Security and Cryptography 2015

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 come and go. Only the skills are important

exam date: TBA (quickly to enable internships in February)

I. EXAMPLE TO LEARN FROM: PKI FAILURE

reasons for PKI failure (According to Schneier):

1. whom we trust and for what? why CA should be trusted??
2. who is using my key? (private key - there are really no clones??)
3. how secure is the verifying computer? (no cryptography can help is the verifier software is cheating)
4. who is the signer? (ambiguity unless there is a trustworthy ID registry)
5. is CA an authority? (really not an authority for data contained in the certificate. Certificate bsed on fake documents...)
6. is the user part of the security design? (no, the user is free to behave in a stupid way)
7. separation CA and RA brings new threats
8. How did CA verify the certificate holder? (certificate issued for ..., but how to know that this was really this person)
9. How secure are the certification practices? (revocation, etc)
10. the customers wish to run single-sign-on

Answers: yes, but

- tradition in Nordic countries
- honest system participants
- the best one can do
- ...

  no solutions from crypto community

MAJOR PROBLEM: how to design/buy sound systems?

II. COMMON CRITERIA FRAMEWORK

http://www.commoncriteriaportal.org/

Idea:

- standardize the process of
  - designing products (Security Target ST),
  - designing requirements (Protection Profile, PP)
  - evaluation of products (licensed labs checking conformance of implementation with the documentation)
- 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,
- support for certification industry

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 $\Rightarrow$ high EAL level (common mistake in public procurement: EAL level ... without specifying PP
- clean up the zoo of different assumptions, descriptions, ...

Example for PP: BAC (Basic Access Control)

- 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 $K = \{\mathrm{KB}_{\mathrm{Enc}}, \mathrm{KB}_{\mathrm{Mac}}\}$
- steps:
  1. The MRTD chip sends the 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_{Enc}, K_{Mac} from master key K_{PICC} XOR K_{PCD} and sequence number derived from randoms (key derivation function)

- K derived from information available on the machine readable zone (optical)
- implementation: biometric passports.
- simple system. Really?

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

1. Introduction

1.1 PP reference

1 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, eg. 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 manufacturer 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.
- 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 s ecurity flaws in each release of the TOE.

  - The flaw remediation procedures sha ll 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:

  - first four like before

  - The flaw remediation procedures sh all 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 repor ted security flaws shall provide safeguards that any corr ections to these security flaws do not introduce any new flaws.

  - The flaw remediation guidance sha ll 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 distri bution 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.

- 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

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 ⇒ action
  - developer action element ⇒ work-unit
  - content and presentation of evidence element ⇒ work unit

- 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 (de scribe 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

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

  – 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

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

  – but not when they are 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. EIDAS REGULATION

goals:

- interoperability, comparable levels of trust
- merging national systems into pan-European one
- trust services, in particular: identification, authentication, signature, electornic seal, timestamping, delivery, Web authentication
- supervision
- information about
- focused on public administration systems. However, the rules for all trust services except for closed systems (not available to anyone).

tools:

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

technical concept:

- Member State provides an online system enabling identification and authentication with means from ths member state used abroad
- notification scheme for national systems
- if notified (some formal and technical conditions must be fulfilled), then every member state must admit 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 zoo of solutions
- breakthrough claimed, but likely to fail

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 proofing and verification (for mnatural 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 identiŋ fication 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 authoriŋ tative 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 differŋ 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.

IV. eIDAS TOKEN SPECIFICATION, BSI

- Technical guideline, security mechanisms for electronic travel documents, not focused on readers

- cryptographic mechanisms:
  - Password Authenticated Connection Establishment (PACE)
  - Terminal Authentication Version 2 (TA2).
  - Chip Authentication Version 3 (CA3)
  - Restricted Identification (RI)
  - Pseudonymous Signatures (PS)
- procedures
  - General Authentication Procedure (GAP)
  - Enhanced Role Authentication (ERA)
  - PIN Management
- terminal types:
  - inspection system
  - authentication terminal - government or private, terminal rights to be checked, GAP must be used
  - attribute terminal- extension of Authentication Terminal, ERA must be used
  - signature management terminal - key creation, signature creation
  - signature terminal - GAP must be used
  - priviledged terminals: category: inspection terminals and some authentication terminals explicitly authorized. Signature terminals are never priviledged
- user credentials:
  - MRZ-Password
  - CAN
  - PIN - always blocking (RC reaches 0 then blocked)
  - PUK - blocking or non-blocking
- password blocking: RC=0 password blocked, RC=1 - password suspended and the correct CAN must be entered during the same session to resume the password. Resume is volatile.
- switching session context: a stack of protocols, when terminating a protocol we return to the context on the top of the stack
- password authentication:
  - PACE- global passwords, VERIFY-application local
  - Inspection terminal SHALL use CAN or MRZ

  - authentication terminal SHALL use PIN, but the CAN can be allowed by the terminal

  - signature terminal: PIN, CAN or PUK

- Extended Access Control:

  - 1. Terminal Authentication v2: terminal SHALL generate ephemeral keys used later for Chip Authentication, only standard parameters, ephemeral keys authenticated, result: read/write access granted

  - 2. Passive Authentication: terminal reads and verifies Security Objects, compares the data obtained before PACE

  - 3. Chip Authentication v2 or v3: afterwards secure channel restarted

- General Authentication Procedure:

  i. password verification - PACE

  ii. EAC

  iii. read/write data

- Enhanced Role Authentication – authentication terminal with proper rules can proceed as follows:

  i. authentication terminal sends an ATTRIBUTE REQUEST to eIDAS token. token makes a link between the request and the terminal's sector

  ii. restore session context of PACE, store context of Chip authentication

  iii. EAC with attribute provider

  iv. proceed attribute request, write the resulting attributes to the eIDAS token, the access rights restricted to terminals with proper rights

  v. restore session context: PACE, then Chip Authentication

  vi. terminal may read the stored attributes

- online authentication:

  - eID server: remote part of authentication terminal

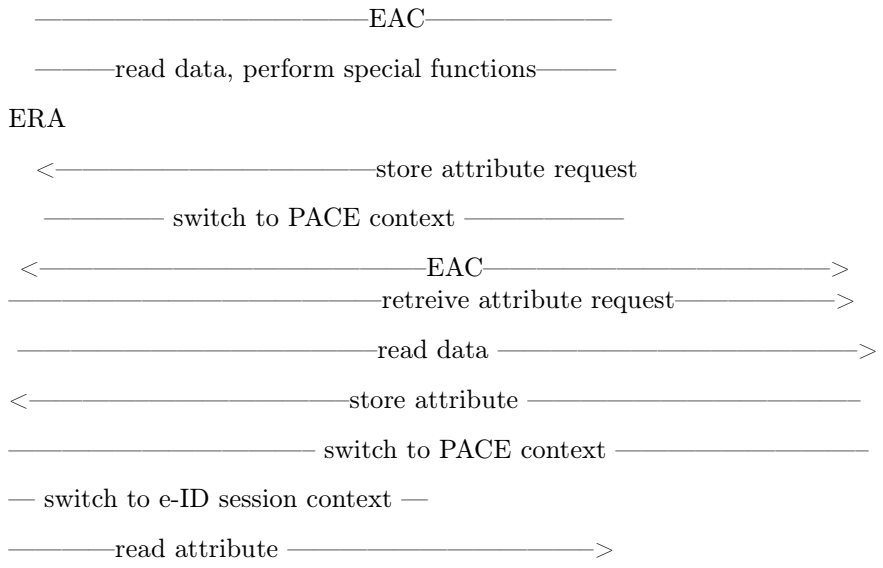
  - user device: interacts with user, eIDAS token and eID server, but not authorized to read eIDAS token data, access rights only after authentication with the eID server

  Protocol chart:

  token user device eID server Attribute Provider

  GENERAL AUTHENTICATION

  ————PACE————



- unauthenticated terminals:
  i. password verification based on PACE:
     – terminal does not show its type
     – can choose password type
     – after authentication secure messaging
  ii. authentication with CAN resumes PIN
  iii. updating retry counter

- authenticated terminals: after terminal authentication the terminal becomes authenticated

Cryptographic building blocks:

- hash $\boldsymbol{H}(m)$
- compression function for public key: $\boldsymbol{Comp}$(PK)
- projected representation of a public key $\boldsymbol{\Pi}$(PK)
- symmetric key algorithms:
  - deriving key for encryption $K_{\text{Enc}} = \boldsymbol{KDF}_{\text{Enc}}(K, [r])$
  - $K_{\text{Mac}} = \boldsymbol{KDF}_{\text{Mac}}(K, [r])$
  - $K_{\pi} = \boldsymbol{KDF}_{\pi}(\pi)$
  - encryption and decryption

  - MAC
- asymmetric algorithms:
  - domain parameters
  - keys (page 19):
    - eIDAS: ephemeral on both sides
    - Chip authentication: static on side of the chip
    - Chip authentication version 3: ephemeral on both sides based on static Chip's key
    - Restricted Identification: token uses a static key, sector public key, sector specific identifier
  - KA - key agreement (like DH)
  - signatures, mapping to RSA and ECDSA described

Pseudonymous signature:

- used for anonymous signature and for Chip Authentication v3
- 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
- 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

  - $Q_1 = (\mathrm{PK}_{\mathrm{ICC}})^c \cdot g^{s_1} \cdot (\mathrm{PK}_M)^{s_2}$
  - $A_1 = (I^{\mathrm{sector}}_{\mathrm{ICC},1})^c \cdot (\mathrm{PK}_{\mathrm{sector}})^{s_1}$
  - $A_2 = (I^{\mathrm{sector}}_{\mathrm{ICC},2})^c \cdot (\mathrm{PK}_{\mathrm{sector}})^{s_2}$
  - recompute c and check against the c from the signature

- why it works?

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

- there is a version without A_1, A_2 and the pseudonyms $I^{\mathrm{sector}}_{\mathrm{ICC},1}, I^{\mathrm{sector}}_{\mathrm{ICC},2}$

PACE (Password Authenticated Connection Establishment)

- ICAO Doc 9303: Basic Access Control/PACE and EAC v1 (=Chip Authentication v1+ Terminal Authentication v1) MUST be used
- password based authentication protocol
- password on the side of the token: stored, on the terminal: input by the user
- steps:

  i. token chooses s at random

  ii. token computes $z = \mathrm{Enc}(K_\pi, s)$, where $K_\pi = \mathrm{KDF}(\pi)$ and sends z to the reader together with the parameters D_{PICC}

  iii. the reader recovers s

  iv. the reader and the token compute $D_{\mathrm{Mapped}} = \mathrm{Map}(D_{\mathrm{PICC}}, s)$ (mapping function)

  v. the reader and the token perform anonymous Diffie-Hellman key agreement based on the ephemeral domain parameters (ephemeral values based on D_{Mapped} as an generator), shared secret K obtained

  vi. they create session keys $K_{\mathrm{Mac}} = \mathrm{KDF}_{\mathrm{Mac}}(K)$ and $K_{\mathrm{Enc}} = \mathrm{KDF}_{\mathrm{Enc}}(K)$

  vii. exchange and verification of tokens: $T_{\mathrm{PCD}} = \mathrm{MAC}(K_{\mathrm{MAC}}, \text{ephemeral key of PICC})$

$T_{\mathrm{PICC}} = \mathrm{MAC}(K_{\mathrm{MAC}}, \text{ephemeral key of PCD})$

viii. Secure Messaging restarted

Terminal authentication v2

- Chip Authentication MUST be performed after Terminal Authentication (condition repeated in the description of CHA v2 only)
- simple challenge-response algorithm, undeniable, resistant to replay
- ephemeral public key for ChA as a side effect
- steps:

  i. the terminal send the certificate chain to eIDAS token, it has to confirm the key $\mathrm{PK}_{\mathrm{PCD}}$

  ii. the token checks $\mathrm{PK}_{\mathrm{PCD}}$

  iii. the terminal creates ephemeral pair of keys, sends the compressed version of $\widetilde{\mathrm{PK}^{\mathrm{CA}}_{\mathrm{PCD}}}$ to the token

  iv. token replies with a random nonce r_{PICC}

  v. the terminal signs with $\mathrm{SK}_{\mathrm{PCD}}$ the following data: r_{PICC}, compressed version of $\widetilde{\mathrm{PK}^{\mathrm{CA}}_{\mathrm{PCD}}}$

  vi. the token checks the signature

Chip authentication v2

- static DH authentication with the ephemeral key of the terminal
- steps:

  i. the token sends its public key $\mathrm{PK}_{\mathrm{PICC}}$

  ii. the terminal sends ephemeral public key from TA (uncompressed)

  iii. static DH key agreement with $\mathrm{SK}_{\mathrm{PICC}}$ and ephemeral public key on side of the token, and $\mathrm{PK}_{\mathrm{PICC}}$ and ephemeral secret key on side of the terminal, master key K generated

  iv. token chooses r_{PICC}, computes $K_{\mathrm{Enc}} = \mathrm{KDF}_{\mathrm{Enc}}(K, r_{\mathrm{PICC}})$, $K_{\mathrm{Mac}} = \mathrm{KDF}_{\mathrm{Mac}}(K, r_{\mathrm{PICC}})$

  v. token computes the tag $T_{\mathrm{PICC}} = \mathrm{MAC}(K_{\mathrm{Mac}}, \text{ephemeral public key of the terminal})$

  vi. the terminal checks the tag

vii. secure messaging restarted using K_{Enc} and K_{Mac}

Chip authentication v3

- alternative to Chip authentication v2 and RI
- claimed: "message-deniable strong authentication", "pseudonymity without using the same key on several chips", "possibility of whitelisting eIDAS tokens"
- scheme:

  i. phase 0: terminal authentication, ephemeral key for terminal in phase 1 chosen and signed

  ii. phase 1: key agreement like DH with ephemeral keys on both sides, restarting secure messaging with new keys

  iii. phase 2:

  – static keys on the side of the chip: $\text{SK}_{\text{ICC},1}, \text{SK}_{\text{ICC},2}, \text{PK}_{\text{ICC}}$ and the parameters

  – terminal sends $\text{PK}_{\text{sector}}$ to the chip, the chip compares it with the "compressed" version received during Terminal Authentication

  – chip reconstructs $I^{\text{sector}}_{\text{ICC},1} = (\text{PK}_{\text{sector}})^{\text{SK}_{\text{ICC},1}}$ and $I^{\text{sector}}_{\text{ICC},2} = (\text{PK}_{\text{sector}})^{\text{SK}_{\text{ICC},2}}$

  – chip creates pseudonymous signature using $I_{\text{ICC},1}$, $I_{\text{ICC},2}$ as pseudonym and the secret keys $\text{SK}_{\text{ICC},1}$, $\text{SK}_{\text{ICC},2}$ over the ephemeral key given by the terminal

- If PACE GM used before ChA v3 then one can reuse the ephemeral key from the terminal
- checking the key PK_M is obligatory (otherwise it would be easy to forge the token)

Restricted Identification

- optional
- depending on the version, deanonymization might be possible or not (depending on $\text{PK}_{\text{sector}}$)
- executed after Terminal Authentication and Chip Authentication (not specified which version, but with v3 it does not makes sense)
- sector specific identifier computed as Hash(key computed via DH from $\text{PK}_{\text{sector}}$ and SK_{ID})
- blacklisting impossible in case of group key compromise (from ChA v2)

Pseudonymous Signature as replacement of RI

- whitelisting possible in case of group key compromise (claimed as new but possible for RI)
- the second part from ChA v3, the key $\text{PK}_{\text{sector}}$ used as sector public key

PSA - Pseudonymous Signature Authentication

- the sector public key = the ephemeral public key from ephemeral DH key agreement (now DH explicitly mentioned)

PSM - Pseudonymous Signature of a Message

- TA and ChA must be executed before
- message to be signed comes from the terminal
- public key unspecified

PSC - Pseudonymous Signature of Credentials

- used in combination with ERA
- Attribute Terminal involved, but eIDAS token creates the signature himself (after breaking group key one can also create the PSC)
- public key unspecified
- terminal rights to get the attributes are to be checked

PROBLEMS:

- security properties not stated, they can be derived via tedious analysis
- lack of security proofs
- underspecified (details may turn the token to be insecure)
- powerful adversary able to break into the token may crate fake ID's, unless whitelist approach used

---

V. STANDARS VERSUS SECURITY

Bleichenbacherr's RSA signature forgery based on implementation error

The attack works for PKCS-1 padding.

The PKCS-1 padding consists of a byte of 0, then 1, then a string of 0xFF bytes,
then a byte of zero, then the "payload" which is the hash+ASN.1 data.
Graphically:

```
00 01 FF FF FF ... FF 00  ASN.1  HASH
```

The signature verifier first applies the RSA public exponent to reveal this PKCS-1 padded data, checks and removes the PKCS-1 padding, then compares the hash with its own hash value computed over the signed data.

The error that Bleichenbacher exploits is if the implementation does not check that the hash+ASN.1 data is right-justified within the PKCS-1 padding. Some implementations remove the PKCS-1 padding by looking for the high bytes of 0 and 1, then the 0xFF bytes, then the zero byte; and then they start parsing the ASN.1 data and hash. The ASN.1 data encodes the length of the hash within it, so this tells them how big the hash value is. These broken implementations go ahead and use the hash, without verifying that there is no more data after it. Failing to add this extra check makes implementations vulnerable to a signature forgery, as follows.

An attacker forges the RSA signature for an exponent of 3 by constructing a value which is a perfect cube. Then he can use its cube root as the RSA signature. He starts by putting the ASN.1+hash in the middle of
the data field instead of at the right side as it should be. Graphically:

```
00 01 FF FF ... FF 00  ASN.1  HASH  GARBAGE
```

This gives him complete freedom to put anything he wants to the right of the hash. This gives him enough flexibility that he can arrange for the value to be a perfect cube.

There are some other variations of this attack, for example some implementations of the signature verification algorithm neglect to check if the field "parameters" inside "digestAlgorithm" field is NULL. In such a case an attacker may put some GARBAGE here, making the attack still possible even if the algorithm verifies that the HASH is right-justified.

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

An attacker has access to an oracle that, for any chosen ciphertext, indicates whether the corresponding plaintext $C^d \bmod n$ has the correct format according to the encryption standard.

Then after querying the oracle with about 2^{20} adaptively chosen ciphertexts the attacker is able to calculate the plaintext correpsonding to the original ciphertext.

The oracle might be for example a HSM that receives ciphertexts resulting from the key-wrap algorithm.

The attack exploits such vulnerabilities like

- different 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. FORMAL SECURITY PROOFS

Security model e.g. for PACE

Background (Hanzlik, MK)

data confidentiality: nobody can understand any data from the communication between an eID and a terminal, except for this eID and this terminal. By "data" we mean:

- workload data to be transmitted via the channel established according to the protocol,
- partner specific data (such as partner identity) - if sending them (explicitly or implicitly) results from the protocol execution.

data integrity: a third person cannot manipulate without detection the data exchanged between the eID and the terminal. This concerns in particular manipulating identity data.

session integrity: if a party A accepts at some moment a session executed presumably with a single partner, then indeed this interaction of A emerged in interaction with a single partner.

partner authentication: if a partner A accepts a session as a session withparty C, then A indeed has been talking with C until this moment (maybe with somebody playing man-in-the-middle, but only passively). Partner authentication might be mutual or one-sided. In case of PACE, there is one-sided authentication of an eID.

owner's consent: eID is used only when the user agrees and with the terminal chosen by the user.

proof non-transferability: a party A interacting with a party B cannot prove against a third party C that it is interacting with B, and cannot authenticate in this way the data received from B. This should be understood that executing the protocol does not provide additional cryptographic evidence over the data mentioned in the data confidentiality condition.

Case study: KEA

- Diffie-Hellman based key-exchange protocol, mutual authentication for the parties
- developed by NSA, declassified in 1998, no security analysis
- attacked in 2005, Lauter, Mityagin, extension KEA+ proposed, security proven by reduction proofs
- naive protocol:
  - party A chooses x at random and sends to B:
    - g^x and $\text{sign}_A(g^x, B)$
  - party B chooses y at random and sends to BA:
    - g^y and $\text{sign}_B(g^y, A)$
  - both: verify the signature, compute $g^{x \cdot y}$ as for DH protocol
  - attack:
    - if ephemeral x of A from communication between A and B revealed, then ...
    - the adversary resends g^x and $\text{sign}_A(g^x, B)$ to C and can impersonate A as he can compute the session key

- KEA:
  - A and B hold, respectively, the keys: private a and b, and public keys g^a and g^b
  - A and B select ephemeral secret keys x and y at random and exchange g^x and g^y
  - each party computes $g^{a \cdot y}$ and $g^{b \cdot x}$ (static DH protocol)
  - session key computed as $F(g^{a \cdot y} \text{ xor } g^{b \cdot x})$ (just like Blake-Wilson, D. Johnson, and A. Menezes: Hash($g^{a \cdot y}$, $g^{b \cdot x}$))
- Unknown Key Share (UKS) – a formal attack on KEA:
  - Mallet registers the same key g^a as Alice
  - Alice starts a session with Bob but session intercepted by Mallet
  - Mallet starts a session with Bob as Mallet
  - Mallet forwards the values g^x and g^y
  - therefore Alice and Bob compute the same session key
  - Mallet corrupts one sesson and get a session key for the second one - contradicting AKE security
- KEA+
  - session key computed as $F(g^{a \cdot y}, g^{b \cdot x}, A, B)$
- KEA+C
  - keys as for KEA
  - A chooses x at random and sends g^x
  - B chooses y at random, computes $L = \text{Hash}(g^{a \cdot y}, g^{b \cdot x}, A, B)$
  - B responds with g^y and $\text{MAC}_L(0)$
  - A computes L, checks $\text{MAC}_L(0)$and responds with $\text{MAC}_L(1)$
  - B checks $\text{MAC}_L(1)$
- security properties:
  - AKE (Authenticated Key Exchange)-
    - the adversary controls all communication
    - the adversary can corrupt some of the parties.
    - the adversary must select an uncorrupted session called a test session and then he is given a challenge, which is either the session key of the test session or a randomly selected key.
    - the adversary wins if can distinguish between these 2 cases.

- PFS (Perfect Forward Security):
  - AKE experiment
  - the adversary can corrupt a party A (reveal the long-term secret key),
  - test session: a session of A occurred before corrupting A
- KCI (Key Compromise Impersonation)
  - the adversary gets a long-term secret key of A
  - attempt to impersonate as other party to A
  - of course, the adversary can impersonate A to anyone
- advantage of the adversary A running algorithm $\mathcal{A}$:
  $|\Pr(\mathcal{A}(\text{data}, \text{real key}) = 1) - \Pr(\mathcal{A}(\text{data}, \text{random key}))|$
  the advantage should be "negligibly small"

- reduction proofs:
  - → assume that that there is an advarsary A breaking scheme U
  - → choose a cryptographic assumption P
  - → from a case p for P construct a case u for U
  - → show how to run A on u
    - → the environment need not to behave exactly as the scheme U
    - → the difference between real U and the simulated one should be impossible to detect by A
  - → breaking u should lead to breaking p with a fair probability
  - → finally: compute the advantage of the resulting adversary breaking p

- modelling via oracles:
  - atomic actions that can be initiated by the adversary
  - all interactions with the system defined by the oracles
  - specification of adversary's power

- typical oracles:
  - Reveal: reveal ephemeral key
  - Reveal: reveal session key
  - Corrupt: reveal long-time key
  - Execute(A, B): make A and B execute the protocol

- Send: send a message to A and get its reaction (if any) – the messages may come form the protocol, but might be faulty
- Test: a session ends after key establishment, no workload communication (this can be added with the tested key), must concern a *fresh session*
- *fresh session:* exclude situation where for instance via corruptions it is possible to break the session

- AKE for KEA+:
  - reduction via Gap Diffie-Hellman (CDH under assumption that DDH easy)
  - ROM for hash function
  - ways to distinguish between the random from real key: hash value must be asked
  - possibilities for the real key K to appear in the experiment:
    1. Forging: enforce Hash on the tuple $(\mathrm{CDH}(A,Y), \mathrm{CDH}(B,X), A, B)$
    2. Key-replication attack. succeed to create another session with the same "signature" $(\mathrm{CDH}(A,Y), \mathrm{CDH}(B,X), A, B\ )$ and so the same secret key
  - key replication: impossible, since $A' = A$ and $\mathrm{CDH}(A', Y') = \mathrm{CDH}(A, Y)$ implies $Y = Y'$. Similarly $X = X'$ and the sessions are identical
  - forging: case of a single session:
    - adversary observes a single session between honest A and B
    - problem GDH for (X_0, Y_0)
    - the long term key of A chosen as X_0, the response of B chosen as Y_0, the rest executed as in the scheme description
    - learning the key requires asking hash oracle about $(\mathrm{CDH}(X_0, Y_0), g^{b \cdot x}, A, B)$
  - forging the in general case: problem since A involved in many interactions but we do not know the secret key. Idea: replace with a random key
    - all users initialized according to the scheme, except for A
    - Hash simulated by HSim
    - sessions not involving A executed according to the protocol (and HSim)
    - a session $(\boldsymbol{A}, \boldsymbol{C}, \mathrm{role})$:
      - C public key of $\boldsymbol{C}$
      - if $\boldsymbol{A}$ initiator, then it chooses x at random, sends g^x, gets reply Y, session key $\mathrm{HSpec}(1, Y, C^x, A, C)$
      - if $\boldsymbol{A}$ responder, then it waits for X, chooses y at random, sends g^y, gets reply Y, session key $\mathrm{HSpec}(2, X, C^y, C, A)$

- a session $(\boldsymbol{C}, \boldsymbol{A}, \text{role})$:
  - as in the scheme description
  - except for test session where Y_0 sent and the session key not computed
- reveal and corrupt key: as described by the scheme
- $\text{HSim}(Z_1, Z_2, B, C)$ – random oracle on valid signatures
  - if asked before, then repeat the answer
  - check all previous $\text{HSpec}(i, Y, Z, B, C) = v$ and check if $Z = Z_{3-i}$ and $\text{DDH}(X_0, Y, Z_i) = \text{true}$. If yes, then return v.
  - if not found then return random w and remember it
- $\text{HSpec}(i, Y, Z, B, C)$ - random oracle for cases when adversary does not know the secret key of A. For input (Z_1, Z_2, B, C), where $Z_i = \text{CDH}(X_0, Y)$ and $Z_{3-i} = Z$

VII. 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, ...
- catastrophy is already there

TLS and DH real security

mistakes:

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

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

    - 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 ($\frac{\text{p-1}}{2}$ 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

VIII. 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)

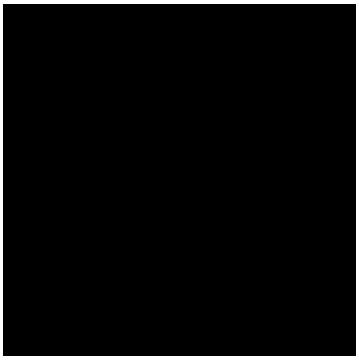


Figure 1.

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)

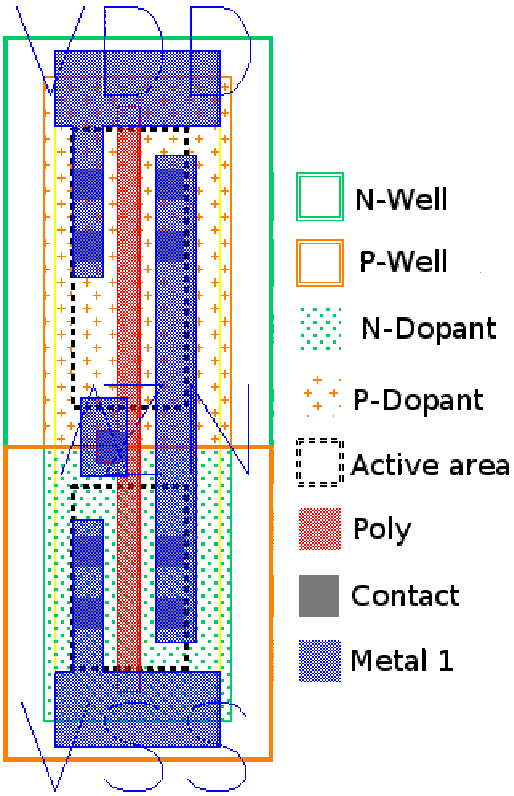
how it works:

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

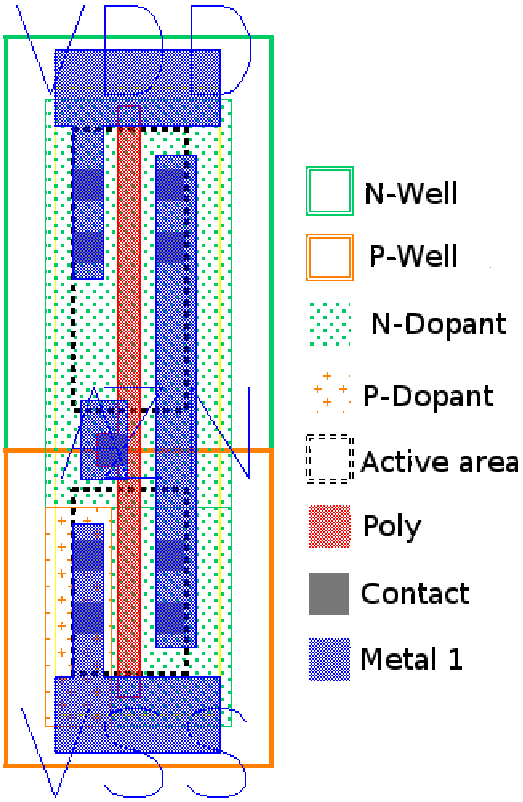
PMOS: in dopant area "holes" (positive) playing the role of conductor, low voltage creates depletion area, high voltage attracts them

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 - inline 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

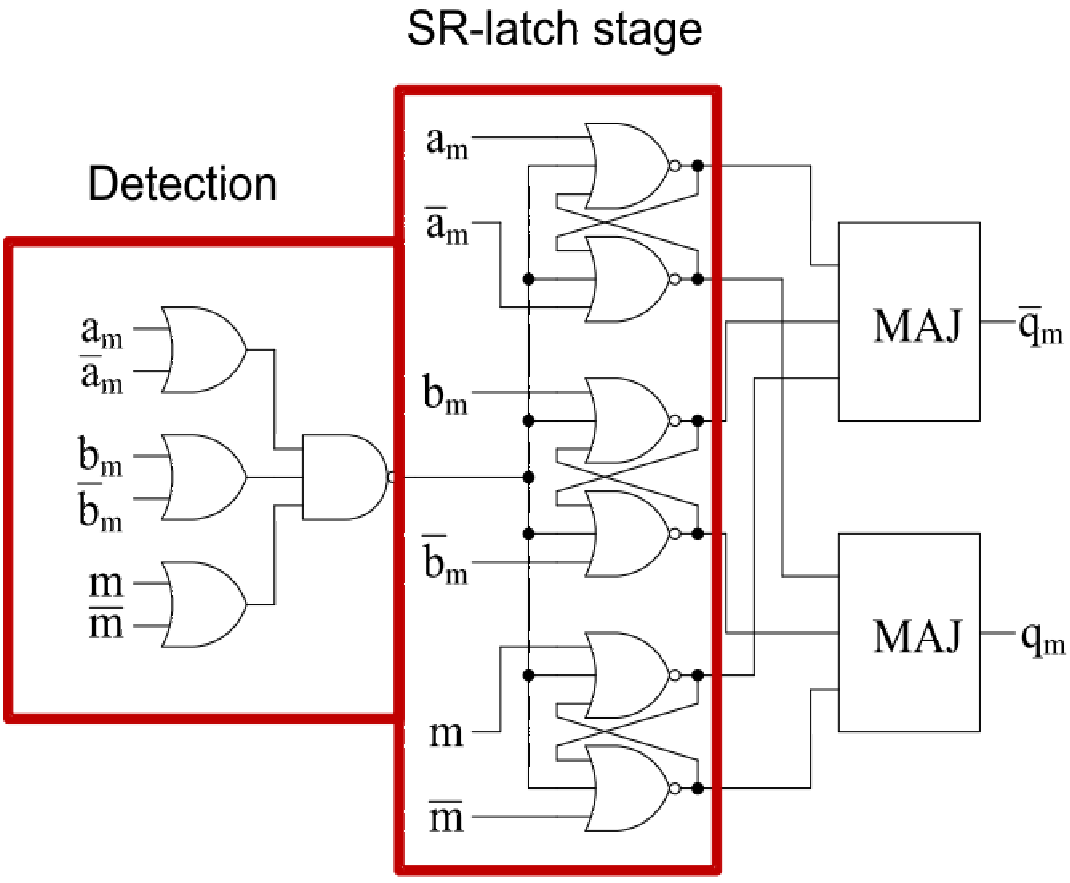
ii. precharge: each phase preceded by charging all gates

iii. masking operations by random numbers:

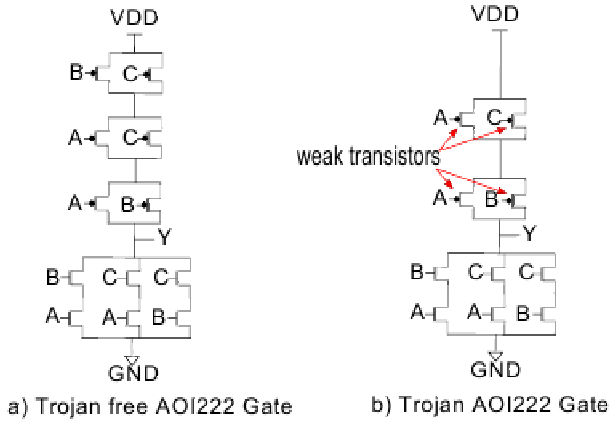
computing $a \wedge b$:

– input $a \oplus m$, $a \oplus \neg m$, $b \oplus m$, $b \oplus \neg m$, m, $\neg m$

– detection, SR-latch stage and majority gat



attacking not-majority 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

VIII. MODELLING UNSECURITY

attacks:

- hit-and-run
- hit-and-stay
- insider

life-cycle of signing keys:

1. T_0: key created
2. T_1 : forensics-based key non-compromised
3. T_2: key compromise (known to the adversary)
4. T_3 : forensics-based key already compromised
5. T_4: key ceased from operation

$[T_1,T_4]$ is a gray period. Should be as short as possible

DSAS framework

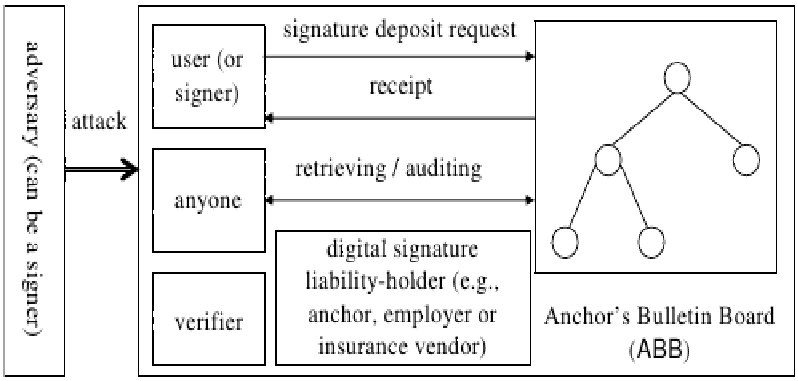
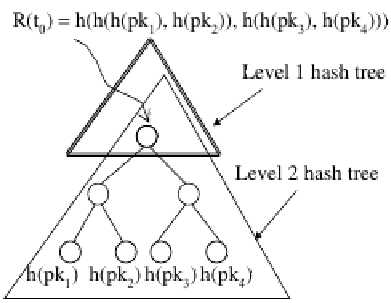
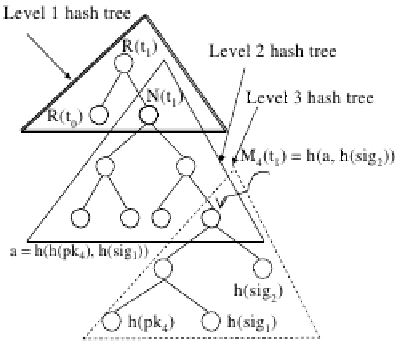


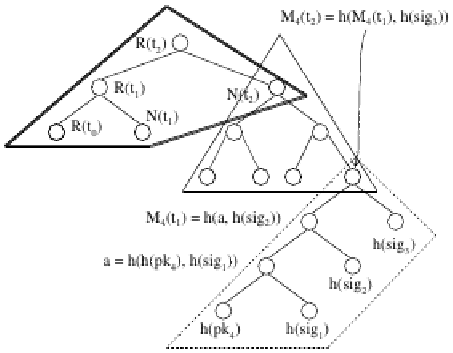
ABB:



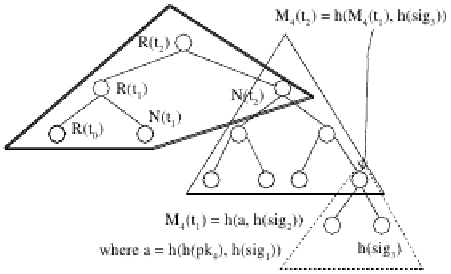
(a) ABB_0 at time t_0 (signatures-preserved case and signatures-compressed case): initialization with four user public keys



(b) ABB_1 at time t_1 (signatures-preserved case and signatures-compressed case): user pk_4 deposited two signatures



(c) ABB_2 at time t_2 (signatures-preserved case): user pk_4 deposited one new signature



(d) ABB_2 at time t_2 (signatures-compressed case): user pk_4 deposited one new signature