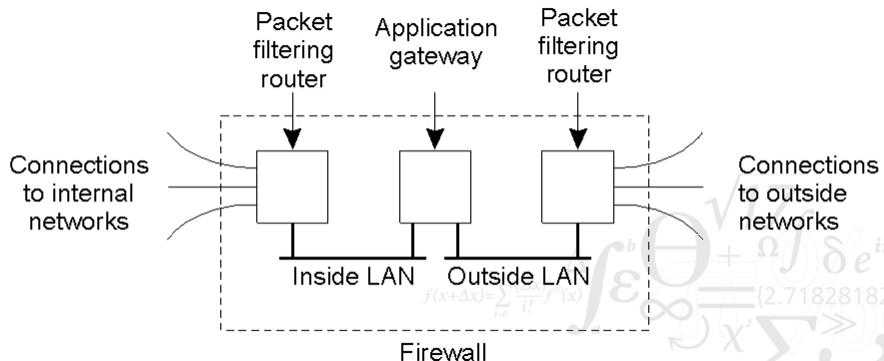


- Packet-Filter-Based Solution

- example
    - ( 192.12.13.14, 1234, 128.7.6.5, 80 )
    - ( \*, \*, 128.7.6.5, 80 )
  - default: forward or not forward?
  - how dynamic?

## Firewalls

- A common implementation of a firewall



## Impersonation Attacks falsely obtain valid authentication details

- Guess identity and authentication details (login, password)
  - Try popular logins (guest, admin) and passwords (secret, password)
- Obtain identity and authentication details by eavesdropping
  - `telnet` sends login/password over the wire in the clear
  - Bad encryption is sometimes used (MS LAN manager, WEP)
- Circumvent/disable authentication
  - Exploit buffer overflows
  - Social engineering ("I forgot my password, please reset to BANANA")
- Use a target that will not be authenticated
  - Remote authentication from "trusted hosts" is sometimes disabled
  - Public accounts (anonymous FTP, GUEST accounts)
- Use a target whose authentication details are well known
  - Standard accounts with default passwords  
*You can sometimes find these in the manuals*



## Spoofing falsify authentication details

- Masquerading
  - Falsify DNS entries, choose similar hostnames, ...
    - *www.whitehouse.com used to be a porn site*
    - *www.whitehouse.net is a spoof site against George W. Bush*
    - *www.whitehouse.gov is the official site*
- Session Hijacking
  - Intercepting a session initiated by another entity
    - *Redirect network communication to the attacker's host*
    - *Insert redirection on the victim's web-site*
- Man-in-the-Middle Attack
  - Alice wishes to talk to Bob
    - *Charlie pretends to be Alice to Bob and to be Bob to Alice*
    - *Charlie intercepts all messages between Alice and Bob and forwards, possibly modified versions, to the other entity*

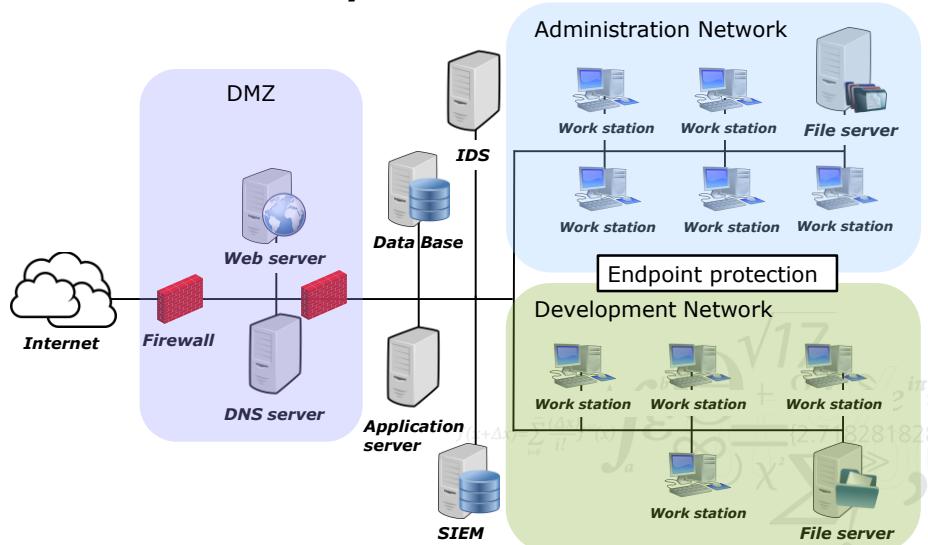
## Availability Threats Denial of Service (DoS)

- Transmission failure
  - hardware/software failure of nodes or links
- Connection flooding
  - Exhaust bandwidth (sending too much traffic to the victim)
    - *This is what the Low Orbit Ion Cannon (LOIC) does*
  - Exhausting resources at the victim's site (SYN floods)
  - DNS attacks
    - *Redirecting traffic to the victim's site*
    - *Redirecting traffic away from the victim's site*
- Distributed Denial of Service (DDoS)
  - Break into many hosts (zombies) on the network (create a Botnet)
  - Instruct zombies to overload victim at a given time
  - Individual zombies may appear to be valid clients

**Pause**

17 DTU Compute Technical University of Denmark

02239 – Data Security

**Network Security Architecture**

18 DTU Compute Technical University of Denmark

02239 – Data Security

## Communication Security

### link encryption

- Encryption performed in “the Data Link Layer”
- Each link on the route is encrypted independently
- Advantages:
  - Transparent to hosts (OS and applications)
  - May protect packet meta data (headers)
  - Fast encryption hardware can be used
- Disadvantages:
  - Data decrypted/re-encrypted on all intermediate nodes
    - *Modification of standard protocols*
    - *Introduces a performance penalty*
    - *Data is exposed on all intermediate nodes (routers)*
  - Everything or nothing gets encrypted

## Communication Security

### end-to-end encryption

- Encryption performed in the “application layer”
- Encrypted at source and only decrypted at destination
- Advantages:
  - Data protected all the way from sender to receiver
  - Flexible choice of encryption algorithms
    - *Balance security and performance*
  - Some data can be sent unencrypted
  - No modification of the existing routing infrastructure
- Disadvantages:
  - User (programmer) needs to manage encryption explicitly
  - Applications and services may need to be modified
    - *Expensive and difficult to update encryption technology*

## Virtual Private Networks (VPNs)

- VPNs use a public network (usually the Internet) to provide a secure connection between two private parts of a network or remote users on a network
- They can be used to connect a PC into a LAN (i.e., a telecommuter) or two connect two LANs (i.e., a branch office to a corporate headquarters)
- They are inexpensive to use
  - they eliminate the need for a costly private leased circuit
  - and they provide privacy over a public network

$$f(x+\Delta x) = \sum_{l=0}^{\infty} \frac{(\Delta x)^l}{l!} f^{(l)}(x)$$
$$\int_a^b \Theta^{\sqrt{17}} e^{\Omega f \delta e^{i\pi}} dx = [2.718281828]$$

## Tunneling

- Tunneling is the technique used by the ends of a VPN to communicate
  - a packet send over a VPN is encapsulated in a secure protocol prior to being embedded into the IP protocol
- The receiving end (at the router) strips off the header and trailer information of the packet and the private network protocols decapsulates the packet and send the packet on
- Several private network protocols are available to encrypt the packet on a VPN



## Tunneling

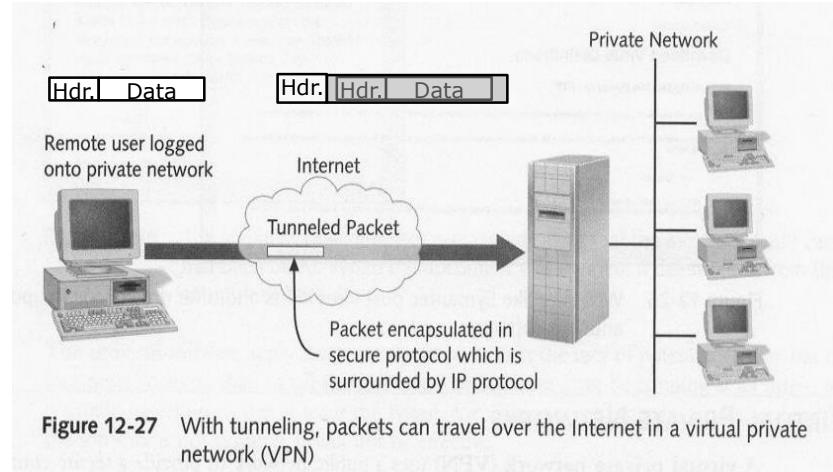


Figure 12-27 With tunneling, packets can travel over the Internet in a virtual private network (VPN)

## Intrusion Detection definition

- *Intrusion detection is the process of identifying and responding to malicious activity targeted at computing and networking resources*
  - Process
    - *Interaction between people and tools, it takes time*
  - Identifying
    - *Before, during or after the intrusion*
  - Responding
    - *Collect evidence, limit damage (honey pots), shut-out*
  - Malicious activity
    - *Intentional attempts to do harm*
  - Computing and networking resources
    - *Logical intrusions as opposed to physical intrusions*

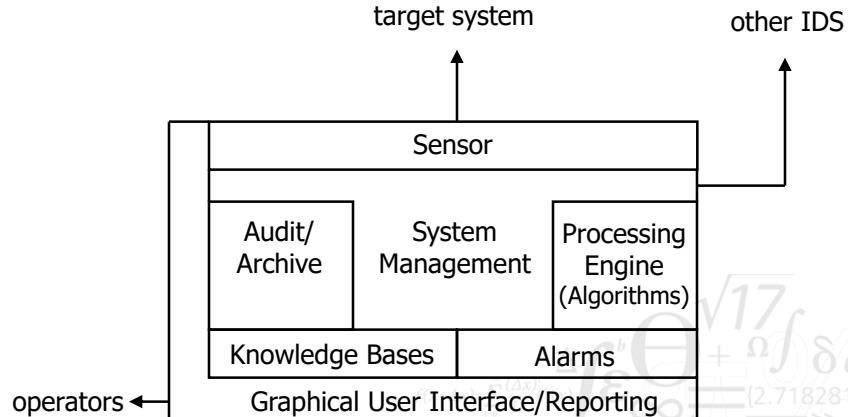
## Two Types of IDS Systems

- Host Based Systems
  - Inspects locally available information
    - Application log-files
    - System log-files
    - Monitoring running activities
- Network Based Systems
  - Inspects traffic on the network
  - Signature Based Systems
    - Relies on established patterns ("signatures") in malicious network traffic
    - Similar to the signatures used in virus checkers
  - Anomaly Detection Systems
    - Establishes base line for normal communication
    - Detects abnormal traffic pattern
  - Employs techniques from machine learning, pattern matching, big data

## IDS Infrastructure

- The IDS must be capable of collect and process enormous amounts of data
- Collect data from various sources
  - Network sniffers, login programs, logfiles, ...
- Capacity to examine all collected data
  - Look for intrusion patterns
  - Ignored data may contain evidence of an intrusion

## IDS Components



## Sensors

- Audit trail processing
  - Collect and check all logfiles (OS, servers, ...)
  - Most of the information is already collected
    - *Consolidate in Security Information and Event Management (SIEM)*
- On-the-fly processing
  - Look for “dirty words”, e.g., /etc/passwd
  - Complements audit trail processing
- Profiles of normal behaviour
  - Uses a priori information about system usage
  - Fine tuning of initial profile based on usage

## Processing Engine

### how to detect an intrusion?

- The processing engine is responsible for detecting intrusions
- Heuristics based intrusion detection
  - Repetition of a suspicious action
  - Mistyped command from an automated sequence
  - Known signatures (exploits)
  - Directional inconsistencies (inbound and outbound traffic)
  - Unexpected attributes of some service request or packet
  - Unexplained problems in a subsystem
- Anomaly detection
  - Statistics based
    - *Markov process, outlier detection,*  $f(x) = \sum_{i=0}^n \frac{(\Delta x)^i}{i!} f^{(i)}(x)$
  - AI based
    - *Machine learning, Artificial Neural Networks, Deep Learning, ...*

## Trapping Intruders

- The IDS may include a copy of the real system
  - Redirect intruders to this trap system
  - Must present a consistent view of the system in order to prevent detection
- The IDS may construct a honey pot
  - A trap system that looks attractive to intruder
    - *Based on his search for information*
    - *Example in Clifford Stoll: The Cuckoo's Egg*



## Honeypots/Honeynets

- Machines that appear interesting to attackers
  - Offers no real data or services
  - Only “users” are attackers
    - Configured with sensors to detect and record usage
      - If used raise alarm
    - Provides clues to objectives and motives of attackers
- Purpose is to attract and retain attackers
  - Attract them away from production environment
  - Retain them while communication is analysed and tracked
  - Possibility of misinformation or misdirection
- Honeypots can be extended to networks
  - Check: <https://www.honeynet.org>



02239 – Data Security

31 DTU Compute Technical University of Denmark



## Ethics of Intrusion Detection

- Intrusion detection requires monitoring of all network communication
  - Equivalent to Orwell’s Big Brother!
- Data collected for intrusion detection may be used (or abused) in another context
  - Establish activity of employees in a company



02239 – Data Security

32 DTU Compute Technical University of Denmark

## Authentication



"On the Internet, nobody knows you're a dog."

DTU Compute

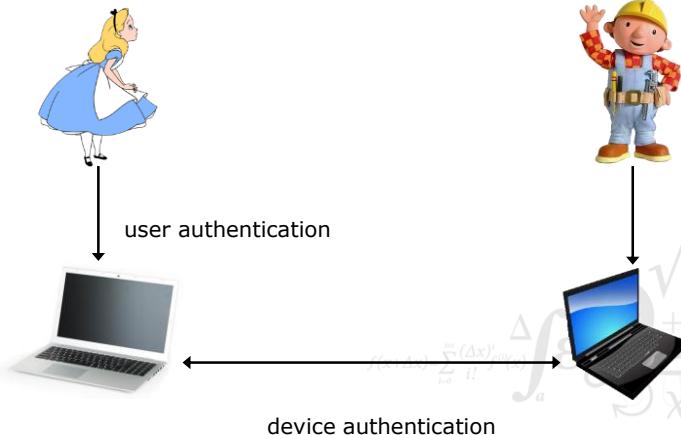
Department of Applied Mathematics and Computer Science

$$\int_a^b \varepsilon^{\sqrt{17}} \theta + \Omega \int \delta e^{i\pi} = \{2.7182818284 \dots\}$$

## The purpose of authentication

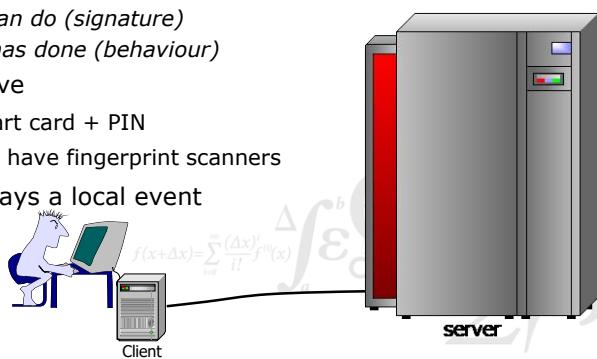
- Authentication is the process of verifying a claimed identity
- Allowing people to access the system
  - Ability to authenticate demonstrates authorization to access system
- Binding an identity to system entities
  - Authorizations are linked to identities (or roles that are assigned to ID)
- Associating a real world identity (transitively) with system events
  - Accountability facilitated through recording ID of requesting entities

## Different Types of Authentication

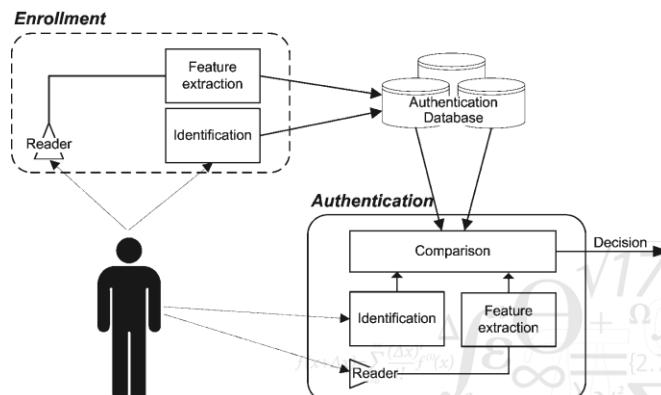


## User Authentication

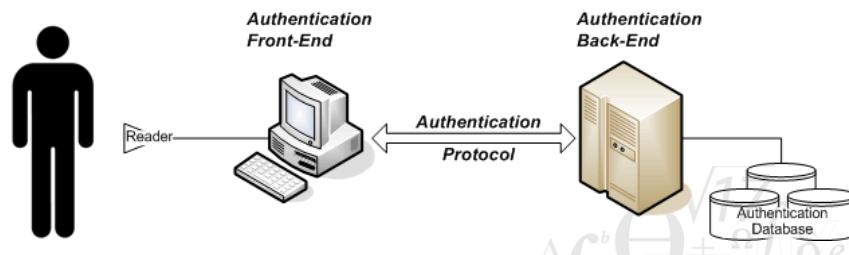
- Establish the identity of principals by means of:
  - something he knows (*shared secrets*, e.g., *password, PIN*)
  - something he possesses (*smart card, USB token, mobile phone*)
  - something he is (*fingerprint, face, voice, retina scan*)
    - *something he can do (signature)*
    - *Something he has done (behaviour)*
- Combinations of above
  - VISA cards has smart card + PIN
  - Many smart phones have fingerprint scanners
- Authentication is always a local event



## Authentication Mechanism



## General Model of Authentication



Where can this go wrong?

## Password Authentication

- Most operating systems rely exclusively on passwords
  - login: username*
  - password: \*\*\*\*\**
- Password is checked by “login” program
  - Prompts for username
  - Prompts for password
  - Performs one-way function on password (hash)
  - Compares hashed password with password stored in password file
- Example – Classic Unix password file

```
Name : Password : UserID : GroupID : Gecos : HomeDirectory : Shell
- Example
bill:5fg63fhD3d5gh:157:5:Bill Smith:/home/bill:/bin/sh
- Password is used as key to encrypt a null string
  •  $E(pw+salt)+salt$  – password = 64 bit, salt = 12 bit
```

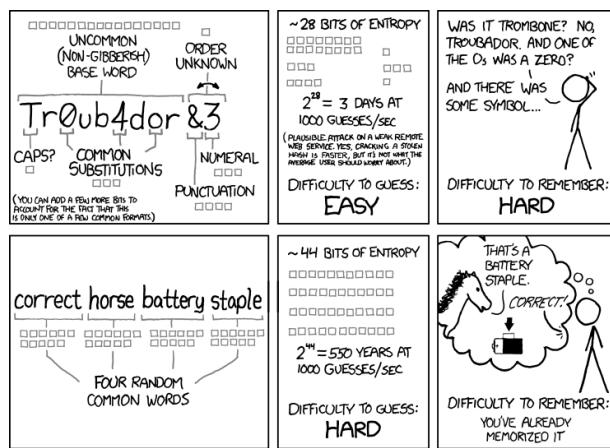
## Passwords

- Single password is the most common method for authentication
  - Simple and “generally accepted”
    - *Everyone knows how they work*
  - Cheap to implement
    - *No additional hardware required*
- Password Security
  - Anyone who knows the password will be authenticated
  - Passwords must be difficult to guess
    - *Resistance to brute force attacks*
      - Passwords must be long (more than 12 characters)
      - Passwords must be complex (difficult to remember)
    - *Resistance to guessing-/dictionary attacks*
    - *Passwords should be unique (no reuse across websites)*
- Remembering many long complex passwords is hard for users

## Passphrases

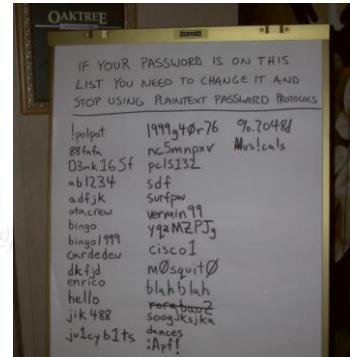
- Passphrases include several words
  - Technology is similar to password implementation
- Three major challenges
  - Usability vs. Security
    - Random words (similar to passwords but with a larger alphabet)
    - Natural language sentences (obey syntax and grammar)
      - Length improves security
      - Structure reduces entropy
  - Passphrase retention
    - Structure of passphrases makes them easier to remember
    - Common advice to construct and recall complex passwords
  - Passphrase Entry
    - Time consuming (typing more characters)
    - Error prone (getting all the characters exactly right)
- Improve usability by accepting passphrases with a few small errors

## Password Strength



## Attacks on Password Entry

- Passwords can be read over the shoulder (shoulder surfing)
  - Computers, ATM machine design, pay phones, ...
  - Try to hide what you are typing
- Passwords may be compromised every time they are used over a network (packet sniffers)
- Prevented by One Time Passwords (OTP)
- Password can be intercepted in transit between user and system
  - Trojan horse login screen
  - Trusted path between user and system is required
    - *ctrl-alt-del on Windows gives a genuine login screen*

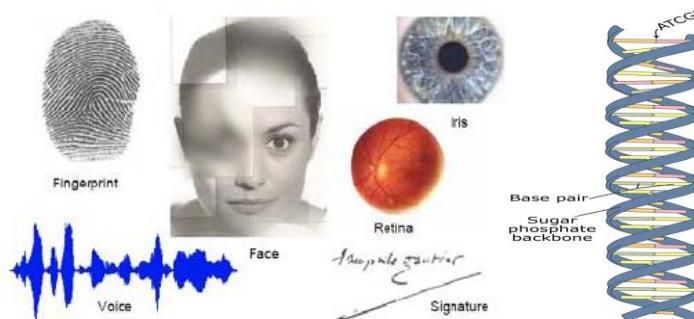


11 DTU Compute Technical University of Denmark

02239 – Data Security

## Biometrics

*Biometrics identify people by measuring some aspect of individual anatomy or physiology (hand geometry or fingerprint), some deeply ingrained skill, or other behavioural characteristic (handwritten signature), or something that is a combination of the two (voice)*



12 DTU Compute Technical University of Denmark

02239 – Data Security

## Biometric Authentication Systems

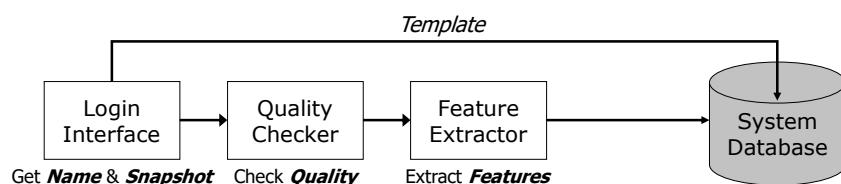
- Biometric systems have three types of operations
  - Enrollment (just like any other authentication system)
  - Verification (biometric authentication)
    - *match 1:1 one captured template to one stored template*
  - Identification
    - *match 1:N one captured template to N (or all) stored templates*



13 DTU Compute Technical University of Denmark

02239 – Data Security

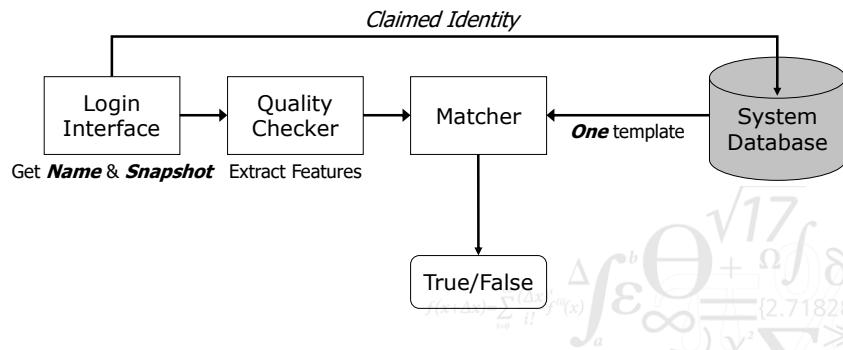
## Enrollment in Biometric Systems



14 DTU Compute Technical University of Denmark

02239 – Data Security

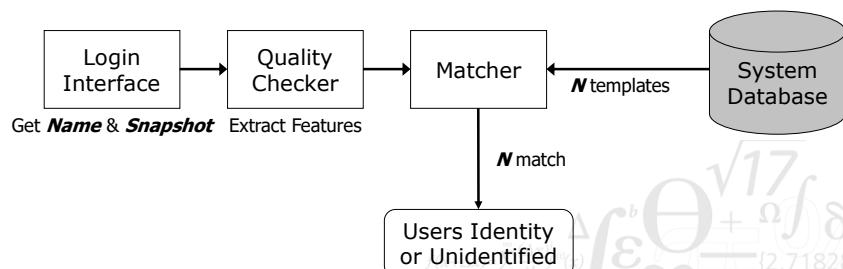
## Verification in Biometric Systems



15 DTU Compute Technical University of Denmark

15 02239 – Data Security

## Identification in Biometric Systems



16 DTU Compute Technical University of Denmark

16 02239 – Data Security

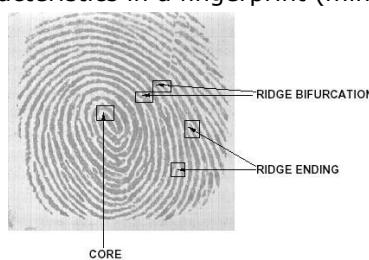
## Handwritten Signatures

- Identifying handwriting is difficult, experts have an error rate of 6.5%, non experts have an error rate at 38%
- Problem with false accepts and rejects
  - false accepts result in fraud
  - false rejects result in insult (bad for business)
  - systems can be tuned to favour one over the other
- Optical systems are unreliable
- Signature tablets record shape, speed and dynamics of signature
  - more reliable

$$f(x+\Delta x) = \sum_{l=0}^{\infty} \frac{(\Delta x)^l}{l!} f^{(l)}(x)$$
$$\int_a^b \Theta + \Omega \int \delta e^{i\pi} = [2.718281828]$$
$$\Sigma \gg x^2 \sum \gg , ! ?$$

## Fingerprints

- Fingerprints have been used as signatures for several centuries
- They are currently used to identify criminals (affects acceptability)
- Measures unique characteristics in a fingerprint (minutiae)
  - Crossover
  - Core
  - Bifurcations
  - Ridge ending
  - Island
  - Delta
  - Pore
- Error rate can be affected by scars, wear, etc.
  - Very old and very young have weak fingerprints
- Many systems defeated by Gummy Fingers
  - Requires liveness detection



## Cloning a Finger

[Matsumoto]

### Making an Artificial Finger from a Residual Fingerprint

#### Materials

A photosensitive  
coated Printed Circuit  
Board (PCB)  
"10K" by Sanhayato Co., Ltd.



Solid gelatin sheet  
"GELATINE LEAF"  
by MARUHA CORP



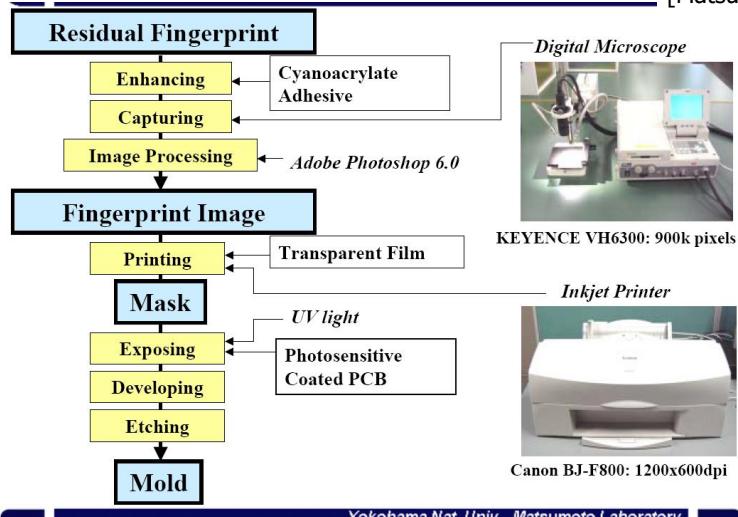
Yokohama Nat. Univ. Matsumoto Laboratory

19 DTU Compute Technical University of Denmark

02239 – Data Security

## Cloning Process

[Matsumoto]

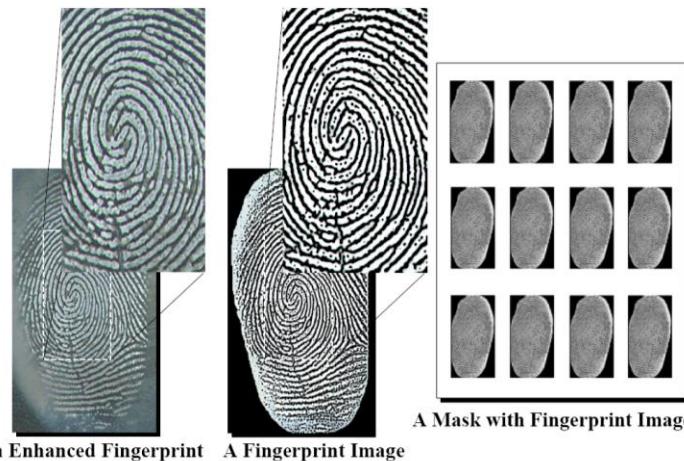


20 DTU Compute Technical University of Denmark

02239 – Data Security

## Fingerprint Image

[Matsumoto]



Yokohama Nat. Univ. Matsumoto Laboratory

21

DTU Compute Technical University of Denmark

02239 – Data Security

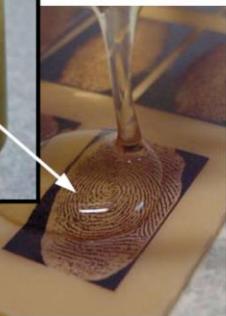
## Molding

[Matsumoto]

Gelatin Liquid



Drip the liquid onto the mold.

Put this mold into  
a refrigerator to cool,  
and then peel carefully.

Yokohama Nat. Univ. Matsumoto Laboratory

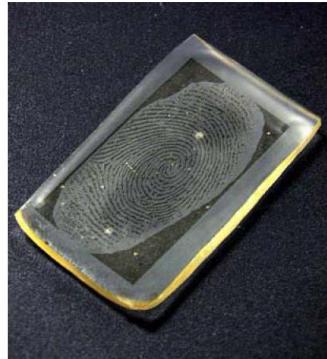
22

DTU Compute Technical University of Denmark

02239 – Data Security

## The Mold and the Gummy Finger

[Matsumoto]



**Mold: 70JPY/piece**  
(Ten molds can be obtained  
in the PCB.)

**Gummy Finger: 50JPY/piece**

23

DTU Compute Technical University of Denmark

Yokohama Nat. Univ. Matsumoto Laboratory

02239 – Data Security

## Side By Side

[Matsumoto]

Pores can be observed.



Enhanced Fingerprint



Captured Fingerprint Image of  
the Gummy Finger  
with the device H (a capacitive sensor)

24

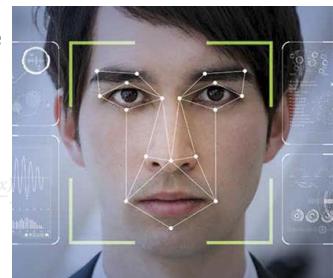
DTU Compute Technical University of Denmark

Yokohama Nat. Univ. Matsumoto Laboratory

02239 – Data Security

## Face Recognition

- Face recognition is the oldest and most widespread form of identification
  - Manually used in photo ID (passport, ...)
  - Increasingly used in smartphones
- Uses off-the-shelf camera to measure the following facial features:
  - Distance between the eyes
  - Distance between the eyes and nose ridge
  - Angle of a cheek
  - Slope of the nose
  - Facial Temperatures (with IR camera)
- Multiple cameras allows 3D models
  - Stereo vision (2 normal cameras)
  - 1 normal camera + 1 IR camera

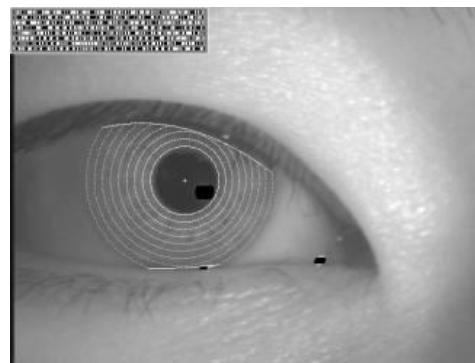


25 DTU Compute Technical University of Denmark

02239 – Data Security

## Iris Scan

- Measures unique characteristics of the iris
  - Ridges (rings)
  - Furrows
  - Straitions (freckles)



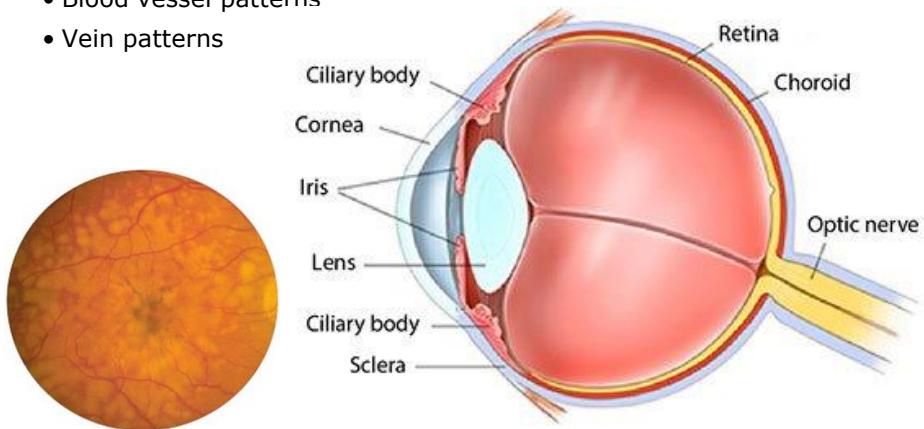
- Simple unattended systems can be defeated by photographs

26 DTU Compute Technical University of Denmark

02239 – Data Security

## Retina Scan

- Measures unique characteristics of the retina (back of the eye ball)
- Blood vessel patterns
- Vein patterns



27 DTU Compute Technical University of Denmark

02239 – Data Security

## Voice recognition

- Voice recognition identifies the speaker
  - Not to be confused with speech recognition (identifies what she says)
- Can be used for authentication over the phone
  - Include context to bind authentication to transaction:  
"Transfer 200 Kr to account 123456789"
- Systems exist with < 1% error rate
- Tape recorders distort the voice enough to prevent "replay attacks"
- Digital recorders may be used to attack voice recognition systems in the future

$$f(x+\Delta x) = \sum_{n=0}^{\infty} \frac{(\Delta x)^n}{n!} f^{(n)}(x)$$

$$\int_a^b \epsilon^{\Theta} + \Omega \int_a^b \delta e^{i\pi} = 2.718281828$$

$$\infty \approx x^2 \sum_{n=0}^{\infty}$$

28 DTU Compute Technical University of Denmark

02239 – Data Security

## Biometric Authentication Summary

- Automated identification systems (scanners) have been developed
- Quality of system depends on accuracy (error rate)
- Error rate is often below 1%
  - What does error rate < 1% mean in practice?
    - Heathrow Airport has ~111.000 passengers arriving every day (2018)
    - Around 4625 passengers arrive every hour
    - Around 77 passengers arrive every minute
    - Around 1 error every second for passengers arriving at Heathrow
  - Reducing error rate by an order of magnitude makes little difference



29 DTU Compute Technical University of Denmark

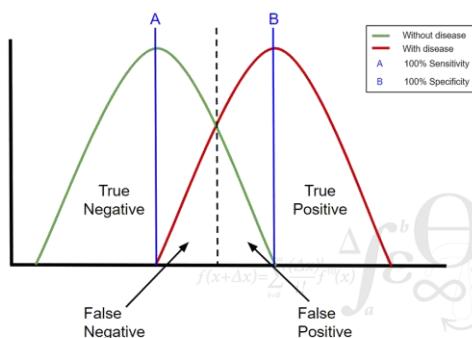


02239 – Data Security

## Threshold Based Authentication Systems

- Comparison of presented features with stored template
  - Rarely an exact match, so verification is based on threshold

Sensitivity vs. Specificity



30 DTU Compute Technical University of Denmark

02239 – Data Security

## Authentication Mechanism Quality Metrics

- Threshold based mechanisms give four possible results

	Is the person claimed	Is <b>not</b> the person claimed
Test is positive (there is a match)	True Positive	False Positive
Test is negative (there is no match)	False Negative	True Negative

$$\text{Sensitivity} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{Specificity} = \frac{\text{True Negative}}{\text{False Positive} + \text{True Negative}}$$

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{False Positive} + \text{False Negative} + \text{True Negative}}$$

$$\text{Prevalence} = \frac{\text{True Positive} + \text{False Negative}}{\text{True Positive} + \text{False Positive} + \text{False Negative} + \text{True Negative}}$$

## Break



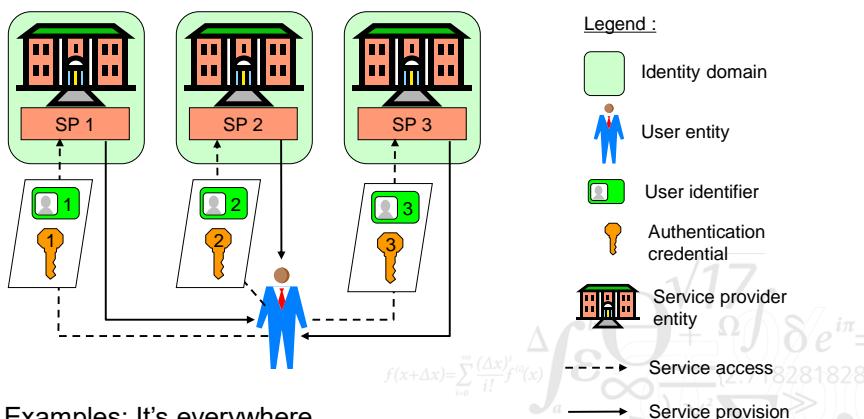
## Identity Management Models

- Authentication in distributed systems require an agreed model f identities and authentication
- Three fundamental models:
  - Identity Silos
  - Single Sign-On systems
  - Federated Identity models

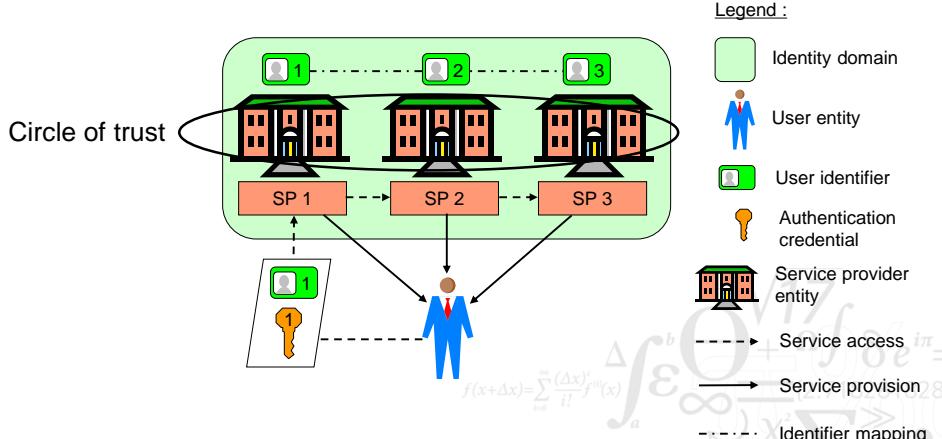


$$f(x+\Delta x) = \sum_{l=0}^{\infty} \frac{(\Delta x)^l}{l!} f^{(l)}(x)$$
$$\int_a^b \Theta + \Omega \int \delta e^{i\pi} = [2.718281828]$$
$$\infty \sum \gg !$$

## Isolated User Identity Model

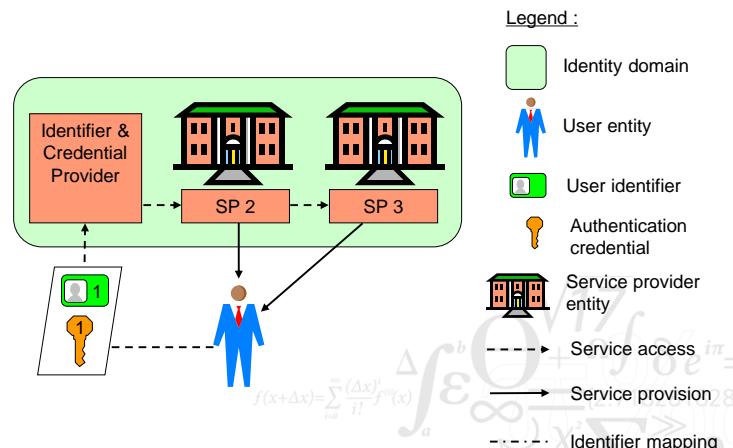


## Federated Identity Model



Examples: SAML2.0, WS-Federation, Shibboleth, Eduroam

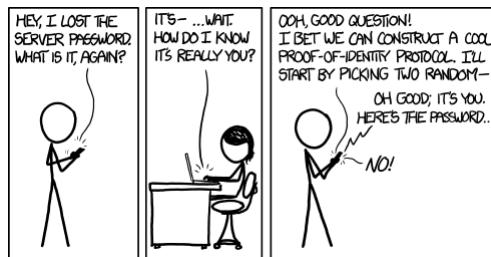
## Single Sign On (SSO) Systems



Examples: Kerberos, Nem-ID

## Authentication Protocols

- Authentication protocols extend authentication across networks
  - Authentication of remote users
  - Authentication of remote devices
  - Authentication of intention to authenticate
    - Protection against *relay attacks*
    - Protection against *replay attacks*



38 DTU Compute Technical University of Denmark

02239 – Data Security

## Kerberos

- Project Athena at MIT (mid to late 1980s)
- Hundreds of diskless workstations
  - Open terminal access, no physical security
  - Insecure network
- Few servers (programs, files, print, ...)
  - Physically secure



39 DTU Compute Technical University of Denmark

02239 – Data Security

## Simple Authentication

- One password per service is infeasible
  - Identity silos
- New authentication service (Charon) introduced
  - Single sign-on service
- Both users and services have passwords
- Charon identifies user by password
- Charon returns a “ticket” to the user
  - Ticket includes identity encrypted with the service’s password
  - If the ticket decrypts properly, access to the service is granted
- How do we know that we got the right ticket?

## Stronger Authentication

- Include service name in the ticket to prove that it was properly encrypted
- Include client workstation address to prevent network sniffers
- Remaining problems:
  - Re-authentication every time a new service is contacted
  - Password sent across the network in the clear

## Ticket Granting Service

- TGS has access to the Charon database
- TGS provides service tickets to users with a TGS ticket (eliminate resubmitting password)
- Users obtain ticket granting tickets from Charon
- User sends username, receives TGS ticket encrypted with user's password
- Tickets can be reused

## Insecure Workstations

- What happens to tickets after a user has logged out?
  - An opponent could log on to the workstation and use the tickets
  - Could be explicitly destroyed when user logs out
  - Sniffer could be used to capture tickets, hacker may then login to the same workstation and use the tickets (replay session)
- This demonstrates common problems in authentication
  - In-memory caches of data and re-use of physical resources
  - Problem with secure revocation of authorisations
  - Ensuring freshness of authentication data

## Limiting Ticket Lifespan

- Charon timestamps ticket when it is issued
- Charon includes lifespan along with timestamp
- Remaining problems:
  - Workstation clocks must be synchronized
  - What should the lifespan of a ticket be?

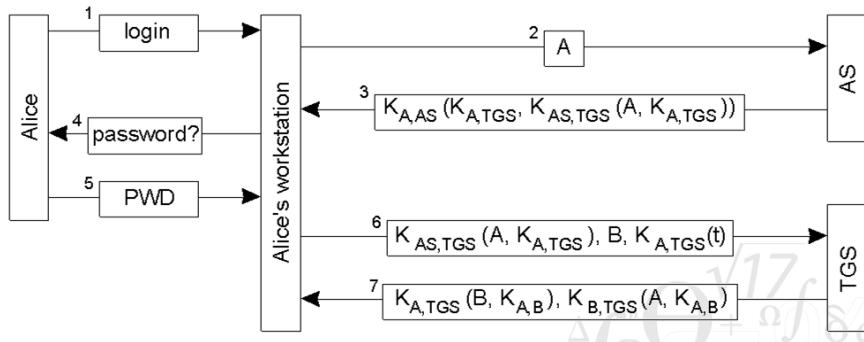
## Verifying Tickets

- Authentication relies on the following tests:
  - Can the service decrypt the ticket?
  - Has the ticket expired?
  - Do the username and workstation address correspond?
- The tests prove:
  - The ticket came from Charon
  - The ticket is still valid
  - Failure proves that the ticket is false, success does not prove anything, tickets can be stolen and reused on the same workstation

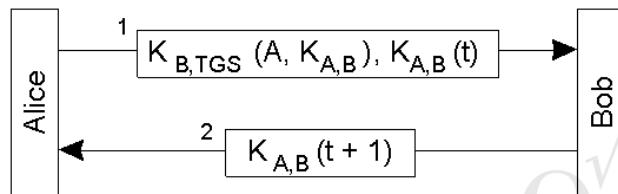
## Session Key and Authenticator

- Session key returned along with ticket
  - Charon reply = [sessionkey, ticket]
  - Ticket = [sessionkey:username:address:servicename:lifespan:timestamp]
- Authenticator used to contact service
  - Authenticator = {username:address} encrypted with sessionkey
- Client sends authenticator and ticket to service
- The service now knows:
  - The ticket's lifespan and timestamp
  - The ticket-owner's name and address
  - The sessionkey
- The session key authenticates the authenticator and vice versa
- Both can be stolen at the same time and reused
- Include timestamp and *short* lifespan in authenticators
- The session key is also used to authenticate the service

## Authentication in Kerberos

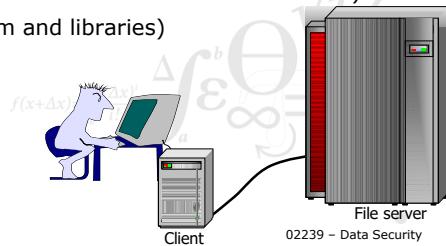


## Setting up a secure channel in Kerberos



## User Authentication in Distributed Systems

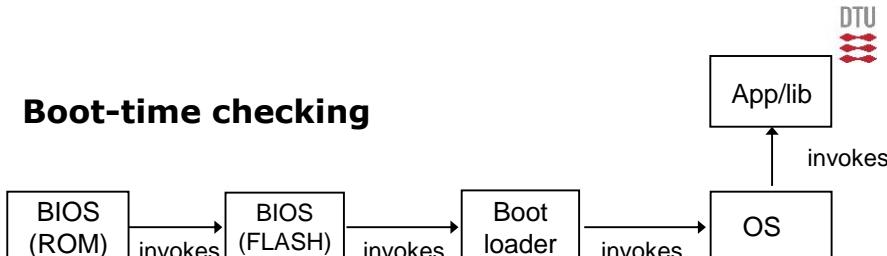
- How can we prove that a remote user is who he claims to be?
- User authentication takes place remotely at the client
  - Needs federated identity management
- Communication channel must be secure (CIA)
  - Integrity is required
  - Confidentiality and Availability as required
- Device authentication needed to determine if client is trusted
  - Client applications (programs that handle authentication tokens)
  - Client system (operating system and libraries)
  - Client hardware



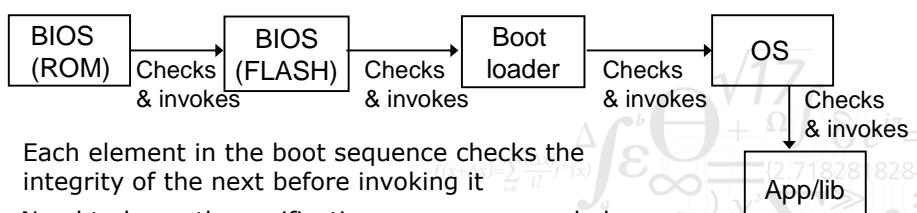
## Trusted Computing Group

- Trusted Computing Group (TCG) defines the TPM standard
  - TPM defines a special processor
    - *Tamper resistant hardware (environment for secure storage and processing)*
    - *Support for cryptographic operations*
- TPM provides the following functionalities
  - Protected capabilities
    - *Commands with exclusive access to shielded locations*
  - Shielded locations
    - *Domain where it is safe to access sensitive (shielded) data*
- TCG does not control the implementation
  - Vendors are free to differentiate the TPM implementation
  - TPM must still meet the protected capabilities and shielded locations requirements

## Boot-time checking



A well-defined sequence of software modules get executed at boot time.

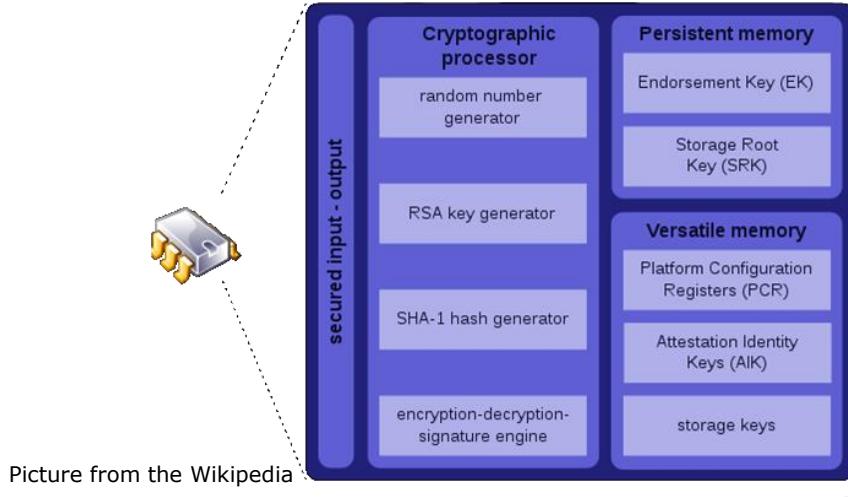


Each element in the boot sequence checks the integrity of the next before invoking it

Need to know the verification process succeeded

Trusted boot or secure boot

## TPM Architecture



52 DTU Compute Technical University of Denmark

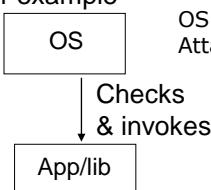
02239 – Data Security

## Platform Configuration Registers (PCRs)

- PCRs are used to securely measure software (by computing hash) during boot
- Each PCR can contain a SHA-1 hash value (20bytes)
  - At least 16 PCRs
- PCRs are reset to 0 at boot time
- Write to a PCR # n by extending it – **hash extension**

$$\text{TPM\_Extend}(n,D): \quad \text{PCR}[n] \leftarrow \text{SHA-1}(\text{PCR}[n] \parallel D)$$

For example



OS computes  $h_3 = \text{SHA-1}(\text{module3})$ ; stores  $\text{SHA-1}(0, h_3) \rightarrow \text{PCR}[3]$   
 Attacker substitutes module3 with module3',  $h_3' = \text{SHA-1}(\text{module3}')$

53 DTU Compute Technical University of Denmark

02239 – Data Security

## Trusted/Secure Boot Sequence

- At power-up PCR[n] initialized to 0
- BIOS boot block executes
  - Calls `PCR_Extend( n, <BIOS code> )`
  - Then loads and runs BIOS post boot code
- BIOS executes:
  - Calls `PCR_Extend( n, <MBR code> )`
  - Then runs MBR (master boot record)
- MBR executes:
  - Calls `PCR_Extend( n, <OS loader code, config params> )`
  - Then runs OS loader

Which PCRs to use is defined by specifications

$$f(x+\Delta x) = \sum_{l=0}^n \frac{(\Delta x)^l}{l!} f^{(l)}(x)$$

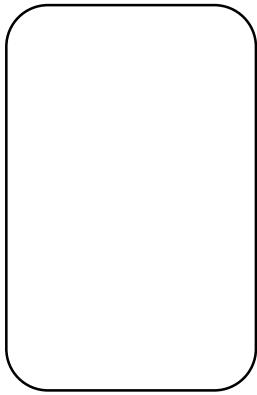
## Thinking about Authentication





## OAUTH 2.0

- Nate Barbetinitis slides??



$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$
$$\int_a^b \Theta_{\infty}^{\sqrt{17}} \delta e^{i\pi} = \frac{\Omega}{2.718281828} \sum_{n=1}^{\infty} \frac{(-1)^n}{n^n} x^n$$

## Protection Mechanisms



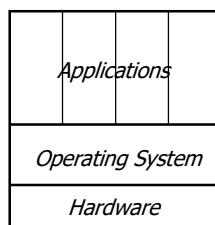
$$\Delta \int_a^b \varepsilon^{\sqrt{17}} \Theta + \Omega \int \delta e^{i\pi} = \{2.71828182845904523536028747135266249 \dots\}$$

DTU Compute

Department of Applied Mathematics and Computer Science

## Protection in Operating Systems

- OS implements the fundamental security mechanisms



- What needs to be protected
  - Memory
  - Sharable I/O devices (disks, network interfaces, ...)
  - Serially reusable I/O devices (printers, tape drives, ...)
  - Shared programs and sub-procedures (services)
  - Shared data (files, databases, ...)

## Separation of Subjects' Access to Objects

- Separation forms basis for most protection mechanisms
- Processes may have different security requirements
- Physical separation
  - Different processes use different physical objects (separate hardware)
- Temporal separation
  - Different processes are executed at different times
- Logical separation
  - OS creates illusion of physical separation
- Cryptographic separation
  - Processes conceal data and computations in a way that makes them unintelligible to outside processes
    - Encrypt data
    - Some algorithms for computation on encrypted data exist
      - Homomorphic encryption

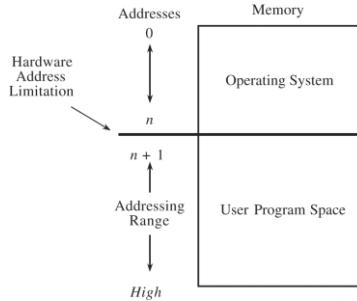
## Principles of Protection

- Do not protect
  - Appropriate when physical/temporal separation is used
- Isolate
  - Processes are completely unaware of other processes (virtual machines)
- Share all or share nothing
  - Public or private data
- Selective sharing (*share via access limitations/share by capabilities*)
  - OS enforces a policy that defines how objects can be shared by users
    - Mandatory-/Discretionary policies
    - Generally implemented in a reference monitor
- Usage control (*limit use of an object*)
  - Restricts use of objects after access has been granted
  - Typical goal for DRM systems
    - Applications require support from hardware and OS

## Fence

### Memory and Address Space Protection I

- Separation between OS and user programs

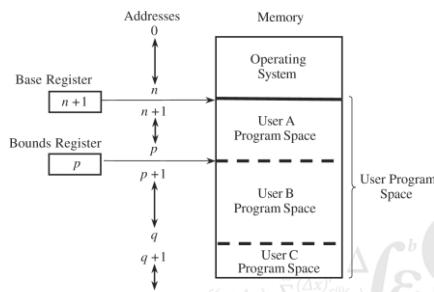


- Predefined memory address
  - Operating system resides below this address
  - Programs are loaded from this address and cannot access OS memory
  - Special *Fence Register* allows re-allocation of memory

## Base/Bounds Registers

### Memory and Address Space Protection II

- Fence only protects in one direction (underflow)
- Base/Bounds registers protect in both directions
  - Base register corresponds to fence

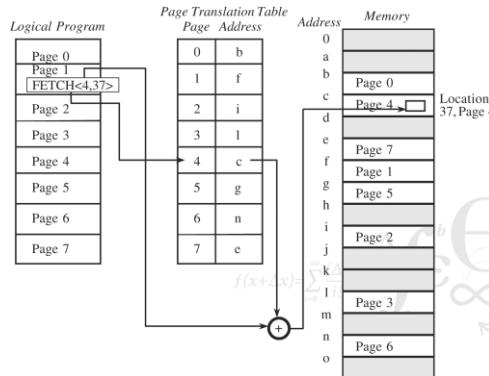


- Each process has its own pair of base/bounds registers
  - Protects processes from each other
    - One man's bounds is another man's base*

## Paging

### Memory and Address Space Protection III

- Variable size segments are difficult/expensive to manage
- Paging introduces fixed sized segments (page frames)
- Typically powers of 2 between 512 and 4096 bytes

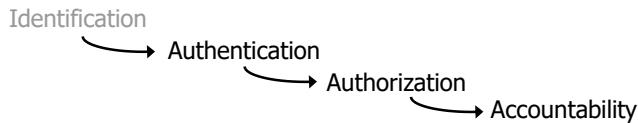


## Paging II

### Memory and Address Space Protection V

- Page translation tables define the addressable memory of a process
  - Managed by the OS
    - Prevent user processes from "mapping" OS memory into its address space
- There is no logical structure to memory pages
  - Data with different security requirements may reside on the same page
- Security benefits of paging include:
  - Address references can be checked for protection
    - When the relevant page is "mapped" (inserted in page translation table)
  - Users can share data by sharing physical memory pages
    - Access rights do not have to be the same for all users
  - Users cannot access main memory directly
- Most current systems implement a paging architecture

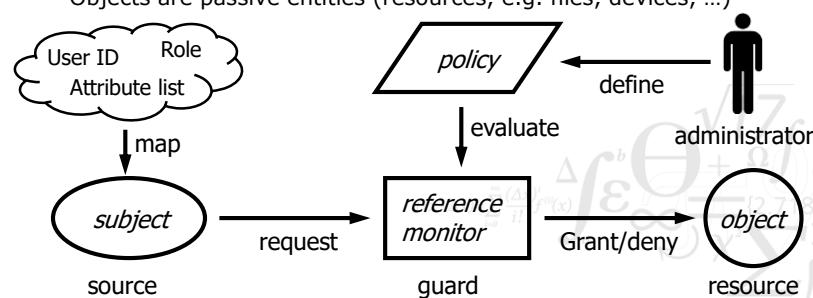
## Classic view of security



- Authentication
  - Verifies the claimed identity of subjects
- Authorization
  - Enforces access control policy
    - Decides whether a subject has the right to perform an operation on an object
- Accountability
  - Records security relevant events
    - What happened? and who did what?

## Access Control Model

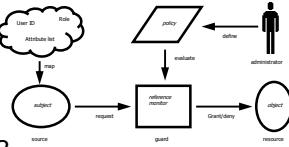
- Security policy is evaluated every time an object is accessed
  - Reference Monitor mediates all access by subjects to objects
    - Guards access to object
    - Interprets access control policy
  - Subjects are active entities (users, processes)
  - Objects are passive entities (resources, e.g. files, devices, ...)



## Reference Monitors in Distributed Systems

- Concept developed for centralised Operating Systems

- Policy enforced by components in the OS
- Policy defined by local system administrators
- Policy based on local information



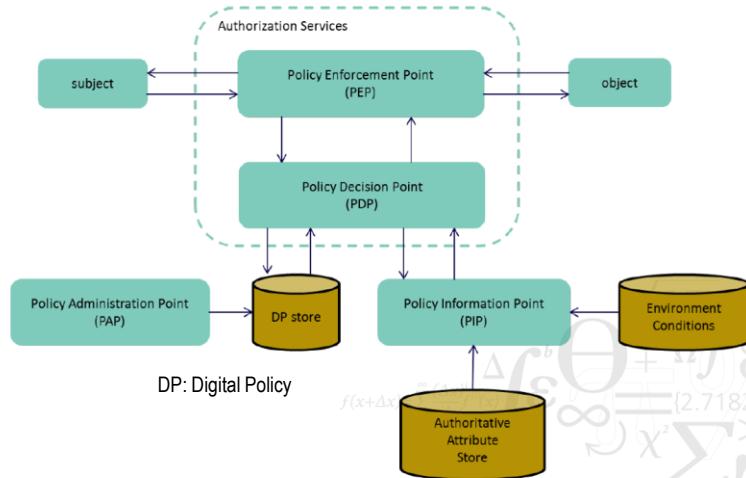
- How does this extend to distributed systems?

- Resources hosted on different machines
  - Possibly managed by different local administrators
  - Possibly belonging to different administrative domains
- Access Control decisions may be federation of local policies
  - Federated identity management
  - Federated access control policies
- Distributed enforcement of policies

## Access Control Architectural Elements

- PEP: Policy Enforcement Point:
  - Grants or denies access
- PDP: Policy Decision Point:
  - Decides whether access should be granted or denied
  - Uses the policies recorded in the Policy Store
- PAP: Policy Administration Point
  - Manages the Policy Store: adds, removes and modifies policies
- PIP: Policy Information Point
  - Provides the information that the PDP needs to make decisions
    - Model parameters, roles, attributes, hierarchies, constraints
    - State of the environment:
      - Examples: Time of Day, Normal Working Hours, ...
      - Location of users and/or resources
      - Etc.

## Access Control Architecture



Source: NIST Special Publication 800-162

13 DTU Compute Technical University of Denmark

02239 – Data Security

## Mapping Subjects

- Identity Based Access Control
  - Permissions are granted directly to users
  - Unique system identifier (UID) for every user
  - User identity must be verified before use (authentication)
- Role Based Access Control
  - Permissions are granted to roles
  - Users assigned one or more roles
  - User identity must be verified before role is assumed (authentication)
- Attribute Based Access Control
  - Permissions depend on user's attributes
  - Users must prove possession of attributes
    - *Attributes are often encoded in certificates*
  - Use of certificates often require user's public-key
    - *Use of public-key certificate implies authentication*
- Ultimately, users must prove identity to exercise access rights

14 DTU Compute Technical University of Denmark

02239 – Data Security

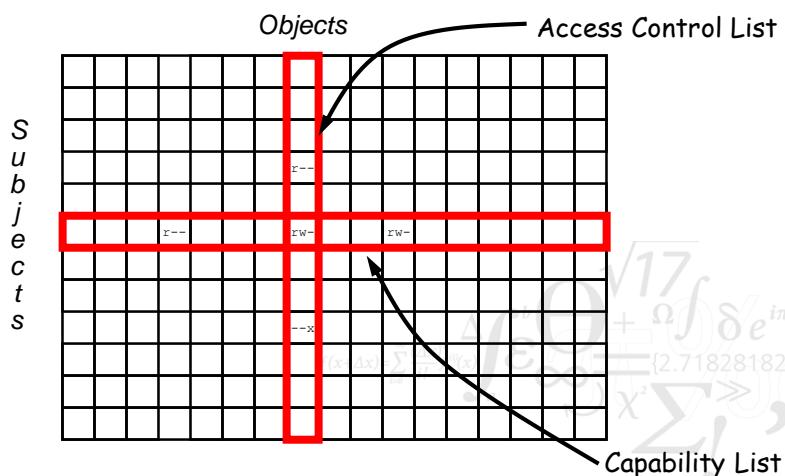
## Access Control Matrix Model

- Access Control Matrix defined by
  - Set of subjects S (active entities in the system)
  - Set of objects O (passive entities in the system)
  - Set of rights R (defines operations that subjects can do on objects)
- A denotes the entire access control matrix
  - Encodes the access rights of subjects to objects
  - A is often a sparse matrix
- $a[s,o]$  denotes the element at row  $s$ , column  $o$ ;  $a[s,o] \in R$

subjects	objects			
	file 1	file 2	process 1	process 2
process 1	read, write, own	read	read, write, execute, own	write
process 2	append	read, own	read	read, write, execute, own

## Representing the Access Control Matrix

- The Access Control Matrix is often sparse



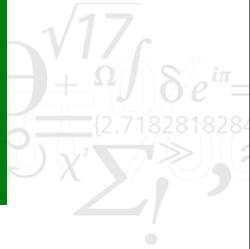
## Access Control Lists

- Associated with every object in the system
  - List of pairs:  $\langle \text{subject name}, \text{access rights} \rangle$
- Access is granted if
  - Subject name is in the list
  - Access rights include requested operation
  - Otherwise access is denied
- Some ACL systems allow special default actions (**grant** or **deny**)
  - Useful with negative access rights
    - *ACL becomes a list of people to exclude*
- Delegation is difficult
  - Requires the right to modify the ACL
- Questions about access rights
  - Easy to know who may access an object
  - Difficult to know what objects a subject may access

## Capabilities

- List of capabilities is associated with every subject in the system
  - List of pairs:  $\langle \text{unique object identifier}, \text{access rights} \rangle$
- Capabilities are used to reference the object
  - Without a capability, object cannot be addressed
  - Access is granted if rights in the capability includes requested operation
- Three types of capabilities
  - Hardware capabilities
  - Segregated capabilities
  - Encrypted capabilities
- Capabilities are easy to delegate
- Questions about access rights
  - Difficult to know who may access an object (who has a capability)
  - Easy to know what objects a subject may access (and how)

## Break

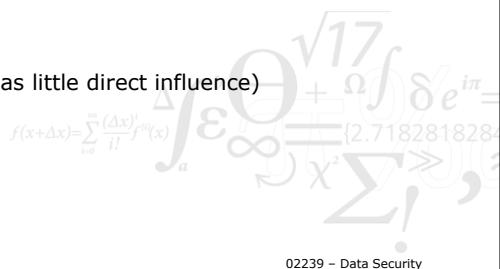


19 DTU Compute Technical University of Denmark

02239 – Data Security

## Security Policies

- Prevent disclosure or corruption of sensitive data
  - Controlled access to protected resources
  - Isolation (confinement)
  - Separation of functions (place order and sign check)
  - Well formed transactions
- Mandatory Access Control
  - System defines policies (users have little direct influence)
    - System "owns" resources
- Discretionary Access Control
  - Users define policies (system has little direct influence)
    - User "owns" resources

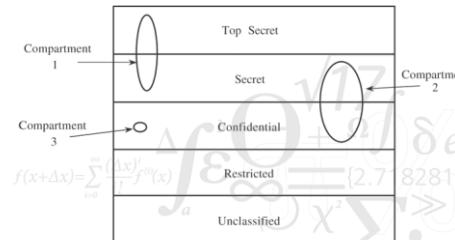
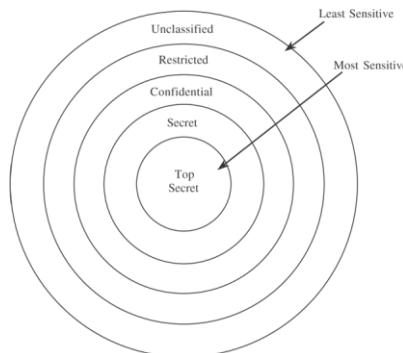


20 DTU Compute Technical University of Denmark

02239 – Data Security

## Military Access Control Policies

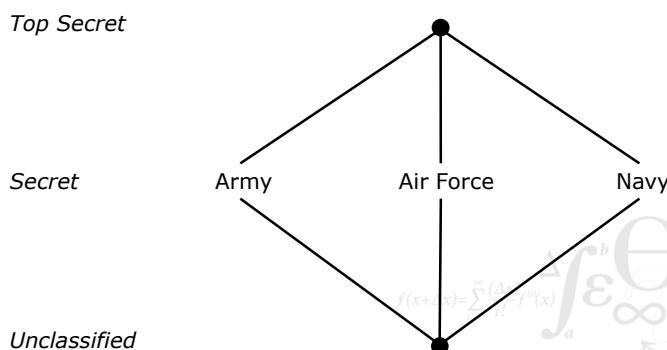
- Keeping military plans secret
  - Confidentiality is primary concern
    - *Need-to-know principle*
  - Traditional model based on safes and marked binders



21 DTU Compute Technical University of Denmark

02239 – Data Security

## Access Control Lattice



22 DTU Compute Technical University of Denmark

02239 – Data Security

## Bell & LaPadula

### Multilevel Security

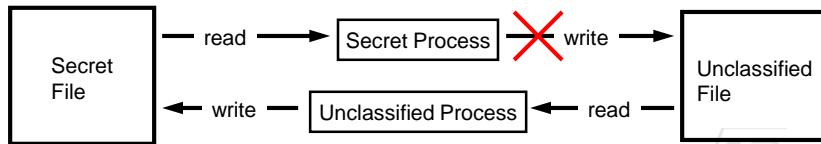
- Mandatory access control model
  - Separate users with multiple levels of privileges on the same system
  - Military system
    - *Security labels: unclassified ≤ restricted ≤ confidential ≤ secret ≤ top secret*
- Basic definitions:
  - **object:** passive entity, stores information
  - **subject:** active entity, manipulates information
  - **label:** identifies the *secrecy classification* of the object
  - **clearance:** specifies the most secret class of information available to the subject
  - **permission:** specifies the operations that the subject is allowed to invoke on the object, the model defines: *read, write, append, and execute* permissions

## Bell & LaPadula II

- **Domination:** (relation)
  - Label (or clearance)  $A$  is said to *dominate* a label  $B$ , if a flow of information from  $B$  to  $A$  is authorized
  - $A$  dominates  $B$  is written  $A \geq B$
- **Security Rules:**
  - Simple security condition
    - *Subject s may only access an object o, if the clearance of s dominates the label of o*
  - The \*-property
    - *Subject s may only use the content of an object o<sub>1</sub> to modify an object o<sub>2</sub>, if the label of o<sub>2</sub> dominates the label of o<sub>1</sub>*

*NB! A consequence of the \*-property is that objects tend to rise slowly towards the highest classification*

## Bell & LaPadula III



## Bell & LaPadula IV

- Implementation issues:
  - Unavailability of passive objects
    - Objects must be activated before they are accessed
  - Tranquillity principle
    - The label of an active object cannot be changed
  - Initialization of objects
    - The initial state of an object does not depend on any previously allocated resource
- The system call open() is an example of activation

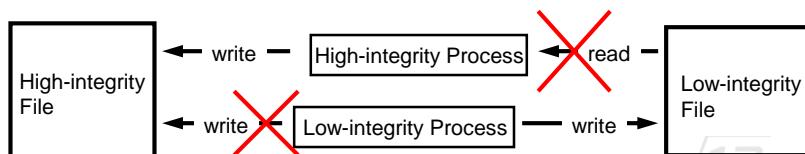
## Biba Integrity Model

- In civilian systems, integrity is more important than secrecy
- Biba defines an integrity model similar to the Bell & LaPadula model
  - Introduces integrity classes
  - Prevents information from objects with low integrity to contaminate objects with a higher integrity

### Integrity Rules:

1. Simple integrity: Subject  $s$  can only modify an object  $o$  if the integrity class of  $s$  dominates the integrity class of  $o$
2. Confined integrity: Subject  $s$  can only read the content of an object  $o$  if the integrity class of  $o$  dominates the integrity class of  $s$

## Biba Integrity Model II



## Role Based Access Control (RBAC)

- In many cases, authorization should be based on the function (role) of the subject in the manipulation of the object
  - Consider the following example:
    - *Anne, accountant for DTU Compute, has access to financial records*
    - *She leaves*
    - *Eva is hired as the new accountant, so she now has access to those records*
  - How are all the necessary permissions transferred from Anne to Eva?
- Examples of Roles:
  - Function in a bank
    - *Teller clerk, Financial advisor, Branch manager, Regional manager, Bank director*
  - Function in a hospital
    - *Doctors (GP, consultant, treating doctor, ...), Nurses (ward nurse, nurse, ...), Hospital administrators*
  - Functions at a university
    - *Academics (teachers, research fellows, ...), Non-academic staff (secretaries, system administrators, ...), Students*

## Common RBAC Concepts

### Definitions:

- **Active role:**

$AR(s : subject) = \{\text{the active role for subject } s\}$

- **Authorized roles:**

$RA(s : subject) = \{\text{authorized roles for subject } s\}$

- **Authorized transactions:**

$TA(r : role) = \{\text{authorized transactions for role } r\}$

- **Predicate exec:**

$\text{exec}(s : subject, t : transaction) = \text{true iff } s \text{ can execute } t$

- **Session:**

*Binds a user to a set of currently activated roles*

## General RBAC Rules

### Rules:

#### 1. Role assignment:

$\forall s : subject, t : transaction \ (exec(s,t) \Rightarrow AR(s) \neq \emptyset)$

A subject can only execute a transaction if it has selected a role

#### 2. Role authorization:

$\forall s : subject \ (AR(s) \subseteq RA(s))$

A subject's active role must be authorized for the subject

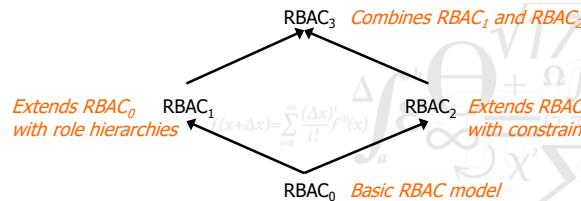
#### 3. Transaction authorization:

$\forall s : subject, t : transaction \ (exec(s,t) \Rightarrow t \in TA(AR(s)))$

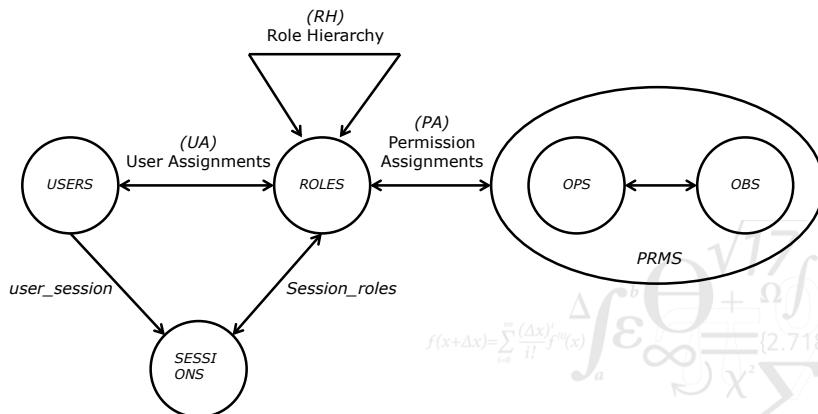
A subject can only execute a transaction if it is authorized for its active role

## RBAC96

- Role-Based Access Control was initially defined by Ferraiolo & Kuhn from NIST in 1992
- A family of related RBAC models were defined by Sandhu et al. in 1996 – this family is commonly known as RBAC96
  - RBAC96 forms the basis for most of the continued work on Role-based Access Control
- RBAC96 defines the following models:



## RBAC



## Attribute Based Access Control (ABAC)

- KeyNote [RFC 2704] builds on “assertions” (credentials)
  - Blaze, Feigenbaum, Ioannidis, Keromytis; 1999
- An assertion consists of two parts
  - Identification of an agent (could be the public-key)
  - Specification of an allowed operation on a resource
  - An assertion is digitally signed by the issuer
- Assertions may be provided by:
  - The system (from the security policy)
  - The agent itself (“credential”)
- An operation is allowed if there exists an assertion that permits the operation
  - Explicit permission from the issuer
  - Implicit through other assertions from the same issuer
    - *This requires an inference engine to derive new assertions.*

## Monotony in ABAC

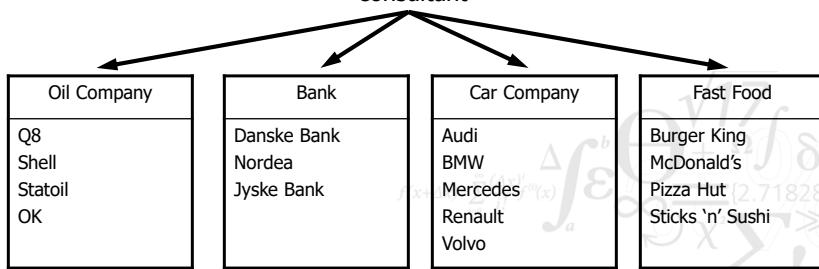
- Assertions are Monotonous
  - Addition of an assertion never disallows an operation
  - Deletion of an assertion never allows a prohibited operation
  - Everything is prohibited unless explicitly allowed
- Significance of monotony
  - Safe to use in distributed systems
    - *Lost assertions cannot break a policy*
  - Set of assertions that combines to allow an operation constitutes a proof that the security policy is enforced
  - Clients may collect signed assertions and send them to the server
    - *Offloads work from server to clients*
  - No conflicts are possible
    - *If an operation can be allowed based on the system's assertions, the operation will be allowed*

## ABAC in Practice

- Suitable for large distributed systems
  - Decentralized specification of security policies
  - Decentralized (autonomous) enforcement of security policies
- Simultaneously gives permission and the justification for allowing an operation
  - Set of assertions used to authorize the operation
- Allows dynamic evolution of security policies
  - Addition of new assertions may add new users, roles permissions or resources
- Not obvious how context may be encoded in assertions
  - This is one potential obstacle to its application in pervasive computing environments

## Chinese Wall Model

- Developed to avoid conflict of interest in consultants
- The consultancy firm divides clients into business areas
- Each consultant may work for several clients
  - a priori, no limitations are assumed
  - only *one* client in each business area is allowed



A consultant may work for any one company in each class

# 02239 Data Security Privacy

Sebastian Mödersheim



November 3, 2021

# Outline

- ① Why Privacy?
- ② Defining Privacy
- ③ Privacy in Corona Apps
- ④ Non-Technical Approaches
- ⑤ Mixes
- ⑥ Zero-Knowledge Protocols

# Outline

① Why Privacy?

② Defining Privacy

③ Privacy in Corona Apps

④ Non-Technical Approaches

⑤ Mixes

⑥ Zero-Knowledge Protocols

# Extreme Views

- “An honest person has nothing to hide!”



- “1984”



## An Example [from Mark Ryan]

- Parents and their teenage daughter:
  - ★ The daughter wants to go out, but not tell the parents where she goes.
  - ★ The parents want to know where she is in case of any emergency.
  - ★ Both are legitimate interests!
  - ★ Danger: the parents may overreach – out of concern for their daughter – not respecting her privacy.

# An Example [from Mark Ryan]

- Parents and their teenage daughter:
  - ★ The daughter wants to go out, but not tell the parents where she goes.
  - ★ The parents want to know where she is in case of any emergency.
  - ★ Both are legitimate interests!
  - ★ Danger: the parents may overreach – out of concern for their daughter – not respecting her privacy.
- Non-electronic solution:
  - ★ The daughter writes where she is going in a sealed letter.
  - ★ The parents can open the letter, but are compelled not to — unless there is an emergency
- Technical solution: with trusted hardware (e.g. a TPM)

# Why Privacy?



Why not vote in public?

# Why Privacy?

- Being observed, or believing to be observed, can have an influence on ones behavior.
  - ★ Feeling compelled/coerced to act according to others' expectations.
  - ★ Can mean a subtle restriction of the freedom.
- This is getting more sensitive than 20 years ago. Life leaves more traces in different IT systems. Technology allows for:
  - ★ cheap storage of data
  - ★ cheap evaluations of data
  - ★ cheap surveillance
- Protesting against a totalitarian regime, exchange information with other opposition members.

# Outline

① Why Privacy?

② Defining Privacy

③ Privacy in Corona Apps

④ Non-Technical Approaches

⑤ Mixes

⑥ Zero-Knowledge Protocols

# Defining Privacy

What *is* privacy really?

Several informal and semi-formal notions:

- Anonymity: the identity of the actors are protected
- Unlinkability: different actions of the same actor cannot be associated
- ...

# Formally Defining Privacy

- Difficult: it is not a classical secrecy problem!
  - ★ Example: poll where you can vote *yes* or *no*
  - ★ *yes* and *no* are values that the intruder knows, also the names of the voters may be no secrets.
  - ★ What is actually secret is: who voted *yes* and who voted *no*!

# Formally Defining Privacy

- Difficult: it is not a classical secrecy problem!
  - ★ Example: poll where you can vote *yes* or *no*
  - ★ *yes* and *no* are values that the intruder knows, also the names of the voters may be no secrets.
  - ★ What is actually secret is: who voted *yes* and who voted *no*!
- Inspiration from Cryptography: security formulated as equivalence notions:

$$\text{crypt}(k, 0) \sim \text{crypt}(k, 1)?$$

The intruder knows that the plain-text is either 0 or 1, the challenge is for him to tell correctly whether it is 0 or 1.

# Formally Defining Privacy

- Difficult: it is not a classical secrecy problem!
  - ★ Example: poll where you can vote *yes* or *no*
  - ★ *yes* and *no* are values that the intruder knows, also the names of the voters may be no secrets.
  - ★ What is actually secret is: who voted *yes* and who voted *no*!
- Inspiration from Cryptography: security formulated as equivalence notions:

$$\text{crypt}(k, 0) \sim \text{crypt}(k, 1)?$$

The intruder knows that the plain-text is either 0 or 1, the challenge is for him to tell correctly whether it is 0 or 1.

- Similar in formal verification (with perfect cryptography): static equivalence of frames.

# Alpha-Beta Privacy

- Novel approach to formulating privacy:
  - ★ Specify by a logical formula  $\alpha$  what information the intruder is allowed to know, e.g. election result:
$$\alpha \equiv v_1, \dots, v_n \in \{0, 1\} \wedge \sum_{i=0}^n v_i = 42$$
  - ★ Specify by a logical formula  $\beta$  what the intruder actually knows, e.g., observed cryptographic messages, what he knows about their structure, etc.
  - ★ Privacy: the intruder cannot derive anything from  $\beta$  except what already follows from  $\alpha$ .
- Ongoing research effort by Luca Viganò, Sébastien Gondron, Laouen Fernet and myself.
  - ★ We welcome more colleagues to join us e.g. for a MSc thesis!

# Outline

① Why Privacy?

② Defining Privacy

③ Privacy in Corona Apps

④ Non-Technical Approaches

⑤ Mixes

⑥ Zero-Knowledge Protocols

# Privacy in Corona Apps

Several Corona-Apps for contact tracing using Bluetooth Low Energy have been developed:

- DP3-T: Decentralized Privacy-Preserving Proximity Tracing
  - ★ open platform
  - ★ developed by several universities and research centers
  - ★ used by many European countries
- GAEN: Apple/Google Exposure Notification:
  - ★ very similar to DP3-T
  - ★ by Apple & Google
  - ★ used for instance in Denmark
- PEPP-PT/PEPP: Pan-European Privacy-Preserving Proximity Tracing
  - ★ Different idea: more information centrally stored
  - ★ Many developers left the consortium (and did DP3-T)

## DP3-T

- Clients generate day keys and ephemeral identities.
- Day keys:  $SK_{i+1} := h(SK_i)$ .
- Ephemeral identities: new identity in short intervals, say 15 minutes.  $EphID_{i,j} := prg(SK_i, j)$  for the  $j$ -th period on day  $i$ .
- Normal operation: exchange current ephemeral ID with devices in proximity, store all received ephemeral IDs that were for ca. 10 minutes in proximity.
- When sick and user gives consent, publish the day keys  $SK_i$  for the relevant days on a server.
  - ★ In most circumstances, the server will also need a proof that the submitter has indeed tested positive.
- Every phone can download the published  $SK_i$ , compute the ephemeral IDs and compare them to the proximity list.
- How much is privacy is sacrificed here? What can an attacker learn about infected people and their identities?

# Outline

- ① Why Privacy?
- ② Defining Privacy
- ③ Privacy in Corona Apps
- ④ Non-Technical Approaches
- ⑤ Mixes
- ⑥ Zero-Knowledge Protocols

# The Law

## Data Protection Laws

GDPR regulations of the EU

Personal Data must be “handled with care”

- Personal data includes e.g. name, date of birth, CPR-number, address, medical data, ...
- Always think of the **purpose** of storing such data
  - ★ Why the data has to be stored?
  - ★ Who shall have access to it?
  - ★ How long shall it be stored?
- Protect reasonably against theft
- Transparency
  - ★ Every person has the right to their data
  - ★ Consensus, information what is stored, right to correct errors, and right to be forgotten

## The Tom Waits Approach



*I was born in the back seat of a Yellow Cab  
in a hospital loading zone and with the meter  
still running. I emerged needing a shave and  
shouted 'Time Square, and step on it!'*

# The Tom Waits Approach



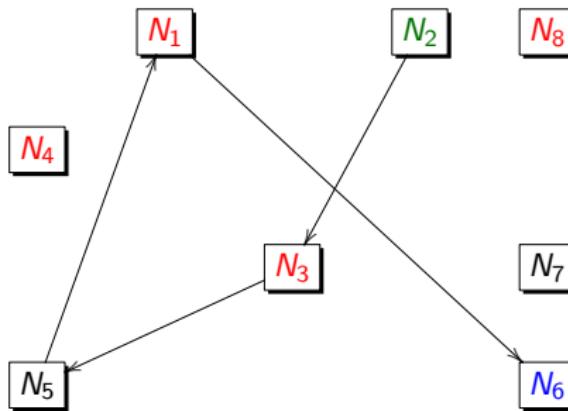
*I was born in the back seat of a Yellow Cab  
in a hospital loading zone and with the meter  
still running. I emerged needing a shave and  
shouted 'Time Square, and step on it!'*

- Tom Waits has told many absurd stories about his life.
- Hard to tell what is true. (Was he born in a taxi?)

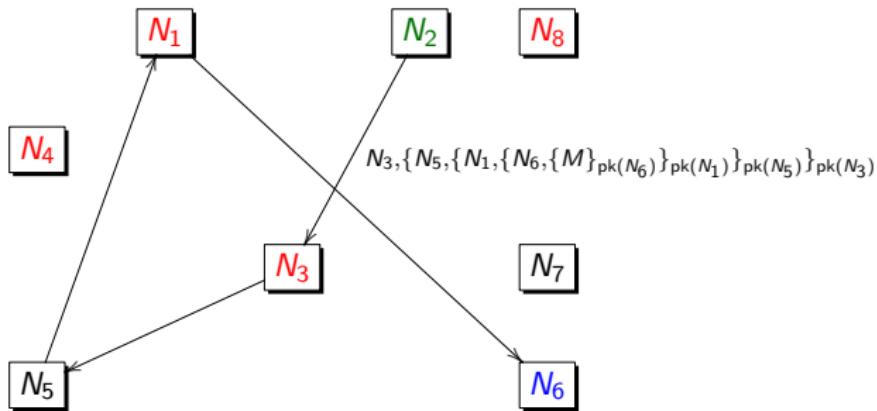
# Outline

- ① Why Privacy?
- ② Defining Privacy
- ③ Privacy in Corona Apps
- ④ Non-Technical Approaches
- ⑤ Mixes
- ⑥ Zero-Knowledge Protocols

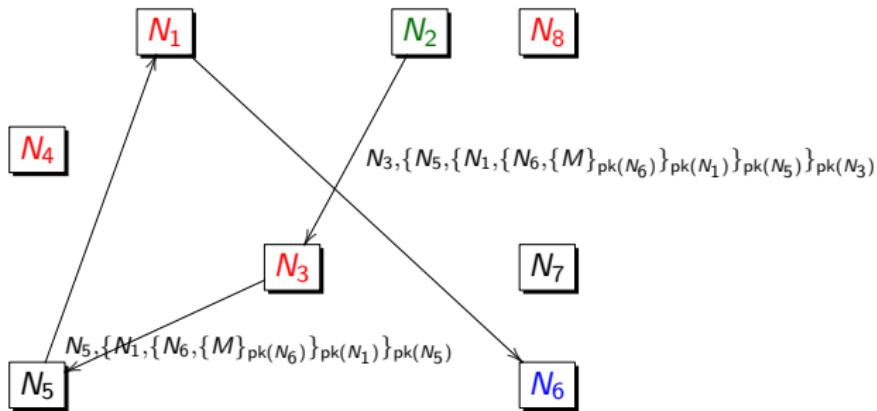
# Mixes/Onion Routing



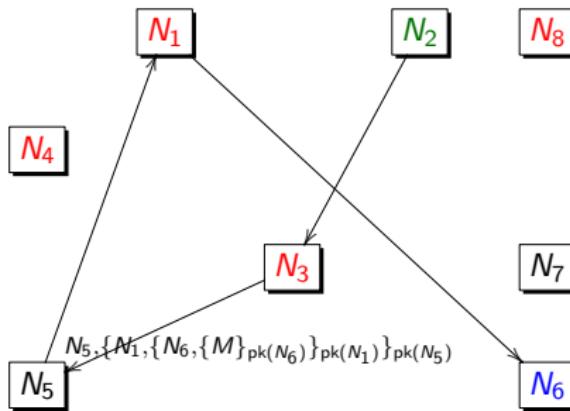
# Mixes/Onion Routing



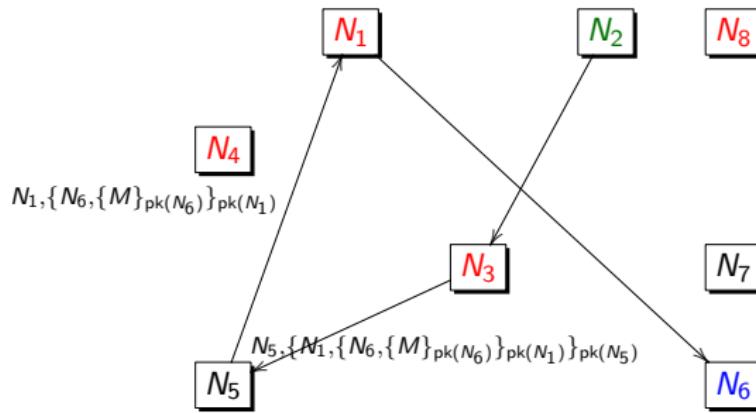
# Mixes/Onion Routing



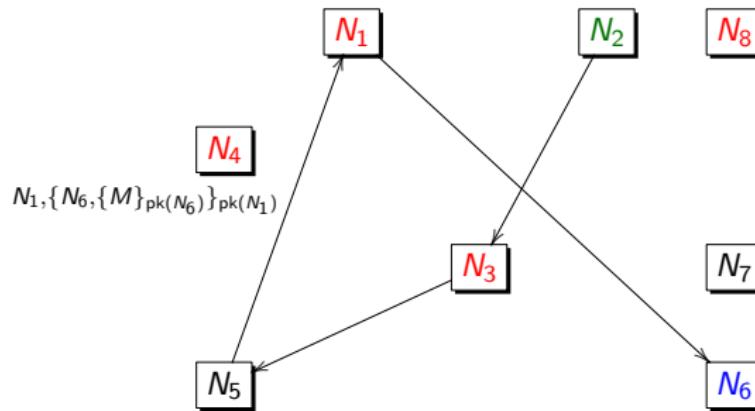
# Mixes/Onion Routing



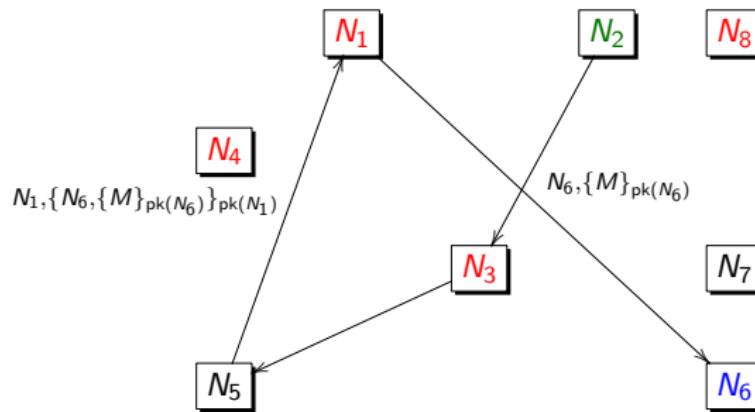
# Mixes/Onion Routing



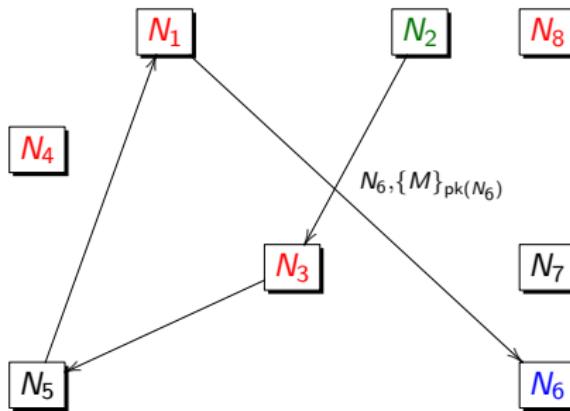
# Mixes/Onion Routing



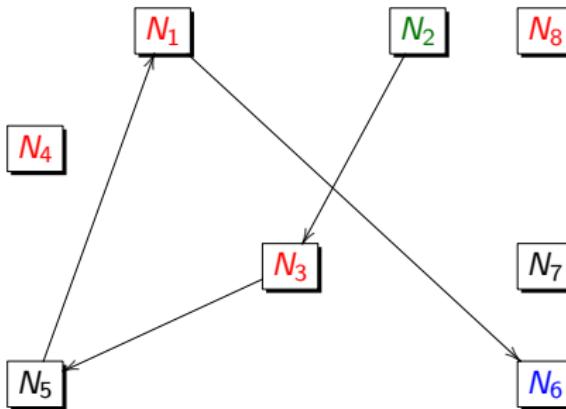
# Mixes/Onion Routing



# Mixes/Onion Routing

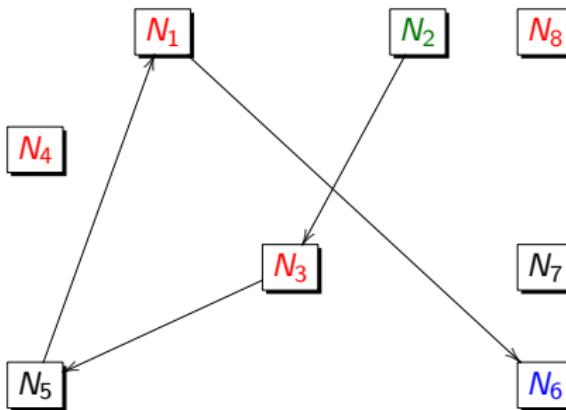


# Mixes/Onion Routing



- Each node knows only who is the previous/next on the chain
- Only the **destination** learns the message  $M$
- Nobody knows who is **source** except the source itself
- Nobody knows who is the **destination** (except source and destination)
- Attacker model: can see all exchanged messages and may control **some nodes** (may include **destination**)

# Mixes/Onion Routing: Assumptions



- Sender knows the true public key of all nodes.
- At least one node on the path is not controlled by the attacker
- Traffic is evenly distributed between all nodes
- Replay of messages is prevented
- Length of an encrypted message does not reveal the number of encryption layers.
- Note also: if **destination** is not part of mix network, cleartext messages are transmitted on the last leg.

# Outline

- ① Why Privacy?
- ② Defining Privacy
- ③ Privacy in Corona Apps
- ④ Non-Technical Approaches
- ⑤ Mixes
- ⑥ Zero-Knowledge Protocols

# Idea

## Zero-knowledge proofs

In zero-knowledge proofs we can usually specify a **statement** that is being proved.

- Definitely, that statement is revealed to the verifier
- The verifier (or others) should not learn anything else
- Everybody can draw conclusions from everything they learned

# Zero-Knowledge Protocols

## Scenario

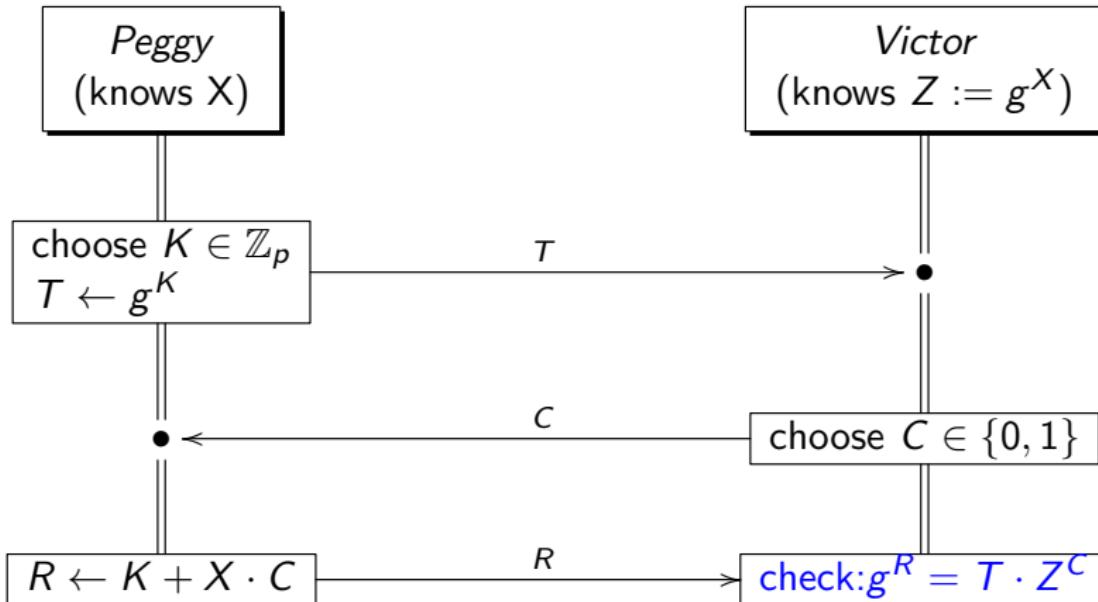
- A Prover Peggy and a Verifier Victor
- Peggy has a secret and wants to convince Victor of that fact **without** telling the secret.
- Example Schnorr protocol:
  - ★ Victor knows a public value  $Z \in \mathbb{Z}_p^*$ .
  - ★ Peggy wants to prove that she **knows**  $X \in \mathbb{Z}_p^*$  with  $Z \equiv_p g^X$ .
- Why? What's the point of proving this?

# Zero-Knowledge Protocols

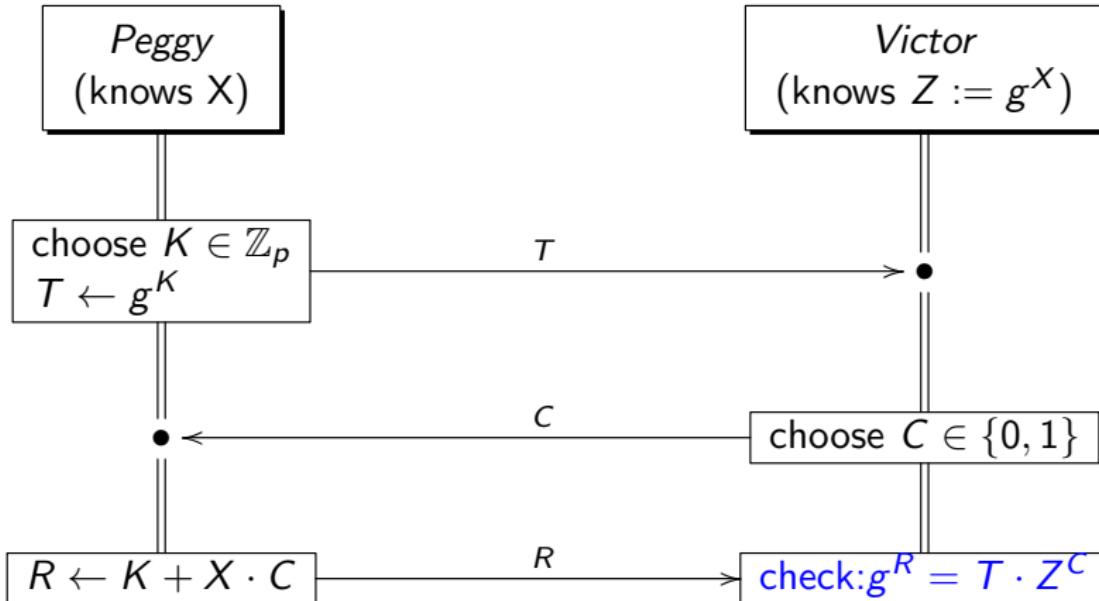
## Scenario

- A Prover Peggy and a Verifier Victor
- Peggy has a secret and wants to convince Victor of that fact **without** telling the secret.
- Example Schnorr protocol:
  - ★ Victor knows a public value  $Z \in \mathbb{Z}_p^*$ .
  - ★ Peggy wants to prove that she **knows**  $X \in \mathbb{Z}_p^*$  with  $Z \equiv_p g^X$ .
- Why? What's the point of proving this?
  - ★ Peggy can authenticate this way
  - ★ Anonymous credential systems ("Private authentication")
  - ★ Voting protocols like Helios/Belenios
  - ★ Alternatives to Proof-of-Work in blockchain implementations.

# Schnorr: the Protocol



# Schnorr: the Protocol



If Peggy worked correctly:

$$g^R = g^{K+X \cdot C} = g^K \cdot g^{X \cdot C} = T \cdot Z^C$$

## Schnorr: Trying to Cheat

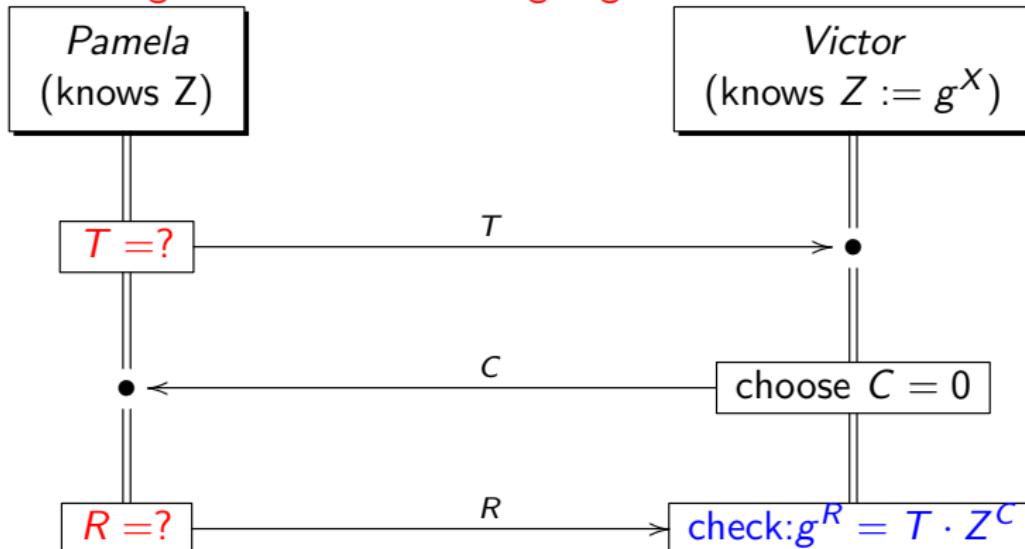
Pamela does not know the secret, but tries to prove its knowledge.

Pamela guesses that Victor is going to choose  $C = 0$

# Schnorr: Trying to Cheat

Pamela does not know the secret, but tries to prove its knowledge.

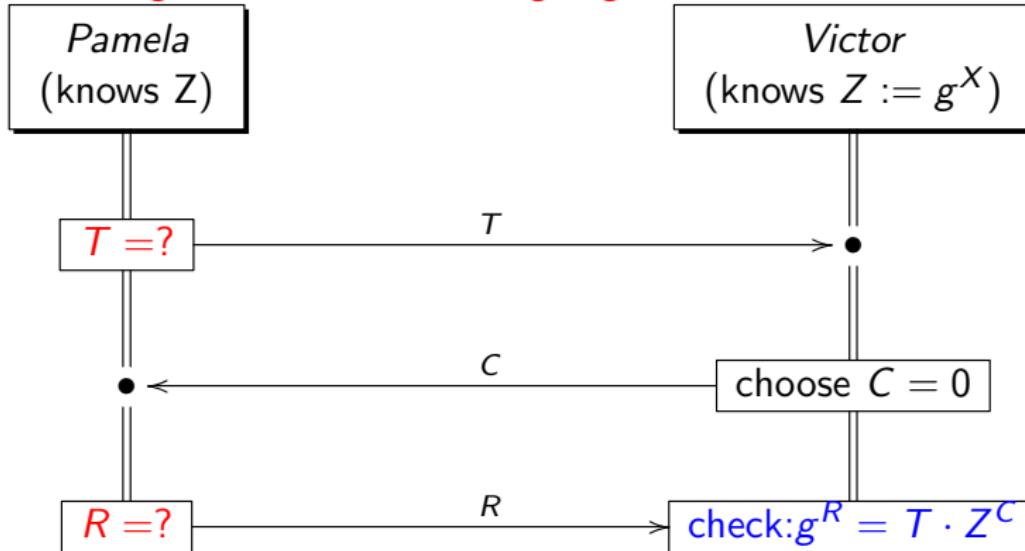
Pamela guesses that Victor is going to choose  $C = 0$



# Schnorr: Trying to Cheat

Pamela does not know the secret, but tries to prove its knowledge.

Pamela guesses that Victor is going to choose  $C = 0$



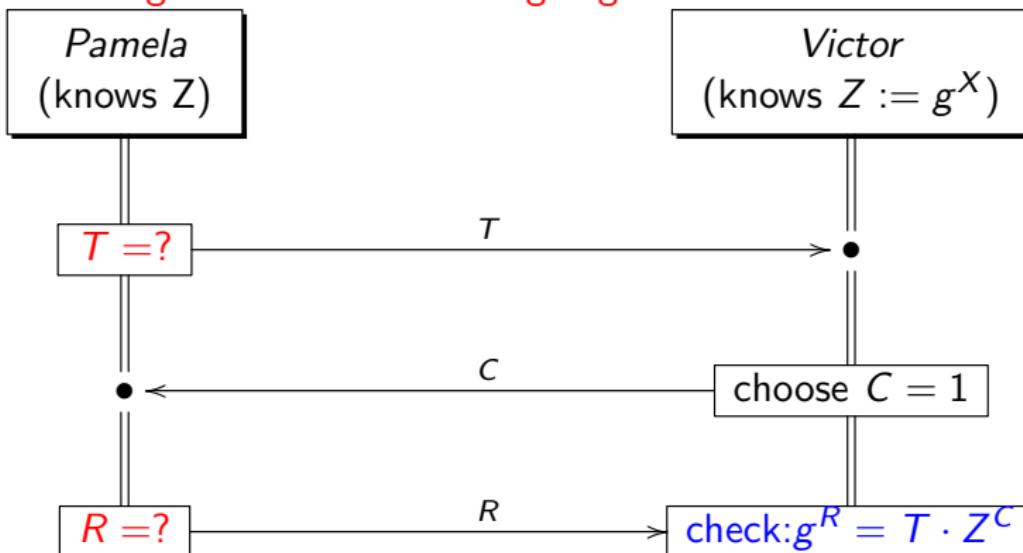
Strategy for Pamela: Choose  $R$  randomly and set  $T = g^R$ .

# Schnorr: Trying to Cheat

Pamela guesses that Victor is going to choose  $C = 1$

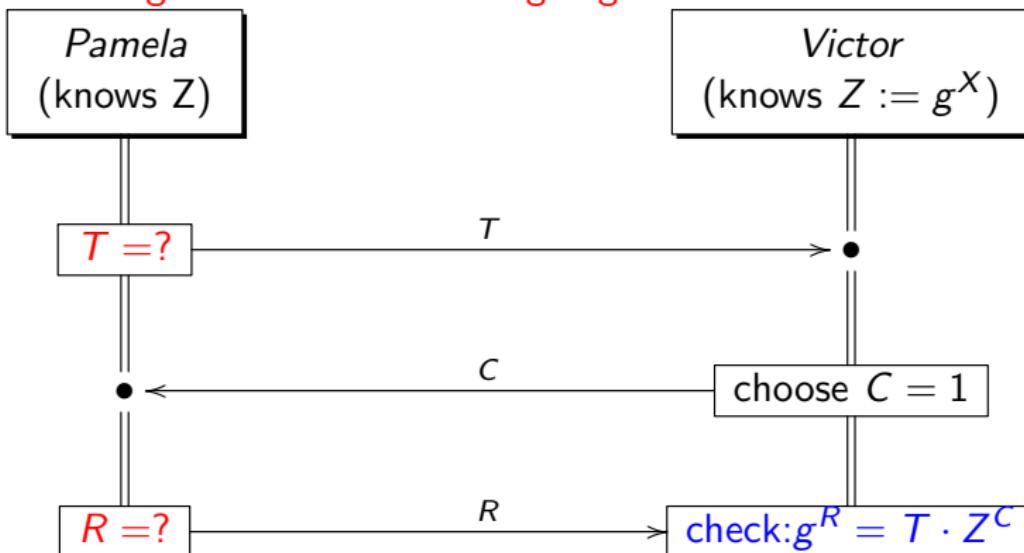
# Schnorr: Trying to Cheat

Pamela guesses that Victor is going to choose  $C = 1$



# Schnorr: Trying to Cheat

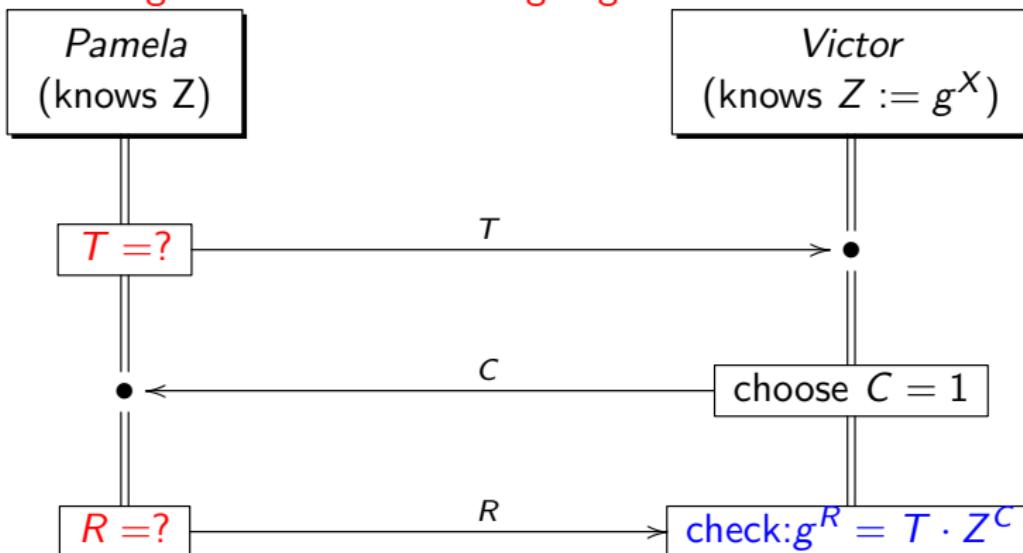
Pamela guesses that Victor is going to choose  $C = 1$



Find numbers  $T$  and  $R$  such that  $g^R = T \cdot Z$ .

# Schnorr: Trying to Cheat

Pamela guesses that Victor is going to choose  $C = 1$

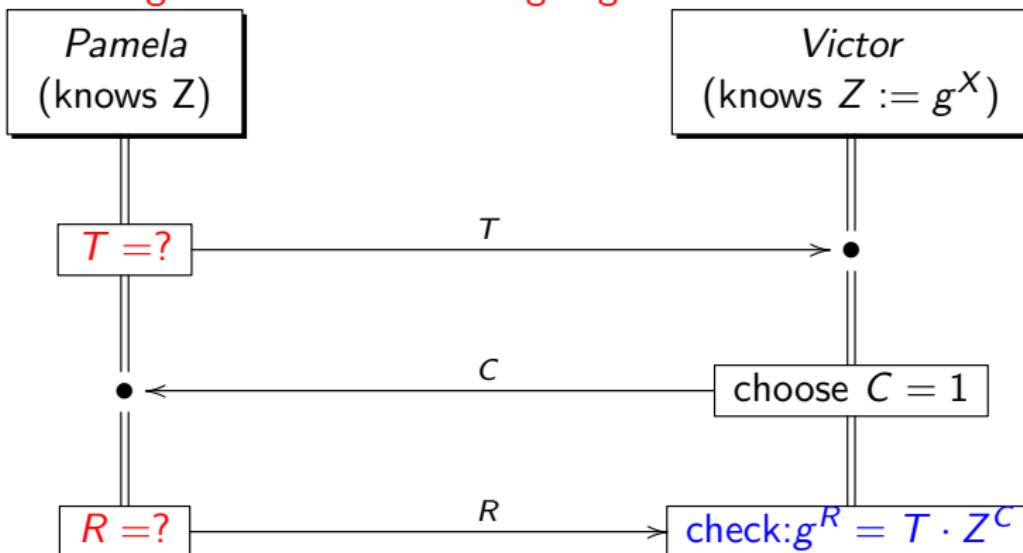


Find numbers  $T$  and  $R$  such that  $g^R = T \cdot Z$ .

Hint: division modulo  $p$  is also easy!

# Schnorr: Trying to Cheat

Pamela guesses that Victor is going to choose  $C = 1$



Find numbers  $T$  and  $R$  such that  $g^R = T \cdot Z$ .

Hint: division modulo  $p$  is also easy!

**Strategy for Pamela:** Choose  $R$  randomly, set  $T = \frac{g^R}{Z}$ .

## Schnorr: Trying to Cheat

- Pamela has a strategy to cheat if  $C = 0$ :

Choose  $R$  randomly, set  $T = g^R$ .

## Schnorr: Trying to Cheat

- Pamela has a strategy to cheat if  $C = 0$ :

Choose  $R$  randomly, set  $T = g^R$ .

- Pamela has a strategy to cheat if  $C = 1$ :

Choose  $R$  randomly, set  $T = \frac{g^R}{Z}$

## Schnorr: Trying to Cheat

- Pamela has a strategy to cheat if  $C = 0$ :

Choose  $R$  randomly, set  $T = g^R$ .

- Pamela has a strategy to cheat if  $C = 1$ :

Choose  $R$  randomly, set  $T = \frac{g^R}{Z}$

- Since  $T$  must be sent before  $C$  is known, she can **prepare** only for one possibility
  - ★ Chance of  $\frac{1}{2}$  to cheat (if  $C \in \{0, 1\}$  uniformly chosen)

## Schnorr: Trying to Cheat

- Pamela has a strategy to cheat if  $C = 0$ :

Choose  $R$  randomly, set  $T = g^R$ .

- Pamela has a strategy to cheat if  $C = 1$ :

Choose  $R$  randomly, set  $T = \frac{g^R}{Z}$

- Since  $T$  must be sent before  $C$  is known, she can **prepare** only for one possibility
  - ★ Chance of  $\frac{1}{2}$  to cheat (if  $C \in \{0, 1\}$  uniformly chosen)
- Claim: Pamela cannot do better (to prove...)

## Schnorr: Trying to Cheat

- Pamela has a strategy to cheat if  $C = 0$ :

Choose  $R$  randomly, set  $T = g^R$ .

- Pamela has a strategy to cheat if  $C = 1$ :

Choose  $R$  randomly, set  $T = \frac{g^R}{Z}$

- Since  $T$  must be sent before  $C$  is known, she can **prepare** only for one possibility
  - ★ Chance of  $\frac{1}{2}$  to cheat (if  $C \in \{0, 1\}$  uniformly chosen)
- Claim: Pamela cannot do better (to prove...)
- If Victor accepts only after  $n$  successful rounds of the protocol, the chance to cheat is only  $2^{-n}$ .

# Schnorr: Trying to Cheat

## Claim

Pamela has no strategy to cheat for unpredictable  $C$ .

## Proof sketch

# Schnorr: Trying to Cheat

## Claim

Pamela has no strategy to cheat for unpredictable  $C$ .

## Proof sketch

- Suppose Pamela has a strategy that works both for  $C = 0$  and for  $C = 1$ .

# Schnorr: Trying to Cheat

## Claim

Pamela has no strategy to cheat for unpredictable  $C$ .

## Proof sketch

- Suppose Pamela has a strategy that works both for  $C = 0$  and for  $C = 1$ .
- Then she has values  $T$ ,  $R_0$ , and  $R_1$  such that
  - ★ (If Victor chooses  $C = 0$ :)  $g^{R_0} = T$
  - ★ (If Victor chooses  $C = 1$ :)  $g^{R_1} = T \cdot Z$

# Schnorr: Trying to Cheat

## Claim

Pamela has no strategy to cheat for unpredictable  $C$ .

## Proof sketch

- Suppose Pamela has a strategy that works both for  $C = 0$  and for  $C = 1$ .
- Then she has values  $T$ ,  $R_0$ , and  $R_1$  such that
  - ★ (If Victor chooses  $C = 0$ :)  $g^{R_0} = T$
  - ★ (If Victor chooses  $C = 1$ :)  $g^{R_1} = T \cdot Z$
- Then she can compute  $Q = R_1 - R_0$ .

# Schnorr: Trying to Cheat

## Claim

Pamela has no strategy to cheat for unpredictable  $C$ .

## Proof sketch

- Suppose Pamela has a strategy that works both for  $C = 0$  and for  $C = 1$ .
- Then she has values  $T$ ,  $R_0$ , and  $R_1$  such that
  - ★ (If Victor chooses  $C = 0$ :)  $g^{R_0} = T$
  - ★ (If Victor chooses  $C = 1$ :)  $g^{R_1} = T \cdot Z$
- Then she can compute  $Q = R_1 - R_0$ .
- It holds that  $g^Q = g^{R_1 - R_0} = \frac{g^{R_1}}{g^{R_0}} = \frac{T \cdot Z}{T} = Z$ .

# Schnorr: Trying to Cheat

## Claim

Pamela has no strategy to cheat for unpredictable  $C$ .

## Proof sketch

- Suppose Pamela has a strategy that works both for  $C = 0$  and for  $C = 1$ .
- Then she has values  $T$ ,  $R_0$ , and  $R_1$  such that
  - ★ (If Victor chooses  $C = 0$ :)  $g^{R_0} = T$
  - ★ (If Victor chooses  $C = 1$ :)  $g^{R_1} = T \cdot Z$
- Then she can compute  $Q = R_1 - R_0$ .
- It holds that  $g^Q = g^{R_1 - R_0} = \frac{g^{R_1}}{g^{R_0}} = \frac{T \cdot Z}{T} = Z$ .
- So with  $Q = X$  she indeed knows the discrete logarithm of  $Z$ .

# Schnorr: Trying to Cheat

## Claim

Pamela has no strategy to cheat for unpredictable  $C$ .

## Proof sketch

- Suppose Pamela has a strategy that works both for  $C = 0$  and for  $C = 1$ .
- Then she has values  $T$ ,  $R_0$ , and  $R_1$  such that
  - ★ (If Victor chooses  $C = 0$ :)  $g^{R_0} = T$
  - ★ (If Victor chooses  $C = 1$ :)  $g^{R_1} = T \cdot Z$
- Then she can compute  $Q = R_1 - R_0$ .
- It holds that  $g^Q = g^{R_1 - R_0} = \frac{g^{R_1}}{g^{R_0}} = \frac{T \cdot Z}{T} = Z$ .
- So with  $Q = X$  she indeed knows the discrete logarithm of  $Z$ .
- **So it is not cheating!**

# Schnorr: Curious Victor

Victor would like to learn the secret  $X$ ...

## Zero-Knowledge Property

Victor learns nothing except the proved statement.

## Proof sketch

# Schnorr: Curious Victor

Victor would like to learn the secret  $X$ ...

## Zero-Knowledge Property

Victor learns nothing except the proved statement.

## Proof sketch

Consider a transcript of an exchange between Peggy and Victor. (Assume Victor chooses challenges randomly.)

$T_1$	$C_1$	$R_1$
$T_2$	$C_2$	$R_2$
$\dots$		
$T_n$	$C_n$	$R_n$

# Schnorr: Curious Victor

Victor would like to learn the secret  $X$ ...

## Zero-Knowledge Property

Victor learns nothing except the proved statement.

## Proof sketch

Consider a transcript of an exchange between Peggy and Victor. (Assume Victor chooses challenges randomly.)

$T_1$	$C_1$	$R_1$
$T_2$	$C_2$	$R_2$
...		
$T_n$	$C_n$	$R_n$

Create a fake-transcript using random challenges  $C'_i$

$C'_1$
$C'_2$
...
$C'_n$

# Schnorr: Curious Victor

Victor would like to learn the secret  $X$ ...

## Zero-Knowledge Property

Victor learns nothing except the proved statement.

## Proof sketch

Consider a transcript of an exchange between Peggy and Victor. (Assume Victor chooses challenges randomly.)

$$\begin{array}{lll} T_1 & C_1 & R_1 \\ T_2 & C_2 & R_2 \\ \dots & & \\ T_n & C_n & R_n \end{array}$$

Create a fake-transcript using random challenges  $C'_i$  and then filling the  $T'_i$  and  $R'_i$  according to the cheat strategies of Pamela for known challenges:

$$\begin{array}{lll} T'_1 & C'_1 & R'_1 \\ T'_2 & C'_2 & R'_2 \\ \dots & & \\ T'_n & C'_n & R'_n \end{array}$$

# Schnorr: Curious Victor

Victor would like to learn the secret  $X$ ...

## Zero-Knowledge Property

Victor learns nothing except the proved statement.

## Proof sketch

- Give the two transcripts to a third party. Can one tell the real from the fake?

# Schnorr: Curious Victor

Victor would like to learn the secret  $X$ ...

## Zero-Knowledge Property

Victor learns nothing except the proved statement.

## Proof sketch

- Give the two transcripts to a third party. Can one tell the real from the fake?
- **No:** statically indistinguishable. (Not even distinguishable with unbounded computing resources!)

# Schnorr: Curious Victor

Victor would like to learn the secret  $X$ ...

## Zero-Knowledge Property

Victor learns nothing except the proved statement.

## Proof sketch

- Give the two transcripts to a third party. Can one tell the real from the fake?
- **No:** statically indistinguishable. (Not even distinguishable with unbounded computing resources!)
- So, whatever information Victor is getting out of the transcript, he may have created that himself without Peggy!

# The Mafia-owned Restaurant

## Claim (Shamir)

I can go to a Mafia-owned restaurant and pay with my zero-knowledge proof-based credit card frequently, and yet the mobsters cannot impersonate me.

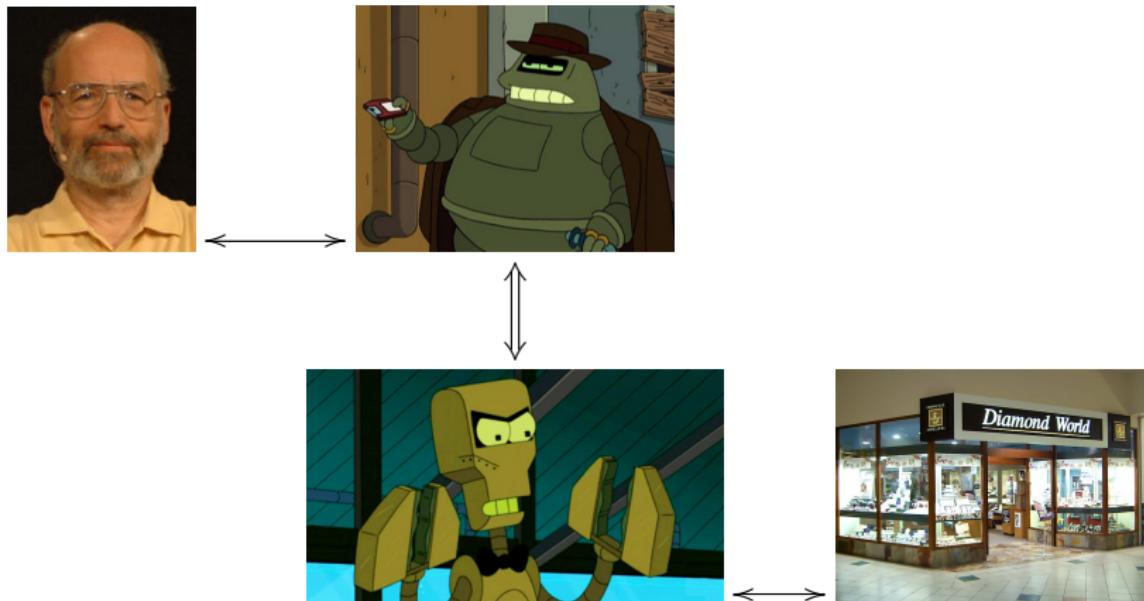
# The Mafia-owned Restaurant

## Claim (Shamir)

I can go to a Mafia-owned restaurant and pay with my zero-knowledge proof-based credit card frequently, and yet the mobsters cannot impersonate me.

They can, if they do it online.

# The Mafia Attack



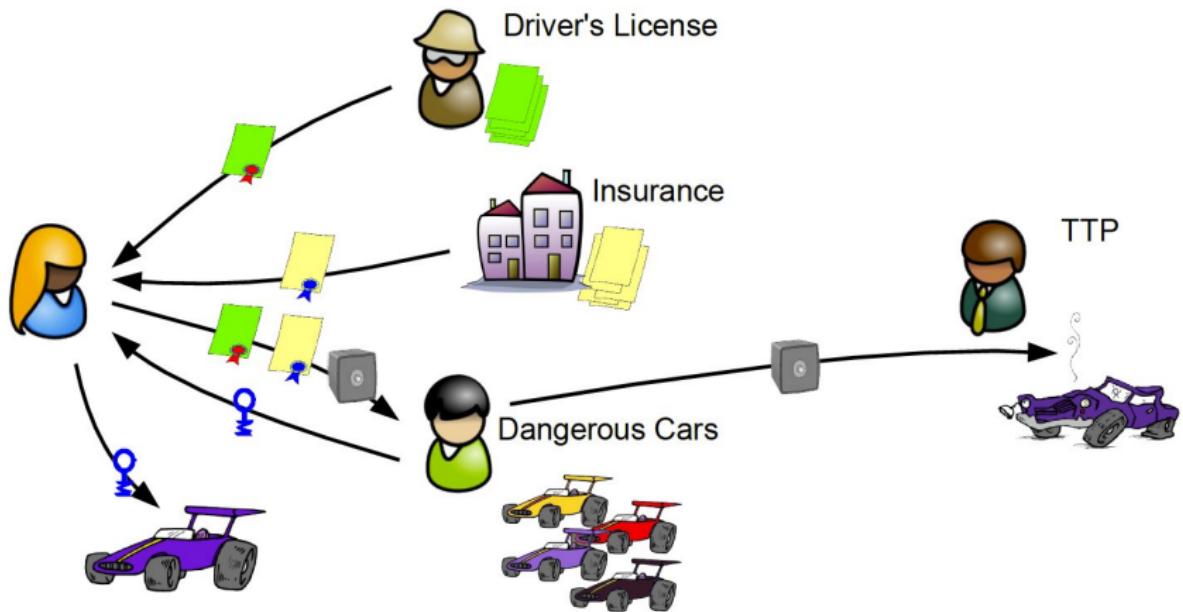
Classical Man in the middle attack: forward all messages!

# Countermeasures

- Distance bounding protocols.
- Link the intended verifier to the statement being proved.

# IBM's Idemix

A privacy friendly credential system



# IBM's Idemix

- Credentials: list of attribute-value pairs, signed by an issuer.  
`sign(copenhagen-commune; name="Johanna Gossner", birthdate=14/12/73, kind="Drivers License", class="3", ...)`
- Signatures are special: Camenisch-Lysyanskaya. Allows for special zero-knowledge proofs
  - ★ Proving possession of credential without transmitting it
  - ★ Revealing only some attributes (e.g. kind and class)
  - ★ Proving properties about attributes (e.g. birthdate < 9/10/2000)
  - ★ Proving relations between credentials (e.g. passport on same name)
  - ★ Signing a statement (e.g. "I confirm this electronic order...")

# IBM's Idemix

- Credentials: list of attribute-value pairs, signed by an issuer.  
`sign(copenhagen-commune; name="Johanna Gossner", birthdate=14/12/73, kind="Drivers License", class="3", ...)`
- Signatures are special: Camenisch-Lysyanskaya. Allows for special zero-knowledge proofs
  - ★ Proving possession of credential without transmitting it
  - ★ Revealing only some attributes (e.g. kind and class)
  - ★ Proving properties about attributes (e.g. birthdate < 9/10/2000)
  - ★ Proving relations between credentials (e.g. passport on same name)
  - ★ Signing a statement (e.g. "I confirm this electronic order...")
- Privacy: A verifier cannot find out more than what was proved.
  - ★ E.g. different transactions are unlinkable (unless linked by prover).
  - ★ Holds even if the issuer is dishonest and collaborates with a dishonest verifier.

# Idemix: Privacy with Accountability?

How can we ensure that we

- protect the privacy of the innocent
- but not the privacy of the guilty?

# Idemix: Privacy with Accountability?

How can we ensure that we

- protect the privacy of the innocent
  - but not the privacy of the guilty?
- 
- Can be achieved using escrow with verifiable encryptions

# Idemix: Privacy with Accountability?

How can we ensure that we

- protect the privacy of the innocent
  - but not the privacy of the guilty?
- 
- Can be achieved using escrow with **verifiable encryptions**
  - Provers have to encrypt their real names with the public key of an authority.

# Idemix: Privacy with Accountability?

How can we ensure that we

- protect the privacy of the innocent
  - but not the privacy of the guilty?
- 
- Can be achieved using escrow with **verifiable encryptions**
  - Provers have to encrypt their real names with the public key of an authority.
  - Provers have to prove (in zero-knowledge) that they have encrypted the real name (and it is correctly encrypted).

# Idemix: Privacy with Accountability?

How can we ensure that we

- protect the privacy of the innocent
  - but not the privacy of the guilty?
- 
- Can be achieved using escrow with **verifiable encryptions**
  - Provers have to encrypt their real names with the public key of an authority.
  - Provers have to prove (in zero-knowledge) that they have encrypted the real name (and it is correctly encrypted).
  - Then the authority can **revoke** their privacy when needed.

# Idemix: Privacy with Accountability?

How can we ensure that we

- protect the privacy of the innocent
  - but not the privacy of the guilty?
- 
- Can be achieved using escrow with **verifiable encryptions**
  - Provers have to encrypt their real names with the public key of an authority.
  - Provers have to prove (in zero-knowledge) that they have encrypted the real name (and it is correctly encrypted).
  - Then the authority can **revoke** their privacy when needed.
  - This should be tied to a legal system, e.g. the privacy revocation happens only on court order.

## Administering Security



$$\int_a^b \mathcal{E}^{\theta} = \sum_{n=0}^{\infty} \frac{(\Delta x)^n}{n!} f^{(n)}(x) + \Omega \int_a^b \delta e^{i\pi} = \{2.7182818284\}$$

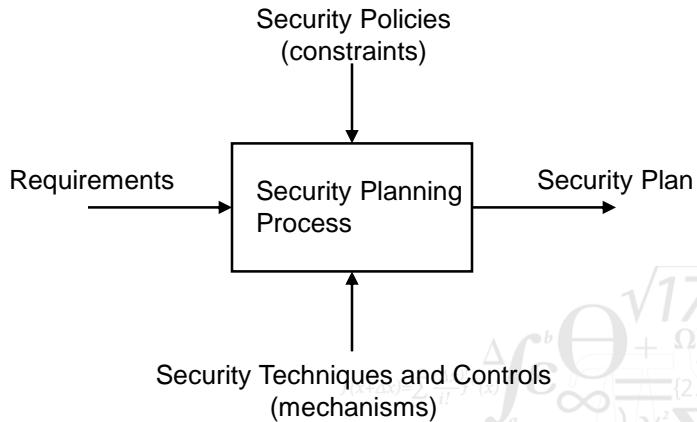
DTU Compute

Department of Applied Mathematics and Computer Science

## Four Areas of Security Administration

- Security planning
  - Advance preparation for when things go wrong
  - Define measurable objectives
  - Establish procedures for system evolution
- Security policy definition and enforcement
  - Define goals and ensure that these goals are continuously met
- Physical security
  - Aspects of the physical environment that impact on security
- Risk analysis
  - Analyse threats
  - Weigh cost and benefits of controls

## Security Planning Process



## Contents of a Security Plan

- **Policy definition:** establish the overall security goals
- **Establish base line:** describe the current state of the system to provide a base line for the work
- **Identify requirements:** identify steps needed to achieve the defined security goals
- **Identify recommended controls:** identify controls to vulnerabilities defined in policy and requirements
- **Establish accountability:** delegate responsibility
- **Define timetable:** define deadlines for phased implementation
- **Show continual vigilance:** define responsibility for maintaining security when system evolves

## Defining the Security Policy

- Security policies must have support from the top management
  - Ultimate arbiter in conflicts between security and business goals
  - Ensures buy-in from all levels in the organisation
    - Requires that CEO walks the talk
  - Allocates funds for the security work (people and equipment)
- Security policies must balance the need for security with the inconvenience to the users
  - Security mechanisms that are not understood will not be used
  - Security mechanisms that are not used will not be effective
  - Security mechanisms that are not effective are a waste of time and money



5 DTU Compute Technical University of Denmark

02239 – Data Security

## Contents of a security Policy

- Security policy should specify:
  - The organizations *goals* for security, e.g., confidentiality, integrity and availability, and their relative importance
  - Responsibility for enforcing security policies
    - Historically IT Department
    - Increasingly responsibility of Finance Department
  - The organizations commitment to security
- Overall policy must include an access control policy:
  - Who should be allowed access to the system?
  - To what systems and resources should access be allowed?
  - What type of access should be allowed for each resource

6 DTU Compute Technical University of Denmark

02239 – Data Security

## Defining the Security Requirements

- The security requirements must address all issues mentioned in the security policy
- Important characteristics:
  - *Correctness*: are requirements understandable and correct?
  - *Consistency*: are there conflicting or ambiguous requirements?
  - *Completeness*: are all possibilities covered?
  - *Realism*: is it possible to achieve the requirements?
  - *Need*: are the requirements unnecessarily restrictive?
  - *Verifiability*: can tests be written to verify that the requirements are met?
  - *Traceability*: can all requirements be traced to the related controls, so that changes in requirements can be easily implemented

## Responsibility for Implementation

- Assigning responsibility must also identify coordinators for implementation of the security requirements
  - Who checks for new vulnerabilities
  - Who implements new controls when new vulnerabilities are discovered
- Different responsibilities may be assigned to different groups of users:
  - PC users may be responsible for their own PC
  - Project leaders may be responsible for the data and computers allocated to the project
  - Managers may be responsible for overseeing their project leaders
  - Database administrators are responsible for the DB system

## Ensuring commitment to Security Plan

- Acceptance by the organisation is required for the implementation of any security plan
  - Security team must ensure backing from management
  - Security team must be sensitive to the needs of the group affected by the security policy
  - Those affected by the security policy must understand the importance of the imposed controls
    - *This includes awareness running programs*



- Environments change, so security policies must be able to evolve
  - This requires periodic review of security policies, including establishment of a new baseline
  - Evaluation and modification of unfortunate policies

## Writing Security Policies

- Security policies are written for several purposes
  - Identify sensitive information assets
  - Clarify security responsibilities
  - Promote awareness for existing employees
  - Guiding new employees
- Security policies have several audiences
  - Users
  - Owners
  - Beneficiaries
  - It is important to balance the interest of all parties

## Content of a Security Policy

- State purpose
  - Promote efficient business operation
  - Facilitate sharing of information throughout the organization
  - Safeguard business and personal information
  - Ensure accurate information is available to support business decisions
  - Ensure a safe and productive place to work
  - Comply with applicable laws and regulations
- Protected Resources
  - Explicitly list assets covered by the policy (*which resources*)
- Nature of Protection
  - Indicate *who* should have access and how access is granted/denied

## Characteristics of a Good Security Policy

- Coverage
  - The policy should be comprehensive - any incident that may occur should be covered by the policy (generality)
- Durability
  - The policy should evolve with the system, but remain largely unchanged (keep it general)
- Realism
  - The policy should not set unobtainable goals or impose unworkable restrictions
- Usefulness
  - The policy should be seen to be useful, otherwise it will be ignored

## Physical Security

- Physical security encompass all aspects of security that involve physical controls: walls, locks, guards, emergency generators, fire inhibitors, ...
- Physical security should consider:
  - Break-ins, theft, espionage, ...
  - Natural disasters
  - Loss of power
  - Loss of communication
  - Human vandals
  - Interception of sensitive information



## Natural disasters; threats to availability

- Floods
  - Natural floods come from rising water (ground water, rivers)
    - Flooding is normally slow and some precautions can be made
  - Sudden floods come from bursting dams, water mains, ...
    - Flooding is difficult to prevent and few precautions can be made
- Fires
  - Fire is often sudden
    - Short time to react
  - Extinguished with water (implies flooding as well)
  - Equipment can be damaged by smoke as well as fire
- Storms, earthquakes, volcanoes
  - Often determined by geographical location

## Loss of power and Communications

- Uninterruptible power supply (UPS)
  - Stores energy (battery or wheel) that can be released if the power is cut - normally only allows a clean shutdown
  - UPS can be complemented by a backup generator
- Surge suppressor
  - Some countries have problems with sudden drops, spikes and surges in electrical power, which may damage electrical equipment - surge suppressors limit variations in power
- Multiple ISPs
  - Prevent breakdown if one ISP fails
- Multiple phone lines?
  - In Europe one company (the national PTT) has historically installed the infrastructure and other operators simply lease access to the wire
    - this is particularly true of the local loop

## Dependability of Data Center

Tier	Requirement
I	<ul style="list-style-type: none"><li>• Single non-redundant distribution path serving the critical loads</li><li>• Non-redundant critical capacity components</li></ul>
II	<ul style="list-style-type: none"><li>• Meets all Tier I requirements, in addition to:</li><li>• Redundant critical capacity components</li><li>• Critical capacity components must be able to be isolated and removed from service while still providing N capacity to the critical loads.</li></ul>
III	<ul style="list-style-type: none"><li>• Meets all Tier II requirements in addition to:</li><li>• Multiple independent distinct distribution paths serving the IT equipment critical loads</li><li>• All IT equipment must be dual-powered provided with two redundant, distinct UPS feeders. Single-corded IT devices must use a Point of Use Transfer Switch to allow the device to receive power from and select between the two UPS feeders.</li><li>• Each and every critical capacity component, distribution path and component of any critical system must be able to be fully compatible with the topology of a site's architecture isolated for planned events (replacement, maintenance, or upgrade) while still providing N capacity to the critical loads.</li><li>• Onsite energy production systems (such as engine generator systems) must not have runtime limitations at the site conditions and design load.</li></ul>
IV	<ul style="list-style-type: none"><li>• Meets all Tier III requirements in addition to:</li><li>• Multiple independent distinct and active distribution paths serving the critical loads</li><li>• Compartmentalization of critical capacity components and distribution paths</li><li>• Critical systems must be able to autonomously provide N capacity to the critical loads after any single fault or failure</li><li>• Continuous Cooling is required for IT and UPS systems.</li></ul>

## Human Vandals

- Unauthorised access and use
  - Hire a guard to prevent outsiders from accessing the site
  - Lock servers in rooms that can only be accessed by authorised personnel
  - Smart cards, RFID or biometrics
- Theft
  - An increasing problem with ubiquitous computing
    - Example: loss of a MI5 laptop with classified information on it
  - Locks or “screamers” can be used to reduce the risk of theft
  - Smart cards, RFID or biometrics

## Interception of sensitive information

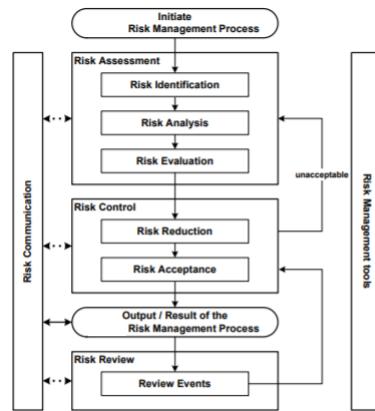
- Be careful when disposing of physical media that contains sensitive information
  - A Danish credit information company (RKI) dumped all their paper files in the garbage where journalists got hold of them
  - Recycled hard disks may contain sensitive information
- Shred all paper
  - Especially sensitive information can be burned after shredding
- Overwriting magnetic media
  - Magnetic media retains some magnetisation even after several overwrite - smash media after overwriting
- Degaussing
  - Create a magnetic field that realign magnetic charges
- Emanation protection: Tempest

## Break



**It's time for my break**

## Basic Elements of Risk Management



## Risk Analysis

- Three elements have to be identified:
  - Loss associated with an event (*risk impact or cost*)
  - Likelihood that the event will occur
  - Degree to which we can change the outcome (*risk control*)
    - *By reducing cost or decreasing likelihood of a particular event*
- Exposure associated with a particular event (aka. risk)
  - Cost of event × likelihood of event
- Three strategies for dealing with risk
  - Avoiding the risk by changing the requirements
  - Transferring the risk, e.g., through insurance
  - Assuming the risk by accepting it; control it with the available resources and accept the loss if the event occurs
- Risk leverage is the relative benefit of reducing risk

$$\frac{(\text{exposure before reduction}) - (\text{exposure after reduction})}{(\text{cost of risk reduction})}$$

## Difficulty of Measuring Risk

- Estimating likelihood of threats
  - Precise models must include everything
    - *It must effectively describe the whole world*
  - How relevant is past data to the calculation of future probabilities?
    - *The nature of future attacks is unpredictable*
    - *The actions of future attackers are unpredictable*
  - Subjective probabilities look most promising
- Businesses want to measure “costs” in money, but
  - Many assets are difficult to measure in this way
    - *Value of data and in-house software - no market value*
    - *Value of goodwill and customer confidence*
- Measurement of benefit from security measures
  - Problems with the difference of two approximate quantities
    - *How does an extra security measure affect a ~10<sup>-5</sup> probability of an attack?*

## Establish Risk Levels

- Precise monetary values give a false sense of precision
- Better to use levels:
  - High, Medium, Low or Essential-high-medium-low-not important
    - High: major impact on the organisation
    - Medium: noticeable impact ("material" in auditing terms)
    - Low: can be absorbed without difficulty
  - Numerical rating: rating on a scale of 1 – 10
- 3x3 matrix

– Commonly used in industry  
 – Difficult to decide priority  
 • Is 6 > 6 ?

Consequence = High,  
 Probability = Med  
 Consequence = Med,  
 Probability = High

	Probability		
High	3	6	9
Med	2	4	6
Low	1	2	3
	Low	Med	High
			Consequence

02239 – Data Security

23 DTU Compute Technical University of Denmark

## Basic Steps of Risk Analysis

- Identify assets
- Determine possible vulnerabilities
- Estimate likelihood of exploitation
- Compute expected annual loss
- Survey applicable controls and their costs
- Project annual savings of control



02239 – Data Security

24 DTU Compute Technical University of Denmark

## Identify Assets

- **Hardware:** computers, peripheral equipment, communication infrastructure
- **Software:** source code, binary programs, operating systems
- **Data:** temporary data, stored data, printed data, backup media
- **People:** skills needed to run systems or programs
- **Documentation:** hardware, software, administrative procedures
- **Supplies:** paper, forms, toner cartridges, magnetic media (including CD-ROMs)

## Determine Vulnerabilities

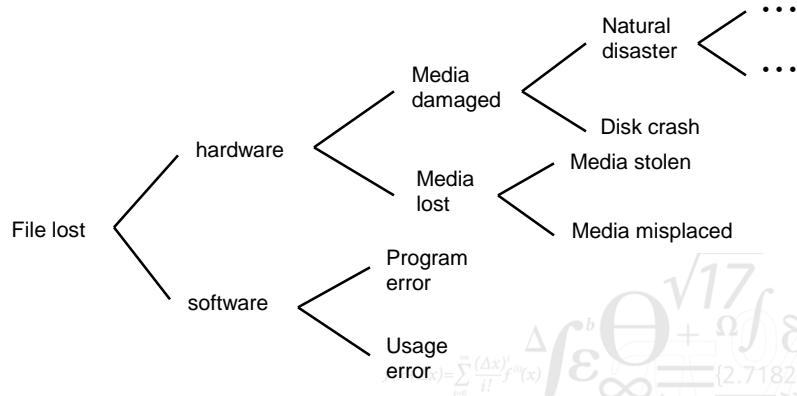
Asset	Confidentiality	Integrity	Availability
Hardware		tampered with	stolen, destroyed
Software	copied, pirated	Trojan horse, backdoor	deleted, license expire
Data	disclosure, inference	damaged	deleted, misplaced
People			quit, retire, vacation
Documentation	copied, stolen	tampered with	lost, stolen, destroyed
Supplies			lost, stolen, damaged

- Fill in the matrix

- Effects of unintentional errors: typos, insecure disposal of output
- Effects of malicious insiders: disgruntled employees, bribery, curious browsers (what is the salary of the CEO?)
- Effects of outsiders: network access, dial-in access, hackers, uninvited visitors, dumpster diving detectives
- Effects of natural disasters

## Fault Tree Analysis

- Common method to determine vulnerabilities

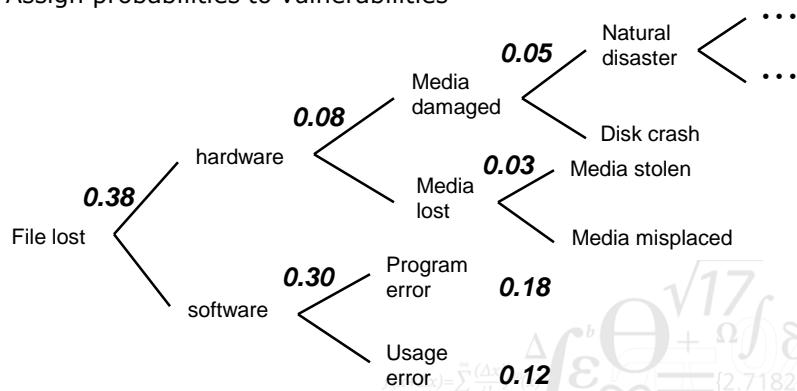


27 DTU Compute Technical University of Denmark

02239 – Data Security

## Estimate Likelihood of Exploitation

- Assign probabilities to vulnerabilities



28 DTU Compute Technical University of Denmark

02239 – Data Security

## Compute Expected Annual Loss

- Not all risks are equally severe
  - Safety critical: human lives are at stake
  - Mission critical: survival of the company is at stake
  - Costly: major impact on the earnings of the company
  - Negligible: costs are small compared with operational costs
- Knowing probability and cost of the event allows computation of the expected annual loss
- Example: Credit card companies accept a certain amount of fraud



29 DTU Compute Technical University of Denmark

02239 – Data Security

## Measure All Costs

- Hidden costs are easy to forget:
  - Unavailability of the system, restore data from backups, reinstallation of software
- Consider the following questions:
  - What legal obligations exist for confidentiality and integrity of data
  - What contractual obligations may be applicable (*fines?*)
  - Could release of data harm individuals or organizations
    - *Will they litigate?*
  - Could event cause missed business in the future
  - What are the psychological effects of event
    - *Embarrassment, loss of credibility, loss of business, ...*
  - What is the value of access to data (worth a backup site)
  - What other problems could arise from loss of data
    - *Can data be restored*

30 DTU Compute Technical University of Denmark

02239 – Data Security

## Management Frameworks and Governance Structures

- A number of standards and best practices are commonly used
- ISO 27000 series
  - Family of standards on Information Security Management Systems
  - *Available as "Dansk Standard" through the DTU Library*
- NIST Special Publication 800 series
  - NIST SP 800-39
  - NIST Cybersecurity Framework
  - *All publications available online at NIST*
- CIS Critical Security Controls
  - CIS defines a number of controls to improve security
  - Latest version is CIS v8 (earlier version is known as CIS20)
  - *Available at: <https://www.cisecurity.org/controls/v8/>*



# 02239 Data Security Software Security I

Sebastian Mödersheim

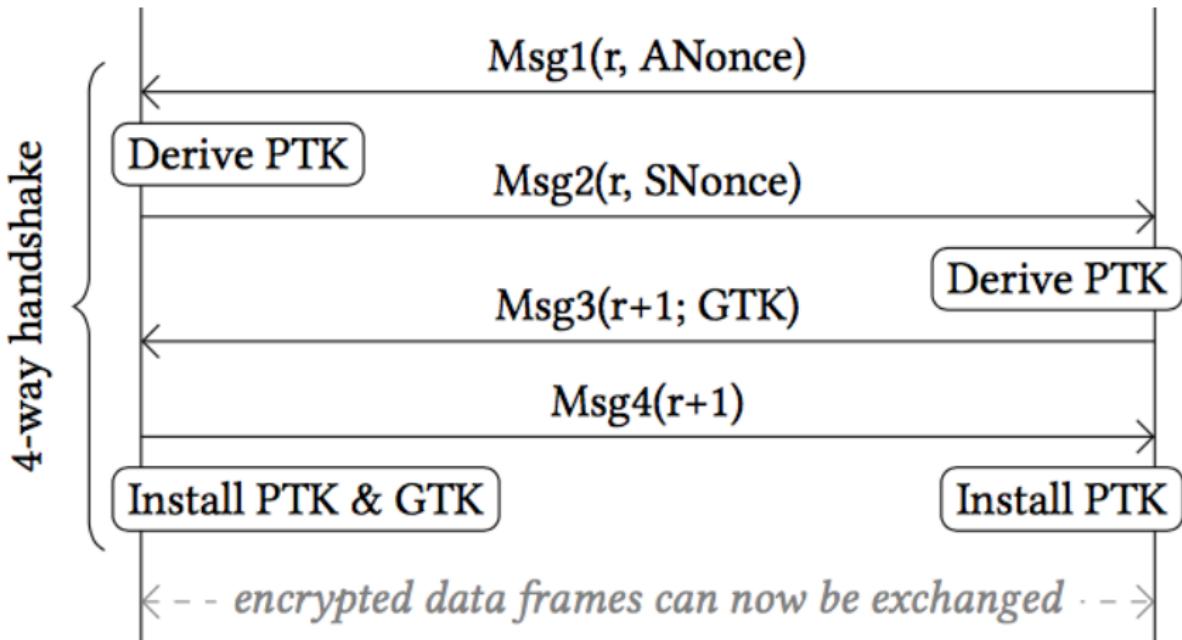
November 17, 2021

# An Attack on WPA2

- M. Vanhoef and F. Piessens. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. (CCS 2017)
- On the border of protocol and software security:
  - ★ Cryptographic primitives have been analyzed
  - ★ Protocol has been analyzed
  - ★ Neither of them is in itself broken...

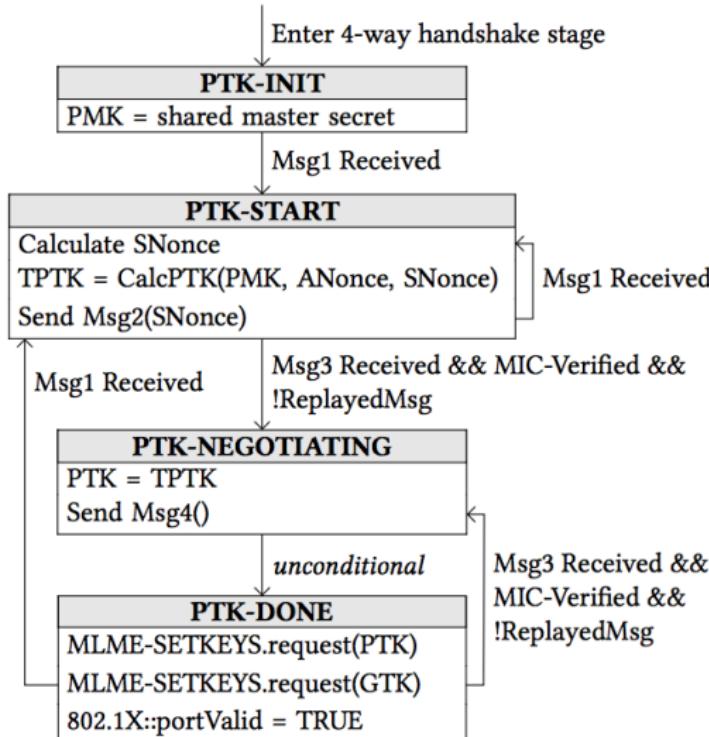
# WPA2: 4-Way Handshake

Source: Vanhoef and Piessens 2017

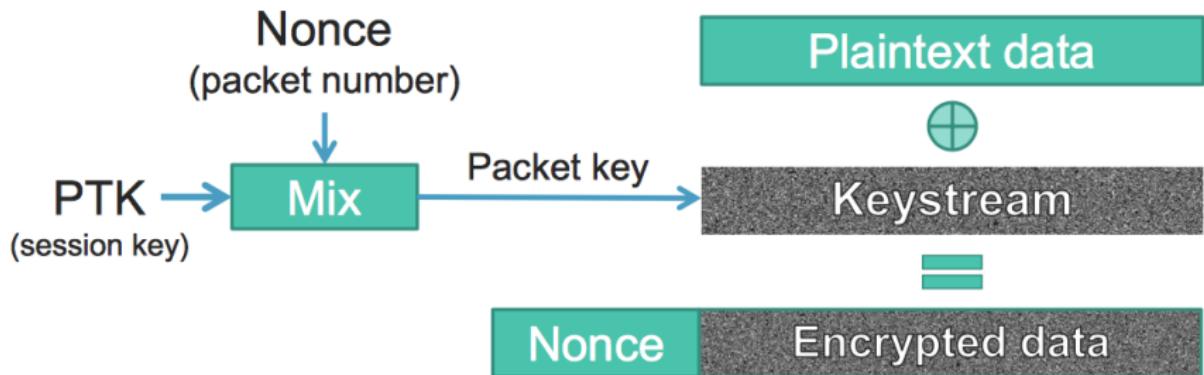


# WPA2: A StateMachine

Source: Vanhoef and Piessens 2017



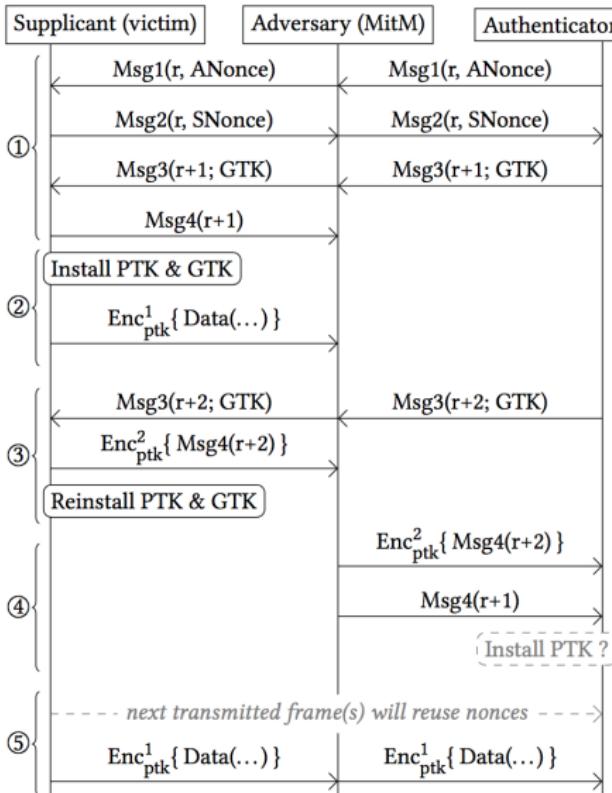
## Frame encryption (simplified)



→ Nonce reuse implies keystream reuse (in all WPA2 ciphers)

# WPA2: Simplemost Attack

Source: Vanhoef and Piessens 2017



# Software Security Problems

## Honest Mistakes

Programs may contain vulnerabilities that an attacker can exploit.

Mitigations:

- Developers: Prudent engineering practices, standards, libraries
- Analytic: Testing and verification
- Constructive: Secure by design and compilers

## Trojans and the like

Executing code from a potentially malicious source:

- Installing new software, apps, plugins
- Web browser executing Javascript code
- Cloud hosting client applications

Typically offering a nice function for free, and including a hidden malicious function.

# Buffer Overflow

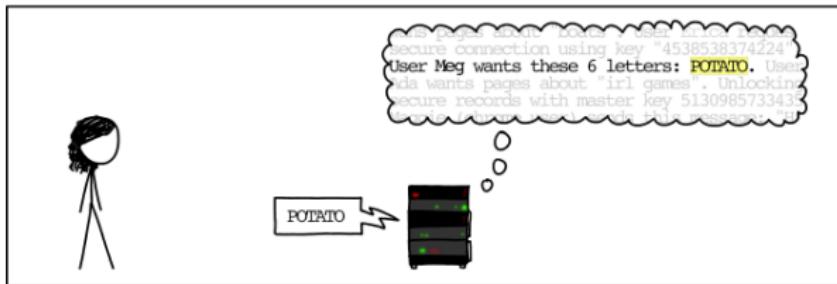
## What is a buffer overflow?

- Read/write beyond memory allocated to buffer
  - ★ Unchecked input
- May overwrite return addresses (e.g., stack overflow)
- May insert jumps to library code (e.g., heap overflow)

## Beware

- “Basically a solved problem” (common belief in the 1990s)
- Some buffer overflows are easy to find
- Some are very hard to find/avoid

## HOW THE HEARTBLEED BUG WORKS:



(Source: [xkcd.com](http://xkcd.com))

Sebastian Mödersheim

November 17, 2021

9 of 35

HMM...



BIRD



User Olivia likes random words pages about cars. Gees in car why". Note: Files for IP 375.381. 83.17 are in /tmp/files-3843. User Meg wants these 4 letters: **BIRD**. There are currently 34 connections open. User Brendan uploaded the file 15 minutes ago: 234b-062c-2acb04f6201-12-f69.

SERVER, ARE YOU STILL THERE?  
IF SO, REPLY "HAT" (500 LETTERS).



connection. Jake requested pictures of user User Meg wants these 500 letters: **HAT**. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about snakes but not too long". User Karen wants to change account password to "CofBa2021".



BIRD, Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about snakes but not too long". User Karen wants to change account password to CofBa2021".

(Source: [xkcd.com](https://xkcd.com/128))

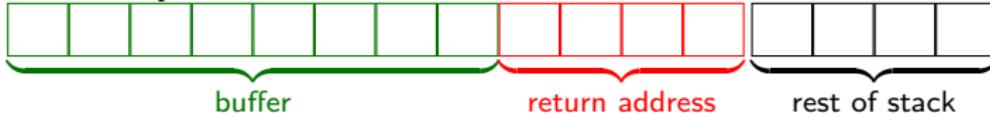
## Example: Stack Overflow

```
#include <string.h>
int main (int argc, char **argv)
{
    char buffer[8];
    strcpy(buffer, argv[1]);
}
```

# Example: Stack Overflow

```
#include <string.h>
int main (int argc, char **argv)
{
    char buffer[8];
    strcpy(buffer, argv[1]);
}
```

Stack Layout:



# Example: Stack Overflow

```
#include <string.h>
int main (int argc, char **argv)
{
    char buffer[8];
    strcpy(buffer, argv[1]);
}
```

Stack Layout:



Attack: give an argument of 12 chars XXXXXXXXXXXXAAAA

# Example: Stack Overflow

```
#include <string.h>
int main (int argc, char **argv)
{
    char buffer[8];
    strcpy(buffer, argv[1]);
}
```

Stack Layout:



Attack: give an argument of 12 chars XXXXXXXXAAAA

If you can guess address AAAA of buffer, you can execute code XXXXXXXX under the privilege of this application.

## Example: Stack Overflow

```
#include <string.h>
int main (int argc, char **argv)
{
    char buffer[8];
    strcpy(buffer, argv[1]);
}
```



Gives the attacker a shell with the privilege of this application.  
This type of buffer overflow is called **code injection**.

# Example: Stack Overflow

```
#include <string.h>
int main (int argc, char **argv)
{
    char buffer[8];
    strcpy(buffer, argv[1]);
}
```



The code will continue at **address** (intruder's choice).  
This type of buffer overflow is called **code reuse**.

## But that is difficult to guess?

- The attacker actually needs to guess addresses.
- Actually, mounting a buffer overflow often requires a lot of trial and error for an attacker.
- But suppose that
  - ★ the attacker knows the programs source and binary code,
  - ★ can try the attack any number of times,
  - ★ he can write a script that automatically tries the attack with different input values...
- Techniques like stack obfuscation (reordering, encryptions of values, ...) and canaries (special value before the return address) are no ultimate solution, but make the life of the attacker harder.

# Unsafe Functions

- Strings in C are `char` arrays of **arbitrary** length: they are terminated by a 0-byte.
- Several functions in the C library work on strings **without stopping** before reading a 0-byte.
- When copying into a buffer, then a too long string means overwriting memory areas following the buffer.
  - ★ If you are lucky, the program just crashes.
  - ★ If you are unlucky, the attacker succeeds without anybody noticing.
- Avoid `strcpy`, `gets`, `printf`, ... on unknown inputs
- Alternatives like `strncpy` and `fgets` are not magically making it safe. Always mind the size of buffers
  - ★ including space for the trailing 0 byte!!

# Overflows

```
char *array;  
int size;  
...  
if (size>1024) size=1024;  
array = malloc(size);  
read(file,array,size);
```

# Overflows

```
char *array;  
int size;  
...  
if (size>1024) size=1024;  
array = malloc(size);  
read(file,array,size);
```

What if `size<0`?

# Overflows

```
char *array;  
int size;  
...  
if (size<0) size=0;  
if (size>1024) size=1024;  
array = malloc(size);  
read(file,array,size)
```

## Why not check this automatically?

- Many languages like Java, JavaScript, ... do check bounds of arrays, overflows and null-pointers.

# Why not check this automatically?

- Many languages like Java, JavaScript, ... do check bounds of arrays, overflows and null-pointers.
- “C/C++ does not do this for efficiency.”

# Why not check this automatically?

- Many languages like Java, JavaScript, ... do check bounds of arrays, overflows and null-pointers.
- “C/C++ does not do this for efficiency.”
- Idea of static analysis: a compiler that tries to prove **statically** that a piece of code will never get out of bounds, and insert the check only when it fails to prove that.
  - ★ similar to type checking.

# Why not check this automatically?

- Many languages like Java, JavaScript, ... do check bounds of arrays, overflows and null-pointers.
- “C/C++ does not do this for efficiency.”
- Idea of static analysis: a compiler that tries to prove **statically** that a piece of code will never get out of bounds, and insert the check only when it fails to prove that.
  - ★ similar to type checking.
- The concept of C/C++ does not allow for that in general!
  - ★ Arrays are just pointers without an implicit length information.  
Allows freedom, but invites for mistakes.
  - ★ Using classes with checks from C++ like String, Vector, Array, can prevent much damage.

# How to Prevent Bad Programming

## Some Partial Solutions

- Teach programmers to be more careful
- Security mechanisms in languages, libraries, platforms
  - ★ Java bytecode verification, type systems, sandboxing
- More and better testing

# Formal Methods

## Techniques

- Model Checking
- Static Analysis
- Information Flow Analysis
- Certifying Compilers
- Monitors

## Static Analysis

- Roots: optimizing compilers
- Static computation of dynamic behavior
  - ★ Approximation used to sidestep halting-problem

# Splint: Light-Weight Annotation-Based Static Analysis

## Secure Programming LINT

- Based on lint: well-known program checker
- Let the programmer annotate program
- Check that the program is consistent with annotations
- Can find many common errors

## Other Tools

- VCC/Z3 (Microsoft)
- BLAST
- ...

# Splint: Buffer Overflow

## Example

```
#include <string.h>
int main (int argc, char **argv)
{
    char buffer[8];
    strcpy(buffer, argv[1]);
}

$ splint +bounds buffer01.c
...
buffer01:9: Possible out-of-bounds store: strcpy(str,tmp)...
```

# Splint: Buffer Overflow

## Example

```
#include <string.h>
int main (int argc, char **argv)
{
    char buffer[8];
    strcpy(buffer, argv[1], 7);
    buffer[7] = '\\0';
}
```

```
$ splint +bounds buffer01.c
```

```
...
```

```
Finished checking --- no warnings
```

# Splint

## Problems

- May need a lot of annotation/time/experience
- False positives: warnings often do not represent real problems
- False negatives: real problems may be missed
  - ★ Memory modeling
  - ★ Sharing
  - ★ Control Flow

# Malicious Code

Some types of malicious code (incomplete, not disjoint):

**Virus/Worm** replicates, trying to infect more systems

**Trojan horse** hides malicious code behind a desired, primary effect

**Root kit** manipulates system routines to prevent detection

# Malicious Code

Some types of malicious code (incomplete, not disjoint):

**Virus/Worm** replicates, trying to infect more systems

**Trojan horse** hides malicious code behind a desired, primary effect

**Root kit** manipulates system routines to prevent detection

- RootkitRevealer [Russinovich 2006f] compares output of system routines and physical data on hard disk
- Revealed the famous Sony root kit

# Getting the malicious code executed

Malicious software can only harm when executed.

- Email: “Please install the attached software update for urgent security problems.”
- “Download this cool free protocol analysis tool today!”
- Apps for mobile phones
- Most websites have executable code like JavaScript.
  - ★ You can turn off JavaScript, but then many websites become unusable.
- Many documents (MS Word, PDF, ...) may contain scripts/executable code.
- Autoplay (do you know what your system executes automatically?)
- Exploits: intruder inserts code into system e.g. exploiting a buffer overflow.

## Making Yourself Comfortable

Once malicious code has gained control of a computer, how can it protect against detection and removal?

- Inject copy of the code into normal applications or the boot sector of hard disk
  - ★ Ensure that it gets executed again after a restart
- Manipulate system routines so that virus scanners or user will not detect it.
- Obfuscation by encrypting/changing part of the code (avoiding a “fingerprint”).
- Become part of a botnet!

# Some Famous Examples

## The Brain Virus (1986)

- Floppy disc bootsector virus

A screenshot of the ht 2.0.16 hex editor. The title bar says "ht 2.0.16". The menu bar includes File, Edit, Windows, Help, Local-Hex, and a timestamp "14:17 07.01.2010". The main window displays a hex dump of the "brain\_sector" file. The code starts with a series of zeros and then contains ASCII text. The text includes "Welcome to the Dungeon", copyright information for 1986 Basit & Amjad, and details about the virus's origin in Lahore, Pakistan. It also contains a warning message and a call for vaccination.

```
[...]= ...ples\brain_sector\8de894dc6f27e10664fc7db1137ef3ef0af62d5.bin ==2
00000000 fa e9 4a 01 34 12 01 08-06 00 01 00 00 00 00 20 0J04t@ 0
00000010 20 20 20 20 20 57 65-6c 63 6f 6d 65 20 74 6f
00000020 20 74 68 65 20 44 75 6e-67 65 6f 6e 20 20 20 20
00000030 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20 20
00000040 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20 20
00000050 20 28 63 29 20 31 39 38-36 20 42 61 73 69 74 20
00000060 26 20 41 6d 6a 61 64 20-28 70 76 74 29 20 4c 74
00000070 64 2e 20 20 20 20 20 20-20 20 20 20 20 20 20 20
00000080 20 42 52 41 49 4e 20 43-4f 4d 50 55 54 45 52 20
00000090 53 45 52 56 49 43 45 53-2e 2e 37 33 30 20 4e 49
000000a0 5a 41 4d 20 42 4c 4f 43-4b 20 41 4c 4c 41 4d 41
000000b0 20 49 51 42 41 4c 20 54-4f 57 4e 20 20 20 20 20
000000c0 20 20 20 20 20 20 20 20-20 20 20 4c 41 48 4f 52
000000d0 45 2d 50 41 4b 49 53 54-41 4e 2e 2e 50 48 4f 4e
000000e0 45 20 3a 34 33 30 37 39-31 2c 34 34 33 32 34 38
000000f0 2c 32 38 30 35 33 30 2e-20 20 20 20 20 20 20 20
00000100 20 20 42 65 77 61 72 65-20 6f 66 20 74 68 69 73
00000110 20 56 49 52 55 53 2e 2e-2e 2e 2e 43 6f 6e 74 61
00000120 63 74 20 75 73 20 66 6f-72 20 76 61 63 63 69 6e
00000130 61 74 69 6f 6e 2e 2e 2e-2e 2e 2e 2e 2e 2e 2e 2e
00000140 2e 2e 2e 2e 20 24 23 40-25 24 40 21 21 20 8c c8
view e0h/224
1help 2save 3open 4edit 5goto 6mode 7search 8resize 9viewin.0quit
```

# Some Famous Examples

## The Internet Worm (1988)

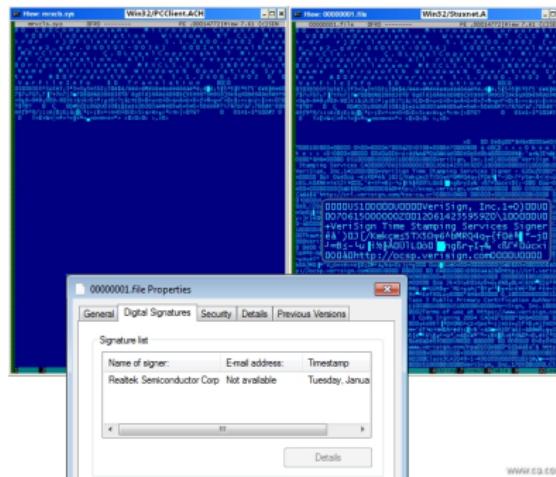
- Meant to just determine how far it can spread;
- By **honest mistake** (!) became a denial-of-service attack.



# Some Famous Examples

## Stuxnet (2010)

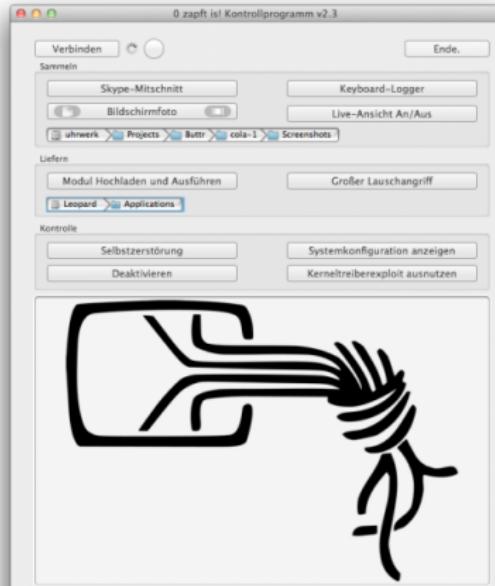
- Targeted at particular industrial software
- 4 zero-day exploits
- signed using stolen certificates
- Speculations that it was written by a secret service



# Some Famous Examples

“Staatstrojaner” (2008/2011)

- Chaos Computer Club claims that it is malware used by law enforcement in several German states for Internet-phone surveillance of suspects (after warrant from a judge).



# Some Famous Examples

## WannaCry (2017)

- Ransomware: encrypt all hard disk and require ransom (in bitcoin) for decryption.
- Used [EternalBlue](#), a Windows exploit first discovered by NSA.
  - ★ NSA had not notified Microsoft about the vulnerability.
  - ★ Apparently this exploit was stolen and leaked.
- US government blamed North Korea.



# The Lazy Mobile Intruder

## How to verify sandboxing?

- Security of a platform that executes potentially malicious code
- Goal: the code cannot break out of its sandbox
- Research Paper:  
Sebastian Mödersheim, Flemming Nielson, Hanne Riis Nielson.  
*Lazy Mobile Intruders*. POST 2013.

# The Lazy Mobile Intruder

## The Problem

Given:

- Platform  $C[\cdot]$  that executes potentially malicious code

# The Lazy Mobile Intruder

## The Problem

Given:

- Platform  $C[\cdot]$  that executes potentially malicious code
  - ★ E.g. web browser, mobile phone, cloud provider

# The Lazy Mobile Intruder

## The Problem

Given:

- Platform  $C[\cdot]$  that executes potentially malicious code
  - ★ E.g. web browser, mobile phone, cloud provider
- Initial intruder knowledge  $K_0$
- Attack predicate  $\text{attack}(S)$

# The Lazy Mobile Intruder

## The Problem

Given:

- Platform  $C[\cdot]$  that executes potentially malicious code
  - ★ E.g. web browser, mobile phone, cloud provider
- Initial intruder knowledge  $K_0$
- Attack predicate  $\text{attack}(S)$

Question: does a program  $P$  exist such that

# The Lazy Mobile Intruder

## The Problem

Given:

- Platform  $C[\cdot]$  that executes potentially malicious code
  - ★ E.g. web browser, mobile phone, cloud provider
- Initial intruder knowledge  $K_0$
- Attack predicate  $\text{attack}(S)$

Question: does a program  $P$  exist such that

- $K_0 \vdash P$  the intruder can write  $P$

# The Lazy Mobile Intruder

## The Problem

Given:

- Platform  $C[\cdot]$  that executes potentially malicious code
  - ★ E.g. web browser, mobile phone, cloud provider
- Initial intruder knowledge  $K_0$
- Attack predicate  $\text{attack}(S)$

Question: does a program  $P$  exist such that

- $K_0 \vdash P$  the intruder can write  $P$
- $C[P] \rightarrow^* S$  when  $C[\cdot]$  executes  $P$  we reach a state  $S$ ...

# The Lazy Mobile Intruder

## The Problem

Given:

- Platform  $C[\cdot]$  that executes potentially malicious code
  - ★ E.g. web browser, mobile phone, cloud provider
- Initial intruder knowledge  $K_0$
- Attack predicate  $attack(S)$

Question: does a program  $P$  exist such that

- $K_0 \vdash P$  the intruder can write  $P$
- $C[P] \rightarrow^* S$  when  $C[\cdot]$  executes  $P$  we reach a state  $S$ ...
- $attack(S)$  ... that is an attack

?

# The Lazy Mobile Intruder

## The Problem

Given:

- Platform  $C[\cdot]$  that executes potentially malicious code
  - ★ E.g. web browser, mobile phone, cloud provider
- Initial intruder knowledge  $K_0$
- Attack predicate  $\text{attack}(S)$

Question: does a program  $P$  exist such that

- $K_0 \vdash P$  the intruder can write  $P$
- $C[P] \rightarrow^* S$  when  $C[\cdot]$  executes  $P$  we reach a state  $S$ ...
- $\text{attack}(S)$  ... that is an attack

?

Obviously we cannot search the space of all programs  $\{P \mid K_0 \vdash P\}$ .

# The Lazy Mobile Intruder

## The Problem

Given:

- Platform  $C[\cdot]$  that executes potentially malicious code
  - ★ E.g. web browser, mobile phone, cloud provider
- Initial intruder knowledge  $K_0$
- Attack predicate  $\text{attack}(S)$

Question: does a program  $P$  exist such that

- $K_0 \vdash P$  the intruder can write  $P$
- $C[P] \rightarrow^* S$  when  $C[\cdot]$  executes  $P$  we reach a state  $S\dots$
- $\text{attack}(S) \dots$  that is an attack

?

Obviously we cannot search the space of all programs  $\{P \mid K_0 \vdash P\}$ .

Uses the **Lazy Intruder** technique from OFMC

# Rules for Symbolic Intruder Processes

Example: Communication rule

$$\underbrace{(x).P}_{p1} \parallel \underbrace{\langle M \rangle}_{p2} \rightarrow P[x \mapsto M]$$

# Rules for Symbolic Intruder Processes

Example: Communication rule

$$\underbrace{(x).P}_{p1} \parallel \underbrace{\langle M \rangle}_{p2} \rightarrow P[x \mapsto M]$$

intruder receiving

$$[K] \parallel \langle M \rangle \Rightarrow [K \cup \{M\}]$$

# Rules for Symbolic Intruder Processes

Example: Communication rule

$$\underbrace{(x).P}_{p1} \parallel \underbrace{\langle M \rangle}_{p2} \rightarrow P[x \mapsto M]$$

intruder receiving

$$\boxed{K} \parallel \langle M \rangle \Rightarrow \boxed{K \cup \{M\}}$$

intruder sending

$$(x).P \parallel \boxed{K} \Rightarrow P \parallel \boxed{K} \text{ and } \psi = K \vdash x$$

# Rules for Symbolic Intruder Processes

Example: Communication rule

$$\underbrace{(x).P \parallel \langle M \rangle}_{p1} \rightarrow P[x \mapsto M] \quad \underbrace{\qquad\qquad\qquad}_{p2}$$

intruder receiving

$$\boxed{K} \parallel \langle M \rangle \Rightarrow \boxed{K \cup \{M\}}$$

intruder sending

$$(x).P \parallel \boxed{K} \Rightarrow P \parallel \boxed{K} \text{ and } \psi = K \vdash x$$

intruder both sending and receiving

$$\boxed{K} \parallel \boxed{K'} \Rightarrow \boxed{K \cup K'}$$

# 02239 Data Security Software Security II

Sebastian Mödersheim

November 24, 2021

# This Week

- Most common software vulnerabilities
- Information flow
- Side channel attacks

# Top 25 Errors

MITRE/SANS Common Weakness Enumeration

<http://cwe.mitre.org/top25>

Collection of the most common security vulnerabilities of software:

- Based on reported security problems.
- Particularly dangerous flaws because **low-hanging fruit** for attackers.
- Software developers shall be made aware of the problems.
- Provide “standardized” ways to avoid the problems.
- Updated regularly to reflect increasing/decreasing significance of problems.
- Also contains specialized recommendations depending on programming language used, preferences, and educational emphasis.
- **Should one publish such a list?**

# SQL Injection

## Webpage

login:

→ \$user

password:

→ \$pass

## PHP script generating an SQL query

```
$SQLquery = "SELECT * FROM customers  
WHERE username = '$user' AND password='$pass';"
```

Given a correct username-password pair, this retrieves all information about this user from the database customers. This can be put into a nice webpage by a script again.

# SQL Injection

## Webpage

login:  → \$user  
password:  → \$pass

## Generated SQL query

```
$SQLquery = "SELECT * FROM customers  
WHERE username = '' OR 1; ---' AND password=''"
```

which returns the entire customer database.

# SQL Injection

## Webpage

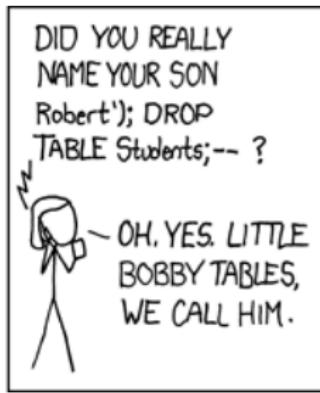
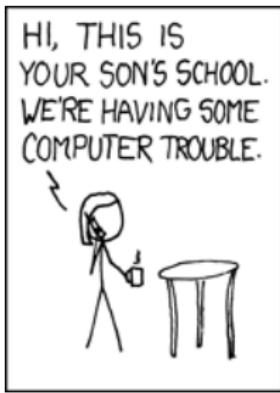
login:  → \$user  
password:  → \$pass

## Generated SQL query

```
$SQLquery = "SELECT * FROM customers  
WHERE username = ''; DELETE * FROM customers; ---'..."
```

which deletes the entire customer database.

# SQL Injection: Exploits of a Mom



Source: [xkcd.com/327](http://xkcd.com/327)

# SQL Injection: Mitigation

- Re-iterate: all input to your programs could be malicious
- Input needs to be sanitized.
- In this case: ensure there is no ' in the input, but is this everything?
- Enforce separation between data and code; strong typing.
- Use different solutions for access-control than SQL databases.
- Try to isolate: separate databases, introduce special account with low privileges

# XSS – Cross Site-Scripting

- Web applications generate webpages, based on input from users.
- That input may contain malicious code e.g. in JavaScript.
- Without proper checking the input, the generated webpage contains the malicious code.
- Code is thus executed under the domain of the webpage.

# Cross Site-Scripting: Example

## Intruder's webpage or email

You can find everything you need [here](#).

# Cross Site-Scripting: Example

## Intruder's webpage or email

You can find everything you need [here](#).

where [here](#) links to

www.dtu.dk/someNonExistingPage<script type="text/javascript">...

# Cross Site-Scripting: Example

## Intruder's webpage or email

You can find everything you need [here](#).

where [here](#) links to

[www.dtu.dk/someNonExistingPage<script type="text/javascript">...](http://www.dtu.dk/someNonExistingPage<script type='text/javascript'>...)

## [www.dtu.dk](#) might reply

404 Page not found: The page

[someNonExistingPage<script type="text/javascript">...](http://someNonExistingPage<script type='text/javascript'>...)

was not found on this server.

# Cross Site-Scripting: Example

## Intruder's webpage or email

You can find everything you need [here](#).

where [here](#) links to

[www.dtu.dk/someNonExistingPage<script type="text/javascript">...](http://www.dtu.dk/someNonExistingPage<script type='text/javascript'>...)

## [www.dtu.dk](#) might reply

404 Page not found: The page

[someNonExistingPage<script type="text/javascript">...](http://someNonExistingPage<script type='text/javascript'>...)

was not found on this server.

And now there is a piece of **malicious code** that comes from DTU!

# Cross Site-Scripting: Same Origin Policy

## Same Origin Policy

- JavaScript on one webpage may access the data and methods of other webpages **on the same domain**.
- Most importantly cookies: often contain session and authentication data.
- It is essential that cookies can be accessed only by the domain to which they belong.

It thus makes a difference whether the malicious code is on the intruder's webpage or on DTU's webpage.

# Cross Site-Scripting: Same Origin Policy

## Same Origin Policy

- JavaScript on one webpage may access the data and methods of other webpages **on the same domain**.
- Most importantly cookies: often contain session and authentication data.
- It is essential that cookies can be accessed only by the domain to which they belong.

It thus makes a difference whether the malicious code is on the intruder's webpage or on DTU's webpage.

But sorry hackers: DTU pages are already secured against this particular attack.

# Cross Site-Scripting: Mitigation

- General rule: **all input to your programs could be malicious**
- Specify: what is acceptable input for your case?
- Refuse/filter all input that does not follow this specification.
- Not trivial: anticipate everything that may could cause trouble.
- Generally, producing HTML/webpages: filter user input for everything that is not pure text.
  - ★ ... but what is pure text? (character encoding!)
  - ★ understand context in which data will be used, what encodings are present
- Understand all areas where untrusted input can enter your software
- Using special libraries and also firewalls can help

# Cross-Site Request Forgery

- Any social media websites allow users to submit/post content in some form.
- Usually they will allow posting images by giving a URL. How's about:

```

```

- Every user who reads the post will send the request `execute?action=...` to `mywebemail.dk`.
- Suppose `mywebemail.dk` has authentication based on cookies.
- Suppose some reader is also legal user of `mywebmail.dk`.
- Suppose this user is currently logged into `mywebmail.dk`.
- ...

# CSRF + XSS

Samy worm combined two of the vulnerabilities:

- Samy Kamkar performed a XSS attack by introducing some Javascript into his Myspace page.
- As a payload of the XSS, he had a CSRF attack. The victim would involuntarily
  - ★ call the **add-as-friend** function of Myspace for Samy
  - ★ insert **Samy is my hero** into the victims profile
  - ★ along with a copy of the worm (the Javascript code)
- Over a million infections within 24h.

# CSRF + XSS

Samy worm combined two of the vulnerabilities:

- Samy Kamkar performed a XSS attack by introducing some Javascript into his Myspace page.
- As a payload of the XSS, he had a CSRF attack. The victim would involuntarily
  - ★ call the **add-as-friend** function of Myspace for Samy
  - ★ insert **Samy is my hero** into the victims profile
  - ★ along with a copy of the worm (the Javascript code)
- Over a million infections within 24h.
- Stupid joke (and he got probation), but shows the potential.

## CSRF: mitigation

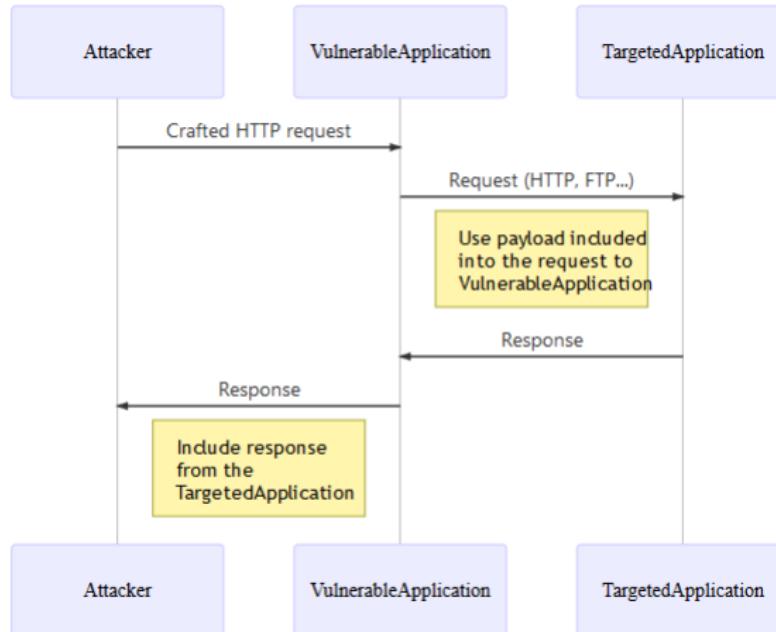
- First ensure that XSS is prevented, because it can bypass CSRF protections.
- Introduce challenge-response on requests using a server-generated nonce.
- Asking back for dangerous operations
- Check http referrer tag.

# SSRF: Server Side Request Forgery

## Scenario

- A web application in a user's browser sends requests to a server-side API
  - ★ e.g. apiCall=https://kurser.dtu.dk/course/02244 to get a course description
- The request causes the server API to access a specified resource (e.g. said url) under permissions of web server
- An attacker observes this behavior and formulates malicious requests (imitating the web application)
  - ★ e.g. apiCall=https://karakterindberetning.dtu.dk/...
-

# SSRF: Server Side Request Forgery



Source: <https://owasp.org>

# OWASP Top Ten 2021



- 
- 1 Broken Access Control
  - 2 Cryptographic Failures
  - 3 Injection
  - 4 Insecure Design
  - 5 Security Misconfiguration
  - 6 Vulnerable and Outdated Components
  - 7 Identification and Authentication Failures
  - 8 Software and Data Integrity Failures
  - 9 Security Logging and Monitoring Failures
  - 10 Server-Side Request Forgery
- 

<https://owasp.org/www-project-top-ten/>

# Monster Mitigations

Effective strategies and principles to address a large number of errors:

- Establish and maintain control over all of your inputs.
- Establish and maintain control over all of your outputs.
- Lock down your environment.
- Assume that external components can be subverted, and your code can be read by anyone.
- Use industry-accepted security features instead of inventing your own.
- Use libraries and frameworks that make it easier to avoid introducing weaknesses.
- Integrate security into the entire software development lifecycle.
- Use a broad mix of methods to comprehensively find and prevent weaknesses.
- Allow locked-down clients to interact with your software.

# Lessons Learned?

- What can we learn from this?
  - ★ Security is an extremely subtle and complicated topic
  - ★ Actually the same mistakes and attacks appear again and again with slight variations
  - ★ The fact that even basic security properties can be broken with relatively simple attacks is scary.
- Good engineering practices:
  - ★ The Top25 list has good suggestions for avoiding top problems.
  - ★ Better education and training can help
  - ★ Development processes that better support security

## Scientific Point of View

Engineering practices to avoid some mistakes are helpful but scientifically unsatisfactory!

- Think of virus scanners that can just check for well-known patterns.
- Attackers become aware of how defenses work and find new holes.
- Our goal is thus to develop:
  - ★ More systematic/general solutions
    - ▶ Systems that by construction prevent entire class of attacks
  - ★ Mathematical proofs that systems/constructions are correct
    - ▶ ... against **any** attack, not just the known ones
  - ★ This requires mathematical models and security definitions that can capture all relevant aspects of reality

## Information Flow: Example

```
classified int b=3;  
int y=12;  
int result=1;  
while (y>0){  
    result=result*b;  
    y=y-1;  
}  
output(result);
```

- Some variables may be considered to hold **classified** information
- Then they should not be output ever!
- Actually nothing should be output that can give an attacker **any clue** on the classified information!

## Information Flow: Example

```
classified int b=3;  
int y=12;  
int result=1;  
while (y>0){  
    result=result*b;  
    y=y-1;  
}  
output(result);
```

- Some variables may be considered to hold **classified** information
- Then they should not be output ever!
- Actually nothing should be output that can give an attacker **any clue** on the classified information!
- Violated in this example: a term that depends on **b** is written into the unclassified variable **result** and that is output.  
This is called an **explicit flow**.

## Information Flow: Example

```
classified int b=3;  
int y=12;  
int result=1;  
while (y>0){  
    result=result*b;  
    y=y-1;  
}  
output(result);
```

- Some variables may be considered to hold **classified** information
- Then they should not be output ever!
- Actually nothing should be output that can give an attacker **any clue** on the classified information!
- Violated in this example: a term that depends on **b** is written into the unclassified variable **result** and that is output.  
This is called an **explicit flow**.
- Rationale: whoever can learn a value, can potentially see all information that has “flown to it”.

## Information Flow: Example

```
int b=3;
classified int y=12;
int result=1;
while (y>0){
    result=result*b;
    y=y-1;
}
output(result);
```

- What about this one?

## Information Flow: Example

```
int b=3;
classified int y=12;
int result=1;
while (y>0){
    result=result*b;
    y=y-1;
}
output(result);
```

- What about this one?
- Actually there is no assignment from classified variables to unclassified ones, so this is ok?

## Information Flow: Example

```
int b=3;
classified int y=12;
int result=1;
while (y>0){
    result=result*b;
    y=y-1;
}
output(result);
```

- What about this one?
- Actually there is no assignment from classified variables to unclassified ones, so this is ok?
- No, the value of the unclassified `result` may depend on the condition  $y > 0$ . This is called an **implicit flow**.

## Application: Information Flow

- High security guarantee: the code does not leak any information about the classified variables to an attacker who can only observe the unclassified variables.

## Application: Information Flow

- High security guarantee: the code does not leak any information about the classified variables to an attacker who can only observe the unclassified variables.
- Except, actually there may be side channels, e.g., how much time an algorithm needs may leak information about the data involved.

# Static Information Flow Analysis

A type-checking-like analysis:

- We have two security classes `public < confidential`
  - ★ In general we have a lattice of security classes
- **Explicit flow:** For every assignment  $x=e$  we check
  - ★ that security class of  $x$  is at least as high as that of  $e$ .
  - ★ The security class of  $e$  is the highest of any variable in  $e$  (and `public` if there are no variables in  $e$ ).
- **Implicit flow:** For every `while e { P }` we check
  - ★ that for every assignment  $x=e$  in  $P$ , the level of  $x$  is at least as high as  $e$ .
  - ★ Similar checks for `if e { P } else { Q }`.

Some similar analysis can be done for checking **integrity** of variables.

# Static Analysis

- With the type-checking we have again static analysis
  - ★ 02244 Logic for Security
- Types can sometimes be automatically inferred.
- Well-studied problems, so a lot of tool support.
- Over-approximation/false positives:
  - ★ an ill-typed program may sometimes be fine. We just cannot prove it with types.
  - ★ Compromise to sidetrack the halting problem.
  - ★ Soundness: when type-checked you can be sure that the information flow is fine.

# Side Channels

Computations can have **side effects**:

- Power consumption
- Timing (computation, memory access vs. cache)
- Noise (audio, electro-magnetic,...)

## Side Channel Analysis

- An intruder may be able to **observe** these side effects
- ... and obtain information about the computation itself.
- Typically this was **not intended** by its designers.

# Spectre



Spectre is a novel side-channel attack:<sup>1</sup>

- Exploit combines two common performance optimization of modern processors.
  - ★ Caching
  - ★ Speculative execution.
- Essentially: using speculative execution to leave “traces” in the cache and obtain these traces.
- Works on a wide variety of processors.

---

<sup>1</sup>Paul Kocher et al.: *Spectre Attacks: Exploiting Speculative Execution*, Security and Privacy 2019. See also [spectreattack.com](http://spectreattack.com)

# Speculative Execution

```
a=3;  
b=Array [6];  
c=f(a);
```

- Accessing Array[6] may be slow if not in cache.
- Instead of waiting, we could start computing  $f(a)$ .

# Speculative Execution

```
a=3;  
if ( Array[6]>3)  
    c=f(a);  
else  
    c=g(a);
```

- If `Array[6]` takes time, we do not know for a while what is the next command.

# Speculative Execution

```
a=3;  
if ( Array[6]>3)  
    c=f(a);  
else  
    c=g(a);
```

- If `Array[6]` takes time, we do not know for a while what is the next command.
- Branch prediction: if we know `Array[6]>3` has been often true in the past, we could **speculate** it is going to be true again.

# Speculative Execution

```
a=3;  
if ( Array[6]>3 )  
    c=f(a);  
else  
    c=g(a);
```

- If `Array[6]` takes time, we do not know for a while what is the next command.
- Branch prediction: if we know `Array[6]>3` has been often true in the past, we could **speculate** it is going to be true again.
  - ★ Go ahead computing `f(a)`
  - ★ If `Array[6]>3` turns out to be true, we win.
  - ★ Otherwise, we have to **revert** the processor,  
but we did not lose anything.

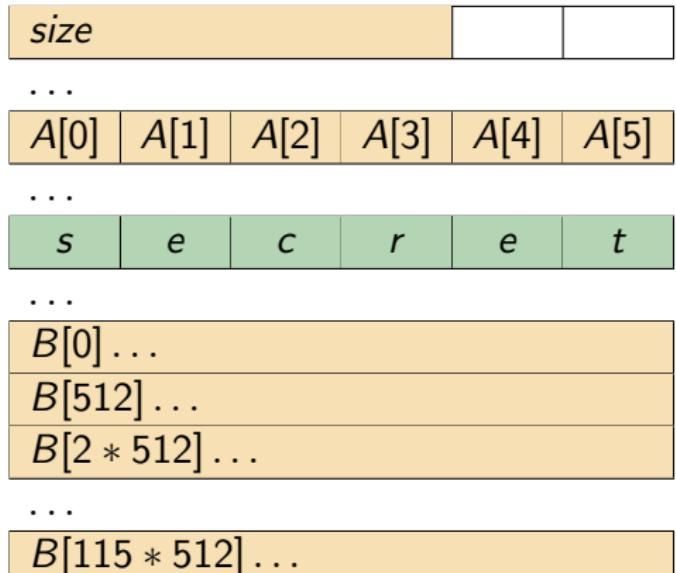
# Spectre: Example

## Victim code

```
if (y < size)
    temp &= B[A[y]*512]
```

## Dramatis personae:

- $y$ : chosen by the intruder.
- $A$ : array of size  $\text{size}$ .
- The intruder is allowed to know arrays  $A$  and  $B$ .
- There is a *secret* in the application memory somewhere after  $A$ .



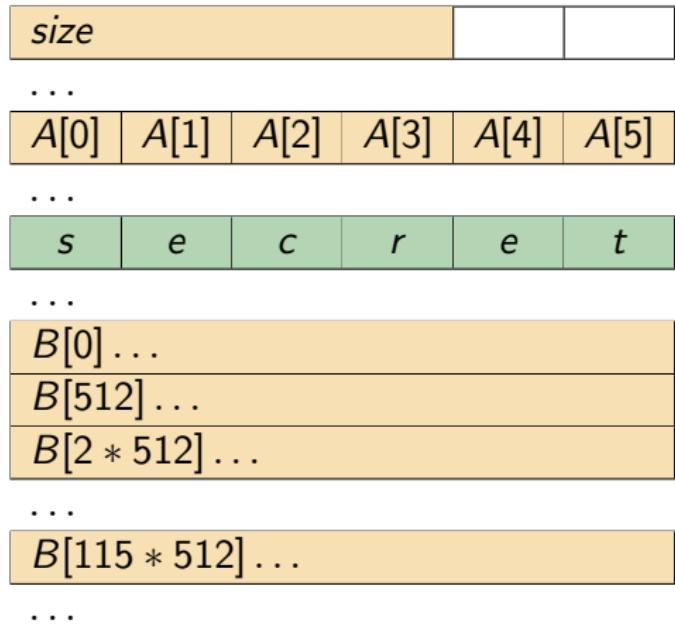
# Spectre: Example

## Victim code

```
if (y < size)
    temp &= B[A[y]*512]
```

Suppose

- size is **not in** the cache.
- A is **not in** the cache.
- B is **not in** the cache.
- *secret* is **in** the cache.
- Branch prediction expects `y < size` to be **true**.

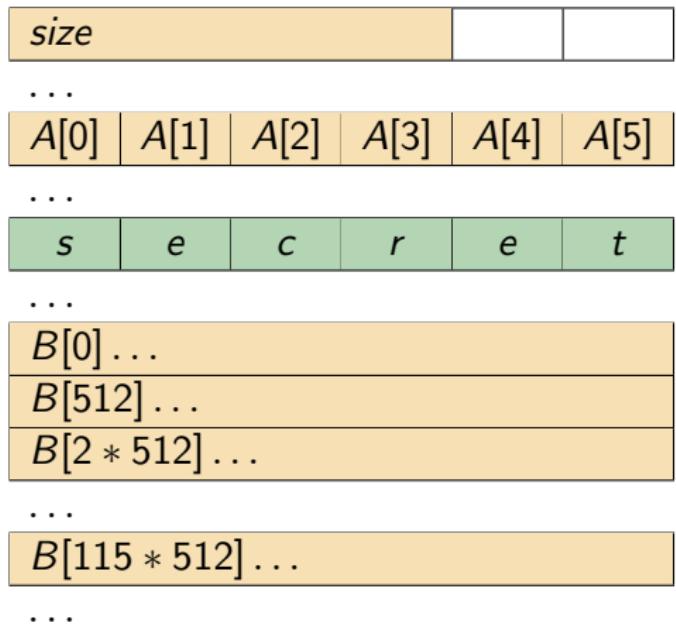


# Spectre: Example

## Victim code

```
if (y<size)
    temp &= B[A[y]*512]
```

- The intruder guesses  $y$ , so that  $A + y$  is the address of **secret**.
  - ★ So:  $y \geq size$ !

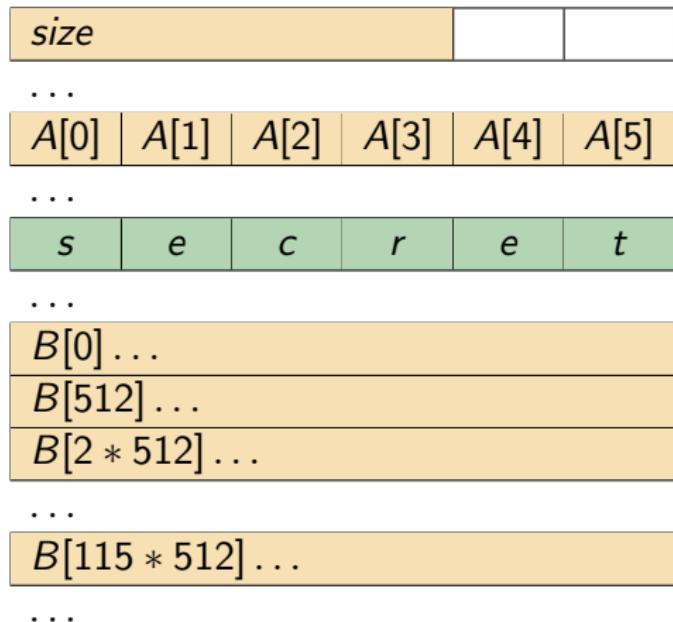


# Spectre: Example

## Victim code

```
if (y<size)
    temp &= B[A[y]*512]
```

- The intruder guesses  $y$ , so that  $A + y$  is the address of **secret**.
  - ★ So:  $y \geq size$ !
- Evaluate  $y < size$ :
  - ★  $size$  not cached,
  - ★ branch prediction is positive

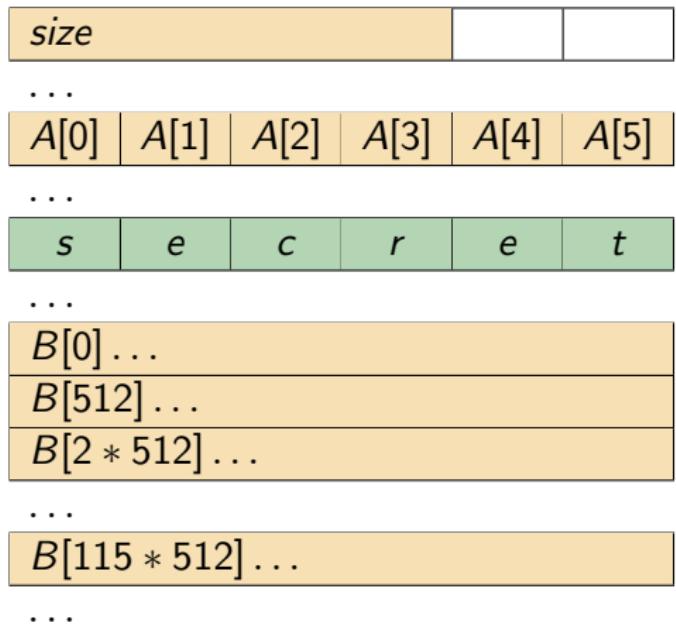


# Spectre: Example

## Victim code

```
if (y<size)
    temp &= B[A[y]*512]
```

- The intruder guesses  $y$ , so that  $A + y$  is the address of **secret**.
  - ★ So:  $y \geq size$ !
- Evaluate  $y < size$ :
  - ★  $size$  not cached,
  - ★ branch prediction is positive



Thus start **speculative execution** of  $B[A[y]*512]$ .

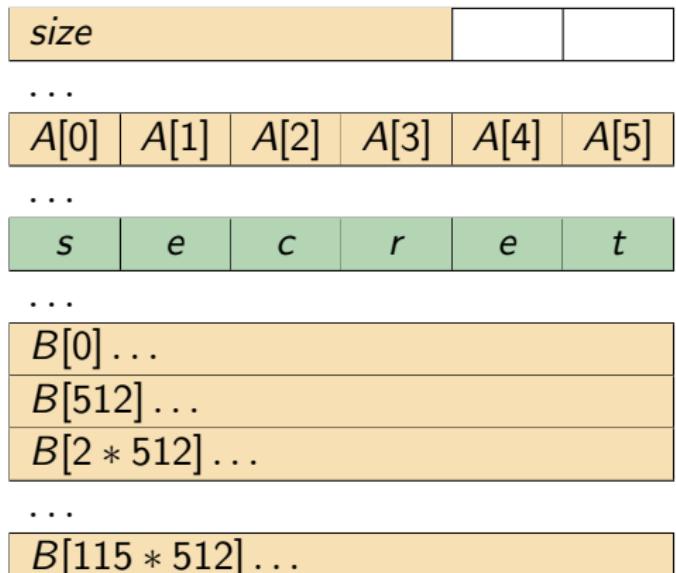
# Spectre: Example

## Victim code

```
if (y < size)
    temp &= B[A[y]*512]
```

Speculative execution of  
 $B[A[y]*512]$

- $A + y$  is address of secret.



# Spectre: Example

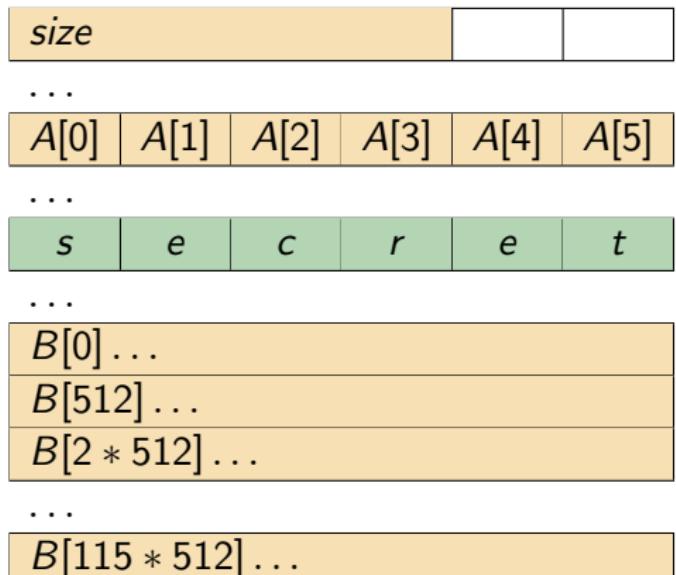
## Victim code

```
if (y < size)
    temp &= B[A[y]*512]
```

Speculative execution of

$B[A[y]*512]$

- $A + y$  is address of **secret**.
- So  $A[y]$  is in the cache!



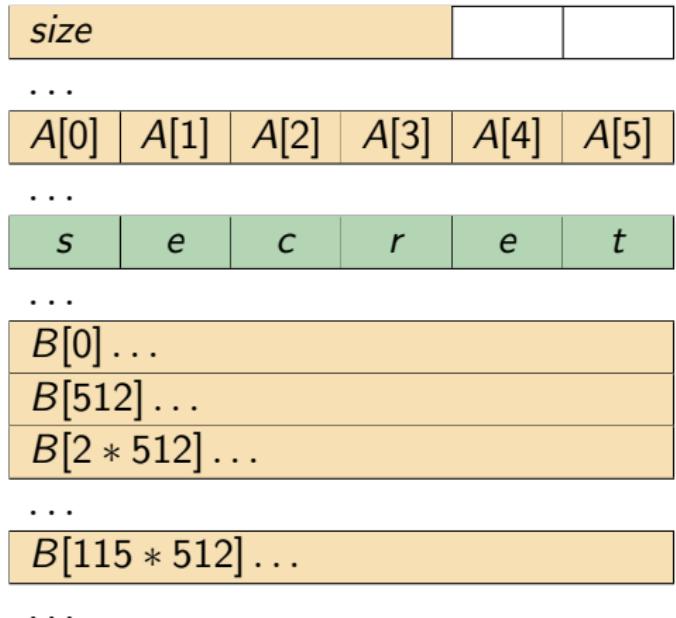
# Spectre: Example

## Victim code

```
if (y < size)
    temp &= B[A[y]*512]
```

Speculative execution of  
 $B[A[y]*512]$

- $A + y$  is address of **secret**.
- So  $A[y]$  is in the cache!
- So processor loads  
 $B[A[y] * 512] =$   
 $B[115 * 512]$



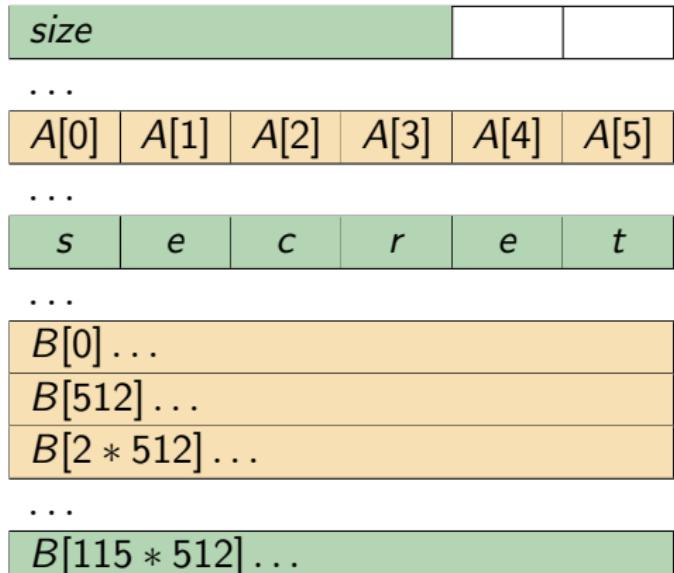
# Spectre: Example

## Victim code

```
if (y < size)
    temp &= B[A[y]*512]
```

Speculative execution of  
 $B[A[y]*512]$

- $A + y$  is address of **secret**.
- So  $A[y]$  is in the cache!
- So processor loads  
 $B[A[y] * 512] =$   
 $B[115 * 512]$
- ... it is now cached.



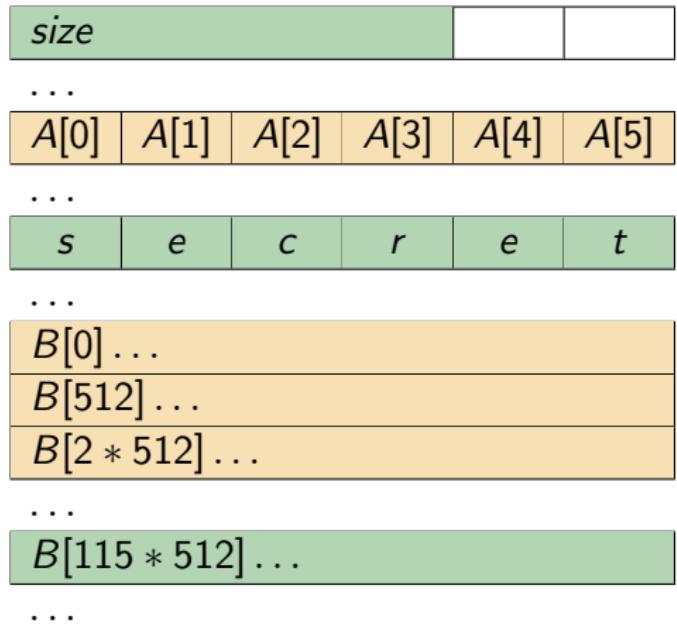
# Spectre: Example

## Victim code

```
if (y<size)
    temp &= B[A[y]*512]
```

Also `size` has arrived

- Thus, `y<size` can be computed to be false.
- Thus, the processor reverts to the state before.



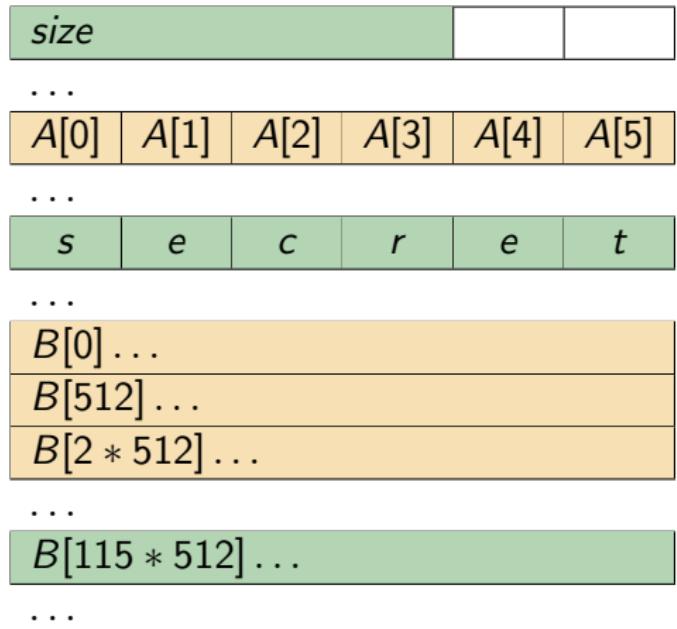
# Spectre: Example

## Victim code

```
if (y<size)
    temp &= B[A[y]*512]
```

Also `size` has arrived

- Thus,  $y < \text{size}$  can be computed to be false.
- Thus, the processor reverts to the state before.
- **But the cache remains!**



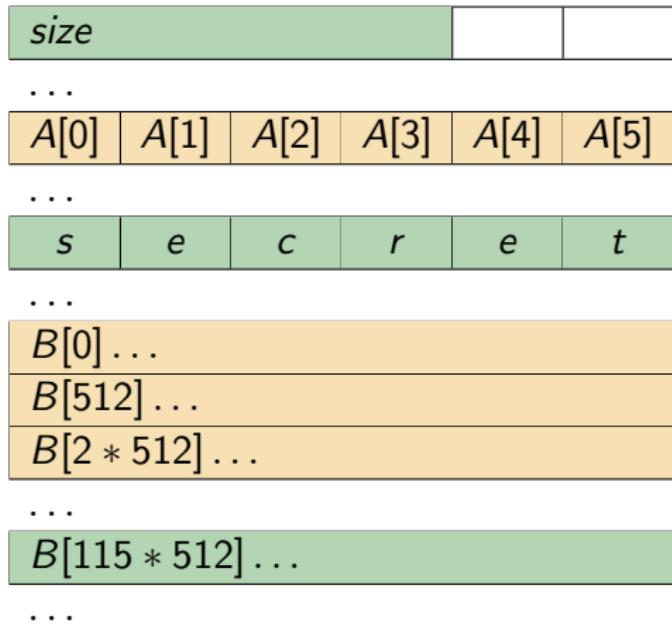
# Spectre: Example

## Victim code

```
if (y<size)
    temp &= B[A[y]*512]
```

Also `size` has arrived

- Thus,  $y < \text{size}$  can be computed to be false.
- Thus, the processor reverts to the state before.
- **But the cache remains!**



The intruder can now try to access  $B[k * 512]$  for  $k \in \{0..255\}$ .

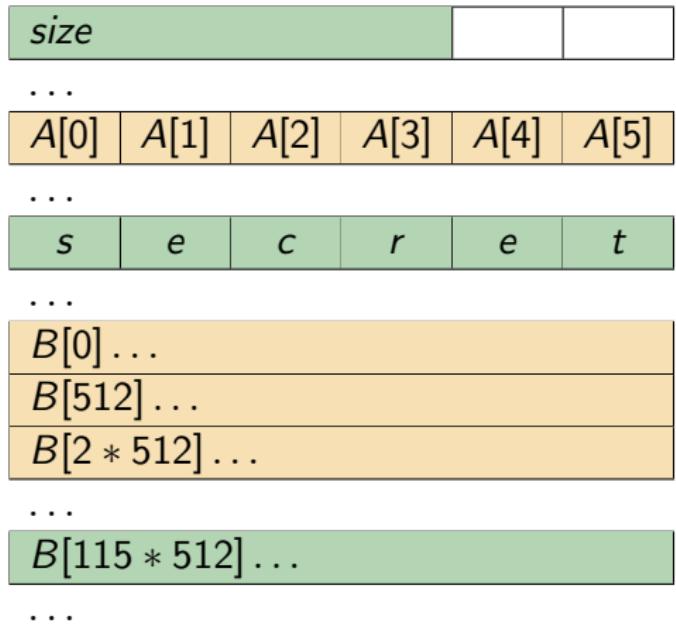
# Spectre: Example

## Victim code

```
if (y<size)
    temp &= B[A[y]*512]
```

Also `size` has arrived

- Thus,  $y < \text{size}$  can be computed to be false.
- Thus, the processor reverts to the state before.
- **But the cache remains!**



The intruder can now try to access  $B[k * 512]$  for  $k \in \{0..255\}$ . This is fast iff  $k = 115$ : he knows the first letter of the secret.

# Spectre: Countermeasures?

What to do about this?

- Speculative load hardening<sup>2</sup>
- Idea: the compiler Clang/LLVM inserts extra operations to mitigate the leak.

---

<sup>2</sup>Chandler Carruth: *Speculative load hardening*  
<https://llvm.org/docs/SpeculativeLoadHardening.html>

# Speculative load hardening

The **hardened code** contains a few more operations:

```
    mov      size, %rax
    mov      y, %rbx
    mov      $0, $rdx
    cmp      %rbx, %rax
    jbe      END
    cmovbe $-1, %rdx
    mov      A(%rbx), %rax
    shl      $9, %rax
    or       %rdx, %rax
    mov      B(%rax), %rax
    or       %rdx, %rax
    and     %rax, temp
END:   ...
```

# Side Channel Analysis

- Side channels are usually leaks that the designers never have thought of.
- Many problems exist for years without getting noticed – how much else is out there?
- Spectre: tons of variations emerging e.g. remote attacks.<sup>3</sup>
- Countermeasures: for instance compiler-based techniques.<sup>4</sup>
- Very active research field:
  - ★ In order to prove a solution correct, one needs a precise model first.
  - ★ For instance Speculative non-interference and the Spectector tool.<sup>5</sup>

---

<sup>3</sup>Michael Schwarz et al.: *NetSpectre*, ESORICS 2019.

<sup>4</sup>Chandler Carruth: *Speculative load hardening*

<https://llvm.org/docs/SpeculativeLoadHardening.html>

<sup>5</sup>Marco Guarnieri et al.: *Spectector*, Security & Privacy 2020.



# Conclusions

- Software security is extremely subtle
  - ★ It is very hard to design systems that are impeccable
- Every small imperfection can be disastrous
  - ★ Imperfections can be amplified by an attacker in unforeseen ways.
  - ★ Don't rely on "*this cannot be exploited*"-statements
- Suggestions for secure software
  - ★ Prudent: Follow good engineering practices
  - ★ Formalize: Try to be precise about your assumptions and goals
  - ★ Proof: Try to formally test/verify automatically or manually.

## Students who like this course also liked... :-)

There are several related courses, e.g.

- on security: 02244 Logic for Security
- on formal methods: model-checking, program analysis, computer science modeling

If you are looking for a Bsc/Msc thesis topic in security or formal methods, come by for a chat!

## Legal Aspects of Computer Security



$$\Theta^{\sqrt{17}} + \Omega^{\int \delta e^{i\pi} = \infty} = \sum_{x^2 > 0}$$

DTU Compute

Department of Applied Mathematics and Computer Science

## Basic Ideas

- Legal issues are related to questions such as:
  - What support does the law give to the protection of computers, programs and data?
  - What support does the law give to the protection of intellectual property?
  - What rights does the law give "legal persons" with respect to computers, programs and data?
- More specific issues:
  - Computer crime (definition, evidence, punishment,...)
  - Legal protection of code and data (copyright, patents,...)
  - Programmers' and employees' rights
  - Protection of personal data
  - Consumer protection

## The Role of Computers in Crime

- Computer (or network) as target
  - Using computer(s) to attack a victim's computer
  - Attack on Confidentiality, Integrity or Availability of data or systems
  - Cyber Terrorism and Cyber Warfare
- Computer as tool
  - Fraud - Phishing, Nigerian 419 (after Fraud § in Nigerian Criminal Code)
  - Ransomware (WannaCry, NotPetya, ...)
  - Gambling
  - Copyright infringements (aka piracy)
  - Harassment (aka cyber bullying), stalking, etc.
- Computer as accomplice
  - Personal information (diaries, downloaded e-mails,)
    - *Other evidence unknown to suspect - web-history, cookies, ...*
  - Contraband - digital goods, copyrighted material, (child) pornography
  - Stolen information – trade secrets, credit card data
  - Monetizing proceeds of cyber crime (online marketplaces, Bitcoins, ...)

## Cyber Crime

- Problem of jurisdiction
  - Laws are mostly national, cyber crime is typically transnational
    - *International treaties/conventions may codify crime in several countries*
    - *Netiquette may (self-)regulate some unwanted behaviour*
  - Where is crime committed?
    - *Who should investigate, prosecute and sentence*
      - Location of victim, criminal or beneficiary?
    - *The crime may not be a legal offence in all relevant jurisdictions*
  - How to investigate
    - *Collecting evidence requires collaboration among law enforcement agencies*
  - How to get hold of the accused person(s) / evidence?
    - *Extradition agreements between national states*
  - How to punish criminals
    - *Some criminals may be tried in absentia*

## Problem of Jurisdiction



## Convention on Cyber Crime

- Convention established by the Council of Europe
  - 30 states sign Convention at opening ceremony in Budapest in 2001
- First international treaty on cyber crimes, dealing particularly with:
  - Infringements of copyright
  - Computer-related fraud
  - Child pornography
  - Violations of network security
  - Hate crimes and Racism as an addendum (optional extras)
- Contains a series of powers and procedures such as search of computer networks and interception
- Main objective is to pursue a common criminal policy aimed at protection of society against cyber-crime, especially by adopting appropriate legislation and fostering international co-operation

## Protecting Intellectual Property

- Three legislative frameworks are applicable to programs and data:
  - Copyright law (publication of works of art)
    - *Copyright law was conceived to protect works of art, music, literature and written scholarship*
    - *Provides incentive to produce works of art*
  - Patent law (public information about inventions)
    - *Patent law was conceived to protect inventions and innovation in science, technology and engineering*
    - *Provides an incentive to inventors to disclose their inventions*
  - Trade secrets law (secret information incl. data and processes)
    - *Trade secrets identify information that must be kept secret*
      - Punish people who reveal the secret to outsiders
    - *Provides a legal framework to deal with disclosure of confidential info.*

## Copyrights, Patents & Trade Secrets

	Copyright	Patent	Trade Secret
Protects	Expression of idea, not idea itself	Invention – the way something works	A secret, competitive advantage
Protected objects made public	Yes, intention is to promote publication	Design filed at Patent Office	No
Requirement to distribute	Yes	No	No
Ease of filing	Very easy, do-it-yourself	Very complicated; specialist lawyer suggested	No filing
Duration	Life of human originator plus 70 years, or total of 95 years for a company	19 years	Indefinite
Legal protection	Sue if unauthorized copy sold	Sue if invention copied	Sue if secret improperly obtained



## Data Protection

- Large amounts of data are being collected about all of us



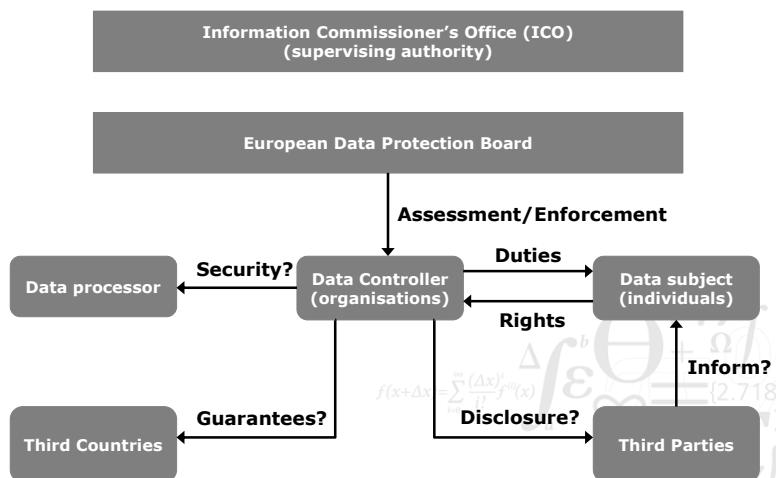
9

DTU Compute Technical University of Denmark

02239 – Data Security



## GDPR - Data Protection Model



10

DTU Compute Technical University of Denmark

02239 – Data Security

## GDPR - Definitions

- Natural person = a living individual
- Natural persons have rights associated with:
  - The protection of personal data
  - The protection of the processing of personal data
  - The unrestricted movement of personal data within the EU
- In material scope:
  - Personal data that is processed wholly or partly by automated means;
  - Personal data that is part of a filing system, or intended to be
- The Regulation applies to controllers and processors in the EU irrespective of where processing takes place
- It applies to controllers not in the EU

## GDPR - Remedies, liabilities and penalties

- Natural Persons have rights
  - Judicial remedy where their rights have been infringed as a result of the processing of personal data.
    - *In the courts of the Member State where the controller or processor has an establishment*
    - *In the courts of the Member State where the data subject habitually resides*
  - Any person who has suffered material, or non-material, damage shall have the right to receive compensation from the controller or processor
  - Controller involved in processing shall be liable for damage caused by processing
- Administrative fines
  - Imposition of administrative fines will in each case be effective, proportionate, and dissuasive (*No administrative fines in Denmark, fines imposed by the courts*)
    - *taking into account technical and organisational measures implemented;*
  - €10,000,000 or, in the case of an undertaking, up to 2% of the total worldwide annual turnover of the preceding financial year
  - €20,000,000 or, in case of an undertaking, 4% total worldwide annual turnover in the preceding financial year

## GDPR - Personal Data Breaches (Article 33)

- The definition of a Personal Data Breach in GDPR:
  - A 'personal data breach' means a breach of security leading to the accidental or unlawful destruction, loss, alteration, unauthorised disclosure of, or access to, personal data transmitted, stored or otherwise processed.
- Obligation for data processor to notify data controller
  - Notification without undue delay after becoming aware
  - No exemptions
  - All data breaches have to be reported
- Obligation for data controller to notify the supervisory authority
  - Notification without undue delay and not later than 72 hours
  - Unnecessary in certain circumstances
  - Description of the nature of the breach
  - No requirement to notify if unlikely to result in a high risk to the rights and freedoms of natural persons
  - Failure to report within 72 hours must be explained

## GDPR - Rights of Data Subjects

- The controller shall take appropriate measures to provide any information ... relating to processing to the data subject in a concise, transparent, intelligible and easily accessible form, using clear and plain language (Article 11-1)
- The controller shall facilitate the exercise of data subject rights (Article 11-2)
  - Rights to
    - *Consent*
    - *Access*
    - *Rectification*
    - *Erasure*
    - *Objection*
  - the right to data portability;
  - the right to withdraw consent at any time;
  - the right to lodge a complaint with a supervisory authority;
  - The right to be informed of the existence of automated decision-making, including profiling, as well as the anticipated consequences for the data subject

## GDPR - the Principle of Accountability

- Governance: Board accountability
  - Corporate risk register
  - Nominated responsible director
- Clear roles and responsibilities
  - Data Protection Officer
- Privacy Compliance Framework
  - PIMS/ISMS
  - Cyber incident response
  - Cyber Essentials a minimum security standard
  - Certification and data seals (Article 42) –ISO 27001
- Data Protection by Design and by Default
  - Data Flow Audits
  - Data Protection Impact Assessments (DPIA)
    - *Mandatory for many organizations*
    - *Legal requirements around how performed and data collected*

## GDPR – Lawfulness (Article 5 & 6)

- Secure against accidental loss, destruction or damage
- Processing must be lawful –which means, inter alia:
  - Data subject must give consent for specific purposes
  - Other specific circumstances where consent is not required
    - *So that controller can comply with legal obligations etc.*
- One month to respond to Subject Access Requests & no charges
- Controllers and processors clearly distinguished
  - Clearly identified obligations
  - Controllers responsible for ensuring processors comply with contractual terms for processing information
  - Processors must operate under a legally binding contract
    - *Note issues around extra-territoriality*

## GDPR – Consent (Article 7-9)

- Consent must be clear and affirmative
  - Must be able to demonstrate that consent was given
  - Silence or inactivity does not constitute consent
  - Consent must be clear, intelligible, easily accessible, to be binding
  - Consent can be withdrawn at any time, and it must be as easy to withdraw consent as give it
- Special conditions apply for children (under 16) to give consent
- Explicit consent necessary for processing sensitive personal data
  - Race, ethnic origin, gender, etc.
  - Specific circumstances allow non-consensual processing,
    - *Regulatory or legal requirements*
    - *To protect vital interests of the data subject*
    - ...
- Secure against accidental loss, destruction or damage (article 5)

## GDPR – Transparency (Article 12 – 18)

- Any communications with a data subject must be concise, transparent, intelligible
  - This excludes legal jargon
- Controller must be transparent in providing information about itself and the purposes of the processing
- Controller must provide data subject with information about their rights
- Specific provisions (Article 14) covering data not obtained directly from the data subject
- Rights to access, rectification, erasure ('right to be forgotten'), to restriction of processing, and data portability

## GDPR - Privacy by Design (Article 25 et seq.)

- Privacy must now be designed into data processing by default
- Data controllers/processors not established in the EU must designate a representative
- Data Privacy Impact Assessments mandatory (article 35)
  - For technologies and processes that are likely to result in a high risk to rights of data subjects
- Data audits
  - GDPR applies to existing data, as well as future data
  - Privacy may have to be designed in retrospectively
  - Organizations need to identify what PII they hold, where, on what grounds, and how it is secured in a way that will meet requirements of GDPR

## GDPR - Security of Personal Data (Article 32)

- Sixth Principle: Data must be processed "*in a manner that ensures appropriate security of the personal data, including protection against unauthorised or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organisational measures*"
- A requirement for data controllers and data processors to implement a level of security appropriate to the risk, including:
  - pseudonymisation and encryption of personal data;
  - ensure the ongoing confidentiality, integrity and availability of systems;
  - a process for regularly testing, assessing and evaluating the effectiveness of security measures;
  - security measures taken need to comply with the concept of privacy by design;
- Certifications demonstrate intent: Cyber Essentials, ISO 27001

## GDPR - Data Protection Officer (DPO)

- DPO mandatory in organizations processing substantial volumes of PII (Article 37)
- A protected position, reporting directly to senior management
  - Appropriately qualified
  - Consulted in respect of all data processing activities
- Will be a 'good practice' appointment outside the mandatory appointments
- Most staff dealing with PII (e.g. HR, marketing, etc.) will need at least basic training
- Staff awareness training also critical (accidental release of PII could have financially damaging consequences)

## GDPR - International Transfers (Article 44)

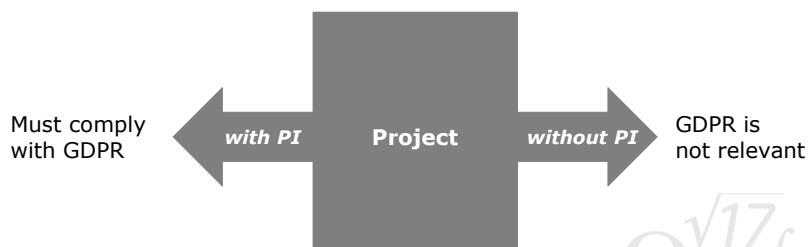
- Any transfer of personal data by controller or processor shall take place only if certain conditions are complied with:
  - Transfers on the basis of adequacy;
  - Transfers subject to the appropriate safeguards
  - Binding corporate rules apply
- All provisions shall be applied to ensure the protection of natural persons is not undermined
- To countries with similar data protection regulations
  - Cloud providers are a key risk area
    - *Schrems II decision in European Court of Justice raises questions about transfer to US and US owned companies*
  - Highest penalties apply to breaches of these provisions

## Nine Steps to GDPR compliance

1. Establish governance framework – board awareness, risk register, accountability framework, review
  2. Appoint and train a DPO
  3. Data inventory – identify processors, unlawfully held data
  4. Data flow audit
  5. Compliance gap analysis
    - Ensure FPN and SAR documents and processes are robust and legal
  6. DPIA and security gap analysis
    - Penetration testing
  7. Remediate
    1. Privacy compliance framework
    2. Cyber Essentials/Ten Steps to Cyber Security/ISO 27001
  8. Data breach response process (NB: Test!)
  9. Monitor, audit and continually improve
- NB: steps can be tackled in parallel

## GDPR in Practice

- When considering the data you collect when implanting a project



- GDPR can be ignored *if you do not collect personal data*
  - This is one reason why privacy enhancing technologies are so important