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:
  - i. **incomplete** – forgetting some threats is common
  - ii. **not embedded:** not corresponding really to the environment where the product has to be deployed
  - iii. **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 :

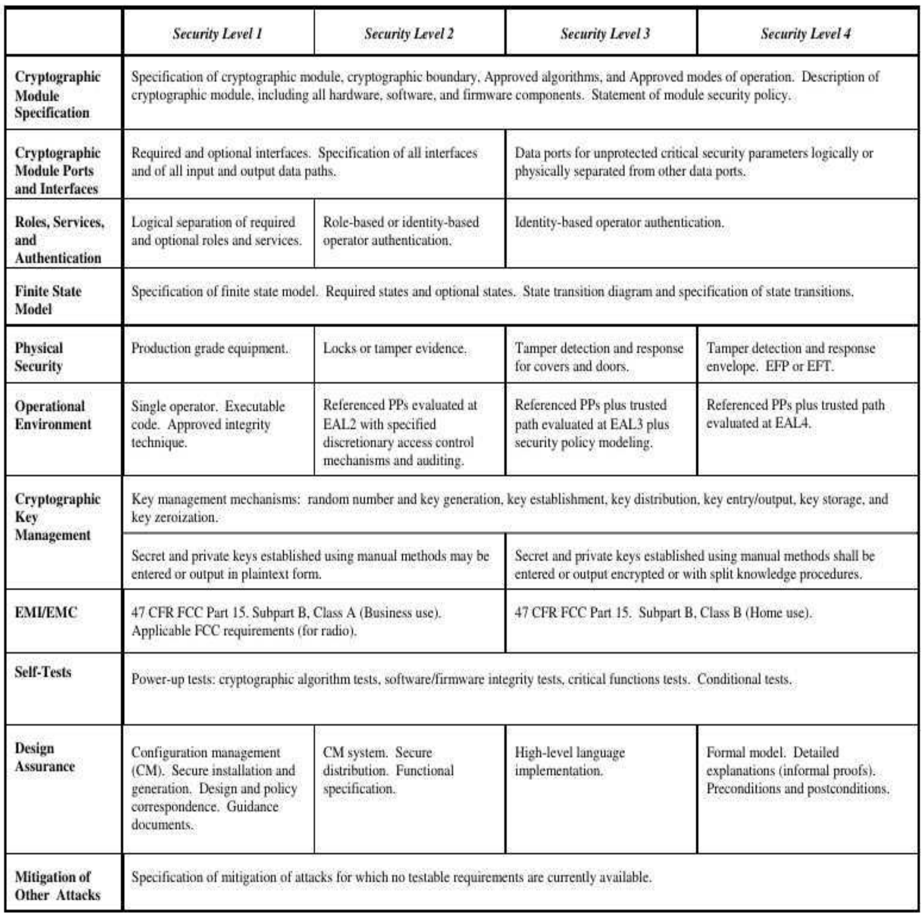
FIPS PUB 140-2, SECURITY REQUIREMENS FOR CRYPTOGRAPHIC MODULES

- 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:

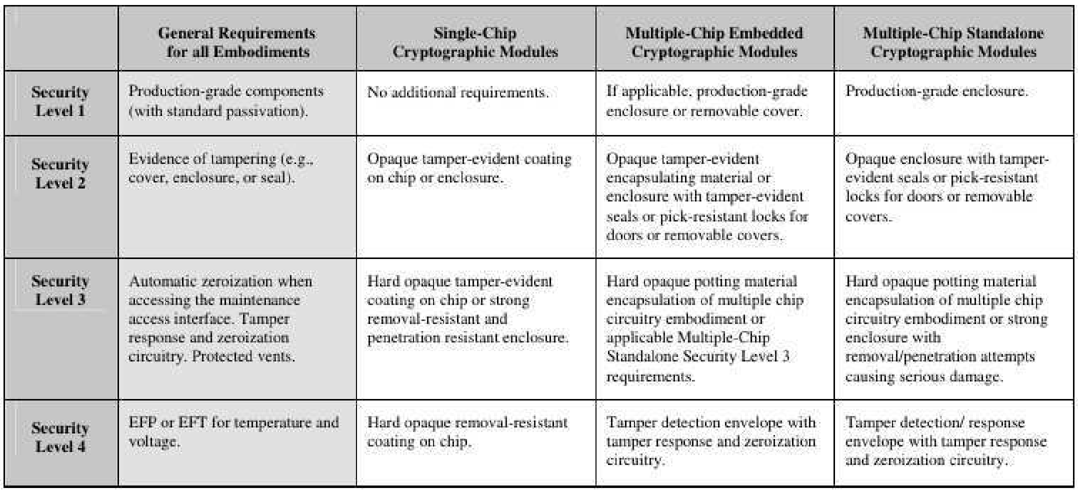


	Security Level 1	*Security Level 2*	*Security Level 3*	*Security Level 4*
Cryptographic Module Specification	Specification of cryptographic module, cryptographic boundary, Approved algorithms, and Approved modes of operation. Description of cryptographic module, including all hardware, software, and 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.

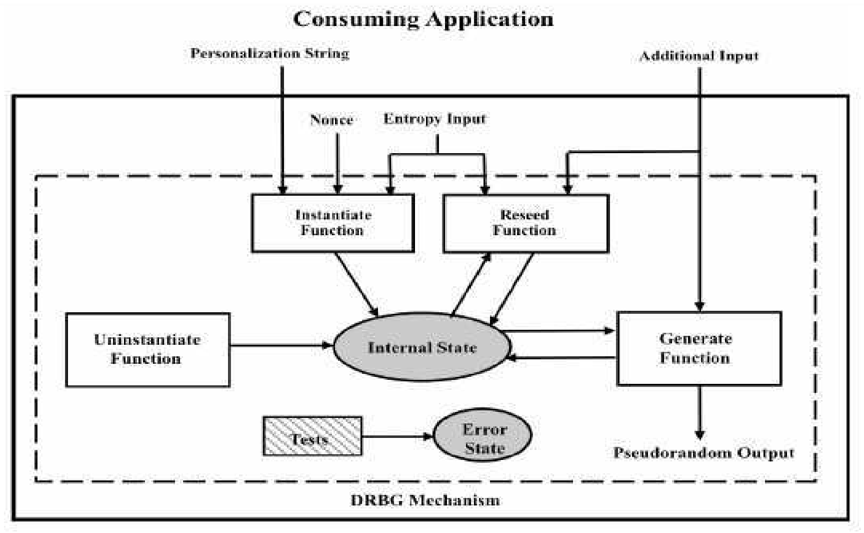
  - L3: EAL3, trusted path (also included in audit trail)
  - L4: EAL4

- 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`)

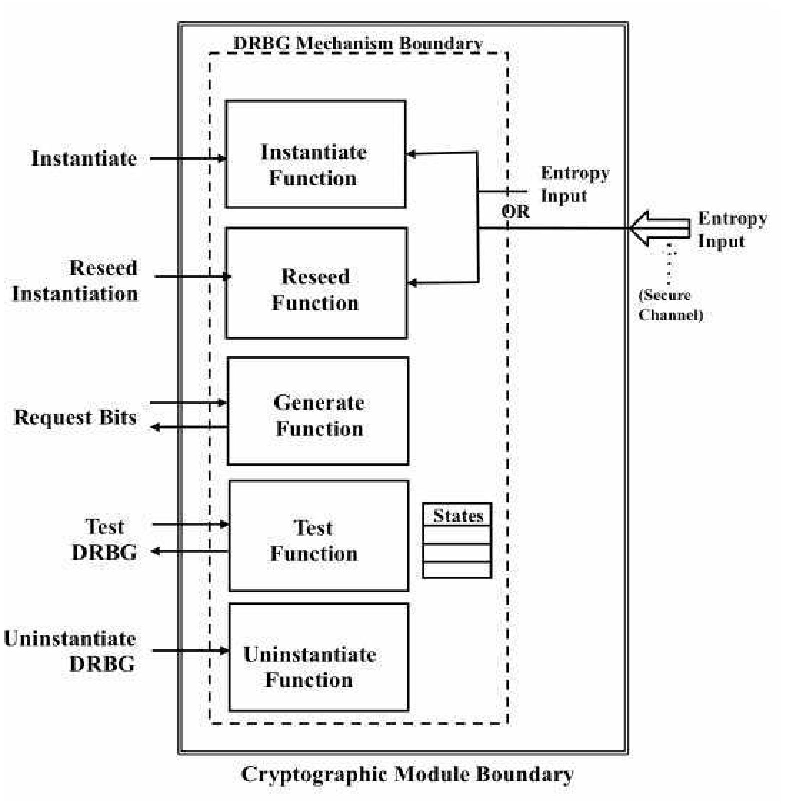
functions:

- 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 $2^{\text{seedlen}}$
3. (returned_bits) = Hashgen (requested_number_of_bits, V)
4. H = Hash (0x03 || V)
5. V = (V + H + C + reseed_counter) mod $2^{\text{seedlen}}$
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 $\oplus$ provided_data
5. Key = leftmost (temp, keylen)
6. V = rightmost (temp, blocklen).
```

Instantiate:

```
1. pad personalization_string with zeroes
2. seed_material = entropy_input $\oplus$ personalization_string
3. Key = $0^{keylen}$
4. V = $0^{blocklen}$
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).
```

A dark side of standards - NIST SP800-90 Dual EC PRNG case

- history of a PRNG from NIST:
  - 2004 draft
  - 2005 patent application
  - 2005 part of ISO/IEC 18031
  - 2006 publications starting evaluate from cryptographic pont of view (indistinguishability from random source)
  - 2007 Dan Shumow and Niels Ferguson show the possibility of a backdoor
  - 2013 NIST finally does not recommend to use SP800-90 Dual EC PRNG
  - 2014 NIST removed it from guide, "...current users of Dual_EC_DRBG transition to one of the three remaining approved algorithms as quickly as possible"

- the algorithm:
  - specific eliptic curve, and points P and Q (chosen by NIST)
  - for a point $S=(x,y)$ of the curve, $\phi(S)=x$
  - one step starting in state s_i:
    1. $r_i=\phi(s_i \cdot P)$
    2. $s_{i+1}=\phi(r_i \cdot P)$
    3. $t_i=\phi(r_i \cdot Q)$
    4. truncate some number of most significant bits of t_i and output the result

- backdoor:
  - assume you know x such that $P=x \cdot Q$
  - you know the output which is t_i except for a few truncated bits (16?)
  - reconstruct the truncated bits and then using the definition of the curve a point U such that $\phi(U)=t_i$,
  - compute $s=\phi(x \cdot U)$, (if $U=r_i \cdot Q$ then $x \cdot U=x \cdot r_i \cdot Q=r_i \cdot x \cdot P=r_i \cdot P=s_{i+1}$)
  - you can check the alleged internal state s against the next output bits

---

ACCESS CONTROL

- decision whether a subject (user, etc) is allowed to carry out a specific action (operation) on an object (resource)
- policy – set of rules of (axioms + derivation rules) for making the decision
- main models:
  - Access Control Matrix or Access Matrix (AM)
  - Access Control List (ACL)
  - Role-Based Access Control (RBAC)
  - Attribute-Based Access Control (ABAC)
- principles:
  - least priviledge