Security and Cryptography 2017

Mirosław Kutyłowski

grading criteria: 50% exam, 50% assignments

skills to be learned: developing end-to-end security systems, they must be flawless!

rules: do not memorize the standards, they change. Only the skills are important

presence: obligatory during the lectures

exam date: TBA (early to enable internships in February), optionally: midterm exam(s)

I. EXAMPLE TO LEARN FROM: PKI FAILURE

PKI - Public Key Infrastructure

- strong authentication of digital documents with digital signatures seems to be possible
- in fact we get an evidence that the holder of a private key has created a signature
- who holds the key? PKI has to provide a certified answer to this question
- PKI is not a cryptographic solution - it is an organizational framework (using some crypto tools)

PKI, X.509 standard

- a certificate binds a public key with an ID of its alleged owner,
- a couple of other fields, like validity date, key usage, certification policy, ...
- certificate signed by CA (Certification Authority)
- tree of CA's (or directed acyclic graph), with roots as "root of trust"
- status of a certificate may change - revocation
- checking status methods: CRL, OCSP

reasons for PKI failure:

a nice concept of digital signatures but

1. big infrastructure required:
   - big effort and cost

- long time planning needed (so possible in China, but not in Europe)
- unclear financial return

2. scope of necessary coordination,
   - in order to work must be designed at least for the Common Market
   - example of killing the concept: link to certification policy in Polish
3. lack of interoperability (sometimes as business goal)
   - companies make efforts to eliminate competition
   - standarization may be focused on having market shares
4. necessary trust in roots
   - how do you know that the root is honest?
5. registration: single point of fraud, (e.g. with fake breeding documents)
   - once you get a certificate you may forge signatures
6. responsibility of CA
   - fiancial risk – based on risk or responsibility
7. cost - who will pay? For the end user the initial cost is too high.
   - certificates are too expensive for just a few signatures (at least initially)
8. legal strength of signatures
   - if scheme broken or signing devices turn out to be insecure you are anyway responsible for the signatures
9. unsolved problem of revocation: possible to check the status in the past but not now

reason: mismatch of requirements and interests with the designed solution

MAJOR PROBLEM: how to design/buy good systems?

II. COMMON CRITERIA FRAMEWORK

http://www.commoncriteriaportal.org/

Problem: somebody has to deploy a secure IT system, how to purchase it?

- problematic requirements according to BSI guide:
  1. **incomplete** – forgetting some threats is common
  2. **not embedded:** not corresponding really to the environment where the product has to be deployed
  3. **implicit:** custemer has in mind but the developer might be unaware of them

iv. **not testable**: ambiguous, source of legal disputes, ...

v. **too detailed:** unnecessary details make it harder to adjust the design

vi. **unspecified meaning:** e.g. "*protect privacy*"

vii. **inconsistent:** e.g. ignoring trade-offs

- *specification-based purchasing process* versus *selection-based purchasing process*
- the user is not capable of determining the properties of the product himself: too complicated, too specialized knowledge required, a single error makes the product useless
- specifications of concrete products might be useless for the customers – hard to understand and compare the products
- informal specifications and descriptions, no crucial data

Idea of Common Criteria Framework:

- standardize the process of
  - designing requirements (Protection Profile, PP) (customer)
  - designing products (Security Target ST), (developer)
  - evaluation of products (licensed labs checking conformance of implementation with the documentation) (certification body)
- international agreement of bodies from some countries (USA, France, UK, Germany, India, Turkey, Sweden, Spain, Australia, Canada, Malaysia, Netherlands, Korea, New Zeland, Italy, Turkey) but Israel only "consuming", no Poland, China, Singapore,
- idea: ease the process, reuse work, build up from standard components
- typically ST as a response for PP:
  - more detailed
  - maybe chooses some concrete options
  - maybe fulfills more requirements (more PP)
  - relation with PP should be testable

Value:

- CC certification does not mean a product is secure
- it only says that is has been developed according to PP
- assurance level concerns only the stated requirements , e.g. trivial requirements ⇒ high EAL level (common mistake in public procurement: EAL level ... without specifying PP)

- but it is cleaning up the zoo of different assumptions, descriptions, ...

Example for PP: BAC (Basic Access Control)

- used to secure wireless communication between a reader and an e-Passport (of an old generation)
- encryption primitive

$$\mathrm{EM}(K,S) = \mathrm{Enc}(\mathrm{KB}_{\mathrm{Enc}}, S) \| \mathrm{MAC}(\mathrm{KB}_{\mathrm{Mac}}, \mathrm{Enc}(\mathrm{KB}_{\mathrm{Enc}}, S), S)$$

  where the key K is $(\mathrm{KB}_{\mathrm{Enc}}, \mathrm{KB}_{\mathrm{Mac}})$

- steps:

  1. The MRTD chip sends a nonce $r_{\mathrm{PI}\mathbb{C}C}$ to the terminal
  2. The terminal sends the encrypted challenge

  $$e_{\mathrm{PCD}} = \mathrm{EM}(K, r_{\mathrm{PCD}}, r_{\mathrm{PI}\mathbb{C}C}, K_{\mathrm{PCD}})$$

     to the MRTD chip, where $r_{\mathrm{PI}\mathbb{C}C}$ is the MRTD chip's nonce, r_{PCD} is the terminal's randomly chosen nonce, and K_{PCD} is keying material for the generation of the session keys.
  3. The MRTD chip decrypts and verifies $r_{\mathrm{PI}\mathbb{C}C}$, responds with

  $$e_{\mathrm{PICC}} = \mathrm{EM}(K, r_{\mathrm{PICC}}, r_{\mathrm{PCD}}, K_{\mathrm{PICC}})$$

  4. The terminal decrypts and verifies r_{PCD}
  5. both sides derive $K_{\mathrm{Enc}}, K_{\mathrm{Mac}}$ from the master key

  $$K_{\mathrm{PICC}} \,\mathrm{XOR}\, K_{\mathrm{PCD}}$$

     and a sequence number derived from the random nonces (key derivation function)

- ***K* derived from information available on the machine readable zone (optical reader applied, not available via wireless connection)**
- implementation: biometric passports.
- a simple system. Really?

Common Criteria Protection Profile Machine Readable Travel Document with ICAO Application, Basic Access Control BSI-CC-PP-0055

1. Introduction

`aimed for customers looking for proper products, overview`

1.1 PP reference

`basic data, registration data`

Title: Protection Profile - Machine Readable Travel Document with ICAO Application and Basic Access Control (MRTD-PP)

Sponsor: Bundesamt für Sicherheit in der Informationstechnik CC Version: 3.1 (Revision 2)

Assurance Level: The minimum assurance level for this PP is EAL4 augmented.

General Status: Final

Version Number: 1.10

Registration: BSI-CC-PP-0055

Keywords: ICAO, machine readable travel document, basic access control

1.2 TOE Overview

- Target of Evaluation
- "is aimed at potential consumers who are looking through lists of evaluated TOEs/Products to find TOEs that may meet their security needs, and are supported by their hardware, software and firmware"
- important sections:
    - Usage and major security features of the TOE
    - TOE type
    - Required non-TOE hardware/software/firmware
- Definition, Type

  which parts, which general purpose, which functionalities are present and which are missing, e.g. ATM card with no contactless payments

- Usage and security features

  crucial properties of the system (high level) and security features from the point of view of the security effect and not how it is achieved

- life cycle

  the product in the whole life cycle including manufacturing, delivery and destroying

- Required non-TOE hardware/software/firmware: other components that can be crucial for evaluation

2. Conformance Claim

- CC Conformance Claim: version of CC
- PP claim: other PP taken into account in a plug-and-play way
- Package claim: which EAL package level

EAL packages:

- The CC formalizes assurance into 6 categories (the so-called "assurance classes" which are further subdivided into 27 sub-categories (the so-called "assurance families"). In each assurance family, the CC allows grading of an evaluation with respect to that assurance family.

- 7 predefined ratings, called evaluation assurance levels or EALs. called EAL1 to EAL7, with EAL1 the lowest and EAL7 the highest

- Each EAL can be seen as a set of 27 numbers, one for each assurance family. EAL1 assigns a rating of 1 to 13 of the assurance families, and 0 to the other 14 assurance families, while EAL2 assigns the rating 2 to 7 assurance families, the rating 1 to 11 assurance families, and 0 to the other 9 assurance families

- monotonic: EALn+1 gives at least the same assurance level as EALn in each assurance families

- levels:

  - EAL1: Functionally Tested:

    - correct operation, no serious threats

    - minimal effort from the manufacturer

  - EAL2: Structurally Tested

    - delivery of design information and test results,

    - effort on the part of the developer than is consistent with good commercial practice.

  - EAL3: Methodically Tested and Checked

    - maximum assurance from positive security engineering at the design stage without substantial alteration of existing sound development practices.

    - developers or users require a moderate level of independently assured security, and require a thorough investigation of the TOE and its development without substantial re-engineering.

  - EAL4: Methodically Designed, Tested and Reviewed

    - maximum assurance from positive security engineering based on good commercial development practices which, though rigorous, do not require substantial specialist knowledge, skills, and other resources.

    - the highest level at which it is likely to be economically feasible to retrofit to an existing product line.

  - EAL5: Semiformally Designed and Tested

  - EAL6: Semiformally Verified Design and Tested

  - EAL7: Formally Verified Design and Tested

- assurance classes:

  → development:

    – ADV_ARC - 1 1 1 1 1 1 architecture requirements

    – ADV_FSP 1 2 3 4 5 5 6 functional specifications

    – ADV_IMP - - - 1 1 2 2 implementation representation

    – ADV_INT - - - - 2 3 3 "is designed and structured such that the likelihood of flaws is reduced and that maintenance can be more readily performed without the introduction of flaws"?

    – ADV_SPM - - - - - 1 1 security policy modeling

    – ADV_TDS - 1 2 3 4 5 6 TOE design

  → guidance documents

    – AGD_OPE 1 1 1 1 1 1 1 Operational user guid ance

    – AGD_PRE 1 1 1 1 1 1 1 Preparative procedures

  → life-cycle support

    – ALC_CMC 1 2 3 4 4 5 5 CM capabilities

    – ALC_CMS 1 2 3 4 5 5 5 CM scope

    – ALC_DEL - 1 1 1 1 1 1 Delivery

    – ALC_DVS - - 1 1 1 2 2 Development securit

    – ALC_FLR - - - - - - - Flaw remediation

    – ALC_LCD - - 1 1 1 1 2 Life-cycle definition

    – ALC_TAT - - - 1 2 3 3 Tools and techniques

  → security target evaluation

    – ASE_CCL 1 1 1 1 1 1 1 Conformance claims

    – ASE_ECD 1 1 1 1 1 1 1 Extended components definition

    – ASE_INT 1 1 1 1 1 1 1 ST introduction

    – ASE_OBJ 1 2 2 2 2 2 2 Security objectives

    – ASE_REQ 1 2 2 2 2 2 2 Security requirements

    – ASE_SPD - 1 1 1 1 1 1 Security problem definition

    – ASE_TSS - 1 1 1 1 1 1 TOE summary specification

- → tests
    - ATE_COV 1 2 2 2 3 3 Coverage
    - ATE_DPT 1 1 3 3 4 Depth
    - ATE_FUN 1 1 1 1 2 2 Functional tests
    - ATE_IND 1 2 2 2 2 2 3 Independent testing
- → vulnerability assesment
    - AVA_VAN 1 2 2 3 4 5 5 Vulnerability analysis

- for example, a product could score in the assurance family developer test coverage (ATE_COV):
  - 0: It is not known whether the developer has performed tests on the product;
  - 1: The developer has performed some tests on some interfaces of the product;
  - 2: The developer has performed some tests on all interfaces of the product;
  - 3: The developer has performed a very large amount of tests on all interfaces of the product
- example more formal: ALC_FLR
  - ALC_FLR.1:
    - *The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.*
    - *The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.*
    - *The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.*
    - *The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.*
  - ALC_FLR.2:
    - ALC_FLR.1 as before
    - *The flaw remediation procedures shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE.*
    - *The procedures for processing reported security flaws shall ensure that any reported flaws are remediated and the remediation procedures issued to TOE users.*
    - *The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.*

  - *The flaw remediation guidance shall describe a means by which TOE users report to the developer any susp ected security flaws in the TOE.*

- ALC_FLR.3:
  - first 5 as before
  - *The flaw remediation procedures shall include a procedure requiring timely response and the automatic distribution of security flaw reports and the associated corrections to registered users who might be affected by the security flaw.*
  - next 3 as before
  - *The flaw remediation guidance shall describe a means by which TOE users may register with the developer, to be eligible to receive security flaw reports and corrections.*
  - *The flaw remediation guidance shall iden tify the specific points of contact for all reports and enquiries about security issues involving the TOE.*

CEM -Common Evaluation Methodology

- given CC documentation, EAL classification etc, perform a check
- idea: evaluation by non-experts, semi-automated, mainly paper work
- mapping:
  - assurance class $\Rightarrow$ activity
  - assurance component $\Rightarrow$ sub-activity
  - evaluator action element $\Rightarrow$ action
- responsibilities:
  - sponsor: requesting and supporting an evaluation. different agreements for the evaluation (e.g. commissioning the evaluation), providing evaluation evidence.
  - developer: produces TOE, providing the evidence required for the evaluation on behalf of the sponsor.
  - evaluator: performs the evaluation tasks required in the context of an evaluation, performs the evaluation sub-activities and provides the results of the evaluation assessment to the evaluation authority.
  - evaluation authority: establishes and maintains the scheme, monitors the evaluation conducted by the evaluator, issues certification/validation reports as well as certificates based on the evaluation results
- verdicts: pass, fail, inconclusive
- parts:
  - evaluation input task (are all documents available to perform evaluation?)
  - evaluation sub-activities

- evaluation output task (deliver the Observation Report (OR) and the Evaluation Technical Report (ETR)).

- demonstration of the technical competence task

3 Security Problem Definition

- **Object Security Problem (OSP)**: "The security problem definition defines the security problem that is to be addressed.

  – `axiomatic:` deriving the security problem definition outside the CC scope

  – `crucial:` the usefulness of the results of an evaluation strongly depends on the security problem definition.

  – `requires work:` spend significant resources and use well-defined processes and analyses to derive a good security problem definition.

- good example:

  Secure signature-creation devices must, by appropriate technical and operational means, ensure at the least that:

  1) The signature-creation-data used for signature-creation can practically occur only once, and that their secrecy is reasonably assured;

  2) The signature-creation-data used for signature-creation cannot, with reasonable assurance, be derived and the signature is protected against forgery using currently available technology;

  3) The signature-creation-data used for signature-creation can be reliably protected by the legitimate signatory against the use of others

- **assets:** entities that someone places value upon. Examples of assets include: - contents of a file or a server; - the authenticity of votes cast in an election; - the availability of an electronic commerce process; - the ability to use an expensive printer; - access to a classified facility.

  no threat no asset!

- **Threats:** threats to assets, what can happen that endengers assets

- **Assumptions:** assumptions are acceptable, where certain properties of the TOE environment are already known or can be assumed

  this is NOT the place for putting properties derived from specific properties of the TOE

4. Security objectives

- "The security objectives are a concise and abstract statement of the intended solution to the problem defined by the security problem definition. Their role:

  - a high-level, natural language solution of the problem;

  - divide this solution into partwise solutions, each addressing a part of the problem;

  - demonstrate that these partwise solutions form a complete solution to the problem.

- bridge between the security problem and Security Functional Requirements (SFR)

- **mapping objectives to threats**: table, each threat shoud be covered, each objective has to respond to some threat

  answers to questions:

  - what is really needed?
  - have we forgot about something?

- **rationale:** verifiable explanation why the mapping is sound

5. Extended Component Definition

- In many cases the security requirements (see the next section) in an ST are based on components in CC Part 2 or CC Part 3.
- in some cases, there may be requirements in an ST that are not based on components in CC Part 2 or CC Part 3.
- in this case new components (extended components) need to be defined

6.1 **SFR (Security Functional requirements)**

- *The SFRs are a translation of the security objectives for the TOE. They are usually at a more detailed level of abstraction, but they have to be a complete translation (the security objectives must be completely addressed) and be independent of any specific technical solution (implementation). The CC requires this translation into a standardised language for several reasons: - to provide an exact description of what is to be evaluated. As security objectives for the TOE are usually formulated in natural language, translation into a standardised language enforces a more exact description of the functionality of the TOE. - to allow comparison between two STs. As different ST authors may use different terminology in describing their security objectives, the standardised language enforces using the same terminology and concepts. This allows easy comparison.*

- predefined classes:

  - Logging and audit class FAU

  - Identification and authentication class FIA

  - Cryptographic operation class FCS

  - Access control families FDP_ACC, FDP_ACF

  - Information flow control families FDP_IFC, FDP_IFF

  - Management functions class FMT

  - (Technical) protection of user data families FDP_RIP, FDP_ITT, FDP_ROL

  - (Technical) protection of TSF data class FPT

  - Protection of (user) data during communication with external entities families FDP_ETC, FDP_ITC, FDP_UCT, FDP_UIT, FDP_DAU, classes FCO and FTP

- There is no translation required in the CC for the security objectives for the operational environment, because the operational environment is not evaluated

- customizing SFRs: `refinement` (more requirements), `selection` (options), `assignment` (values), `iterations` (the same component may appear at different places with different roles)

- rules:

  check dependencies between SFR - In the CC Part 2 language, an SFR can have a dependency on other SFRs. This signifies that if an ST uses that SFR, it generally needs to use those other SFRs as well. This makes it much harder for the ST writer to overlook including necessary SFRs and thereby improves the completeness of the ST.

  security objectives must follow from SFR's - Security Requirements Rationale section (Sect.6.3) in PP

  if possible, use only standard SFR's

6.2 Security Assurance Requirements

- The SARs are a description of how the TOE is to be evaluated. This description uses a standardised language (to provide exact description, to allow comparison between two PP).

III. LEGAL FRAMEWORK - EXAMPLE: **EIDAS REGULATION**

goals:

- interoperability, comparable levels of trust
- merging national systems into pan-European one
- trust services, in particular: identification, authentication, signature, electronic seal, time-stamping, electronic delivery, Web authentication
- supervision system
- information about security breaches
- focused on public administration systems. However, the rules for all trust services except for closed systems (not available to anyone). Private sector encouraged to reuse the same means.

tools:

- common legal framework
- supervision system
- obligatory exchange of information about security problems
- common understanding of assurance levels

technical concept:

- each Member State provides an online system enabling identification and authentication with means from this Member State to be used abroad
- a notification scheme for national systems

- if notified (some formal and technical conditions must be fulfilled), then every member state must implement it in own country within 12 month

identification and authentication:

- eID cards – Member States are free to introduce any solution, the Regulation attempts to change it and build a common framework from a variety of (incompatible) solutions
- breakthrough claimed, but likely to fail

changes regarding electornic signature:

- electronic seal with the same conditions as electornic signature,
- the seal is aimed for legal persons
- weakening conditions for qualified electronic signatures: admitting server signatures and delegating usage of private keys

new:

- electronic registered delivery service
- Webpage authentication

Example of requirements (electronic seal):

Definition:

"electronic seal creation device" means configured software or hardware used to create an electronic seal;

"qualified electronic seal creation device" means an electronic seal creation device that meets mutatis mutandis the requirements laid down in Annex II;

Art. 36

An advanced electronic seal shall meet the following requirements:

(a) it is uniquely linked to the creator of the seal;

(b)it is capable of identifying the creator of the seal;

(c)it is created using electronic seal creation data that the creator of the seal can, with a high level of confidence under its control, use for electronic seal creation; and

(d) it is linked to the data to which it relates in such a way that any subsequent change in the data is detectable.

Annex II:

(a) the confidentiality of the electronic signature creation data used for electronic signature creation is reasonably assured;

(b) the electronic signature creation data used for electronic signature creation can practically occur only once;

(c) the electronic signature creation data used for electronic signature creation cannot, with reasonable assurance, be derived and the electronic signature is reliably protected against forgery using currently available technology;

(d) the electronic signature creation data used for electronic signature creation can be reliably protected by the legitimate signatory against use by others.

2. Qualified electronic signature creation devices shall not alter the data to be signed or prevent such data from being presented to the signatory prior to signing.

3. Generating or managing electronic signature creation data on behalf of the signatory may only be done by a qualified trust service provider.

4. Without prejudice to point (d) of point 1, qualified trust service providers managing electronic signature creation data on behalf of the signatory may duplicate the electronic signature creation data only for back-up purposes provided the following requirements are met:

(a) the security of the duplicated datasets must be at the same level as for the original datasets;

(b) the number of duplicated datasets shall not exceed the minimum needed to ensure continuity of the service.

Art. 30

1. Conformity of qualified electronic signature creation devices with the requirements laid down in Annex II shall be certified by appropriate public or private bodies designated by Member States.

notification system:

An electronic identification scheme eligible for notification if:

(a) issued by the notifying state

(b) at least one service available in this state;

(c) at least assurance level low;

(d) ensured that the person identification data is given to the right person

(e) ...

(f) availability of authentication online, for interaction with foreign systems (free of charge for public services), no specific disproportionate technical requirements

(g) description of that scheme published 6 months in advance

(h) meets the requirements from the implementing act

Assurance levels:

- regulation, Sept. 2015, implementation of eIDAS
- reliability and quality of
  - enrolment
  - electronic identification means management
  - authentication
  - management and organization
- authentication factors
  - posession based
  - knowledge based
  - inherent (physical properties)

- enrolment: (for all levels):

  1. Ensure the applicant is aware of the terms and conditions related to the use of the electronic identification means.

  2. Ensure the applicant is aware of recommended security precautions related to the electronic identification means.

  3. Collect the relevant identity data required for identity proofing and verification.

- identity proving and verification (for natural persons):

  low:

  1. The person can be assumed to be in possession of evidence recognised by the Member State in which the application for the electronic identity means is being made and representing the claimed identity.

  2. The evidence can be assumed to be genuine, or to exist according to an authoritative source and the evidence appears to be valid.

  3. It is known by an authoritative source that the claimed identity exists and it may be assumed that the person claiming the identity is one and the same.

  substantial: low plus:

  1. The person has been verified to be in possession of evidence recognised by the Member State in which the application for the electronic identity means is being made and reprensenting the claimed identity

  and

  the evidence is checked to determine that it is genuine; or, according to an authoritative source, it is known to exist and relates to a real person

  and

  steps have been taken to minimise the risk that the person's identity is not the claimed identity, taking into account for instance the risk of lost, stolen, suspended, revoked or expired evidence; or

  2. options related to other trustful sources

  high: substantial plus

  (a) Where the person has been verified to be in possession of photo or biometric identification evidence recognised by the Member State in which the application for the electronic identity means is being made and that evidence represents the claimed identity, the evidence is checked to determine that it is valid according to an authoritative source; and the applicant is identified as the claimed identity through comparison of one or more physical characteristic of the person with an authoritative source; or ...

- electornic identification means management:

  low:

  1. The electronic identification means utilises at least one authentication factor.

  2. The electronic identification means is designed so that the issuer takes reasonable steps to check that it is used only under the control or possession of the person to whom it belongs.

  substantial:

  1. The electronic identification means utilises at least two authentication factors from differ- ent categories.

2. The electronic identification means is designed so that it can be assumed to be used only if under the control or possession of the person to whom it belongs.

high:

1. The electronic identification means protects against duplication and tampering as well as against attackers with high attack potential

2. The electronic identification means is designed so that it can be reliably protected by the person to whom it belongs against use by others.

- Issuance , delivery and activation:

  low:

  After issuance, the electronic identification means is delivered via a mechanism by which it can be assumed to reach only the intended person.

  substantial:

  After issuance, the electronic identification means is delivered via a mechanism by which it can be assumed that it is delivered only into the possession of the person to whom it belongs.

  high:

  The activation process verifies that the electronic identification means was delivered only into the possession of the person to whom it belongs.

- suspencion, revocation and reactivation:

  all levels:

  1. It is possible to suspend and/or revoke an electronic identification means in a timely and effective manner.

  2. The existence of measures taken to prevent unauthorised suspension, revocation and/or reactivation.

  3. Reactivation shall take place only if the same assurance requirements as established before the suspension or revocation continue to be met.

- authentication mechanism:

  substantial:

  1. The release of person identification data is preceded by reliable verification of the electronic identification means and its validity.

  2. Where person identification data is stored as part of the authentication mechanism, that information is secured in order to protect against loss and against compromise, including analysis offline.

  3. The authentication mechanism implements security controls for the verification of the electronic identification means, so that it is highly unlikely that activities such as guessing, eavesdropping, replay or manipulation of communication by an attacker with enhanced-**basic attack potential** can subvert the authentication mechanisms.

  high:

  .... by an attacker with **high attack potentia**l can subvert the authentication mechanisms.

- audit:

  low:

  The existence of periodical internal audits scoped to include all parts relevant to the supply of the provided services to ensure compliance with relevant policy.

substantial:

The existence of periodical **independent** internal or external audits ....

high:

1. The existence of periodical i**ndependent external audits** scoped to include all parts relevant to the supply of the provided services to ensure compliance with relevant policy.

2. Where a scheme is directly managed by a government body, it is audited in accordance with the national law.

III. TECHNICAL GOVERNMENT REGULATIONS - EXAMPLE :

FIPS PUB 140-2, SECURITY REQUIREMENS FOR CRYPTOGRAPHIC MODULES

- Federal Information Processing Standards, NIST, recommendations and standards based on US law
- for sensitive but unclassified information
- levels: 1-4
- Cryptographic Module Validation Program (certification by NIST and Canadian authority)
- need to use "approved security functions" if to be used in public sector, waivers concerning some features are possible
- Levels:
  - Level 1: cryptographic module with at least one approved algorithm, no physical protection (like a PC)
  - Level 2:
    - tamper evident seals for access to CSP (critical security parameters)
    - role base authentication for operator,
    - refers to PPs, EAL2 or higher

    or secure operating system
  - Level 3:
    - protection against unauthorized access and attempts to modify cryptographic module, detection probability should be high,
    - CSP separated in a physical way from the rest
    - identity based authentication+ role based of an identified person (and not solely role based as on level 2)
    - CSP input and output - encrypted

- components of cryptographic module can be executed in a general purpose operating system if
    - PP fulfilled, Trusted Path fulfilled
    - EAL 3 or higher
    - security policy model (ADV.SPM1)
  - or a trusted operating system
- Level 4:
  - like level 3 but at least EAL4

- a more detailed overview:

	Security Level 1	*Security Level 2*	*Security Level 3*	*Security Level 4*
Cryptographic Module Specification	Specification of cryptographic module, cryptographic boundary, Approved algorithms, and Approved modes of operation. Description of cryptographic module, including all hardware, software, and firmware components. Statement of module security policy.			
Cryptographic Module Ports and Interfaces	Required and optional interfaces. Specification of all interfaces and of all input and output data paths.		Data ports for unprotected critical security parameters logically or physically separated from other data ports.	
Roles, Services, and Authentication	Logical separation of required and optional roles and services.	Role-based or identity-based operator authentication.	Identity-based operator authentication.	
Finite State Model	Specification of finite state model. Required states and optional states. State transition diagram and specification of state transitions.			
Physical Security	Production grade equipment.	Locks or tamper evidence.	Tamper detection and response for covers and doors.	Tamper detection and response envelope. EFP or EFT.
Operational Environment	Single operator. Executable code. Approved integrity technique.	Referenced PPs evaluated at EAL2 with specified discretionary access control mechanisms and auditing.	Referenced PPs plus trusted path evaluated at EAL3 plus security policy modeling.	Referenced PPs plus trusted path evaluated at EAL4.
Cryptographic Key Management	Key management mechanisms: random number and key generation, key establishment, key distribution, key entry/output, key storage, and key zeroization.			
	Secret and private keys established using manual methods may be entered or output in plaintext form.		Secret and private keys established using manual methods shall be entered or output encrypted or with split knowledge procedures.	
EMI/EMC	47 CFR FCC Part 15. Subpart B, Class A (Business use). Applicable FCC requirements (for radio).		47 CFR FCC Part 15. Subpart B, Class B (Home use).	
Self-Tests	Power-up tests: cryptographic algorithm tests, software/firmware integrity tests, critical functions tests. Conditional tests.			
Design Assurance	Configuration management (CM). Secure installation and generation. Design and policy correspondence. Guidance documents.	CM system. Secure distribution. Functional specification.	High-level language implementation.	Formal model. Detailed explanations (informal proofs). Preconditions and postconditions.
Mitigation of Other Attacks	Specification of mitigation of attacks for which no testable requirements are currently available.			

Table 1: *Summary of security requirements*

- more details:
  - roles: user, crypto officer, maintenance
  - services: to operator: show status, perform self-tests, perform approved security function, bypassing cryptographic operations must be documented etc.

– auhentication: pbb of a random guess $<\frac{1}{1000000}$, one minute attemps: $<\frac{1}{100000}$, feedback obscured

– physical security:

  – full documentation,

  – if maintenance functionalities, then many features including erasing the key when accessed

  – protected holes, you cannot put probing devices through the holes

  – level 4: environmental failure protection (EFP) features or undergo environmental failure testing (EFT) – prevent leakage through unusual conditions

	General Requirements for all Embodiments	Single-Chip Cryptographic Modules	Multiple-Chip Embedded Cryptographic Modules	Multiple-Chip Standalone Cryptographic Modules
Security Level 1	Production-grade components (with standard passivation).	No additional requirements.	If applicable, production-grade enclosure or removable cover.	Production-grade enclosure.
Security Level 2	Evidence of tampering (e.g., cover, enclosure, or seal).	Opaque tamper-evident coating on chip or enclosure.	Opaque tamper-evident encapsulating material or enclosure with tamper-evident seals or pick-resistant locks for doors or removable covers.	Opaque enclosure with tamper-evident seals or pick-resistant locks for doors or removable covers.
Security Level 3	Automatic zeroization when accessing the maintenance access interface. Tamper response and zeroization circuitry. Protected vents.	Hard opaque tamper-evident coating on chip or strong removal-resistant and penetration resistant enclosure.	Hard opaque potting material encapsulation of multiple chip circuitry embodiment or applicable Multiple-Chip Standalone Security Level 3 requirements.	Hard opaque potting material encapsulation of multiple chip circuitry embodiment or strong enclosure with removal/penetration attempts causing serious damage.
Security Level 4	EFP or EFT for temperature and voltage.	Hard opaque removal-resistant coating on chip.	Tamper detection envelope with tamper response and zeroization circuitry.	Tamper detection/ response envelope with tamper response and zeroization circuitry.

Table 2: *Summary of physical security requirements*

• more details:

  – operational environment:

    – L1: separation of processes, concurrent operators excluded, no interrupting cryptographic module,Approved integrity technique (HMAC?)

    – L2: operating system control functions under EAL2, specify roles to operate, modify,..., crypto software withing cryptographic boundary, audit: recording invalid operatons, capable of auditing the following events:

      operations to process audit data from the audit trail,

      requests to use authentication data management mechanisms,

        - use of a security-relevant crypto officer function,
        - requests to access user authentication data associated with the cryptographic module,
        - use of an authentication mechanism (e.g., login) associated with the cryptographic module,
        - explicit requests to assume a crypto officer role,
        - the allocation of a function to a crypto officer role.

    - L3: EAL3, trusted path (also included in audit trail)
    - L4: EAL4

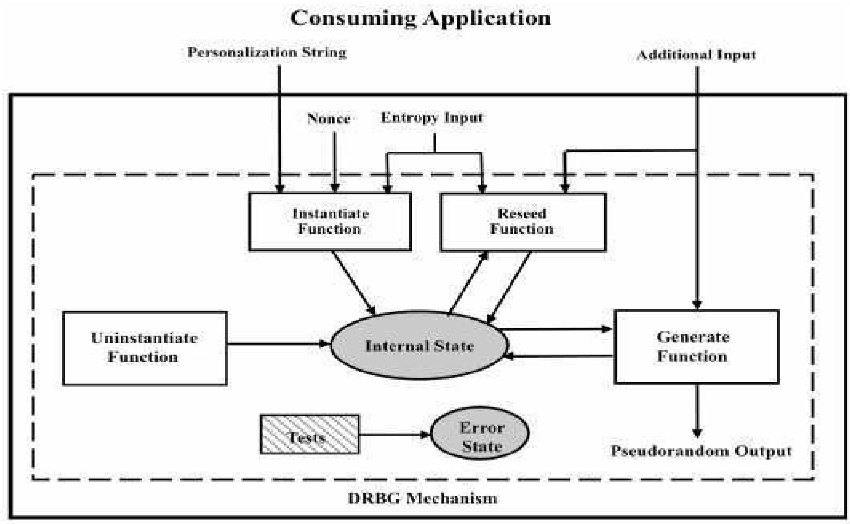
- key management:
    - non-approved RNG can be used for IV or as input to approved RNG
    - list of approved RNG: refers to an annex and annex to NIST document from 2016 (with a link to 2015)
    - list of approved key establishment - again links
    - key in out: automated (encrypted) or manual (splitted in L3 or L4)

- tests: self-test and power-up. No crypto operation if something wrong. tests based on known outputs
    - Pair-wise consistency test (for public and private keys).
    - Software/firmware load test.
    - Manual key entry test.
    - Continuous random number generator test.
    - Bypass test – proper switching between bypass and crypto

---

FIPS Approved Random Number Generators

- nondeterministic generators not approved
- deterministic: special NIST Recommendation,
- first approved entropy source creates a seed , then deterministic part



Instantiation:

- seed has a limited period
- reseeded function requires a different seed
- different instantiations can exist at the same time, they MUST be independent in terms of the seeds and usage

Internal state:

- contains cryptographic chain value AND the number of requests so far
- different instantiations of DRBG must have separate internal states

Instantiation strength:

- frmally defined as "112, 128, 192, 256 bits", intuition: number of bits to be guessed
- `Security_strength_of_output` = min(`output_length`, `DRBG_security_strength`)

functions:

- instantiate: initializing the internal state, prepering to use
- generate: generating output bits as PRNG
- reseed: combines the internal state with new entropy to change the seed
- uninstantiate: erase the internal state
- test: checks correctnes of components on the chip

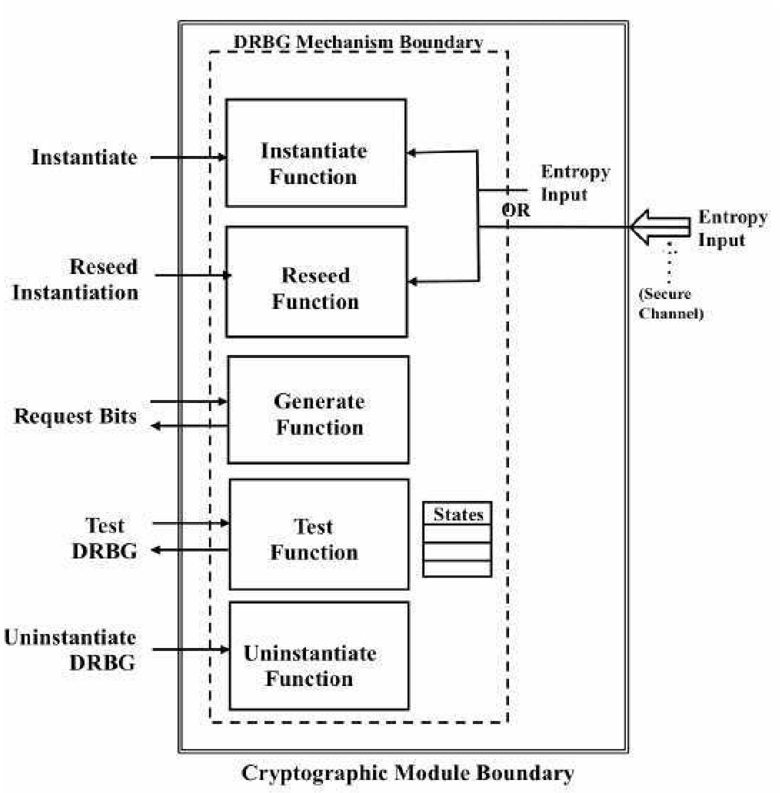
DRBG mechanism boundary:

- not cryptographic module boundary

– DRBG internal state and operation shall only be affected according to the DRBG mechanism specification

– the state exists solely within the DRBG mechanism boundary, not -accessible from outside

– information about internal state only via specified output



Seed:

– entropy is obligatory, entropy strength should be at least the entropy of the output

– reseeding: nonce not used, the internal state used

– *approved randomness source* obligatory for entropy source

– nonce: not secret. Example nonces:

  – a random value from an approved generator

  – a trusted timestamp of sufficient resolution (never use the same timestamp)

  – monotonically increasing sequence number

  – combination of a timestamp and a monotonically increasing sequence number, such that the sequence number is reset when and only when the timestamp change

– not used for any other purposes

reseed:

- ”for security”
- argument: it might be better than `uninstantiate` and `instantiate` due to aging of randomness source

personalization:

- not security critical, but the adversary might be unaware of it (analogous to a login)

resistance:

- backtracking resistance: given internal state at time t it is infeasible to distinguish between the output for period $[1, t-1]$ and a random output
- prediction resistance: *“Prediction resistance means that a compromise of the DRBG internal state has no effect on the security of future DRBG outputs. That is, an adversary who is given access to all of the output sequence after the compromise cannot distinguish it from random output with less work than is associated with the security strength of the instantiation; if the adversary knows only part of the future output sequence, he cannot predict any bit of that future output sequence that he does not already know (with better than a 50-50 chance).* – **refers only to reseeding** (before reseeding the output is predictable)

  distinguishability from random input or predicting missing output bits

specific functions:

- (status, entropy_input) = Get_entropy_input (min_entropy, min_ length, max_

  length, prediction_resistance_request),
- Instantiation:
  - → checks validity of parameters
  - → determines security strength
  - → obtains entropy input, nonce
  - → runs instantiate algorithm to get initial state
  - → returns a handle

  Instantiate_function(requested_instantiation_security_strength, prediction_resistance_flag, personalization_string)

  prediction_resistance_flag determines whether consuming application may request reseeding
- Reseed:
  - → Explicit request by a consuming application,
  - → if prediction resistance is requested
  - → also if the upper bound on the number of genereted outpus reached
  - → also due to external events

    steps:

→ checks validity of the input parameters,

→ determines the security strength

→ obtains entropy input, nonce

→ runs reseed algorithm to get initial state

- Generate function:
  Generate_function(state_handle,requested_number_of_bits,requested_security_strength, prediction_resistance_request, additional_input)

- Removing a DRBG Instantiation:
  Uninstantiate_function (state_handle)
  internal state zeroized

Hash_DRBG

variants:

– hash algorithms: SHA-1 up tp SHA-512

– parameters determined, e.g. maximum length of personalization string

– seed length typically 440 (but also 888)

state:

→ value V updated during each call to the DRBG

→ constant C that depends on the seed

→ counter `reseed_counter`: storing the number of requests for pseudorandom bits since new entropy_input was obtained during instantiation or reseeding

instantiation:

1. `seed_material` = `entropy_input || nonce || personalization_string`
2. `seed` = `Hash_df (seed_material, seedlen)` (hash derivation function)
3. `V = seed`
4. `C = Hash_df ((0x00 || V), seedlen)`
5. Return `(V, C, reseed_counter)`

reseed:

1. `seed_material = 0x01 || V || entropy_input || additional_input`
2. `seed = Hash_df (seed_material, seedlen)`
3. `V = seed`

4. C = Hash_df ((0x00 || V), seedlen)

5. reseed_counter = 1

6. Return (V, C, reseed_counter).

generating bits:

1. If reseed_counter > reseed_interval, then return "reseed required"

2. If (additional_input $\neq$ Null), then do

   2.1 w = Hash (0x02 || V || additional_input)

   2.2 V = (V + w) mod 2^{seedlen}

3. (returned_bits) = Hashgen (requested_number_of_bits, V)

4. H = Hash (0x03 || V)

5. V = (V + H + C + reseed_counter) mod 2^{seedlen}

6. reseed_counter = reseed_counter + 1

7. Return (SUCCESS, returned_bits, V, C, reseed_counter)

Hashgen:

1. m $= \frac{\text{requested} - \text{no} - \text{of} - \text{bits}}{\text{outlen}}$

2. data = V

3. W = Null string

4. For i = 1 to m

   4.1 w = Hash (data).

   4.2 W = W || w

   4.3 data = (data + 1) mod 2seedlen

5. returned_bits = leftmost (W, requested_no_of_bits)

6. Return (returned_bits).

HMAC_DRBG

Update (used for instantiation and reseeding)

1. K = HMAC (K, V || 0x00 || provided_data)

2. V = HMAC (K, V)

3. If (provided_data = Null), then return K and V

4. K = HMAC (K, V || 0x01 || provided_data)

5. V = HMAC (K, V)

6. Return (K, V).

Instantiate:

1. seed_material = entropy_input || nonce || personalization_string
2. Key = 0x00 00...00
3. V = 0x01 01...01
4. (Key, V) = HMAC_DRBG_Update (seed_material, Key, V)
5. reseed_counter = 1
6. Return (V, Key, reseed_counter)

Reseed:

1. seed_material = entropy_input || additional_input
2. (Key, V) = HMAC_DRBG_Update (seed_material, Key, V)
3. reseed_counter = 1
4. Return (V, Key, reseed_counter).

Generate bits:

1. If reseed_counter > reseed_interval, then return "reseed required"
2. If additional_input $\neq$ Null, then
   (Key, V) = HMAC_DRBG_Update (additional_input, Key, V)
3. temp = Null
4. While len (temp) < requested_number_of_bits do:
   4.1 V = HMAC (Key, V)
   4.2 temp = temp || V
5. returned_bits = leftmost (temp, requested_number_of_bits)
6. (Key, V) = HMAC_DRBG_Update (additional_input, Key, V)
7. reseed_counter = reseed_counter + 1
8. Return (SUCCESS, returned_bits, Key, V, reseed_counter).

CTR_DRBG

a generator based on an encryption function, AES versions

internal state:

- value V of blocklen bits, updated each time another blocklen bits of output are produced

– keylen-bit Key, updated whenever a predetermined number of output blocks are generated

– counter (reseed_counter) = the number of requests for pseudorandom bits since instantiation or reseeding

– ctr_len is a parameter known by implementation

Update Process:

1. temp = Null

2. While (len (temp)< seedlen, do

   2.1 If ctr_len < blocklen

      2.1.1 inc = (rightmost (V, ctr_len) + 1) mod 2^{ctrlen}.

      2.1.2 V = leftmost (V, blocklen-ctr_len) || inc

   Else V = (V+1) mod 2^{blocklen}

   2.2 output_block = Block_Encrypt (Key, V)

   2.3 temp = temp || output_block

3. temp = leftmost (temp, seedlen)

4. temp = temp ⊕ provided_data

5. Key = leftmost (temp, keylen)

6. V = rightmost (temp, blocklen).

Instantiate:

1. pad personalization_string with zeroes

2. seed_material = entropy_input ⊕ personalization_string

3. Key $= 0^{\text{keylen}}$

4. V $= 0^{\text{blocklen}}$

5. (Key, V) = CTR_DRBG_Update (seed_material, Key, V).

6. reseed_counter = 1

7. Return (V, Key, reseed_counter).

reseeding is similar

Generate:

1. If reseed_counter > reseed_interval, then "reseed required"

2. If (additional_input≠ Null), then

   2.1 temp = len (additional_input).

2.2 If (temp< seedlem) then pad additional_input with zeroes

2.3 (Key, V) = CTR_DRBG_Update (additional_input, Key, V).

Else additional_input = $0^{seedlen}$

3. temp = Null

4. While (len (temp)<requested_number_of_bit), do

   4.1 If ctr_len< blocklen

      4.1.1 inc = (rightmost (V, ctr_len) + 1) mod 2^{ctrlen}

      4.1.2 V = leftmost (V, blocklen-ctr_len) || inc

   Else V = (V+1) mod $2^{blocklen}$

   4.2 output_block = Block_Encrypt (Key, V).

   4.3 temp = temp || output_block

5. returned_bits = leftmost (temp, requested_number_of_bits)

6. (Key, V) = CTR_DRBG_Update (additional_input, Key, V)

7. reseed_counter = reseed_counter + 1

8. Return (SUCCESS, returned_bits, Key, V, reseed_counter).

IV. GDPR

TEXT TO COME HERE

DOMAIN SIGNATURES

Definition of Domain Signatures

TEXT TO COME HERE

Domain signatures and applications

1. replacing login+ password mechanisms. Like Restricted Identification from German Personal Identity cards. Advantages

   1. one PIN per person

   2. high entropy, not predictable

3. no problem for

2. anonymization in datasets

3. recommendations,

4. ”randomness” and self-organization

Pseudonymous Signature:

- BSI standard
- keys:
  - domain parameters D_M and a pair of global keys $(\mathrm{PK}_M, \mathrm{SK}_M)$
  - public key $\mathrm{PK}_{\mathrm{ICC}}$ for a group of eIDAS tokens, the private key $\mathrm{SK}_{\mathrm{ICC}}$ known to the issuer of eIDAS tokens (called manager)
  - for a token the manager chooses $\mathrm{SK}_{\mathrm{ICC},2}$ at random, then computes $\mathrm{SK}_{\mathrm{ICC},1}$ such that $\mathrm{SK}_{\mathrm{ICC}} = \mathrm{SK}_{\mathrm{ICC},1} + \mathrm{SK}_M \cdot \mathrm{SK}_{\mathrm{ICC},2}$
  - a sector (domain) holds private key $\mathrm{SK}_{\mathrm{sector}}$ and public key $\mathrm{PK}_{\mathrm{sector}}$.
  - a sector has revocation private key $\mathrm{SK}_{\mathrm{revocation}}$ and public key $\mathrm{PK}_{\mathrm{revocation}}$
  - sector specific identifiers $I^{\mathrm{sector}}_{\mathrm{ICC},1}$ and $I^{\mathrm{sector}}_{\mathrm{ICC},2}$ of the eIDAS token in the sector: $I^{\mathrm{sector}}_{\mathrm{ICC},1} = (\mathrm{PK}_{\mathrm{sector}})^{\mathrm{SK}_{\mathrm{ICC},1}}$ and $I^{\mathrm{sector}}_{\mathrm{ICC},2} = (\mathrm{PK}_{\mathrm{sector}})^{\mathrm{SK}_{\mathrm{ICC},2}}$
- signing: with keys $\mathrm{SK}_{\mathrm{ICC},1}$, $\mathrm{SK}_{\mathrm{ICC},2}$ and $I^{\mathrm{sector}}_{\mathrm{ICC},1}$ and $I^{\mathrm{sector}}_{\mathrm{ICC},2}$ for $\mathrm{PK}_{\mathrm{sector}}$ and message m

  i. choose K_1, K_2 at random

  ii. compute
  - $Q_1 = g^{K_1} \cdot (\mathrm{PK}_M)^{K_2}$
  - $A_1 = (\mathrm{PK}_{\mathrm{sector}})^{K_1}$
  - $A_2 = (\mathrm{PK}_{\mathrm{sector}})^{K_2}$

  iii. $c = \mathrm{Hash}(Q_1, I^{\mathrm{sector}}_{\mathrm{ICC},1}, A_1, I^{\mathrm{sector}}_{\mathrm{ICC},2}, A_2, \mathrm{PK}_{\mathrm{sector}}, m)$

  (variant parameters and $\mathbf{\Pi}$ omitted here)

  iv. compute
  - $s_1 = K_1 - c \cdot \mathrm{SK}_{\mathrm{ICC},1}$
  - $s_1 = K_2 - c \cdot \mathrm{SK}_{\mathrm{ICC},2}$

  v. output (c, s_1, s_2)
- verification:

  compute
  - $Q_1 = (\mathrm{PK}_{\mathrm{ICC}})^c \cdot g^{s_1} \cdot (\mathrm{PK}_M)^{s_2}$

- $A_1 = (I_{\mathrm{ICC},1}^{\mathrm{sector}})^c \cdot (\mathrm{PK}_{\mathrm{sector}})^{s_1}$
- $A_2 = (I_{\mathrm{ICC},2}^{\mathrm{sector}})^c \cdot (\mathrm{PK}_{\mathrm{sector}})^{s_2}$
- recompute c and check against the c from the signature

- why it works?

$$\begin{aligned}(\mathrm{PK}_{\mathrm{ICC}})^c \cdot g^{s_1} \cdot (\mathrm{PK}_M)^{s_2} &= (\mathrm{PK}_{\mathrm{ICC}})^c \cdot g^{K_1} \cdot (\mathrm{PK}_M)^{K_2} \cdot g^{-c \cdot \mathrm{SK}_{\mathrm{ICC},1}} \cdot (\mathrm{PK}_M)^{c \cdot \mathrm{SK}_{\mathrm{ICC},2}} \\ &= (\mathrm{PK}_{\mathrm{ICC}})^c \cdot g^{K_1} \cdot (\mathrm{PK}_M)^{K_2} \cdot g^{-c \cdot \mathrm{SK}_{\mathrm{ICC},1}} \cdot (g)^{-c \cdot \mathrm{SK}_M \cdot \mathrm{SK}_{\mathrm{ICC},2}} \\ &= (\mathrm{PK}_{\mathrm{ICC}})^c \cdot g^{K_1} \cdot (\mathrm{PK}_M)^{K_2} \cdot g^{-c \cdot \mathrm{SK}_{\mathrm{ICC}}} = g^{K_1} \cdot (\mathrm{PK}_M)^{K_2} = Q_1\end{aligned}$$

- there is a version without A_1, A_2 and the pseudonyms $I_{\mathrm{ICC},1}^{\mathrm{sector}}, I_{\mathrm{ICC},2}^{\mathrm{sector}}$
- Problems:
  - the authorities know the private keys (there is a way to solve it when the user gets two pairs of keys on the device and takes their linear combination)
  - breaking into just 2 devices reveals the system keys

V. STANDARS VERSUS SECURITY

It is not true that a standard solution is by definition a secure solution.

Standardisation process:

- representatives of countries, not necessarily specialists
- represent interests of industry
- target: a unified solution
- no open evaluation as in case of NIST competitions
- long process, many standards never used in practice

result: no guarantee for security

Example: Bleichenbacher's RSA signature forgery based on implementation error

- background: one has to padd the hash value before applying RSA signing operation, after padding the input no more a small number

- PKCS#1: RSA Laboratories standard for formats of RSA signatures (ceuurently 1.5 and OEAP (optimal Asymmetric encryption padding)
- The attack works for PKCS-1.5 padding:
  - a byte 0
  - a byte 1
  - string of 0xFF bytes (their number depends on RSA modulus and the rest)
  - a byte 0
  - ASN.1 data
  - hash

    00 01 FF FF FF ... FF 00 ASN.1 HASH
- RSA signature verification: exponentiation with the public key, remove padding, check the hash
- implementation based on the standard: recognize the structure 00 01 FF FF FF ... FF 00 and after them parse the hash
- attack mechanism:
  - hash not right adjusted (padding short), after the hash there is a part that is not parsed (it could be anything)
  - concern RSA systems with public key 3 (sometimes it is done so to speed-up verification) – according to strong RSA assumption computing roots of degree 3 is generally infeasible without the secret key
  - part after the hash adjusted so that the resulting number is a cube as an integer
  - compute the root ... and this is the signature!
- attack variants: some fields declared but not checked. then Bleichenbacher's freedom to adjust the number to become a cube even if the hash is right justified

Chosen Ciphertext Attacks Against Protocols Based on RSA Encryption Standard PKCS-1 – the Million Message Attack.

- RSA decryption device, returns an error message if the ciphertext not in PKCS-1 format (HSM,...)
- the ciphertext c to be broken is manipulated many times and based on error messages we narrow the set of choices for the plaintext
- attack (find m such that $m^d = c \bmod n$:
  1. phase: blind the ciphertext: $c_0 := c \cdot s^e \bmod n$ by choosing s such that c_0 is a valid PKCS-1 ciphertext.
     - Observe $c_0^e = m \cdot s$ and it starts with msb: 00, 02,
     - let k the byte length of n, $B = 2^{8(k-2)}$

– then $2B \leq m \cdot s < 3B$

– let $M_0 = [2B, 3B - 1]$

2. phase: narrowing the set of intervals defining $s_1 < s_2 < \ldots$ and $M_1, M_2, \ldots$ such that $M_{i+1} \subset M_i$, each M_i is a set of intervals

   ◦ if M_{i-1} is a single interval $[a, b]$ then choose small values s_i and r_i such that

     $\frac{2B + r_i n}{b} < s_i < \frac{3B + r_i n}{a}$

     and $c_0 \cdot s_i^e$ is PKCS-1 conforming

   ◦ if M_{i-1} is not a single interval, then simply smallest s_i such that $c_0 \cdot s_i^e$ is PKCS-1 conforming

   ◦ M_i consists of all intervals

     $[\max\left(a, \frac{2B + r_i \cdot n}{s_i}\right), \min\left(b, \frac{3B - 1 + r_i \cdot n}{s_i}\right)]$ for $[a, b]$ from M_{i-1} and

     $\frac{a \cdot s_i - 3B + 1}{n} \leq r \leq \frac{b \cdot s_i - 2B}{n}$

   ◦ when eventually $M_i = [a, a]$ then set $m = a \cdot s_0^{-1} \bmod n$

The attack exploits such vulnerabilities like

- error messages returned by the attacked device when decryption fails on different stages of the decryption algorithm
- different timings of execution of the decryption algorithm when the PKCS-1 encryption padding is correct and when it is incorrect.

If a device supports the PKCS-1 encryption padding and the implementation of the PKCS-1 decryption on the device is vulnerable, then the million message attack works also when

- the ciphertext is calculated according to a padding different than PKCS-1
- the “ciphertext” is the plaintext for which we want to obtain a signature (dangerous for a situation when the same key is used for decryption and for signatures, and decryption is not PIN protected).

VI. RFC DOCUMENTS

RFC

”Request for Comments”

- by Internet Engineering Task Force (IETF) and the Internet Society
- semi-standard, developed from rfc from ARPANET
- authors of RFC versus standards with commitees

- peer review, some reach status of "Internet Standards"
- RFC editor provided
- streams:
  - Internet Engineering Task Force (IETF) - current issues
    - BCP Best Current Practice;
    - FYI For Your Information; informational
    - STD Standard: with 2 maturity levels
  - Internet Research Task Force (IRTF) - more long term issues
  - Internet Architecture Board (IAB) (a body over task forces)
  - independent
- Status:
  - informational
  - experimental
  - best current practice
  - standard: Proposed Standard, Draft Standard, Internet Standard

RFC as an example of specification of a protocol

A. Exemplary RFC: draft of TLS 1.3 spec.

B. Required level of detail - ensuring unambiguous implementation.

C. Structure of the document: from general view to detailed description - to facilitate reading many datastructures and technicalities are shifted to the appendices.

- Abstract: What the document is about?
- Status: Internet draft, expiration date
- Copyright notice
- Table of contents:
  1. Introduction
     - a very nice note for RFC editors (present in the draft only)
     - the goal of the protocol (authentication, confidentiality, integrity + definitions of the three terms, to let the non-cryptographers understand the document)
     - the high level view - primary components

- conventions and terminology - for clear and precise understanding
- change log - for editors (present in the draft only)
- major differences from previous version of the protocol (TLS 1.2)

2. Protocol Overview. Handshake: what must be negotiated, what are the basic key exchange modes?

3. Presentation language: Big or little endian? basic block size? etc.

4.-9. Protocol components, further details.

10. Security considerations

D. Exemplary protocol detail: certificate request from server side (dnames of CAs) cross-certification.

EXAMPLE: RFC2560

Network Working Group

Category: Standards Track

authors ... June 1999

title: *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This document specifies a protocol useful in determining the current status of a digital certificate without requiring CRLs. Additional mechanisms addressing PKIX operational requirements are specified in separate documents.

... contents of sections

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document (in uppercase, as shown) are to be interpreted as described in [RFC2119].

–

MUST=REQUIRED=SHALL: an absolute requirement

MUST NOT=SHALL NOT: an absolute prohibition of the specification

SHOULD=RECOMMENDED: ‘*‘there may exist valid reasons in particular circumstances to ignore, but implications must be understood and carefully weighed before choosing a different course*”

SHOULD NOT=NOT RECOMMENDED: negation of SHOULD (think twice before implementing it in this way!)

MAY=OPTIONAL: real option, **but** implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option,

2. Protocol Overview

- supplement to periodical checking CRL
- enables to determine the state of an identified certificate
- more timely, with more information
- RFC defines data exchanged

2.1 Request

– protocol version – service request – target certificate identifier – optional extensions which MAY be processed by the OCSP Responder

OCSP Responder checks:

1. request well formed
2. responder configured to serve such request
3. all necessary data given in the request

otherwise: error message

2.2 Response

- type+actual response
- basic type MUST be supported
- *"All definitive response messages SHALL be digitally signed."*
- signer MUST be one of: CA who issued the certificate, or a Trusted Responder of the requester, CA Designated Responder (Authorized Responder) - agent of CA with a certificate from CA
- response message: version of the response syntax – name of the responder – responses for each of the certificates in a request – optional extensions – signature algorithm OID – signature computed across hash of the response
- for each target certificate: certificate status value – response validity interval – optional extensions
- values:
  - good: *"At a minimum, this positive response indicates that the certificate is not revoked, but does not necessarily mean that the certificate was ever issued or that the time at which the response was produced is within the certificate's validity interval. Response extensions may be used to convey additional information on assertions made by the responder regarding the status of the certificate such as positive statement about issuance, validity, etc."*

- revoked: the certificate has been revoked (permanantly or temporarily (on hold))
- unknown: responder has no data

2.3 Exception Cases

- error messages not signed
- types: – malformedRequest – internalError – tryLater – sigRequired – unauthorized
- "internalError" = responder reached an inconsistent internal state. The query should be retried
- "tryLater" = temporarily unable to respond
- "sigRequired"= the server requires the client sign
- "unauthorized"=the client is not authorized to make this query

2.4 Semantics of thisUpdate, nextUpdate and producedAt

- thisUpdate = time at which the indicated status is known to be correct
- nextUpdate= time at or before which newer information will be available about the certificate status
- producedAt = time at which the OCSP signed this response.

2.5 Response Pre-production

''*OCSP responders MAY pre-produce signed responses specifying the status of certificates at a specified time. The time at which the status was known to be correct SHALL be reflected in the thisUpdate field of the response. The time at or before which newer information will be available is reflected in the nextUpdate field, while the time at which the response was produced will appear in the producedAt field of the response.*"

- means that OCSP is not checking the status of the certificate but status on the CRL!

2.6 OCSP Signature Authority Delegation

- the OCSP might be an agent of CA explicitely apointed,
- signing key must allow signing it

2.7 CA Key Compromise

- if CA's private key compromised, then OCSP MAY return the revoked state for all certificates issued by that CA.

3. Functional Requirements

3.1 Certificate Content

- CAs SHALL provide the capability to include the AuthorityInfoAccess extension in certificates that can be checked using OCSP

- accessLocation for the OCSP provider may be configured locally at the OCSP client
- CAs supporting OCSP MUST "*provide for the inclusion of a value for a uniformResourceIndicator (URI) accessLocation and the OID value id-ad-ocsp for the accessMethod in the AccessDescription SEQUENCE*"
- accessLocation field in the subject certificate defines the transport (e.g. HTTP) used to access OCSP responder and data (e.g. a URL)

3.2 Signed Response Acceptance Requirements

Before accepting response clients SHALL confirm that:

1. certificate in response=certificate asked
2. signature valid
3. signature of the responder
4. responder authorized
5. thisUpdate sufficiently recent
6. nextUpdate is greater than the current time

4. Detailed Protocol

- data to be signed encoded using ASN.1 distinguished encoding rules (DER)
- ASN.1 EXPLICIT tagging as a default
- "terms imported from elsewhere are: Extensions, CertificateSerialNumber, SubjectPublicKeyInfo, Name, AlgorithmIdentifier, CRLReason"

4.1 Requests

4.1.1 Request Syntax

OCSPRequest ::= SEQUENCE { tbsRequest TBSRequest, optionalSignature [0] EXPLICIT Signature OPTIONAL }

TBSRequest ::= SEQUENCE {version [0] EXPLICIT Version DEFAULT v1, requestorName [1] EXPLICIT GeneralName OPTIONAL, requestList SEQUENCE OF Request, requestExtensions [2] EXPLICIT Extensions OPTIONAL }

Signature ::= SEQUENCE { signatureAlgorithm AlgorithmIdentifier, signature BIT STRING, certs [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL}

Version ::= INTEGER { v1(0) }

Request ::= SEQUENCE { reqCert CertID, singleRequestExtensions [0] EXPLICIT Extensions OPTIONAL }

CertID ::= SEQUENCE { hashAlgorithm AlgorithmIdentifier, issuerNameHash OCTET STRING, – Hash of Issuer's DN issuerKeyHash OCTET STRING, – Hash of Issuers public key serialNumber CertificateSerialNumber }

- public key hashed together with name (names may repeat, public key must not)
- Support for any specific extension is OPTIONAL

- ”Unrecognized extensions MUST be ignored (unless they have the critical flag set and are not understood)”.

- requestor MAY sign the OCSP request, data included for easy verification (name:SHALL, certificate: MAY)

4.2 Response Syntax

OCSPResponse ::= SEQUENCE { responseStatus OCSPResponseStatus, responseBytes [0] EXPLICIT ResponseBytes OPTIONAL }

OCSPResponseStatus ::= ENUMERATED { successful (0), –Response has valid confirmations malformedRequest (1), –Illegal confirmation request internalError (2), –Internal error in issuer tryLater (3), –Try again later –(4) is not used sigRequired (5), –Must sign the request unauthorized (6) –Request unauthorized }

The value for responseBytes consists of an OBJECT IDENTIFIER and a response syntax identified by that OID encoded as an OCTET STRING.

ResponseBytes ::= SEQUENCE { responseType OBJECT IDENTIFIER, response OCTET STRING }

For a basic OCSP responder, responseType will be id-pkix-ocsp-basic.

id-pkix-ocsp OBJECT IDENTIFIER ::= { id-ad-ocsp } id-pkix-ocsp-basic OBJECT IDENTIFIER ::= { id-pkix-ocsp 1 }

4.3 Mandatory and Optional Cryptographic Algorithms

- clients SHALL: DSA sig-alg-oid specified in section 7.2.2 of [RFC2459]

- clients SHOULD: RSA signatures as specified in section 7.2.1 of [RFC2459]

- responders SHALL: SHA1

4.4 Extensions

4.4.1 Nonce

nonce against replay:

- nonce as one of the requestExtensions in requests

- in responses it would be included as one of the responseExtensions

- object identifier id-pkix-ocsp-nonce

4.4.2 CRL References

if revoked then indicate CRL where revoked

id-pkix-ocsp-crl OBJECT IDENTIFIER ::= { id-pkix-ocsp 3 }

CrlID ::= SEQUENCE { crlUrl [0] EXPLICIT IA5String OPTIONAL, crlNum [1] EXPLICIT INTEGER OPTIONAL, crlTime [2] EXPLICIT GeneralizedTime OPTIONAL }

For the choice crlUrl, the IA5String will specify the URL at which the CRL is available. For crlNum, the INTEGER will specify the value of the CRL number extension of the relevant CRL. For crlTime, the GeneralizedTime will indicate the time at which the relevant CRL was issued.

4.4.3 Acceptable Response Types

d-pkix-ocsp-response OBJECT IDENTIFIER ::= { id-pkix-ocsp 4 }

AcceptableResponses ::= SEQUENCE OF OBJECT IDENTIFIER

4.4.4 Archive Cutoff

- specifies how many years after expiration the revocation inforamation is retained, this si"archive cutoff" date

4.4.5 CRL Entry Extensions

All the extensions specified as CRL Entry Extensions - in Section 5.3 of [RFC2459] - are also supported as singleExtensions.

4.4.6 Service Locator

OCSP server receives a request and reroutes it to another OCSP

serviceLocator request extension used

d-pkix-ocsp-service-locator OBJECT IDENTIFIER ::= { id-pkix-ocsp 7 }

ServiceLocator ::= SEQUENCE { issuer Name, locator AuthorityInfoAccessSyntax OPTIONAL }

Values defined in certificate asked

5. Security Considerations

- flood of queries,
- signed and unsigned both enable DOS
- precomputation helps
- HTTP caching might be risky: "*Implementors are advised to take the reliability of HTTP cache mechanisms into account when deploying OCSP over HTTP*."

VII. HARDWARE TROJANS

methods of testing:

- functional tests
- internal tests circuitry
- optical inspection (distructive) - can detect modifications on layout level

Idea: change properties that are not visible under microscope: increase aging effects, manipulate transistors so that the output is fixed

Dopant Trojans

CMOS inverter: (image Wikipedia)

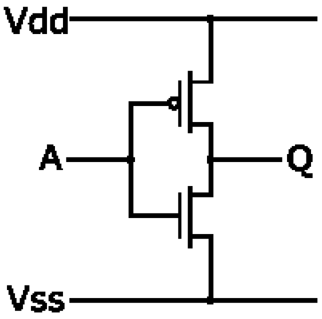


Figure 1.

where: A is the source, Vdd positive supply , Vss is ground

upper transistor: PMOS (allows current flow at low voltage)

lower transistor: NMOS (allows current flow at high voltage)

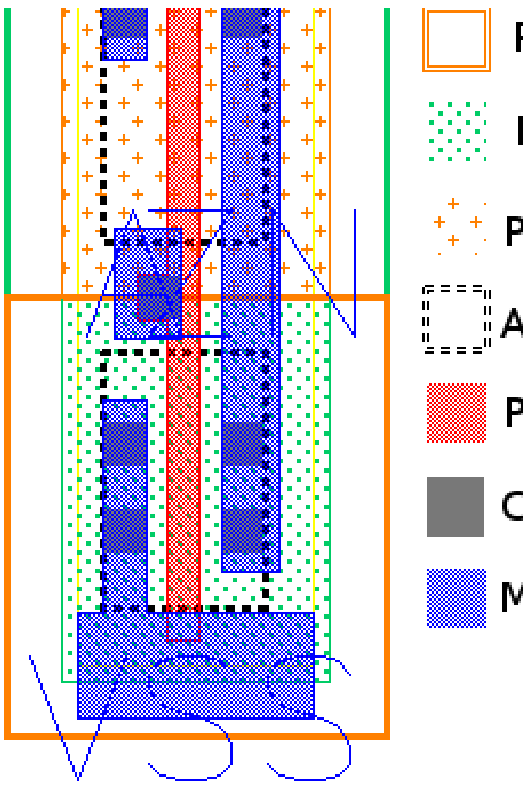
how it works:

- if voltage is low then the lower transistor (NMOS is in high resistance state and the current from Q flows to Vdd (high voltge)
- if voltage is high then the upper transistor MOS) is in high resistance state and the current from Q flows to Vss while Vdd has low voltage

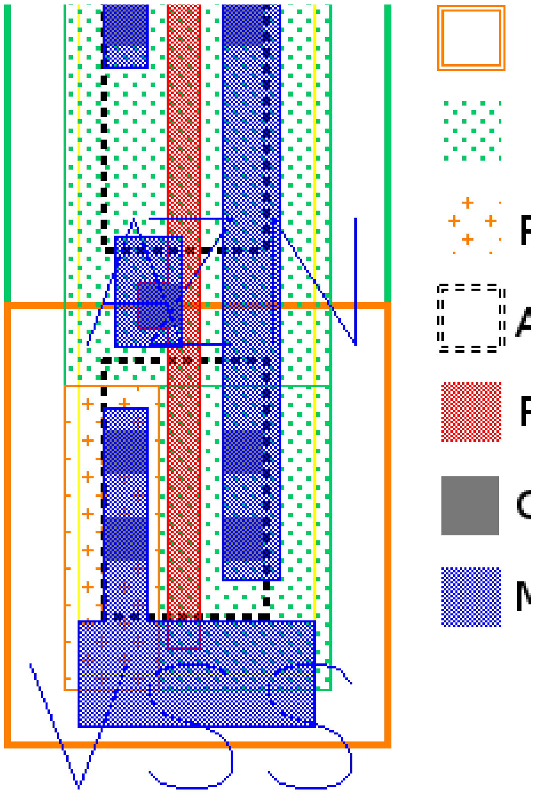
PMOS: in dopant area "holes" (positive) playing the role of conductor, low voltage creates depletion area, high voltage attracts them

NMOS: in dopant area electrons (negative) playing the role of conductor, high voltage pushes the electrons out

CMOS inverter in the "bird eye perspective":



Trojan design:



– whatever happens the VDD is connected to the output

Trojan TRNG

TRNG cosnsists of

– entropy source (physical)

– self test circuit (OHT - online health test)

– deterministic RNG, Intel version:

  – conditioner (computes seeds to rate matcher) and rate matcher (computes 128 bit numbers)

  – derivation, internal state (K, c):

    1. $c := c + 1$, $r := \mathrm{AES}_K(c)$

    2. $c := c + 1$, $x := \mathrm{AES}_K(c)$

    3. $c := c + 1$, $y := \mathrm{AES}_K(c)$

    4. $K := K \oplus x$

    5. $c := c \oplus y$

– attack: fix K by applying Trojan transistors, if K is known, then it is easy to find internal state c from r and then the consecutive random numbers r

- problem with OHT: tests with some values have to create known outputs (32 CRC from the last 4 outputs), knowing the test one can find K by exaustive search

Side channel Trojan:

- side channel resistant logic: Masked Dual Rail Logic

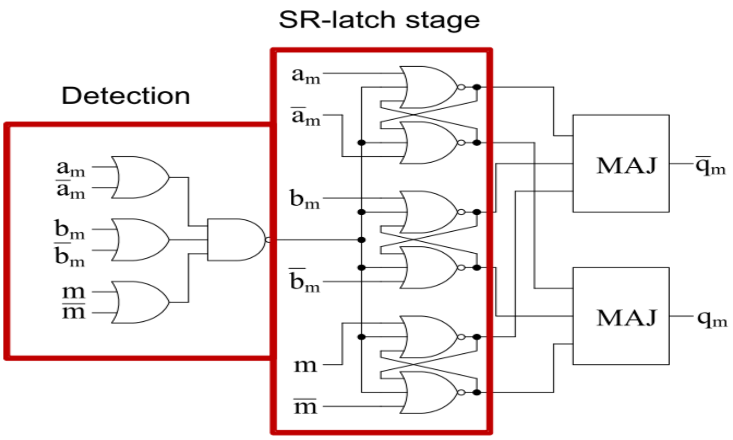
  i. for each a both a and negation of a computed

  ii. precharge: each phase preceded by charging all gates

  iii. masking operations by random numbers:

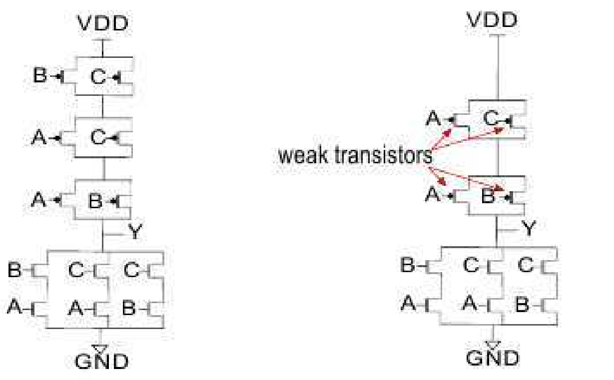
  computing $a \wedge b$:

  - input $a \oplus m$, $a \oplus \neg m$, $b \oplus m$, $b \oplus \neg m$, m, $\neg m$
  - detection, SR-latch stage and majority gate



gates on the picture: OR, AND, NOR (output 1 if all inputs 0)

attacking not-majority gate:



a) Trojan free AOI222 Gate b) Trojan AOI222 Gate

Idea: instead of cutting output a low voltage

- the same behavior except for $A = 0$ and $B, C = 1$, where good output but high power consumption due to connection between VDD and VSS

Artificial aging

make some transistors disfuctional (as ithe case of PRNG)

method:

- apply to high voltage at certain areas
- the electrons accelerate and break barrier - damages
- effect the same as of aging a chip
- the transistor changes its operational characteristic

Defense methods:

- problem: Trojan may be triggered by some particular event, detection becomes harder
- problem: Trojan may work in very particular physical conditions, e.g. temperature, voltage
- on-chip checks: detection of unexpected behavior, e.g. delay characteristics: workload path and a shadow path that provides result after fixed time, + comparison
- ring osscilators on the chip detecting nonstandard behavior
- methods to enable activation in certain areas only
- inserting PUFs, (either randomize as much a s possible - noise over trojan information)
- keep algorithm deterministic
- secure coding: take into account the situation that certain components are not working properly
- external watchdogs techniques

VIII. QUANTUM COMPUTING AND QUANTUM DEVICES

as there are problems to guarantee security of devices in the traditional way, maybe there is a way out using physics? three directions:

1) quantum based random number generators

2) key transport

3) quantum cryptanalysis

Qubit

the concept is as follows:

- instead of a bit with discrete states 0 and 1 we have a linear combination of basis vectors denoted by $|0\rangle$ and $|1\rangle$:

$\alpha \cdot |0\rangle + \beta \cdot |1\rangle$

- with α , β complex numbers
- a measurement of $\alpha \cdot |0\rangle + \beta \cdot |1\rangle$ yields $|0\rangle$ with pbb $|\alpha|^2$ and $|1\rangle$ with pbb $|\beta|^2$ – this is quite annoying but ...
- measurement may be performed only for orthogonal basis. the basis can be different from $|0\rangle$ and $|1\rangle$. E.g.:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad , \quad \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

- measuring $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ in basis $|0\rangle$ and $|1\rangle$ yields both 0 and 1 with perfect probability 0.5: it seems to be a perfect source of random bits:
  - generate fotons $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
  - measure them in basis $|0\rangle$ and $|1\rangle$
- moreover: reading changes the state to the state read: if the result is $|0\rangle$ then the physical state becomes $|0\rangle$ as well. There is no state $\alpha \cdot |0\rangle + \beta \cdot |1\rangle$ anymore.
- **In fact, this is the core of Shor'a algorithm - a reading operation creates a change in a physical system that would be infeasible to compute on a classical computer**
- instead of a single bit we may have strings of qubits, say of length l where $l > n$

Random Number Generators

Problems:

- high price (1000 EUR and more)
- while physical source might be ok, reading circuit introduces high bias, very poor results (2017) in the standard randomness tests for devices available on the market
- bias can be removed via additional logic, but extra hardware may mean place for Trojans and the whole advantage is gone

Quantum key transport, BB84

- Charles Bennett and Gilles Brassard, 1984, 1st quantum protocol, even implemented
- key agreement immune to eavsdropping (reading qubits is detectable)
- two bases used: $|0\rangle$ and $|1\rangle$ (denoted +) or $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ (denoted $\times$)
- encoding of bits:

  basis +: $|0\rangle = 0$ and $|1\rangle = 1$

  basis $\times$: $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = 0, \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = 1$

– Steps:

1. Alice chooses at random bitstrings a and b of length n
2. for $i \leq n$ Alice encodes a_i according to basis indicated by b_i (0 indictes $+$, 1 indicates $\times$)
3. Alice sends n photons (codes for a)
4. Bob chooses at random string $\hat{b}$ of n bits,
5. Bob measures the photons using basis indicated by $\hat{b_i}$ for the ith photon
6. Alice sends b to Bob over a traditional (public) channel, Bob sends $\hat{b}$ to Alice
7. Alice and Bob take the substring K of bits a_i such that $b_i = \hat{b_i}$
8. Alice chooses a subset of 50% of bit of K and discloses them to Bob
9. Bob checks how many of them disagree with his measurement. If above some threshold then it is likely that an adversary has measured the transmission as well and the protocol is aborted (environment may also create inconsistencies)
10. the unpublished substring of K may differ between Alice and Bob: an error correcting procedure applied (error correction attracts the bitstrings to the closest codewords, so if the strings of Alice and Bob differ slightly, then they result in the same codeword)
11. "privacy amplification": hashing to a much shorter string

– effect of eavesdropping:

- say Alice chooses basis $\times$ to encode 0,
- eavesdropper Eve chooses a different basis for the measurement: namely $+$
- Eve gets $|0\rangle$ with pbb .5 and $|1\rangle$ with pbb .5, say $|0\rangle$ has been obtained
- at the same time the photon converted to $|0\rangle$!!!!!!
- Bob measures the photon according to basis $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
- $|0\rangle = \frac{1}{\sqrt{2}}\Big(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) + \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\Big)$, so both results of the measurements (i.e $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ or $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ are equally probably for Bob, so the measurement of Bob indicated 1 with pbb .5
- corollary: eavesdropping creates inconsistency between Alice and Bob with pbb .5 once Eve chooses a different basis than Alice and Bob

– quantum hacking: in theory wonderful, but the problems come with physical realization

→ sending many photons to Bob at the time when his hardware already set for a measurement. reflected photons show the basis used

Eckert algorithm:

- entangled pair of photons: measurement of one of them makes the mirror change of the other photon
- procedure:
    1. generate entangled qubits $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ by some source
    2. measure them by Alice and Bob on both ends of the channel

Properties:

- long distance transmissions possible, even to moving airplane
- over optical fibre or in free space (better vacuum)
- 1203 km between ground stations over sattelite (China)
- both BB84 and Eckert used
- high price
- does not solve man-in-the-middle issue

Quantum computing and Shor factorization algorithm

Problem and its algebraic context:

- given an RSA number $n = p \cdot q$ for prime factors p and q of a similar size, the goal is to find p or q
- many modern crypto products are based on difficulty of this factorization problem. There are many software systems and embedded devices with RSA, no update is possible
- in order to break factorization problem it suffices to learn a nontrivial root r of 1:
    - $r \neq -1$
    - $r^2 = 1 \bmod n$

  indeed

    - $r^2 - 1 = (r-1)(r+1) = 0 \mod p \cdot q$
    - therefore p divides either $r-1$ or $r+1$
    - if p divides $r-1$ then q cannot divide $r-1$ as then $r-1$ would be at least n, but $r-1 < n$
    - in this situation we compute $\mathrm{GCD}(n, r-1)$, the result must be p

- ◦ if p divides $r+1$ then q cannot divide $r+1$ and therefore q must divide $r-1$. In this case $\mathrm{GCD}(n, r-1)$ yields q

- if for a given $a < n$ we find s such that $a^s = 1$, then with probability ≥ 0.5 we get $a^{s/2}$ as a nontrivial root of 1. Indeed:

  - ◦ by Chinese Reminder Theorem a number $a < n$ is represented by $a_p = a \bmod p$ and $a_q = a \bmod q$
  - ◦ given a and b we may compute representation of $a \cdot b \bmod n$ by computing $a_p \cdot b_p \bmod p$ and $a_q \cdot b_q \bmod q$
  - ◦ there are two roots of 1 modulo prime number p: 1 and $p-1$
  - ◦ if $a^s = 1 \bmod n$, while $a^{s/2} \neq 1 \bmod n$, then $a^{s/2} \bmod p$ is 1 or -1
  - ◦ there are the following cases:

    1. $a^{s/2} = 1 \bmod p$, $a^{s/2} = -1 \bmod q$
    2. $a^{s/2} = -1 \bmod p$, $a^{s/2} = 1 \bmod q$
    3. $a^{s/2} = -1 \bmod p$, $a^{s/2} = -1 \bmod q$

    the last case corresponds to $-1 \bmod n$, the first two ones to a nontrivial roots of -1

- so it suffices to find such an s - the order of a. By repeating the procedure for different a's we finally find a nontrivial root of $-1 \bmod n$

Quantum operations and gates

- a quantum computer should perform some operations on qubits, technical realization is a challenge, but in theory possible

- we consider $l-$qubit numbers as representing numbers mod 2^l (well, this is fuzzy as each bit is fuzzy as a qubit), in this way we a get quantum state for each $a < q = 2^l$

- Hadamard transformation: an easy way to create a quantum state such that takes any value a (denoted $|a\rangle$) with the same probability. The way to achieve this is:

  - create the state $|0....0\rangle$
  - apply Hadamard transformation gate to it
  - each ccordinate is transformed by

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1, 1 \\ 1, -1 \end{pmatrix}$$

    so $|0\rangle$ is transformed to

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

- Quantum Fourier transform:
  - regular FT: $(x_1, ..., x_N)$ transformed to $(y_1, ..., y_N)$ where
    $y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \cdot e^{(2\pi i \cdot j \cdot k)/N}$
  - quantum:
    $\sum x_i \cdot |i\rangle$ transformed to $\sum y_i \cdot |i\rangle$ where
    $y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \cdot e^{(2\pi i \cdot j \cdot k)/N}$
  - in other words:

    $|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{(2\pi i \cdot j \cdot k)/N} \cdot |k\rangle$

  „efficient implementation" based on similar algebra as for DFT

Shor's algorithm (based on presentation of Eric Moorhouse)

1. fix q such that $2n^2 < q < 3n^2$, $q = 2^l$ (or a product of small primes)
   we use states with $2l$ qubits, notation $|a, b\rangle$ or $|a\rangle|b\rangle$
2. prepare state $|0, 0\rangle$ and apply Hadamard transformation to the first register. Its result is a state
   $$|\psi\rangle = \frac{1}{\sqrt{q}} \cdot \sum_{a=0}^{q-1} |a, 0\rangle$$
3. fix $x < n$ at random
4. to the state $|\psi\rangle$ apply the quantum transformation
   $$|a, 0\rangle \rightarrow |a, x^a \bmod n\rangle$$
   the result is
   $$\frac{1}{\sqrt{q}} \cdot \sum_{a=0}^{q-1} |a, x^a \bmod n\rangle$$
   (there is a theory how to make such a computation with quantum gates)
5. measure the second register. The result is some k. But then the measured state changes to
   $$\frac{1}{\sqrt{M}} \cdot \sum_{a \in A}^{M-1} |a, k\rangle$$
   where A is the set of all a such that $x^a = k \bmod n$
   so $A = \{a_0, a_0 + r, a_0 + 2r....\}$ where r is the order of x and $M = |A|$ (so $M \approx q/r$)
   $$\frac{1}{\sqrt{M}} \cdot \sum_{d=0}^{M-1} |a_0 + d \cdot r, k\rangle$$

6. apply the DFT to the first register. This changes the state

$$\frac{1}{\sqrt{M}} \cdot \sum_{d=0}^{M-1} |a_0 + d \cdot r, k\rangle$$

to

$$\frac{1}{\sqrt{q \cdot M}} \cdot \sum_{c=0}^{q-1} \sum_{d=0}^{M-1} e^{2\pi i \cdot c(a_0 + d \cdot r)/q} \cdot |c, k\rangle$$

which is equal to

$$\sum_{c=0}^{q-1} \frac{e^{2\pi i \cdot c \cdot a_0/q}}{\sqrt{q \cdot M}} \sum_{d=0}^{M-1} e^{2\pi i \cdot c \cdot d \cdot r/q} \cdot |c, k\rangle$$

$$\sum_{c=0}^{q-1} \frac{e^{2\pi i \cdot c \cdot a_0/q}}{\sqrt{q \cdot M}} \sum_{d=0}^{M-1} \zeta^d \cdot |c, k\rangle$$

where

$$\zeta = e^{2\pi i \cdot c \cdot r/q}$$

7. measure the first register (this is the key moment!!)

   - which c is read depends on the values of $\sum_{d=0}^{M-1} \zeta^d$ which in turn corresponds to the probability
   - if $c \cdot r/q$ is not very close to an integer, then the sum is $\frac{1-\zeta^M}{1-\zeta}$
   - if $c \cdot r/q$ is an integer, then we sum up M ones
   - so the former case is unlikely and the readings are concentrated around values c such that

     $$c/q \approx d/r$$

     for an integer d
   - the rest is a classical computation involving c, q and trying different d's. The search space is relatively narrow

CATACRYPT catastrophy cryptography

– what happens if assumptions broken (e.g. DL solvable for some group)?

– "post-quantum crypto"

reality:

– post-quantum is at early stage, no industrial products, logistically impossible to replace

– no plans, scenarios, ...

– consequences of building a quantum computer or anything that breaks the current mechanisms:

   i. option 1: the situation is well hidden, the party controlling the means uses it without leaving traces but still claims the system is secure

ii. option 2: disclosed, call for massive conversion to new products (may occur even if there is no quantum computer but it works for increasing sale numbers)

iii. option 3: disclosed, neglected

- catastrophy is already there

XI. COMMUNICATION SECURITY – SSL/TLS

mistakes:

- risk of common (standard) groups
- cryptanalysis: most efficient number field sieve (NFS):
  - complexity subexponential (for $\mathbb{Z}_p$ it is

$$\exp\left(1.93 + o(1)\right)\left(\log p^{\,1/3} (\log\log p)^{2/3}\right)$$

  - most time precomputation independent from the target number y (where $\log y$ to be computed in a given group)
  - the time dependant from y can be optimized to subexponential but much lower
  - 512-bit groups can be broken, MitM attack can be mounted
- standard safe primes – seem to be ok, but attacker can amortize the cost over many attacks
- TLS with DH: frequently "export-grade" DH with 512 bit primes, about 5% of servers support DHE_EXPORT, most servers (90% and more) use a few primes of a given length, after a precomputation breaking for a given prime: reported as 90 sec
- TLS: client wants DHE, server offers DHE_EXPORT, but one can manipulate the messages exchanged, so that the client treats the (p_{512}, g, g^b) as a response to DHE – it is not an implementation bug!
  - handshake time is a problem, but some protocols allow. sending TLS warning alert that reset the countdown
  - ephemeral key hashing
  - sometimes non safe prime used ($\frac{p-1}{2}$ composite), Pohling-Hellman method can be used
  - DH-768 breakable on academic level, DH-1024 on the state level
- recommendations:
  - avoid fixed prime groups
  - transition to EC
  - deliberately do not downgrade security even if seems to be ok

- follow the progress in computer algebra

Padding attack (Serge Vaudenay)

Scenario:

- for encryption the plaintext should have the length as a multiply of b
- always pad something
- if i positions have to be padded, then writes i's there. de-padding is then easy.
- encrypt the resulting padded plaintext $x_1,...,x_N$ in CBC mode with IV (fixed or random) and a block cipher Enc:

$$y_1 = \mathrm{Enc}(\mathrm{IV} \oplus a_1), \quad y_i = \mathrm{Enc}(y_{i-1} \oplus x_i)$$

- properties of CBC:
  - efficiency
  - confidentiality limits: if IV fixed one can check that two plaintexts have the same prefix of a given size

attack:

- manipulate the ciphertext
- destination node decrypts, it can detect incorrect padding
- decision: what to do if the padding is incorrect? Each reaction is wrong:
  - → reject: creates padding oracle (attacker tests the behavior)
  - → proceed: enables manipulation of the plaintext data

last word oracle:

- goal: compute $\mathrm{Dec}(y)$ for a block y
- create an input for padding oracle:
  - create a 2 block ciphertext: $r = r_1...r_b$ chosen at random, $c := r|y$
  - oracle call: if $\mathrm{Oracle}(c) = \mathrm{valid}$, then $\mathrm{Dec}(y) \otimes r$ should yield a correct padding. whp this happens if $y_b = r_b \oplus 1$
  - it may happen that the oracle says valid because of other correct padding. The following procedure solves the problem (idea: change conseqtive words in the padding until invalid:
    1. pick $r_1, r_2..., r_b$ at random, take $i = 0$
    2. put $r = r_1 r_2 ... r_{b-1}(r_b \oplus i)$
    3. run padding oracle on $r|y$, if the result "invalid" then increment i and goto (2)
    4. $r_b := r_b \oplus i$

5. for $j = b$ to 2:

   $r := r_1 ... r_{b-j}(r_{b-j+1} \oplus 1) r_{b-j} ... r_b$

   ask padding oracle for $r|y$, if "invalid" then output $(r_{b-j+1} \oplus j) ... (r_b \oplus j)$ and halt

6. output $r_b \oplus 1$

block decryption oracle

let $a_1 ... a_b$ be the plaintext of y

decryption:

- get a_b via the last word oracle
- proceed step by step learning a_{j-1} once $a_j, ..., a_b$ are already known
  1. set $r_k := a_k \oplus (b - j + 2)$ for $k = j, ..., b$ /* preparing the values so that the padding values $(b - j + 2)$ appear at the end)
  2. set $r_1, ..., r_{j-1}$ at random, $i := 0$ /* search for the value that makes a proper padding
  3. $r := r_1 ... r_{j-2}(r_{j-1} \oplus i) r_j ... r_b$
  4. if $O(r|y) =$ invalid, then $i := i + 1$ and goto 3
  5. output $r_{j-1} \oplus i \oplus (b - j + 2)$

decryption oracle

- block by block
- the only problem with the first block if IV is secret

bomb oracles:

- padding oracle in SSL/TLS breaks the connection if a padding error occurs , so can be used only once
- bomb oracle: try a longer part at once

other paddings:

- $00....0n$ instead of $nn....n$ – also vulnerable
- $12....n$ instead of $nn....n$ – also vulnerable

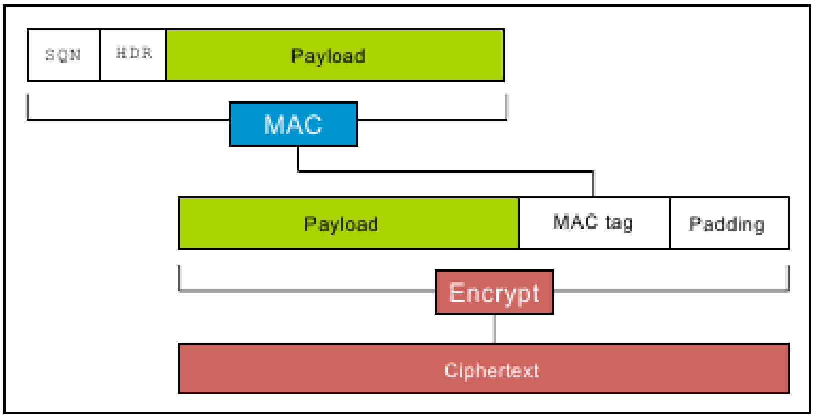
Applications for (old) versions of SSL/TLS, ...

- MAC applied before padding, so padding oracle techniques can be applied
- wrong MAC and wrong padding create the same error message - from SSL v3.0, debatable whether it is impossible to recognize situation via side channel (response time)

- TLS attempts to hide the plaintext length by variable padding:
  - checking the length of padding: take the last block y, send $r|y$ where the last word of r is $n \oplus 1$. acceptance means that the padding is of length n
  - checking paddings longer than a block: send ry_1y_2 where y_1y_2 are the last blocks
- IPSEC: discards message with a wrong padding, no error message, other activities to process errors (they may leak information)
- WTLS: decryption-failed message in clear (!) session not interrupted
- SSH: MAC after padding (+)

Lucky Thirteen

- concerns DTLS (similar to TLS for UDP connections)
- MAC-Encode-Encrypt paradigm (MEE), MAC is HMAC based



- 8-byte SQN, 5-byte HDR (2 byte version field, 1 byte type field, 2 byte length field)
- size of the MAC: 16 bytes (HMAC-MD5), 20 bytes (HMAC-SHA1), 32 bytes (HMAC-SHA-256)
- padding: $p+1$ copies of p, at least one byte must be added
- after receiving: checking the details: padding, MAC, (underflow possible if padding manipulated and removing blindly)
- HMAC of M:

  $T := H((K_a \oplus \text{opad}) || H((K_a \oplus \text{ipad}) || M))$
- **Distinguishing attack:**

  → M_0 : 32 arbitrary bytes followed by 256 copies of 0xFF

  → M_1: 287 bytes followed by $0x00$

→ both 288 bytes, 18 plaintext blocks

→ encoded $M_d||T||$pad, we aim to guess d

→ $C -$ the ciphertext

→ create a ciphertext C' by truncating all parts corresponding to $T||$pad

→ give HDR$||C'$ for decryption

→ if M_0: the 256 copies of 0xFF interpreted as padding and removed, remaining 32 bytes as short message and MAC, calculating MAC: 4 hash computed, then typically error returned to the attacker

→ if M_1: 8 hash evaluations

Plaintext recovery attacks

- C^* – the block of ciphertext to be broken, C' – the ciphertext block preceding it
- we look for P^*, where $P^* = \mathrm{Dec}(C^*) \oplus C'$
- assume CBC with known IV, $b = 16$ (as for AES). $t = 20$ (as for HMAC-SHA-1)
- let Δ be a block of 16 bytes, consider

$$C^{\mathrm{att}}(\Delta) = \mathrm{HDR}||C_0||C_1||C_2||C' \oplus \Delta||C^*$$

  it represents 4 non-IV blocks in the plaintext, the last block is:

$$P_4 = \mathrm{Dec}(C^*) \oplus (C' \oplus \Delta) = P^* \oplus \Delta$$

- case 1: P_4 ends with 0x00 byte:
  - 1 byte of padding is removed, the next 20 bytes interpreted as MAC, 43 bytes left - say R. MAC computed on SQN|HDR|R of 56 bytes
- case 2: P_4 ends with padding pattern of ≥ 2 bytes:
  - at least 2 bytes of padding removed, 20 bytes interpreted as MAC, at most 42 bytes left, MAC over at most 42+13=55 bytes
- case 3: P_4 ends with no valid padding:
  - according to RFC of TLS 1.1, 1.2 treated as with no padding , 20 bytes treated as MAC, verification of MAC over 44+13=57 bytes

    – MAC is computed to avoid other timing attack!
- time: case 1 and 3: 5 evaluations of SHA-1, case 2: 4 evaluations of SHA-1, detection of case 2 possible in LAN
- in case 2: most probable is the padding 0x01 0x01, all other paddings have probability about $\approx \frac{1}{256}$ of probability of 0x01 0x01, so we may assume that $P_4 = P^* \oplus \Delta$ ends with 0x01 0x01. Then we derive the last two bytes of P^*.

  repeat the attack with Δ' that has the same last two bytes to check if the padding has the length bigger than 2.

- after recovery of the last two bytes the rest recovered byte by byte from right to left:
  - the original padding attack
  - e.g. to find 3rd rightmost byte set the last two bytes Δ so that P_4 ends with 0x02 0x02, then try different values for the Δ_{13} so that Case 2 occurs (meaning that P_4 ends with 3 bytes 0x02
  - average time: $14 \cdot 2^7$ trials
- practical issues:
  - → for TLS after each trial connection broken, so multi-session scenario
  - → timing difference small, so necessary to gather statistical data
  - → complexity in fact lower, since the plaintexts not from full domain: e.g. http username and password are encoded Base64
  - → partial knowledge may speed up the recovery of the last 2 bytes
  - → less efficient configuration of the lengths for HMAC-MD5 and HMAC-SHA-256

BEAST

attack, phase 0:

1. P to be recovered (e.g. a password, cookie, etc), requires ability to force Alice to put secret bits on certain positions
2. force Alice to send $0...0P_0$ (requires malware on Alice computer) – of course encrypted
3. eavesdrop and get $C_p = \mathrm{Enc}(C_{p-1} \oplus 0...0P_0)$
4. guess a byte g
5. force Alice to send the plaintext $C_{i-1} \oplus C_{p-1} \oplus 0...0g$
6. Alice sends $C_i = \mathrm{Enc}(C_{i-1} \oplus C_{i-1} \oplus C_{p-1} \oplus 0...0g) = \mathrm{Enc}(C_{p-1} \oplus 0...0g)$
7. if $C_i = C_p$ then $P_0 = g$

attack phase 1:

1. P_0 already known
2. force Alice to send $0...0P_0P_1$ and proceed as in phase 0

last phase: we get the test for the whole $P_0...P_{15}$

protection: browser must be carefully designed and do not admit injecting plaintexts (SOP- Same Origin Protection). Some products do not implement it.

CRIME (2012)

- based on compression algorithm used by some (more advanced) versions of TLS
- compression: LZ77 and then Huffman encoding, LZ77- sliding window approach: instead of a string put a reference to a previous occurence of the same substring
- idea of recovering cookie:

```
POST / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1
Cookie: secretcookie=7xc89f94wa96fd7cb4cb0031ba249ca2
Accept-Language: en-US,en;q=0.8

(... body of the request ...)
```

Listing 1: *HTTP request of the client*

modified POST:

```
POST /secretcookie=0 HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1
Cookie: secretcookie=7xc89f94wa96fd7cb4cb0031ba249ca2
Accept-Language: en-US,en;q=0.8

( ... body of the request ...)
```

Listing 2: *HTTP request modified by the attacker*

LZ77 compresses the 2nd occurence of secretcookie= or secretcookie=0. We try all

secretcookie=i to find out the case when compression is easier (secretcookie=7)

when the first character recovered the attacker repeats the attack for the second character (trying all "secretcookie=7i" in the preamble)

TIME

- again based on compression but now on the server side (from the client to the server compression might be disabled and CRIME fails)
- works if the server includes the client's request in the response (most do!)
- works even if SOP is enabled. SOP does not control data with the tag `img`, so the attacker can manipulate length
- the attacker requires malicious Javascript on the client's browser
- the attacker tries to get the secret value sent from the server to the client
- mechanism:
  - → as in CRIME, the request sends "secretvalue=x" where x varies
  - → the response is compressed, so it takes either "secretvalue=" or "secretvalue=x"

→ the length manipulated so that either two or one packets – connection specific data must be used: Maximum Transmission Unit

→ RTT (round trip time) measured

- independent on the browser, it is not an implementation attack!
- countermeasure: restrict displaying images

BREACH

Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext

- attack against HTTP compression and not TLS compression as in case of CRIME
- a victim visits attacker-controlled website (phishing etc).
- force victim's computer to send multiple requests to the target website.
- check sizes of responses

```
GET /product/?id=12345&user=CSRFtoken=<guess> HTTP/1.1
Host: example.com
```

Listing 4: *Compromised HTTP request*

```
<form target="https://example.com:443/products/catalogue.aspx?id=12345&user=CSRFtoken=<guess>" >
...
<td nowrap id="tdErrLgf">
<a href="logoff.aspx?CSRFtoken=4bd634cda846fd7cb4cb0031ba249ca2">Log Off</a></td>
```

Listing 5: *HTTP response*

- requirements: application supports http compression, user's input in the response, sensitive data in the response
- countermeasures:

→ disabling compression

→ hiding length

→ no secrets in the same response as the user's data

→ masking secret: instead of S send $R||S \oplus R$ for random R (fresh in each response)

→ trace behaviour of requests and warn the user

POODLE (2014)

in SSL v.3.0 using technique from BEAST:

– encrypted POST request:

POST /path Cookie: name=value... ⟨\r\n\r\n⟩ body ||20-byte MAC||padding

- manipulations such that:
  - the padding fills the entire block (encrypted to C_n)
  - the last unknown byte of the cookie appears as the last byte in an earlier block encrypted into C_i
- attack: replace C_n by C_i and forward to the server

  usually reject

  accept if $\mathrm{Dec}_K(C_i)[15] \oplus C_{n-1}[15] = 15$, thereby $P_i[15] = 15 \oplus C_{n-1}[15] \oplus C_{i-1}[15]$

  proceed in this way byte by byte
- downgrade dance: provoke lower level of protection by creating errors say in TLS 1.0, and create connection with SSL v3.0
- the attack does not work with weak (!) RC4 because of no padding

Weaknesses of RC4

- known weaknesses:
  - → the first 257 bytes of encryption strongly biased, ≈200 bytes can be recovered if ≈232 encryptions of the same plaintext available

    simply gather statistics as in case of Ceasar cipher
  - → at some positions (multiplies of 256) if a zero occurs then the next position more likely to contain a zero
- broadcast attack: force the user to encrypt the same secret repeatedly and close to the beginning
- countermeasure: no secrets in the initial part!

TLS 1.2

differences with TLS 1.1 and TLS 1.0 (Edukacja runs with TLS 1.0):

- explicit IV instead of implicit IV
- IDEA and DES 64bit removed
- MD5/SHA-1 PRF 65 is replaced with a suite specified hash function – SHA-256 for all TLS 1.2 suites, but in the future also SHA-3, ...
- digitally-signed element includes the hash algorithm used
- Verify_data length is no longer fixed length ⇒ TLS 1.2 can define SHA-256 based cipher suites
- new encryption modes allowed: CCM, GCM

TLS 1.3 (draft)

many old algorithms removed (RC4,...)

CCM encryption mode

Prerequisites: block cipher algorithm; key K; counter generation function; formatting function; MAC length $Tlen$

Input: nonce N; payload P of $Plen$ bits; valid associated data A

Computation: Steps:

1. formatting applied to (N,A,P), result: blocks $B_0,\ldots,B_r$
2. $Y_0 := \mathrm{Enc}_K(B_0)$
3. for $i=1$ to r: $Y_i := \mathrm{Enc}_K(B_i \oplus Y_{i-1})$
4. $T := \mathrm{MSB}_{Tlen}(Y_r)$
5. generate the counter blocks $\mathrm{Ctr}_0, \mathrm{Ctr}_1, \ldots, \mathrm{Ctr}_m$ for $m = Plen/128$
6. for $j=0$ to m: $S_j := \mathrm{Enc}_K(\mathrm{Ctr}_j)$
7. $S := S_1 || \ldots || S_m$
8. $C := (P \oplus \mathrm{MSB}_{Plen}(S)) || (T \oplus S_0)$

Decryption:

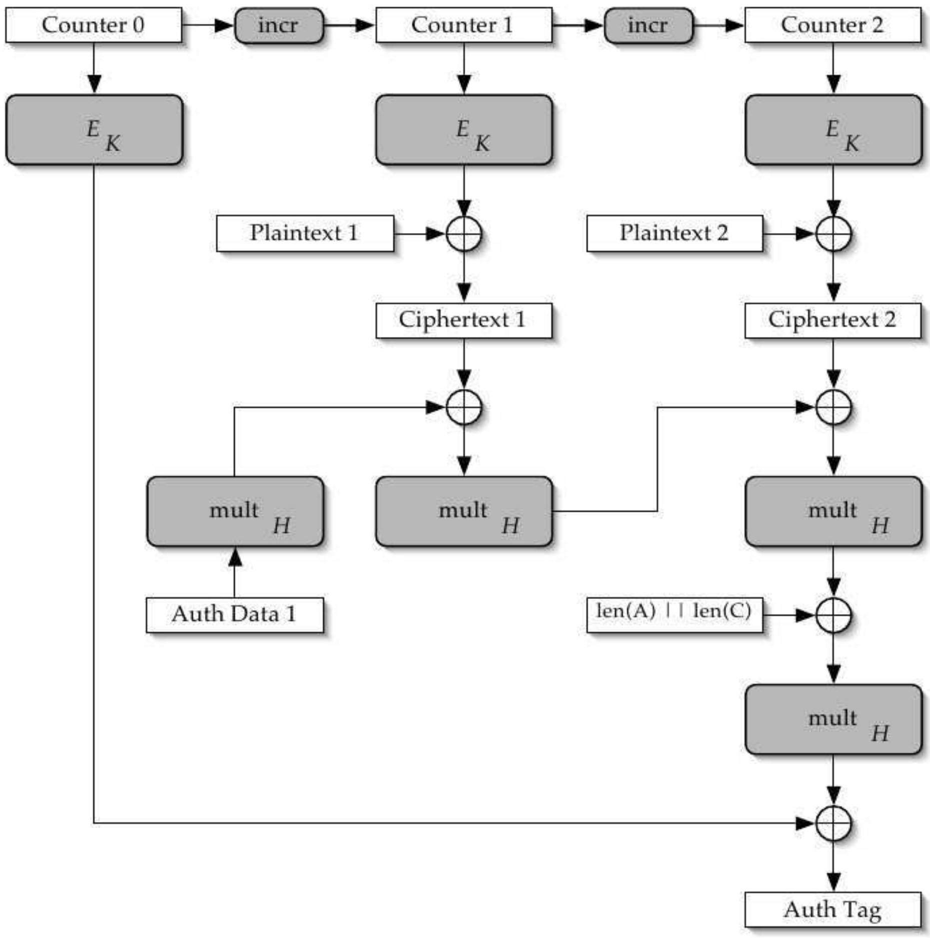
1. return INVALID, if $Clen < Tlen$
2. generate the counter blocks $\mathrm{Ctr}_0, \mathrm{Ctr}_1, \ldots, \mathrm{Ctr}_m$ for $m = Plen/128$
3. for $j=0$ to m: $S_j := \mathrm{Enc}_K(\mathrm{Ctr}_j)$
4. $S := S_1 || \ldots || S_m$
5. $P := \mathrm{MSB}_{Clen}(C) \oplus \mathrm{MSB}_{Plen}(S)$
6. $T := \mathrm{LSB}_{Tlen}(C) \oplus \mathrm{MSB}_{Tlen}(S_0)$
7. If N, A or P invalid, then return INVALID, else reconstruct $B_0,\ldots,B_r$
8. recompute $Y_0,\ldots,Y_r$
9. if $T \neq \mathrm{MSB}_{Tlen}(Y_r)$, then return INVALID, else return P.

GCM (The Galois/Counter Mode)

Computation: Steps:

1. $H := \mathrm{Enc}_K(0^{128})$
2. $Y_0 := \mathrm{IV} || 0^{31}1$ if length of IV should be 96

   or $Y_0 := \mathrm{GHASH}(H, \{\}, \mathrm{IV})$
3. $Y_i := \mathrm{incr}(Y_{i-1})$ for $i = 1,\ldots,n$ (counter computation)
4. $C_i := P_i \oplus \mathrm{Enc}_K(Y_i)$ for $i = 1,\ldots,n-1$ (counter based encryption)
5. $C_n^* := P_n \oplus \mathrm{MSB}_u(\mathrm{Enc}_K(Y_n))$ (the last block need not to be full)
6. $T := \mathrm{MSB}_t(\mathrm{GHASH}(H,A,C)) \oplus \mathrm{Enc}_K(Y_0)$



Details of computation of the tag

$\mathrm{GHASH}(H, A, C) = X_{m+n+1}$ where m is the length of authenticating information A, and:

X_i equals:

0	for $i = 0$
$(X_{i-1} \oplus A_i) \cdot H$	for $i = 1, \ldots, m-1$
$((X_{i-1} \oplus (A_m^* \| 0^{128-v})) \cdot H$	for $i = m$
$(X_{i-1} \oplus C_i) \cdot H$	for $i = m+1, \ldots, m+n-1$
$((X_{m+n-1} \oplus (C_m^* \| 0^{128-u})) \cdot H$	for $i = m+n$
$((X_{m+n} \oplus (\mathrm{len}(A) \| \mathrm{len}(C))) \cdot H$	for $i = m+n+1$

Decryption:

1. $H := \mathrm{Enc}_K(0^{128})$
2. $Y_0 := \mathrm{IV} \| 0^{31}1$ if length of IV should be 96
   or $Y_0 := \mathrm{GHASH}(H, \{\}, \mathrm{IV})$
3. $T' := \mathrm{MSB}_t(\mathrm{GHASH}(H, A, C)) \oplus \mathrm{Enc}_K(Y_0)$, is $T = T'$?
4. $Y_i := \mathrm{incr}(Y_{i-1})$ for $i = 1, \ldots, n$
5. $P_i := C_i \oplus \mathrm{Enc}_K(Y_i)$ for $i = 1, \ldots, n$
6. $P_n^* := C_n^* \oplus \mathrm{MSB}_u(\mathrm{Enc}_K(Y_n))$

XII. CERTIFICATES and – SSL/TLS

"Certified Lies"

- rogue certificates + MitM attack: the user believes that is directed elsewhere
- no control over root CA's worldwide, indicated either by operating system or the browser
- compelled assistance from CA's ?

ROGUE Certificates and MD5

- target: create a certificate (webserver, client) that has not been issued by CA
- not forging a signature contained in the certificate but:
  i. find two messages that Hash(M_0) = Hash(M_1) and M_0 as well as M_1 have some common prefix that you expect in a certificate (e.g. the CA name)
  ii. submit a request corresponding to M_0, get a certificate with the signature over Hash(M_0)
  iii. copy the signature from the certificate concerning M_0 to a certificate based on M_1
- problems: some data in M_0 are to be guessed : sequential number, validity period,

  some other are known in advance: distinguished name, ...

legitimate website certificate		rogue CA certificate
serial number		serial number
issuing CA		issuing CA
validity period		validity period
domain name	chosen prefixes	rogue CA name
		1024 bit RSA public key
		extensions
		"CA=true"
		tumor
2048 RSA public key	collision bits	
extension "CA=false"	identical suffix	

Table.

- finding M_0 and M_1 has to be fast (otherwise the guess about the serial number and validity will fail) - e.g. a day over the weekend
- attack on MD5, general picture:

message A		message B
prefix P		prefix P'
padding S_r		padding S'_r
birthday blocks S_b		birthday blocks S'_b
near-collision block $S_{c,1}$		near-collision block $S'_{c,1}$
near-collision block $S_{c,2}$		near-collision block $S'_{c,2}$
...		...
near-collision block $S_{c,r}$	←collision→	near-collision block $S'_{c,r}$
suffix		suffix

Table.

- identical prefix, birthday bits, near collision blocks:
  - birthday bits: 96, end at the block boundary, they are RSA bits – in the genuine certificate, "tumor" (ignored part by almost all software- marked as a comment extension) – in the rogue certificate

    birthday bits make the difference of intermediate hash values computed for both certificates fall into a *good class*
  - then apply 3 near-collision blocks of 512-bits. website: we have "consumed" 208 + 96 + 3·512 = 1840 bits of the RSA modulus. Rogue certificate: all bits concerned are in the "tumor"
  - after collision bits: 2048-1840 = 208 bits needed to complete the RSA modulus of the webpage – we have to generate an RSA number with the prefix of 1840 bits already fixed

    – continued so that two prime factors:

    → B denotes the fixed 1840-bit part of the RSA modulus followed by 208 ones

    → select at random 224-bit integer q until $B \bmod q < 2^{208}$, continue until both q and $p = \lfloor B/q \rfloor$ are prime. Then
      - $p \cdot q$ is an RSA number
      - $p \cdot q < B$, , $B - p \cdot q = B - q \cdot \lfloor B/q \rfloor < 2^{208}$. Hence $p \cdot q$ has the same 1840 most significant bits as B

    → this RSA number is not secure, but still factorizing it is not feasible and cannot be checked by CA before signing (as the smallest factor is more than 67-digit prime)

    → ... one can create RSA signature for the webpage for the certificate request

- attack complexity (number of hash block evaluations) for a chosen prefix MD5: 2^{49} at 2007, 2^{39} in 2009, not much motivation for more work - remove MD5 certificates! (For a collision: 2^{16})

  for SHA-1 still 2^{77} in 2012 (for a collision: 2^{65})
- history:

  → attack found

  → real collision computed as a proof-of-concept

→ CA informed and given time

→ publication

→ code available

FLAME

- malware discovered 2012, 20MB, sophisticated code, mainly in Middle East, government servers attacked
- draft of the attack:
  - client attempts to resolve a computer name on the network, in particular make WPAD (Web Proxy Auto-Discovery Protocol) requests
  - Flame claims to be WPAD server, provides wpad.dat configuration file
  - victim that gets wpad.dat sets its proxy server to a Flame computer (later no sniffing necessary!)
  - Windows updates provided by the Flame computer. The main problem is that the updates must be properly signed!
  - signatures obtained for terminal Services, certificates issued by Microsoft LSRA PA. No Extended Key Usage restrictions – allows code signing, (except for Microsoft Hydra X.509 extension – this cannot be used for code-signing on Vista and Windows 7)
  - till 2012 still signatures with MD5 hash used
  - MD5 collision necessary to remove extension