



TBSS

(Telematic Base Security Services)

**Approved procedures and mechanisms for the
protection of electronic data communications.**

IBO 920 353 12.96
Version 1.2 English

6. December 1996

Preface

The TBSS (Telematic Base Security Services) is the result of efforts by the Swiss banks to create a standardized security platform for Switzerland's finance industry that simultaneously caters for current Videotex and EDIFACT requirements. This document contains the Swiss position and is a basis for national and international multi-sectorial harmonisation.

Telekurs was asked to take charge of project management.

The TBSS was defined by a group of experts of the Swiss Banks.

The TBSS is a base document for the Swiss financial industry and will be expanded on demand. Application-specific implementation guidelines shall reference the required mechanisms and procedures by TBSS version number, chapter and title.

Copyright by Telekurs, 6. December 1996.

Contents

| | |
|--|------------|
| 1 OBJECTIVES | 1-1 |
| 2 CONTENT | 2-1 |
| 2.1 Cryptographic algorithms as basic elements | 2-2 |
| 2.2 Modes of operation | 2-2 |
| 2.3 Combinations of mechanisms | 2-3 |
| 2.4 Protocol mechanisms | 2-3 |
| 3 SECURITY PROCESSING | 3-1 |
| 3.1 Data authentication | 3-1 |
| 3.1.1 Integrated security | 3-1 |
| 3.1.2 Data authentication with independent security message | 3-2 |
| 3.2 Data encryption | 3-3 |
| 3.3 Combined data authentication and encryption | 3-3 |
| 4 SECURITY SERVICES | 4-1 |
| 4.1 Non-repudiation | 4-1 |
| 4.1.1 Non-repudiation of origin (NRO) | 4-1 |
| 4.1.2 Non-repudiation of receipt (NRR) | 4-1 |
| 4.2 Entity authentication | 4-2 |
| 4.3 Confidentiality | 4-2 |
| 4.4 Relationship between security services and security mechanisms | 4-2 |
| 4.4.1 Non-repudiation | 4-2 |
| 4.4.2 Entity authentication | 4-3 |
| 4.4.3 Confidentiality | 4-3 |
| 4.4.4 Key management | 4-3 |
| 4.4.4.1 Key transport | 4-3 |
| 4.4.4.2 Public key transport | 4-4 |
| 4.4.5 Support mechanisms | 4-4 |
| 4.4.5.1 Hash functions | 4-4 |
| 5 DIGITAL SIGNATURE | 5-1 |
| 5.1 Overview | 5-1 |
| 5.1.1 Digital signature 1 (DS1) | 5-1 |

| | |
|--|-------------|
| 5.1.2 Hash function (HF) | 5-2 |
| 5.1.3 Digital signature with data hashing (DSH) | 5-2 |
| 5.2 Use of the digital signature | 5-2 |
| 5.2.1 Generating a signed message | 5-2 |
| 5.2.2 Signature verification | 5-3 |
| 5.3 DS1 - Digital signature complying with ISO/IEC 9796 | 5-3 |
| 5.3.1 ISO/IEC 9796 with a fixed exponent | 5-4 |
| 5.3.2 ISO/IEC 9796 with free exponent | 5-4 |
| 5.3.3 Key generation for DS1 | 5-5 |
| 5.3.3.1 Public verification exponent | 5-5 |
| 5.3.3.2 Prime number generation | 5-5 |
| 5.3.3.3 Modulus | 5-5 |
| 5.3.3.4 Secret signature exponent | 5-5 |
| 5.4 DSH - Digital signature with data hashing | 5-5 |
| 5.4.1 Signature process | 5-6 |
| 5.4.2 Verification process | 5-6 |
| 5.5 Hash function | 5-7 |
| 5.5.1 Recommendations | 5-8 |
| 5.5.2 MD strengthening | 5-8 |
| 5.6 HF1 - ISO/IEC 10118-2 with DES | 5-8 |
| 5.6.1 Key transformations u and u' | 5-9 |
| 5.6.2 Initialization vectors | 5-9 |
| 5.6.3 Padding rule | 5-9 |
| 5.7 HF2 - ISO/IEC 10118-3 RIPEMD-128 | 5-9 |
| 5.7.1 Hash-code computation | 5-10 |
| 5.7.2 Padding rule | 5-10 |
| 5.7.3 Round function and initializing value | 5-10 |
| 5.8 HF3 - ISO/IEC 10118-3 RIPEMD-160 | 5-10 |
| 5.8.1 Hash-code computation | 5-10 |
| 5.8.2 Padding rule | 5-10 |
| 5.8.3 Round function and initializing value | 5-10 |
| 6 SYMMETRIC ENCRYPTION | 6-1 |
| 6.1 Guidelines for usage and key length | 6-1 |
| 6.2 Implementation | 6-2 |
| 6.2.1 Batch mode | 6-2 |
| 6.2.2 Dialog mode | 6-3 |
| 6.2.3 Key generation | 6-4 |
| 6.3 SC1 - DES with ISO/IEC 10116 | 6-4 |
| 6.3.1 Implementation | 6-4 |

| | |
|--|------------|
| 6.4 SC2 - IDEA with ISO/IEC 10116 | 6-5 |
| 6.4.1 Implementation | 6-5 |
| 6.5 Other encryption mechanisms | 6-5 |
| 6.5.1 Triple DES | 6-5 |
| 6.5.2 SAFER SK128 | 6-6 |
| 6.6 ISO/IEC 10116 | 6-6 |
| 6.6.1 Electronic Code Book (ECB) mode | 6-6 |
| 6.6.1.1 Padding | 6-6 |
| 6.6.2 Cipher Block Chaining (CBC) mode | 6-7 |
| 6.6.2.1 Padding | 6-7 |
| 6.6.2.2 Initialization vector (IV) | 6-7 |
| 6.6.2.3 Implementation | 6-8 |
| 6.6.3 Cipher Feedback (CFB) mode | 6-8 |
| 6.6.3.1 Padding | 6-8 |
| 6.6.3.2 Initialization vector (IV) | 6-8 |
| 6.6.3.3 Implementation | 6-9 |
| 6.6.4 Output Feedback (OFB) mode | 6-9 |
| 6.6.4.1 Padding | 6-9 |
| 6.6.4.2 Initialization vector (IV) | 6-9 |
| 6.6.4.3 Implementation | 6-10 |
| | |
| 7 ASYMMETRIC ENCRYPTION | 7-1 |
| | |
| 7.1 AC1 - RSA for protocol data | 7-1 |
| 7.1.1 Coding of protocol data | 7-2 |
| 7.1.2 Preliminary processing | 7-3 |
| 7.1.3 Encryption | 7-3 |
| 7.1.4 Decryption | 7-3 |
| 7.1.5 Verification and extraction | 7-4 |
| 7.1.6 Decoding of protocol data | 7-4 |
| | |
| 8 ENTITY AUTHENTICATION | 8-1 |
| | |
| 8.1 One-way authentication (EA1) | 8-1 |
| 8.1.1 Technical description | 8-1 |
| 8.1.2 Characteristics | 8-2 |
| | |
| 8.2 One-way authentication (EA2) | 8-3 |
| 8.2.1 Technical description | 8-3 |
| 8.2.2 Characteristics | 8-3 |
| | |
| 8.3 Mutual authentication (EA3) | 8-4 |
| 8.3.1 Technical description | 8-4 |
| 8.3.2 Characteristics | 8-5 |
| | |
| 9 KEY TRANSPORT | 9-1 |
| | |
| 9.1 Key transport 1 (SKT1) | 9-1 |

| | |
|---|-------------|
| 9.1.1 Technical description | 9-2 |
| 9.1.2 Characteristics | 9-2 |
| 9.2 Key transport 2 (SKT2) | 9-3 |
| 9.2.1 Technical description | 9-3 |
| 9.2.2 Characteristics | 9-4 |
| 9.3 Key transport 3 (SKT3) | 9-4 |
| 9.3.1 Technical description | 9-4 |
| 9.3.2 Characteristics | 9-5 |
| | |
| 10 PUBLIC KEY TRANSPORT WITHOUT CERTIFICATES | 10-1 |
| | |
| 10.1 Public key transport mechanism 1 (PKT1) | 10-2 |
| 10.1.1 Technical description | 10-2 |
| 10.1.2 Properties | 10-3 |
| 10.2 Public key transport (PKT2) | 10-3 |
| 10.2.1 Technical description | 10-4 |
| 10.2.2 Properties | 10-5 |
| | |
| 11 MANAGEMENT INFRASTRUCTURE FOR PUBLIC KEY CERTIFICATES | 11-1 |
| | |
| 11.1 Introduction | 11-1 |
| 11.1.1 Registration | 11-2 |
| 11.1.2 Certification | 11-2 |
| 11.1.3 Certificate distribution and usage | 11-3 |
| 11.1.4 Subscriber initialization | 11-3 |
| 11.1.5 Certificate change management | 11-3 |
| 11.1.6 Revocation and suspension | 11-3 |
| 11.1.7 Auxiliary services | 11-4 |
| 11.1.7.1 Archiving | 11-4 |
| 11.1.7.2 Time stamping | 11-4 |
| 11.1.7.3 Attribute certification | 11-4 |
| 11.1.8 Overview | 11-4 |
| 11.2 Subscriber model | 11-5 |
| | |
| 11.3 Subscriber initialization models | 11-6 |
| 11.3.1 Model 1 | 11-7 |
| 11.3.2 Model 2 | 11-7 |
| | |
| 11.4 Registration | 11-8 |
| 11.4.1 Registration of residential persons | 11-9 |
| 11.4.2 Registration of organizations | 11-10 |
| 11.4.3 Registration of organizational persons | 11-11 |
| 11.4.4 Registration of organizational units and application processes | 11-12 |
| 11.4.5 Registration through security administrator | 11-13 |
| 11.4.6 Registration combined with public key submission | 11-13 |
| 11.4.6.1 First-time registration of subscriber public key | 11-14 |

| | |
|---|--------------|
| 11.4.6.2 Registration of secondary public key | 11-14 |
| 11.4.6.3 Public key registration through security administrator | 11-14 |
| 11.5 Key generation | 11-14 |
| 11.5.1 Key generation by subscriber | 11-14 |
| 11.5.2 Key generation by central authority | 11-15 |
| 11.5.3 Key escrow | 11-15 |
| 11.6 Certification | 11-15 |
| 11.6.1 The Certification Authority | 11-15 |
| 11.6.2 Maintaining integrity of the CA public key | 11-16 |
| 11.7 Certificate implementation | 11-16 |
| 11.7.1 Minimal requirements | 11-16 |
| 11.7.1.1 Version number | 11-17 |
| 11.7.1.2 Certificate reference number | 11-17 |
| 11.7.1.3 Certificate issuer identification | 11-17 |
| 11.7.1.4 Certificate issuer algorithms | 11-17 |
| 11.7.1.5 Certificate issuer public key identification | 11-17 |
| 11.7.1.6 Certificate owner identification | 11-17 |
| 11.7.1.7 Validity period | 11-17 |
| 11.7.1.8 Certificate owner - public key information | 11-18 |
| 11.7.2 Attribute certification | 11-18 |
| 11.8 Subscriber initialization mechanisms | 11-18 |
| 11.8.1 Subscriber initialization mechanism 1 | 11-18 |
| 11.8.2 Subscriber Initialization Mechanism 2 | 11-21 |
| 11.9 Certificate usage | 11-25 |
| 11.9.1 Validation of certificates | 11-28 |
| 11.10 Certificate change management | 11-29 |
| 11.10.1 Certificate change | 11-29 |
| 11.10.2 Certificate issuance based on an existing certificate | 11-29 |
| 11.11 Revocation | 11-29 |
| 11.11.1 The Certificate Revocation Authority | 11-30 |
| 11.11.2 Certificate revocation mechanism | 11-30 |
| 11.11.2.1 Revocation request | 11-32 |
| 11.11.2.2 Marking a certificate as revoked | 11-33 |
| 11.11.2.3 Giving notice of revoked certificates | 11-33 |
| 11.11.3 Suspending certificates | 11-33 |
| 11.11.4 The Certificate Suspension Authority | 11-34 |
| 11.11.5 Certificate suspension mechanism | 11-34 |
| 11.11.5.1 Suspension request | 11-36 |
| 11.11.5.2 Marking a certificate as suspended | 11-37 |
| 11.11.5.3 Giving notice of suspended certificates | 11-37 |
| 11.11.6 Reactivating a suspended certificate | 11-37 |
| 11.11.6.1 Reactivation request | 11-37 |
| 11.11.6.2 Giving notice of reactivated certificates | 11-38 |
| 11.11.7 Combined suspension and revocation | 11-38 |

12 GLOSSARY**12-1****13 RELEVANT STANDARDS****13-1**

List of figures

| | |
|--|-------|
| Figure 1-1 TBSS as a basis for different applications | 1-1 |
| Figure 2-1 Content of the TBSS | 2-1 |
| Figure 3-1 Security processing for data authentication | 3-1 |
| Figure 3-2 Independent security message | 3-2 |
| Figure 3-3 Security processing for encryption | 3-3 |
| Figure 3-4 Combined authentication and encryption | 3-4 |
| Figure 5-1 Structure and interaction of digital signature mechanisms | 5-1 |
| Figure 5-2 Signing a message M | 5-2 |
| Figure 5-3 Digital signature verification | 5-3 |
| Figure 5-4 Signature process with ISO/IEC 9796 | 5-6 |
| Figure 5-5 Verification process with ISO/IEC 9796 | 5-7 |
| Figure 5-6 MD strengthening | 5-8 |
| Figure 6-1 Overview of encryption mechanisms | 6-1 |
| Figure 6-2 Encryption in batch mode | 6-2 |
| Figure 6-3 Decryption in batch mode | 6-3 |
| Figure 6-4 Encryption in dialog mode | 6-4 |
| Figure 6-5 Electronic Code Book mode | 6-6 |
| Figure 6-6 Cipher Block Chaining (CBC) mode complying with ISO/IEC 10116 | 6-7 |
| Figure 6-7 Cipher Feedback (CFB) mode complying with ISO/IEC 10116 | 6-8 |
| Figure 6-8 Output Feedback (OFB) mode complying with ISO/IEC 10116 | 6-9 |
| Figure 7-1 Overview of asymmetric encryption mechanisms | 7-1 |
| Figure 8-1 Overview of entity authentication mechanisms | 8-1 |
| Figure 8-2 EA1 One-way authentication with one pass | 8-2 |
| Figure 8-3 EA2 One-way authentication with two passes | 8-3 |
| Figure 8-4 EA3 Mutual authentication with three passes | 8-4 |
| Figure 9-1 Overview of key transport mechanisms | 9-1 |
| Figure 9-2 SKT1 key transport with authentication | 9-2 |
| Figure 10-1 Typical telebanking relationships | 10-1 |
| Figure 10-2 Public key transport using an authenticated channel. | 10-2 |
| Figure 10-3 Authentication using a second channel | 10-4 |
| Figure 11-1 Distribution of public keys via a key management infrastructure | 11-1 |
| Figure 11-2 Certification: separating authentication from distribution | 11-2 |
| Figure 11-3 Functions of the public key management infrastructure | 11-4 |
| Figure 11-4 Subscriber model and associated roles | 11-6 |
| Figure 11-5 Subscriber initialization with key generation before registration. | 11-7 |
| Figure 11-6 Subscriber initialization with key generation at certification. | 11-8 |
| Figure 11-7 General registration procedure | 11-9 |
| Figure 11-8 Registration of organizational persons | 11-11 |
| Figure 11-9 Subscriber Initialization Mechanism 1 | 11-19 |
| Figure 11-10 Subscriber Initialization Mechanism 2 | 11-22 |
| Figure 11-11 Certificate usage | 11-26 |
| Figure 11-12 Certificate revocation mechanism | 11-31 |
| Figure 11-13 Certificate suspension mechanism | 11-35 |

1 Objectives

The Swiss finance industry requires an open and standardized IT security solution. In order to fully exploit the synergy effects between the various security solutions of the different services and applications, the Swiss banks decided to develop this common **Security Standard** for the Swiss finance industry. The objectives of the TBSS are as follows:

1. Interoperability: targeted selection of mechanisms that can be implemented across different business sectors and countries.
2. Standardization: use of international standards and harmonization with national standards (Swiss Banking EDIFACT Security).
3. Openness: combination of implementations from different manufacturers should be possible.
4. Platform independence: not dependent on a specific implementation environment.
5. Transparency: not dependent on specific communication protocols and media.
6. Cost-efficiency: the TBSS must be economical to implement on the most popular systems.
7. Expandability: the solutions can be expanded or replaced at any time. Basic variants (e.g. software) and options with higher security (e.g. smart card) can be defined.
8. Acceptance: the TBSS must be designed so that it is readily accepted by providers and users.
9. Applicability: the same mechanisms and procedures can be used in different environments and applications, for example with EDI, Videotex and DTA.
10. Investment protection: all the points listed above should encourage manufacturers and service providers to produce harmonized solutions, since the basis is an expandable, open and widely supported security standard.

The TBSS is the core document for different services and applications. Application-specific rules for the TBSS shall be defined in the relevant Implementation Guidelines and Integration Concepts. The next figure illustrates these objectives.

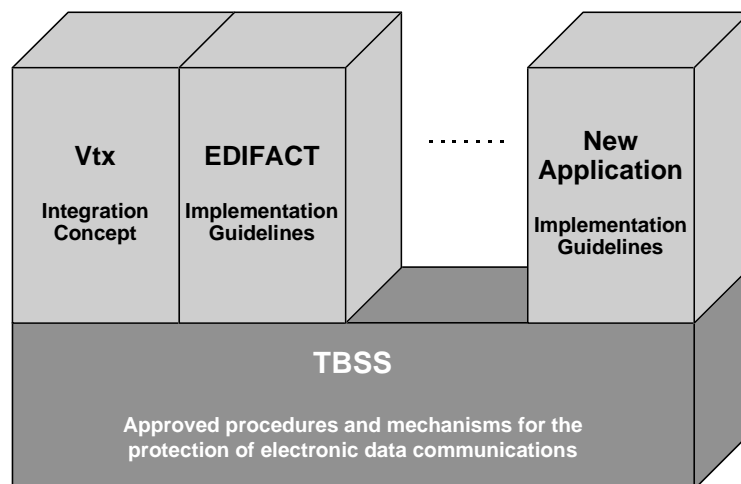


Figure 1-1 TBSS as a basis for different applications

This means the TBSS must not be restricted to a specific application, but must make available a selective range of techniques that are suitable for use in different application scenarios.

2 Content

The TBSS specifies technical procedures and mechanisms for the protection of electronic data communications between two computer systems. These technical specifications ensure secure data communications between service providers and customers. They also allow secure data exchange between two service providers or two customers.

The TBSS does not deal with the security of end systems, especially not local access control. However, the annex does contain a number of minimum requirements for the security of end-user systems.

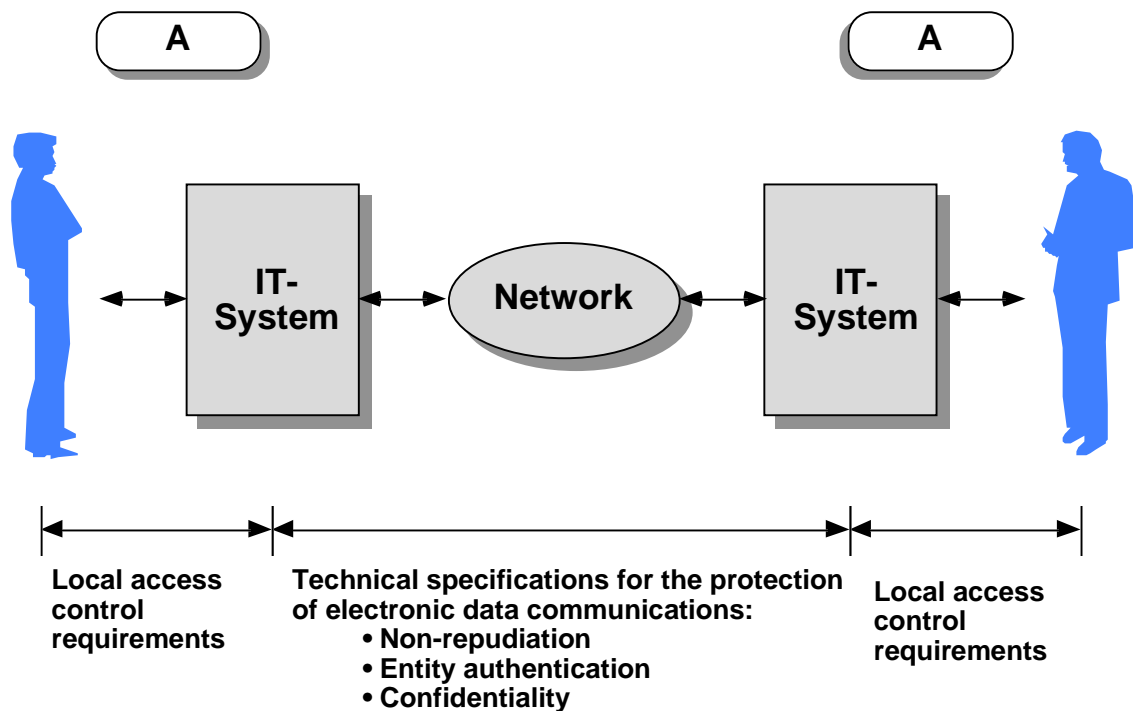


Figure 2-1 Content of the TBSS

The TBSS must provide techniques and mechanisms that satisfy the principal security requirements of financial application scenarios. These include:

1. Integrity of transmitted data.
2. Non-repudiation between sender and receiver.
3. Entity authentication.
4. Confidentiality.

It must also be taken into account that the services and mechanisms are based on cryptographic algorithms and therefore must be provided with the appropriate keys. Technical specifications are needed for key management in order to ensure interoperability between different sectors and countries.

If the TBSS is to satisfy all these requirements and still be independent of a particular application, it must offer a range of techniques and mechanisms. Programmers must then select mechanisms from the TBSS to suit a specific application. For example, an interactive application cannot, a priori, use the same mechanisms as a batch application. To this end, the TBSS offers a tool box of modular standardized elements that provide the maximum degree of synergy in their implementation.

The TBSS mechanisms can be divided into the following hierarchical classes:

1. Basic elements: cryptographic algorithms
2. Basic mechanisms: the algorithms' modes of operation and types of implementation
3. Combined mechanisms: rules for combining different algorithms.
4. Protocol mechanisms: rules for the interoperability of business partner systems

We now provide a brief introduction to the philosophy of TBSS mechanisms.

2.1 Cryptographic algorithms as basic elements

Cryptographic algorithms are the basic elements on which the other mechanisms are built. The TBSS currently contains the following algorithms:

| Algorithm | Explanation |
|--------------------------|-------------------------|
| RSA | Public key method |
| DES IDEA | Block cipher algorithms |
| RIPEMD-128 RIPEMD-160 | Dedicated hash function |

Table 2-1 TBSS cryptographic algorithms

Each of these basic elements is essential. In other words, if one of the algorithms is omitted, the security requirements can no longer be fully satisfied.

The selection of cryptographic algorithm is by no means sufficient to ensure interoperability between different business sectors and countries. This requires the definition of the algorithm's modes of operation at the next level.

2.2 Modes of operation

Cryptographic algorithms can be implemented in a number of ways, depending on which security requirements need to be met. With the public key method RSA you have to specify the following implementation aspects, amongst others:

- Use in digital signature computation
- Use as an encryption technique
- Formatting of data blocks before and after algorithmic processing
- Use of additional procedures, such as redundancy check methods.

With the block cipher algorithms, the following implementation aspects (amongst others) must be specified:

- Modes of operation as encryption techniques
- Use as a hash function
- Use as a message authentication code.

With the dedicated hash functions, the following implementation aspects (amongst others) must be specified:

- Formatting of data blocks before and after algorithmic processing

The TBSS currently contains technical specifications for the following modes of operation:

| TBSS mechanism | Based on | Purpose |
|----------------|------------|--------------------------------------|
| DS1 | RSA | Use as a digital signature mechanism |
| AC1 | RSA | Use as an encryption mechanism |
| SC1 | DES | Use as an encryption mechanism |
| SC2 | IDEA | Use as an encryption mechanism |
| HF1 | DES | Use as a hash function |
| HF2 | RIPEMD-128 | Use as a hash function |
| HF3 | RIPEMD-160 | Use as a hash function |

Table 2-2 Types of implementation for cryptographic algorithms in TBSS

The next stage is to specify any combination of individual mechanisms (where necessary).

2.3 Combinations of mechanisms

In some cases several cryptographic algorithms have to interact in precisely defined types of use in order to ensure that security and interoperability requirements are met.

For example, if you want to attach digital signatures to messages or files of arbitrary length, there have to be technical specifications governing the way the hash function connects to the signature transformation. The TBSS currently contains technical specifications for the following combinations of mechanisms:

| TBSS mechanisms | Based on | Purpose |
|-----------------|----------|--|
| DSH1 | DS1, HF1 | Combined use of digital signature and hash function 1 (DES) |
| DSH2 | DS1, HF2 | Combined use of digital signature and hash function 2 (RIPEMD-128) |
| DSH3 | DS1, HF3 | Combined use of digital signature and hash function 3 (RIPEMD-160) |

Table 2-3 Combination of mechanisms in TBSS

The next stage is to specify the combination of mechanisms on different end-user systems (where necessary).

2.4 Protocol mechanisms

With certain security requirements, the business partners’ systems have to interact in a precisely defined manner. This type of mechanism is referred to as a protocol.

If we take the example of entity authentication, this security requirement involves the reciprocal exchange and processing of security protocol data before any productive user data can be exchanged. Consideration also has to be given to the way in which different applications are implemented. The following questions have to be answered:

- One-way or mutual entity authentication.
- Use of timestamps, sequence numbers or random numbers.
- Integration of key exchange and entity authentication.

The TBSS currently contains technical specifications for the following protocol mechanisms:

| TBSS mechanism | Based on | Purpose |
|----------------|----------|--|
| EA1 | DSH | One-way authentication in one pass using timestamps or sequence numbers. |
| EA2 | DSH | One-way authentication in two passes using random numbers. |
| EA3 | DSH | Mutual authentication in three passes using random numbers. |
| SKT1 | DSH, AC1 | Key transport integrated in EA1. |
| SKT2 | DSH, AC1 | Key transport integrated in EA2. |
| SKT3 | DSH, AC1 | Key transport integrated in EA3. |

Table 2-4 TBSS protocol mechanisms

Procedures and protocol mechanisms also have to be provided to ensure secure initialization of the business partners' systems. For example, if public key systems are used, there must be secure distribution of either partner's public key. To this end, open systems often employ sector-specific or national public key management infrastructures. Chapter 11 discusses the question of Public Key Certification. The TBSS currently contains technical specifications for the following business partner initialization procedures:

| TBSS mechanism | Based on | Purpose |
|----------------|----------------------------|--|
| PKT1 | physical security | Registration and distribution of public key data via secure communications medium |
| PKT2 | DSH, HF, physical security | Registration and distribution of public key data via non-secure communications medium. Authentication via second independent communications channel (e.g. registered letter) |

Table 2-5 TBSS initialization procedures

3 Security processing

To ensure secure data communications, the following points must be observed:

1. The scope of the security service, i.e. which data are subject to security processing. This varies depending on the type of security service.
2. The insertion of supplementary data and parameters which are needed at the receiver's end to perform the security services. These supplementary data must be provided by security management and are covered by the generic term 'security header'.
3. The insertion and representation of security results. In this chapter security results are covered by the term 'security trailer'.

The precise format of the security header and trailer is defined in the application-specific Implementation Guidelines.

3.1 Data authentication

3.1.1 Integrated security

The next figure shows the typical stages of security processing during data authentication. Data authentication services include Non-Repudiation of Origin (NRO), Non-Repudiation of Receipt (NRR) and Message Origin Authentication (MOA). These services are explained in greater detail in Chapter 4 *Security services*.

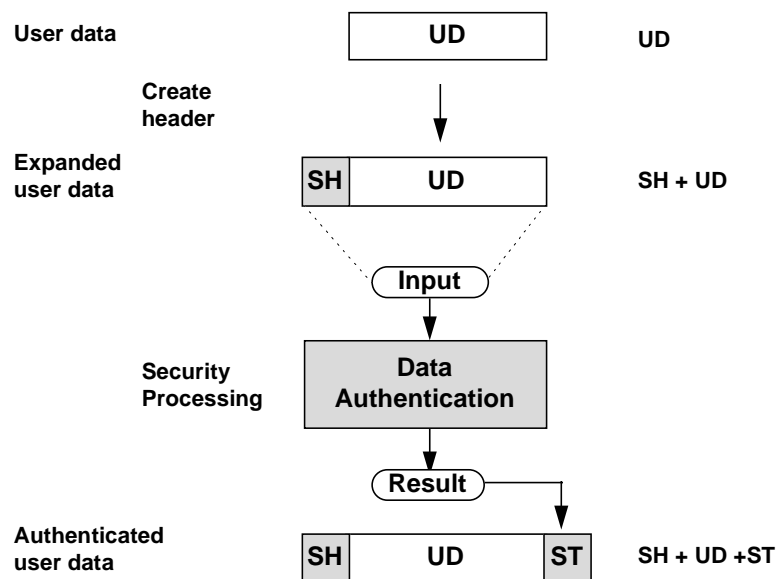


Figure 3-1 Security processing for data authentication

1. The user data (UD) must be extended by a security header (SH) containing the supplementary data and parameters needed to perform the security services. Examples include key identification, timestamp, methods used, etc.
2. The user data with the appended SH form the input for the security processing. It is important that the SH is also protected by the security processing.

3. The result of the security processing (digital signature or MAC) is written into the security trailer.
4. The secured user data thus consist of security header, user data and security trailer.

As soon as the data is secured at the sender’s end, no further changes may be made (e.g. reformatting). At the receiver’s end, the received data must be forwarded without modification to the security processing.

3.1.2 Data authentication with independent security message

In principle, for data authentication the security elements (SH/ST) can be separated from the user data and can be forwarded separately to the intended recipient (e.g UN/EDIFACT AUTACK). For example, user data can be transferred automatically from A to B overnight without any security elements, thereby benefiting from off-peak rates. On the following day, the security elements can be released by the relevant employee at the sender’s end (A) and be forwarded to B. The receiver B has been able to perform preliminary processing on the user data already received, and can continue processing once the security elements are received and successfully checked. Alternatively, the security elements can be sent first, followed by the user data. The next figure shows the procedure required for security processing with independent security messages.

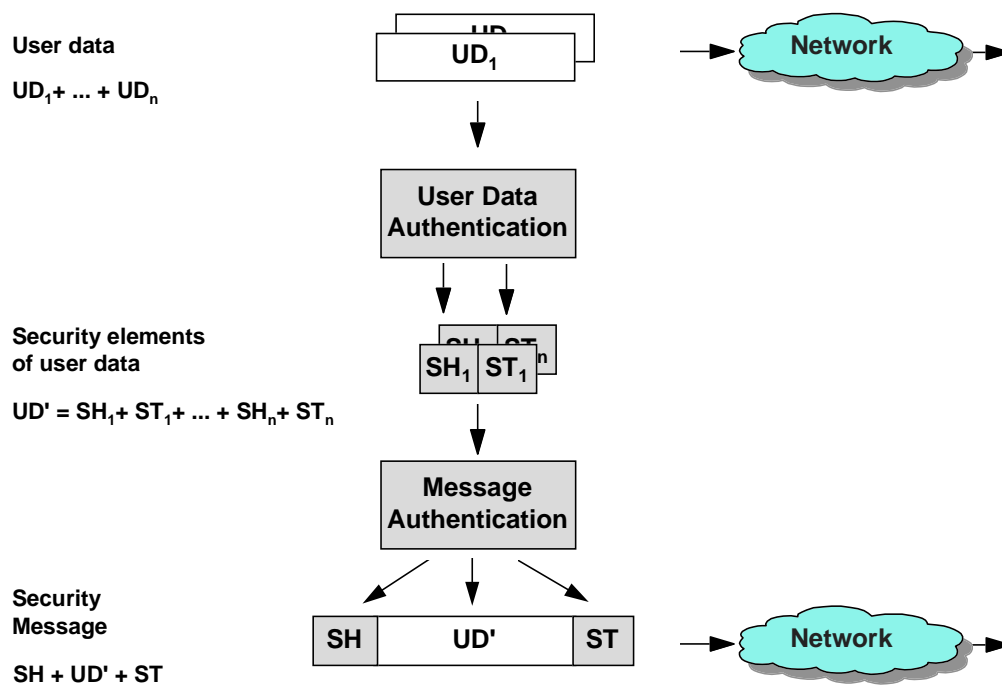


Figure 3-2 Independent security message

1. The user data (UD_i , message or file, for example) are pre-processed. Administrative details (e.g. methods used) are written into the security header (SH_i). A hash value is computed or the user data are secured directly (MAC or digital signature).
2. The results of this pre-processing (security elements SH_i/ST_i) are collected. They serve as user data UD' for subsequent security processing.
3. The user data UD' (comprising the combined security elements) are secured. The processing steps are the same as those required for data authentication, data encryption or a combination of the two. The result of processing is the security message consisting of $SH+UD'+ST$.

3.2 Data encryption

The next figure shows the typical steps involved in security processing in the case of data encryption.

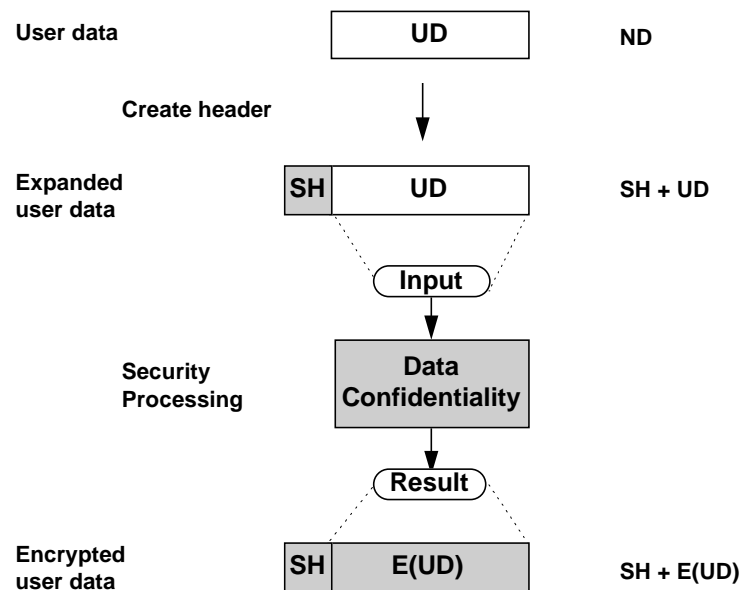


Figure 3-3 Security processing for encryption

1. The user data (UD) must be extended by a security header (SH) containing the supplementary data and parameters needed to perform encryption. Examples include key identification, methods used, etc.
2. However, only the user data are encrypted. The SH cannot be encrypted, otherwise it would be impossible for the receiver to access the supplementary data and parameters needed to perform decryption.
3. The result of the encryption are the encrypted user data E(UD). They need to be filtered into a data format that is compatible with the communication methods used.
4. The final encrypted user data therefore comprise SH+E(UD).

Although the SH is not encrypted, there is no direct threat to the security service, since modifications to the SH do not endanger the confidentiality of the user data. Unauthorized manipulation of the SH means the data will be rejected, as happens with data authentication services.

Since the SH is generated independently of the user data encryption, it is also possible to change the sequence of the steps illustrated above in a current implementation: you could start with the encryption of the user data, then create the SH.

3.3 Combined data authentication and encryption

In certain cases both data authentication services (NRO, NRR, MOA) and confidentiality are required. This section describes the processing steps for combined data authentication and encryption.

As with conventional paper-based transactions, the user data first have to be authenticated (signed) and then encrypted (put in an envelope). The following figure shows the necessary steps in combined security processing.

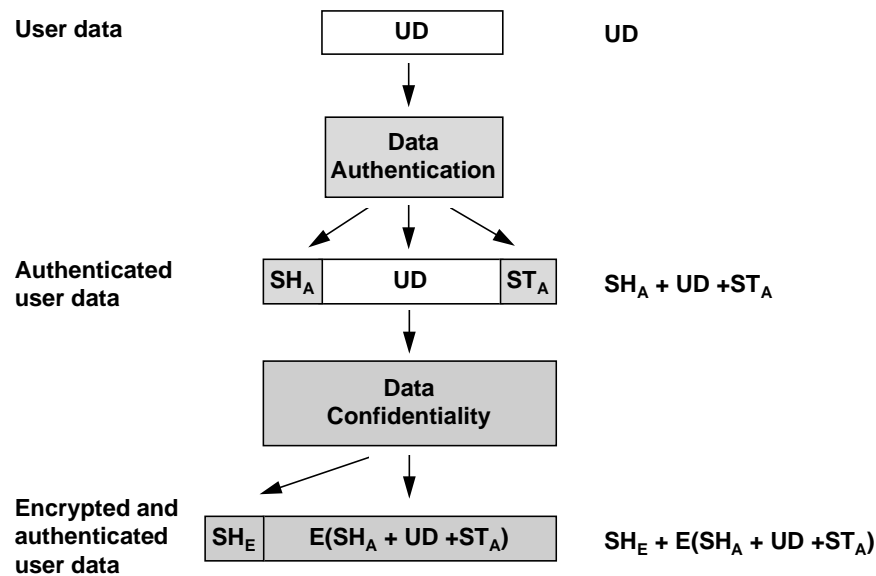


Figure 3-4 Combined authentication and encryption

1. The user data (UD) must be extended by a security header (SH_A) for authentication purposes. This security header contains the supplementary data and parameters needed to perform the authentication service. Examples include key identification, timestamp, methods used, etc.
2. The user data and the appended SH_A form the input for authentication. It is important that the SH_A is also protected by security processing.
3. The result of security processing (digital signature or MAC) is written to the security trailer (ST_A). The authenticated user data thus consist of SH_A+UD+ST_A.
4. The authenticated user data must be extended by another security header (SH_E) for encryption, which contains the supplementary data and parameters needed to perform the encryption service. Examples include key identification, methods used, etc.
5. However, only the authenticated user data are encrypted. The encryption header SH_E cannot be encrypted, otherwise it would be impossible for the receiver to access the supplementary data and parameters needed to perform decryption.
6. The result of the encryption are the encrypted authenticated user data E(SH_A+UD+ST_A). They need to be filtered into a data format that is compatible with the communication. The final authenticated and encrypted user data therefore consist of SH_E+E(SH_A+UD+ST_A).

The sequence just described is used when authentication and encryption apply to all the user data (or apply to the same data area). In certain cases (e.g. key management), however, encryption is only applied to a few data fields (selective confidentiality). Here the specific data fields are encrypted first, and in the second pass all the user data are authenticated.

4 Security services

4.1 Non-repudiation

The purpose of non-repudiation is to enable a binding transaction to be performed between business partners using electronic media. Just as the signature of an authorized agent confirms the validity of a conventional printed payment instruction, electronic media also need to have a mechanism for verifying beyond all doubt the authorship of the electronic payment instruction. The security services that make electronic transactions binding are known as non-repudiation services. A distinction is made between "non-repudiation of origin" (the sender cannot dispute the transmission of an electronic transaction) and "non-repudiation of receipt" (the receiver cannot dispute the receipt of an electronic transaction).

4.1.1 Non-repudiation of origin (NRO)

This service protects the recipient of a message in the event that the sender disputes having sent it. The sender is unable to contest either the transmission of the message or its contents. NRO therefore makes it possible to unmistakably distinguish the sender and receiver of a message. With conventional methods such as MAC or data encryption such a distinction is not possible, since the sender and receiver have the same information available and could falsify the messages of the other party.

NRO: The sender of a message/file confirms the origin and integrity of the message/file in such a way that the authenticity of the message/file can also be verified by a third party.

The following mechanisms are required for implementing NRO:

1. Digital signature
2. Hash function
3. Key generation for the signature procedure
4. Registration of the users' public key data
5. Distribution of the users' public verification key

The digital signature not only provides non-repudiation of origin but also message content integrity and message origin authentication.

4.1.2 Non-repudiation of receipt (NRR)

Non-Repudiation of Receipt (NRR) is based on the same principles and mechanisms as NRO. The only difference is that a signature is attached to the acknowledgement of receipt, rather than to the original data. The receiver *B* issues an acknowledgement of successful receipt of the data, appends a digital signature to it and then forwards it to *A*, who sent the original data.

NRR: the receiver gives irrefutable acknowledgement of having received a message/file from the sender in such a way that this acknowledgement can be verified by a third party.

With financial service providers, a distinction is made between

1. Acknowledgement of individual messages.
2. Batch acknowledgement

Both can be sent to customers in electronic format with digital signatures appended (the signature key of the service provider).

In some cases, however, it may be more appropriate to send acknowledgements via fax, letter, diskette etc. depending on the communications resources of the customers' systems, or as a fallback solution.

The NRR service requires the same mechanisms as NRO.

4.2 Entity authentication

This service protects from masquerading under a false user identity. The TBSS defines a number of different authentication mechanisms, since no one mechanism is ideal for every application. The following aspects are important for specific implementation:

1. One-way or mutual authentication.
2. Number of passes.
3. Use of classic challenge-and-response techniques. In this case both parties must be able to generate random numbers.
4. Use of timestamp. In this case both parties must have synchronized clocks.
5. Use of sequence numbers, In this case both parties must be able to manage bilateral counters.
6. Key distribution: authentication mechanisms can also be used for transporting a secret key K from A to B . The relevant cryptosystems must be available for this.

Entity authentication mechanisms are described in detail in Chapters 8 *Entity authentication* and 9 *Key transport*.

4.3 Confidentiality

This service protects users from unauthorized eavesdropping or disclosure of transmitted data. Data confidentiality is achieved by using symmetric encryption techniques.

The following aspects are important for the specific implementation of encryption techniques:

1. Message or dialog oriented data encryption.
2. Static or dynamic message or session keys.
3. Generation of data keys.
4. Distribution of data keys.

Encryption mechanisms are described in detail in Chapters 6 *Symmetric encryption* and 7 *Asymmetric encryption*.

4.4 Relationship between security services and security mechanisms

This chapter provides details of which mechanisms can be used for which security services. The tables are structured as follows:

1. Security mechanism
2. Description: brief description of the security mechanism.
3. Standard: the international standards with which the mechanism complies or is based on.
4. Based on: list of the mechanisms, procedures or algorithms on which the mechanism is based.

Explanations of the abbreviations used in this chapter are provided in the glossary.

4.4.1 Non-repudiation

The next table gives an overview of the security mechanisms that can be used for non-repudiation services.

| Mechanism | Description | Standard | Based on |
|-----------|--|---------------------------------|------------|
| DS1 | Digital signature with message recovery. | ISO/IEC 9796 | RSA |
| DSH1 | Digital signature with data hashing. Blockcipher-based hash function with DES as basic method. | ISO/IEC 9796 ISO/IEC 10118-2 | DS1 HF1 |
| DSH2 | Digital signature with data hashing. Dedicated hash function with RIPEMD-128 as basic method. | ISO/IEC 9796 ISO/IEC 10118-3 | DS1 HF2 |
| DSH3 | Digital signature with data hashing. Dedicated hash function with RIPEMD-160 as basic method. | ISO/IEC 9796 ISO/IEC 10118-3 | DS1 HF3 |

4.4.2 Entity authentication

The next table provides an overview of the security mechanisms that can be used for entity authentication.

| Mechanism | Description | Standard | Based on |
|-----------|--|----------------|----------|
| EA1 | One-way authentication in one pass using timestamps or sequence numbers. | ISO/IEC 9798-3 | DSH |
| EA2 | One-way authentication in two passes using random numbers. | ISO/IEC 9798-3 | DSH |
| EA3 | Mutual authentication in three passes using random numbers. | ISO/IEC 9798-3 | DSH |

4.4.3 Confidentiality

The next table provides an overview of the security mechanisms that can be used for symmetric encryption of user data.

| Mechanism | Description | Standard | Based on |
|-----------|--|--------------------------------|----------|
| SC1 | Data Encryption Standard with ECB, CBC, CFB, OFB modes of operation. | ANSI X3.92 ISO/IEC 10116 | DES |
| SC2 | IDEA with ECB, CBC, CFB, OFB modes of operation. | ISO 9979/0002 ISO/IEC 10116 | IDEA |

The next table provides an overview of the security mechanisms that can be used for asymmetric encryption of protocol data.

| Mechanism | Description | Standard | Based on |
|-----------|----------------------------------|--------------------------|----------|
| AC1 | Asymmetric encryption using RSA. | related to ANSI X9.31 | RSA |

4.4.4 Key management

4.4.4.1 Key transport

The next table provides an overview of the security mechanisms that can be used for the transport of symmetric keys.

| Mechanism | Description | Standard | Based on |
|-----------|---|--|------------|
| SKT1 | Key transport (one-pass) using timestamps or sequence numbers. This mechanism is an extension of EA1. | ISO/IEC 9798-3 ISO/IEC 11770-3 ISO 11166-1 | DSH AC1 |
| SKT2 | Key transport (two-pass) with one-way authentication based on random numbers. This mechanism is an extension of EA2 | ISO/IEC 11770-3 ISO/IEC 9798-3 | DSH AC1 |
| SKT3 | Key transport (three-pass) with mutual authentication based on random numbers. This mechanism is an extension of EA3. | ISO/IEC 11770-3 ISO/IEC 9798-3 | DSH AC1 |

4.4.4.2 Public key transport

The next table provides an overview of the security mechanisms that can be used for the registration and transport of public key data (public verification key).

| Mechanism | Description | Standard | Based on |
|-----------|--|--------------------------|--------------------------------------|
| PKT1 | Registration and distribution of public key data via a secure communications medium | ISO/IEC 11770-3 X.509 | physical security |
| PKT2 | Registration and distribution of public key data via a non-secure communications medium. Authentication via a second independent communications channel (e.g. registered letter) | ISO/IEC 11770-3 X.509 | DSH1/2 HF1/2 physical security |

4.4.5 Support mechanisms

4.4.5.1 Hash functions

The next table provides an overview of hash functions which are required for the digital signature in particular, but can also be used independently.

| Mechanism | Description | Standard | Based on |
|-----------|---|-----------------|------------|
| HF1 | Blockcipher-based hash function with DES as basic method. | ISO/IEC 10118-2 | DES |
| HF2 | Dedicated hash function with RIPEMD-128 as basic method. | ISO/IEC 10118-3 | RIPEMD-128 |
| HF3 | Dedicated hash function with RIPEMD-160 as basic method. | ISO/IEC 10118-3 | RIPEMD-160 |

5 Digital signature

A digital signature is the security mechanism on which non-repudiation services can be based. A distinction is made between

1. Digital signature with message recovery: the message is transported within the signature. This is only possible if the user data are shorter than the block length of the signature procedure.
2. Digital signature with data hashing: in this case a signature is applied not to the data, but to a hash value derived from the data. The data can be transported along with the signature or separately.

5.1 Overview

Different mechanisms have to be provided for, and in connection with, the digital signature (depending on the intended use):

The following figure gives an overview of the interaction between the mechanisms described in this chapter.

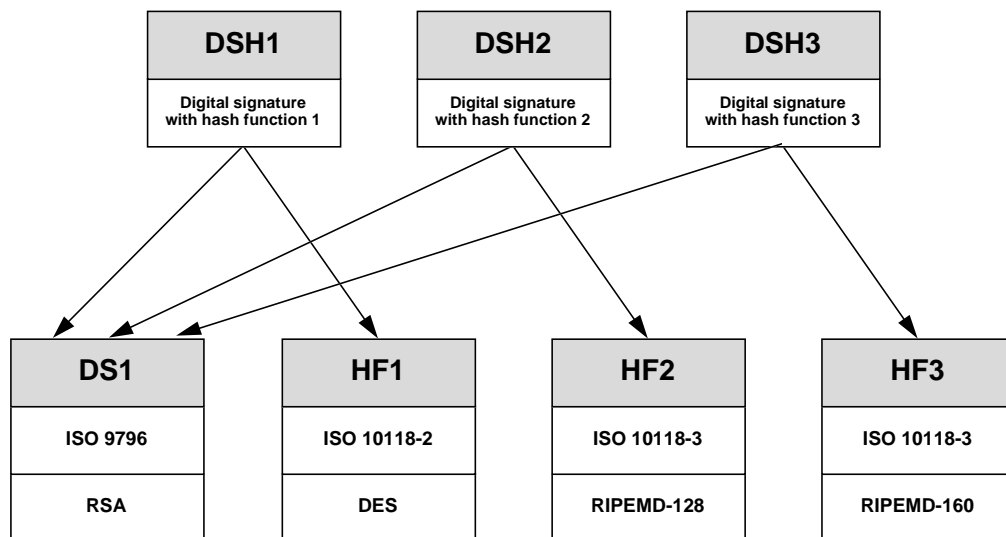


Figure 5-1 Structure and interaction of digital signature mechanisms

5.1.1 Digital signature 1 (DS1)

The basic method here is the RSA public-key system. As an extension of this, ISO/IEC 9796 defines the use of RSA to compute a digital signature with message recovery. ISO/IEC 9796 specifies important additional aspects such as user data padding and the introduction of verifiable redundancy.

RSA can operate with a fixed (constant) public exponent (as part of the verification key) for all system users. This has a number of advantages, e.g. the users' public keys are half the size, and the efficiency of digital signature verification is improved by as much as two orders of magnitude. In the general case the user is also free to select the public exponent (as part of the verification key) for the digital signature.

5.1.2 Hash function (HF)

In the TBSS, the following hash functions are currently approved when using the digital signature with user data of arbitrary length:

- HF1: ISO/IEC 10118-2 Blockcipher- based hash function using DES as the basic algorithm.
- HF2: ISO/IEC 10118-3 Dedicated hash function using RIPEMD-128 as the basic algorithm.
- HF3: ISO/IEC 10118-3 Dedicated hash function using RIPEMD-160 as the basic algorithm.

Dedicated hash functions are in general an order of magnitude faster than blockcipher-based hash functions.

The security of the hash functions can be significantly improved by using a method known as MD strengthening. With ISO/IEC 10118-3 Dedicated hash functions, this technique is already integrated in the function’s specification. In the case of ISO/IEC 10118-2 with DES, MD strengthening has to be specified additionally.

5.1.3 Digital signature with data hashing (DSH)

For general use, the digital signature DS1 has to be combined with one of the available hash functions (HF1, HF2 or HF3):

- DSH1: Digital signature 1 with hash function 1 (DES based)
- DSH2: Digital signature 1 with hash function 2 (RIPEMD-128)
- DSH3: Digital signature 1 with hash function 3 (RIPEMD-160)

5.2 Use of the digital signature

Typically a customer will sign a message such as a payment instruction and a service provider will verify the received signature. However, since the procedures are generally applicable, the following description is based on two imaginary users *A* and *B*.

5.2.1 Generating a signed message

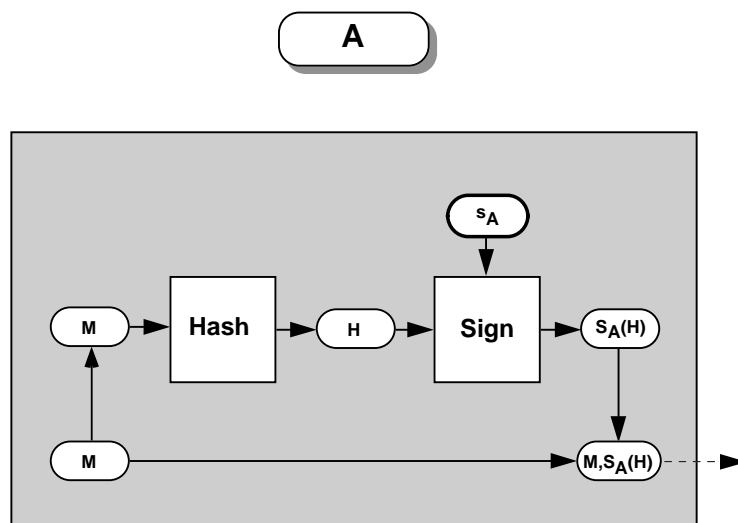


Figure 5-2 Signing a message *M*

With the help of a hash function, user *A* produces a fixed-length hash value H from the message M . This hash value is then digitally signed using *A*'s signature key s_A . The resulting signature $S_A(H)$ is inserted into the message data at a specified point (typically at the end of the message, in the security trailer).

Additional security-related information, such as the reference number of the signature key employed, is inserted at a second specified point within the message data (typically at the start of the message, in the security header). It is important to remember here that the hash value must also embrace the security header.

5.2.2 Signature verification

User *B* verifies *A*'s signature. To do so, he requires the counterpart of *A*'s private signature key. This is *A*'s public verification key v_A . With the help of the hash function, user *B* takes the received message M and produces the hash value H , then checks the validity of the received signature $S_A(H)$ using the public verification key v_A and the locally computed hash value H .

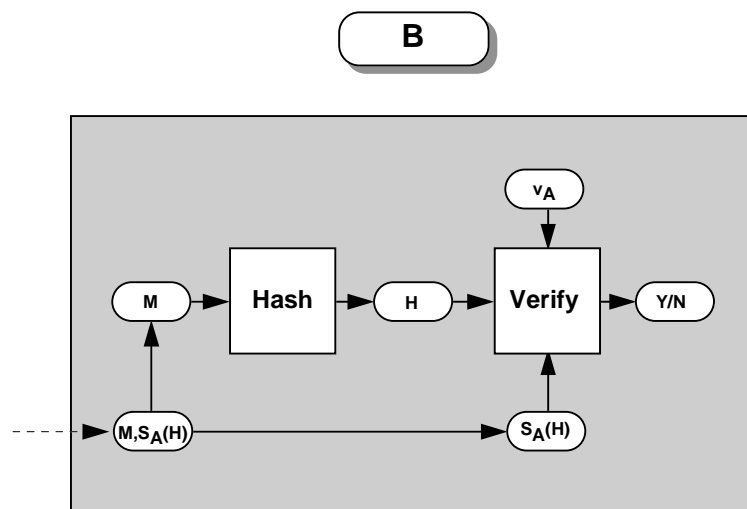


Figure 5-3 Digital signature verification

The verification keys can be stored openly at *B*'s end (e.g. in the customer file). Care must be taken to protect the integrity of the verification keys.

5.3 DS1 - Digital signature complying with ISO/IEC 9796

The title of ISO/IEC 9796 is *Digital Signature Scheme giving Message Recovery*. The title refers to the fact that the data block is transformed into a digital signature by the signature process, and is then restored at the receiver's end by the verification process (Message Recovery). Messages that are shorter than one signature block can therefore be transported within the signature itself. In this type of application the message does not have to be sent as well as the signature, nor is any hash function required. ISO/IEC 9796 defines the computation and verification of digital signatures using RSA or Rabin's method (exponent 2).

The big advantage of ISO/IEC 9796 is that no assumptions need to be made about the structure of the input data. ISO/IEC 9796 signature transformation expands the input data with padding bits and cyclical iteration, and inserts specifically selected redundancy for verification purposes. The hash value calculated from the hash function can therefore be entered straight into the signature transformation, without prior processing. In this way, any of the hash functions can be used with ISO/IEC 9796.

ISO/IEC 9796 signature transformation specifies the following steps:

1. Padding of input data to multiples of 8 bits;
2. Expansion of padded input data to half the block length;
3. Incorporation of redundancy;
4. Truncation and selective bit processing;
5. Selection of the representative element (only relevant for even exponents);
6. Signature computation (with the secret exponent).

ISO/IEC 9796 verification transformation stipulates the following steps for recovering and verifying the original data block (comprising the hash value H):

1. Recovery of the signature using the sender's public key (public exponent and modulus);
2. Computing the intermediate integer;
3. Computing the data block (comprising the hash value with built-in redundancy);
4. Checking redundancy and extracting the hash value H . If the redundancy check is unsuccessful, the signature is rejected.

5.3.1 ISO/IEC 9796 with a fixed exponent

RSA can function with a fixed (constant) public exponent for all the system users. This offers the following advantages:

1. The users' public keys are only half the size. This is advantageous in terms of storage space requirements, certificate size and transmission bandwidth.
2. As the public exponent can be used selectively, digital signature verification runs very efficiently. Throughput can be increased by as much as a factor of 100 in this way.

With a constant exponent v , a user's public verification key consists of a personal modulus n_A .

| | |
|-----------------------------------|------------------------------|
| Public exponent | v , constant for all users |
| User A 's public key | n_A |
| User A 's private signature key | s_A |

Table 5-1 Key format when verification exponent is fixed

Both the use and the selection of a fixed exponent are governed by the application-specific integration concept.

5.3.2 ISO/IEC 9796 with free exponent

In general user A is free to choose the public verification key, comprising modulus n_A and exponent v_A .

Both components (n_A, v_A) of A 's verification key must be supplied in an authenticated fashion to all recipients of A 's digital signatures.

| | |
|-----------------------------------|------------|
| User A 's public key | n_A, v_A |
| User A 's private signature key | s_A |

Table 5-2 Key format when verification exponent is freely selectable

5.3.3 Key generation for DS1

The first step is to define the required length $|n|$, in bits, of the modulus n . The bit length has an impact on the security and performance of the technique used. A large modulus enhances security but slows down the transformations.

When implementing the TBSS, the length $|n|$ of the modulus n must be at least 512 bits.

5.3.3.1 Public verification exponent

With systems that use fixed public exponents, the verification exponent is predefined for all users. Both the use and the selection of a fixed exponent are governed by the application-specific integration concept.

With systems that use freely selectable public exponents, each user is free to choose the verification exponent v . This selection should be a random one.

5.3.3.2 Prime number generation

Each user generates two large prime numbers p and q . These prime numbers must satisfy the following criteria:

1. The product pq must have the required bit length $|n|$.
2. The numbers p and q must be randomly selected strong primes.
3. $p-1$ and $q-1$ must be relatively prime to the public exponent v .
4. The prime numbers p and q must be kept secret.
5. The error probability in stochastic prime number generation must be less than 2^{-64} .

Procedures for prime number generation are described in Annex C.

5.3.3.3 Modulus

The user's modulus n is calculated as the product of the two secret prime numbers p and q , $n = pq$.

5.3.3.4 Secret signature exponent

The user's secret signature key comprises his signature exponent s . This exponent is calculated as the smallest positive integer s , so that $sv-1$ is a multiple of the least common multiple of $p-1$ and $q-1$.

5.4 DSH - Digital signature with data hashing

In general the messages or user data to be processed are longer than one signature block. In such cases DS1 (ISO/IEC 9796 with RSA) has to be used in combination with a hash function. During verification the hash value is first restored from the digital signature, and is then compared with the locally computed hash value in the received message.

For general use, the digital signature DS1 has to be combined with one of the available hash functions (HF1, HF2 or HF3):

- DSH1: Digital signature 1 with hash function 1 (DES based)
- DSH2: Digital signature 1 with hash function 2 (RIPEMD-128)
- DSH3: Digital signature 1 with hash function 3 (RIPEMD-160)

In this chapter we simply use the abbreviation DSH, since the selection of the hash function does not have any effect on the processing steps.

5.4.1 Signature process

The next figure shows the signature process in detail.

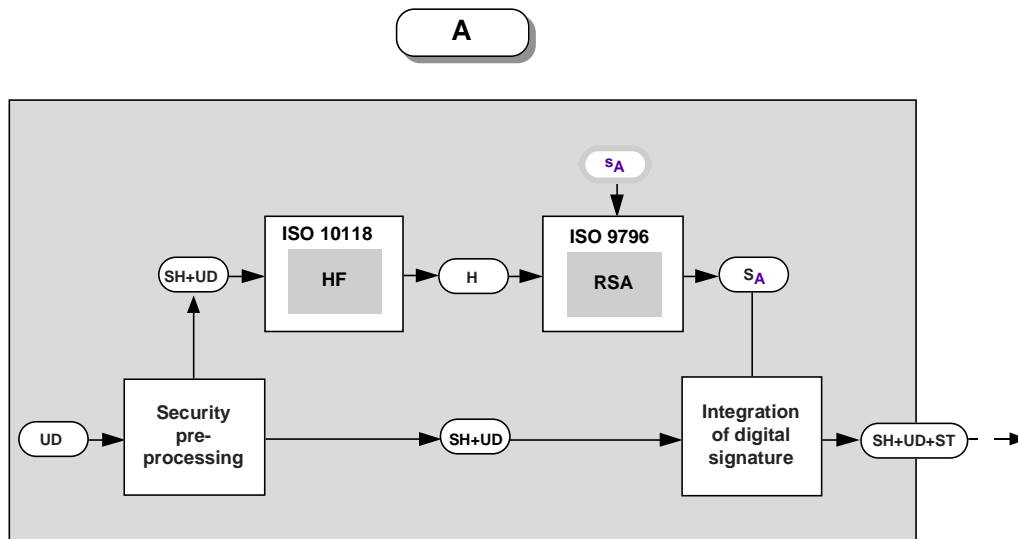


Figure 5-4 Signature process with ISO/IEC 9796

1. The user data UD are expanded to include the security header SH.
2. The expanded user data (SH+UD) are forwarded to the hash function.
3. The resulting hash value H is signed using the secret signature key s_A (ISO/IEC 9796 signature transformation).
4. The resulting signature is combined with the expanded user data, i.e. integrated in the security trailer ST. The signed user data comprises SH+UD+ST.

5.4.2 Verification process

The following figure shows the verification process in detail.

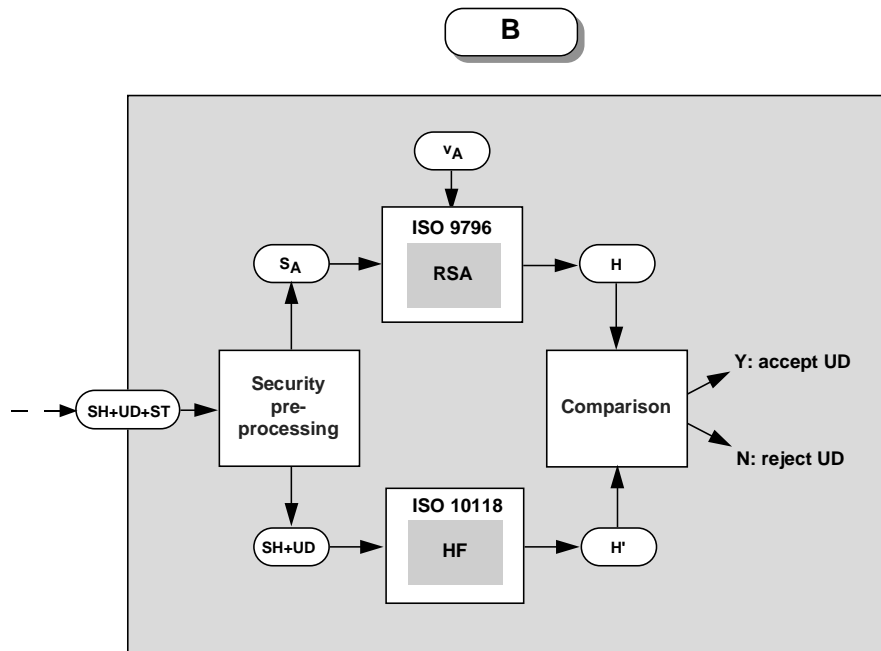


Figure 5-5 Verification process with ISO/IEC 9796

1. The secured data are separated. The signature S_A is extracted from the ST and forwarded to the ISO/IEC 9796 verification transformation. The expanded user data are forwarded to the hash function.
2. The signature is restored using the sender's public verification key and verified (ISO/IEC 9796 verification transformation). The result is the sender's hash value H . If verification is not successful, the digital signature is immediately rejected at this stage.
3. The hash value of the expanded user data (SH+UD) is calculated locally with the hash function. The result is the receiver's hash value H' .
4. The two hash values H and H' are compared with each other. The user data or their signature are only accepted if $H=H'$.

5.5 Hash function

The role of the hash function is to reduce arbitrarily long data sequences to a compact, cryptologically secure checksum. No secret elements, such as keys, are needed to perform the hash function.

When used with digital signatures, the hash function H must satisfy the following security requirements:

1. H must be a one-way function: For the message M and the relevant hash value y , it must be hard to find a different message M' , so that $H(M') = y$.
2. H must be collision resistant: it must be hard to find two different messages M and M' whose hash values collide, i.e. $H(M) = H(M')$.

The security requirements are essential. Otherwise it would be easy to forge digital signatures, irrespective of the quality of the signature process.

The following hash functions are currently approved for use with the TBSS:

- HF1: ISO/IEC 10118-2 Blockcipher- based hash function using DES as the basic algorithm.
- HF2: ISO/IEC 10118-3 Dedicated hash function using RIPEMD-128 as the basic algorithm.
- HF3: ISO/IEC 10118-3 Dedicated hash function using RIPEMD-160 as the basic algorithm.

5.5.1 Recommendations

The following recommendations apply to the selection of a hash function for a specific application:

1. Because of the performance enhancement use, wherever possible, a dedicated hash function (that is, HF2 or HF3).
2. Because of the security enhancement use, wherever possible, dedicated hash function HF3 (RIPEMD-160).
3. HF2 provides a hash function for application environments where 128 bit hash-codes are required.
4. HF1 provides a hash function for application environments where DES is available but a dedicated hash function cannot readily be implemented.

5.5.2 MD strengthening

Security can be significantly enhanced, irrespective of the hash function used, by employing the following MD strengthening procedure.

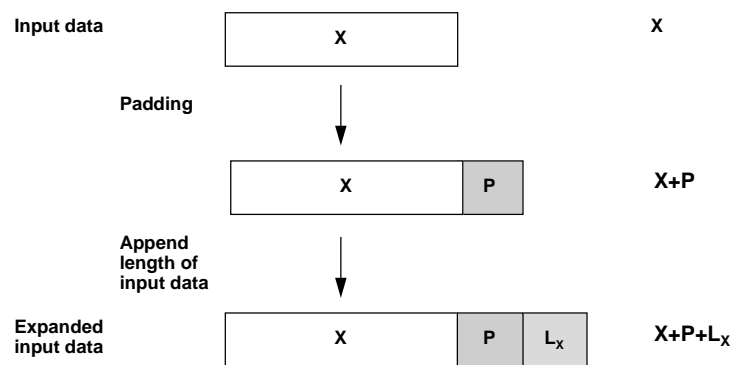


Figure 5-6 MD strengthening

The input for the hash function typically consists of the security header and user data (SH+UD), but its composition may vary depending on the type of application. In the following description we therefore use X as a generic designator for the input data.

1. The length L_X of the input data X (in bits) is calculated. L_X is written left justified in a separate 64-bit block (represented as an integer $0 < L_X < 2^{64}$).
2. The data X are filled out with padding bits (P) if necessary, so that $X+P+L_X$ satisfies the hash function's length condition. The padding bits and the length condition are defined in the hash function.
3. The block defining the length of the data is appended, so that the final format of the expanded input data is $X+P+L_X$.

MD strengthening can be seen as part of the hash function. ISO/IEC 10118-3 *Dedicated hash functions* already incorporates this method.

5.6 HF1 - ISO/IEC 10118-2 with DES

Part 2 of ISO/IEC 10118 defines hash functions based on n -bit block ciphers. The following conventions apply when using DES in the TBSS:

1. The DES data encryption algorithm is specified in ANSI X3.92 - 1981. The block length is $n = 64$ bit.

2. The length of the hash code is $L_H = 128$ bit; i.e. only the double length hash method is used (in compliance with Clause 7 of ISO/IEC 10118-2).
3. A data block D comprises 64 bits (8 bytes), with the first bit (byte) in a left justified position.

5.6.1 Key transformations u and u'

DES has a block length of $n = 64$ bits and a key length of $L_K = 56$ bits. Since block and key length are different, transformations u and u' must be defined which map the hash value H_{i-1} on the key value K_i (or H'_{i-1} on K'_i).

Let's assume that the binary representation of the 64-bit block X is $x_1x_2x_3\dots x_{64}$. The transformations u and u' should be selected as in 10118-2 Annex A *Use of DEA*:

Transformation u:

1. Deletion of bits $x_8, x_{16}, x_{24}, x_{32}, x_{40}, x_{48}, x_{56}, x_{64}$
2. Setting of bits $(x_2, x_3) = (1,0)$

Transformation u':

1. Deletion of bits $x_8, x_{16}, x_{24}, x_{32}, x_{40}, x_{48}, x_{56}, x_{64}$
2. Setting of bits $(x_2, x_3) = (0,1)$

5.6.2 Initialization vectors

The initialization vectors are produced as follows:

$$IV = IV_0 \oplus D_1$$

$$IV' = IV'_0 \oplus D_1$$

where D_1 is the first data block of the data to be hashed and IV_0 and IV'_0 are set in accordance with 10118-2 Annex A *Use of DEA*:

$$IV_0 = '5252525252525252' \text{ (in hexadecimal notation)}$$

$$IV'_0 = '2525252525252525' \text{ (in hexadecimal notation)}$$

Reasoning: if constant initialization vectors are widely used, the probability of a random collision becomes greater with time (birthday paradox). This risk is avoided since the initialization vectors are message-dependent.

5.6.3 Padding rule

Since MD strengthening is used for all hash methods, we can select method 1 from 10118-1 Annex B *Padding Methods*:

The following padding rule applies to data which are not a multiple of $n=64$ bits:

Method 1: padding the block with 0s

5.7 HF2 - ISO/IEC 10118-3 RIPEMD-128

HF2 conforms to ISO/IEC 10118-3, clause 8, Dedicated hash-function 2. For HF2, the hash-code shall have the maximum length of 128 bits.

5.7.1 Hash-code computation

The hash-code is computed using the following steps:

1. Padding: the input data string is padded to a multiple of 512 bits.
2. Splitting: the padded data string is split into 512-bit blocks, D_1, D_2, \dots, D_q .
3. Iteration: using the round function ϕ and an initializing value IV , the data blocks are hashed iteratively as follows:

$$H_0 = IV$$

$$H_i = \Phi(D_i, H_{i-1}); \quad i = 1, \dots, q$$

where H_i denote the intermediate hash values of length 128 bits.

4. The final hash value H_q is directly taken as the hash-code H of the hash function.

5.7.2 Padding rule

Padding comprises the following steps:

1. Padding of data X to a multiple of 512 bits minus 64 bits according to the following rule: append a "1" and then as many "0"s as necessary. Padding is also performed if the message is already a multiple of 512 bits.
2. The length L_x of the data X (in bits) is written with least significant byte first (left) into the free 64-bit block (i.e. MD strengthening is already integrated in the padding rule).

See also the description of MD strengthening earlier in this chapter.

5.7.3 Round function and initializing value

Round function and initializing value are specified in ISO/IEC 10118-3, clause 8.

5.8 HF3 - ISO/IEC 10118-3 RIPEMD-160

HF3 conforms to ISO/IEC 10118-3, clause 7, Dedicated hash-function 1. For HF3, the hash-code shall have the maximum length of 160 bits.

5.8.1 Hash-code computation

The hash-code computation applicable to HF3 follows the model described in section 5.7.1 with the following differences:

1. The intermediate hash values H_i have length 160 bits.
2. The round function ϕ and the initializing value IV are different.

5.8.2 Padding rule

The padding rule applicable to the use of HF3 shall be the same as the one defined in section 5.7.2.

5.8.3 Round function and initializing value

Round function and initializing value are specified in ISO/IEC 10118-3, clause 7.

6 Symmetric encryption

This chapter describes symmetric techniques for encrypting user data (without associated key management). The following mechanisms are approved:

- SC1 Data Encryption Standard (DES) in accordance with ANSI X3.92. This method is important for backward compatibility. Many systems and standards - especially in the financial sector - are based on DES. After almost 20 years, however, the useful life of DES is slowly approaching its end.
- SC2 The International Data Encryption Algorithm (IDEA) in accordance with the "ISO Register of Cryptographic Algorithms" Nr ISO 9979/0002. IDEA has a key length of 128 bits and is suited for high-security applications. IDEA has become one of the most widely used encryption algorithms worldwide through its use in several Internet protocols and applications (e.g. PGP; SSL).

Both DES and IDEA are block cipher algorithms with a block length of 64 bits. The native Electronic Code Book mode (ECB) is not sufficient on its own, except in very few application scenarios. Because of this, ISO/IEC 10116 defines the following four modes of operation for block ciphers:

1. Electronic Code Book (ECB) mode
2. Cipher Block Chaining (CBC) mode
3. Cipher Feedback (CFB) mode
4. Output Feedback (OFB) mode

The characteristics and potential uses of these different modes are described in Annex B.

SC stands for Symmetric Cryptosystem. The next figure provides an overview of the available mechanisms. Other encryption mechanisms may be approved for TBSS use at a later stage.

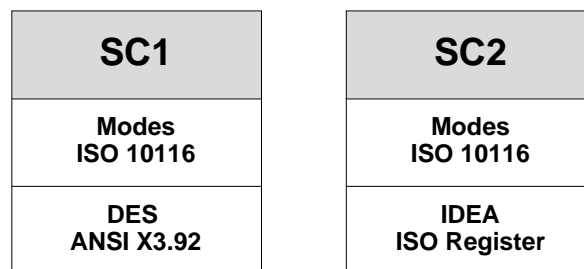


Figure 6-1 Overview of encryption mechanisms

6.1 Guidelines for usage and key length

The following guidelines apply to the choice of key length and the use of SC1 and SC2 in new applications:

- For authentication of high-value transactions or for long-term encryption, a minimum key length of 90 bits is recommended.
- Usage of symmetric cryptosystems with a key length of less than or equal to 64 bits (e.g. single-DES) is restricted to authentication of low-value transactions or for short-time encryption.

6.2 Implementation

Encryption can either be activated in batch mode (offline) or in dialog mode (online). Both these implementations have different requirements as regards the encryption method's mode of operation.

Basically a distinction is made between two implementation modes for data encryption:

1. Batch mode: user data are encrypted offline. This mode is necessary if there is no direct communications link between A and B (e.g. store-and-forward) or if the security applications are not directly linked (e.g. isolated file transfer).
2. Dialog mode: the user data are encrypted online. This mode is necessary if A and B are holding an interactive dialog (session) and the user data (or data key) are not known in advance.

6.2.1 Batch mode

In batch mode the user data are encrypted offline prior to transmission. For example, the user data can be transmitted as an electronic message or as a file with the help of a suitable communication protocol. Batch mode is required for all implementations where A's and B's applications have no direct contact, i.e. are not holding an interactive dialog. Communication via store-and-forward protocols is a typical implementation. In the description that follows, 'message (M)' is used as a generic designator for user data prepared offline.

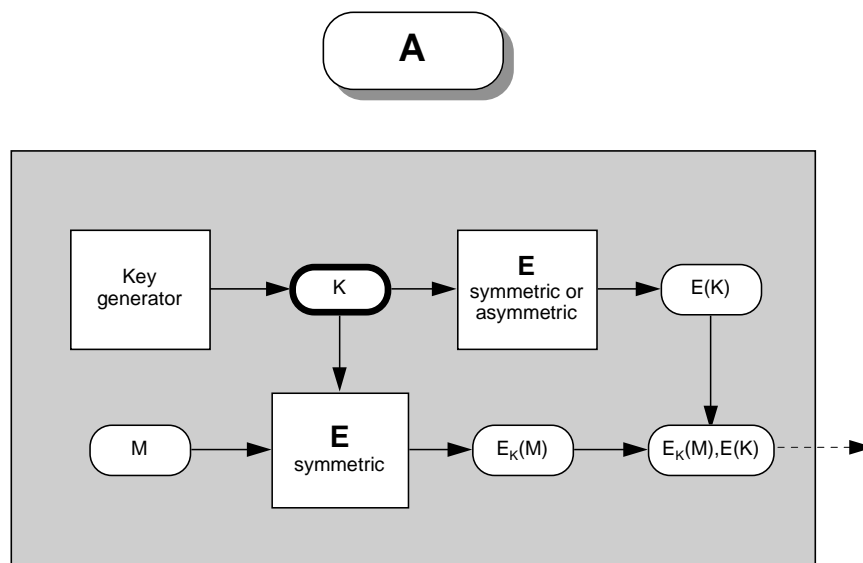


Figure 6-2 Encryption in batch mode

Encryption in batch mode entails the following steps:

1. User A has a message M which he wants to encrypt prior to transporting to user B. A generates a key K . K is a dynamic message key which is only used for this message M and can be deleted again afterwards.
2. The message M is encrypted using a symmetric cryptosystem under the control of K . The result is $E_k(M)$.
3. The dynamic message key K is encrypted using a symmetric or asymmetric cryptosystem. The result is $E(K)$.

If a symmetric cryptosystem is used, A and B must have exchanged a common secret master key MK beforehand. If an asymmetric cryptosystem is used, A must hold B's public encryption key e_B .

4. $E_K(M)$ and $E(K)$ are collated in a security message. The encrypted message key $E(K)$ is inserted in the security header, and the encrypted message $E_K(M)$ forms the data part of the secured message. The encrypted data may need to be filtered as well, depending on the communication protocol used.

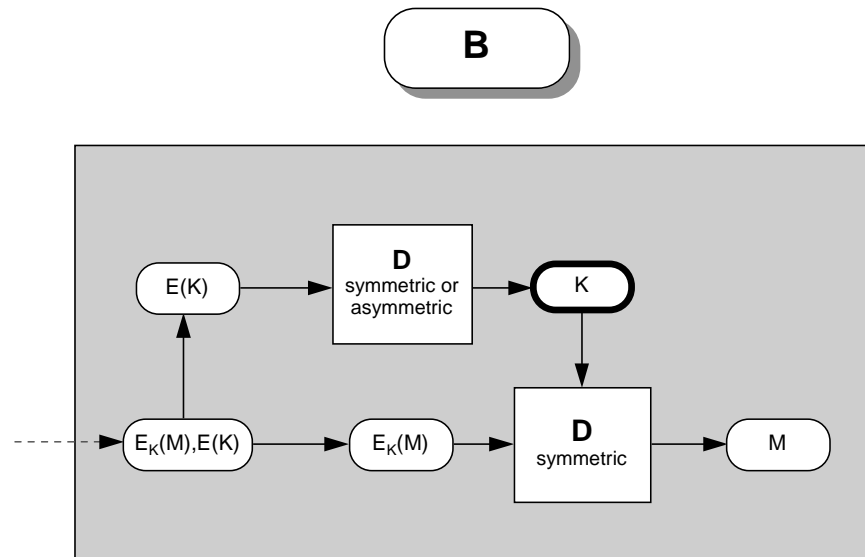


Figure 6-3 Decryption in batch mode

Decryption in batch mode entails the following steps:

1. User B receives the secured message, comprising the security header and encrypted user data. If the encrypted data were filtered at the sender's end, the filter process first has to be reversed.
2. B extracts the encrypted message key $E(K)$ from the security header and deciphers it with the help of the symmetric or asymmetric cryptosystem. The result is K .
3. The encrypted message $E_K(M)$ is deciphered with the help of the symmetric cryptosystem, under the control of the dynamic message key K . The result is M . Decryption can be verified by performing a check on M (e.g. redundancy or explicit message authentication).

If a dynamic message key K is used for encryption in batch mode, it can be encrypted using a symmetric or asymmetric cryptosystem. Since key management is simpler for asymmetric cryptosystems, the asymmetric variant is recommended if digital signatures are already deployed between A and B .

6.2.2 Dialog mode

In dialog mode, user data are encrypted online (i.e. during the communications session between A and B). Interactive exchange of encrypted data is possible in this mode.

In the figure that follows, it is assumed that a communications link has already been established between A and B . Encryption in dialog mode involves the following steps:

1. One-way/mutual authentication of communication partners.
2. Agreement of common session key. This can be combined with the authentication mechanism.
3. Encrypted dialog. Encryption does not have to apply to the whole session. It is possible to encrypt just the user data (or parts of it) in one or both directions.

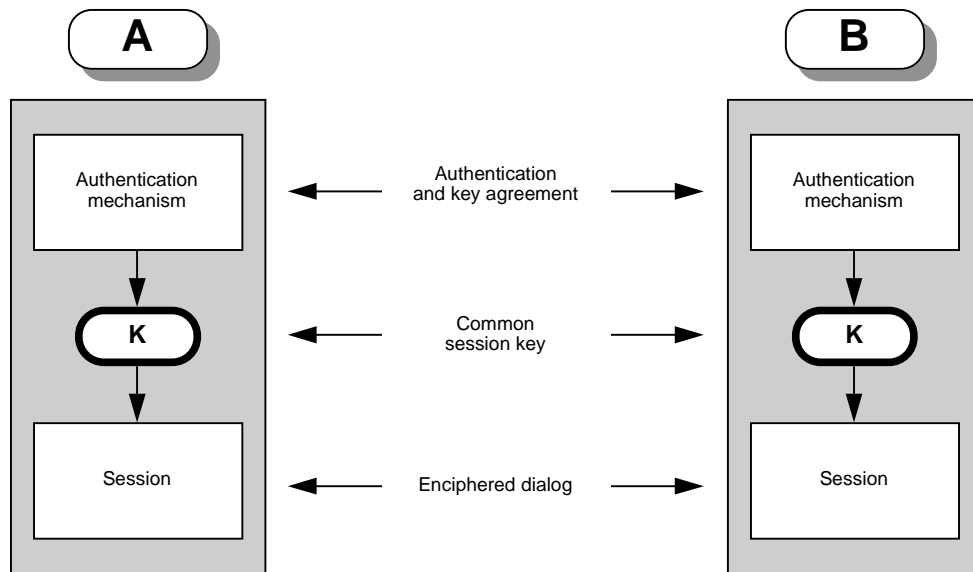


Figure 6-4 Encryption in dialog mode

Examples of combined key transport and authentication can be found in the relevant chapters on these topics.

6.2.3 Key generation

Users *A* and *B* must be able to generate dynamic data keys for encryption using a symmetric cryptosystem. These data keys can either be generated with a true random number generator or a pseudorandom number generator.

Key generation techniques are described in Annex C.

6.3 SC1 - DES with ISO/IEC 10116

DES (Data Encryption Standard) is specified in the standards FIPS PUB 46 and ANSI X3.92. DES is a block cipher algorithm with a block length of 64 bits and a 56-bit key. DES was originally published back in 1977 and is the most widely used symmetric encryption algorithm. In 1993 DES was approved (and supported) as a standard by the U.S. organization NIST for a further 5 years. This technique is important for backward compatibility. Many systems and standards are based on DES - especially in the financial sector.

The direct implementation of DES in Electronic Code Book (ECB) mode is adequate in only very few cases. There are other modes of operation for the block cipher technique, most of them internationally standardized. The section on ISO/IEC 10116 deals with recommendations on padding, initialization vectors and the implementation of the various DES operating modes.

6.3.1 Implementation

The following rules apply for the implementation of DES in the TBSS:

1. Weak and semi-weak keys are not admissible for encryption, i.e. they must be filtered out through tests.

2. The section on ISO/IEC 10116 deals with recommendations on padding, initialization vectors and the implementation of the various standard modes of operation.

6.4 SC2 - IDEA with ISO/IEC 10116

The International Data Encryption Algorithm (IDEA) was developed at the Swiss Federal Institute of Technology in 1990 and has become one of the most widely used encryption algorithms worldwide through its use in several Internet protocols and applications (e.g. PGP; SSL). IDEA operates with 64-bit plaintext and ciphertext blocks and has a key length of 128 bits.

IDEA is a registered algorithm in the "ISO Register of Cryptographic Algorithms" Nr ISO 9979/0002 and the UN/EDIFACT code list.

The major references for IDEA are:

- X.Lai and J.L.Massey, "A Proposal for a New Block Encryption Standard", Advances in Cryptology, Proc. Eurocrypt'90, pp.389-404. This paper contains the first version of IDEA, called PES (Proposed Encryption Standard).
- X.Lai, J.L.Massey and S.Murphy, "Markov Ciphers and Differential Cryptanalysis", Adv. in Cryptology-Eurocrypt'91, pp.17-38. This paper gives some results on the differential cryptanalysis of the PES cipher. An improvement of PES was suggested, called IPES (Improved Proposed Encryption Standard). Later, the name was changed to IDEA (Nov. 1991).

6.4.1 Implementation

The following rules apply for the implementation of IDEA in the TBSS:

1. The all-zero key is not admissible for encryption, i.e. it must be filtered out through tests.
2. The section on ISO/IEC 10116 deals with recommendations on padding, initialization vectors and the implementation of the various standard modes of operation.

6.5 Other encryption mechanisms

Based on criteria such as key length, export restrictions, international acceptance and implementation constraints other encryption mechanisms may be added to the TBSS at a later stage. See the Annex for a general survey of available encryption mechanisms. Two candidates that have been considered for approval are Triple-DES and SAFER SK128.

6.5.1 Triple DES

As DES is the most widely used encryption algorithm, a natural way to enhance the security level is to cascade several DES operations with different keys. At least three operations are required for a real improvement. ISO 8732 defines the basic Triple DES encryption operations as

$$C = DES_{K1} (DES_{K2}^{-1} (DES_{K3}(M)))$$

Based on ISO 8732, ANSI has developed a working draft, ANSI X9.52-19XX Triple Data Encryption Algorithm Modes of Operation, which currently contains 10 modes of operations and 3 keying options.

Triple DES is not one of the approved encryption mechanisms of the TBSS. However, Triple DES may be used as a temporary solution in application environments where it is absolutely necessary to enhance security but where it is deemed economically infeasible to directly switch to one of the approved encryption mechanism. Use of Triple DES

in an application requires the choice of one of the 10 modes of operation and one of the 3 keying options. As ANSI X9.52-19XX becomes stable, Triple DES may be re-evaluated for approval in the TBSS.

6.5.2 SAFER SK128

The Secure And Fast Encryption Routine (SAFER) was proposed by Massey in 1993. SAFER is an iterated block cipher with 64 bit data blocks, a 64 bit and a 128 bit key mode. SAFER is designed especially for implementation on 8-bit microprocessors, for high performance and low memory requirements.

SAFER is not one of the approved encryption mechanisms of the TBSS. However, as SAFER SK128 is interface-compatible with IDEA, it provides an alternative or fall-back mechanism for IDEA (SC2). SAFER SK128 may be re-evaluated for approval in the TBSS. Prerequisites are a stable public level of acceptance and an international registration, e.g. in the ISO Register of Cryptographic Algorithms.

6.6 ISO/IEC 10116

ISO/IEC 10116 specifies the different modes of operation for general n-bit block cipher algorithms. This section provides implementation recommendations on these modes of operation when used in conjunction with DES or IDEA.

6.6.1 Electronic Code Book (ECB) mode

In ECB mode, each data block M is encrypted individually, independently of the other data blocks. This means that two identical data blocks will always result in two identical ciphertext blocks, as long as the key K remains the same.

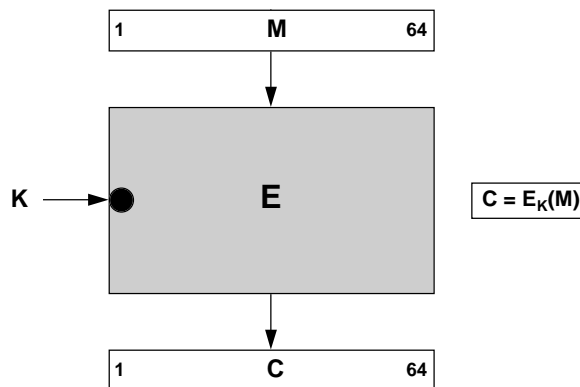


Figure 6-5 Electronic Code Book mode

The characteristics and potential uses of ECB mode are described in Annex B.

6.6.1.1 Padding

DES and IDEA have a data block length of 64 bits. Any user data that are shorter than one data block should always be padded with random bits. The general guideline is that the user data space (2^{64} possible data blocks) should be utilized as fully as possible.

6.6.2 Cipher Block Chaining (CBC) mode

In CBC mode, data blocks are concatenated at the sender's end. The encryption of a data block always takes into account the preceding ciphertext block (see figure). An initialization vector IV is used to encrypt the first data block (where no previous ciphertext block is available). A ciphertext block therefore depends on all the previous user data blocks and the IV.

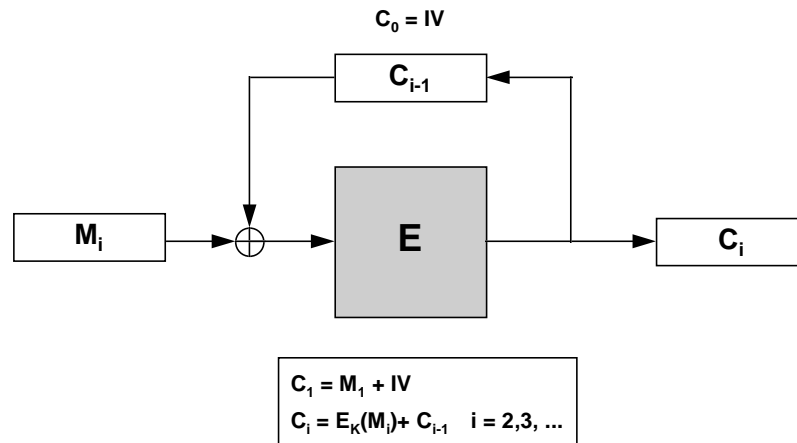


Figure 6-6 Cipher Block Chaining (CBC) mode complying with ISO/IEC 10116

The characteristics and potential uses of CBC mode are described in Annex B.

6.6.2.1 Padding

As long as the resulting data expansion achieved through padding is not a problem, the following rule should be observed:

Padding of user data to a multiple of 64 bits by appending a "1" and then as many "0"s as necessary. This rule also applies if the user data are already a multiple of 64 bits.

6.6.2.2 Initialization vector (IV)

With CBC the initialization vector need not be kept secret.

1. Where the initialization vector is constant, the value

$IV = '5A5A5A5A5A5A5A5A'$ (in hexadecimal notation)

is recommended.

2. Where the initialization vector is variable, the receiver must know the value used for encryption. The IV can be generated by a combination of known sender and receiver information (e.g. addresses); in this case the IV need not be transmitted to the receiver. The IV can also be generated each time (e.g. with the help of a pseudorandom number generator); it then needs to be transmitted to the receiver. If dynamic data keys are used the IV can be encrypted along with the data key.

6.6.2.3 Implementation

CBC is the recommended mode of operation for batch encryption. CBC can also be used for dialog encryption, depending on the implementation requirements.

6.6.3 Cipher Feedback (CFB) mode

In the CFB mode, the data encryption block size can be set to 1 - 64 bits (for 64-bit encryption algorithms). Typical block sizes are 1, 8, 64 (bit, character or block encryption). The input to the encryption algorithm is supplied by the previous ciphertext elements, and not by the user data as in the case of ECB and CBC modes. The output of the encryption algorithm produces a stream of key elements, which are added to the user data elements to form the ciphertext elements. The first block cipher operation requires a 64-bit initialization vector. A ciphertext element therefore depends on all the preceding user data elements and the IV.

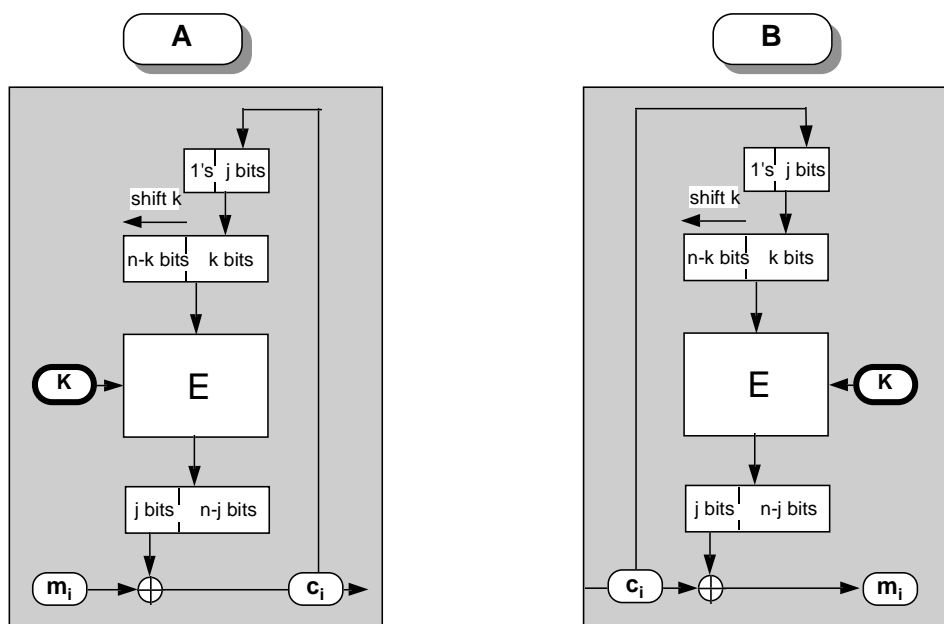


Figure 6-7 Cipher Feedback (CFB) mode complying with ISO/IEC 10116

The characteristics and potential uses of CFB mode are described in Annex B.

6.6.3.1 Padding

This is not usually required in CFB mode, since the element lengths are adjusted to the user data formats. But if for some reason the user data are not a multiple of the element length, there are two alternatives:

1. Padding with '0's (data expansion)
2. Truncation of the superfluous bits of the last key element (no data expansion).

6.6.3.2 Initialization vector (IV)

1. It is recommended to use a variable initialization vector for TBSS implementations. The IV can be chosen at random or worked out from bilateral counters. If the IV is chosen at random, the value used for encryption must be notified to the receiver. If dynamic data keys are used the IV can also be encrypted along with the data key.

2. On the other hand, there is no need to keep the IV secret, since in CFB mode the input for the encryption algorithm (i.e. the ciphertext) is already known.

6.6.3.3 Implementation

It is recommended to use $j=k$ for TBSS implementations. Typical lengths are $j=1,8$ and 64 bits.

CFB is suitable for use in dialog mode, especially if data elements shorter than 64 bits have to be encrypted individually.

6.6.4 Output Feedback (OFB) mode

In OFB mode the block cipher algorithm is used to generate a key stream (pseudorandom sequence). The user data are encrypted through bit-by-bit addition with the key stream. The input block of the block cipher must be loaded with a randomly selected initialization vector. The output block (the result of the encryption operation) is fed back into the input register of the block cipher. A key stream element of size j is extracted from the output block.

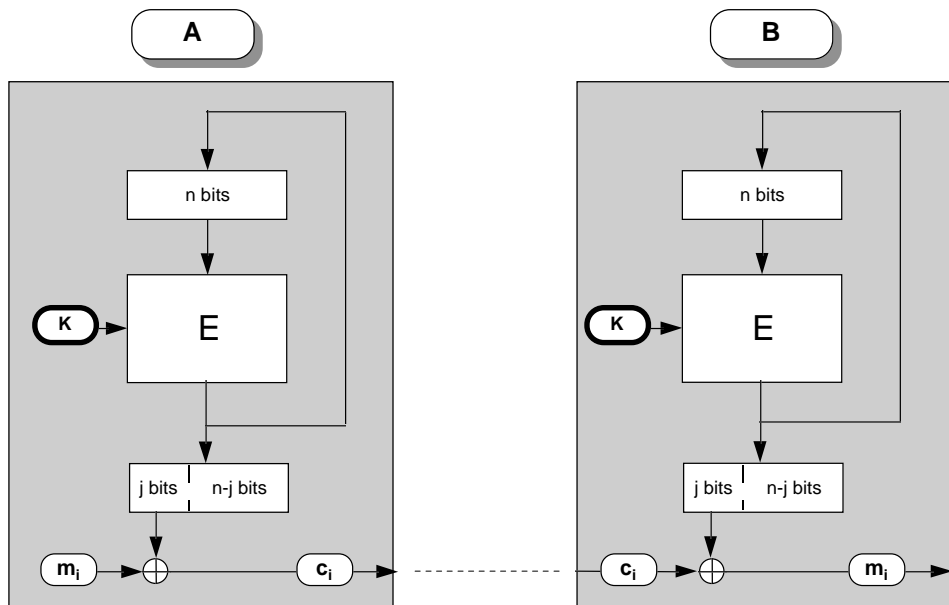


Figure 6-8 Output Feedback (OFB) mode complying with ISO/IEC 10116

The characteristics and potential uses of OFB mode are described in Annex B.

6.6.4.1 Padding

This is not usually required in OFB mode, since the element lengths are adjusted to the user data formats. But if for some reason the user data is not a multiple of the element length, there are two alternatives:

1. Padding with '0's (data expansion)
2. Truncation of the superfluous bits of the last key element (no data expansion).

6.6.4.2 Initialization vector (IV)

1. It is recommended to use a variable initialization vector for TBSS implementations (if the key stays the same). The IV value used for encryption must be notified to the receiver.

2. It is also recommended to keep the IV secret. If dynamic data keys are used the IV can be encrypted along with the data key.

6.6.4.3 Implementation

Typical lengths are $j=1,8$ and 64 bits.

OFB is suitable for use as a pseudorandom generator and for dialog encryption, especially with synchronous data transmission and high speeds.

7 Asymmetric encryption

The TBSS requires the use of an asymmetric cryptosystem (AC) in key transport (key management) and entity authentication (encryption of authentication token). Key management and authentication data are referred to under the generic term of protocol data in this chapter.

In principle an asymmetric cryptosystem can also be used for direct encryption of user data, but this is seldom done in practice for performance reasons.

This chapter describes asymmetric cryptosystems. The following mechanisms are approved for TBSS use:

- AC1 Asymmetric cryptosystem using RSA. RSA was developed in 1978 by Rivest, Shamir and Adleman at the MIT and already provides the basis for the digital signature in compliance with ISO/IEC 9796 (DS1).

AC stands for asymmetric cryptosystem. The next figure gives an overview of the mechanisms.

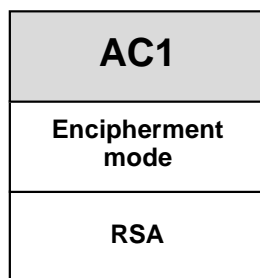


Figure 7-1 Overview of asymmetric encryption mechanisms

The specifications contained in this chapter also allow asymmetric encryption of individual optional data blocks. The AC1 mechanism performs this function if the stages Coding (7.1.1) and Decoding (7.1.6) of protocol data are omitted.

7.1 AC1 - RSA for protocol data

This section specifies the implementation of RSA for the encryption of protocol data. RSA can also be used for security management. In this case protocol data which have to be kept secret (e.g. keys and initialization vectors for symmetric encryption) are encrypted with the help of the receiver's public key. The following points must be observed:

1. Protocol parameters must be coded in a way that enables the receiver to identify the parameter's type and function.
2. A data block that is compatible with asymmetric encryption has to be created. In this process, aspects such as redundancy, security and data representation need to be taken into account.

When using RSA as an asymmetric cryptosystem, the sender needs the receiver's public encryption key. This consists of the modulus n and the encryption exponent e . The receiver's secret decryption key consists of the exponent d .

| | |
|------------------------|-------------------|
| Public encryption key | n, e |
| Private decryption key | d |
| Encryption function | $y = x^e \bmod n$ |
| Decryption function | $x = y^d \bmod n$ |

Table 7-1 Asymmetric encryption: key pairs and functions

The two function pairs (signature, verification) and (encryption, decryption) can be realized with RSA using one asymmetric key pair (for both function pairs) or two separate key pairs (one for each function pair).

Given the receiver's modulus n has length $|n|$ then there are $\lceil |n|/8 \rceil - 10$ bytes available for data block encryption ($\lceil x \rceil$ means x rounded up to the next integer). Example: when $|n|=521$ there are 56 bytes or 448 bits available for data encryption.

7.1.1 Coding of protocol data

User A creates the protocol data block

$$BP = Q_1 // P_1 // \dots // Q_m // P_m$$

which in each case comprises

- Q Qualifier (1 byte) specifying the type, function and length of the parameter that follows
- P Protocol parameters

Protocol parameters can be keys, initialization vectors, challenges and responses. The qualifier is coded as follows (based on EDIFACT):

| Qualifier | Meaning |
|-----------|--------------------------------|
| 0 | not used |
| 1 | 64-bit key for encryption |
| 2 | 64-bit key for authentication |
| 11 | 64-bit initialization vector |
| 21 | 64-bit random number |
| ... | |
| 101 | 128-bit key for encryption |
| 102 | 128-bit key for authentication |
| 111 | 128-bit initialization vector |
| 121 | 128-bit random number |
| ... | |
| 201 | 256-bit key |
| ... | |
| 255 | Agreed bilaterally |

Table 7-2 Possible code values for the qualifier

7.1.2 Preliminary processing

The protocol data BP must be expanded into a data block BD which is suitable for asymmetric encryption. To do so the sender produces

$$BD = 0\|A\|L_{BP}\|BP\|P$$

which comprises

- 0 The zero byte at the start of the data block ensures that BD , interpreted as an integer, is smaller than the receiver's modulus.
- A The sender address enhances the security (the encrypted data block cannot be misused by other parties) and the integrity of the protocol data (the address provides verifiable redundancy). 8 bytes are provided for A .
- L_{BP} The length of the protocol data is specified in order to differentiate between the protocol data BP and subsequent padding. 1 byte is provided for L_{BP} .
- BP The protocol data to be encrypted. $\lceil n/8 \rceil - 10$ bytes are available ($\lceil x \rceil$ means x rounded up to the next integer).
- P A padding string comprising (pseudo) random bits. This is needed to securely pad the data block to the length of the receiver's modulus.

Note: If the 8 bytes provided are not enough for the sender address A , this can be replaced by a 64-bit hash result of the data block (including address) if necessary. This option provides an explicit integrity check on the whole data block.

7.1.3 Encryption

The data block BD is interpreted as a positive integer x . The first bit of BD forms the most significant bit of the number x and the last bit of BD forms the least significant bit of the number x . The integer x is encrypted with the receiver's public key (n, e)

$$y = x^e \text{ mod } n$$

The resulting integer y is interpreted as an encrypted block BE . The most significant bit of the number y forms the first bit of BE and the least significant bit of the number y forms the last bit of BE . As an option, the block BE can also be filtered (e.g. HEX filter) if the communications channel is not bit-transparent.

7.1.4 Decryption

The encrypted block BE is interpreted as a positive integer y . The first bit of BE forms the most significant bit of the number y and the last bit of BE forms the least significant bit of the number y . The integer y is decrypted with the receiver's secret key (n, d)

$$x = y^d \text{ mod } n$$

The resulting integer x is interpreted as a data block BD . The most significant bit of the number x forms the first bit of BD and the least significant bit of the number x forms the last bit of BD . If necessary BD is padded with zero bits to a multiple of 8.

7.1.5 Verification and extraction

The receiver of the data block BD verifies the structure and content of the data block by performing the following tests:

1. The first byte of the block BD must be 0
2. The sender address A in the encrypted data block must match the sender address in the protocol message,

Then the protocol data block BP is extracted from the data block BD with the help of the length parameter L_{BP} .

7.1.6 Decoding of protocol data

Using the qualifiers, the parameters in the protocol data block are extracted one at a time and forwarded to their destination. Keys and initialization vectors are used for the encryption and decryption of user data. Challenges and responses are processed for authentication purposes.

8 Entity authentication

Different entity authentication mechanisms have to be provided depending on the type of application. The following mechanisms are described:

| Mechanism | Passes | Authentication | Basic mechanism | Additional requirements |
|-----------|--------|----------------|-------------------|-------------------------------|
| EA1 | 1 | one-way | Digital signature | Timestamp or sequence numbers |
| EA2 | 2 | one-way | Digital signature | Random numbers |
| EA3 | 3 | mutual | Digital signature | Random numbers |

EA stands for Entity Authentication. The next figure provides an overview of the structure and interaction of entity authentication mechanisms.

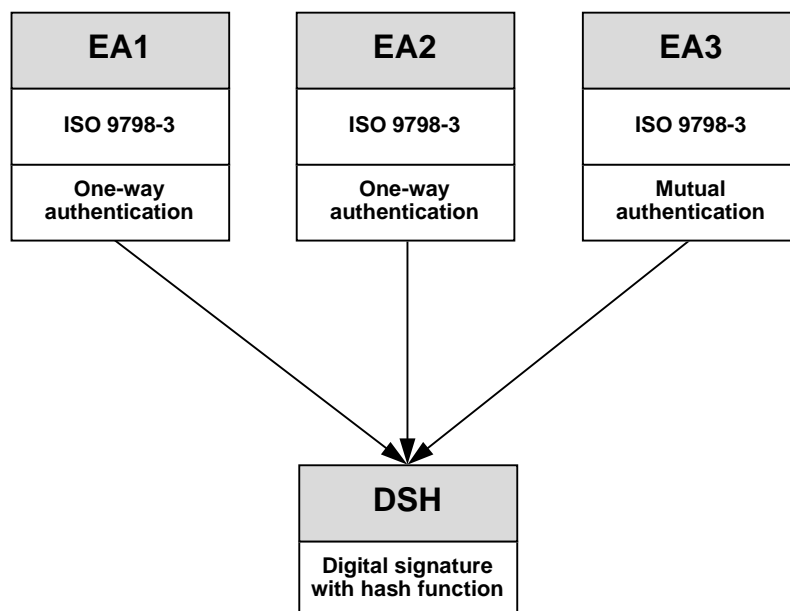


Figure 8-1 Overview of entity authentication mechanisms

In the context of this chapter, *token* refers to a data packet transmitted during a pass; the abbreviation AT is used for authentication token.

8.1 One-way authentication (EA1)

This authentication mechanism is based on the use of a digital signature combined with timestamps or sequence numbers. User A starts the protocol and provides direct authentication of himself in the first step.

8.1.1 Technical description

The prerequisites for this mechanism are as follows:

1. User A has a digital signature with the functions (S_A, V_A) .

2. User *B* has access to an authenticated copy of *A*'s verification key v_A .
3. Users *A* and *B* can either manage synchronous clocks or bilateral counters (a separate counter for each communications link).

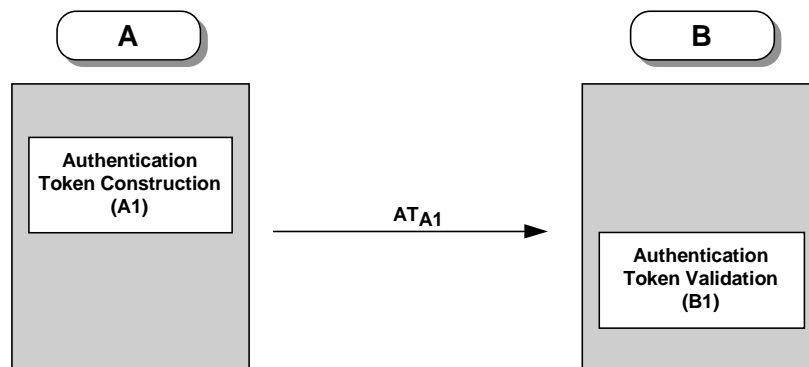


Figure 8-2 EA1 One-way authentication with one pass

Entity Authentication 1 (EA1)

Authentication token construction (A1) *A* generates the authentication data, comprising a *TVP* (timestamp or sequence number), the distinguished name *B* of the receiver and the optional data field *Data1*. *A* then signs these data with its private signature key and sends the authentication token

$$AT_{A1} = S_A(TVP || B || Data1) || Data2$$

to *B*. *Data2* is an optional unsigned data field.

Authentication token validation (B1) *B* checks the validity of *A*'s verification key. *B* then checks the *TVP* (timestamp or sequence number), verifies the digital signature of the token AT_{A1} and checks the receiver identifier *B*.

8.1.2 Characteristics

This authentication mechanism has the following characteristics:

1. Standards: compliance with ISO/IEC 9798-3, Clause 5.1.1, *One pass authentication*.
2. Authentication: one-way, from *A* to *B*.
3. Number of passes: 1.
4. Data field *TVP*: must contain a timestamp or sequence number. When using timestamps both *A* and *B* must have synchronized clocks. When using sequence numbers both *A* and *B* must control bilateral counters.
5. Data field *B*: *B*'s name must be included in the authentication data to prevent the token being misused for authentication attempts with other parties.
6. Data fields *Data1* and *Data2*: for greater flexibility, optional data fields are defined in addition to the mandatory data fields. Depending on how the mechanism is implemented, these fields can be filled with additional data elements. The purpose of the relevant application-specific Implementation Guidelines is to define the use of these optional data fields. Points 7 and 8 provide examples of possible uses.
7. Key distribution: this authentication mechanism can also be used both for transporting a secret key *K* from *A* to *B* (ISO/IEC 11770-3). A new data field is set up for key transport in the optional data field. To this end a public key cryptosystem must be available. See Key Transport Mechanism 1 in the chapter on key transport.
8. Public key certificates: if necessary *A*'s certificate can be supplied in the unsigned data field *Data2*.

8.2 One-way authentication (EA2)

This authentication mechanism is based on the use of the digital signature combined with the classic challenge-and-response technique. User *B* authenticates *A* in two passes.

8.2.1 Technical description

The prerequisites for this mechanism are as follows:

1. User *A* has a digital signature with the functions (S_A, V_A) .
2. User *B* has access to an authenticated copy of *A*'s verification key v_A .
3. Users *A* and *B* can generate random numbers. Procedures for pseudorandom number generation are described in Annex C.

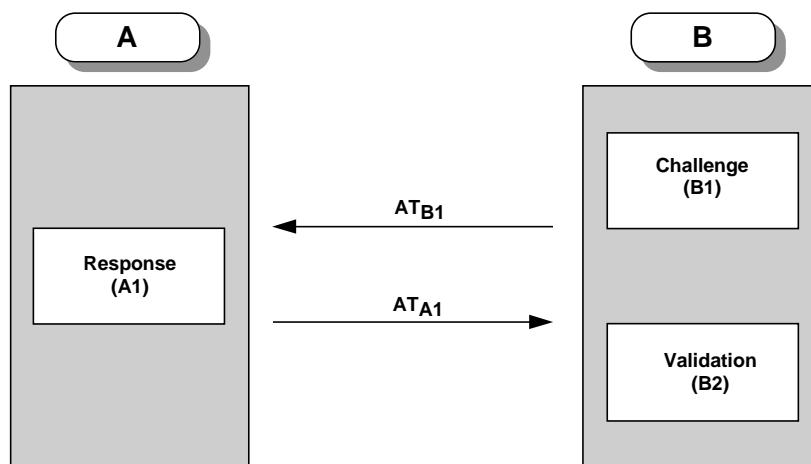


Figure 8-3 EA2 One-way authentication with two passes

Entity Authentication 2 (EA2)

Challenge (B1) *B* produces the challenge AT_{B1} , consisting of a random number r_B and an optional data field *Data1*,

$$AT_{B1} = r_B || Data1$$

and sends it to *A*.

Response (A1) *A* generates a second random number r_A and produces the authentication data, consisting of r_A , r_B , the receiver ID *B* and optional data. *A* then signs these data with its private signature function S_A and sends the authentication token

$$AT_{A1} = S_A(B || r_A || r_B || Data2) || Data3$$

as a response to *B*.

Validation (B2) *B* checks the validity of *A*'s verification key. *B* then verifies the digital signature of the token AT_{A1} and checks the receiver address *B*. Finally it checks that the random number r_B used in the challenge matches with the one contained in the response.

8.2.2 Characteristics

Authentication mechanism 2 has the following characteristics:

1. Standards: compliance with ISO/IEC 9798-3, Clause 5.1.2, *Two pass authentication*.
2. Authentication: one-way from *A* to *B*, based on the classic Challenge-and-response technique.
3. No. of passes: 2.
4. Data field r_B : contains *B*'s challenge .
5. Data field r_A : prevents *A* from signing data that are outside its control.
6. Data field *B*: *B*'s name must be included in the authentication data to prevent the token from being misused for authentication attempts with other parties.
7. Data fields *Data1*, *Data2* and *Data3*: for greater flexibility, optional data fields are defined in addition to the mandatory data fields. Depending on how the mechanism is implemented, these fields can be filled with additional data elements. The purpose of the relevant application-specific Implementation Guidelines is to define the use of these optional data fields. Point 8 provides an example of a potential implementation.
8. Public key certificates: if necessary *A*'s certificate can be supplied in the unsigned data field *Data2*.

8.3 Mutual authentication (EA3)

This authentication mechanism is based on the use of digital signatures.

8.3.1 Technical description

The prerequisites for this mechanism are as follows:

1. Each user *A* and *B* has a digital signature with the functions (S_A, V_A) and (S_B, V_B) , respectively.
2. Each user has access to an authenticated copy of the other user's verification key.
3. Users *A* and *B* can generate random numbers. Procedures for pseudorandom number generation are described in Annex C.

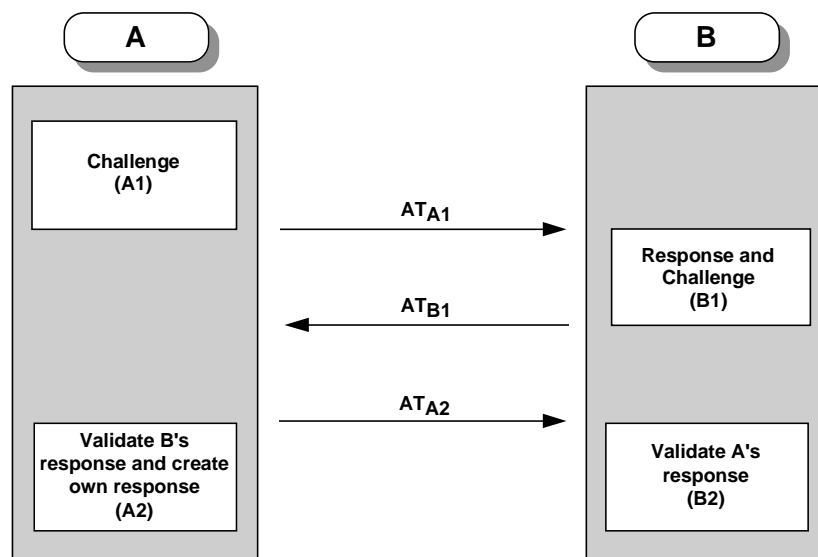


Figure 8-4 EA3 Mutual authentication with three passes

Mutual authentication (EA3)

Challenge (A1) *A* produces the challenge AT_{A1} , consisting of a random number r_A and an optional data field *Data1*,

$$AT_{A1} = r_A || Data1$$

and sends it to *B*.

Response and Challenge (B1) *B* generates a second random number r_B and produces the authentication data, consisting of r_B , r_A , the identifier *A* and optional data. *B* then signs these data with its private signature function S_B and sends the authentication token

$$AT_{B1} = S_B(A || r_B || r_A || Data2) || Data3$$

to *A* as a simultaneous response and challenge.

Validate B's response and create own response (A2) *A* checks the validity of *B*'s verification key. *A* then verifies the digital signature of the token AT_{B1} and checks the receiver identifier *A*. It then checks that the random number r_A contained in the challenge (A1) matches with the one contained in the response (B1). If all the tests are OK, *B* is successfully authenticated by *A*.

A then generates its own authentication data, consisting of r_A , r_B , the identifier *B* and optional data. *A* then signs these data with its private signature function S_A and sends the authentication token

$$AT_{A2} = S_A(B || r_A || r_B || Data4) || Data5$$

as a response to *B*.

Validate A's response (B2) *B* checks the validity of *A*'s verification key. *B* then verifies the digital signature of the token AT_{A2} and checks the receiver identifier *B*. Finally it checks that the random number r_B used in the challenge (B1) matches with the one contained in the response (A2). If all the tests are OK, *A* is successfully authenticated by *B*.

8.3.2 Characteristics

Authentication mechanism 2 has the following characteristics:

1. Standards: compliance with ISO/IEC 9798-3, Clause 5.2.2, *Three pass authentication*.
2. Authentication: mutual, based on the classic Challenge-and-response technique combined with digital signature.
3. No. of passes: 3.
4. Data fields *A* and *B*: the receiver identifier must be included in the signed authentication data to prevent the token being misused for authentication attempts with other parties.
5. Data fields r_A and r_B : contain the relevant challenge of the communication partner. r_B must be signed in AT_{B1} to prevent *B* from signing data which are entirely under *A*'s control. By the same analogy, r_A must be signed in AT_{A2} to prevent *A* from appending a digital signature to data which are entirely under *B*'s control.
6. Data fields *Data1*, *Data2*, *Data3*, *Data4* and *Data5*: for greater flexibility, optional data fields are defined in addition to the mandatory data fields. Depending on how the mechanism is implemented, these fields can be filled with additional data elements. The purpose of the relevant application-specific Implementation Guidelines is to define the use of these optional data fields. Point 7 provides an example of a potential implementation.
7. Public key certificates: if necessary the certificates of whichever party sent the token can be supplied in the unsigned data fields *Data1*, *Data3*, *Data5*.

9 Key transport

This chapter describes mechanisms for exchanging secret keys for use with symmetric cryptosystems. Key transport must always be authenticated. For this reason, all key transport mechanisms are also entity authentication mechanisms.

Different key transport mechanisms have to be provided depending on the type of application:

- SKT1 Key transport (one-pass) with one-way authentication, based on timestamps or sequence numbers. This mechanism is an extension of EA1.
- SKT2 Key transport (two-pass) with one-way authentication, based on random numbers. This mechanism is an extension of EA2.
- SKT3 Key transport (three-pass) with mutual authentication, based on random numbers. This mechanism is an extension of EA3.

SKT stands for Secret Key Transport. The next figure provides an overview of the structure and interaction of key transport mechanisms.

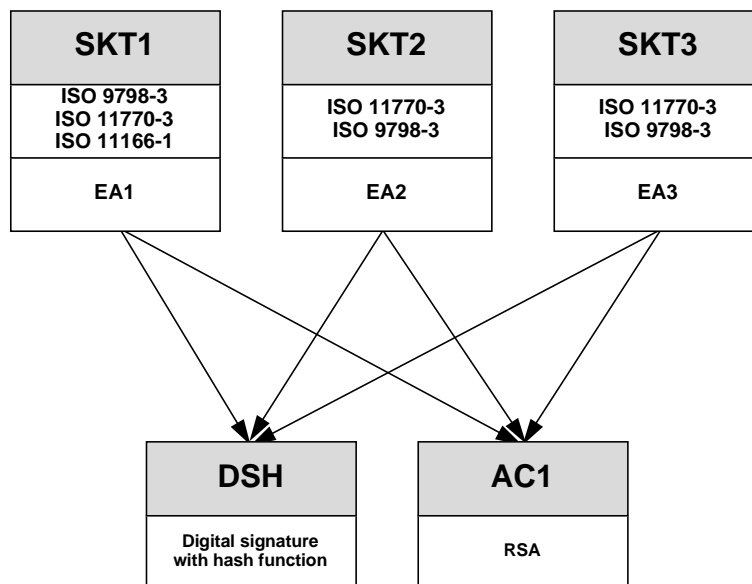


Figure 9-1 Overview of key transport mechanisms

In the context of this chapter, *token* refers to a data packet transmitted during a pass. For the sake of clarity, the abbreviation KT stands for key token.

9.1 Key transport 1 (SKT1)

This mechanism is based on the entity authentication mechanism EA1. *A* sends *B* an encrypted key along with the authentication token. The optional data field *Data1* is replaced by the encrypted block *BE* and the data field *Data2*. This provides a form of mutual authentication. *A* is explicitly authenticated by the use of the *TVP* and its signature transformation. *B* is implicitly authenticated by the use of its public encryption transformation, since it means that only *B* is able to decrypt the data.

9.1.1 Technical description

Prerequisites:

1. Both users have their own asymmetric cryptosystem (E_X, D_X) and asymmetric signature system (S_X, V_X) .
2. User *A* has access to an authenticated copy of *B*'s public encryption key and user *B* has access to an authenticated copy of *A*'s public verification key.

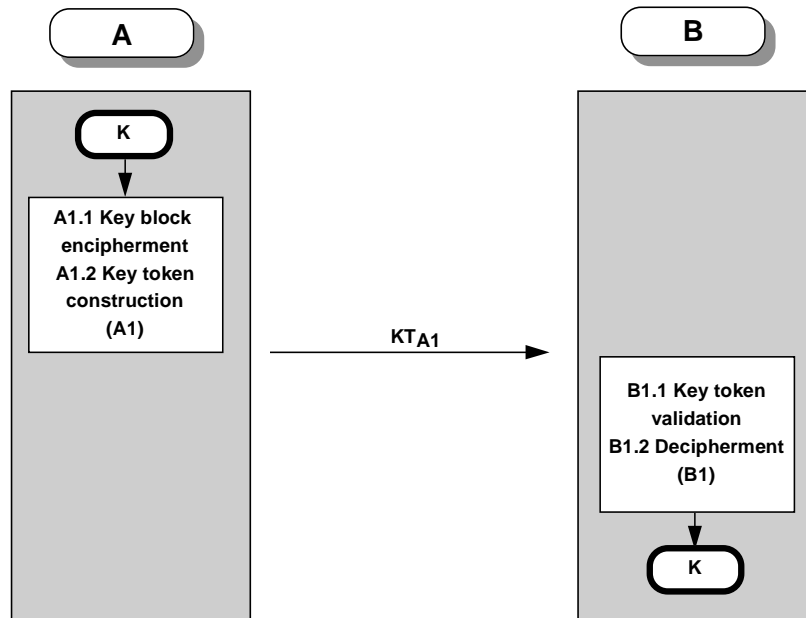


Figure 9-2 SKT1 key transport with authentication

Key transport 1 (SKT1)

Key block encipherment (A1.1) *A* generates a key K and produces a data block comprising the sender identifier A , the key K and the optional data field $Data1$. Then *A* encrypts the data block using the receiver's public encryption transformation E_B and generates the encrypted block,

$$BE = E_B(A||K||Data1)$$

Key token construction (A1.2) *A* produces the transport data, consisting of a TVP (timestamp or sequence number), the receiver identifier B , the encrypted block BE and the optional data field $Data2$. *A* then signs these data with its private signature transformation and sends the key token

$$KT_{A1} = S_A(TVP||B||BE ||Data2)||Data3$$

to *B*.

Key token validation (B1.1) *B* checks the validity of *A*'s verification key. *B* then checks the TVP (timestamp or sequence number), verifies *A*'s digital signature of the key token KT_{A1} and also checks the receiver identifier B .

Decipherment (B1.2) *B* decrypts the block BE using its private decryption transformation D_B . Then *B* checks the sender identifier A . If all the tests are OK, *B* accepts the key K .

9.1.2 Characteristics

This key transport mechanism has the following characteristics:

1. Standards: compliance with ISO/IEC 11770-3, *Key Transport Mechanism 2*. Compliance with ISO/IEC 9798-3, Clause 5.1.1, *One pass authentication*, with the optional data field *Data1* replaced by the encrypted block *BE* and the data field *Data2*. Compliance with ISO 11166-1, *Banking - Key management by means of asymmetric algorithms, Part 1: Principles, procedures and formats*, Clause 11.1.9, Methods (b).
2. No. of passes: 1.
3. Authentication: mutual. *B* is implicitly authenticated by the use of its public encryption transformation. *A* is explicitly authenticated by the use of the *TVP* and its signature transformation. On the other hand, *A* can only be sure that *B* really received the token, and therefore the key, once he receives a response from *B*.
4. Key confirmation: from *A* to *B*.
5. Replay: of the token is impossible.
6. Key control: *A*'s responsibility.
7. Data field *TVP*: must contain a timestamp or sequence number. When using timestamps both *A* and *B* must have synchronized clocks. When using sequence numbers both *A* and *B* must control bilateral counters.
8. Data field *A*: *A*'s identifier must be part of the key data, so that the key's origin is specified.
9. Data field *B*: *B*'s name must be included in the authentication data in *EA1* to prevent the token from being misused for authentication attempts by other parties.
10. Data fields *Data1*, *Data2* and *Data3*: for greater flexibility, optional data fields are defined in addition to the mandatory data fields. Depending on how the mechanism is implemented, these fields can be filled with additional data elements. The purpose of the relevant application-specific Implementation Guidelines is to define the use of these optional data fields. Point 11 provides an example of a potential implementation.
11. Public key certificates: if necessary *A*'s certificate can be supplied in the unsigned data field *Data3*.

9.2 Key transport 2 (SKT2)

This mechanism is based on the entity authentication mechanism *EA2*. User *B* authenticates *A* in two passes. The digital signature is combined with the classic challenge-and-response technique. At the same time key transport takes place from *A* to *B* with key confirmation. *B* is implicitly authenticated by the use of its public encryption transformation, since only *B* is able to decrypt the data.

9.2.1 Technical description

The same prerequisites apply as for the authentication mechanism *EA2*, plus:

1. User *B* has his own asymmetric cryptosystem (E_B, D_B).
2. User *A* has access to an authenticated copy of *B*'s public encryption key.

For a description of the passes, see the figure on entity authentication *EA2*.

Key transport 2 (SKT2)

Challenge (B1) *B* produces the challenge KT_{B1} , consisting of a random number r_B and an optional data field *Data1*,

$$KT_{B1} = r_B || Data1$$

and sends it to *A*.

Encryption (A1.1) *A* generates a key *K* and produces a data block comprising the sender identifier *A*, the key *K* and the optional data field *Data2*. Then *A* encrypts the data block using the receiver's public encryption transformation E_B and generates the encrypted block.

$$BE = E_B(A||K||Data2)$$

Response (A1.2) *A* signs the token data, comprising the receiver identifier *B*, the new random number r_A (optional), the random number r_B , the encrypted block *BE*, and optional data with its private signature transformation S_A and sends the token

$$KT_{AI} = S_A(B||r_A||r_B||BE||Data3)||Data4$$

as a response to *B*.

Check response (B2.1) *B* checks the validity of *A*'s verification key. *B* then verifies the digital signature of the token KT_{AI} and checks the receiver identifier *B*. Finally it checks that the random number r_B used in the challenge matches with the one in the response.

Decryption (B2.2) *B* decrypts the block *BE* using its private decryption transformation D_B . Then he checks the sender identifier *A*. If all the tests are OK, *B* accepts the key *K*.

9.2.2 Characteristics

This key transport mechanism has the same characteristics as the authentication mechanism EA2, plus:

1. Authentication: explicitly from *A* to *B*, based on the classic challenge-and-response technique, implicitly from *B* to *A*, since only *B* is able to decrypt the block *BE*.
2. Key confirmation: from *A* to *B*.
3. Key control: *A*'s responsibility
4. Data field r_A : optional here - the presence of *BE* in KT_{AI} makes it superfluous.
5. Data field *BE*: the encrypted block contains the key *K* and, as an option, additional secret elements, such as an initialization vector. *BE* replaces the data field r_A in EA2 and at the same time prevents *A* from having to sign data that are outside its control.

9.3 Key transport 3 (SKT3)

This mechanism is based on the entity authentication mechanism EA3. The users mutually authenticate each other based on digital signatures combined with the classic challenge-and-response technique. At the same time key transport occurs from *A* to *B* with key confirmation.

9.3.1 Technical description

The same prerequisites apply as for the authentication mechanism EA3, plus:

1. User *B* has his own asymmetric cryptosystem (E_B, D_B) .
2. User *A* has access to an authenticated copy of *B*'s public encryption key.

For a description of the passes, see the figure on entity authentication EA3.

Key transport 3 (SKT3)

Challenge (A1) *A* produces the challenge KT_{AI} , consisting of a random number r_A and an optional data field *Data1*,

$$KT_{AI} = r_A || Data1$$

and sends it to *B*.

Response and Challenge (B1) *B* generates a second random number r_B and produces the authentication data, consisting of the partner identifier *A*, the random numbers r_B and r_A and the optional data. *B* then signs these data with its private signature transformation S_B and sends the token

$$KT_{B1} = S_B(A || r_B || r_A || Data2) || Data3$$

to *A* as a simultaneous response and challenge.

Validate *B*'s response (A2.1) *A* checks the validity of *B*'s verification key. *A* then verifies the digital signature of the token KT_{B1} and checks the receiver identifier *A*. It then checks that the random number r_A used in the challenge (A1) and response (B1) matches. If all the tests are OK, *B* is successfully authenticated by *A*.

Encryption (A2.2) *A* generates a key K and produces a data block comprising the sender identifier *A*, the key K and the optional data field $Data4$. Then *A* encrypts the data block using the receiver's public encryption transformation E_B and generates the encrypted block.

$$BE = E_B(A || K || Data4)$$

Response (A2.3) *A* produces the authentication and key data, consisting of the partner identifier *B*, the random number r_A (optional), the random number r_B , the encrypted block BE , and optional data. *A* then signs these data with its private signature transformation S_A and sends the token

$$KT_{A2} = S_A(B || r_A || r_B || BE || Data5) || Data6$$

as a response to *B*.

Validate response (B2.1) *B* checks the validity of *A*'s verification key. *B* then verifies the digital signature of the token KT_{A2} and checks the receiver identifier *B*. Finally it checks that the random number r_B used in the challenge (B1) and response (A2) matches. If all the tests are OK, *A* is successfully authenticated by *B*.

Decryption (B2.2) *B* decrypts the block BE using its private decryption transformation D_B . Then he checks the sender identifier *A*. If all the tests are OK, *B* accepts the key K .

9.3.2 Characteristics

This key transport mechanism has the same characteristics as the authentication mechanism EA3, plus:

1. Standards: compliance with ISO/IEC 11770-3, Clause 6.5, *Key Transport Mechanism 5*, based on ISO/IEC 9798-3, Clause 5.2.2, *Three pass authentication*.
2. Key confirmation: from *A* to *B*.
3. Key control: *A*'s responsibility
4. Data field r_A : no longer required because of the presence of BE in KT_{A2} .
5. Data field BE : the encrypted block contains the key K and, as an option, additional secret elements, such as an initialization vector.

10 Public key transport without certificates

The authenticated registration and distribution of the public verification keys of the customers and service providers is a prerequisite for the provision of secure electronic communication and services. Only after the authenticated exchange of public keys is it possible to use digital signatures and other security mechanisms (compare the prerequisites of the various mechanisms). The following figure illustrates the typical telebanking relationships:

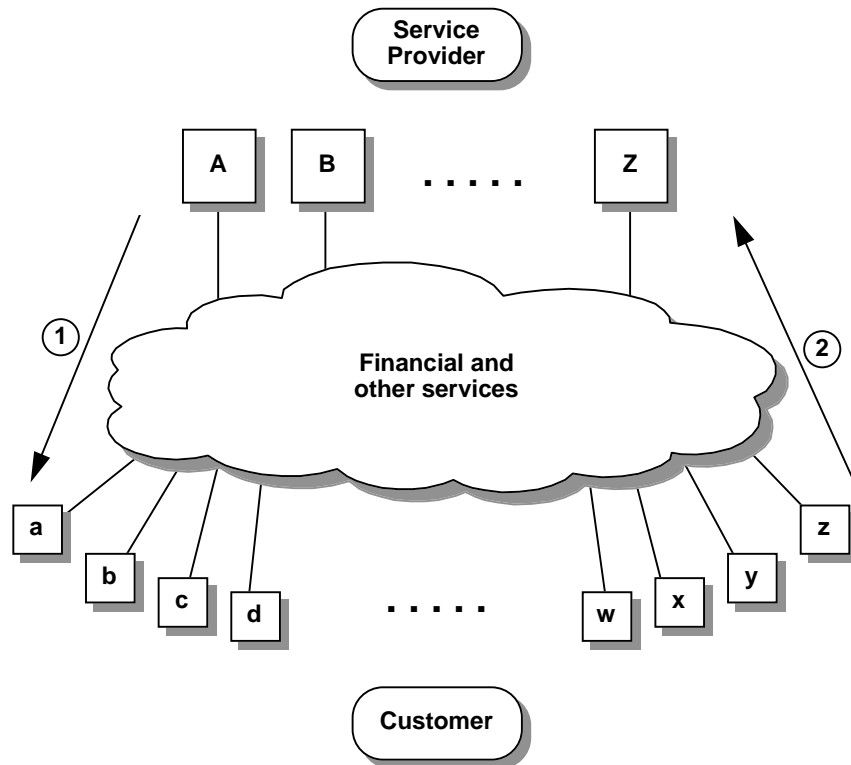


Figure 10-1 Typical telebanking relationships

In a telebanking situation the service provider provides its services to a multitude of customers. The resulting topology is typically a star and requires the following mechanisms for the distribution of public keys (see figure above):

1. Distribution of the verification key of the service provider to all its customers.
2. Distribution of the verification key of a customer to the service provider.

This chapter specifies a pragmatic key management without certificates (and the corresponding certification authorities). The described public key transport mechanisms are sufficient to satisfy the requirements of telebanking scenarios. They are also sufficient to exchange public keys bilaterally. The fundamentals of this asymmetric key management without certificates are:

- Each participant (customer as well as service provider) generates its own asymmetric key pair, consisting of a secret signature key and a public verification key.
- The public verification key of a service provider can be made available in one of several ways to the customers: delivery with the initial software package; delivery by diskette or electronic transfer and associated independent letter; or by publishing the public key.

Public Key Transport Mechanism 1 or a modification of Public Key Transport Mechanism 2 can achieve this type of distribution procedure.

- The customer sends its public verification key to the service provider by electronic transfer. To provide a contractual basis and to authenticate the public key the customer also sends a manually signed letter containing the hash value of the key data (the public key) to the service provider. This procedure can be executed for each service provider.
- Public Key Transport Mechanism 2 can achieve this type of distribution procedure.

The public key transport mechanisms can not only be used to initialize a new subscriber, but also to change keys of existing subscribers. Reasons for a key change are for instance:

1. Loss of a signature key.
2. Suspected or proved compromise of a signature key.
3. Periodic key change.

It is recommended to treat a key change in the same way as an initial registration and not to rely on the security of the old key pair. Confidentiality is not required for the exchange of public keys. However, a central requirement is the assurance of integrity and authenticity of the public keys. The distribution of public encryption keys can be handled in the same way (ie using the mechanisms in this chapter) as the distribution of public verification keys.

The described Public Key Transport Mechanisms are generally applicable and may also be used between two service providers or between two customers.

10.1 Public key transport mechanism 1 (PKT1)

If *A* has access to an authenticated channel to *B* (such as a courier or registered mail) then *A* may transport its public key information directly via that authenticated channel to *B*. This is the most elementary form of transferring a public key.

10.1.1 Technical description

Requirements:

1. There is an authenticated channel between *A* and *B*.

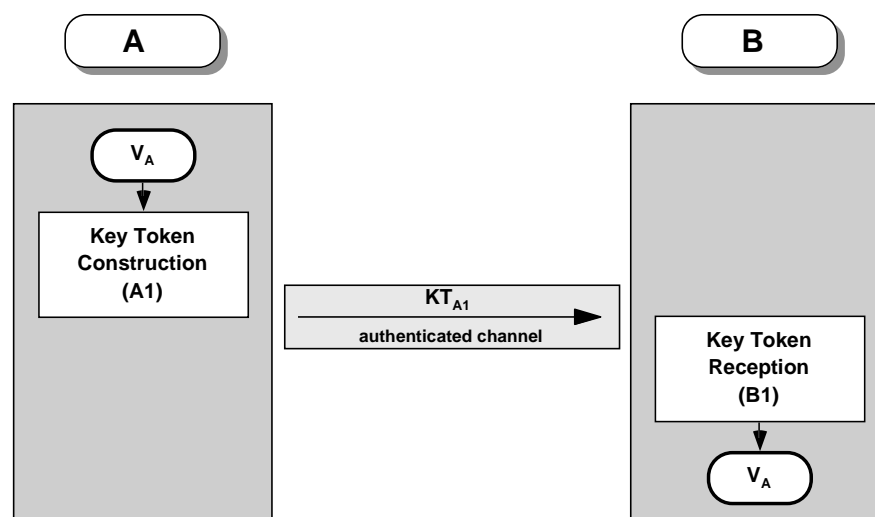


Figure 10-2 Public key transport using an authenticated channel.

Public Key Transport 1 (PKT1)

Key token construction (A1) *A* has obtained an asymmetric key pair and wants to register its public verification key v_A with *B*. *A* constructs the key token, consisting of the sender identifier *A*, the public key serial number *N*, the personal verification key v_A , the validity period *VAL*, the date *DATE* and an optional data field *Data1*. Then *A* sends, using the authenticated channel, the key token

$$KT_{AI} = A//N//v_A//VAL//DATE//Data1$$

to *B*.

Key Token Reception (B1) *B* receives the key token KT_{AI} via the authenticated channel from *A*, which may require some verification steps. Then *B* extracts the public verification key v_A and the other public key information from the key token, stores it in the local data base and activates user *A*.

10.1.2 Properties

This Public Key Transport Mechanism has the following properties:

1. Standards: conformance with ISO/IEC CD 11770-3, *Public Key Transport Mechanism 1*, and with X.509 regarding the structure of the public key information.
2. Communication channel: since verification keys are public, the channel need not offer confidentiality. But authenticity of the public key information is essential. In this context authenticity includes both data integrity and data origin authentication.
3. *Data1*: usage of the optional data field is to be specified in the application-specific implementation guidelines.
4. This mechanism can analogously be used for the distribution and registration of the public key e_A of an asymmetric encipherment system.
5. Possible implementations of this mechanism are:
 - Delivery (e.g. on a diskette) using registered mail or a courier.
 - Joint delivery of the service provider verification key together with the telebanking software. This is possible, when the telebanking software is distributed from a central place. The delivery must be in an authenticated manner.
 - Publishing of the telebanking service provider verification key in a suitable and authenticated medium (e.g. a newspaper). This method can also be used to validate a public key which was distributed using a different (non-authenticated) channel.
 - The properties of the authenticated channel can also be achieved, if the public key data are transferred using an insecure channel, but a third party independently vouches for the authenticity of the public key data.
6. This mechanism is suitable for the transport of public key information from a telebanking service provider to its customers.

10.2 Public key transport (PKT2)

This mechanism transfers the public key information of entity *A* via an unprotected channel to *B*. To verify the integrity and the origin of the received public key information a second authenticated channel is used. Such a mechanism is useful when the public key information is transferred electronically on a high bandwidth channel, whereas the authentication of the public key information takes place over an authenticated low bandwidth channel. This second channel is implemented using a written and manually signed application form. This application may form the contractual basis for the use of digital signatures from *A* to *B*.

10.2.1 Technical description

This mechanism is based on the following requirements:

1. Entity *A* has a personal asymmetric signature system with transformations (S_A, V_A) .
2. Both entities share a common cryptologically secure hash function h .

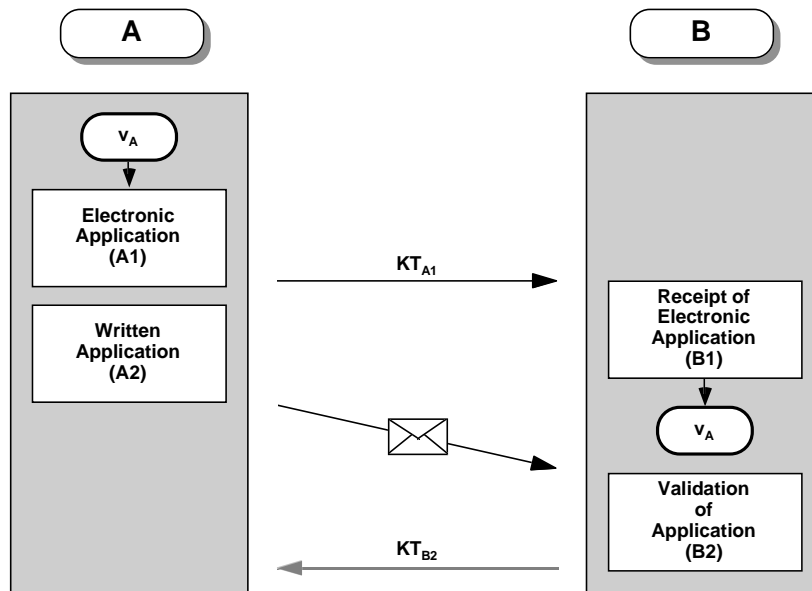


Figure 10-3 Authentication using a second channel

Public Key Transport 2 (PKT2)

Electronic Application (A1) *A* has an asymmetric key pair and wishes to register its public verification key v_A with *B*. *A* constructs a data block DB_A , consisting of its sender identifier *A*, the key serial number N , the personal verification key v_A , the validity period Val , the date $DATE$ and the optional data field $Data1$,

$$DB_A = A||N||v_A||Val||DATE||Data1$$

Then *A* signs the hash value of the data block DB_A with its private signature transformation and sends the key token

$$KT_{A1} = DB_A||S_A(h(DB_A)||Data2)$$

to *B*.

Written Application (A2) *A* prints the application form AF_A , containing the information of the data block DB_A , the hash value $h(DB_A)$ and some optional data field $Data3$:

$$AF_A = DB_A||h(DB_A)||Data3$$

Then the application form is manually signed by an authorised representative of *A* and sent via an authenticated channel to *B* (e.g. mail, registered mail, courier). This application form may provide the legal basis for the automatic processing of *A*'s digital signatures at *B* (using *A*'s registered public verification key v_A).

Receipt of Electronic Application (B1) *B* receives *A*'s electronic application KT_{A1} , extracts the public verification key v_A and verifies *A*'s digital signature. Then *B* checks the $DATE$, the validity period Val and loads the public key information into its local data base with the flag 'not authenticated'.

Validation of Application (B2) *B* validates the manual signature and the date on the written application form, e.g. by cross-checking with a registered signature. Then *B* checks the authenticity of the electronic application (in particular, of *A*'s public verification key v_A). For that purpose the hash value of the electronically received data block DB_A is locally computed and compared with the hash value $h(DB_A)$ received on the printed application form. If those hash values agree, then all the components of the data block DB_A are authenticated (sender identifier *A*, key serial number *N*, public verification key v_A , validity period *Val* and date *DATE*). If all the checks are successful, then *B* may activate entity *A*'s public key information in its local data base as 'authenticated'.

Optional: Since the electronic and the written application can be separated in time by several days, it may be useful in certain application environments to respond to *A* with an acknowledgment

KT_{B2}

Depending on the application environment, this acknowledgment may be a signed electronic message, a facsimile message, a letter, etc. In such a way *A* knows from when on its digital signatures can be processed by *B*.

10.2.2 Properties

This Public Key Transport Mechanism has the following properties:

1. Standards: conformance with ISO/IEC CD 11770-3, *Public Key Transport Mechanism 2*, and with X.509 regarding the structure of the public key information.
2. Data field *A*: the sender identifier *A* must be able to accommodate the identification of a company/organisation and the identification of a person who is authorized to sign..
3. Data field *Date*: contains the current date (and possibly the time). The date is necessary, because it proves that the sender is in possession of the private signature key and does not replay the key token of another user.
4. Data fields *Data1*, *Data2* and *Data3*: for reasons of flexibility optional data fields are included in the definitions of the tokens. They can be filled with additional data elements based on the particular application environment of the mechanism. It is the task of the application-specific implementation guidelines to specify the use of the free data fields.
5. Processing step (B1): Since at this time the public key v_A is not yet authenticated, the verification of the digital signature simply is a functional test. The electronic application serves to minimise the manual handling of application data.
6. Instead of computing the hash value $h(DB_A)$ on the complete data block, one may alternately compute the hash value $h(v_A)$ on *A*'s public verification key only. But then the additional information contained in the application, such as validity period and key serial number must be verified using other methods, e.g. manual procedures.
7. The written and manually signed application form is only necessary when a new subscriber registers its public key or when an existing subscriber changes its public key.
8. The manually signed application also forms the contractual basis for the use of digital signatures between *A* and *B*. It authorises the service provider *B* to accept digital signatures from *A* and to process *A*'s messages accordingly.
9. A successful attack against this mechanism requires a successful forgery of the registered letter and the manual signature of *A*.
10. This mechanism can analogously be used for the registration of the public key e_A of an asymmetric encipherment system.
11. This mechanism is suitable for the transport and registration of public key data from a (telebanking) customer to a (telebanking) service provider.

11 Management infrastructure for public key certificates

This chapter contains a description of the procedures and the management infrastructure used for public key certification. It concentrates on the procedures (i.e. the interface) between the subscribers and the public key management infrastructure. The security within the public key management infrastructure is outside the scope of this document. However, it is assumed that secure links are established between the various entities within the public key management infrastructure.

11.1 Introduction

The use of digital signatures and other TBSS security mechanisms must be preceded by the mutual exchange of public keys between the communicating parties. Chapter 10 describes some pragmatic solutions between service providers and customers. These techniques are suitable for star topologies, but have limitations in open systems where every user may securely communicate with every other user.

In general it should be possible for any two users to set up a secure communication link quickly and efficiently without having to consult beforehand or to go through time-consuming administrative procedures.

The general solution to this problem is to provide a hierarchical key management infrastructure which handles the authenticated distribution of users' public keys. If a user needs another user's public key (e.g. when communication is established for the first time) he can go to the key management infrastructure and request the relevant public key, which is delivered to him electronically.

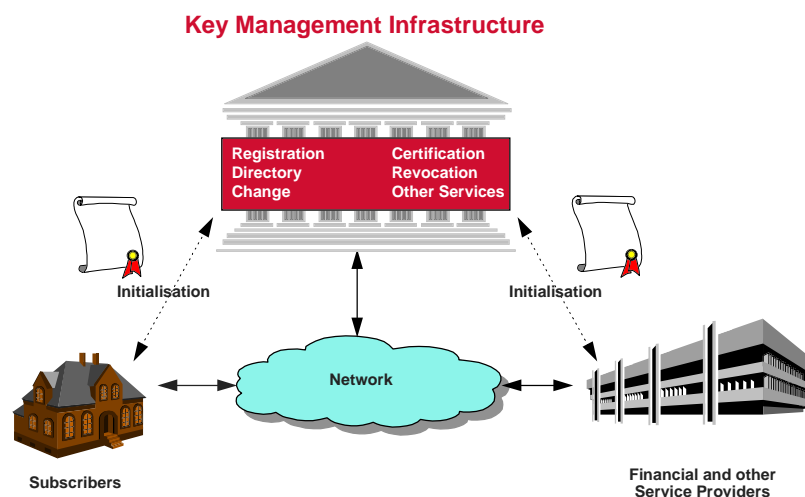


Figure 11-1 Distribution of public keys via a key management infrastructure

There are a number of general prerequisites for the implementation of a public key management infrastructure. Procedures have to be made available for secure:

1. Registration of subscribers.
2. Certification of public keys.
3. Certificate distribution and usage.
4. Subscriber initialization.
5. Certificate change management.
6. Revocation and suspension of certificates.
7. Possibly auxiliary services such as time stamping of electronic documents.

Certification is the dominant model for the secure distribution of public keys. The next figure shows the main steps involved.

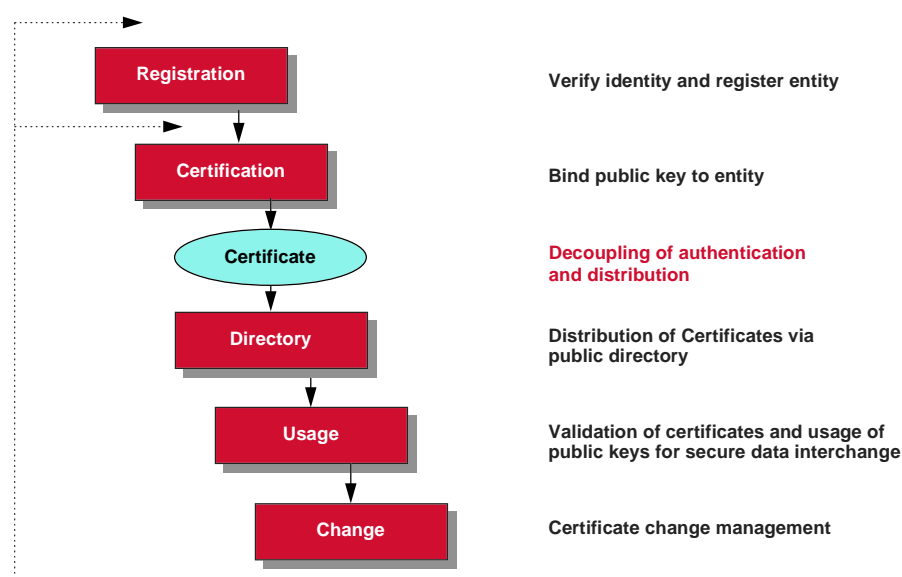


Figure 11-2 Certification: separating authentication from distribution

11.1.1 Registration

Secure registration involves the validation of the subscriber's identity and credentials and the assignment of a unique reference number. In addition, it may include the authenticated submission of the subscriber's public key.

Registration Authorities (RAs) must be appointed to look after the secure registration of subscribers. Each RA is responsible for a specific user area (RA domain). Procedures have to be provided for the secure transmission of registered user data for subsequent processing steps (such as certification).

11.1.2 Certification

It must be possible to unmistakably verify that public keys are authentic and genuinely belong to a specific user. Digital signatures once again provide a reliable method for the secure distribution of public keys. Just as users sign their messages to prove their authenticity, one of the authorities in the public key management infrastructure can sign the users' public keys to prevent them being falsified. The certification of public keys therefore involves the issuance

of a certificate by a Certification Authority (CA). This certificate not only contains the public key, but the user identity (confirming the user's ownership of the public key) and other important information such as the validity period of the certificate (e.g. 2 years).

To allow users to verify the certificates issued, they must have the Certification Authority's public verification key. An authenticated procedure must also be used to distribute the CA's key to all users.

Certification Authorities must be appointed to handle the secure certification of the users' public keys. Each CA is responsible for its own domain. If users from different domains are to communicate with each other, procedures have to be provided to allow the certificates to be exchanged and verified across different domains.

11.1.3 Certificate distribution and usage

In general public key certificates can be distributed without additional security measures. The best solution is to store the certificates in a public directory, from where any interested user can retrieve them. These certificates can also be exchanged directly between users. For example, a user can forward his certificate along with a signed message and the receiver first verifies the validity of the certificate using the CA's public key, then checks the validity of the message signature using the sender's authenticated public key.

11.1.4 Subscriber initialization

One of the most challenging tasks in any public key management infrastructure is the initialization of new subscribers. Subscriber initialization includes (not necessarily in that order): generating an asymmetric key pair; registering the new subscriber, possibly including the registration of a public key; certifying the subscriber's public key; distributing the public key certificate (possibly together with the private key) to the subscriber; making the new subscriber's public key certificate available to other persons using appropriate directory services. The responsibility for the key generation may be with the subscriber or with the public key management infrastructure.

11.1.5 Certificate change management

Any public key management infrastructure must address the problem of updating existing certificates or issuing new certificates based on existing certificate relationships. A certificate may have to be re-issued because of changes in the contents of the certificate (for instance, extension of the validity period of a certificate or changes in the credential information, such as the address of a subscriber). New certificates may be requested for an already registered subscriber, based on one or more existing valid certificates.

11.1.6 Revocation and suspension

Certificates are usually issued for a fixed period. However, circumstances may arise in which certificates may have to be revoked before the end of their validity period. This could be due to the loss or compromise of the relevant signature key, or because the user wants to change his key for some other reason. If this happens, the existing certificate has to be declared invalid by an authorized body (e.g. the CA) and a new one issued as a replacement (if necessary). Procedures are required for submitting and processing applications for revocation of certificates. Furthermore, users have to be informed about any certificates that have been revoked. In certain emergency situations it must be possible for a subscriber to quickly and directly make a request to suspend his certificate temporarily. After a suspension, a certificate may be reactivated or definitely revoked.

Since the certificates are not only stored in the controlled environment of the public key management infrastructure (e.g. they may be stored locally on users' systems), there is the basic problem that certificates that have been revoked may continue to be used inadvertently. It is impossible to see from the certificate itself that it has since been revoked. If a user receives a certificate and wants to be absolutely sure that it is valid, the only way is a secure acknowledgement from the public key management infrastructure at the time of the transaction.

11.1.7 Auxiliary services

A public key management infrastructure may include auxiliary services such as archiving or time stamping of electronic documents. Auxiliary services are not within the scope of this chapter.

11.1.7.1 Archiving

To ensure that signatures on electronic documents can still be verified in years to come, the relevant public keys and their certificates have to be archived. Every service provider can create an archive holding its customers' certificates. But for open electronic transactions to be possible, a general archive has to be created as well, where users' public key certificates are stored for a specific period (e.g. 10 years) for verification purposes at any time.

11.1.7.2 Time stamping

If a certificate is revoked because the integrity of the relevant signature key is compromised, the signatures performed with the key could be contested. If one wants to ensure that a signature remains binding even after the certificate is revoked, one needs to use an independent time stamping service to confirm that the signature was made before the certificate was revoked.

11.1.7.3 Attribute certification

It may be useful to certify other attributes of a subscriber (besides the public key). These attributes could be included in the basic public key certificate. However, typically they are kept in separate structures called attribute certificates. The attributes are not needed in every transaction, the attributes would increase the size of the basic certificate, and attributes may vary much more often than the basic certificate information, which would require a reissuance of the certificate at each change.

11.1.8 Overview

The next figure shows the logical functions which the public key management infrastructure generally has to provide.

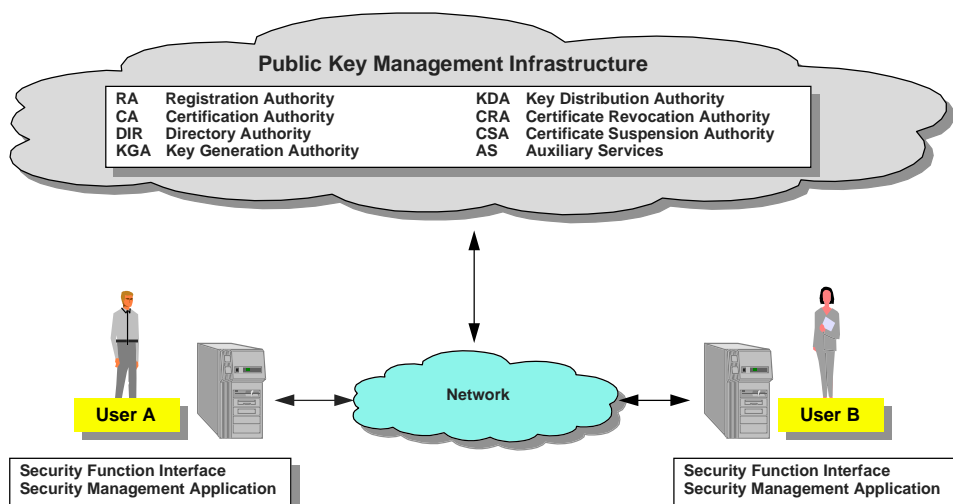


Figure 11-3 Functions of the public key management infrastructure

The subscribers need a

- Security Function Interface (SFI), to invoke security services in the context of an application, for instance to apply a digital signature to a message.
- Security Management Application (SMA), to manage the local security environment and to make contact with the public key management infrastructure, ie use the services provided by the infrastructure.

In concrete implementations of the public key management infrastructure, responsibility for these functions has to be assigned to particular departments or organizations. The organizational framework of these functions is beyond the scope of the TBSS as it concerns the actual implementation.

11.2 Subscriber model

There are various types of subscribers to be considered in a public key management infrastructure, ranging from private persons to application processes in a department of a service provider. Since *subscriber* is defined as a 'person who digitally signs a message' and since *person* is defined as 'a human being or any organization capable of signing a document, either legally or as a matter of fact' we must distinguish the following types of subscribers:

- organisational person
- residential person
- organisational unit
- application process

In our current legal system, a juridical person can only be legally bound by one or more natural persons holding the necessary authorization. The acceptance of the digital signature as a replacement of the hand-written signature is based on the assumption that the existing business practices concerning validity checking can be maintained (although realized electronically). The concept of organizational or process signatures has been brought to us through technological advancement. The acceptance of organizational or process signatures requires either an extension of today's commercial register procedures to allow for the registration of organizational signatures, or must be based on a contractual agreement.

Depending on the type of subscriber there may be different (human) persons involved in different roles. There may be one or more users of the key. The user is the natural person which has the right to activate the key, i.e. execute a digital signature. The owner of the key, that is, the person responsible for the key and its actions, is either the user, in case of a personal signature, or the organization, in case of an organizational signature. There may be a security officer. That is the person which is responsible for the management of the key (generation, distribution, backup etc.). There may be one or more authorizing managers from the executive body of the organization.. These are the persons which sign the contract that legally authorizes the usage of the key. The following figure shows the general model:

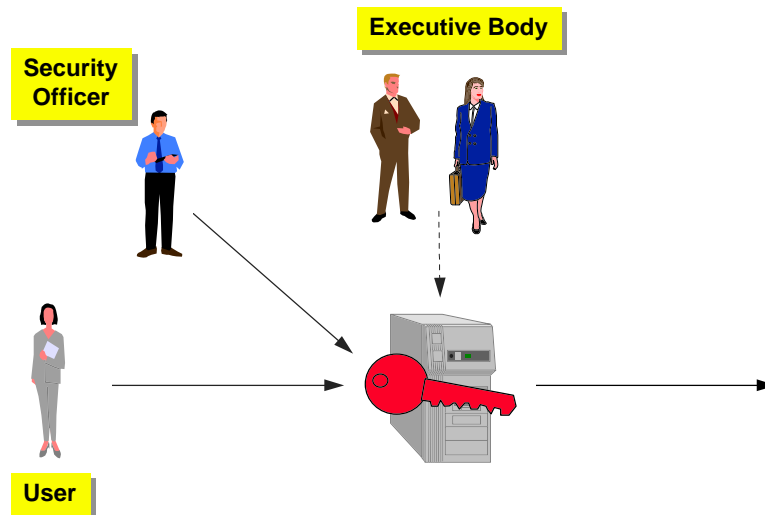


Figure 11-4 Subscriber model and associated roles

Residential persons and *organizational persons* have personal keys. With personal keys the user and the owner coincide. For subscribers of type *residential person* things are conceptually simple. There is one human person who owns, uses, manages, and authorizes the key. For subscribers of type *organizational person* the key is also personal. There is one human person who owns and uses the key. There may be a security officer who manages the key on behalf of the organizational person. There typically are one or more authorizing managers who have to grant the rights associated with the key, i.e. the right to sign certain transactions on behalf of the organization. Personal signature keys allow to simulate handwritten signatures, provided the legal framework is correspondingly set.

For subscribers of type *organizational unit* and *application process* the key is no longer associated to a human person, the key becomes a corporate key. With corporate keys, the actual users need not be registered. The organization is responsible for the actions of the key. One or more persons from the executive body of the organization have to authorize the rights associated to the signature key. There may be a security officer who manages the key. Regarding usage of the key, there are some variations. The key may be used with an automated process, i.e. a digital signature is executed whenever certain system conditions are met (e.g. an EDI server). On the other hand, there may be several human persons who are allowed to activate the corporate key (e.g. a financial department where several controllers can check payment transactions). In both cases, the organization has to define an appropriate key usage policy. Access control, audit procedures, liability and risk management issues have to be addressed. As discussed above, the acceptance of organizational or process signatures requires either an extension of today's commercial register procedures or must be based on a contractual agreement

11.3 Subscriber initialization models

One of the most challenging tasks in any public key management infrastructure is the initialization of new subscribers. Subscriber initialization comprises (not necessarily in that order):

- generating an asymmetric key pair
- registering the new subscriber, possibly including the registration of a public key
- certifying the subscriber's public key
- distributing the public key certificate (possibly together with the private key) to the subscriber
- making the new subscriber's public key certificate available to other persons using appropriate directory services.

Depending on who has responsibility for the generation of the subscriber’s asymmetric key pair, two basic models for the subscriber initialization can be distinguished. In model 1, the subscriber is responsible for the key generation. In model 2, the public key management infrastructure is responsible for the key generation.

11.3.1 Model 1

In Model 1 the subscriber is responsible for the key generation. That is, the subscriber (or a trusted third party on behalf of the subscriber) generates the asymmetric key pair before registration.

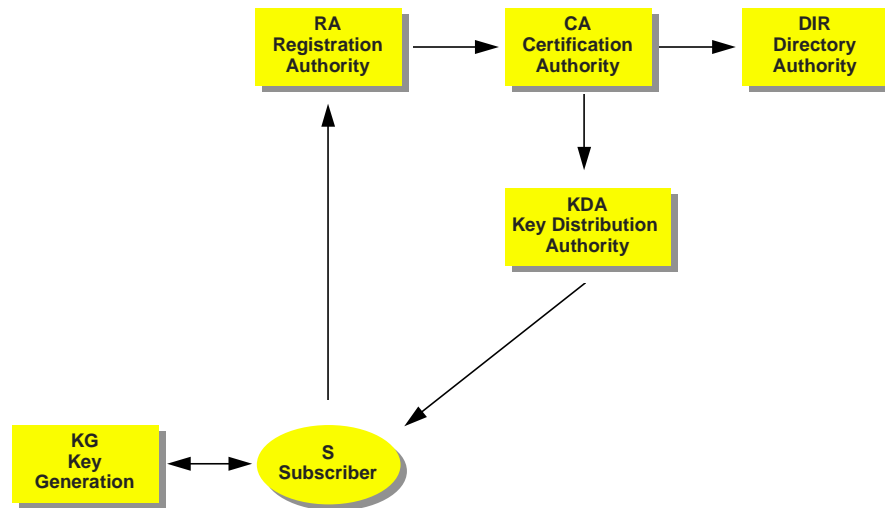


Figure 11-5 Subscriber initialization with key generation before registration.

The general procedure is as follows:

| Nr | Step | Parties | Comments |
|----|--------------------------|---------------------|---|
| 1. | Key generation | S ↔ KG | The subscriber generates an asymmetric key pair himself or acquires an asymmetric key pair using a key generation service. |
| 2. | Registration | S → RA | The subscriber and his public key are registered at the registration authority. The RA has to verify the subscriber’s credentials and assign an unambiguous name. |
| 3. | Certification | RA → CA CA | The registration authority forwards the subscriber credentials and his public key to the Certification Authority. The Certification Authority generates the subscriber certificate. |
| 4. | Certificate distribution | CA → KDA KDA → S | The CA forwards the certificate to the key distribution authority. The key distribution authority distributes the certificate to the legitimate subscriber. |
| 5. | Directory | CA → DIR | The Certification Authority places the certificate in the directory. This requires a secure access to the directory. |

11.3.2 Model 2

In Model 2 the public key management infrastructure is responsible for the key generation during the initialization cycle. That is, the subscriber registers his credentials and requests key generation by the public key management infrastructure.

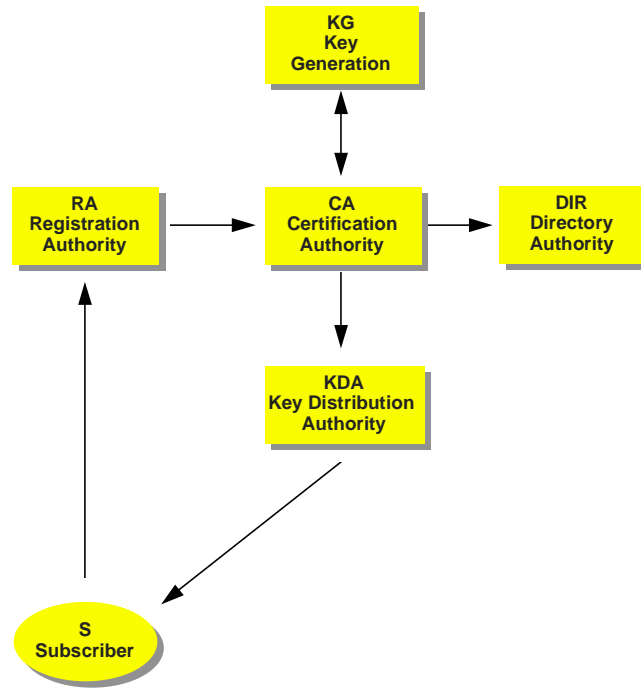


Figure 11-6 Subscriber initialization with key generation at certification.

The general procedure is as follows:

| Nr | Step | Parties | Comments |
|----|------------------|-------------------|---|
| 1. | Registration | S → RA RA → CA | The subscriber is registered at the registration authority. The RA has to verify the subscriber’s credentials and assign an unambiguous name. The registration authority forwards the subscriber credentials to the Certification Authority with a key generation request. |
| 2. | Key generation | CA ↔ KG | The Certification Authority acquires an asymmetric key pair using a key generation authority. |
| 3. | Certification | CA CA → KDA | The Certification Authority generates the subscriber certificate. The Certification Authority forwards the subscriber private key and his certificate to the key distribution authority. |
| 4. | Key distribution | KDA → S | The key distribution authority forwards the subscriber private key and his certificate to the legitimate subscriber. This distribution requires a confidential and authenticated channel. |
| 5. | Directory | CA → DIR | The Certification Authority places the certificate in the directory. This requires a secure access to the directory. |

11.4 Registration

Subscriber registration involves the submission of the subscriber credentials and their validation by the Registration Authority. The following figure shows the general steps involved in the registration of subscribers. In each of the following subsections these steps will be specifically addressed regarding the type of subscriber to be registered.

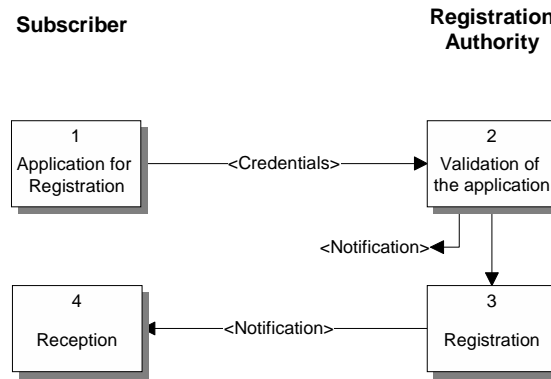


Figure 11-7 General registration procedure

11.4.1 Registration of residential persons

1. Application

For residential persons, the application must include full name, residential address, nationality, date of application, and a handwritten signature by the person. Optionally, the application may contain the telephone number, the facsimile number, the e-mail address, the desired registration validity period, the authorization for publication, and an existing reference registration identifier such as the social security number or ID card number.

In addition, residential persons must prove their identity using a legitimate id document. This can consist of a valid ID card, passport, driver's licence, birth certificate, social security card, certified signature, etc.

For residential person companies, the same rules apply. In addition, documents proving the identity of the company can be required.

2. Validation

The validation of the application is based on proper identification of the residential person applying. The Registration Authority must validate the credentials supplied by the residential person. The registration officer checks that all the mandatory and all the optional information is supplied correctly on the application form. Then, the registration officer must validate the identity of the applying residential person, based on the identification document supplied. Depending on the application and the liabilities incurred there will be different assurance levels regarding the quality of identity validation. The Registration Authority must define a policy and corresponding procedures. For instance, the Registration Authority must define in which cases appearance in person is required, whether an authorized agent may apply on behalf of the subscriber, and if an independent identification by a trusted third party is to be accepted at the Registration Authority.

In case the validation fails, the Registration Authority must notify the applying subscriber with an indication of the problem encountered.

3. Registration

Based on the chosen registration scheme, the Registration Authority assigns a registration identifier to the applying residential person. Other registration specific data, such as the date of registration, the name of the RA, the authentication method used or the credentials supplied, and the validity period, are added. A notification of acceptance is generated containing all the registration-relevant information. The notification is signed and sent / handed over to the applying residential person. The id document presented to confirm the application is returned as well to the residential person.

The Registration Authority archives the application, a copy of the proof of identity and a copy of the notification of acceptance for an appropriate time period.

4. Reception

Receipt of the notification of acceptance completes the registration cycle. The subscriber files it securely.

11.4.2 Registration of organizations

1. Application

For organizations, the application must include the name of the organization, postal address, name of the person making the application, organizational title of the person, postal address of the person, date of application, and one or more handwritten signatures by persons authorized to sign on behalf of the organization. For organizations, which are registered in the commercial register, the application must contain the corresponding commercial register number. Optionally, the application may contain the trade name of the organization, the telephone number, facsimile number, and e-mail address of the person making the application, the desired registration validity period, the authorization for publication, a reference to an interchange agreement, and a reference to the credentials supplied.

Organizations, which are registered in the commercial register, may supply a certified transcript of the commercial register entry together with the application. This depends on the procedures chosen, as the Registration Authority may itself wish to consult the commercial register in order to validate the application. Organizations, which are not registered in the commercial register, must supply appropriate identification information such as certificate of foundation, signed statutes, signed minutes from the general assembly, board of directors, management board, the organization's signatory rights, the organization's auditor, the organization's attorney.

The application must be accompanied by a letter with official organization letterhead and authorized signatures.

2. Validation

The validation of the application is based on proper identification of the organization and of the representatives of the organization applying. Depending on the application and the liabilities incurred there will be different assurance levels regarding the quality of identification of organizations. The Registration Authority must define a policy and corresponding procedures for the registration of organizations. The Registration Authority checks that all the mandatory and all the optional information is supplied correctly on the application form. Then the RA establishes the identity of the organization and its signatories, (a) using the commercial register transcript or by consulting the commercial register directly, (b) using the supplied identification information for organizations not registered in the commercial register (as described above, this may involve consultation of a trusted third party). Then, using reasonable commercial practices, the RA verifies (a) that the official letterhead and signature(s) are authentic, (b) that the application and signature(s) are authentic, (c) the identity of the representative(s) or the authorized agent of the organization.

In case the validation fails, the Registration Authority must notify the applying subscriber with an indication of the problem encountered.

3. Registration

Based on the chosen registration scheme, the Registration Authority assigns a registration identifier to the applying organization. Other registration specific data, such as the date of registration, the name of the RA, the authentication method used or the credentials supplied, and the validity period, are added. If organizational persons and/or application processes are to be registered subsequently, a block of registration numbers is

assigned to the organization. This is typically done by requiring that the organization’s identifier be the unique prefix within the final registration number. A notification of acceptance is generated containing all the registration-relevant information. The notification is signed and sent / handed over to the representative or authorized agent of the applying organization. To enhance security, a copy of the notification may be sent directly to the organization’s executives who signed the application.

The Registration Authority archives the application, a copy of the proof of identity and and a copy of the notification of acceptance for an appropriate time period.

4. Reception

Receipt of the notification of acceptance completes the registration cycle. The notification has to be filed securely.

11.4.3 Registration of organizational persons

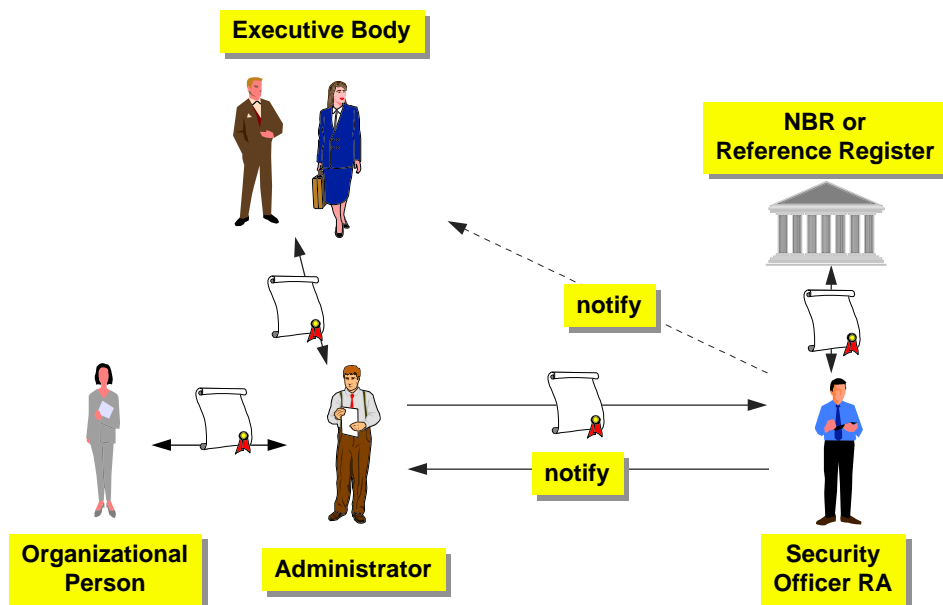


Figure 11-8 Registration of organizational persons

A prerequisite for the registration of organizational persons is that the organization itself has a registered organization identifier.

1. Application

For organizational persons, the application must include the name of the organization, the registered organization identifier, the list of organizational persons to be registered, and the name, organizational title and postal address of the person making the application (this could be an administrator). The application must include the date of application, and one or more handwritten signatures by persons authorized to sign on behalf

of the organization (i.e. executives of the organization). Optionally, for each organizational person the application may contain information such as desired registration number, desired registration validity period, telephone number, facsimile number, e-mail address. In addition, it may be required that every organizational person to be registered supplies a handwritten signature him/herself. Optionally, the application may contain the telephone number, facsimile number, and e-mail address of the person making the application (ie security officer), the authorization for publication, a reference to an interchange agreement.

The application must be accompanied by a letter with official organization letterhead and authorized signatures. Depending on the RA's policy, it may be required to supply identity information such as a certified transcript of the commercial register each time an organizational person is registered.

2. Validation

The validation of the application is based on proper identification of the organization and its signatories. Appearance in person is not required, since the organization is already registered. The Registration Authority checks that all the mandatory and all the optional information is supplied correctly on the application form. Then the RA establishes the identity of the organization and its signatories, using the information supplied during the registration of the organization. Given that some time has passed since initial registration of the organization, reasonable business practice may demand that the identification information of the organization is verified again (eg that the commercial register is consulted again). Then, the RA verifies (a) that the official letterhead and signature(s) are authentic, and (b) that the application and signature(s) are authentic. The RA need not verify the signatures of the organizational persons to be registered.

In case the validation fails, the Registration Authority must notify the applying organization with an indication of the problem encountered. To enhance security, a copy of the notification may be sent directly to the organization's executives who signed the application.

3. Registration

Based on the chosen registration scheme, the Registration Authority confirms the desired or assigns a new registration identifier to each of the applying organizational persons. Other registration specific data, such as the date of registration, the name of the RA, the authentication method used or the credentials supplied, and the validity period, are added. A notification of acceptance is generated containing all the registration-relevant information. The notification is signed and sent to the person making the application (ie security officer). To enhance security, a copy of the notification may be sent directly to the organization's executives who signed the application.

The Registration Authority archives the application, a copy of the proof of identity and a copy of the notification of acceptance for an appropriate time period.

4. Reception

Receipt of the notification of acceptance completes the registration cycle. The notification has to be filed securely.

In systems, where digital signatures have been established as an accepted means to provide non-repudiation, some or all of the messages involved in this registration cycle may be implemented using electronic means.

11.4.4 Registration of organizational units and application processes

The registration of organizational units and application processes is similar to the registration of organizational persons. A prerequisite for the registration of organizational units and application processes is that the organization itself has a registered organization identifier.

1. Application

For organizational units and application processes, the application must include the name of the organization, the registered organization identifier, the list of organizational units / application processes to be registered, and the name, organizational title and postal address of the person making the (this could be an administrator). The application must include the date of application, and one or more handwritten signatures by persons authorized to sign on behalf of the organization (i.e. executives of the organization). Optionally, for each organizational unit / application process the application may contain information such as desired registration identifier and desired registration validity period. Optionally, the application may contain the telephone number, facsimile number, and e-mail address of the person making the application (ie security officer), the authorization for publication, a reference to an interchange agreement.

The application must be accompanied by a letter with official organization letterhead and authorized signatures. Depending on the RA's policy, it may be required to supply identity information such as a certified transcript of the commercial register each time an organizational unit / application process is registered.

2. Validation

This is identical to the validation of the application for organizational persons.

3. Registration

This is identical to the registration of organizational persons.

4. Reception

Receipt of the notification of acceptance completes the registration cycle. The notification has to be filed securely.

In systems, where digital signatures have been established as an accepted means to provide non-repudiation, some or all of the messages involved in this registration cycle may be implemented using electronic means.

11.4.5 Registration through security administrator

In the case of organizational persons and application processes the registration procedures can be simplified. If the organizations's security administrator is already registered and certified by the public key management infrastructure and if he is correspondingly authorized, he can act as a local registration authority within the organization. That is, the security administrator validates the credentials of the intra-organizational registration requests directly. No paper forms are required to be sent between the organization and the public key management infrastructure. Instead, the security administrator forwards the validated registration and certification requests digitally signed with his private signature key to the Registration Authority. The Registration Authority then processes the requests electronically based on the already established digital signature of the security administrator. Additional digital signatures (such as an authorizing signature by an executive of the organization) may be required, depending on the RA's policy.

This approach may be extended to the general case where subscribers are registered electronically through a trusted agent.

11.4.6 Registration combined with public key submission

In the case where subscribers generate their keys themselves, they need an authenticated channel to the Registration Authority or Certification Authority to submit the public key. The submission requires a proof that the subscriber is in possession of the corresponding private key and that the public key was not inserted by a malicious third party.

11.4.6.1 First-time registration of subscriber public key

The recommended implementation is TBSS Public Key Transport Mechanism 2. The subscriber sends his/her public key together with the registration identifier and a time stamp in an electronic message to the Registration Authority or Certification Authority. The message must be signed with the corresponding private key. In addition, the subscriber sends a key authentication form which contains the registration identifier and a hash value of the public key. The form must be manually signed by the person who generated the key. In case of organizational persons and application processes, the key authentication form must also be authorized by one or more persons with signatory rights (ie executives of the organization). Who has to sign the key authentication form is determined during the registration procedure. The Registration Authority validates the signatures on the key authentication form and then compares the hash value printed on the form with the hash value computed locally from the received public key.

11.4.6.2 Registration of secondary public key

If the subscriber already has an operational key pair, then the submission of a secondary public key can be simplified. The key authentication form can be replaced by a second digital signature applied to the electronic submission message. The signature is based on the operational key pair and thereby establishes authenticity and origin of the additional public key to be registered (compare also section 11.10.2 on certificate change management).

11.4.6.3 Public key registration through security administrator

In the case of organizational persons and application processes, the organizations's security administrator can forward the validated registration and certification requests to the Registration Authority, together with the corresponding public keys and digitally signed with his private signature key. The Registration Authority then processes the requests electronically based on the already established digital signature of the security administrator. Additional digital signatures (such as an authorizing signature by an executive of the organization) may be required, depending on the RA's policy.

11.5 Key generation

Key generation may take place at the subscriber or at the central key management infrastructure. Depending on the choice, different trust models and initialization procedures will result.

11.5.1 Key generation by subscriber

Key generation at the subscriber raises the following requirements:

1. The subscriber needs a trusted security software to generate the asymmetric key pairs. This security software has to be installed prior to the start of the registration and certification procedure. The central public key management infrastructure need not be trusted with the management of the subscriber private keys.
2. The subscriber needs an authenticated channel to the Registration Authority or Certification Authority to submit the public key. The submission requires a proof that the subscriber is in possession of the corresponding private key and that the public key was not inserted by a malicious third party.

For liability reasons, it is preferable that signature keys are generated by the subscriber. Key generation at the subscriber requires a two-step installation process. First, the security software which performs the key generation has to be installed. Then, after central registration and certification of the public key, the private key can be activated.

At the subscriber, it need not be the same person which generates the keys and which uses the keys. At an organization or organizational unit, there may be a security officer which is in charge of the key management on behalf of the organizational persons or application processes.

11.5.2 Key generation by central authority

If the CA or some other central component of the key management infrastructure generates the key pairs on behalf of the subscribers, the following requirements result:

1. The subscribers must trust the central authority to perform the key generation correctly, to handle their private keys securely, and to not misuse the knowledge of their private keys.
2. The central key management authority needs a secure channel to deliver the key pair to the subscriber. In particular, the delivery of the subscriber's private key requires confidentiality.

To protect against loss of data, it is preferable that encryption keys are generated centrally. This implies that a copy of the decryption key is held in a secure backup. If signature keys are generated centrally, then any copy of the private keys must be destroyed immediately after personalization, in order to ensure non-repudiation properties.

Central key generation allows that subscribers receive their security software and their keys at the same time and perform a one-step installation procedure to become operational. However, later key changes require local activation and deactivation procedures.

11.5.3 Key escrow

Key escrow is the process whereby a copy of the subscriber private key (or shares of the private key) is registered and stored with one or more trusted agents of the public key management infrastructure. Key escrow can occur independent of whether the key generation is performed by the subscriber or by a central authority.

If keys are generated by the subscribers and key escrow is required, then the private key must be separated into shares at the subscriber, the shares must be delivered and registered with trusted agents, and the delivery has to occur in confidentiality.

If keys are generated centrally and key escrow is required, then the central authority must be trusted that it performs the separation of the private key into shares correctly, that the shares are distributed and stored securely (e.g. confidentially), and that the knowledge of the private keys or its shares is not misused.

11.6 Certification

Certification, that is, issuing public key certificates for subscribers is at the heart of a public key management infrastructure. However, certification is typically one step in a larger procedure such as initializing a subscriber or changing a certificate.

11.6.1 The Certification Authority

The duties of a Certification Authority are:

1. A Certification Authority must have sufficient resources (a) to maintain its operations in conformity with its duties, and (b) to be reasonably able to bear its risk of liability to subscribers and persons relying on certificates issued by the Certification Authority.
2. A Certification Authority must employ personnel practices which provide reasonable assurance of trustworthiness.
3. If a Certification Authority holds the private key corresponding to a public key named in a certificate which it has issued, it holds it as a fiduciary of the subscriber named in the certificate, and may use it only with the prior, written consent of the subscriber.
4. A Certification Authority must keep records supporting the certificates it issues for a reasonable length of time.
5. A Certification Authority must make its certificate/public key readily and reliably available to persons whose need to rely on it is reasonably foreseeable.

6. A Certification Authority must utilize trustworthy systems in performing its services.

11.6.2 Maintaining integrity of the CA public key

In order to verify the certificates issued by a Certification Authority, one needs the public key of that CA. That CA public key may again be sealed in a certificate issued by another Certification Authority (depending on the certification hierarchy and domain model). But eventually, there will be one CA public key which has to be distributed in a non-certificate form to all the involved parties. This distribution has to be achieved in an authenticated and reliable fashion.

The following methods are commonly used to distribute the CA public key in an authenticated fashion:

1. Delivery (e.g. on a diskette) using registered mail or a courier.
2. Joint delivery of the CA public key together with the installation software.
3. Direct delivery using an unprotected medium (such as e-mail) and separate authentication, e.g. by sending the hash-value of the public key by mail.

Once a subscriber is in possession of the CA public key, precautions have to be taken to maintain the authenticity of the locally stored copy of the CA public key. The following practice is recommended from the subscriber to the CA:

1. Signing the CA public key with the subscriber private key each time a subscriber public key is registered.
2. Inclusion of a check value of the CA public key on the key authentication form, each time a subscriber public key is registered.

Doing so allows the CA to verify if the subscriber is still in possession of the authentic copy of the CA public key. The following practice is recommended from the CA to the subscribers:

1. Publication of the CA public key in a suitable and authenticated medium (e.g. a newspaper, WWW server) for easy reference and access.
2. Inclusion of the CA public key or its check value in direct communication with subscribers.

Doing so allows the subscribers to verify if they are still in possession of the authentic copy of the CA public key. The following practice is recommended at the subscribers:

1. Storing the CA public key in such a way that modifications can be detected.
2. Keeping a copy of the initially received CA public key in a secure backup for reference.
3. Periodic verification of the local copy of the CA public key based on the published version or received check values.
4. Keeping track of the CA public key validity period.

11.7 Certificate implementation

Currently, there are two global standards which address the format of a public key certificate: X.509 and UN/EDIFACT. The table in the Annex shows the combined functionality of the two standards. Different applications and environments will demand that different formats and different versions of certificates are supported. Each application must choose the certificate standard suitable and must provide an implementation guideline for its certificate. Beside the basic public key certificate, there are other certificates which need to be addressed by a specific application, such as attribute certificates, transaction certificates, service certificates, etc.

11.7.1 Minimal requirements

This section contains the minimal requirements for any TBSS public key certificate implementation. The following data items must be present across all TBSS public key certificate implementations:

11.7.1.1 Version number

Any TBSS certificate implementation must contain a version number. In case of a recognized certificate standard, the version number is determined by the version of the standard. In case of a nationally defined certificate standard, it is recommended to add a reference to the standard as well in this field.

11.7.1.2 Certificate reference number

The certificate must contain a reference number which is unique within the certificate issuer's domain. It must be possible to provide a globally unique reference to the certificate by specifying the issuer's identification / name and the certificate reference. A certificate issuer must ensure that no two distinct certificates within its domain contain the same reference number. Often the certificate reference is coded in such a way that it allows to deduce information about the country, the version, the application, the certificate issuer, the registration authority, or the certificate owner. It is recommended to structure the certificate reference in such a way that the following information can be deduced:

- Country identification
- Certificate issuer identification
- Certificate sequence number
- Representation identification (i.e. a code identifying the different representations of the same certificate information and public key, e.g. EDIFACT, X.509, etc.).

11.7.1.3 Certificate issuer identification

The certificate must contain an identification of the certificate issuer. As a minimal requirement there must be the logical name, ie the registration identifier of the certificate issuer present.

11.7.1.4 Certificate issuer algorithms

The certificate must contain an identification of the signature and the hash algorithm used by the certificate issuer.

11.7.1.5 Certificate issuer public key identification

The certificate must support key life cycle management for the certificate issuer. There may be different keys in different lifecycle states and it must be possible to separate them by name or reference. Therefore, the certificate must contain an identification of the public key associated to the signature key used by the certificate issuer to sign the certificate.

11.7.1.6 Certificate owner identification

The certificate must contain an identification of the certificate owner. As a minimal requirement there must be the logical name, ie the registration identifier of the certificate owner present. However, it is recommended to include the natural name of the owner.

11.7.1.7 Validity period

The certificate must contain a validity period, consisting of a start and end date. In addition, it is useful to have a date of issue (generation) or last update field. Any certificate date and time shall have the following format:

Date YYYYMMDD

Time HHMMSS

11.7.1.8 Certificate owner - public key information

The certificate must contain the public key (possibly consisting of more than one component) and the identification of the associated public key algorithm.

11.7.2 Attribute certification

It may be useful to certify other attributes of a subscriber (besides the public key). There are attributes which directly relate to the public key certificate, such as

- reference to the CA's policy which supports the certificate,
- assurance level obtained in the identification of the subscriber,
- restrictions in usage of the corresponding secret key
- alternative names for the certificate owner,
- information on trusted third parties involved in the registration,
- information on domains and cross-certification.

There are attributes which relate to the signing capabilities of an entity, such as

- liability limit of the certificate owner
- authorized signatory (formal authorization of individuals to sign for the organisation),
- transaction limit (maximum monetary value of a message which the entity may sign),
- transaction type (transaction types which the entity may sign),
- location (from which signatures of the entity are considered valid),
- pre-approved counter party (to indicate with which parties the entity can conduct signed transactions),
- delegation control (indicate the amount of authority that the entity may delegate to some other entity).

These attributes could be included in the basic public key certificate or, they could be kept in separate structures called attribute certificates. Many of the attributes are not needed in every transaction. If they are included in the basic certificate, they increase the size. Also, attributes may vary much more often than the basic certificate information, which would require a re-issuance of the certificate at each change.

11.8 Subscriber initialization mechanisms

This section contains two generic mechanisms which provide subscriber initialization, one for each of the basic models described in section 11.3. Depending on the type of subscriber the registration part will be different. A residential person can directly apply for a registration identifier within the initialization mechanism. But an organization has to obtain a unique <registered organization identifier> before it can initialize organizational persons, units or application processes with the infrastructure. It is also assumed that subscribers can send and receive messages and that the components of the public key management infrastructure (such as Registration Authority, Key Distribution Authority, Certification Authority and Directory Authority) are able to communicate securely (for instance, using digital signatures). Note that in an actual implementation, the different functions of the public key management infrastructure, as shown in the diagrams, can be combined.

11.8.1 Subscriber initialization mechanism 1

In model 1 the subscriber is responsible for the key generation. That is, the subscriber (or a trusted third party on behalf of the subscriber) generates the asymmetric key pair before registration. Therefore, the subscriber must have installed the necessary soft- and hardware to generate asymmetric key pairs and to register the credentials

electronically. It is also assumed that the subscriber is in possession of an authenticated copy of the Certification Authority's public certification key and the Key Distribution Authority's public signature validation key.

In the following process diagram the parties involved in the subscriber initialization are shown as headings across the first line. Time evolves from top to bottom. Each party is involved in the process steps in its own column (ie the processes underneath the heading).

Note: The exact content and form of the messages between the Subscriber and the Public Key Management Infrastructure are to be defined in the implementation guidelines or the respective application.

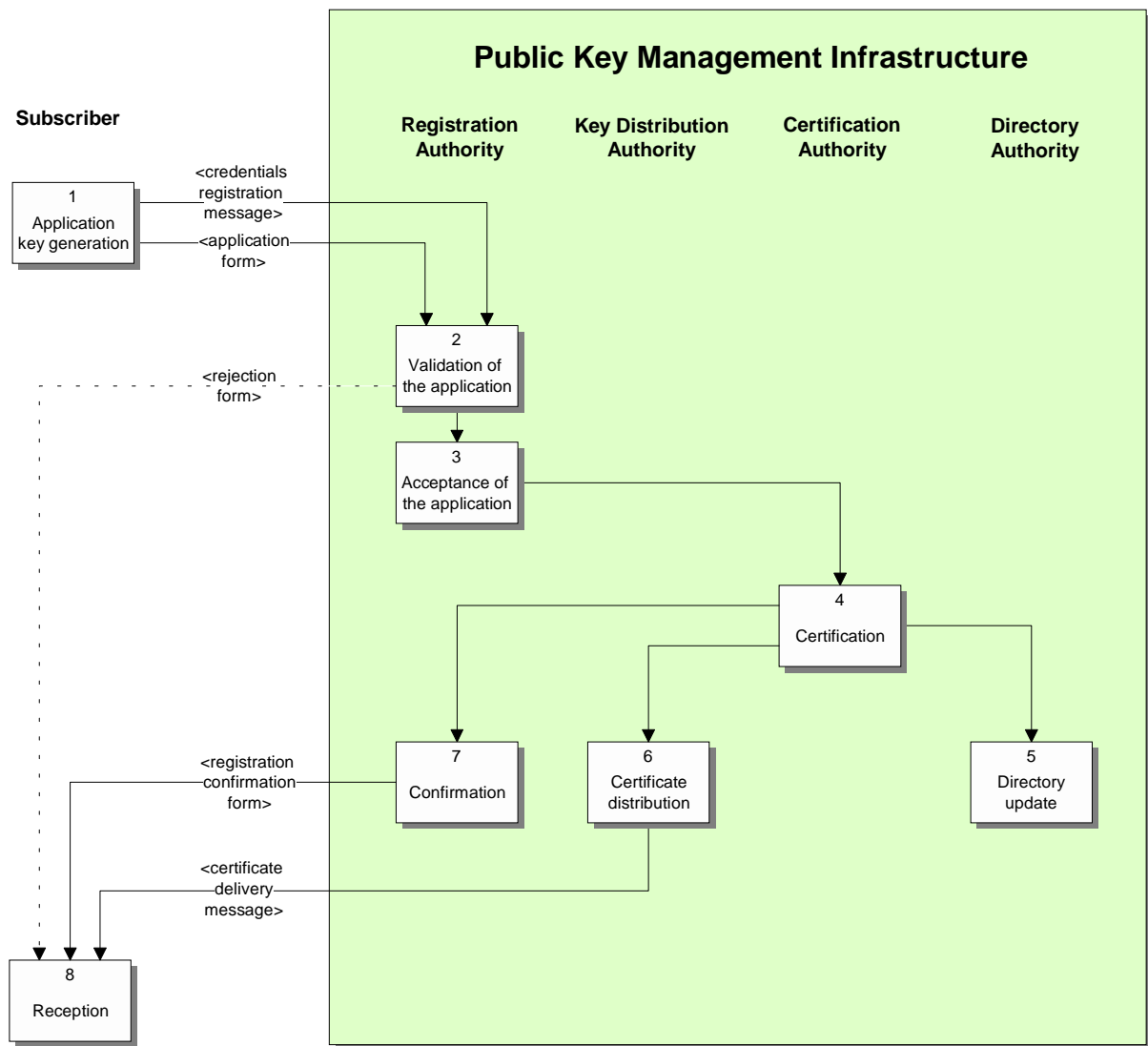


Figure 11-9 Subscriber Initialization Mechanism 1

1 Application and key generation

- 1.1 The subscriber generates an asymmetric key pair or acquires an asymmetric key pair using a key generation service.
- 1.2 The subscriber's credentials, together with the public key, are placed in a <credentials registration message>. The <credentials registration message> is timestamped and signed with the corresponding private key.

- 1.3 The <application form> is printed and manually signed by an authorized person. The form contains a hash value of the public key and, possibly, the registration identifier (this depends on the type of subscriber).
- 1.4 Both the <credentials registration message> and the <application form> are sent to the Registration Authority.

2 Validation of the application

- 2.1 The Registration Authority checks that all the mandatory and all the optional information is supplied on the <application form> in the correct syntax. For organizational applications, the Registration Authority checks that the selected identifier is consistent and unique.
- 2.2 The signature(s) and signatory rights of the subscriber are validated.
- 2.3 The digital signature of the <credentials registration message> is validated. The information contained in the <credentials registration message> is checked for timeliness, consistency and correct syntax.
- 2.4 The authenticity of the public key is verified by comparing the locally computed hash value with the one supplied on the <application form>.
- 2.5 If any of the results of the checks are negative, the Registration Authority completes a <rejection form> specifying the reason for rejection and sends it to the subscriber. The subscriber initialization mechanism is suspended.

3 Acceptance of the application

- 3.1 The Registration Authority assigns or accepts the registration number.
- 3.2 The Registration Authority completes the register entry in the registration database.
- 3.3 A certification request, containing the subscriber's credentials, together with the subscriber's public key and other registration relevant information, is submitted in a secure way to the Certification Authority.

4 Certification

- 4.1 The Certification Authority validates the certification request. If any of the results of the checks are negative, the Certification Authority suspends the subscriber initialization mechanism and resolves the issue together with the Registration Authority.
- 4.2 The subscriber's credentials, together with the subscriber's public key, the registration relevant information and certification relevant information are placed in a public key certificate. The certificate is signed with the Certification Authority's private certification key.
- 4.3 The Certification Authority completes the register entry in the certification database.
- 4.4 The subscriber's certificate is sent securely
 - a) to the Directory Authority for inclusion in the Directory.
 - b) to the Key Distribution Authority for distribution to the subscriber.
- 4.5 The Certification Authority notifies the Registration Authority that the subscriber's certificate has been generated.

5 Directory update

- 5.1 Upon reception of the directory update request from the Certification Authority, the Directory Authority places the subscriber's certificate in the Directory.

- 5.2 Alternatively, if the Certification Authority has the capability of a secure online write access to the Directory, the certification officer can directly store the certificate in the Directory.

6 Certificate distribution

- 6.1 Upon reception of the certificate distribution request from the Certification Authority, the subscriber's certificate is placed in a <certificate delivery message>. The <certificate delivery message> is timestamped and signed with the Key Distribution Authority's private transaction key and sent to the subscriber.

7 Confirmation of the registration

- 7.1 Upon reception of the Certification Authority's notification, the registration officer prints a <registration confirmation form>, signs it and sends it to the subscriber.

8 Reception

- 8.1 Upon reception of a <rejection form> from the Registration Authority, the subscriber reconsiders his application and resolves the issue.
- 8.2 Upon reception of the <registration confirmation form> from the Registration Authority, the subscriber archives it properly.
- 8.3 The subscriber validates the Key Distribution Authority's digital signature of the <certificate delivery message>. The information contained in the <certificate delivery message> is checked for timeliness, consistency and correct syntax.
- 8.4 The public key certificate is validated using the Certification Authority's public certification key.
- 8.5 If any of the results of the checks are negative, the subscriber contacts the RA to resolve the issue.
- 8.6 The certificate is made available at the subscriber. This may involve placement in a local repository.

11.8.2 Subscriber Initialization Mechanism 2

In Model 2 the public key management infrastructure is responsible for the key generation during the initialization cycle. That is, the subscriber registers his credentials and requests key generation by the public key management infrastructure.

In the following process diagram the parties involved in the subscriber initialization are shown as headings across the first line. Time evolves from top to bottom. Each party is involved in the process steps in its own column (ie the processes underneath the heading).

Note: The exact content and form of the messages between the Subscriber and the Public Key Management Infrastructure are to be defined in the implementation guidelines or the respective application.

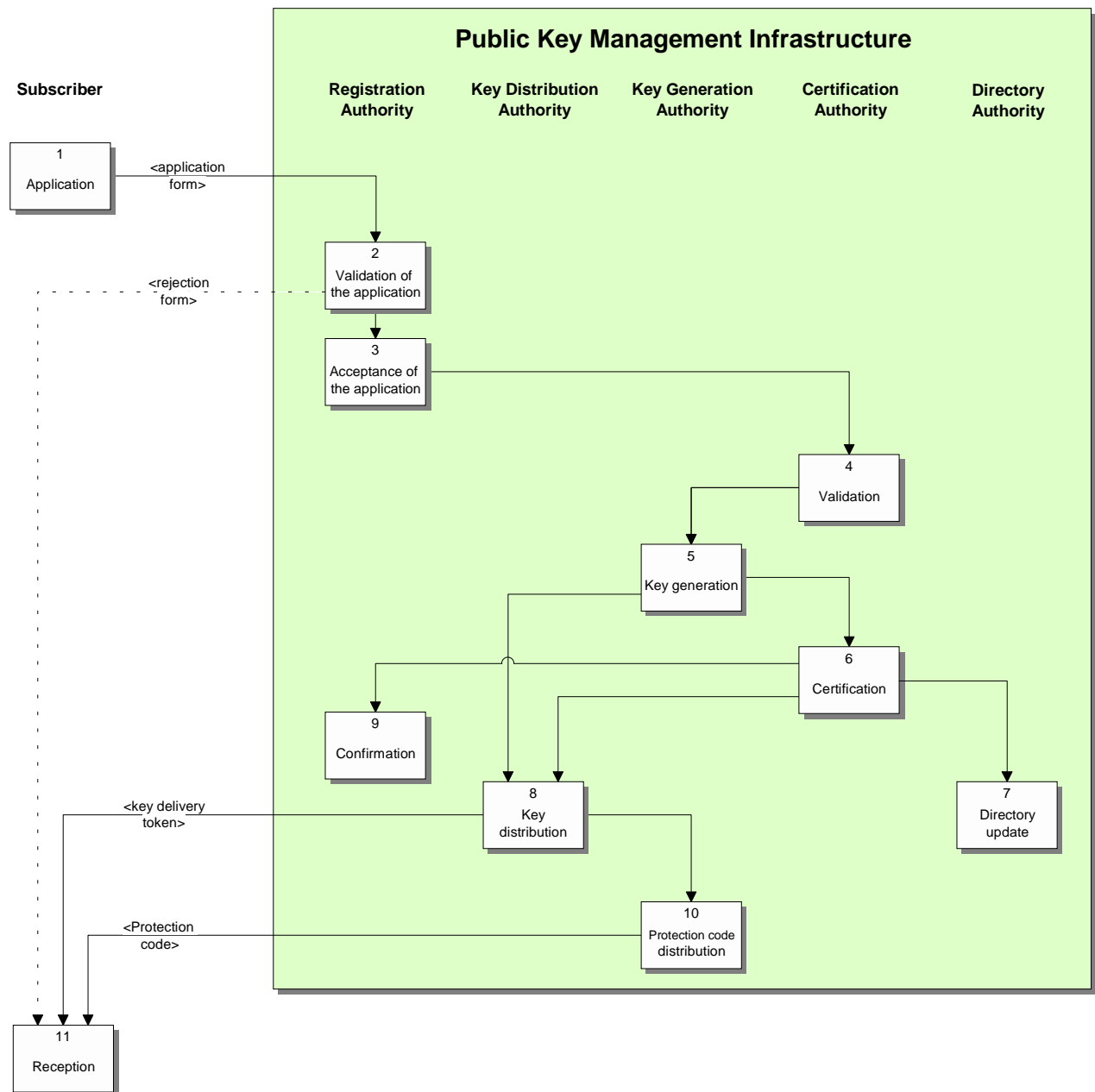


Figure 11-10 Subscriber Initialization Mechanism 2

1 Application

1.1 The <application form>, possibly containing the registration number, is manually signed by an authorized person and sent to the Registration Authority.

2 Validation of the application

2.1 The Registration Authority checks that all the mandatory and all the optional information is supplied on the <application form> in the correct syntax. For organizational applications, the Registration Authority checks that the selected identifier is consistent and unique.

2.2 The signature(s) and signatory rights of the subscriber are validated.

- 2.3 If any of the results of the checks are negative, the Registration Authority completes a <rejection form> specifying the reason for rejection and sends it to the subscriber. The subscriber initialization mechanism is suspended.

3 Acceptance of the application

- 3.1 The Registration Authority assigns or accepts the registration number.
- 3.2 The Registration Authority completes the register entry in the registration database.
- 3.3 A key generation and certification request, containing the subscriber's credentials, together with other registration-relevant information, is submitted in a secure way to the Certification Authority.

4 Validation of the certification request

- 4.1 The Certification Authority validates the certification request. If any of the results of the checks are negative, the Certification Authority suspends the subscriber initialization mechanism and resolves the issue together with the Registration Authority.
- 4.2 A key generation request, containing the subscriber's credentials, together with other key generation relevant information is submitted in a secure way to the Key Generation Authority.
Alternatively, the Certification Authority may ask the Key Generation Authority to produce the keys anonymously. Then no information about the subscriber is disclosed to the Key Generation Authority.

5 Key generation

- 5.1 The Key Generation Authority validates the key generation request. If any of the results of the checks are negative, the Key Generation Authority suspends the subscriber initialization mechanism and resolves the issue together with the Certification Authority.
- 5.2 The Key Generation Authority generates an asymmetric key pair and a <protection code> for the private key. The private key is immediately sealed into a security token (software or hardware) using the <protection code>.
- 5.3 Depending on the key escrow requirements (compare also section 11.5.3), the private key may have to be retained in a recoverable form.
Note: the key escrow requirements typically only apply to encryption keys and vary depending on the application range of the key (intra- or inter-organizational, private or public use, etc.).
- 5.4 The Key Generation Authority completes the register entry in the key generation database.
- 5.5 A public key delivery, containing the subscriber's credentials, together with the subscriber's public key and other certification relevant information is sent in a secure way to the Certification Authority. In case of anonymous key generation, the public key delivery does not contain any information about the subscriber but a reference number pointing to the corresponding private key or security token, respectively.
- 5.6 The sealed private key is sent to the Key Distribution Authority. The form of delivery depends on the type of security token. If the security token is a software token, then it may be delivered in a secured electronic message. A software token may also be placed on a physical token and be delivered by mail. If the security token is a hardware token (such as a chipcard), then it may be delivered by mail.
Note: the private key is sealed (e.g. encrypted) with the <protection code> and therefore no additional confidentiality measures are needed.

6 Certification

- 6.1 The Certification Authority validates the public key delivery, as received from the Key Generation Authority. If any of the results of the checks are negative, the Certification Authority suspends the subscriber initialization mechanism and resolves the issue together with the Key Generation Authority.
- 6.2 The subscriber's credentials, together with the subscriber's public key, the registration relevant information and certification relevant information are placed in a public key certificate. The certificate is signed with the Certification Authority's private certification key.
- 6.3 The Certification Authority completes the register entry in the certification database.
- 6.4 The subscriber's certificate is sent
 - a) to the Directory Authority for inclusion in the Directory.
 - b) to the Key Distribution Authority for distribution to the subscriber.
- 6.5 The Certification Authority notifies the Registration Authority that the subscriber's certificate has been generated.

7 Directory update

- 7.1 Upon reception of the directory update request from the Certification Authority, the Directory Authority places the subscriber's certificate in the Directory.
- 7.2 Alternatively, if the Certification Authority has the capability of a secure online write access to the Directory, the certification officer can directly store the certificate in the Directory.

8 Key distribution

- 8.1 Upon reception of the security token containing the sealed private key, the Key Distribution Authority validates the authenticity of the key delivery request. This may require checking a digital signature of an electronic message, or checking a manual signature of a mail or courier delivery.
- 8.2 Upon reception of the certificate, the Key Distribution Authority validates the authenticity of the certificate delivery request.
- 8.3 If any of the results of the checks are negative, the Key Distribution Authority suspends the subscriber initialization mechanism and resolves the issue together with the Certification Authority and the Key Generation Authority.
- 8.4 The subscriber's sealed private key and public key certificate are placed in a <key delivery token>. In case of anonymous key generation, the sealed private key is personalized to the subscriber in such a way that the Key Distribution Authority does not learn the private key. This token can take the form of a message or of a physical envelope.
- 8.5 The Key Distribution Authority notifies the Key Generation Authority that the <key delivery token> is ready and requests distribution of the <protection code> to the subscriber.
- 8.6 The <key delivery token> is transferred in a secure way to the subscriber (possibly with a time delay to insure that the <protection code> arrives beforehand at the subscriber). If required, the Certification Authority and the Registration Authority are notified that the subscriber's sealed private key and public key certificate have been sent to the subscriber.

9 Confirmation of the registration

- 9.1 Upon reception of the Certification Authority's notification, the registration officer knows that the subscriber has been certified and files the confirmation.

10 Protection code distribution

- 10.1 Upon reception of the delivery notification from the Key Distribution Authority, the <protection code> is printed on an authenticated form (e.g. PIN mailer) and is sent securely to the subscriber.

11 Reception

- 11.1 Upon reception of a <rejection form> from the Registration Authority, the subscriber reconsiders his application and resolves the issue.
- 11.2 Upon reception of the <key delivery token> the subscriber stores it securely.
- 11.3 Upon reception of the <protection code>, the subscriber takes the <key delivery token>, and installs the private key at the target system using the <protection code>. After installation, a new <protection code> may be chosen, since after the distribution, the <protection code> has only local significance.
- 11.4 The certificate is validated using the Certification Authority's public certification key. The certificate is made available at the subscriber. This may involve placement in a local repository.
- 11.5 If any problems arise in the course of the installation, the subscriber contacts the Registration Authority or another specified point of contact to resolve the issue.

Note: if the subscriber's direct point of contact is the Registration Authority and he expects all communication to go through the RA, then the Key Distribution Authority may also send the <key delivery token> to the RA which then in turn sends the <key delivery token> to the subscriber.

11.9 Certificate usage

This section explains the generic usage of certificates in a message transfer from subscriber A to subscriber B, involving digital signatures and/or message confidentiality.

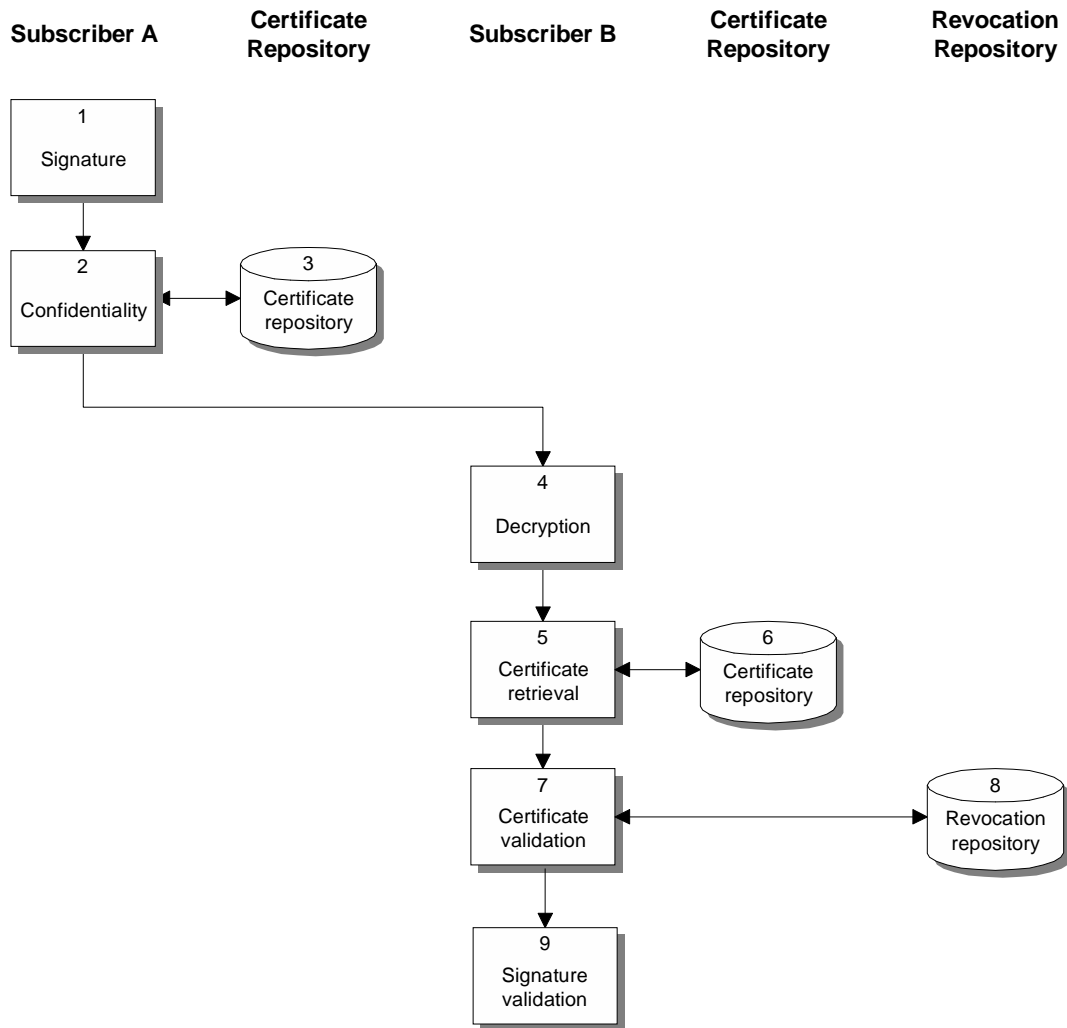


Figure 11-11 Certificate usage

1 Signature

- 1.1 The subscriber has generated or otherwise acquired a message that is to be secured (signed and/or encrypted).
- 1.2 The subscriber checks the message according to the local signature policy and determines which signature(s) have to be applied.
- 1.3 The subscriber adds a timestamp, possibly includes his/her certificate and applies the appropriate signature(s) to the message.
- 1.4 The subscriber logs and/or archives the signature details, as required by the local signature policy.

2 Confidentiality

- 2.1 The subscriber checks the message according to the local confidentiality policy and determines the level of confidentiality.
- 2.2 The subscriber retrieves the confidentiality certificate of the intended recipient from the certificate repository.

- 2.3 The message is encrypted using a dynamic data key. The data key is encrypted under the recipient's public key and is added to the header of the encrypted message.
- 2.4 The subscriber logs and/or archives the encryption details, as required by the local confidentiality policy.
- 2.5 The message is sent to the intended recipient.

3 Certificate repository - Sender

The term *certificate repository* is used as an abstract concept. The sender needs to retrieve the intended recipient's certificate from somewhere. Depending on the actual system, there are various implementation forms for the sender's certificate repository:

- Local cache: the sender may have a local certificate file
- Directory service - offline or online. There may be a file server holding an offline certificate file or there may be an online directory that can be accessed on demand.
- Certification Authority delivery service - The CA may offer offline certificate files or an online request - delivery service.
- The sender may also contact the intended recipient directly and request delivery of the certificate.

4 Decryption

- 4.1 The subscriber checks the received message for correct syntax and timeliness and determines the recipient.
- 4.2 Then the encrypted data key is extracted and decrypted using the private decryption key of the recipient. Then the message is decrypted using the recovered data key.
- 4.3 The subscriber logs and/or archives the decryption details, as required by the local confidentiality policy.

5 Certificate retrieval

- 5.1 The message is checked for timeliness, consistency and correct syntax.
- 5.2 The subscriber retrieves the signature certificate of the sender from the certificate repository.

6 Certificate repository - Recipient

The term *certificate repository* is used as an abstract concept. The recipient needs to retrieve the sender's certificate from somewhere. Depending on the actual system, there are various implementation forms for the recipient's certificate repository:

- The sender's certificate may be sent integrated within the transferred message (contained in the security header).
- Local cache: the sender may have a local certificate file
- Directory service - offline or online. There may be a file server holding an offline certificate file or there may be an online directory that can be accessed on demand.
- Certification Authority delivery service - The CA may offer offline certificate files or an online request - delivery service.
- The sender may also contact the sender directly and request subsequent delivery of the certificate.

7 Certificate validation

- 7.1 The subscriber checks the validity period and the digital signature of the Certification Authority on the signature certificate.
- 7.2 The subscriber checks the revocation status of the certificate using the revocation repository.
- 7.3 The subscriber logs and/or archives the signature details of the received message, as required by the local signature policy.

8 Revocation repository

The revocation repository allows various implementation forms:

- local cache of revocation certificates or of revocation lists
- Directory service for revocation certificates or of revocation lists
- CA delivery or status notification service

9 Signature validation

- 9.1 The subscriber validates the digital signature of the message using the public key from the signature certificate.

11.9.1 Validation of certificates

Since certificates are inherently offline, a major concern in any certificate-based system is the question of how to obtain assurance that a certificate is still valid. This concern is in particular important, if non-repudiation is to be supported as a security service. Non-repudiation requires that the recipient of a signed message can obtain full assurance about the validity of the corresponding certificate. Suppose a recipient has just received a signed message from a sender including his certificate. The basic validation of a certificate consists of two steps: (a) verify the validity period of the certificate, and (b) verify the signature of the Certification Authority. The second step requires that the verifier is in possession of an authenticated copy of the CA's public verification key. This basic validation gives the recipient the assurance that the CA has in fact once issued the certificate. However, if the issuing date of the certificate lies far back, there may be considerable uncertainty whether the certificate has not been revoked in the meantime (i.e. is still valid).

If the public key management infrastructure supports certificate revocation lists (CRLs), the recipient may consult the CRL to obtain higher confidence in the validity of the certificate. This requires (a) obtain a copy of the current CRL, (b) verify that the CRL does not contain the certificate serial number of the certificate in question and, (c) verify that the CRL is genuine by verifying the date of issue and the CA's signature. Consultation of the CRL does only provide partial assurance about the current validity of the certificate. After consultation of the CRL, the recipient knows that the certificate was valid at the time the CRL was issued. Whether that assurance is high enough depends on how far back the CRL has been compiled and signed. To obtain full assurance about the current validity of the certificate based on the CRL concept, one would have to wait for the next CRL. If the serial number of the certificate in question is also not contained in the next CRL, the recipient can be sure that the certificate has been valid when he received the signature in question.

If full assurance about the current validity of the certificate is needed, the public key management infrastructure must be directly contacted. This can be done by either requesting information about the status of the certificate or, equivalently, by requesting the currently valid certificate of the corresponding subscriber. Note that full assurance requires that there is an instant update policy in place and that the link to the public key management infrastructure is secure (in particular, when the information is retrieved from an online directory).

11.10 Certificate change management

Any public key management infrastructure must address the problem of updating existing certificates or issuing new certificates based on existing certificate relationships. Therefore, we distinguish the following cases:

1. Certificate change: the contents of an existing certificate need to be modified in some way and the existing certificate must be replaced by a new one. Such changes of content can be:
 - extending the validity period of a certificate
 - changing credential information such as the address of a subscriber
 - replacing the public key contained in the certificate
2. Certificate issuance: a new certificate is requested for an already registered subscriber, based on one or more existing valid certificates. However, the existing certificates are not to be replaced.

11.10.1 Certificate change

Recommended implementation:

1. Certificate change requests have to be sent to the Registration Authority. The RA has the information necessary to validate any change requests. The RA may also consult with the Certification Authority to come to a decision.
2. Certificate change requests must be properly authorized. If the key does not change, then at least the digital signature of the current key is required. In case of organizational units, organizational persons, and application processes one or more authorizing digital signatures are required (compare section 11.4). If the key itself changes and the key was generated by the subscriber, then an additional self-certifying signature with the new key is required. If the authorizing persons do not have a digital signature capability, then manually signed forms are to be used.
3. If the content of a certificate changes, then a new serial number must be issued by the Certification Authority (this holds even true for the simple case of extending the validity period).
4. A certificate change always must be accompanied by a revocation of the existing 'old' certificate. The reason for revocation should be included in the revocation certificate or the revocation list.

11.10.2 Certificate issuance based on an existing certificate

Recommended implementation:

1. Certificate issuance requests have to be sent to the Registration Authority. The RA has the information necessary to validate any issuance requests. The RA may also consult with the Certification Authority to come to a decision.
2. Certificate issuance requests must be properly authorized. Depending on the type of subscriber, one or more authorizing digital signatures based on existing certificates are required (compare section 11.4). If the key was generated by the subscriber, then an additional self-certifying signature with the new key is required. If the authorizing persons do not have a digital signature capability, then manually signed forms are to be used.

11.11 Revocation

Certificates typically have a fixed validity period. That is, they expire after the end of their validity period. However, there is a number of reasons why a certificate may have to be revoked before the end of that validity period (note: revocation is defined as *to make a certificate ineffective from a specified time forward perpetually*). These reasons include the compromise of a subscriber's private key (suspected or verified), the change of the certificate information, the termination of a subscriber's operation. More dramatically, similar reasons may require to revoke a CA's certificate or public key, and consequently, may require the revocation and subsequent re-certification of all subscriber keys in a domain.

In addition, the revocation process must be handled very carefully, since an erroneous revocation of a subscriber certificate could potentially result in disastrous business consequences. There must be utmost care in verifying the authenticity of a revocation request.

11.11.1 The Certificate Revocation Authority

Throughout this document certificate revocation is treated as an independent function of the public key management infrastructure. Correspondingly, the responsibility for executing the function of certificate revocation is assigned to the abstract entity CRA (Certificate Revocation Authority).

In a real system, certificate revocation is likely to be handled by the same person which has issued the certificate (i.e. the CA). Some guidelines actually recommend that only the CA who has issued a certificate should revoke it, because only the issuing CA is assured of having the records and/or familiarity with the subscriber necessary to confirm the request to revoke, particularly in the urgent circumstances that can give rise to the need to revoke.

The revocation of a certificate can be requested by the subscriber himself or a person acting as an agent with legitimate authority.

The duties of a Certificate Revocation Authority are:

1. The CRA must revoke the certificate if the subscriber or a legitimate agent requests it.
2. Before revoking a certificate, the CRA must confirm that the request is genuine and that the requestor has sufficient authority to cause the requested revocation.
3. Promptly upon revoking a certificate the CRA must publish reasonable notice of the revocation if the certificate was published. Otherwise, the CRA must disclose the fact of revocation on inquiry by a person who has relied or is about to rely on the certificate.
4. The CRA must effect the revocation before expiration of any suspension of the certificate to be revoked. If no suspension is in effect, the CRA must effect the revocation as soon as possible.

11.11.2 Certificate revocation mechanism

A revocation requires the following generic procedure.

Note: The exact content and form of the messages between the Subscriber and the Public Key Management Infrastructure are to be defined in the implementation guidelines or the respective application.

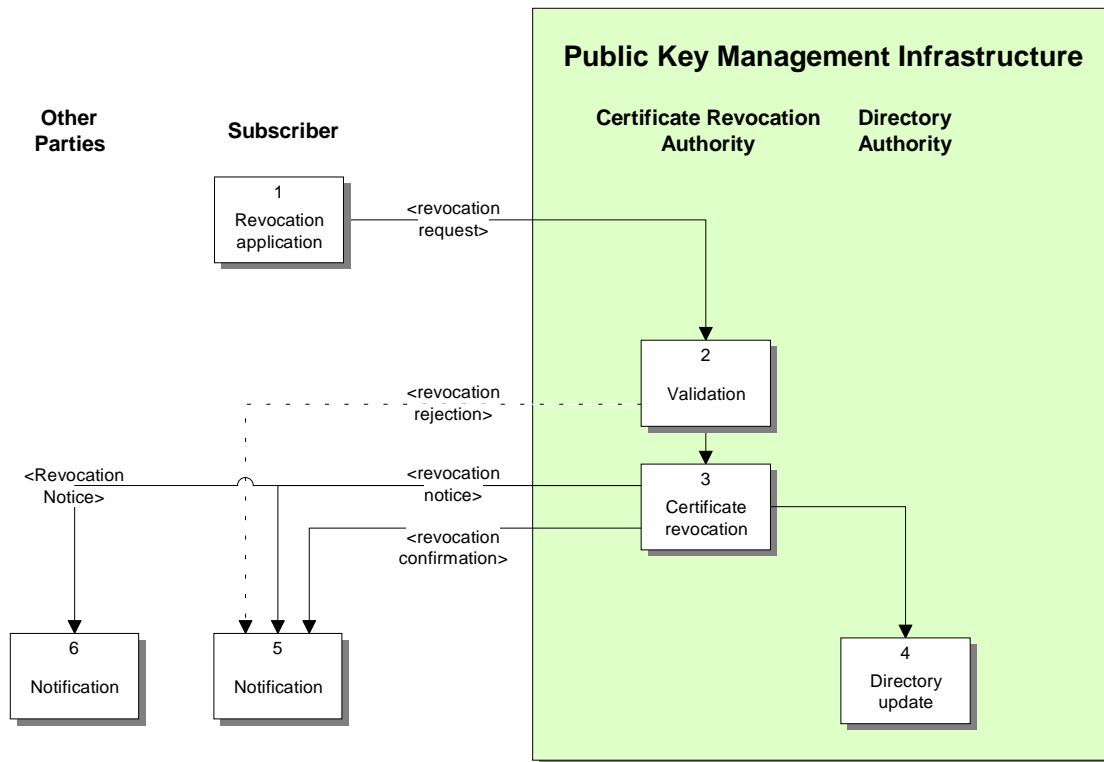


Figure 11-12 Certificate revocation mechanism

1 Revocation application

- 1.1 The subscriber collects the information required for certificate revocation. This information contains name, affiliation, certificate serial number, and possibly revocation reason.
- 1.2 The revocation information is transferred in a <revocation request> to the CRA, using a procedure consistent with the CRA's revocation policy (compare section 11.11.2.1 for implementations of the <revocation request>).
- 1.3 The subscriber disables the use of the private key corresponding to the certificate to be revoked.

2 Validation of revocation request

- 2.1 The CRA validates that all the mandatory and all the optional information is supplied correctly in the <revocation request>.
- 2.2 The CRA checks the authenticity of the <revocation request>, using a procedure consistent with the CRA's revocation policy (compare section 11.11.2.1 for implementations of the <revocation request>).
- 2.3 If any of the results of the checks are negative, the CRA completes a <revocation rejection> specifying the reason for rejection and sends it to the subscriber.

3 Certificate revocation

- 3.1 The CRA revokes the certificate by creating a <revocation certificate>. Alternatively, the CRA may revoke the certificate by including the certificate serial number in a Certificate Revocation List, or by flagging the certificate in the Directory.

- 3.2 The <revocation certificate> is sent in a secure way to the Directory Authority for inclusion in the Directory.
- 3.3 The CRA generates a <revocation notice> which is timestamped and signed with the CRA's private transaction key and is sent to the subscriber and to other parties relying on the certificate.
- 3.4 The CRA prints a <revocation confirmation>, applies an authorized signature and sends it to the subscriber.

4 Directory update

- 4.1 Upon reception of the directory update request from the CRA, the Directory Authority replaces the subscriber's certificate by the new <revocation certificate> in the Directory.
- 4.2 Alternatively, if the CRA has the capability of a secure online write access to the Directory, the CRA can directly store the <revocation certificate> in the Directory.

5 Subscriber notification

- 5.1 Upon reception of a <revocation rejection> from the CRA, the subscriber reconsiders his request and resolves the issue.
- 5.2 The subscriber validates the CRA's digital signature of the <revocation notice>. The information contained in the <revocation notice> is checked for timeliness, consistency and correct syntax.

Note: if the <revocation notice> contains the <revocation certificate> the certificate is stored at the required locations, such as a local directory.
- 5.3 The subscriber receives the <revocation confirmation> and archives it properly.

6 Other parties notification

- 6.1 The party validates the CRA's digital signature of the <revocation notice>. The information contained in the <revocation notice> is checked for timeliness, consistency and correct syntax.
- 6.2 The revocation information is distributed to all relevant entities within the organization and it is ensured that all messages containing a digital signature corresponding to the revoked certificate are rejected.

Note: other parties may include RAs, CAs, and organizations which have subscribed to a special notification service.

11.11.2.1 Revocation request

The following procedures are in general applicable. Which procedures are implemented, depends on the CRA policy. But it is mandatory, that the trust level required from the revocation meets the trust level applied in the registration procedure.

1. In the simplest case the subscriber uses a signed (possibly registered) letter to revoke his certificate. The handwritten signature allows the CRA to confirm authority by comparison with the signature stored in the certification records. The revocation letter must contain name, affiliation, certificate serial number, and revocation date.
2. If the certificate has been suspended by quick (but unauthenticated) communication means such as telephone or fax, the CRA must confirm the revocation request. To that end the CRA can send a confirmation request letter to the subscriber. The subscriber (or his legitimate agent) then returns a signed letter to confirm the revocation.
3. The subscriber uses an electronic revocation request, secured by the signature capability which is to be suspended. Such use assures the CRA that the person requesting suspension is actually in possession of the private

signature key. The risk of fraudulent revocation requests is minimal, however, this procedure does not cover the case where the subscriber is no longer able to use his signature capability (e.g. when he has lost his key). In this case, however, the CRA should still take the precaution to obtain confirmation of the revocation request in writing.

4. The subscriber uses an electronic revocation request, secured by a secondary signature capability recognizable by the CRA. In this case confirmation in writing is not needed.

5. It may be necessary for the CRA to establish the reason of revocation. However, to publish the reason of revocation may not be allowable in all legal systems.

11.11.2.2 Marking a certificate as revoked

The following procedures are in general applicable. Which procedures are implemented, depends on the CRA policy.

1. The CRA can create a *revocation certificate*: that is, it takes the certificate, strips off its signature, changes (or adds) the type into revoked, adds the revocation reason, adds the revocation date and signs the certificate again. The result is a fully recognizable certificate with an included flag indicating the revocation status and giving additional revocation information.

2. The CRA can include the certificate serial number in a *Certificate Revocation List*. To do so, the CRA takes the CRL, strips off its signature, adds the new certificate serial number and the revocation information, changes the last update time, and signs the CRL again.

11.11.2.3 Giving notice of revoked certificates

The following procedures are in general applicable. Which procedures are implemented, depends on the CRA policy.

1. Active notification: the CRA sends revocation notices to persons whom the CRA knows are relying on the certificate.

2. Passive notification with revocation certificates: if the certificate was published in a Directory the CRA can also publish the revocation certificate in the same Directory. This method depends on the availability of revocation certificates in the public key management infrastructure.

3. Passive notification with Certificate Revocation Lists: if the certificate was published in a Directory the CRA can notify persons by publishing an updated Certificate Revocation List in the same Directory. This method depends on the availability of CRLs in the public key management infrastructure.

Whether publication notifies a person of the published information depends on whether it was reasonably likely to impart actual knowledge to the person.

11.11.3 Suspending certificates

There may be emergency situations such as when a subscriber has lost a key or has discovered a compromise and can no longer effectively use his signature key. In such situations it must be possible for a subscriber to quickly and directly make a request to suspend his certificate temporarily. Such direct requests typically do not allow for a full confirmation. However, even in emergency situations, as much precaution as possible must be taken to avoid fraudulent suspensions of a certificate by an illegitimate party.

To confirm the revocation request of a subscriber requires specific procedures and a certain amount of time. During that time, the certificate can neither be used as usual nor can be revoked completely. There are also cases where a subscriber suspects a compromise and wants to temporarily make his certificate ineffective to clarify the situation. Therefore, an intermediate status for certificates is needed. A certificate in such a state is called *suspended*, *on hold*, or *on alert*. The ability to suspend a certificate is a means for the subscriber to manage the risk of holding a private key. The availability of suspension can be varied by contract between the certificate suspension authority and the

subscriber. Subscribers with good key management capabilities may wish to eliminate the prospect of suspension, since they do not want to bear the risk of fraudulent requests for suspension by illegitimate persons.

Suspension is defined as *to make a certificate ineffective temporarily from a specified time forward*.

11.11.4 The Certificate Suspension Authority

Throughout this document certificate suspension is treated as an independent function of the public key management infrastructure. Correspondingly, the responsibility for executing the function of certificate suspension is assigned to the abstract entity CSA (Certificate Suspension Authority).

In a real system, certificate suspension is typically handled by the same person which has issued the certificate (i.e. the CA).

The duties of a Certificate Suspension Authority are:

1. The CSA must provide for quick and effective communication procedures to allow subscribers to submit suspension requests, even in the case when they have lost their signature capability.
2. The CSA must be able to act on suspension requests within seconds and minutes.
3. The CSA must verify to a reasonable degree, based on available or preagreed information, that a suspension request is genuine. However, the CSA is not required to confirm the request.
4. The CSA must carry out the suspension if the subscriber or a legitimate party requests it.
5. Promptly upon suspending a certificate the CSA must publish reasonable notice of the suspension if the certificate was published. Otherwise, the CSA must disclose the fact of suspension on inquiry by a person who has relied or is about to rely on the certificate.

A contract between the certificate suspension authority and the subscriber may provide for different requirements and duties regarding suspension.

11.11.5 Certificate suspension mechanism

A suspension requires the following generic procedure.

Note: The exact content and form of the messages between the Subscriber and the Public Key Management Infrastructure are to be defined in the implementation guidelines or the respective application.

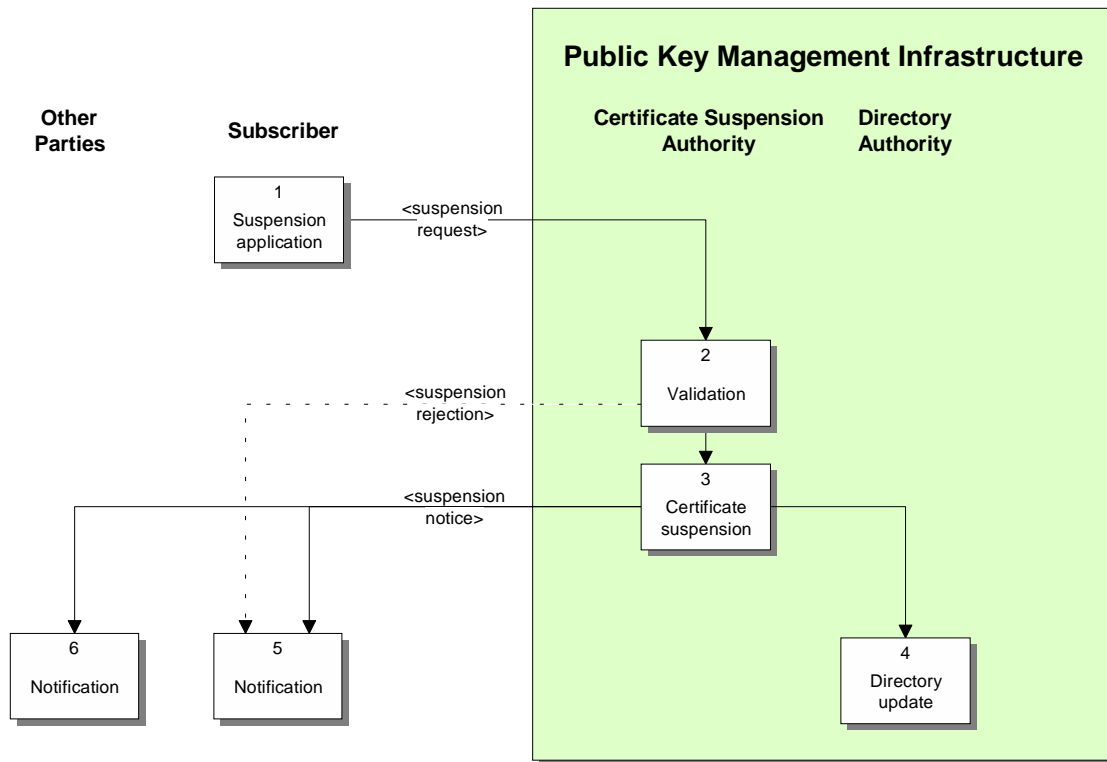


Figure 11-13 Certificate suspension mechanism

1 Suspension application

- 1.1 The subscriber collects the information required for certificate suspension. This information contains name, affiliation, certificate serial number, and suspension time period.
- 1.2 The subscriber composes a <suspension request>, possibly adds an authenticator (such as a password or digital signature) and sends the <suspension request> to the CSA using the preagreed communication means (compare section 11.11.5.1 for implementations of the <suspension request>).
- 1.3 The subscriber disables the use of the private key corresponding to the certificate to be suspended.

2 Validation of suspension request

- 2.1 The CSA verifies the consistency of the information supplied in the <suspension request>.
- 2.2 The CSA verifies any authentication information supplied with the <suspension request> using the preagreed identification details.
- 2.3 If any of the results of the checks are negative, the CSA completes a <suspension rejection> specifying the reason for rejection and sends it to the subscriber.

3 Certificate suspension

- 3.1 The CSA suspends the certificate by creating a <suspension certificate>. Alternatively, the CSA may suspend the certificate by including the certificate serial number in a Certificate Suspension List, or by flagging the certificate in the Directory.
- 3.2 The <suspension certificate> is sent in a secure way to the Directory Authority for inclusion in the Directory.

- 3.3 The CSA generates a <suspension notice> which is timestamped and signed with the CSA's private transaction key and is sent to the subscriber and to other parties relying on the certificate.

4 Directory update

- 4.1 Upon reception of the directory update request from the CSA, the Directory Authority replaces the subscriber's certificate by the <suspension certificate> in the Directory.
- 4.2 Alternatively, if the CSA has the capability of a secure online write access to the Directory, the CSA can directly store the <suspension certificate> in the Directory.

5 Subscriber notification

- 5.1 Upon reception of a <suspension rejection> from the CSA, the subscriber reconsiders his request and resolves the issue.
- 5.2 The subscriber validates the CSA's digital signature of the <suspension notice>. The information contained in the <suspension notice> is checked for timeliness, consistency and correct syntax.

6 Other person notification

- 6.1 The person validates the CSA's digital signature of the <suspension notice>. The information contained in the <suspension notice> is checked for timeliness, consistency and correct syntax.
- 6.2 The suspension information is distributed to all relevant entities within the organization and it is ensured that all messages containing a digital signature corresponding to the suspended certificate are put on hold.

Note: other parties may include RAs, CAs, and organizations which have subscribed to a special notification service.

11.11.5.1 Suspension request

The following procedures are in general applicable. Which procedures are implemented, depends on the CSA policy. It is recommended, that the trust level applied in the subsuspension meets the trust level applied in the registration procedure.

1. In the simplest case (similar to today's credit card procedures) the subscriber uses a telephone call or facsimile message to give the CSA his name, affiliation and certificate serial number, and request suspension for a specified period of time. The CSA only verifies consistency of the supplied information.

Warning: in such a case the subscriber bears a high risk of fraudulent suspension requests.

2. The subscriber and the CSA agree in advance about a secret suspension password only to be used in case of emergency. The communication means could be a telephone call. The subscriber must provide the following identification details: name, affiliation, certificate serial number, a specified period of time, and the suspension password to authenticate the request. Both the subscriber and the CSA must take precautions not to disclose that secret suspension password.

3. The subscriber uses an electronic suspension request, secured by the signature capability which is to be suspended. Such use assures the CSA that the person requesting suspension is actually in possession of the private signature key. The risk of fraudulent suspension requests is minimal, however, this procedure does not cover the case where the subscriber is no longer able to use his signature capability (e.g. when he has lost his key).

4. The subscriber uses an electronic suspension request, secured by secondary cryptographic techniques. If symmetric techniques are used, the subscriber and the CSA must have exchanged a secret key in advance for use in such situations. If asymmetric techniques are used, the subscriber must be in possession of a second signature

capability recognizable by the CSA. The later case actually eliminates the need for suspension and could result in a direct revocation.

11.11.5.2 Marking a certificate as suspended

The following procedures are in general applicable. Which procedures are implemented, depends on the CSA policy.

1. The CSA creates a *suspension certificate*: that is, it takes the certificate, strips off its signature, changes (or adds) the type into suspended, adds the suspension period and signs the certificate again. The result is a fully recognizable certificate with an included flag indicating the suspension status and the corresponding period of time.
2. The CSA includes the certificate serial number in a *Certificate Suspension List*. To do so, the CSA takes the CSL, strips off its signature, adds the new certificate serial number and the suspension period, changes the last update time, and signs the CSL again.
3. The CSA adds the suspension information to the Directory, indicating the suspended status of the certificate, the time of suspension and the suspension period. The suspension information need not be included in the certificate provided there is a secure link to the Directory. Requests for the certificate in question would be answered by signed deliveries including the suspension information and, possibly, the certificate itself.

11.11.5.3 Giving notice of suspended certificates

The following procedures are in general applicable. Which procedures are implemented, depends on the CSA policy.

1. The CSA sends suspension notices to the subscriber and persons whom the CSA knows are relying on the certificate (active notification).
2. If the certificate was published in a Directory the CSA can also publish the suspension certificate in the same Directory (passive notification). This method depends on the availability of suspension certificates in the public key management infrastructure.
3. If the certificate was published in a Directory the CSA can notify persons by publishing an updated Certificate Suspension List in the same Directory (passive notification). This method depends on the availability of CSLs in the public key management infrastructure. In a real system, CSLs and CRLs may be merged.

Whether publication notifies a person of the published information depends on whether it was reasonably likely to impart actual knowledge to the person.

11.11.6 Reactivating a suspended certificate

A suspended certificate must eventually be revoked or be reactivated. The reactivation may be automatic or at the request of the subscriber. It is mandatory, that the trust level applied in the reactivation procedure is at least as high as the trust level applied in the suspension request.

11.11.6.1 Reactivation request

A suspended certificate can be reactivated. The following procedures are in general applicable. Which procedures are implemented, depends on the CSA policy.

1. Automatic reactivation: the suspended certificate is automatically reactivated after expiry of the specified suspension period, provided no further actions are taken by the subscriber (such as revoking the certificate).
2. The subscriber uses the secret suspension password to authenticate the reactivation request, for instance, by telephone. The subscriber must provide the following identification details: name, affiliation, certificate serial number, and the suspension password to authenticate the request. Both the subscriber and the CSA must take precautions not to disclose that secret suspension password.
3. The subscriber uses an electronic reactivation request, secured by the signature capability which has been suspended. Such use assures the CSA that the person requesting reactivation is actually in possession of the private

signature key. This procedure does not cover the case where the subscriber is no longer able to use his signature capability (e.g. when he has lost his key).

4. The subscriber uses an electronic reactivation request, secured by secondary cryptographic techniques.

11.11.6.2 Giving notice of reactivated certificates

The following procedures are in general applicable. Which procedures are implemented, depends on the CSA policy.

1. The legitimate owner of the certificate must receive a signed reactivation confirmation letter from the CSA.
2. The reactivated certificate must be published in the Directory (passive notification) and, in case a certificate suspension list was used, the certificate serial number must be removed from the CSL.
3. To those parties which require an active notification, the CSA sends a reactivation notice.

11.11.7 Combined suspension and revocation

In many instances, subscribers will want to combine suspension and revocation. While the goal may be to revoke the certificate permanently, subscribers will want to make sure that the certificate is suspended immediately, ie. to avoid any damages. An immediate suspension allows for some time to process the slower, but more thorough, revocation procedure.

The diagram and the description of the combined mechanism are not repeated here, since this procedure basically consists of the concatenation of the suspension mechanism with the revocation mechanism (see previous subsections).

12 Glossary

| | |
|---------|---|
| A | Subscriber - may either be a service provider or customer. |
| AC | Asymmetric Cryptosystem based on public key technology. |
| AC1 | Asymmetric Cryptosystem based on RSA. |
| AS | Auxiliary Services. |
| AT | Authentication Token - data packet for entity authentication. |
| B | Subscriber - may be a service provider or customer. |
| BD | General data block. |
| BE | Encrypted data block. |
| BP | Protocol data block. |
| CA | Certification Authority |
| CBC | Cipher Block Chaining mode, for block cipher algorithms. |
| CFB | Cipher Feedback mode, for block cipher algorithms |
| CONF | Confidentiality. |
| CRA | Certificate Revocation Authority |
| CRL | Certificate Revocation List |
| CSA | Certificate Suspension Authority |
| CSL | Certificate Suspension List |
| D | Decryption. |
| d_A | Private key for user A's decryption function (public key technology, counterpart to public encryption key). |
| D_A | User A's private decryption function (public key technology, counterpart to public encryption function). |
| DES | Data Encryption Standard. |
| DIR | Directory Authority |
| DS | Digital signature procedure with recovery of message or user data (public key technology). |
| DS1 | Digital signature with message recovery based on RSA. |
| DSH | Digital signature procedure with data hashing (public key technology). |
| DSH1 | Digital signature with hashing of data 1, combination of DS1 and HF1. |
| DSH2 | Digital signature with hashing of data 2, combination of DS1 and HF2. |
| DSH3 | Digital signature with hashing of data 3, combination of DS1 and HF3. |
| E | Encryption. |
| EA | Entity Authentication. |
| e_A | Public key for user A's encryption function (public key technology, counterpart to private decryption key). |
| E_A | User A's public encryption function (public key technology, counterpart to private decryption function). |
| EA1 | Entity authentication mechanism 1. |
| EA2 | Entity authentication mechanism 2. |
| EA3 | Entity authentication mechanism 3. |
| ECB | Electronic Code Book mode, for block cipher algorithms. |
| EDIFACT | Electronic Data Interchange for Finance, Administration, Commerce and Trade. |

| | |
|---------|---|
| HF | Hash function, support mechanism for the digital signature. |
| HF1 | Block cipher-based hash function using DES. |
| HF2 | Dedicated hash function based on RIPEMD-128. |
| HF3 | Dedicated hash function based on RIPEMD-160. |
| IDEA | International Data Encryption Algorithm. |
| ISO | International Standards Organisation. |
| IT | Information Technology |
| KDA | Key Distribution Authority |
| KG | Key Generation |
| KGA | Key Generation Authority |
| KM | Key Management. |
| KT | Key Token - the data packet transmitted within a key transport protocol. |
| MAC | Message Authentication Code - authentication of user data calculated by symmetric techniques. |
| MOA | Message Origin Authentication. |
| n | Modulus (for RSA signature procedure) |
| NRO | Non-Repudiation of Origin |
| NRR | Non-Repudiation of Receipt |
| OFB | Output Feedback mode, for block cipher algorithms. |
| p | Prime number, factor of modulus n |
| Padding | Padding of user data with extra bits to a specified length. |
| PKMI | Public Key Management Infrastructure |
| PKT | Public Key Transport - procedure for transmitting the public key for asymmetric cryptosystems (verification key, encryption key). |
| PKT1 | Public Key Transport mechanism 1 |
| PKT2 | Public Key Transport mechanism 2. |
| q | Prime number, factor of modulus n |
| RA | Registration Authority |
| s_A | Private key of user A's digital signature function (public key technology, counterpart to public verification key). |
| S_A | User A's private digital signature function (public key technology, counterpart to public verification function). |
| SAFER | Secure And Fast Encryption Routine |
| SC | Symmetric Cryptosystem. |
| SC1 | Symmetric Cryptosystem based on DES. |
| SC2 | Symmetric Cryptosystem based on IDEA. |
| SH | Security header, contains control information for security processing. |
| SKT | Secret Key Transport - procedure for transmitting secret keys for use with symmetric cryptosystems. |
| SKT1 | Key transport mechanism 1. |
| SKT2 | Key transport mechanism 2. |
| SKT3 | Key transport mechanism 3. |
| ST | Security trailer, contains results of security processing. |

| | |
|-------|---|
| Token | In security mechanisms, a <i>token</i> is a data packet transmitted during a pass. A distinction is made between authentication tokens and key tokens. In the field of access control, <i>token</i> also refers to a physical medium (e.g. smart card) for storing security data. |
| TVP | Time-Variant Parameter - can be the time and date, sequence number or random number. |
| UD | User data. |
| UN | United Nations. |
| v | Public verification exponent (in the RSA signature procedure) |
| v_A | Public key of user A's verification function (public key technology, counterpart to private signature key). |
| V_A | User A's verification function (public key technology, counterpart to private digital signature function). |

13 Relevant standards

ISO/IEC 9796 *Digital signature scheme giving message recovery*

ISO/IEC 9798-3 Entity Authentication, Part 3: *Entity authentication using a public key algorithm*

ISO/IEC 9979 *Procedures for the registration of cryptographic algorithms* (currently under revision)

ISO/IEC 10118-2 Hash functions, Part 2: *Hash functions using an n-bit block cipher*

ISO/IEC 10118-3 Hash functions, Part 3: *Dedicated hash functions* (Committee Draft)

ISO/IEC 11770-3 Key Management, Part 3: *Key management using asymmetric techniques* (Committee Draft)

ISO/IEC 10116 *Modes of operation of an n-bit block cipher*

If no international standard exists yet, the status of the document is indicated in brackets at the time of publication of the TBSS. If new versions of these documents are published, they shall serve as binding standards.

Contents - Annexes

| | |
|---|------------|
| A DEFINITIONS | A-1 |
| B REGISTER OF SECURITY MECHANISMS | B-1 |
| C MINIMAL REQUIREMENTS FOR END-USER SYSTEMS | C-1 |
| C.1 Digital signature requirements | C-1 |
| C.1.1 Personal signature | C-1 |
| C.1.2 Corporate signatures | C-1 |
| C.2 Local access control and authorization | C-2 |
| C.2.1 Local access code | C-2 |
| C.2.2 Smart card | C-3 |
| C.3 Archiving of signatures | C-3 |
| C.4 Key Management | C-3 |
| C.4.1 Generation | C-3 |
| C.4.2 Storage | C-3 |
| C.4.3 Backup | C-4 |
| C.4.4 Revocation | C-4 |
| C.4.5 Archiving of keys | C-4 |
| C.5 Service agreement | C-4 |
| D MODES OF OPERATION FOR CRYPTOSYSTEMS | D-1 |
| D.1 Electronic Code Book (ECB) mode | D-1 |
| D.2 Cipher Block Chaining (CBC) mode | D-1 |
| D.3 Cipher Feedback (CFB) mode | D-2 |
| D.4 Output Feedback (OFB) mode | D-3 |
| E KEY AND PSEUDORANDOM NUMBER GENERATION | E-1 |
| E.1 Prime number generation | E-1 |
| E.1.1 Prime number generation using the Miller-Rabin test | E-1 |
| E.2 Pseudorandom number generator based on DES | E-2 |
| E.2.1 ISO 8732 (Annex C) | E-2 |
| E.2.2 Modified generator based on ISO 8732 (Annex C) | E-3 |
| F CERTIFICATE STANDARDS | F-1 |
| G THE CHAMBERS OF COMMERCE REGISTRATION SCHEME | G-1 |
| G.1 The Chambers of Commerce EDI Identifier | G-1 |

| | |
|--|------------|
| G.2 Procedures for Registration | G-1 |
| G.2.1 Request for registration form | G-3 |
| G.2.2 Credentials | G-3 |
| G.3 Additional procedures | G-3 |

Annex

A Definitions

This annex is for information only and is not an integral part of the TBSS standard.

Application process

An object (program) executing on a computer system, which performs the information processing for a particular application (see ISO 7498).

Asymmetric cryptographic technique

A cryptographic technique that uses two related transformations, a public transformation (defined by the public key) and a private transformation (defined by the private key). The two transformations have the property that, given the public transformation, it is computationally infeasible to derive the private transformation. A system based on asymmetric cryptographic techniques can either be an encipherment system or a signature system or both (cf. ISO/IEC 11770-3 Key management using asymmetric techniques).

With asymmetric cryptographic techniques there are four elementary transformations: sign and verify for signature systems, encipher and decipher for encipherment systems. The signature and the decipherment transformation are kept private by the owning entity, whereas the corresponding verification and encipherment transformation are published. There exist asymmetric cryptosystems (e.g. RSA) where the four elementary functions may be achieved by only two transformations: one private transformation suffices for both signing and decrypting messages, and one public transformation suffices for both verifying and encrypting messages.

Asymmetric encipherment system

A system based on asymmetric techniques whose public transformation is used for encipherment and whose private transformation is used for decipherment.

Asymmetric key pair

A pair of related keys where the private key defines the private transformation and the public key defines the public transformation.

Asymmetric signature system

A system based on asymmetric techniques whose public transformation is used for verification and whose private transformation is used for signing.

Certificate

A computer-based record which is digitally signed by an identified certification authority, identifies a subscriber, and contains at least the subscriber's public key.

Certification authority

A person trusted to create and assign certificates. Optionally, the certification authority may create and assign keys. Closely synonymous terms: certificate issuer.

Directory

A database of certificates/public keys and other relevant information available on-line. Closely synonymous terms: repository.

Other relevant information are notices of suspension or revocation. The directory may also contain further information about certification authorities, subscribers, industry standards, etc.

Organizational person

A person (human) employed by, or in some other important way associated with, an organization (see ISO/IEC 9594-7 / ITU-T Rec X.521, The Directory: Selected Object Classes).

Organizational unit

A subdivision of an organization (see ISO/IEC 9594-7 / ITU-T Rec X.521, The Directory: Selected Object Classes).

Owner

The person responsible for the key and its actions.

Person

A human being or any organization capable of signing a document, either legally or as a matter of fact.

Closely synonymous terms: entity.

Private key

That key of a subscriber's asymmetric key pair which is rightfully usable only by that subscriber. In the case of an asymmetric signature system the private key defines the signature transformation. In the case of an asymmetric encipherment system the private key defines the decipherment transformation.

Public key

That key of a subscriber's asymmetric key pair which can be made public. In the case of an asymmetric signature system the public key defines the verification transformation. In the case of an asymmetric encipherment system the public key defines the encipherment transformation.

A key that is 'publicly known' is not necessarily globally available. The key may only be available to all members of a pre-specified group.

Registration authority

A person trusted to identify persons and to assign unambiguous names.

Residential person

A person (human) in the residential (private) environment (cf. ISO/IEC 9594-7 / ITU-T Rec X.521, The Directory: Selected Object Classes).

Revocation

To make a certificate ineffective from a specified time forward perpetually.

Subscriber

A person who is registered with the Public Key Management Infrastructure and rightfully holds a private key corresponding to a registered and valid public key.

Closely synonymous terms: signer, signatory, user.

Suspension

To make a certificate ineffective temporarily from a specified time forward.

User

A person who has the right to activate the private key corresponding to a registered public key, i.e. execute a digital signature.

Annex

B Register of security mechanisms

This annex is for information only and is not an integral part of the TBSS standard.

The following table lists all the trusted security mechanisms in TBSS in alphabetical order. The table is laid out as follows:

1. Security mechanism
2. Service: security service that can be realized with the mechanism.
 - NRO: Non-Repudiation of Origin
 - NRR: Non-Repudiation of Receipt
 - EA: Entity Authentication
 - CONF: Confidentiality (encryption)
 - KM: Key Management
3. Description: brief description of the security mechanism.
4. Standard: the international standards that the mechanism complies with or is based on.
5. Based on: the mechanisms, procedures or algorithms on which the mechanism is based.

| Mechanism | Service | Description | Standard | Based on |
|-----------|------------------|--|---------------------------------|------------|
| AC1 | KM | Asymmetric encryption using RSA. | related to ANSI X9.31 | RSA |
| DS1 | NRO NRR | Digital signature with message recovery. | ISO/IEC 9796 | RSA |
| DSH1 | NRO NRR | Digital signature with data hashing. Block cipher-based hash function with DES as basic technique. | ISO/IEC 9796 ISO/IEC 10118-2 | DS1 HF1 |
| DSH2 | NRO NRR | Digital signature with data hashing. Dedicated hash function with RIPEMD-128 as basic technique. | ISO/IEC 9796 ISO/IEC 10118-3 | DS1 HF2 |
| DSH3 | NRO NRR | Digital signature with data hashing. Dedicated hash function with RIPEMD-160 as basic technique. | ISO/IEC 9796 ISO/IEC 10118-3 | DS1 HF3 |
| EA1 | EA | One-way authentication in one pass using timestamps or sequence numbers. | ISO/IEC 9798-3 | DSH |
| EA2 | EA | One-way authentication in two passes using random numbers. | ISO/IEC 9798-3 | DSH |
| EA3 | EA | Mutual authentication in three passes using random numbers. | ISO/IEC 9798-3 | DSH |
| HF1 | NRO NRR KM | Block cipher-based hash function with DES as basic technique. | ISO/IEC 10118-2 | DES |
| HF2 | NRO NRR KM | Dedicated hash function with RIPEMD-128 as basic technique. | ISO/IEC 10118-3 | RIPEMD-128 |

| | | | | |
|------|------------------|--|--|--------------------------------|
| HF3 | NRO NRR KM | Dedicated hash function with RIPEMD-160 as basic technique. | ISO/IEC 10118-3 | RIPEMD-160 |
| SKT1 | EA KM | Key management (one-pass) using timestamps or sequence numbers. This mechanism is an extension of EA1. | ISO/IEC 9798-3 ISO/IEC 11770-3 ISO 11166-1 | DSH AC1 |
| SKT2 | EA KM | Key management (two-pass) with one-way authentication based on random numbers. This mechanism is an extension of EA2. | ISO/IEC 11770-3 ISO/IEC 9798-3 | DSH AC1 |
| SKT3 | EA KM | Key management (three-pass) with mutual authentication based on random numbers. This mechanism is an extension of EA3. | ISO/IEC 11770-3 ISO/IEC 9798-3 | DSH AC1 |
| PKT1 | KM | Registration and distribution of public key data via a secure communications medium. | ISO/IEC 11770-3 X.509 | physical security |
| PKT2 | KM | Registration and distribution of public key data via a non-secure communications medium. Authentication via a second independent communications medium (e.g. registered letter). | ISO/IEC 11770-3 X.509 | DSH HF physical security |
| SC1 | CONF | Data Encryption Standard with operating modes ECB, CBC, CFB, OFB. | ANSI X3.92 ISO/IEC 10116 | DES |
| SC2 | CONF | IDEA with ECB, CBC, CFB, OFB modes of operation. | ISO 9979/0002 ISO/IEC 10116 | IDEA |

Table: Register of security mechanisms

Annex

C Minimal requirements for end-user systems

This annex is for information only and is not an integral part of the TBSS standard.

C.1 Digital signature requirements

The digital signature can belong to a private individual (like a conventional handwritten signature) or a company (as a binding technical security measure for a corporate body).

C.1.1 Personal signature

The following requirements are needed to satisfy current business practices:

1. Just as at present a signature card is used to verify whether the party issuing an transaction is actually an authorized signatory, in future authentication of signatory power must also be possible by electronic means.
2. Just as at present certain transactions require more than one signature, in future joint (or multiple) signatures must also be possible by electronic means.

For the first requirement to be satisfied, each of the customer's authorized signatories must be assigned a signature key pair, and the corresponding verification data must be added to the customer's master file held on the Service provider's database.

For the second requirement to be satisfied, data (i.e. a file or message) that have already been signed once have to be signed a second time. It must be possible for each signing to be performed separately. This means that the digital signatures only protect the actual user data, so the sequence is not important. As an alternative it should also be possible for the second digital signature to include the first one as a data element ("onion skin" principle). This gives the receiver the technical means to verify the sequence of signing (left and right signatories).

The following organizational and technical measures are needed for signatory authorization and multiple signatures:

1. Personal digital signatures at the customer's end.
2. Personal access control to the signature functions at the customer's end.
3. Management of authorized signatories at the customer's end and at the service provider's end.
Management of criteria that require joint or multiple signatures, and their technical implementation.
Notifying service providers of all changes to signatory powers.

C.1.2 Corporate signatures

The growth of automatic data processing has led to greater demand for corporate digital signature techniques. Simplification of administrative procedures offers potential cost savings at the service provider's end in particular. Examples of expedient use of corporate signatures include:

1. Signature of the certification authority (to generate a public key certificate)
2. Automatic acknowledgement of receipt by the service provider, where immediate confirmation of a transaction is required.
3. Automatic acknowledgement of delivery by a network operator (carrier), which performs a similar function as a registered letter.
4. Batch transactions from a company, where individual transactions are collected with internal control methods and the batch is then signed with the company key.

Basically these requirements can also be satisfied with personal signatures, as in current business practice, but in most cases this leads to more time-consuming administrative procedures.

When using corporate digital signatures, special attention must be paid to internal access control to private signature keys and to the relevant signature function. Since the signatures are binding on the company, they must be protected from internal abuse. This also means that the use of corporate digital signatures is restricted to certain applications.

C.2 Local access control and authorization

As already mentioned in Chapter 1, *Contents* of the TBSS provides procedures for secure data communications between two end-user systems that enable:

- Non-repudiation between the communication partners and
- Protection against intruders (encryption, entity authentication).

But overall security not only depends on the techniques used for end-to-end communications, but on the access control implemented locally on the end-user systems. Suitable access control measures must be provided so that

- Secret keys (especially signature keys) cannot be disclosed
- The functions associated with the keys (especially digital signatures) can only be performed by users with the appropriate privileges.

This means that users who perform security-critical functions must be correctly identified and authorized. The following basic principles apply to user identification:

1. Identification by personal characteristics ("something you are"): finger prints, retina or voice pattern, etc. These biometric techniques are not very widespread at the moment.
2. Identification based on certain secret information ("something you know"): secret code. This is the most common identification method currently used.
3. Identification based on the possession of a characteristic object ("something you possess"): swipe card, smart card, etc.

Combinations of (2) and (3) are fairly common. The following basic principles apply to access control to the secret signature key or the relevant signature function:

1. Logical protection, through encryption for example.
2. Physical protection, for example by incorporating the signature function (including the signature key) in hardware that is resistant to manipulation.
3. Physical separation, for example by transferring the signature key (or the whole signature function) to an external medium (diskette, smart card, etc.).

In current practice, the following solutions are commonly used for local access control:

1. Software-based solution: logical protection of the signature key through encryption and transfer to a backup diskette. For the person to be successfully identified, he must possess the key diskette and know the correct secret code.
2. Hardware-based solution: the whole signature function (including the signature key) is transferred to a smart card. For the person to be successfully identified, he must possess the smart card and know the correct secret code to activate the smart card.

The open concept used for the TBSS means that the access control module can be replaced by another variant at any time.

C.2.1 Local access code

At the customer's end, access control can be via a local secret code. The advantages of this pragmatic approach are flexibility and the separation of local access control from message security. The secret code can be used to decrypt the signature key or to activate a signature token.

Since the secret code is a key element in the central access control concept to the Telebanking functionality, it must be carefully managed at the customer's end. The secret code must satisfy minimum requirements as regards length and diversity, it must be kept secret, it must be changed at regular intervals and backup copies must be stored in a safe place. The Telebanking secret codes must not be simultaneously used as passwords for logging in to local systems and applications.

C.2.2 Smart card

Access control can be via a smart card or other intelligent token. A smart card substantially enhances the level of security, since it also solves the problem of key storage with physical protection. A distinction is made between

1. Smart card without signature capability: this can only be used to save security-critical elements. The computation of the digital signature and the generation of the asymmetric key pair must be performed outside the smart card.
2. Smart card with signature capability: the digital signatures are computed directly on the smart card, which acts as a security module.

The implementation of the smart card must be carefully planned. The different steps involved in the issue of smart cards to customers must be specified: initialization, registration, key generation, personalization.

A secret code must protect smart cards from misuse if lost.

C.3 Archiving of signatures

Transactions and signatures usually have to be held on file for several years. If a digital signature is to be archived, we have to ensure that it can be verified for several years as well. This means that:

1. The signed message must be archived in recoverable format, along with a reference to the matching verification key,
2. The customer's verification key must be archived together with its period of validity. If a certification scheme is subsequently introduced, the certificates may also be archived at a central place in the key management infrastructure in order to comply with compulsory safekeeping rules (see also Chapter 11).

C.4 Key Management

This chapter describes the requirements for key management of the asymmetric key pairs. An asymmetric cryptosystem comprises two keys: the secret signature/decryption key and the public verification/encryption key.

C.4.1 Generation

Each user generates his own key pair. To this end, his IT system or application must have a suitable key generation function. Care must be taken to ensure that key generation satisfies modern requirements as far as key pair diversity and strength are concerned.

Some users may find it more convenient to commission an independent third party to perform time-consuming key generation functions. The generation of asymmetric key pairs could therefore be offered as a supplementary service. In this case the service provider would have to be able to prove the confidentiality of this supplementary service so as not to jeopardize the non-repudiation feature (the essential advantage of asymmetric cryptosystems).

C.4.2 Storage

The security requirements for storing both components of the asymmetric key pair vary:

- The user's signature key must be kept secret. In a software environment this can be achieved by encryption, or through physical security (such as storage in a smart card).

- The user's verification key does not have to be kept secret. It has to be transmitted in an authenticated manner to the intended receiver (service provider or key management infrastructure), and after transmission it is in theory no longer required. Even so, it is a good idea to archive the verification key.
- In open systems the public verification key can be certified by a certification authority.

C.4.3 Backup

The user should make a secure copy of his key pair for backup purposes. The keys can be stored in encrypted form (e.g. using the user's own secret code) on diskette or on a physically secure token (e.g. on a smart card or memory card). Whichever backup media is selected, the user's key pair must be stored in a secure place (such as a safe).

C.4.4 Revocation

Revocation is when a key pair is withdrawn from operational use. The reasons for revocation can be:

- The user wants to stop using his key pair or he is withdrawing from the Telebanking service.
- The user's key pair is no longer valid.
- The user knows or suspects that his signature key has been compromised.

The user must therefore have a fast and independent means of communication to the service provider (the key management infrastructure) in order to revoke his key pair (or his public verification key). The service provider must also publish a list of all expired/revoked user keys, together with their relevant expiry date, to avoid potential conflicts.

C.4.5 Archiving of keys

After finishing their service life, keys need to be archived for audit purposes. Care must be taken to archive any other relevant data (ID, validity, etc.) along with the key to ensure that verification is possible in the event of disputes subsequently arising.

1. The secret signature key does not have to be archived.
2. The user's public verification key must be stored by the receiver (i.e. Service provider) for non-repudiation purposes. In this context it is important to note that the user's signed letters must also be archived with the hash value of the verification key, since this is the only way of proving the link between the user and his key. The authenticity of the key archive must be guaranteed as well.

If a certification scheme is subsequently introduced, the certificates may also be archived at a central unit in the key management infrastructure in order to comply with compulsory safekeeping rules (see also Chapter 11).

C.5 Service agreement

The TBSS must be incorporated in a service agreement that deals with matters such as liability, archiving, signatory authorization, etc.

Annex

D Modes of operation for cryptosystems

This annex is for information only and is not an integral part of the TBSS standard.

ISO 10116 specifies the different modes of operation for general n-bit block cipher techniques. There are four modes of operation:

1. Electronic Code Book (ECB) mode
2. Cipher Block Chaining (CBC) mode
3. Cipher Feedback (CFB) mode
4. Output Feedback (OFB) mode

This annex describes the characteristics and potential uses of these four modes of operation.

D.1 Electronic Code Book (ECB) mode

ECB mode has the following **characteristics**:

1. Encryption/decryption of individual data blocks can be performed independently of the encryption/decryption of other data blocks.
2. ECB mode does not offer any protection against deleting individual ciphertext blocks or manipulation of the sequence of ciphertext blocks.
3. Identical plaintext blocks are always transformed into the same ciphertext blocks (if the key is the same). This allows conclusions to be drawn about the data blocks by comparing the ciphertext blocks (*ciphertext searching* or *dictionary attack*).
4. Transmission errors always affect the relevant user data block. A bit error in a ciphertext block on average corrupts about 50% of the bits in the data block in question.
5. If bit synchronization fails between sender and receiver (i.e. if a bit is lost or appended during transmission), decryption will be disrupted from this point onwards.

ECB mode is suitable for the following **implementations**:

The encryption of short user data (less than one block), e.g. when used in key management, if keys or initialization vectors are to be encrypted.

D.2 Cipher Block Chaining (CBC) mode

CBC mode has the following **characteristics**:

1. The suppression of data blocks is detected, apart from the last data block of the ciphertext. A missing ciphertext block means that none of the subsequent user data blocks will be correctly decrypted.
2. The interchange of data blocks is detected. If two adjacent ciphertext blocks are interchanged three plaintext data blocks will not be correctly decrypted.
3. Random access: data decryption can commence anywhere within the ciphertext. To this end, the ciphertext block must be loaded as an initialization vector immediately before the data to be decrypted.

4. The initialization vector offers additional variation. By varying the IV, the same user data are transformed into different ciphertexts (as long as the key is the same). This makes dictionary attacks virtually impossible.
5. Error propagation/error tolerance: bit errors in a ciphertext block corrupt the corresponding and the subsequent plaintext block. About 50% of the bits in the relevant block are corrupted, whilst in the subsequent block there are errors at the positions where the bit errors occurred in the ciphertext block. But as soon as two correct ciphertext blocks are detected at the receiver's end, the decryption procedure automatically recovers and once again produces correct user data.
6. If bit synchronization fails between sender and receiver (i.e. if a bit is lost or appended during transmission), decryption will be disrupted from this point onwards.
7. The loss of block synchronization is similar to the problem experienced with the suppression of ciphertext blocks (see above).
8. Data expansion: if the user data is not a multiple of the encryption algorithm's block length, data expansion can be averted by a special treatment of the last incomplete user data block. Annex A2.3 of ISO 10116 describes two methods suited to this purpose. The first is easy to implement, but it also poses certain security problems. The second method is more secure, but entails complex processing stages.
9. Performance: one block per encryption. Pipelining is possible.

CBC mode is suitable for the following **implementations**:

CBC is the most versatile mode of operation. It is secure for both long and short user data and allows random data access at any point in the ciphertext. CBC can function with or without data expansion. CBC also provides the basis for the recognized MAC technique.

D.3 Cipher Feedback (CFB) mode

CFB mode has the following **characteristics**:

1. The suppression of data elements is detected. A missing ciphertext element means that at least 64 bits of the user data will not be correctly decrypted.
2. Random access: data decryption can commence anywhere within the ciphertext. To this end, the ciphertext elements must be loaded as initialization vectors, immediately before the data to be decrypted (64 bits).
3. The role of the initialization vector is to ensure that the same user data are encrypted as different ciphertexts (when the key remains the same). This makes dictionary attacks virtually impossible.
4. Error propagation/error tolerance: bit errors in a ciphertext element corrupt that particular data element and some subsequent ones (at least 64 bits). As soon as sufficient correct ciphertext elements arrived at the receiver's end (64 bits), the decryption procedure automatically recovers and once again produces correct user data. In other words CFB is error-tolerant.
5. If element synchronization fails between sender and receiver (i.e. if an element is lost or appended during transmission), decryption will be disrupted from this point onwards until the receive register is again filled with correct ciphertext elements (64 correct bits). In other words, CFB is self-synchronizing as far as elements are concerned
6. If bit synchronization fails between sender and receiver (i.e. if a bit is lost or appended during transmission) and the data elements are more than one bit long, decryption will be disrupted from this point onwards.
7. Data expansion: in general there is no data expansion, since the element length in CFB mode is typically adjusted to the source format (e.g. element = ASCII character).
8. Performance: for each encryption operation, one element of length j can be encrypted. The performance level is therefore $j/64$, as long as no pipelining is used.

CFB mode is suitable for the following **implementations**:

CFB is suitable for encrypting short user data elements (e.g. 8-bit characters) or where self-synchronizing features are required.

D.4 Output Feedback (OFB) mode

OFB mode has the following **characteristics**:

1. The suppression of data blocks is detected. A missing ciphertext element leads to the loss of synchronization between sender and receiver.
2. Random access: it is difficult to decrypt data starting from anywhere within the ciphertext.
3. The role of the initialization vector is to ensure that the same user data are encrypted as different ciphertexts (when the key remains the same).
4. Error propagation/error tolerance: there is no error propagation. A bit error in a ciphertext element only affects the bit in question. However, this poses the risk that the plaintext can be manipulated by selective manipulation of the ciphertext.
5. Integrity protection: since there is no error propagation, there is no integrity protection in OFB mode.
6. If element or bit synchronization fails between sender and receiver (i.e. if an element or bit is lost or appended during transmission), decryption will be disrupted from this point onwards. OFB is not self-synchronizing.
7. Data expansion: in general there is no data expansion since the element length in OFB mode is adjusted to the source format (e.g. element = ASCII character).
8. Performance: for each encryption operation, one element of length j can be encrypted. The performance level is therefore $j/64$, as long as no pipelining is used. On the other hand, the key stream can be calculated in advance. The actual data encryption consists of the bitwise addition of the key stream to the user data stream; in other words, the block cipher is no longer required for the actual data encryption.

OFB mode is suitable for the following **implementations**:

OFB can be used as a pseudorandom number generator or for encryption at very high speeds (the key stream can be calculated in advance). However, OFB does not offer self-synchronization and integrity protection. When using OFB, the integrity of the user data therefore has to be guaranteed by a separate security service.

Annex

E Key and pseudorandom number generation

This annex is for information only and is not an integral part of the TBSS standard.

E.1 Prime number generation

First the range I in which the prime number is supposed to lie has to be defined. With RSA the range for both factors p and q is dictated by the requirement that the modulus n should have a length of $|n|=k$ bits and that both factors should be roughly the same size.

E.1.1 Prime number generation using the Miller-Rabin test

The Miller-Rabin test can be used to differentiate prime numbers from composite numbers. The test has the following property: if a composite number is submitted, the test detects it with a probability at least $3/4$. The test can be run several times, with different test parameters, on the same candidate prime. Successful completion of each test run therefore increases the likelihood that the candidate number examined is in fact a prime number. For example, after the Miller-Rabin test has been successfully completed 6 times, the error probability for a 250-bit candidate prime number is less than 2^{-54} . The probability of error can be reduced as much as required by increasing the number of test runs.

Prime number generation using the Miller-Rabin Test:

Input: range I in which the prime number is to be.

1. Generate a random odd number p in the required range I .
2. Check whether p is divisible by one prime number smaller than a threshold r . If so, return to step 1.
3. Check whether p successfully passes the Miller-Rabin test t times. If not, return to step 1.

Miller-Rabin test

Input: integer p

Output: 'OK' or 'not OK'

1. Break down $p-1 = a \cdot 2^h$, a is odd
2. Select a random b from the interval $[2..p-2]$ and calculate $x = b^a \pmod p$. If $x = 1$, then end the test and display 'OK'.
3. Check whether there is an i with $0 < i < h$, so that the congruence $x^{2^i} \pmod p = p-1$ is met. If so, end the test and display 'OK'.
4. Otherwise display 'not OK'.

Output: p

With this prime number generator, a uniformly random choice of prime number candidates results in a uniformly random choice of prime numbers.

With prime number generation a random number has to be selected (step 1). But most computer systems do not have a source of true random numbers. In such cases, one must fall back on a pseudorandom number generator that is initialized with a starting value.

Since in RSA the prime numbers are part of the secret user key, care also has to be taken to ensure that the starting values for prime number generation are kept secret as well.

The starting values must be sufficiently diverse, otherwise the prime number could be worked out by testing all the starting values. The best choice is $k/2$ bits, but the starting value should have a diversity of at least 64 bits.

As a rule the high security requirements for key generation do not allow the use of a pseudorandom number generator shipped as standard with a programming language. Nor must starting values be derived from publicly known parameters (such as system time).

E.2 Pseudorandom number generator based on DES

E.2.1 ISO 8732 (Annex C)

Annex C of ISO 8732 (see also ANSI X9.17) describes the following pseudorandom number generator.

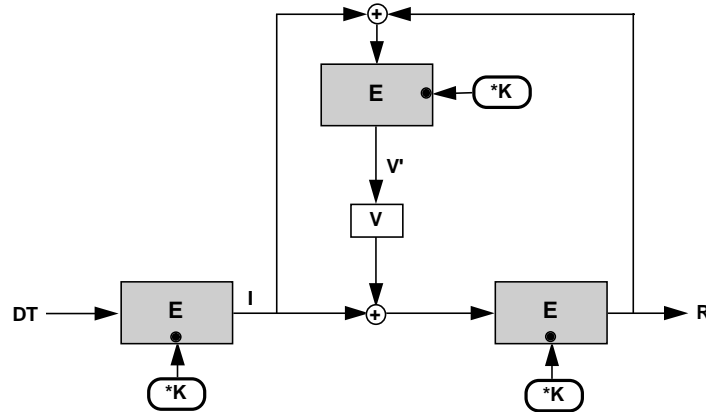


Figure E-1 Pseudorandom number generator according to ISO 8732

The following conventions apply:

| Abbreviation | Meaning |
|--------------|--|
| ede*K(Y) | Triple DES encryption of Y with key pair *K |
| *K | DES key pair |
| DT | Date and time vector |
| V | Stored internal 64-bit state (initial value V ₀) |
| V' | Newly computed internal 64-bit state |
| + | Bitwise XOR |

A random number R is generated using the following equations:

$$I = ede*K(DT)$$

$$R = ede*K(I+V)$$

A new state V' is generated by

$$V' = ede*K(R+I)$$

If the key pair *K, the output R and the date DT are known, the internal state V can be calculated. If the previous date is known, the previous value R can be worked out as well. This can be avoided by making the modification described in the next section.

E.2.2 Modified generator based on ISO 8732 (Annex C)

To prevent inversion of the generator provided for under ISO 8732 (Annex C), the design philosophy of the hash function in ISO 10118-2 can be applied. This results in the following modified generator.

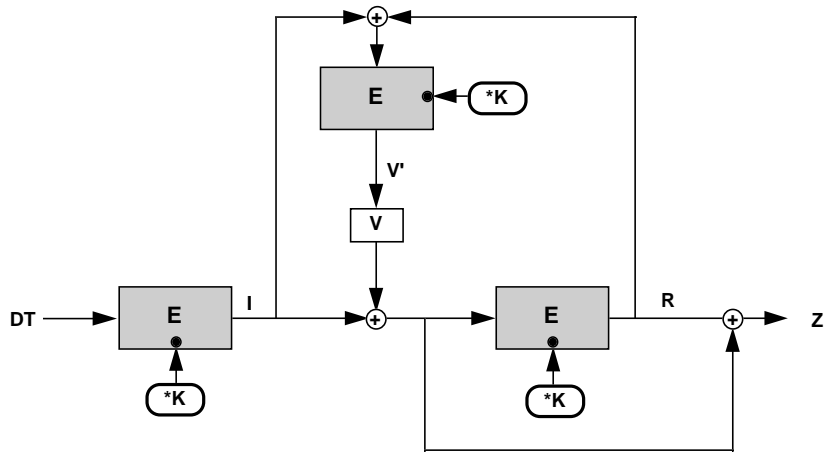


Figure E-2 Modified pseudorandom number generator based on ISO 8732

Instead of outputting R directly, the random number Z is generated with the following additional equation:

$$Z = R + I + V$$

Annex

F Certificate standards

This annex is for information only and is not an integral part of the TBSS standard.

Currently, there are two global standards which address the format of a public key certificate: X.509 and UN/EDIFACT. The following table shows the combined functionality of the two standards. Only the X.509 version v3 certificate is shown, since it contains the versions v1 and v2 as subsets. The first column called Meta Certificate lists the generic names of the data items contained in the different representations.

| Meta Certificate | EDIFACT | X.509 v3 |
|--|--|----------------------------------|
| Version number | FORMAT CERTIFICATE VERSION | version |
| Certificate reference number | CERTIFICATE REFERENCE | serialNumber |
| Certificate issuer identification | SECURITY IDENTIFICATION DETAILS: Authenticating party | issuer |
| Logical name | Party Id Identification | "X.509 Distinguished Name" |
| Code list qualifier | Code list qualifier | - |
| Code list responsible agency | Code list responsible agency | - |
| Natural name | Party name | - |
| Alternate issuer name | - | issuerAltName |
| Additional issuer identifier | - | issuerUniqueIdentifier |
| Certificate issuer signature algorithm | SECURITY ALGORITHM: Issuer signing | signature |
| Cryptographic mode of operation, coded | Cryptographic mode of operation, coded | - |
| Mode of operation code list identifier | Mode of operation code list identifier | - |
| Algorithm, coded | Algorithm, coded | algorithm |
| Algorithm code list identifier | Algorithm code list identifier | - |
| parameters | ALGORITHM PARAMETER (repeated) | parameters |
| Certificate issuer hash algorithm | SECURITY ALGORITHM: Issuer hashing | - |
| Cryptographic mode of operation, coded | Cryptographic mode of operation, coded | - |
| Mode of operation code list identifier | Mode of operation code list identifier | - |
| Algorithm, coded | Algorithm, coded | - |
| Algorithm code list identifier | Algorithm code list identifier | - |
| parameters | ALGORITHM PARAMETER (repeated) | - |
| Certificate issuer public key identification | SECURITY IDENTIFICATION DETAILS: Authenticating party | authorityKeyIdentifier |
| Key identification | key name | keyIdentifier |
| Reference to issuer certificate | - | certIssuer + certSerialNumber |

| Meta Certificate | EDIFACT | X.509 v3 |
|---|--|--------------------------------|
| Certificate owner identification | SECURITY IDENTIFICATION DETAILS: Certificate owner | subject |
| Logical name | Party Id Identification | "X.509 Distinguished Name" |
| Code list qualifier | Code list qualifier | - |
| Code list responsible agency | Code list responsible agency | - |
| Natural name | Party name | - |
| Alternate certificate owner name | - | subjectAltName |
| Additional owner identifier | - | subjectUniqueIdentifier |
| Certificate validity period | SECURITY DATE AND TIME | validity |
| Begin of validity period | Certificate start of validity period | notBefore |
| End of validity period | Certificate end of validity period | notAfter |
| Certificate, last update time | SECURITY DATE AND TIME: Certificate generation time | - |
| Certificate owner public, key algorithm information | SECURITY ALGORITHM: owner | subjectPublicKeyInfo |
| Associated algorithm identification | Algorithm, coded | algorithm |
| Algorithm code list identifier | Algorithm code list identifier | - |
| Associated algorithm, mode of operation | Cryptographic mode of operation, coded | - |
| Mode of operation code list identifier | Mode of operation code list identifier | - |
| Certificate owner, public key identification | SECURITY IDENTIFICATION DETAILS: Certificate owner | primaryKeyAttributes |
| key identification | key name | keyIdentifier |
| Certificate owner, public key components | ALGORITHM PARAMETER (repeated) | subjectPublicKeyInfo |
| Public key component value | Algorithm parameter value | subjectPublicKey parameters |
| Public key component qualifier, coded | Algorithm parameter qualifier, coded | - |
| Certificate owner, public key usage | SECURITY ALGORITHM: owner | primaryKeyAttributes |
| Intended use | Use of algorithm, coded | keyUsage |
| Usage restriction | - | primaryKeyUsageRestriction |
| Certificate owner, private key validity period | - | primaryKeyAttributes |
| Validity period | - | privateKeyUsagePeriod |
| Certificate owner, additional public key, algorithm information | - | otherPublicKey |
| Associated algorithm identification | - | algorithm |
| Algorithm code list identifier | - | - |
| Associated algorithm, mode of operation | - | - |
| Mode of operation code list identifier | - | - |
| Certificate owner, additional public key, identification | - | otherPublicKey |
| key identification | - | keyAttributes; keyIdentifier |
| Certificate owner, additional public key, components | - | otherPublicKey |

| Meta Certificate | EDIFACT | X.509 v3 |
|---|--|---|
| Public key component value | - | otherPublicKey |
| Public key component qualifier, coded | - | - |
| Certificate owner, additional public key, usage | - | otherPublicKey |
| Intended use | - | keyAttributes; keyUsage |
| Usage restriction | - | keyAttributes; keyUsageRestriction |
| Certificate owner, additional private key, validity period | - | otherPublicKey |
| Validity period | - | keyAttributes; privateKeyUsagePeriod |
| Certificate owner, authorisation to act as CA | USER AUTHORISATION LEVEL | cAorEndEntityIndicator |
| Certificate policy information | - | |
| Supported certificate policies | - | certificatePolicies |
| Equivalent certificate policies | - | policyMappings |
| Constraints for names to be used for the remainder of the certification path | - | nameConstraints |
| Constraints on the policies to be used for the remainder of the certification path. | - | policyConstraints |
| Additional Directory attributes | - | subjectDirectoryAttributes |
| Syntax information | USC Segment | - |
| filter function, coded | FILTER FUNCTION, CODED | - |
| character set encoding, coded | CHARACTER SET ENCODING, CODED | - |
| character set repertoire, coded | CHARACTER SET REPERTOIRE, CODED | - |
| separator for signature, coded (repeat) | SEPARATOR FOR SIGNATURE | - |
| Type of certificate | CERTIFICATE TYPE | - |
| Revocation information | | - |
| time of revocation | SECURITY DATE AND TIME: Certificate revocation time | - |
| reason, coded | REVOCAION REASON | - |
| Application context | - | - |

Annex

G The Chambers of Commerce Registration Scheme

This annex is for information only and is not an integral part of the TBSS standard.

This section explains the registration scheme and procedures adopted by the Chambers of Commerce, based on the European project EDIRA. As a requirement by the Swiss banking sector, this registration scheme shall be supported by the TBSS management infrastructure.

The Chambers of Commerce have agreed to register

1. Organisations registered in the commercial register
2. Organisations (legal or natural persons) not registered in the commercial register

G.1 The Chambers of Commerce EDI Identifier

According to the Chambers of Commerce's Annex Z of the EDIRA Memorandum of Understanding, the Chambers of Commerce will provide a registration service for organizations. The organization desiring registration will be allocated a registration number, the so called EDI identifier, whose structure is as follows:

EDI identifier = [A, B, C, D, E]

where

| Part | Digits | Explanation |
|------|--------|---|
| A | 4 | International Code Designator value ICD ('0085') allocated by ISO to the national RA, used as a qualifier for the scheme. This value is harmonized with the International Code Qualifier ICQ ('085') allocated by UN/ECE to the Chambers of Commerce in Switzerland. This value is part of <registration authority identifier> |
| B | 3 | Numerical value allocated by the RA to the regional sub-authority This value is part of <registration authority identifier> |
| C | 6 | Numerical value allocated by the sub-authority to the registered organisation (mandatory part of the identifier). This value is the <registered organization identifier>. |
| D | 6 | Numerical value used by the registered organisation. This value is the free part of the EDI identifier. |
| E | 2 | Numerical check digits calculated by the registered organisation. |

The usage of part B is still under discussion among the Swiss Chambers of Commerce.

G.2 Procedures for Registration

This section explains the generic registration procedure for organizations, as defined by the Chambers of Commerce in the EDIRA Memorandum of Understanding.

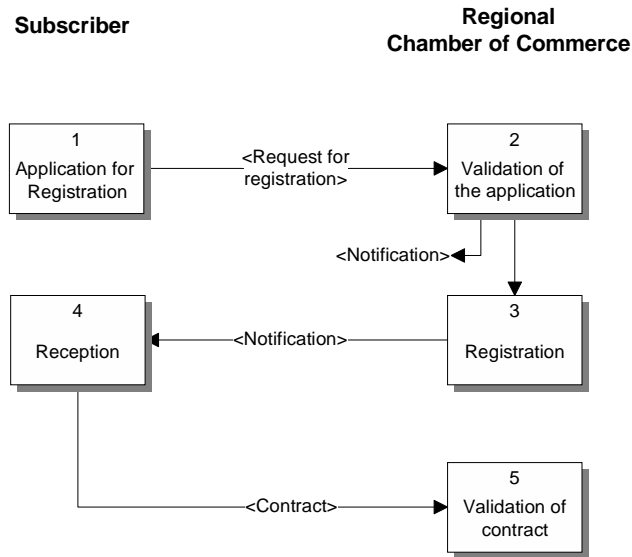


Figure G-1 Registration of organizations

I. Application for registration

- A. The subscriber submits a completed <Request for registration form> to the regional RA. Note: the form need not be legally signed and can be sent in by e-mail.

II. Validation of the application

- A. The registration officer checks that all the mandatory and all the optional information is supplied on the <Request for registration form> form in the correct syntax.
- B. The registration officer checks the existence of the organization using the 'Ragionenbuch' and validates the telephone number using the electronic telephone directory of the PTT as a reference.
- C. If any of the results of the checks are negative, the registration officer notifies the person making the application, using a telephone or a letter.

III. Registration

- A. The regional registration officer assigns the <registered organization identifier> to the organization.
Note: The EDI identifier comprises the <registration authority identifier> and the <registered organization identifier>.
- B. The regional registration officer completes the register entry in the <registration information base> and files the <Request for registration form>.
- C. The regional registration officer notifies the EDI-SW provider specified in the <Request for registration form>.
- D. The registration officer prints a <Confirmation of registration form>, signs it and sends it to the subscriber. The confirmation contains a note that the registration will only attain legal validity when the subscriber has sent in the signed contract.

IV. Reception of confirmation

- A. The subscriber receives the <confirmation of registration form>, makes the assigned EDI identifier known to the relevant parties within the organization and files it securely.
- B. The subscriber signs the contract and sends it back to the RA.
Note: Registration of organizational units, application processes and public keys is part of the Initialization Mechanism. The step of sending back the contract can directly be combined with the first step of the Initialization Mechanism.

V. Validation of contract

- A. Upon reception of the contract from the subscriber, the regional registration officer checks the validity of the signature(s) supplied on the contract. Then the contract is filed securely.

G.2.1 Request for registration form

The following table shows the content of the <request for registration form>. The data fields with an 'M' are mandatory, the data fields with an 'O' are optional.

| Data field | M/O | Comment |
|--|-----|---------------------------------------|
| Number of form | M | Used for reference |
| Originating organization details | M | |
| • Name of organization | M | Official name |
| • Trade name of organization | O | Optional trade name |
| • Postal address | M | |
| Details of the person making the application | M | |
| • Name of person | M | |
| • Organizational title of person | M | |
| • Postal address of person | M | |
| • Telephone number | M | |
| • Facsimile number | O | |
| • Primary e-mail address | O | |
| SW provider | M | desired or existing EDI-SW provider |
| Date of application | M | |
| Registration validity period | O | |
| Authorization for publication | O | |
| Interchange agreement | O | Reference to an interchange agreement |
| Authentication method used or credentials supplied | O | |
| Signature | O | |

G.2.2 Credentials

A central requirement for registration is the validation of credentials presented by one or more persons who can act on behalf of that organisation. The type of required credentials for the case, where organisations are already registered in the commercial register, and the case, where organisations (legal or natural persons) are not registered in the commercial register depend on national laws and, possibly, banking requirements and are to be defined by the Chambers of Commerce.

G.3 Additional procedures

In addition to the basic registration procedure for new organisations (described above), a set of procedures for maintenance of the <registration information base> is required, including the following:

1. procedures for modification of a register entry
2. procedures for deactivation of a register entry
3. procedures for the handling of register enquiries