PV204 Security technologies

Secure authentication and authorization

Petr Švenda svenda@fi.muni.cz
Faculty of Informatics, Masaryk University

CROCS
Centre for Research on Cryptography and Security

Overview

- Authentication and key exchange protocols
- Problems and design principles
- Authentication protocols in electronic passports

SECURITY PROTOCOLS

Security protocols

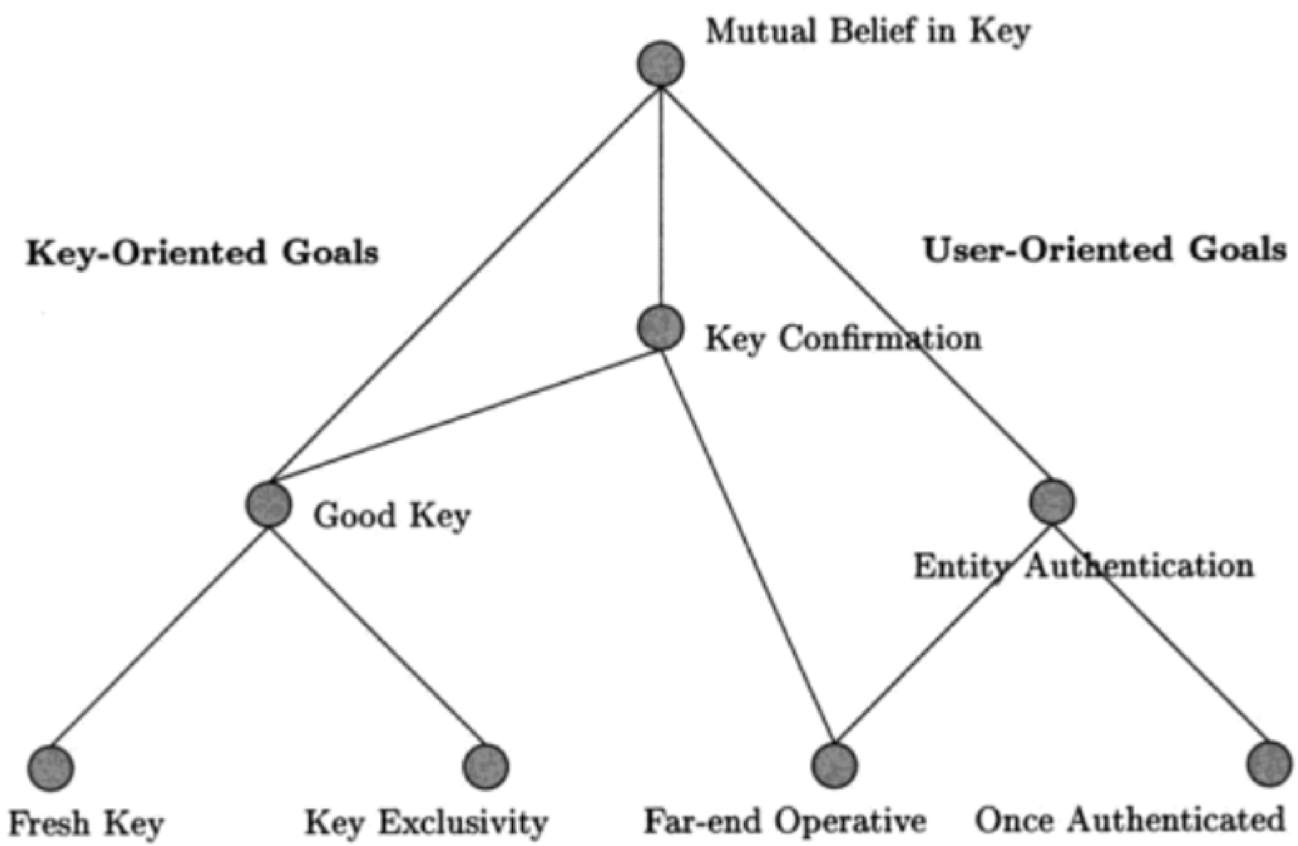
- Security protocol = composition of cryptoprimitives

- *“Security protocols are three line programs that people still manage to get wrong.” (R. Needham)*

Authentication (AUTH) vs. Key establishment (KE)

- Early literature called protocols used to establish session keys as “authentication protocols”
- Authentication is also possible without session keys
  - Example: Challenge-response, active authentication
- Session keys can be established without authentication
  - Example: non-authenticated Diffie-Hellman

Hierarchy of AUTH&KE goals



Protocols for Authentication and Key Establishment *By Colin Boyd, Anish Mathuria*

Entity authentication

- Entity = user, machine/device
- Something entity knows (password, key…)
- Something entity is (biometrics…)
- Something entity have (smartcard…)
- Multi-factor authentication
  - More than one factor (password + smartcard)
  - Aim to increase attacker's cost to compromise multiple security layers (factors)

Methods for key establishment

1. Derive from pre-shared secret (KDF)
2. Establish with help of trusted party (Kerberos, PKI)
3. Establish over insecure channel (Diffie-Hellman)
4. Establish over other (secure) channel
5. Establish over non-eavesdropable channel (BB84)
6. …

Methods for key confirmation

- Goal: ensure that parties use same key value(s)
- Implicit confirmation by use of valid key
  – E.g., MAC by session key on future message is valid
- Explicit confirmation by challenge-response
  – Dedicated steps in protocol

Option	Alice	Bob
1	$R_1 = \text{random}()$	$\text{random}() = R_2$
	$E_{K'}(R_1) \longrightarrow$	
	$\longleftarrow E_{K'}(R_1, R_2)$	
	$E_{K'}(R_2) \longrightarrow$	
2	$H(H(K')) \longrightarrow$	
	$\longleftarrow H(K')$	

http://www.themccallums.org/nathaniel/2014/10/27/authenticated-key-exchange-with-speke-or-dh-eke/

PROTOCOLS AND ATTACKS

Typical models of adversary

- Adversary controls the communication
  - Between all principals
  - Observe, alter, insert, delay or delete messages
- Adversary can obtain session keys
  - used in previous runs
- Malicious insider
  - adversary is legitimate protocol principal
- Attacker can obtain partial knowledge
  - Compromise or side-channels
- …

Needham–Schroeder protocol: symmetric

- Basis for Kerberos protocol (AUTH, KE), 1978
  - Two-party protocol (A,B) + trusted server (S)
  - Session key K_{AB} generated by S and distributed to A together with part intended for B
  - Parties A and B are authenticated via S

Which part ensures:
Authentication
Key confirmatic
Freshness

1. $A \rightarrow S$: A, B, N_A
2. $S \rightarrow A$: $\{N_A, K_{AB}, B, \{K_{AB}, A\}K_{BS}\}K_{AS}$
3. $A \rightarrow B$:
4. $B \rightarrow A$: $\{N_B, A\}K_{AB}$
5. $A \rightarrow B$: $\{N_B - 1\}K_{AB}$

Can you spot problem?

N-S symmetric: Problem?

- Vulnerable to replay attack (Denning, Sacco, 1981)
- If an attacker compromised older K_{AB} then
  - $\{K_{AB}, A\}K_{BS}$ can be replayed to B (step 3.)
  - B will not be able to tell freshness
  - Attacker will then impersonate A using old (replayed, compromised) key K_{AB}
- Fixed by inclusion of nonce/timestamp $\mathbf{N'_B}$ generated by B (two additional steps before step 1.)
  - Bob can now check freshness of $\{K_{AB}, A, \mathbf{N'_B}\}K_{BS}$

What is required attacker model?

What is required attacker model?

- Able to capture valid communication ($\{K_{AB}, A\}K_{BS}$)
- Able to compromise older K_{AB}
- Actively communicate with B (reply ($\{K_{AB}, A\}K_{BS}$)

But is assumption of compromise of old key realistic?

How (not) to reason about potential compromise

- NO: all my (many) keys are in secure hardware and therefore I’m secure (no compromise possible)
  – Nothing like perfect security exists

- YES: assume compromise and evaluate impact
  – Where are sensitive keys
  – How hard is to compromise them
  – What will be the impact of the compromise
  – Can I limit number/exposure of keys? For what price?

What if key is compromised?

- Prevention, detection (hard), reaction
- Prevention of compromise
  - Limit usage of a key
    - master key $\rightarrow$ session keys
    - Use PKI instead of many symmetric keys in trusted terminals
  - Limit key availability
    - Erase after use, no/limited copy in memory, trusted element
  - Limited-time usefulness of keys (key update)
    - (Perfect) forward secrecy: Information before is secure
- Reaction on compromise
  - stop using key, update and let know (revocation)

Needham–Schroeder protocol: asymmetric

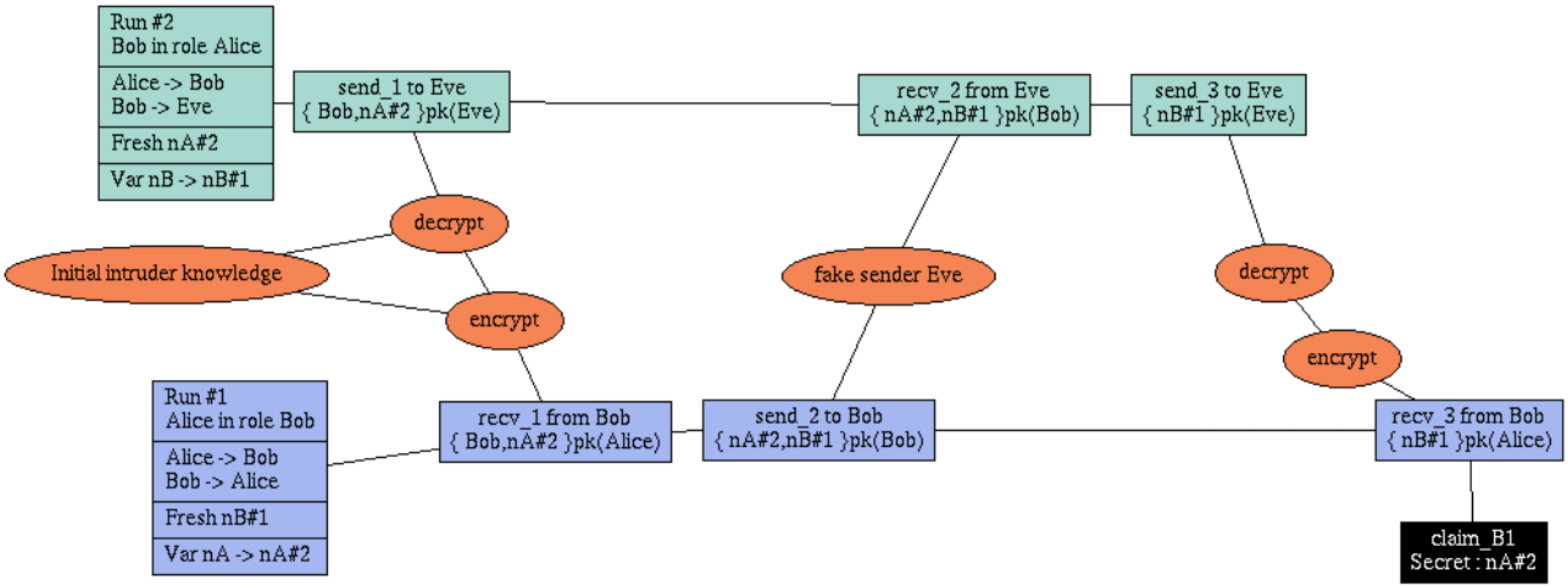
- Simple asymmetric AUTH & KE protocol
- Designed by R. Needham and M. Schroeder (1978)

1. $A \rightarrow B$: $\{A, N_A\}PK_B$
2. $B \rightarrow A$:
3. $A \rightarrow B$: $\{N_B\}PK_B$

Which part ensures:
Authentication
Freshness
Key establishment

Can you spot the problem?

N-S asymmetric: Problem?



- Discovered by G. Lowe 17 years after using formal verification method/tool

Formal verification of protocols

Negatives

- Specific attacker model
  - Different attacker (e.g., side-channels) => attack possible
- Assumes perfect crypto-primitives
- Sensitive to precise specification
- Hard to express real-world complex protocols
  - Search space too large

Positives

- Automated process
- Prevents basic and some advanced design flaws
- Favours simple solutions
  - Complexity is enemy of security

s formal verification panacea?

References

- Security Protocols Open Repository
  - http://www.lsv.ens-cachan.fr/Software/spore/
- C. Cremer, Scyther tool
  - https://github.com/cascremers/scyther/
- Cas Cremer's exercise sheet
  - https://www.cs.ox.ac.uk/people/cas.cremers/scyther/scyther-exercises.html

N-S asymmetric: Fix

- Fixed by addition of B's identity into second step

1. $A \rightarrow B$: $\{A, N_A\}PK_{(B)}$
2. $B \rightarrow A$: $\{B, N_A, N_B\}PK_{(A)}$
3. $A \rightarrow B$: $\{N_B\}PK_{(B)}$

AUTHENTICATED KEY EXCHANGE

Diffie-Hellman key exchange

Which part ensures:
Key establishmen
Key confirmation
Authentication

Diffie-Hellman Key Exchange

Step	Alice	Bob
1	Parameters: p, g	
2	$A = \text{random}()$ $a = g^A \pmod p$	$\text{random}() = B$ $g^B \pmod p = b$
3	$a \longrightarrow$ $\longleftarrow b$	
4	$K = g^{BA} \pmod p = b^A \pmod p$	$a^B \pmod p = g^{AB} \pmod p = K$
5	$\longleftarrow E_K(data) \longrightarrow$	

http://www.themccallums.org/nathaniel/2014/10/27/authenticated-key-exchange-with-speke-or-dh-eke/

Diffie-Hellman in practice

- K is not used directly, but K' = KDF(K) is used
  1. Original K may have weak bits
  2. Multiple keys may be required (K_{ENC}, K_{MAC})
- Is vulnerable to man-in-the-middle attack (MitM)
  – Attacker runs separate DH with A and B simultaneously
  – (Unless a and b are authenticated)
- DH can be used as basis for *Forward secrecy*
- DH can be used as basis for *Password-Authenticated Key Exchange*

Forward secrecy - motivation

- Assume that session keys are exchanged using long-term secrets
  1. Pre-distributed symmetric cryptography keys (SCP'02)
  2. Public key cryptography (TLS_RSA_...)
- What if long-term secret is compromised?
  I. All future transmissions can be read
  II. Attacker can impersonate user in future sessions
  III. All previous transmissions can be compromised if traffic was captured
- Can III. be prevented? (Forward secrecy)

Forward secrecy – how to achieve

- (Perfect) Forward Secrecy
  - Compromise of long-term keys does not compromise past session keys
- Solution: ephemeral key pair (DH/RSA/…)
  1. Fresh keypair generated for every new session
  2. Ephemeral public key used to exchange session key
  3. Ephemeral private key is destroyed after key exchange
     - Captured encrypted transmission cannot be decrypted
- Long-term key is used only to authenticate ephemeral public key to prevent MitM

Use of forward secrecy: examples

- TLS (DHE-RSA, DHE-DSA, ECDHE-RSA, ECDHE-ECDSA…)
- SSH (RFC 4251)
- Off-the-Record Messaging (OTR) protocol
- Axolotl protocol (TextSecure)
- …

Example: Off-The-Record Messaging (OTR)

- Protocol for protection of instant messaging
  - Perfect forward secrecy (via use of DH)
  - OTR ratcheting (new DH generated and advertised for every message)
  - Plausible deniability of messages (via MAC key broadcast)
- Read more
  - M. Green, http://blog.cryptographyengineering.com/2014/07/noodling-about-im-protocols.html
  - TextSecure: https://whispersystems.org/blog/advanced-ratcheting/

PASSWORD-AUTHENTICATED KEY EXCHANGE (PAKE)

PAKE protocols - motivation

- Diffie-Hellman can be used for key establishment
  - Authentication ca be added via pre-shared key
- But why not directly derive session keys from pre-shared instead of running DH?
  1. Compromise of pre-shared key => compromise of all data transmissions (including past) => no forward secrecy
  2. Pre-shared key can have low entropy (password) => attacker can brute-force
- Password-Authenticated Key Exchange (PAKE)
  - Sometimes called Escalation protocols

PAKE protocols - principle

- Goal: prevent MitM <u>and</u> offline brute-force attack

1. Generate asymmetric keypair for every session
   - Both RSA and DH possible, but DH provides better performance in keypair generation
2. Authenticate public key by (potentially weak) shared secret (e.g., password)
   - And limit number of failed authentication requests!
3. Exchange/establish session keys for symmetric key cryptography using authenticated public key

Diffie-Hellman Encrypted Key Exchange

Step	Alice	Bob
1	Shared Secret: $S = H(password)$	
2	Parameters: p, g	
3	$A = \text{random}()$ $a = g^A \pmod{p}$	$\text{random}() = B$ $g^B \pmod{p} = b$
4a	$E_S(a) \longrightarrow$ $\longleftarrow E_S(b)$	
4b	$a \longrightarrow$ $\longleftarrow E_S(b)$	
4c	$E_S(a) \longrightarrow$ $\longleftarrow b$	
5	$K = g^{BA} \pmod{p} = b^A \pmod{p}$	$a^B \pmod{p} = g^{AB} \pmod{p} = K$
6	$\longleftarrow E_K(data) \longrightarrow$	

DESIGN OF PROTOCOLS

Design of cryptographic protocols

- Don’t design own cryptographic protocols
  - Use existing well-studied protocols (TLS, EAC-PACE…)
  - Don’t remove “unnecessary” parts of existing protocols
- Don’t implement on your own (if possible)
  - Potential for error, implementation attacks…
- Follow all required checks on incoming messages

Design principles (Abadi & Needham) I.

- The conditions for a message to be acted should be clearly set out so reviewer can judge if they are acceptable.
  - Documentation, diagrams, formal specification
- Every message should say what it means, message interpretation should depend only on its content.
  - “This is 2nd message of SCP’02 from A to B”
  - No assumptions like next random chunk number should be encrypted 2nd message because I just received 1st message
- Mention name of principal (“Alice01”)
  - Prevents (if checked) unintended parallel runs of protocol
  - Prevents reflection attack

Design principles (Abadi & Needham) II.

- Be clear about why encryption is being done
  - For confidentiality, not to "somewhat" ensure integrity
- When signing encrypted data, it should not be inferred that signing entity knows data content
  - No knowledge of encryption key
- Be clear about properties of nonce
  - random, never repeated, unpredictable, secret
  - Random $\rightarrow$ almost never repeated unintentionally

Design principles (Abadi & Needham) III.

- If predictable quantity is to be effective, it should be protected so that an intruder cannot simulate a challenge and later replay the message
  - Counter as challenge $\rightarrow$ counter freshness verification necessary $\rightarrow$ state
- If timestamps are used as freshness guarantees, then difference between local clocks at various machines must be much less then allowable age of message
  - Otherwise an attacker can replay within time window
- Key may have been used recently and yet be old and possibly compromised
  - Clear session state after session end, check freshness

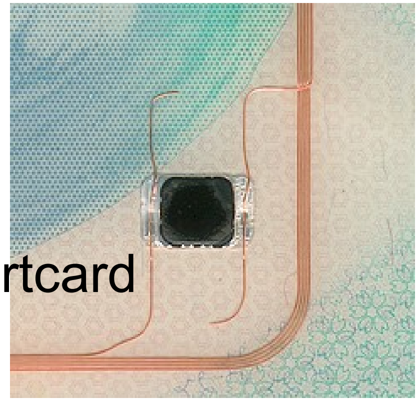
Design principles (Abadi & Needham) IV.

- It should be possible to deduce which protocol and which run of that protocol a message belongs to including order number in the protocol
  - Danger of parallel runs of same protocol
  - MAC and chaining with fresh session keys prevents message mixing
- Trust relation should be made explicit and there should be good reason for its necessity.
  - Less trust needed → better security achieved

ELECTRONIC PASSPORTS AND CITIZEN ID CARDS

Credit: Slides partially based on presentation by Zdenek Říha

Passports of the first generation



- Electronic passport
  - Classical passport booklet + passive contactless smartcard (ISO14443, communication distance 0-10 cm)
  - Chip & antenna integrated in a page or cover
- Technical specification standardized by ICAO
  - Standard 9303, 6th edition
  - References many ISO standards
- Data is organised in 16 data groups (DG) and 2 meta files
  - DG1-DG16, EF.COM, EF.SOD
  - Mandatory is DG1 (MRZ), DG2 (photo), EF.COM and EF.SOD (passive authentication)

Chip and antenna

Data groups

Data group	Stored data
DG1	**Machine readable zone (MRZ)**
DG2	**Biometric data: face**
DG3	Biometric data: fingerprints
DG4	Biometric data: iris
DG5	Picture of the holder as printed in the passport
DG6	Reserved for future use
DG7	Signature of the holder as printed in the passport
DG8	Encoded security features – data features
DG9	Encoded security features – structure features
DG10	Encoded security features – substance features
DG11	Additional personal details (address, phone)
DG12	Additional document details (issue date, issued by)
DG13	Optional data (anything)
DG14	Data for securing secondary biometrics (EAC)
DG15	Active Authentication public key info
DG16	Next of kin

Protocols used in ePassports I.

I. Authentication of inspection system to chip [BAC]
   - Read basic digital data from chip (MRZ, photo)
   - SG: Passport provides basic data only to local terminal with physical access to passport
   - S: Auth. SCP, sym. crypto keys derived from MRZ [BAC]

II. Authorized access to more sensitive chip data
   - SG: Put more sensitive data on chip (fingerprint, iris), but limit availability only to inspection systems of trustworthy countries
   - S: Challenge-response auth. protocol [EAC,EAC-PACE], PKI + cross-signing between trustworthy states [EAC]

Protocols used in ePassports II.

III. Genuine data on passport
   - SG: Are data on passport unmodified?
   - S: digital signatures, PKI [passive authentication]

IV. Authentication of chip to inspection system
   - SG: Is physical chip inside passport genuine?
   - S: Challenge-response authentication protocol [AA, EAC-PACE]

V. Transfer data between chip and IS securely
   - SG: attacker can't eavesdrop/modify/replay
   - S: secure channel [EAC, EAC-PACE]

Authorization in passports

1. Inspection terminal to read basic info from chip
2. Inspection terminal to read biometric data from chip
3. You to enter country based on chip data

Basic Access Control (BAC) protocol

- Authentication&secure channel between inspection terminal and chip
  - Based on symmetric crypto (3DES), similar to SCP'0x protocols
  - Low computational requirements
- Problem: anyone with access to MRZ can authenticate
- Problem: MRZ has insufficient entropy
  - Document number, birth date, expiration date used
  - Theoretically 58/74 bits, but in practice about 32 bits
- Offline attack (eavesdrop then crack)
  - Eavesdrop valid communication between chip and reader
  - Brute-force attack in less then hour (2^{32} ops, offline attack)
- Online attacks against chip (att. model: found passport)
  - Significantly slower, ~20 ms for every attempt

EAC – motivation

- EU passports stores fingerprints (from 2009)
  - More sensitive than facial photo => better protocol needed
- Goal: not everyone with access to passport (and MRZ for BAC) should be able to read out fingerprint
  - Issuing country decides who else can access
- Stronger authentication than BAC required

Mind exercise: symmetric crypto

- What if only symmetric crypto is used?
  - Every chip has own unique symmetric key
  - Large number of keys in inspection terminals
  - Compromise of single terminal breach security
  - => impractical and insecure => not used

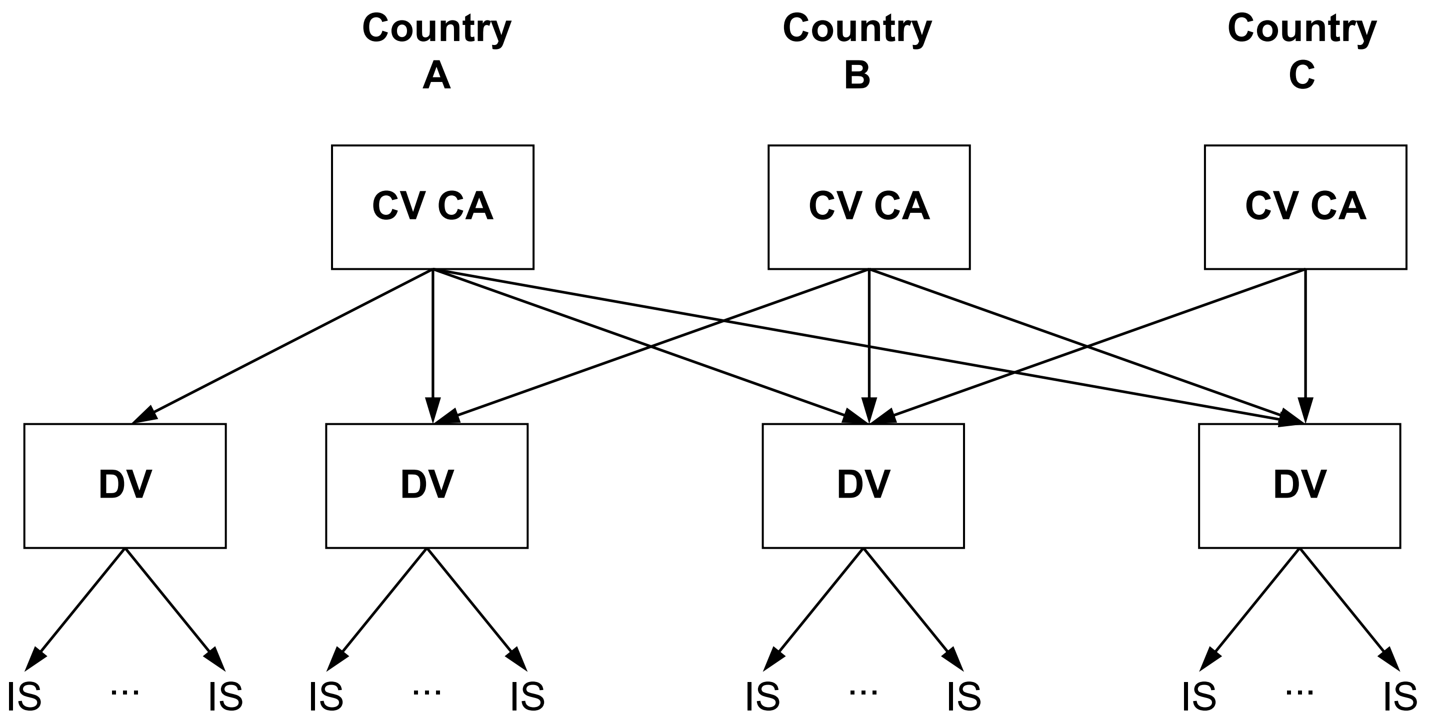
Extended Access Control (EAC) protocol

- Based on asymmetric cryptography (RSA/DH/ECDSA)
- Chip Authentication (CA) based on PACE protocol
  - Password Authenticated Connection Establishment (PACE)
  - Uses chip's static DH/ECDSA key and terminal's ephemeral DH key pair (perfect forward secrecy)
  - Both parties combines chip's public static and ephemeral public key into same key K
  - Keys for encryption and MAC (K_{ENC}, K_{MAC}) are derived from exchanged K
- How can be Terminal sure of authenticity of chip's static key?
  - Signed by Issuing country
- Terminal Authentication (TA)
  - Based on challenge-response protocol (RSA/ECDSA, SHA-1/2)
  - Hash of the ephemeral DH key from previous step hashed with challenges

Terminal authentication I.

- Only authorized border authorities can read the secondary biometric data (fingerprints and iris)
  - The inspection system must prove to the chip it is authorized
  - The chip stores a trust point – root certificate
  - Inspection system presents a valid certificate chain (starting from the passport's trust point) specifying the IS's authorizations (e.g. to read DG3)
  - Challenge-response where IS proves knowledge/access of a secret key (whose public part is certified)
  - Certificates in Card-verifiable (CV) format

Terminal Authentication II



Root public key of issuing country: CV CA

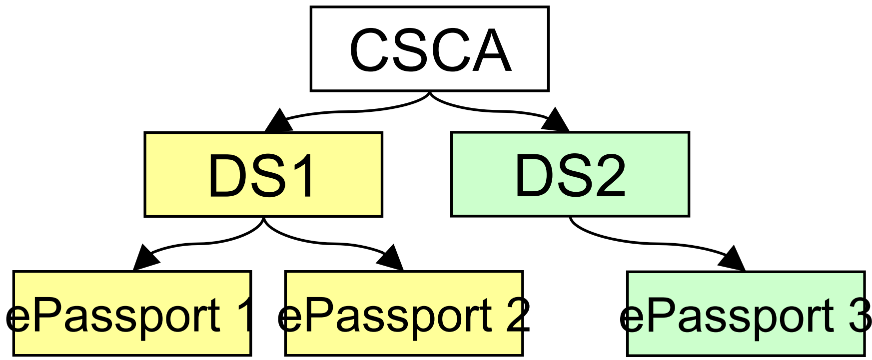
Terminal Authentication III

- Supported algorithms
  - RSA PKCS#1 v1.5 with SHA-1 or SHA-256
  - RSA PSS with SHA-1 or SHA-256
  - ECDSA with SHA-1, SHA-224 or SHA-256
  - Key lengths
    - For ECDSA allowed 160, 192, 224, 256 bits
    - For RSA allowed 1024, 1280, 1536, 2048 and 3072 bits
    - In practice ECDSA is more common, key lengths 192 and 224 bit most popular, existing implementations also support 256 and 384 bits.
    - For RSA the PKCS#1 v1.5 padding is much more popular than PSS, key lengths are between 512 (test only) and 2048 bits.

Active Authentication (AA) protocol

- Motivation: Prevent cloning of passport
  - Is chip inside passport authentic?
- Passport-specific asymmetric key stored on chip
- Public key freely readable (DG15 file, hash signed)
- Authentication against terminal
  - Terminal generates 8 random bytes
  - Chip adds additional 8 random bytes, hash and sign
  - Terminal verifies signature
- Privacy attack: terminal's challenge is date $\rightarrow$ signed
- PACE protocol replaces Active Authentication

Passive Authentication



- Goal: are data in chip unchanged?
- The list of the hashes (SHA-1/2) of all present data groups is digitally signed by the issuing organisation (Document Signer)
  - State printing house, Embassy, Etc.
- The X.509 certificate of the Document Signer issued by the CA of the issuing country (Country Signing CA – e.g. the ministry of interior) is included.
- The CSCA certificates must be exchanged bilaterally
- ICAO PKD for DS certificates, CSCA CRL and cross certificates
- Passive authentication is a **mandatory** security feature of all ePassports

Conclusions

- Design of (secure) protocols is very hard
  - Understand what are your requirements
  - Use existing protocols, e.g., TLS or EAC-PACE
- Strong session keys established with weak passwords
  - Password-Authenticated Key Exchange
- Electronic passport uses variety of protocols
  - Interesting and complex usage scenarios
- Mandatory reading
  - M. Green, Noodling about IM protocols, http://blog.cryptographyengineering.com/2014/07/noodling-about-im-protocols.html