Security requirement

Cryptographic Algorithms and Security Protocols

Deutsche Telekom Group

Version	44 (internal)
Date	Nov 21, 2023
Status	In work

Publication Details

Published by
Deutsche Telekom AG
Vorstandsbereich Technology & Innovation
Chief Security Officer

Reuterstrasse 65, 53315 Bonn
Germany

File name	Document number	Document type
	3.50	Security requirement

Version	State	Status
44 (internal)	Nov 21, 2023	In work

Contact	Validity	Released by
Telekom Security psa.telekom.de		

Summary
Cryptographic Algorithms and Security Protocols

Table of Contents

1. Introduction

1.1. General Introduction

This document has been prepared based on the provisions of the Group Security Policy.

The security requirement document is used as a basis for an approval in the PSA process, among other things. It also serves as an implementation standard for provisions of the Group Security Policy in units which do not participate in the PSA process.

These requirements shall be considered from the very beginning, including during the planning and decision-making processes.
When implementing these security requirements, the precedence of national, international, and supranational law shall be observed.

If compliance with the described requirements cannot be achieved or is only partially feasible in individual cases, a risk assessment must be carried out together with a Security and/or Data Privacy Expert (in accordance with the relevant requirement) and possible alternative protective measures must be agreed upon.

1.2. Remarks

This document is a general base for the work with cryptographic algorithms. This means that not every requirement is applicable if implementation-specific requirements regarding the cryptographic algorithms are already in place. All recommended algorithms are ordered from the most secure variant to the lowest one. The labels of the algorithms within this document are following the notation of the referenced standards.

The document contains requirements which are addressed to developers of cryptographic systems. But it also contains requirements addressed to developers who uses complete cryptographic modules and components to secure the system and interfaces.

Cryptographic algorithms for mobile network are out of scope. Those requirements are already covered by the 3rd Generation Partnership Project (3GPP) and the European Telecommunication Standardization Institution (ETSI).

1.3. User Roles

We created user roles and added this information to the long text of each requirement to determinate the audience of the requirement.

Following definitions have been introduced:

Development:
This user role includes all activities related to the direct development of crypto systems.
A typical task is to write own programming code and/or writing own crypto libraries (e.g. creating an own C-implementation of TLS and respective crypto primitives).

Integration:
This user role includes all activities in terms of integration of existing crypto-modules or libraries and embedding this in a specific application (e.g. include OpenSSL library into an application code in order to encrypt a network interface).

Operations:
This user role includes all activities to operate a cryptographic system. Certificate management, key management, and hardening of crypto applications belong to this user role (e.g. hardening of a SSH daemon).

2. General Principles for Cryptographic Algorithms

Req 1	Only standardized cryptographic algorithms and primitives which are published by accredited organizations may be used.

User roles: Development, Integration

Each deviation must be aligned with the security organization of Deutsche Telekom Group. All requirements fulfill (based on the publication date of this document) the current standards of the following organizations:

- German Information Security Agency - Bundesamt für Sicherheit in der Informationstechnik (BSI) [BSI]
- European Telecommunication Standards Institute (ETSI) [ETSI]
- Senior Officials Group Information Systems Security (SOGIS) [SOGIS]
- National Institute of Standards and Technology (NIST) [NIST]
- International Organization for Standardization (ISO) [ISO]
- Internet Engineering Task Force (IETF) [IETF]

References:
[BSI] https://www.bsi.bund.de/DE/Home/home_node.html (last view: August 4th, 2023)
[ETSI] https://www.etsi.org/ (last view: August 4th, 2023)
[SOGIS] https://www.sogis.eu/ (last view: August 4th, 2023)
[NIST] https://www.nist.gov/ (last view: August 4th, 2023)
[ISO] https://www.iso.org/home.html (last view: August 4th, 2023)
[IETF] https://www.ietf.org/ (last view: August 4th, 2023)

Motivation: Only well analyzed cryptographic algorithms have a sufficient systematic protection against cryptographic attacks. It shall be ensured that the recommendations are based on a common functional consent.

Implementation example: Standards of ISO/IEC JTC 1/SC 27 "IT Security techniques", BSI Technical Recquirements for SSH implementations.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-1/i44

Req 2	Well-established and up-to-date crypto libraries must be used to implement cryptographic algorithms.

User roles: Development, Integration

The use of self-implemented cryptographic systems shall be avoided. Instead, well-established crypto libraries which implement the approved algorithm set of this document must be used. If customized crypto implementations are required, they must follow best practices and must be supervised by the security organization of Deutsche Telekom group.

Motivation: Only well-known crypto libraries have been analyzed by security researchers and provide sufficient protec-

tion. Those libraries will be updated regularly (e.g. in case of vulnerabilities).

Implementation example: Botan, Bouncy Castle, GnuTLS, OpenSSL, wolfCrypt

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-2/i44

Req 3 Cryptographic systems must be implemented in replaceable modules.

User roles: Development, Integration

Static implementations complicate corrections and replacements of faulty implementations. Especially in case of unexpected security incidents or future computer architectures (e.g. quantum computing) a fast replacement of cryptographic modules might be necessary.

Motivation: Cryptographic implementation shall be fully replaceable without any dependencies to the software environment.

Implementation example: Replacement of a crypto library within a web application.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-3/i44

Req 4 The application must be able to configure cryptographic systems and must provide swap functions.

User role: Development

Deactivation and modification functions (e.g. cipher suites modifications) must be implemented during the development of cryptographic systems. Swap functions (i.e. re-encryption so that the ciphertext uses a different cryptographic function) will only be applied on persistent data storage, but not on transport encryption.

Unexpected events like new attacks or future computing architectures (e.g. universal quantum computers), might cause the need for an immediate substitution of broken schemes.

Motivation: Crypto management is a mandatory prerequisite for crypto agility. It allows to reduce risk and effort in case of urgent system changes in a running system.

Implementation example: Change the encryption cipher from AES-128-CBC to AES-256-GCM.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-4/i44

3. Symmetric Encryption Algorithms

Req 5	Following symmetric block ciphers must be used: AES-256, AES-192 or AES-128.

User roles: Operation, Development, Integration

The "Advanced Encryption Standard" (AES) has been standardized in 2001 as a variant of the "Rijndael"-algorithm. AES has been analyzed intensively since 2001. The algorithm is standardized by ISO/IEC 18033-3 and by Federal Information Processing Standard (FIPS) 197. AES is a block cipher with a specified input block length of 128 bit and supports three key sizes (128 bit, 192 bit and 256 bit). AES-128, AES-192 and AES-256 denoting the respective key lengths. AES-256 can be used to prevent (theoretical) attacks by quantum computers (Grover's algorithm).

Remark on data privacy:
All requirements in this document are limited to data and information security in terms of confidentiality and integrity. Requirements on cryptographic methods to fulfill requirements on anonymization and pseudonymization with regards to data privacy are published in document “Anonymization and Pseudonymization”

Motivation: The AES security has been analyzed since almost two decades and provides quantum-safe encryption by using the 256 bit key variant of AES.

Implementation example: A webserver uses a cipher suite with AES to ensure a client TLS-session:
`TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-5/i44

Req 6	Following Authenticated Encryption with Associated Data (AEAD) must be used as symmetric block cipher mode of operation: Galois-Counter-Mode (GCM), Counter with CBC-MAC (CCM).

User roles: Operation, Development, Integration

If the input data is longer than the AES input block length of 128 bit, an operation mode must be used to encrypt all the data.
GCM and CCM are Authentication Encryption with Associated Data (AEAD) modes of operation and providing confidentiality and integrity to the plain text. AEAD also provides integrity of associated data like protocol header information.
GCM and CCM are specified by ISO/IEC 19772, NIST SP 800-38C and NIST SP 800-38D standards. AEAD ciphers should be preferred, as far as possible.
The classical modes CBC, CTR and OFB only provide confidentiality. The following table represents the valid applications and operation conditions for the operation modes:

Operation Mode	**Confidentiality**	**Integrity**
GCM	Yes	Yes
CCM	Yes	Yes
CBC	Yes	No
CTR	Yes	No
OFB	Yes	No

Requirements on AEAD modes:

- Requirements on Galois-Counter-Mode (GCM):
    - The authentication tag must be at least 96 bit long
    - The initialization vector (nonce) must not repeat and shall be at least 96 bit long
- Requirements on Counter Mode with CBC-MAC (CCM):
    - The authentication tag must be at least 96 bit long
    - The initialization vector (nonce) must not repeat with a same key

Remark on data privacy:
All requirements in this document are limited to data and information security in terms of confidentiality and integrity. Requirements on cryptographic methods to fulfill requirements on anonymization and pseudonymization with regards to data privacy are published in document “Anonymization and Pseudonymization”.

Motivation: Wrong implemented mode of operation may cause a weakening of cryptographic systems.

Implementation example: Usage of AES-128-GCM as AEAD cipher in TLSv1.3.

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Unauthorized modification of data
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-6/i44

Req 7	If Authenticated Encryption with AEAD modes of operation is not available, one of the following confidentiality modes of operation must be used: Counter Mode (CTR), Cipher-Block Chaining (CBC) or Output Feedback (OFB).

User roles: Operation, Development, Integration

If the input data is longer than the AES input block length of 128 bit, an operation mode must be used to encrypt all the data.

AEAD ciphers should be preferred, as far as possible.
The classical modes CBC, CTR and OFB only provide confidentiality.
Therefore, they must generally be combined with mechanisms for data authentication (e.g. message authentication codes (MAC)). The mentioned "classical" confidentiality modes are specified by ISO/IEC 10116, or NIST SP 800-38A.

Requirements on confidentiality modes (refer to annex B of ISO/IEC 10116):

- Requirements on Counter Mode (CTR):
    - The counter and initialization vector (IV) are 128 bit long
    - The counter must not repeat with the same key
    - The chosen initialization vector must ensure that the counter will not repeat with the same key
- Requirements on Cipher Block Chaining Mode (CBC):
    - The initialization vector (IV) is 128 bit long and must be randomly chosen
    - The initialization vector (IV) must not be impactable or derivable
    - A secure padding scheme have to be used according to requirement 8
- Requirements on Output Feedback Modus (OFB):
    - The initialization vector (IV) is 128 bit long and must be randomly chosen

Remark on data privacy:
All requirements in this document are limited to data and information security in terms of confidentiality and integrity. Requirements on cryptographic methods to fulfill requirements on anonymization and pseudonymization with regards to data privacy are published in document “Anonymization and Pseudonymization”.

Motivation: Less number implementation failures.

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data

For this requirement the following warranty objectives are relevant:

ID: 3.50-7/i44

Req 8	The following padding procedures must be applied in case of CBC mode of operation: ISO, CMS, ESP or Ciphertext Stealing.

User role: Development

Padding procedures must be applied (to fill up with missing bits) if the length of the input data block is smaller than the multiple of the fixed 128 bit input block length in case of a symmetric block cipher (e.g. AES).

Remark Ciphertext Stealing:
Padding is made unnecessary by the help of the special operation mode "CBC with Ciphertext stealing" (CBC-CS). Expansion of the key text will be avoided by filling up with cipher text.

Specification Sources:
- ISO-Padding [ISO/IEC 7816-4]
- CMS-Padding [IETF-RFC 5652]
- ESP-Padding [IETF-RC 4303]
- Ciphertext Stealing CBC-CS1, CBC-CS2, CBC-CS3 [NIST-SP 800-38A - Addendum]

Motivation: The recommended padding procedures have a high dissemination and don't have known vulnerabilities when used together with CBC.

Implementation example: A 200 bit long data block shall be encrypted by AES-CBC. The length of the input block doesn't match with the multiple of 128 bit (128 bit, 256 bit, 384 bit,...).
56 bits are filled up to the next multiple (in this case: 256 bit) with ISO-Padding.

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-8/i44

4. Hash Functions

Req 9	Cryptographic hash functions of the SHA-3 and SHA-2 family must be used with a minimum output length of 256 bit.

User roles: Operation, Development, Integration

The security of hash functions is driven by their output lengths, which are denoted as suffix (e.g. SHA-3-512 has an output length of 512 bit)

- Standardized variants of the SHA-2 family are [ISO/IEC 10118-3, FIPS 180-4]: SHA-512, SHA-384, SHA-256.
- Standardized variants of the SHA-3 family are [ISO/IEC 10118-3, FIPS 202]: SHA-3-512, SHA-3-384, SHA-3-256.

Following hash applications are valid:

- Digital signatures
- Message Authentication Codes (MAC)
- Mask generation functions (MGF)
- Key Derivation Function (KDF) and Password-Based Key Derivation Functions (PBKDF)
- Pseudo Random Number Generators (PRNG)
- Integrity protection

Remark on authenticity and confidentiality:
Hash function don't provide confidentiality in terms of encryption and no protection against manipulation (active attacks). Hash functions are public known keyless one-way functions. An attacker can easily manipulate messages and calculate a valid hash of the modified message. If authenticity of a message must be guaranteed, a keyed Message Authentication Code (e.g. HMAC) must be used.

Remark on SHA-3-224/SHA-224:
SHA-224/SHA-3-224 may only be used for Message Authentication Codes (MAC) and as primitive of Key Derivation Functions (KDF) for legacy systems and should be replaced by stronger hash functions with a longer output length (>=256 bit).

Reference:
[TR2102-1] BSI-Technical Guideline: Cryptographic Mechanisms - Recommendations and Key Lengths, chapter 5 "Hash Functions", Version 2023-01

Motivation: Cryptographic hash functions are used in many use cases and are basis of several cryptographic schemes, like digital signatures.

Implementation example: Usage of a hash function for a digital signature of a certificate.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-9/i44

5. Message Authentication Codes (MAC)

Req 10	The following Message Authentication Codes (MAC) must be used: HMAC (based on SHA-2/3 hash functions) or CMAC (based on AES).

User roles: Operation, Development, Integration

Message Authentication Codes (MAC) protect the integrity of messages in security protocols like IPSec or TLS/SSL. Moreover, MACs are used in other cryptographic applications, such as Key Derivation Functions (KDF).

The following table represents the valid algorithms and parameters:

MAC	**Approved Algroithms**	**Special Operation Conditions / Remarks**
HMAC [ISO/IEC 9797-2, FIPS 198]	SHA-512 SHA-3-512 SHA-384 SHA-3-384 SHA-256 SHA-3-256	The key lengths must be at least 128 bit long. The length of the MAC must be at least 64 bit long, however it is recommended to use MACs not shorter than 96 bit.
CMAC [ISO/IEC 9797-2, NIST SP 800-38B]	AES-256 AES-192 AES-128	The key length will be defined by the chosen AES variant (128/192/256 bit). The length of the MAC must be at least 64 bit long, however it is recommended to use MACs not shorter than 96 bit.

Remark on CMAC:
Simple implementations of CBC-MAC are not resistant against extension attacks. One in [ISO/IEC 9797] or [NIST SP 800-38B] standardized secure variant of CBC-MAC is denoted as CMAC (MAC Algorithm 5 in [ISO/IEC 9797]).

Remark on GMAC:
The authenticated encryption mode GCM is the underlying basis of GMAC constructions. GMAC is a standalone MAC (not part of GCM) and only has a similar security as HMAC and CMAC in particular cases [BSI TR-02102-1]. Thus, GMAC is not recommended as MAC algorithm.

Restrictions on SHA-224/SHA-3-224/SHA-1:
The usage of SHA-224/SHA-3-224/SHA-1 as HMAC is only permitted for legacy systems and should be replaced by stronger hash algorithms which creating outputs not less than 256 bit.

Remark on data privacy:
All requirements in this document are limited to data and information security in terms of confidentiality and integrity. Requirements on cryptographic methods to fulfill requirements on anonymization and pseudonymization with regards to data privacy are published in document “Anonymization and Pseudonymization”.

Motivation: Message Authentication Codes (MAC) protect the integrity of messages in security protocols like IPSec or TLS/SSL. Moreover, MACs are used in other cryptographic applications, such as Key Derivation Functions (KDF).

Implementation example: Integrity protection of billing data records by HMAC-SHA-256.

For this requirement the following threats are relevant:
- Unauthorized modification of data
- Unnoticeable feasible attacks

For this requirement the following warranty objectives are relevant:

ID: 3.50-10/i44

Req 11	If confidentiality modes of operation (CTR, CBC, OFB) are used, the Message Authentication Code must be generated on the cipher text (“encrypt-then-MAC”) in order to protect the integrity of the messages.

User role: Development

If no AEAD operation mode is used, a Message Authentication Code must be used to ensure the authenticity of the transferred data. The generic and secure combination of encryption and MAC is called "Encrypt-then-MAC". This means, that the message is encrypted fist, and afterwards the MAC is generated on the cipher text. Other implementations such as "MAC-then-Encrypt" are often unsecure and therefore not recommended.

Remark on standardized protocols:
Standardized security protocols such as SSH, SSL/TLS or IPSec use different generic combinations of encryption and MAC. Therefore, several attacks have been published in the past which caused protocol design changes. Requirements on security protocols are in separate sections of this document. It is recommended to use dedicated Authenticated Encryption constructions.

Motivation: Prevention of design weaknesses in encryption systems.

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-11/i44

6. Key Management

6.1. Key creation

Req 12	Physical random number generators or following strong pseudo-random number generators (deterministic sources) with sufficient entropy must be used to generate cryptographic keys for algorithms: HMAC-DRBG, Hash-DRBG or CTR-DRBG.

User roles: Development, Integration

HMAC-DRBG, Hash-DRBG or CTR-DRBG are based on the specifications of NIST SP800-90A and ISO18031. If a pseudo-random number generator (deterministic random bit generator = DRBG) is used to generate a cryptographic key, a non-deterministic value must be used to calculate the seed value.
This non-deterministic source must have a sufficient high entropy.

Motivation: If output values of a random number generator occur with different probabilities, then it can be exploited to deduce or derive a key through statistical methods (stochastic analysis).

Implementation example: Usage of the random function in Linux `/dev/urandom` to generate a seed value for a pseudo-random number generator.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Denial of executed activities
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-12/i44

Req 13	To generate keys for applications with high security requirements, a hardware security module (HSM) or a secure element (e.g. Smart Card) with a physical random number generator must be used.

User role: Development

Generated keys must not leave the hardware security module unprotected.

Motivation: Tamper-proof HSMs or Smart Cards provide a secure storage for secret keys. HSMs ensure a defined access to the stored keys. They also provide protection against a disclosure of cryptographic keys.

Implementation example: - Generation of randomly chosen subscriber keys within SIM production process by a physical random number Generator running in an HSM.
- Utimaco Enterprise Secure Key Manager

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Denial of executed activities
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-13/i44

6.2. Key Agreement and Exchange

Req 14	Key agreement protocols must fulfill Perfect Forward Secrecy (PFS) properties and must be protected against Man-in-the-middle attacks (e.g. Authenticated Ephemeral Elliptic Curve Diffie Hellman).

User role: Development

Today's asymmetric key agreements are based on (Elliptic Curve) Diffie-Hellman key exchange (e.g. Ephemeral EC-DH or MQV). The advantage of this type of key exchange is that the complete key will never be transferred over the communication channel. Ephemeral key exchange means that the session secrets will be deleted after usage. Perfect Forward Secrecy (PFS) means that a recorded ciphertext cannot be decrypted afterwards, regardless of whether the long-term key (e.g. private RSA key) is known by an attacker. There are different key exchange protocols standardized in ISO/IEC11770-3. Each development of PFS key exchange protocols shall follow BSI requirements [BSI TR 02102-1].

Remarks on standardized DH groups:
The generation of secure system parameter (generators, and finite groups) is a complex, sensitive and sophisticated task. As a result of this, developers are often using standardized system parameters [RFC5114]. The minimum lengths or groups and parameter (see chapter 13) must be considered, as they are not part of the IANA numbering of the DH groups.

Motivation: The key agreement solves the key exchange problem of symmetric cryptography. With Perfect Forward Secrecy, the attacker is not able to break the confidentiality of transferred cryptograms if the long-term key (RSA private key) has been compromised. Man-in-the-middle attacks will be avoided by using certificate-based authentication of the peers.

Implementation example: One example is the secure key agreement by using Diffie-Hellman key exchange in TLS. The server will be authenticated by a TLS-certificated issued by a trustworthy CA. Client authentication can also be achieved by using certificates or even at a later stage with username and password authentication over a secured channel. Since TLSv1.3 PFS is mandatory, so forward Secrecy is supported.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Denial of executed activities
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-14/i44

6.3. Asymmetric Key Encapsulation Mechanisms (KEM)

Req 15	If none key exchange protocols with Perfect Forward Secrecy (PFS) are available one of the following Key Encapsulation Mechanism (KEM) must be used to encrypt symmetric keys: ECIES-KEM, PSEC-KEM, RSA-OAEP, DLIES-KEM, or RSA-KEM.

User roles: Development, Integration

Hybrid encryption schemes are used to transfer symmetric keys over an unsecure channel. The actual encryption of the user data will be done via symmetric ciphers (e.g. AES). It is not recommended to encrypt large amounts of user data (bulk) with asymmetric algorithms.

ECIES (Elliptic Curve Integrated Encryption Scheme) and DLIES (Discrete Logarithm Integrated Encryption Scheme) are hybrid encryption methods that combine an asymmetric KEM, e.g. ECIES-KEM, with a symmetric encryption method or authenticated encryption, e.g. AES-GCM.

The following table represents the valid asymmetric algorithms plus the minimum length of security parameters and further system parameters:

Asymmetric algorithm	**Minimum length of the security parameter**	**Symmetric cipher**	**Additional functions**
ECIES-KEM (on elliptic curves)	250 bit (order of the base point on an elliptic curve must be at least 250)	AES-GCM	KDF: HMAC based KDF
RSA-OAEP	3000 bit RSA modulus (n) with a public key e > 2^16	AES-GCM	Hash: SHA3, SHA2 KDF: HMAC based KDF
PSEC-KEM (on elliptic curves)	ord(g) >= 250 bit (order of the base point on an elliptic curve must be at least 250)	AES-GCM	KDF: HMAC based KDF
DLIES (on GF(p))	3000 bit prime p (defines the finite field), 250 bit prime q	AES-GCM	KDF: HMAC based KDF
RSA-KEM	3000 bit RSA modulus (n) with a public key e > 2^16	AES-GCM	KDF: HMAC based KDF

Further information on the security parameter set of asymmetric algorithms can be found in [BSI TR-02102-1]. The Key Encapsulation Mechanisms (KEM) ECIES-KEM, RSA-OAEP, RSA-KEM, PSEC-KEM, are standardized by e.g. ISO/IEC 18033-2.

Remark on used symmetric encryption cipher:
It is recommended to use authenticated encryption (e.g. AES-GCM, AES-CCM). If no AEAD encryption mode is available, the integrity and authenticity of the encryption must be additionally protected by (complementary) message authentication codes (MAC) in an "Encrypt-then-MAC" construction.

Remark on RSA-PKCS#1 v1.5:
RSA PKCS#1 v1.5 is not considered secure anymore. Thus RSA-PKCS#1 v1.5 must not be used for encryption.

The following curves parameter may be used and (according to the status in research) provide the security level required in the above table:

- brainpoolP512r1
- NIST P-521
- brainpoolP384r1
- NIST P-384
- brainpoolP320r1
- brainpoolP256r1
- NIST P-256

Abbreviations:
ECIES = Elliptic Curve Integrated Encryption Scheme
KEM = Key Encapsulation Mechanism
OAEP=Optimal Asymmetric Encryption Padding
PSEC = Provable Secure Elliptic Curve (encryption)
RSA = Rivest-Shamir-Adleman
DLIES = Discrete Logarithm Integrated Encryption Scheme

Motivation: The algorithms provide a sufficient protection and were recommended by acknowledged institutions. Hybrid encryption solves the key exchange problem of the symmetric cryptography and is used in email encryption scenarios.

Implementation example: Encryption of the AES key via RSA public key of the receipt: RSA-KEM_PubKey_B (AES_Key).

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-15/i44

6.4. Key Derivation

Req 16	The following primitives for standardized Key Derivation Functions (KDF) must be used to derivate cryptographic session keys: HMAC, CMAC, AES, SHA-3, SHA-2.

User role: Development

During the usage of a common secret (e.g. after a DH key exchange), normally several "sub-keys" are required for different purposes like session key, or MAC. Key Derivation Functions (KDF) are used to generate several derived keys for dedicated uses cases from a single master key.

Basically, KDFs are distinguished in Extraction-then-Expansion (two-step KDFs according to [ISO/IEC 11770-6] based on HMAC or CMAC and pseudorandom functions (one-step KDFs according to [NIST SP 800-56C], [NIST SP 800-108], or [ISO/IEC 11770-6]).

Use cases of Key Derivation Functions:

- Binding of protocol data with a key (e.g. sender name, receiver name) by „Labels"
- Derivation of symmetric key for dedicated purposes (MAC, encryption)
- Enhancement of secrets with additional entropy as symmetric key (Extraction-then-Expansion)

Motivation: Key Derivation Functions that were well investigated and recommended by acknowledged institutions provide sufficient protection and ensure a comparable cryptographic strength of the derived keys.

Implementation example: Generation of an HMAC key and encryption key after a Diffie-Hellman key exchange by means of an HMAC-SHA-256 based Extraction-then-Expansion KDF.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Denial of executed activities
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-16/i44

Req 17	If a cryptographic key is derived from a password the following algorithms must be used: Argon2 or scrypt [RFC 7914]. If Argon2 and scrypt [RFC 7914] are not available, PBKDF2 [RFC 8018] may be used.

User roles: Operation, Development, Integration

Following parameter set must be implemented:

Algorithm	**Maximum password length**	**Security Parameter / Work Factor**	**Salt Length**
Argon2 [1]	80 characters	n=4,096; m=9, p=1	128 bit
scrypt [RFC7914]	80 characters	N=2^18; r=14; p=4	128 bit
PBKDF2 [RFC8018]	80 characters	n=5,120,000	128 bit

PBKDF2 is to be classified as cryptographically weaker.

The salt is a random number (together with the password) which is part of the key derivation algorithm input. The main purpose of a salt is to keep public Rainbow Table useless.

Definition of salt:
In cryptography, a salt is random data that is used as an additional input to a one-way function that "hashes" data, a password or passphrase. Salts are used to safeguard passwords in storage. Historically a password was stored in plaintext on a system, but over time additional safeguards developed to protect a user's password against being read from the system. A salt is one of those methods.[2]

Remark on passwords:
The current rule set on passwords can be found in the document “3.01 Technical Baseline NT/IT”.

References:
[1] A. Biryukov, D. Dinu, D. Khovratovich: Argon2: the memory-hard function for password hashing and other applications, 2017
[2] https://en.wikipedia.org/wiki/Salt_(cryptography) (last view: August 4th, 2023)
[RFC7914] C. Percival, Tarsnap, S. Josefsson, SJD AB: RFC 7914, The scrypt Password-Based Key Derivation Function, 2016
[RFC8018] K. Moriarty, Ed., Dell EMC, B. Kaliski, Verisign, A. Rusch, RSA: RFC 8018, PKCS #5: Password-Based Cryptography Specification Version 2.1, 2017

Motivation: Specialized password based KDF are designed to derive cryptographic keys from passwords with a sufficient entropy and security.

Implementation example: Derivation of a session key and MAC key from a password by executing the Argon2 algorithm.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-17/i44

6.5. Transport and Storage of Keys

Req 18	High-Security Applications must store keys in a Hardware Security Module (HSM) or in a Secure Element (e.g. Smart Card). Secret keys and secret parameter must not leave the hardware unprotected.

User role: Development

The usage of tamper-proof HSM allows maximum possible protection of key material. HSMs and Secure Elements are robust against side channel attacks.

Recommended methods to protect keys can be found in requirement 19.

Motivation: Cryptographic key must be protected against unauthorized access. Dedicated security hardware provides a very high level of security.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Denial of executed activities
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-18/i44

Req 19	Symmetric keys must be protected against unauthorized access and must neither transferred nor stored unencrypted outside the system boundaries.

User role: Development, Operations

Basic requirements on data access:
The data access on key material must be restricted to necessary system services and user groups inside and outside of the system boundaries.

Requirements on transport and storage of symmetric keys:
AES-Synthetic Initialization Vector (AES-SIV) or AES-Kexwrap must be used to transport and storage of symmetric keys (e.g. Pre-shared keys) outside the system boundaries. This requirement is applicable for the transport of cryptographic keys over an unsecure channel or storage of key material on an unprotected storage medium. The methods above ensuring confidentiality of the key and often called “key-wrap”. Both a specified in [RFC5297] and [SP800-38F]. The requirements for security parameters of these methods correspond to the requirements for AES from Req. 5.

Remark on secure Hardware:
Keys may also be stored and transported via secure hardware (e.g. HSM) or on secure elements (e.g. Smart Card).

Motivation: The security of a cryptographic system following Kerckhoffs’ law which says: “The key must be secret, but the cryptographic algorithms shall be public to ensure confidentiality” Therefore the key protection is very important in cryptography.

Implementation example: Key wrap of an AES-128 key (key length = 128 bit) by means of AES-SIV.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Denial of executed activities
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-19/i44

Req 20	Private keys of asymmetric cryptographic algorithms must be protected against unauthorized access and must not transferred or stored unencrypted outside system boundaries.

User roles: Operation, Integration

Basic requirements on data access:
The data access on key material must be restricted to necessary system services and user groups inside and outside of the system boundaries.
Private keys are used for authentication, decryption of encrypted messages and for digital signatures of messages. A digital signature (= comparable with a personal signature) can only be generated with a private key and ensures the authenticity. That is the reason why private keys may not be located in central storage systems nor be distributed. Private keys of individual persons must be protected against any misuse via a personal secret (PIN or Password) as well.

Remark on remote signatures:
Remote signatures may only be created by certified trust provided (trust center / trust services like www.telesec.de). They must reveal its compliance with the eIDAS standards, and the certificate must be verified and renewed on a regular basis.

Motivation: If an attacker gets access to a private key, they can impersonate themself as the original user / system.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Denial of executed activities
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-20/i44

Req 21	Cryptographic long-term keys must have a limited validity period that is appropriate for the application.

User roles: Operation, Development, Integration

In cryptography there are two types of key material. One type is long-term keys, and the other type is short-term keys. Long-term keys are used for a relatively long period of time (e.g. TLS server certificate keys, SSH keys). Short-term keys are only valid during a running protocol session and will be deleted after the session. The definition of Long-term keys is not related to a concrete timespan. It is defined as reusable key (e.g. reusing the same key for authentication or digital signatures). The limitation of the validity period mitigates the possibilities for cryptanalysis, the impact in case of compromise of the private key as well as the risk that the underlying algorithm or key length is classified as insecure during the validity period of the key.

The validity of asymmetric key pairs is limited by the validity period of the related digital certificate. It is important to note that a certificate renewal does not necessarily requires a key renewal (the policy of the issuing certification authority must be considered). In general, the overall validity of keys (possibly spread over several certificates) that are used for end applications should be between one and three years as those keys are used frequently.

In particular, controllable cases with rare key usage, such as the key of a certification authority, longer validity periods can be chosen. In case of emergency (e.g. key compromise) the key must be revoked early.

The following validity periods are recommended:

- End entities (e.g. user, TLS server): 1-3 years
- Subordinate/Intermediate Certification Authorities: 8-15 years
- Root Certification Authorities: 20-30 years

It may be necessary to use keys beyond its validity. This may be the case if public keys are required to validate signatures in long-term applications or if private keys are required for the decryption of long-term data such as e-mails.

Motivation: By choosing limited validation periods for asymmetric long-term keys the security of the used algorithms/key lengths can be adopted periodically.

Implementation example: TLS-Web Server certificate with a lifetime of 1 to 3 years.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-21/i44

Req 22 The security of cryptographic long-term keys must be checked regularly and adapted if necessary.

User roles: Operation, Development, Integration

Due to the increase in computing power and new attacks, the key lengths of the used cryptographic methods must be adjusted from time to time. In case of special progress in cryptanalysis (e.g. through universal quantum computers) even an adaption the cryptographic method may be necessary. This concerns in particular cryptographic long-term keys such as key pairs that were issued for digital certificates or SSH keys. At the latest at the renewal of the certificates it must be checked whether the used method and key length are still considered secure. For example, RSA must be used with at least 3072 bit prime, the usage of RSA with 2048 bit prime may only be used in legacy systems until the end of year 2025 ([1], [2]).

References:
[1] Bundesamt für Sicherheit in der Informationstechnik, TR-02102-3, Version 2023-01
[2] SOG-IS Crypto Evaluation Scheme: Agreed Cryptographic Mechanisms, v1.3, February 2023

Motivation: Advances in computing might require the adaptation of cryptographic key sizes.

Implementation example: Verify that the key length of long-time keys is still sufficient when renewing a certificate.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Denial of executed activities
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-22/i44

Req 23 Cryptographic keys (symmetric and asymmetric) must be renewable.

User role: Operation, Development

Active attacks and technical developments in computing technology are the main reasons to substitute key material, e.g. if the encryption system is upgraded from AES-128 to AES-256 in order to be quantum-resistant.

Moreover, the security of cryptographic systems relies on the secrecy of the key. Thus, the usage of static keys might be a high risk if the static keys become compromised.

Rekeying is a mandatory countermeasure which is implemented in modern encryption systems. Further information can be found in [ISO/IEC JTC1 SC27 SD 12].

Renewed keys must not be derived from active or previously used key material (e.g. session key). The operator of a system must define a suitable lifetime of the keys.

Motivation: The security of a cryptographic system relies on the secrecy and quality of the key material.

Implementation example: Definition of a key lifetime parameter of IPSec session keys in the system configuration of an IPSec gateway.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Denial of executed activities
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-23/i44

6.6. Key deactivation and destruction

Req 24	Compromised keys or certificates must be deactivatable / revocable.

User roles: Operation, Development, Integration

If a private key compromise is detected, the related digital certificate must be revoked immediately. It must no longer be possible to authenticate with the private key. The revocation procedures must be integrated into other regulating processes.

Motivation: If an attacker gets knowledge of an RSA private key, it is possible to impersonate the identity of the victim to use applications or functions on behalf of the victim's identity.

Implementation example: Automated process for blocking/revocation of compromised certificates by using Certificate Revocation List (CRL) as part of the Certificate Management Protocol (CMP).

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized use of services or resources

For this requirement the following warranty objectives are relevant:

ID: 3.50-24/i44

Req 25	Key material which is no more used must be destroyed or rendered useless immediately and irrecoverable by secure methods.

User role: Operation

Following electronic data erasure procedures are recommended:

- DoD-5220.22-M (ECE)
- Bruce-Schneier-algorithm

- Peter-Gutmann-algorithm
- ATA-"Secure-Erase" (SSD or SSHD disks)- all bits must be set to logical 1

Motivation: If an attacker gets knowledge of an "old" pre-shared key, it is possible to decrypt logged messages (messages which were encrypted with the "old" AES-Key) .

Implementation example: - Overwriting of the key storage via DoD-5220.22-M (ECE) algorithm- Revocation of a web-server certificate
- Automated process for blocking / revocation of unused certificates by using Certificate Revocation List (CRL) as part of the Certificate Management Protocol (CMP).

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Denial of executed activities
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-25/i44

7. Signature Algorithms

Req 26	The following signature algorithms must be used: EC-DSA, EdDSA, RSA-PSS, DSA/DSS.

User roles: Development, Integration
Signature algorithms are used to verify the authenticity of the sender and the integrity of a signed message. The verification is performed by means of the public key.

Signature algorithms with appendix (i.e. signature is attached to the message) always need a cryptographic hash function.

The following table represents the valid parameters for digital signatures:

Signature Algorithm	**Minimum Length of the security parameter**	**Permitted hash algorithms**
EC-DSA [ISO/IEC 14888-3], [FIPS 186-5]	250-bit (the order of the generator base point on the elliptic curve shall not be shorter than 250)	SHA-3, SHA-2 with an output length 256-bit
EdDSA [RFC 8032] [FIPS 186-5]	250 bit	SHAKE256 for Ed448, SHA-512 for Ed25519
RSA-PSS [ISO/IEC 14888-2], [FIPS 186-5]	3000-bit RSA modulus (n) with a public key e > 2^16	SHA-3, SHA-2 with an output length 256-bit
DSA / DSS [ISO/IEC 14888-3], [FIPS 186-5]	3000-bit prime p (defined on finite field); 250-bit prime q	SHA-3, SHA-2 with an output length 256-bit

Further Information on digital signatures can be found in [BSI TR-02102-1].

Remark on EdDSA:
EdDSA is not (explicitly) recommended by BSI, but no weaknesses are known so far, and therefore it is to be classified as secure.

Remark on DSA / DSS:
[FIPS 186-5] only allows this algorithm to check old signatures.

Restrictions on SHA-224/SHA-3-224:
SHA-224/SHA-3-224 may only be used in legacy Systems and have to be replaced by a stronger hash algorithm which has a Output length not less than 256 bit.

Restrictions on RSA PKCS#1 v1.5:
RSA PKCS#1 v1.5 may only be permitted on legacy Systems and should be replaced by state-of-the-art signature algorithms.

Approved elliptic curves:

- brainpoolP512r1
- NIST P-521
- brainpoolP384r1
- NIST P-384
- brainpoolP320r1
- brainpoolP256r1
- NIST P-256
- Curve25519/Ed25519

- Curve448/Ed448

Abbreviations:
DSA = Digital Signature Algorithm
DSS = Digital Signature Standard
EC = Elliptic Curve
PKCS = Public Key Cryptography Standards
PSS = Probabilistic Signature Scheme

Motivation: Secure digital signatures are the base of a unique (digital) identity.

Implementation example: x509 certificate mit following signature algorithm: `sha-256WithRSA`

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized modification of data
- Unauthorized use of services or resources
- Denial of executed activities

For this requirement the following warranty objectives are relevant:

ID: 3.50-26/i44

8. Cryptographic algorithms in security hardware

Req 27	Cryptographic algorithms used in applications with high security requirements must be implemented in a Hardware Security Module (HSM) or in a Secure Element (e.g. Smart Card).

User Role: Development

Security hardware is tailored to operate cryptographic operations and contain physical security and side-channel protection mechanism. Beside of key generation and management, HSMs or Secure Elements are capable to execute cryptographic operations used for application with high-security requirements.

Motivation: The use of tamper-proof devices (HSM) ensures that the session keys cannot be compromised under any circumstances. Furthermore, an HSM protects the cryptographic algorithm against side channel attacks.

Implementation example: Digital signature of certificates by a certification authority (CA) using private keys and signature algorithms stored and running on an HSM.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-27/i44

9. Cryptographic Algorithms for Secure Password Storage and Verification

Req 28 User roles: Operation, Development, Integration

The following cryptographic algorithms for generation of pasword hashes and verification of passwords may only be used: scrypt [RFC 7914], bcrypt or Argon2. If scrypt [RFC 7914], bcrypt and Argon2 are not available, SHA-512-crypt / Crypt(3), SHA-256-crypt / Crypt(3) or PBKDF2 [RFC8018] may be used.

Passwords used for authentication must always be stored as a password hash.

Secure algorithms for password hash generation and password verification:

Algorithm	**Maximum Password Length**	**Salt Length**	**Security Parameter / Work Factor**	**Additional Parameter**
scrypt	80 characters	128 bit	N=2^18; r=4; p=1	none
bcrypt	50 characters	128 bit	2^n with n=13	none
Argon2	80 characters	128 bit	n=256; 2^m with m=10, p=1	none
SHA-512-crypt / Crypt(3)	80 characters	96 bit (16 characters)	n=640.000	id=6
SHA-256-crypt / Crypt(3)	80 characters	96 bit (16 characters)	n=640.000	id=5
PBKDF2	80 characters	128 bit	n=640.000	none

SHA-512-crypt / Crypt(3), SHA-256-crypt / Crypt(3) and PBKDF2 [RFC8018] are to be classified as cryptographically weaker.

Remark on passwords:
The current rule set on passwords can be found in the document “3.01 Technical Baseline NT/IT”.

Remark on the security parameter:
All parameter sets are defined according to security best practices in order to be resistant against offline attacks by speed down the hash generation rate.

Remark on SHA-512/256-crypt:
Generic hash function must not be used for password hashing, due to the fact that those are not resistant against special attack methods (GPU, ASIC/FPGA) which can calculate a large number of hash values per second. Thus SHA-512-crypt is a specific algorithm based on SHA-512 to compute and verify password hashes.

Remark on key derivation:
Recommendations on password-based key derivations can be found in section "Key Management"

References:
[1] https://pthree.org/2016/06/28/lets-talk-password-hashing/ (last view: August 4th, 2023)
[2] https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet (last view: August 4th, 2023)
[3] https://veggiespam.com/painless-password-hash-upgrades/ (last view: August 4th, 2023)
[RFC7914] C. Percival, Tarsnap, S. Josefsson, SJD AB: RFC 7914, The scrypt Password-Based Key Derivation Function, 2016

[RFC8018] K. Moriarty, Ed., Dell EMC, B. Kaliski, Verisign, A. Rusch, RSA: RFC 8018, PKCS #5: Password-Based Cryptography Specification Version 2.1, 2017
[TBS] Deutsche Telekom AG: Technical Baseline Security for IT-/NT-Systems, version 6.0

Motivation: Only special password hash algorithms are resistant against online and offline attacks on password hash tables.

Implementation example: Python:

```
>>> from passlib.hash import sha256_crypt

>>> # generate new salt, hash password
>>> hash = sha256_crypt.hash("password")
>>> hash
'$5$rounds=80000$wnsT7Yr92oJoP28r$cKhJImk5mfuSKV9b3mumNzlbstFUplKtQXXMo4G6Ep5'

>>> # same, but with explict number of rounds
>>> sha256_crypt.using(rounds=640000).hash("password")
'$5$rounds=640000$q3hvJE5mn5jKRsW.$BbbYTFiaImz9rTy03GGi.Jf9YY5bmxN0LU3p3uI1iUB'

>>> # verify password
>>> sha256_crypt.verify("password", hash)
True
>>> sha256_crypt.verify("letmein", hash)
False
```

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unnoticeable feasible attacks

For this requirement the following warranty objectives are relevant:

ID: 3.50-28/i44

10. IP Security (IPSec)

If requirements for IP Security (IPSec) are needed, then only chapter 10 is applicable.

10.1. General IPSec Parameters

Req 29 IPSec must be used with Internet Key Exchange (IKE) Version 2.

User roles: Operation, Development, Integration

IPSec (Internet Protocol Security) is a protocol designed to protect the confidentiality, integrity, and authenticity of information transmitted over the IP protocol. A distinction is made between the following IPSec protocols:

- Authentication Header (AH): Protection of integrity and authenticity
- Encapsulated Security Payload (ESP): additional protection of confidentiality

An essential concept of IPSec is the Security Associations (SA). An SA represents a secure connection between two communication partners and includes the cryptographic algorithms and parameters used. Security associations are negotiated using the Internet Key Exchange (IKEv2) protocol. IPSec and IKEv2 are standardized in various RFCs, including RFC 4301, RFC 4302, RFC 4303, RFC 7296, and RFC 8247.

Motivation: IKEv2 has stronger Key Derivation Functions (KDF) in place. The tunnel establishment procedure for IKEv2 is faster as IKEv1 as well. IKEv1 is vulnerable against denial-of-service (DoS) attacks due to stateful session cookies [BSI TR-02102-3].

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Denial of executed activities
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-29/i44

Req 30 Secure encryption methods according to the table below must be used for IKEv2 and IPSec.

User roles: Operation, Development, Integration

In IKEv2 the encryption method is used for the encryption of the IKE messages. These messages serve for the negotiation of the cryptographic algorithms and parameters used in the IPSec protocol. In the IPSec protocol ESP the encryption method is used to ensure the confidentiality of the ESP packets.

The following table lists the allowed encryption methods as well as the reference specification for the usage in IKEv2 and in the IPSec protocol ESP. Allowed encryption methods for IKEv2 and ESP:

Encryption method	**Reference specification for IKEv2**	**Reference specification for IPSec (ESP)**
ENCR_AES_GCM_16	RFC 5282, RFC 8247	RFC 4106, RFC 8247
ENCR_AES_GCM_12	RFC 5282, RFC 8247	RFC 4106, RFC 8247
ENCR_CHACHA20_POLY1305	RFC 7634	RFC 7634
ENCR_AES_CCM_16	RFC 5282	RFC 4309
ENCR_AES_CCM_12	RFC 5282	RFC 4309

ENCR_AES_CTR	RFC 5930	RFC 3686
ENCR_AES_CBC	RFC 7296	RFC 3602

Remarks:

- The encryption algorithm AES can be used with three key lengths (256, 192 or 128 bit). The key length can be negotiated by means of the key length attribute. It is important to take into account which key lengths the corresponding implementation offers, whereby all three key lengths are secure and can be used.
- The methods ENCR_AES_CTR and ENCR_AES_CBC must be combined with a method for integrity protection
- The suffixes _16 and _12 for the methods ENCR_AES_GCM and ENCR_AES_CCM denote the length of the authentication tag (MAC) in bytes.

Reference:
[1] Bundesamt für Sicherheit in der Informationstechnik, TR-02102-3, Version 2023-01

Motivation: Strong encryption protects the confidentiality of transferred data.

Implementation example: Cisco configuration example for the encryption of ESP:

```
CE1_R1_4(config)#crypto ipsec transform-set test esp-aes
```

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-30/i44

Req 31 The following hash functions must be used for IPSec: SHA-3-512, SHA-512, SHA-3-384, SHA-384, SHA-3-256, or SHA-256.

User roles: Operation, Development, Integration

Hash functions are the base for Message Authentication Codes (MAC) to protect the integrity of IPSec messages as well as for pseudorandom functions for key derivation. Furthermore, they are used for the creation of digital signatures as part of the authentication Generally, the hash algorithm is specified implicitly by configuring the MAC, the PRF or the authentication mechanism. However, because some IPSec systems require the configuration of the hash algorithm, in these cases the hash algorithms specified in this requirement must be used.

Remark on SHA-1:
SHA-1 must not be used for digital signatures. For Message Authentication Codes or pseudorandom functions, it may only be used in legacy systems and must be substituted by a hash algorithm with a 256-bit output at the next opportunity.

Remark on SHA-3-224/SHA-224:
SHA-3-224/SHA-224 may only be used in legacy systems and must be substituted by a hash algorithm with a 256-bit output at the next opportunity.

Reference:
[1] Bundesamt für Sicherheit in der Informationstechnik, TR-02102-3, Version 2023-01

Motivation: Hash algorithms are used for different cryptographic mechanisms within IPSec. Therefore, they must provide a high security level.

Implementation example: Cisco configuration example for hash in phase 1:

```
CE1_R1_4(config-isakmp)#hash sha56
```

For this requirement the following threats are relevant:
- Unauthorized modification of data
- Denial of executed activities

For this requirement the following warranty objectives are relevant:

ID: 3.50-31/i44

Req 32	Secure Message Authentication Codes (MAC) according to the table below must be used for IKEv2 and IPSec.

User roles: Operation, Development, Integration

In IKEv2 Message Authentication Codes are used for integrity protection of the IKE messages. In IPSec they are used to ensure the integrity of the ESP or ASH packets.

The following table lists the allowed Message Authentication Codes as well as the reference specification for the usage in IKEv2 and in the IPSec protocols AH and ESP. A “-” in a table cell means that the respective methods is not intended for the corresponding protocol.

Allowed Message Authentication Codes for IKEv2 and IPSec:

MAC	**Reference specification for IKEv2**	**Reference specification for IPSec (AH, ESP)**
AUTH_HMAC_SHA2_512_256	RFC 4868	RFC 4868
AUTH_HMAC_SHA2_384_192	RFC 4868	RFC 4868
AUTH_HMAC_SHA2_256_128	RFC 4868	RFC 4868
AUTH_AES_CMAC_96	-	RFC 4494

Remarks:

- The suffixes of the HMAC methods have the following meaning:
  - The first suffix (512, 384, 256) describes the output length of the hash function and thus the output length of the HMAC in bits. This also corresponds to the key length.
  - The second suffix (256, 192, 128) describes the length in bits to which the output of the HMAC, i.e. the authentication tag, can be reduced.

Any Message Authentication Codes specified in the future that correspond to the requirements defined in this document can be used as well. For example, this applies for Message Authentication Codes that use a hash function from the SHA-3 family.

Reference:
[1] Bundesamt für Sicherheit in der Informationstechnik, TR-02102-3, Version 2023-01

Motivation: The integrity protection enables the recognition of data manipulation.

Implementation example: Configuration of HMAC_SHA2_512_256 for AH and ESP.

For this requirement the following threats are relevant:
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-32/i44

Req 33 For IKEv2 secure pseudorandom functions (PRF) according to the table below must be used.

User roles: Operation, Development, Integration

In IKEv2 pseudorandom functions (PRF) are used for key derivation. The following table lists the allowed pseudorandom functions as well as the reference specification for the usage in IKEv2.

Allowed PRFs for IKEv2:

PRF	**IANA Nr.**	**Reference specification**
PRF_HMAC_SHA2_512	7	RFC 4868
PRF_HMAC_SHA2_384	6	RFC 4868
PRF_HMAC_SHA2_256	5	RFC 4868
PRF_AES128_CMAC	4	RFC 4615

Remarks:

- The suffix (512, 384, 256) describes the output length of the hash algorithm and thus the output length of the HMAC in bits.
- For PRF_AES128_XCBC and PRF_AES128_CMAC the remarks in [RFC 7296, chapter 2.14] must be taken into account.

Any pseudorandom functions specified in the future that correspond to the requirements defined in this document can be used as well. For example, this applies for pseudorandom functions that use a hash function from the SHA-3 family.

Reference:
[1] Bundesamt für Sicherheit in der Informationstechnik, TR-02102-3, Version 2023-01

Motivation: A PRF enables key derivation.

Implementation example: Configuration of PRF_HMAC_SHA2_384 for IKEv2.

For this requirement the following threats are relevant:

- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-33/i44

Req 34 For IPSec the Diffie Hellman groups according to the table below must be used.

User roles: Operation, Development, Integration

The Diffie Hellman group is used for key exchange with Perfect Forward Secrecy (PFS). Generally, a distinction is made between elliptic curve groups and finite field groups (mod p).

The following table includes the allowed Diffie Hellman groups:

IANA-Nr.	**Diffie Hellman group**	**Reference specification**
30	brainpoolP512r1	RFC 6954
21	521-bit random ECP group	RFC 5903
32	x448	RFC 8031
29	brainpoolP384r1	RFC 6954
20	384-bit random ECP group	RFC 5903
28	brainpoolP256r1	RFC 6954
19	256-bit random ECP group	RFC 5903
31	x25519	RFC 8031
16	4096-bit MODP Group	RFC 3526
15	3072-bit MODP Group	RFC 3526

Remark on DH groups x448 and x25519:
DH groups x448 and x25519 are not (explicitly) recommended by BSI, but no weaknesses are known so far, and therefore they are to be classified as secure.

Remark on DH group 14:
Diffie-Hellman group 14 (IANA no. 14) is based on a 2048-bit prime [1] and may only be used in legacy systems until the **end of year 2025 [2]**. The DH group must be replaced by an approved DH group listed above.

References:
[1] Bundesamt für Sicherheit in der Informationstechnik, TR-02102-3, Version 2023-01
[2] SOG-IS Crypto Evaluation Scheme: Agreed Cryptographic Mechanisms, v1.3, February 2023

Motivation: Standardized Diffie Hellman groups use secure parameters and speed up the key exchange.

Implementation example: Cisco configuration example of Diffie-Hellman group in phase 1:
```
CE1_R1_4(config-isakmp)#group 20
```

Cisco configuration example of Diffie-Hellman group in phase 2:
```
CE1_R1_4(ipsec-profile)#set pfs group20
```

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-34/i44

10.2. Phase 1 (ISAKMP Security Association / IKE-SA)

Req 35 For IPSec a certificate-based authentication with an authentication method according to the table below must be used.

User roles: Operation, Development, Integration

Using a suitable authentication method, certificate-based authentication provides a higher security level than Pre-Shared Key authentication. Therefore, certificates of an appropriate Public Key Infrastructure must be used.

The following table contains the allowed authentication methods and key lengths:

Authentication method	**Hash algorithm**	**Key length**	**Reference specification**
ECDSA-512 with brainpoolP512r1	SHA-512	512	RFC 7427
ECDSA-512 with curvesecp521r1	SHA-512	512	RFC 4753, RFC 4754
Ed448	SHAKE256 (fix)	448	RFC 8420
ECDSA-384 with brainpoolP384r1	SHA-384	384	RFC 7427
ECDSA-384 with curveSecp384r1	SHA-384	384	RFC 4753, RFC 4754
ECDSA-256 with brainpoolP256r1	SHA-256	256	RFC 7427
ECDSA-256 with curveSecp256r1	SHA-256	256	RFC 4753, RFC 4754
Ed25519	SHA512 (fix)	256	RFC 8420
RSASSA-PSS	At least SHA-256	At least 3000 bit	RFC 7427, RFC 4055

Remark on Ed448 and Ed25519:
Ed448 and Ed25519 are not (explicitly) recommended by BSI, but no weaknesses are known so far, and therefore they are to be classified as secure.

Remark on RSA:
Key lengths smaller than 3000 bits may only be used in legacy systems until the end ot the year 2025 and should be substituted at the next opportunity. RSA-PKCS#1 v1.5 may only be used in legacy systems and must be substituted at the next opportunity.

Remark on Pre-Shared Keys (PSK) authentication:
The usage of a PSK-based authentication may only be approved by the securtiy organization of Deutsche Telekom group in exceptional cases. One exceptional case is the unavailability of certificate-based authentication.

If a Pre-Shared Key (PSK) based authentication is needed, the following conditions must be fulfilled:

- A PSK must be at least 128 bit long (or 18 ASCII characters) and randomly chosen (=high entropy).
- Each tunnel endpoint must define a half of the full PSK (e.g. 64-most significant bits by endpoint A / left 8 characters and 64-least significant bits / 8 right characters by endpoint B).
- A PSK must be transferred in an encrypted manner.
- It must be defined dedicated PSKs for each communication association.
- A PSK must be renewed regularly (at least every 24 months)

Motivation: Certificate-based authentication supports the flexible management of network accesses and usage of modern Public Key Infrastructures (PKI) incl. Certificate Revocation Lists (CRL) respectively Online Certificate Status Protocol (OCSP).

Implementation example: Usage of an x509v3 certificate for IKE-SA authentication.

For this requirement the following threats are relevant:

- Unauthorized access to the system
- Unauthorized access or tapping of data

For this requirement the following warranty objectives are relevant:

ID: 3.50-35/i44

Req 36	The IKE Security Association (IKE-SA) lifetime parameter must be a time-based value and must not exceed 86,400 seconds (24 hours).

User roles: Operation, Development, Integration

IKE-SA controls the authentication and authorization of peers. Hence an adequate value must be set.

Motivation: Unrotated keys can cause information leaks even if strong encryption is used.

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Unnoticeable feasible attacks

For this requirement the following warranty objectives are relevant:

ID: 3.50-36/i44

10.3. Phase 2 (Quick Mode / Child-SA)

Req 37	Encapsulated Security Payload (ESP) must be operated in Tunnel Mode.

User roles: Operation, Development, Integration

Remark on Authentication Header (AH):
Authentication Header (AH) can be used in combination with ESP. Nevertheless, ESP already provides all required features for integrity protection and Anti-Replay. That is the reason why the use of AH is not necessary.

Motivation: ESP already provides all required features for integrity protection and Anti-Replay.

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Denial of executed activities
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-37/i44

Req 38	The Child Security Association (Child-SA) lifetime parameter must be a time-based value and must not exceed 7,200 seconds (2 hours).

User roles: Operation, Development, Integration

The session key hierarchy will be renewed after the lifetime counter has exceeded.

Motivation: Unrotated keys can cause information leaks even if strong encryption is used.

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Unnoticeable feasible attacks

For this requirement the following warranty objectives are relevant:

ID: 3.50-38/i44

Req 39	The anti-replay window size must not be higher than 128 packets.

User roles: Operation, Development, Integration

The anti-replay protocol protects the IPSec connection against replay attacks.

Motivation: Protection against Replay attacks.

Implementation example: Router (crypto-map)# set security-association replay window-size 128

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized modification of data
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks

For this requirement the following warranty objectives are relevant:

ID: 3.50-39/i44

11. Transport Layer Security (TLS/SSL) Cipher Suites

Req 40	TLS version 1.2 or 1.3 must be used.

User roles: Operation, Development, Integration

TLS (Transport Layer Security) is a protocol for the secure transmission of information over TCP/IP based connections and is the successor of SSL (Secure Socket Layer). TLS ensures the confidentiality, integrity and authenticity of the information or the communication partners.

TLS in version 1.2 [RFC 5246] and version 1.3 [RFC 8446] provides cipher suites with Authenticated Encryption Associated Data (AEAD). AEAD ensures the confidentiality as well as the integrity and authenticity of the transmitted information.

References:
[RFC 5246] T. Dierks, E. Rescorla: RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2, 2008
[RFC 8446] E. Rescorla: RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3, 2018

Motivation: The current version of TLS fixes previous known security vulnerabilities and attack surfaces on the TLS protocol handshake.

Implementation example: OpenSSL> protocol = tlsv1_3

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-40/i44

Req 41	Only Perfect Forward Secrecy (PFS) TLS-cipher suites must be used according to the tables below.

User roles: Operation, Development, Integration

Cipher suites specify the cryptographic methods of a connection.

Perfect Forward Secrecy (short PFS, also Forward Secrecy) means that transmitted information cannot be decrypted afterwards, even if the long-term key of the communication partners is known.

In TLS v1.2 cipher suites are defined as follows: *TLS_AKE_WITH_Enc_Hash.*
Following, the meaning of the individual components is explained:

- *AKE (Authenticated Key Exchange)*: Key agreement mechanism with authentication for the handshake protocol.
- *Enc (Encryption)*: Encryption algorithm with mode of operation for the record protocol.
- *Hash*:Hash algorithm for HMAC used for key derivation. If*Enc*is not an AEAD encryption mechanism, HMAC is also used for integrity protection.

The following table lists the allowed cipher suites with PFS in TLS v1.2 as well as the reference specifications. The

design philosophy of TLS v1.2 was followed, which is why the table contains only AEAD constructions.
Allowed cipher suites with PFS in TLS v1.2:

Priority	**Cipher Suite**	**Reference specification**
HIGH	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	RFC 5289
HIGH	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	RFC 5289
LOW	TLS_DHE_DSS_ WITH_AES_256_GCM_SHA384	RFC 5288
LOW	TLS_DHE_RSA_ WITH_AES_256_GCM_SHA384	RFC 5288
HIGH	TLS_ECDHE_ECDSA_WITH_CHACHA20POLY1305_SHA256	RFC 7905
HIGH	TLS_ECDHE_RSA_WITH_CHACHA20POLY1305_SHA256	RFC 7905
LOW	TLS_DHE_RSA_WITH_CHACHA20POLY1305_SHA256	RFC 7905
HIGH	TLS_ECDHE_ECDSA_WITH_AES_256_CCM	RFC 7251
LOW	TLS_DHE_RSA_ WITH_AES_256_CCM	RFC 6655
HIGH	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	RFC 5289
HIGH	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	RFC 5289
LOW	TLS_DHE_DSS_ WITH_AES_128_GCM_SHA256	RFC 5288
LOW	TLS_DHE_RSA_ WITH_AES_128_GCM_SHA256	RFC 5288
HIGH	TLS_ECDHE_ECDSA_WITH_AES_128_CCM	RFC 7251
LOW	TLS_DHE_RSA_ WITH_AES_128_CCM	RFC 6655

Furthermore, in legacy systems the cipher suites of the following table are allowed.
Additional cipher suites with PFS in TLS v1.2 with AES-CBC:

Priority	**Cipher Suite**	**Reference specification**
HIGH	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	RFC 5289
HIGH	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	RFC 5289

LOW	TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	RFC 5246
LOW	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	RFC 5246
HIGH	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	RFC 5289
HIGH	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	RFC 5289
LOW	TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	RFC 5246
LOW	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	RFC 5246

Remark on the cipher suites for TLS v1.2:
The table entries are sorted by the symmetric encryption mechanism (Enc). For the authenticated key agreement methods (AKE), mechanism based on elliptic curves (ECDHE_ECDSA) are preferred. DHE (discrete logarithm) key establishment ciphers are more vulnerable against DoS (DHeater) than ECDHE, thus ECDHE should be preferred. The "Priority" column defines which cipher suites are preferred, i.e. cipher suites with a priority of "HIGH" are preferable to those with "LOW".

In TLS v1.3 cipher suites are defined as follows: *TLS_AEAD_Hash.*
Following, the meaning of the individual components is explained:

- AEAD: Authenticated encryption mechanism for the record protocol.
- Hash: Hash algorithm for HMAC and HKDF in the handshake protocol.

The following table lists the allowed cipher suites with PFS in TLS v1.3 as well as the reference specifications.
Allowed cipher suites with PFS in TLS v1.3:

Cipher suites	**Reference specification**
TLS_AES_256_GCM_SHA384	RFC 8446
TLS_CHACHA20_POLY1305_SHA256	RFC 8446
TLS_AES_128_GCM_SHA256	RFC 8446
TLS_AES_128_CCM_SHA256	RFC 8446

Any cipher suites specified in the future that correspond to the requirements defined in this document can be used as well. For example, this applies for cipher suites that use a hash function from the SHA-3 family.

References:
[1] Bundesamt für Sicherheit in der Informationstechnik, TR-02102-2, version 2023-01
[2] https://ciphersuite.info/cs/?security=secure&sort=asc
[3] https://ciphersuite.info/cs/?singlepage=true&security=recommended#

Motivation: The usage of modern cipher suites with Perfect Forward Secrecy protects the transport security in TLS.

Implementation example: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

For this requirement the following threats are relevant:

- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability

- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-41/i44

Req 42 For TLS, Diffie Hellman groups according to the table below must be used.

User roles: Operation, Development, Integration

The Diffie Hellman groups is used for key exchange with Perfect Forward Secrecy (PFS). Generally, a distinction is made between elliptic curve groups and finite field groups (mod p).

The following table contains the allowed Diffie Hellman groups.
Allowed Diffie Helman groups for use in TLS:

Diffie Hellman group	**IANA-No.**	**Referenzspezifikation**
brainpoolP512r1	33	RFC 7027
secp521r1	25	RFC 8422
x448	30	RFC 8422
brainpoolP384r1	27	RFC 7027
secp384r1	24	RFC 8422
brainpoolP256r1	26	RFC 7027
secp256r1	23	RFC 8422
x25519	29	RFC 8422
ffdhe4096	258	RFC 7919
ffdhe3072	257	RFC 7919

Remark on x448 and x25519:
x448 und x25519 are not (explicitly) recommended by BSI, but no weaknesses are known so far, and therefore they are to be classified as secure.

Remark on group 256:
Diffie Hellman group 256 (IANA-No.256) has a key length of 2048 bit [1] [2] and may only be used in legacy systems until the end of the **year 2025 [2]**. The group must be substituted by a stronger method (according to the enumeration above).

Remark on groups 256, 258 and 257:
Those groups are more vulnerable against the DHeater (DoS) attacks on server side than elliptic curve DH groups. Therefore, the brainpool and NIST (secp) groups should be preferred.

References:
[1] Bundesamt für Sicherheit in der Informationstechnik, TR-02102-2, Version 2023-01
[2] SOG-IS Crypto Evaluation Scheme: Agreed Cryptographic Mechanisms, v1.3, February 2023

Motivation: Standardized Diffie Hellman groups use secure parameters and speed up the key exchange.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data

- Unauthorized use of services or resources
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-42/i44

Req 43	For TLS, digital certificates from an appropriate certification authority with a sufficient key length and limited validity must be used.

User roles: Operation, Development, Integration

In TLS, digital certificates are used during the TLS handshake. TLS servers must use an appropriate TLS certificate. Clients need a certificate if mutual authentication is required.

TLS certificates for web servers that are accessible from the internet must be issued by public certification authorities that are classified as trustworthy by browsers and operating systems. These can be ordered, for example, via the service “TeleSec ServerPass” (please refer https://www.telesec.de/de/serverpass).

Regarding key lengths, validity and further configuration options, the Certificate Policy of the Certification Authority must be considered.

For web servers that are used exclusively for internal applications and are not accessible from the internet, digital certificates from a private (internal) Certification Authority can be used.

The minimal requirements according to the following table must be considered for each type of TLS certificates, this means also for client certificates:

Algorithm family	**Key length**	**Hash algorithm**
Elliptic Curve	250 bit	SHA-3, SHA-2 with an output length 256 bit
Digital Signature Algorithm (DSA)	3000 bit	SHA-3, SHA-2 with an output length 256 bit
RSA	3000 bit	SHA-3, SHA-2 with an output length 256 bit

Remarks on DSA and RSA certificates:
For DSA and RSA, key lengths smaller than 3000 bits may only be used in legacy systems [BSI TR 02102-1] until **end of the year 2025** [2] and should be substituted at the next opportunity. Because of the better performance, elliptic curve (EC-DSA) certificates shall be preferred (if supported and technically doable).
RSA-PKCS#1 v1.5 may only be used in legacy systems and should be (if feasible) substituted at the earliest opportunity [BSI TR 02102-1].

Restrictions on SHA-224/SHA-3-224:
SHA-224/SHA-3-224 may only be used in legacy systems and must be substituted by a stronger hash algorithm with an output length of at least 256 bits at the next opportunity.

The validity period of public TLS server certificates (issued by a certification authority, which issues certificates according to the specifications of the [CA/Browser Forum]) must not exceed 397 days. For other, internal TLS certificates, a validity period of 3 years should not be exceeded.

References:
[BSI TR 02102-1] Bundesamt für Sicherheit in der Informationstechnik: Cryptographic Mechanisms: Recommendations and Key Lengths, TR-02102-1, Version 2023-01

[1] Bundesamt für Sicherheit in der Informationstechnik, TR-02102-2, Version 2023-01
[2] SOG-IS Crypto Evaluation Scheme: Agreed Cryptographic Mechanisms, v1.3, February 2023
[CA/Browser Forum] https://cabforum.org/baseline-requirements-documents/

Motivation: Digital certificates form the basis of the authentication and build up trust. Without sufficiently strong authentication, man-in-the-middle attacks are possible.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-43/i44

Req 44 TLS server certificates must not be wildcard certificates.

User roles: Operation, Development, Integration

A wildcard certificate is not valid for individual servers, but for all servers that all belong to the same (sub)domain, e.g. *.abc.telekom.de.

As an alternative to wildcard certificates, certificates are available that can be automatically ordered and provided via the ACME protocol.

If no official CA (Certificate Authority) can be reached for a system to issue certificates via ACME protocol, then the use of wildcard certificates is acceptable – provided that the certificate is not valid for second-level domains (e.g. *.telekom.de) and it is not used for production and non-production.

Motivation: Wildcard certificates can be used for any number of domain names for which the wildcard expression is valid. This is an increased risk of misuse. In addition, the more servers use this certificate, the greater the risk of unauthorized access to the certificate's private key. These risks should be prevented. RFC 6125 also states that wildcard certificates should be avoided for security reasons.

Implementation example: Use of certificates for unique domain names that can be automatically updated via the ACME protocol, e.g. from TeleSec, https://telesec.de/en/ .

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-44/i44

Req 45 The security of TLS webserver implementations and configurations must be verified regularly.

User roles: Operation, Development, Integration

Secure cipher suites consisting of all essential parts of a modern security protocol:

- Authentication scheme
- Key Exchange algorithm
- Encryption algorithm
- MAC algorithm

The level of security is mainly driven by the concrete implementation and applied software. Furthermore, the security depends on the used protocol version. Therefore, it is mandatory to select applied cipher suites based on security best practices.

Recommended sources and test tools are:

- Cipher Suite Info (https://ciphersuite.info/)
- SSL Labs (https://www.ssllabs.com/ssltest/)
- Testssl (https://testssl.sh/)
- DTSP Internal Scan Platform (https://dtsp.telekom-dienste.de/)

Motivation: Only secure cipher suites reduce the attack surface and protect against downgrade attacks.

Implementation example: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

For this requirement the following threats are relevant:

- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.50-45/i44

12. Secure Shell (SSH) Cipher Suites

Req 46	The version 2 of the SSH protocol must be used. (Automatically compliant when using OpenSSH 7.4+).

User roles: Operation, Development, Integration

SSHv1 must permanently be disabled in configuration of the SSH server. With OpenSSH 7.4 support for SSHv1 completely removed and need not be configured explicitly.

Reference:
[BSI TR 02102-4] Bundesamt für Sicherheit in der Informationstechnik: TR-02102-4, Version 2023-01

Motivation: The SSH version 1 has cryptographic weaknesses and is obsolete today. With the use of SSHv1 the confidentiality and integrity of transmitted data cannot be guaranteed.

Implementation example: OpenSSH Parameter: Protocol 2

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability
- Unnoticeable feasible attacks

For this requirement the following warranty objectives are relevant:

ID: 3.50-46/i44

Req 47	SSH moduli smaller than 3072 bit must not be used.

The file "/etc/ssh/moduli" contains pre-generated group parameters – named moduli - for Diffie-Hellman. Here are also moduli available that are not long enough to withstand known attacks. To avoid the use of short values moduli smaller than 3072 bit must be deleted from file " /etc/ssh/moduli ".

Remark on 2048 bit Diffie-Hellman Groups:
Diffie-Hellman Groups with 2048 bit modulo may be used in legacy configurations until end of the year 2025 [1]. Those legacy configurations shall be substituted by elliptic curve key agreement ciphers.

Reference:
[1] SOG-IS Crypto Evaluation Scheme: Agreed Cryptographic Mechanisms, v1.3, February 2023

Motivation: If the DH moduli is too short the key exchange, then it is not protected in an adequate way.

Implementation example:

```
$ sed -i '/ 1023 /d' /etc/ssh/moduli
$ sed -i '/ 1535 /d' /etc/ssh/moduli
$ sed -i '/ 2047 /d' /etc/ssh/moduli
$ sed -i '/ 3071 /d' /etc/ssh/moduli
```

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

For this requirement the following warranty objectives are relevant:

ID: 3.50-47/i44

Req 48 Only approved key exchange algorithms must be used. See long text for a complete list.

User roles: Operation, Development, Integration

For key exchange the following algorithms are allowed:

- curve448-sha512
- sntrup761x25519-sha512@openssh.com
- curve25519-sha256@libssh.org
- curve25519-sha256
- ecdh-sha2-nistp521
- ecdh-sha2-nistp384
- ecdh-sha2-nistp256
- diffie-hellman-group18-sha512
- diffie-hellman-group16-sha512
- diffie-hellman-group-exchange-sha256

Remark on curve448 and curve25519:
curve448 und curve25519 are not (explicitly) recommended by BSI, but no weaknesses are known so far, and therefore they are to be classified as secure.

Remark on Diffie-Hellman Groups:
Key exchange algorithms with 2048 bit MODP may be used in legacy configurations until end of the year 2025 [1].

Reference:
[1] SOG-IS Crypto Evaluation Scheme: Agreed Cryptographic Mechanisms, v1.3, February 2023
[BSI TR 02102-4] Bundesamt für Sicherheit in der Informationstechnik: TR-02102-4, Version 2023-01
[RFC8731] A. Adamantiadis, S. Josefsso, M. Baushke: RFC 8731, Secure Shell (SSH) Key Exchange Method Using Curve25519 and Curve448

Motivation: An attacker can possibly break the encryption of transported data if weak ciphers and algorithms are used to access secret data.

Implementation example: sshd_config contains:

```
KexAlgorithms curve448-sha512, curve25519-sha256@libssh.org,
curve25519-sha256, ecdh-sha2-nistp521, ecdh-sha2-nistp384, ecdh-
sha2nistp256, diffie-hellman-group18-sha512, dif-
fie-hellman-group16sha512, diffie-hellman-group14-sha256, dif-
fie-hellman-groupexchange-sha256
```

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data

For this requirement the following warranty objectives are relevant:

ID: 3.50-48/i44

Req 49 Only approved ciphers algorithms must be used. See long text for a complete list.

User roles: Operation, Development, Integration

Outdated and insecure ciphers and algorithms must not be used. Use the following ciphers for SSH:

- aes256-gcm@openssh.com

- aes128-gcm@openssh.com
- chacha20-poly1305@openssh.com
- aes256-ctr
- aes192-ctr
- aes128-ctr

Motivation: An attacker can possibly break the encryption of transported data if weak ciphers and algorithms are used to access secret data.

Implementation example: sshd_config contains:

```
Ciphers aes256-gcm@openssh.com,aes128-gcm@openssh.com,chacha20-poly1305@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr
```

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Denial of executed activities

For this requirement the following warranty objectives are relevant:

ID: 3.50-49/i44

Req 50	Only approved MAC algorithms must be used. See long text for a complete list.

User roles: Operation, Development, Integration

It is important to avoid the use of insecure MAC algorithms for SSH. Examples of such outdated algorithms are MD5 and SHA1. The following MAC algorithms are allowed and must be configured for SSH daemon:

- hmac-sha2-512-etm@openssh.com
- hmac-sha2-512
- hmac-sha2-256-etm@openssh.com
- hmac-sha2-256

Motivation: An attacker can possibly break the encryption of transported data if weak ciphers and algorithms are used to access secret data.

Implementation example: sshd_config contains:

```
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-512,hmac-sha2-256-etm@openssh.com,hmac-sha2-256
```

For this requirement the following threats are relevant:
- Unauthorized modification of data

For this requirement the following warranty objectives are relevant:

ID: 3.50-50/i44

Req 51	Only approved Host Key Algorithms (a.k.a. public key signature algorithms or server authentication algorithms) must be used. See long text for a complete list. A Host Key must identify a host distinctly. (1:1 relation).

User roles: Operation, Development, Integration

Outdated and insecure algorithms must not be used. Use the following host key algorithms. Naming may differ from the following list:

- ssh-ed448
- ssh-ed25519
- ssh-ed25519-cert-v01@openssh.com
- sk-ssh-ed25519@openssh.com
- sk-ssh-ed25519-cert-v01@openssh.com
- ecdsa-sha2-nistp521
- ecdsa-sha2-nistp521-cert-v01@openssh.com
- ecdsa-sha2-nistp384
- ecdsa-sha2-nistp384-cert-v01@openssh.com
- ecdsa-sha2-nistp256
- ecdsa-sha2-nistp256-cert-v01@openssh.com
- sk-ecdsa-sha2-nistp256@openssh.com
- sk-ecdsa-sha2-nistp256-cert-v01@openssh.com

Remark on ed448 and ed25519:
ed448 and ed25519 are not (explicitly) recommended by BSI, but no weaknesses are known so far, and therefore they are to be classified as secure.

Remark on RSA HostKeys:
RSA HostKeys with a length of 2048 bit may be used in legacy configurations until end of the year 2025 [1].

Reference:
[1] SOG-IS Crypto Evaluation Scheme: Agreed Cryptographic Mechanisms, v1.3, February 2023
[RFC8709] B. Harris, L. Velvindron: RFC 8709, Ed25519 and Ed448 Public Key Algorithms for the Secure Shell (SSH) Protocol

Motivation: An attacker can possibly impersonate a target host undetected.

Implementation example: sshd_config contains:

```
HostKeyAlgorithms ssh-ed448,ssh-ed25519,ssh-ed25519-cert-
v01@openssh.com,sk-ssh-ed25519@openssh.com,sk-ssh-ed25519-cert-
v01@openssh.com,ecdsa-sha2-nistp521,ecdsa-sha2-nistp521-cert-
v01@openssh.com,ecdsa-sha2-nistp384,ecdsa-sha2-nistp384-cert-
v01@openssh.com,ecdsa-sha2-nistp256,ecdsa-sha2-nistp256-cert-
v01@openssh.com,sk-ecdsa-sha2-nistp256@openssh.com,sk-ecdsa-
sha2-nistp256-cert-v01@openssh.com
```

For this requirement the following threats are relevant:

- Unauthorized access to the system
- Unauthorized access or tapping of data

For this requirement the following warranty objectives are relevant:

ID: 3.50-51/i44

13. Table of Comparable Security Levels for Cryptographic Algorithms

Security Level (the higher the better)	**Symmetric Algorithms**	**Asymmetric Algorithms DSA / Diffie-Hellman**	**Asymmetric Algorithms RSA**	**Asymmetric Algorithms ECDSA (elliptic curves)**	**Hash for Signatures, Passwords + Data-integrity**	**Hash for HMAC, Key Derivation + Random Number Generation**
256 (very high)	AES-256 ChaCha20	p=15360/ q=512	n=15360	ord(g) = 512	SHA-512 SHA-3-512	SHA-512 SHA-3-512
192 (high)	AES-192	p=7680/ q=384	n=7680	ord(g) = 384	SHA-384 SHA-3-384	SHA-384 SHA-3-384
128 (standard)	AES-128	p=3000/ q=250	n=3000	ord(g) = 250	SHA-256 SHA-3-256	SHA-256 SHA-3-256
<=112 (legacy)	3DES-168	p=2048/ q=224	n=2048	ord(g) = 224	SHA-224 SHA-3-224	SHA-224 SHA-3-224

Key:
n: RSA modulus
ord(g): order of the base point (generator point g) on an elliptic curve
p: prime number (definition of a finite Galois Field)
q: prime number

Approved elliptic curves:

- brainpoolP512r1
- NIST P-521
- brainpoolP384r1
- NIST P-384
- brainpoolP320r1
- brainpoolP256r1
- NIST P-256
- Curve25519
- Curve448