ETSI TR 103 719 V1.1.1 (2022-03)

TECHNICAL REPORT

Guide to Identity-Based Cryptography

Reference

DTR/CYBER-0045

Keywords

privacy, security, user

Contents

Intellectual Property Rights

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Cyber Security (CYBER).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "will not", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document describes the use and application of Identity-Based Cryptography (IBC) applied to both encryption, as Identity-Based Encryption (IBE), and to digital signature, as Identity-Based Signature (IBS). The present document is intended to allow non-experts in the technology of IBC to be able to gain an understanding of the technology, its domains of application, and its required environment.

In addition the present document reviews use of IBE against more common asymmetric encryption schemes (e.g. RSA and ECC schemes), as well as more generalized Functional Encryption (FE) and Attributed Based Encryption (ABE) schemes. The present document also reviews use of IBS against more common digital signature schemes (e.g. DSA, ECDSA). An overview of algorithms and their modes of operation available to implement IBE, and IBS, and their evolution to quantum safe instances is also covered in the present document.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] D. Boneh, M. Franklin: "Identity-Based Encryption from the Weil Pairing", SIAM J. Comput. 32(3): 586-615, 2003.

[i.2] D. Boneh, G. Di Crescenzo, R. Ostrovsky, G. Persiano: "Public Key Encryption with Keyword Search", EUROCRYPT 2004: 506-522.

NOTE: Available from https://link.springer.com/content/pdf/10.1007/978-3-540-24676-3_30.pdf.

[i.3] M. Abdalla, J. Birkett, D. Catalano, A. W. Dent, J. Malone-Lee, G. Neven, J. C. N. Schuldt and N. P. Smart: "Wildcarded Identity-Based Encryption", J. Cryptology 24(1): 42-82, 2011.

[i.4] V. Goyal, O. Pandey, A. Sahai, B. Waters: "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data", ACM CCS 2006: 89-98.

[i.5] ETSI TS 100 392-7: "Terrestrial Trunked Radio (TETRA); Voice plus Data (V+D); Part 7: Security".

[i.6] ETSI EN 300 396-6: "Terrestrial Trunked Radio (TETRA); Direct Mode Operation (DMO); Part 6: Security".

[i.7] NIST SP 800-57 Part 1 rev 5: "Recommendation for Key Management: Part 1 - General".

NOTE: Available from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf.

[i.8] SAFEcrypto Project.

NOTE: Available from https://www.safecrypto.eu.

[i.9] C. Gentry, A. Silverberg: "Hierarchical ID-Based Cryptography", ASIACRYPT 2002: 548-566.

[i.10] ETSI TS 103 486: "CYBER; Identity Management and Discovery for IoT".

[i.11] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

[i.12] IEEE 1363.3™-2013: "IEEE Standard for Identity-Based Cryptographic Techniques using Pairings".

[i.13] ISO/IEC 14888-3:2018: "IT Security techniques - Digital signatures with appendix - Part 3: Discrete logarithm based mechanisms".

[i.14] ETSI TS 102 940: "Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management; Release 2".

[i.15] N. McCullagh, P. S. L. M. Barreto: "A New Two-Party Identity-Based Authenticated Key Agreement".

NOTE: Available from https://eprint.iacr.org/2004/122.pdf.

[i.16] eXtensible Access Control Markup Language (XACML) Version 3.0 OASIS Standard, 22 January 2013.

[i.17] IETF RFC 6749: "The OAuth 2.0 Authorization Framework".

[i.18] IETF RFC 6350: "vCard Format Specification".

[i.19] IETF RFC 6508: "Sakai-Kasahara Key Encryption (SAKKE)".

[i.20] IETF RFC 6507: "Elliptic Curve-Based Certificateless Signatures for Identity-Based Encryption (ECCSI)".

[i.21] ETSI EG 203 310 (V1.1.1) (2016-06): "CYBER; Quantum Computing Impact on security of ICT Systems; Recommendations on Business Continuity and Algorithm Selection".

[i.22] ETSI TR 103 618: "CYBER; Quantum-Safe Identity-Based Encryption".

[i.23] ISO/IEC 18033-5:2015: "Information technology -- Security techniques -- Encryption algorithms - Part 5: Identity-based ciphers".

[i.24] R. Sakai, K. Ohgishi, M. Kasahara: "ID based cryptosystem based on pairing on elliptic curves", Symposium on Cryptography and Information Security - SCIS, 2001.

[i.25] M. Barbosa et al: "SK-KEM: An Identity-Based KEM", submission for IEEE P1363.3, June 2006.

[i.26] Z. Cheng: "The SM9 Cryptographic Schemes". .

NOTE: Available from https://eprint.iacr.org/2017/117.pdf.

[i.27] R. Tse et al: "The SM9 Identity-Based Cryptographic Algorithms". draft-sca-cfrg-sm9-00.

NOTE: Available from https://ietf.org/staging/draft-sca-cfrg-sm9-00.txt.

[i.28] ISO/IEC 18033-5:2015/Amd.1:2021: "Information technology -- Security techniques -- Encryption algorithms -- Part 5: Identity-based ciphers. Amendment 1: SM9 mechanism".

[i.29] ETSI TS 122 179: "LTE; Mission Critical Push to Talk (MCPTT) over LTE; Stage 1".

[i.30] K. G. Paterson, G. Price: "A comparison between traditional Public Key Infrastructures and Identity-Based Cryptography". PKI re-visited - current issues and future trends: v. 8 n. 3 of Information Security Technical Report. Elsevier, 2003. pp. 57-72.

[i.31] IETF RFC 5091: "Identity-Based Cryptography Standard (IBCS) #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

communication server: abstract entity that forwards messages

IBC-identity: extension of the identifier with additional information that relates to the entity in the IBC context, such as an access control policy

EXAMPLE: IBC-identity = identifier || [access-control-policy] || [other-metadata].

IBE agent: entity that handles operations related to an IBE cryptosystem

Key Management Service (KMS): entity or function that manages the master secret key, the master public key, and user secret keys

Master Public Key (MPK): key holding publicly available parameters that relate to a master secret key

Master Secret Key (MSK): secret value used by the secret key generator to derive user secret keys for an IBE mechanism

public identity: publicly available information that describes characteristics of a user or entity and from which they can be identified

public key: alternative name to the public identity in IBE

public parameter service: entity or function that distributes public parameters (master public keys) to users

Secret Key (SK): decryption or signing key, private to the user, that corresponds to the user's public identity

secret key generator: entity or function that generates secret keys

3.2 Symbols

For the purposes of the present document, the following symbols apply:

C	Ciphertext
e	Bilinear pairing
$\mathbb{G}_i$	Cyclic group of prime order p, written additively
G_i	Generator of the cyclic group $\mathbb{G}_i$
$\mathbb{G}_T$	Cyclic group of prime order p, written multiplicatively
H_i	Hash function
ID	Public identifier
M	Message
p	Prime number
$RK_{ID \rightarrow ID'}$	Re-encryption key allowing re-encryption of a ciphertext for ID into a ciphertext for ID'
SK_{ID}	Secret Key for the public identifier ID
$\mathbb{Z}_p$	Integers modulo the prime p

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ABE	Attribute Based Encryption
BLMQ	Barreto, Libert, McCullagh, Quisquater
CA	Certificate Authority
C-ITS	Co-operative Intelligent Transport System
DNS	Domain Name System

DSA	Digital Signature Algorithm
ECC	Elliptical Curve Cryptography
ECCSI	Elliptic Curve-based Certificateless Signatures for Identity-based
ECDSA	Elliptical Curve Digital Signature Algorithm
FE	Functional Encryption
HIBE	Hierarchical Identity-Based Encryption
IBC	Identity-Based Cryptography
IBE	Identity-Based Encryption
IBS	Identity-Based Signature
IdM	Identity Management
IMAP	Internet Message Access Protocol
IMAP4	Internet Messaging Access Protocol v4
IoT	Internet of Things
ITS	Intelligent Transport System
KMS	Key Management Service
LMTP	Local Mail Transport Protocol
LTE	Long Term Evolution
MCPTT	Mission Critical Push To Talk
MPK	Master Public Key
MSK	Master Secret Key
MTA	Message Transfer Agent
MUA	Message User Agent
NIST	National Institute of Standards and Technology
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PKC	Public-Key Cryptography
PKI	Public-Key Infrastructure
POP	Post Office Protocol
POP3	Post Office Protocol v3
RK	Random Key
RSA	Rivest-Shamir-Adleman
SK	Secret Key
SKID	Secret Key IDentity
SMTP	Simple Mail Transfer Protocol
SMTPS	Simple Mail Transfer Protocol Secure
SP	Special Publication (of NIST)
URI	Uniform Resource Identifier
XACML	eXtensible Authorisation Control Markup Language

4 What lies behind Identity-Based Cryptography?

4.1 Core principles and thought experiments

Using everyday language, the concept behind Identity-Based Cryptography (IBC) is simple: can a human readable public identifier, such as an e-mail address, be used as the public key of a cryptographic system?

Consider the thought experiment for e-mail: if Alice wants to send Bob an e-mail and she knows Bob's e-mail address (say bob@emailserver.com), is there a way for her to encrypt the e-mail to Bob without having to find other, more opaque, data? Bob's e-mail address serves to identify Bob and offer systemic guarantees of delivery, so it would be reasonable for Alice to use it to secure the e-mail.

Similarly, if Alice wants Bob to be able to verify that the e-mail came from her is there a way for Bob to use Alice's e-mail address (say alice@emailserver.com) as the verification key in a digital signature? Alice's e-mail address again serves to identify Alice so it would be reasonable for Bob to use it to verify the e-mail.

More generally, if Alice wants to e-mail a designated role in an organization (say sales@big-corp.com) then the e-mail address for that role is sufficient to identify the individual, or individuals, in the role. In this case, it would still be reasonable for Alice to use the role's e-mail address to secure or verify the e-mails.

All of these identifiers are "public" and come with semantic labelling. The semantic label can be broadened to a Uniform Resource Identifier (URI) [i.11] such as a telephone number, e-mail address, instant messaging account, video messaging account or a file-server address. The question is then whether it is possible to use identifiers, in the form of URIs, as the public keys in a cryptographic security system. This is significantly different from the conventional approach to public key cryptographic schemes where identifiers (semantic or canonical) are not intrinsic elements of the cryptosystem.

The present document makes a distinction between the identifier used as the public key and the identity of the principal (see also ETSI TS 103 486 [i.10]). The full identity of the principal is a complex socio-economic construct, whereas for the purposes of the present document only, an identifier is considered as a public pseudonym (or alias) of the principal.

For simplicity the IBC-identity is considered as an extension of the identifier with additional information that relates to the entity in the IBC context, such as an access control policy.

EXAMPLE: IBC-identity = identifier || [access-control-policy] || [other-metadata].

Public-Key Cryptography (PKC) is widely understood and implemented. In a conventional PKC system, a user randomly chooses their secret key and then derives their public key from the secret key in a way that it is hard to reverse. Public keys generated in this way can have no intrinsic meaning.

The aim of IBC is to replace the randomly generated public key with a human readable public key that has, in some cases, a predefined structure or syntax (e.g. name@domain for an e-mail address). Further, in an IBC scheme a user's public key is known in advance - there is an explicit semantic binding to the identity of the user - so the user's secret key needs to be derived from their public key. The most significant difference between conventional PKC and IBC is that user secret keys are derived by a trusted central authority, the Key Management Service (KMS), rather than users.

This means that for IBC there are additional considerations that are useful to understand in determining the viability of a design. For many "enterprise" identifiers such as e-mail addresses there is a central manager, i.e. a role within the organization that is responsible for assigning Alice and Bob their e-mail identities. The central point of control for the identifier can act as the trusted KMS for the service.

More specifically, the KMS performs two fundamental functions in an IBC scheme:

1) **Setup:** (establishing the ground work)
   - Given a security parameter k, compute a Master Public Key (MPK) and a Master Secret Key (MSK) for the system.
2) **Extract:** (deriving the user secrets)
   - Given the MPK, the MSK and the public identity ID for a user, derive the user secret key SK_{ID}.

NOTE 1: Setup and extraction are performed centrally and do not need to involve users of the scheme.

For example, if the IBC scheme is used for Identity-Based Encryption (IBE):

3) **Encrypt:**
   - Given the MPK, a public identity ID associated to the intended recipient, and a message M, compute the ciphertext C of M under the public identity ID.
4) **Decrypt:**
   - Given MPK, a ciphertext C, and the secret key SK_{ID} associated to the recipient, compute the plaintext M from the ciphertext C.

NOTE 2: The KMS does not need to be involved in the communications between users once they have been given their secret keys.

Figure 1 shows how the 4 steps inter-relate.

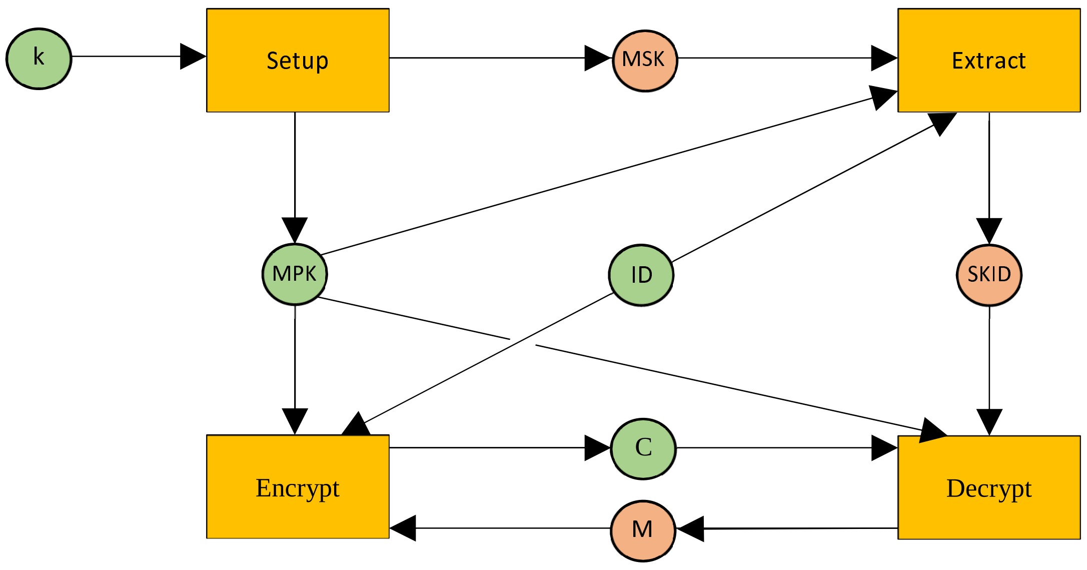


Figure 1: IBE algorithms and their inputs and outputs

The security of the IBE scheme depends on the difficulty of recovering the secret parameters MSK or SK_{ID} (shown in red in Figure 1) given MPK, C and ID (shown in green in Figure 1). An alternative phrasing is that if the adversary, Eve, knows the public parameters of the scheme it is infeasible, from that knowledge alone, to recover any of the secrets of the system. Specific approaches for the cryptographic structure of the algorithms for each of the 4 stages are considered in more detail in Annex D.

4.2 Identity-Based Encryption (IBE)

In more detail, an IBE cryptosystem requires a trusted third party that manages a master public key (the public parameters) and a master secret key. Figure 2 below provides a general model of IBE cryptosystems.

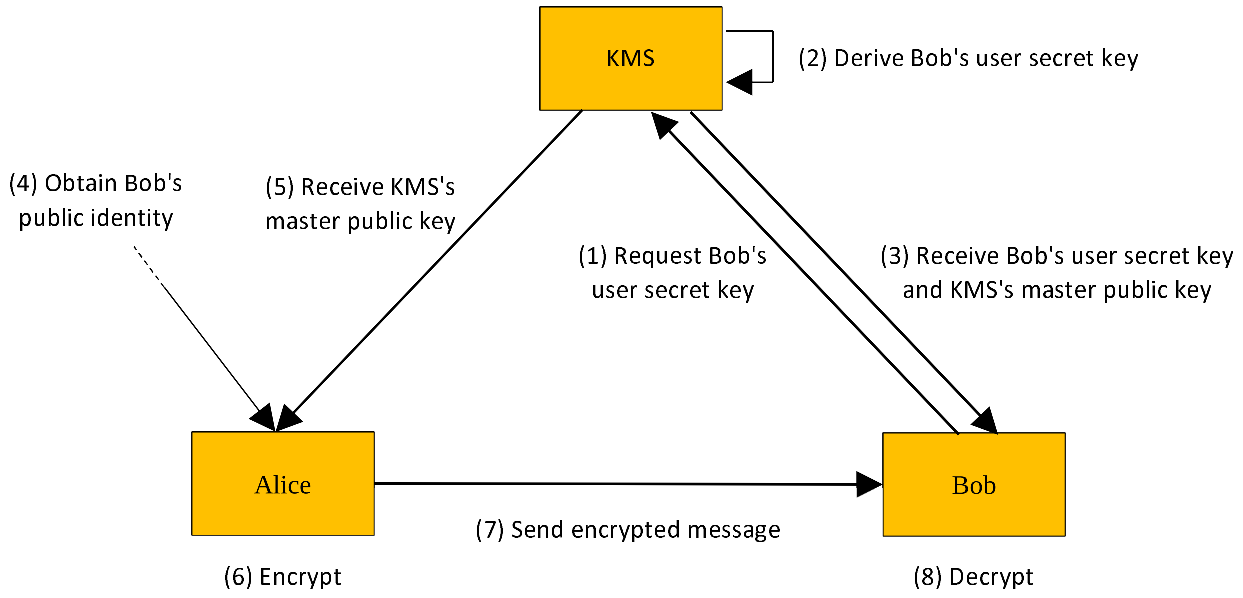

KMS
(2) Derive Bob's user secret key
(4) Obtain Bob's public identity
(5) Receive KMS's master public key
(1) Request Bob's user secret key
(3) Receive Bob's user secret key and KMS's master public key
Alice
Bob
(7) Send encrypted message
(6) Encrypt
(8) Decrypt

Figure 2: General model of IBE cryptosystems

Provisioning a user involves the following steps:

1) The user, Bob, requests the user secret key corresponding to his public identity (e.g. the e-mail address bob@example.com) from the KMS.

2) The KMS derives Bob's user secret key from his public identity and the master secret key.

3) The KMS securely sends Bob his user secret key and the master public key.

Sending an encrypted message involves the following steps:

4) The sender, Alice, obtains the public identity of the recipient, Bob.

5) Alice obtains the master public key from the KMS.

6) Alice encrypts the message using Bob's public identity and the master public key.

7) Alice sends the encrypted message to Bob.

Decrypting a received message involves the following step:

8) The recipient, Bob, decrypts the message using his user secret key and the master public key.

NOTE 1: Alice can send Bob an encrypted message (steps 4-7) before Bob has requested and been given his user secret key (steps 1-3).

The IBE model outlined above requires that the sending party holds both a public identity of the recipient and the master public key of the system in order to encrypt a message. It is reinforced that the security strength in IBE does not rely on properties of the recipient's public identity.

NOTE 2: The convention in the present document is to refer to public identity, rather than public key, and to master public key obtained from the trusted 3rd party, in order to distinguish the use of these terms from those used in traditional public key encryption models.

In order to decrypt a message, the receiving party, Bob, has to have a secret key, which is computed by the KMS from the master secret key and the public identity for which the message was encrypted (most often this will be an identifier of Bob such as his e-mail address). The terminology "secret key" is better suited to IBE than "private key" since the secret keys are not generated by users, and users can be given many secret keys each related to a different public identity.

NOTE 3: In conventional Public-Key Cryptography (PKC) each of Alice and Bob create their own key pairs (the public and private parts of the key) whereas in IBE a trusted third party is always involved in the creation of the key pair.

A risk analysis and considerations for deployment of a KMS are given in annex B. Considerations for the lifecycle of an IBE cryptosystem are given in Annex C. Descriptions of several proposed IBE schemes are given in clause D.3.

4.3 Identity-Based Signatures (IBS)

IBC has two distinct ways of providing Identity-Based Signatures (IBS):

- signature-by-proxy;
- signature using identity-based signing keys.

In the signature-by-proxy method the data to be signed is sent to the KMS and the KMS acts as the signing agent. While the sender relies on the KMS for computing a signature, the recipient can request public verification keys from the KMS and perform signature verification themselves. This is illustrated in Figure 3.

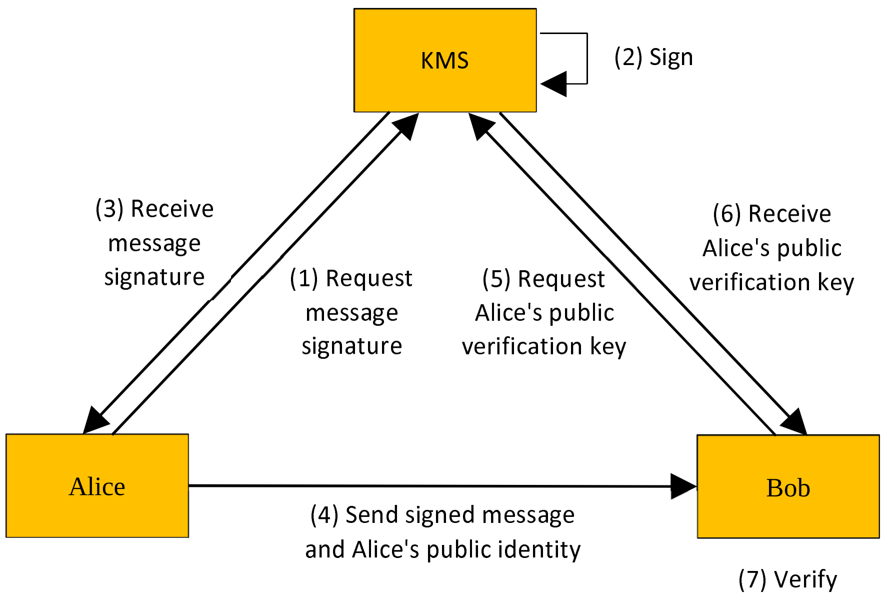

KMS
(2) Sign
(3) Receive message signature
(1) Request message signature
(5) Request Alice's public verification key
(6) Receive Alice's public verification key
Alice
Bob
(4) Send signed message and Alice's public identity
(7) Verify

Figure 3: Identity-based signature proxied by the KMS

Sending a signed message involves the following steps:

1) The sender, Alice, securely sends the KMS the message and her public identity (e.g. the e-mail address alice@example.com) and requests that the KMS signs the message.

2) The KMS signs the message with the user secret key corresponding to her public identity.

3) The KMS sends the signed message to Alice.

4) Alice sends the signed message to the recipient, Bob, together with her public identity.

Verifying the signature on a received message involves the following steps:

5) Bob requests the public verification key corresponding to Alice's public identity from the KMS.

6) The KMS sends Alice's public verification key to Bob.

7) Bob verifies the signature using Alice's public verification key.

NOTE: In a signature-by-proxy scheme, Alice cannot verify the signature herself without first being given her public verification key by the KMS.

The second way of providing digital signatures using IBC is where the KMS distributes identity-based signing keys and verification keys. This is illustrated in Figure 4.

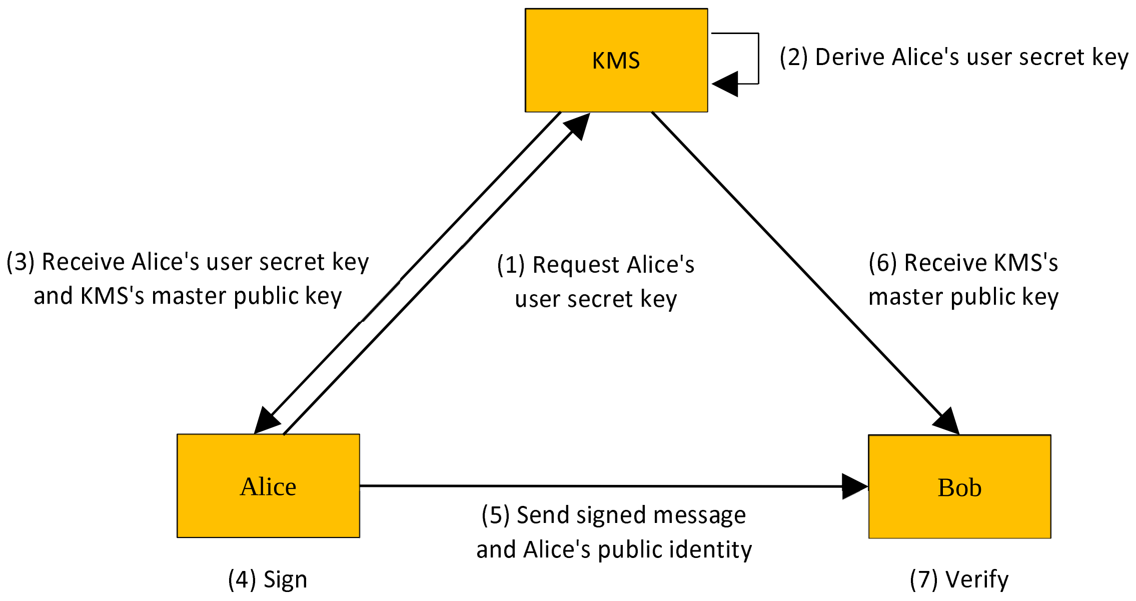

KMS
(2) Derive Alice's user secret key
(3) Receive Alice's user secret key and KMS's master public key
(1) Request Alice's user secret key
(6) Receive KMS's master public key
Alice
Bob
(5) Send signed message and Alice's public identity
(4) Sign
(7) Verify

Figure 4: Identity-based digital signature scheme

Provisioning a user involves the following steps:

1) The user, Alice, requests the user secret key corresponding to her public identity (e.g. the e-mail address alice@example.com) from the KMS.

2) The KMS derives Alice's user secret key from her public identity and the master secret key.

3) The KMS securely sends Alice her user secret key and the master public key.

Sending a signed message involves the following steps:

4) The sender, Alice, signs the message using her user private key and the master public key.

5) Alice sends the signed message to the recipient, Bob, together with her public identity.

Verifying the signature on a received message involves the following steps:

6) Bob obtains the master public key from the KMS.

7) Bob verifies Alice's signature using her public identity and the master public key.

A risk analysis and considerations for deployment options are given in Annex B. Descriptions of several proposed IBS schemes are given in clause D.4.

4.4 IBC for key encapsulation

In addition to encryption and digital signatures, key encapsulation is another major use of public-key cryptography. Key encapsulation uses the IBC framework to send a symmetric key from Alice to Bob.

Descriptions of several identity-based key encapsulation mechanisms are given in clause D.3.

4.5 Advantages and disadvantages over conventional PKC

4.5.1 Architectural model comparisons

A detailed analysis comparing public-key infrastructures and identity-based cryptography can be found in [i.30].

Key infrastructure:

- Conventional PKC solutions require a means of binding a public key to the identity, or some other attribute, of its user. A common method for doing this is via public key certificates that are managed in a Public Key Infrastructure (PKI). Certificates are issued by a trusted Certificate Authority (CA) who needs to ensure that the user requesting the certificate is the legitimate owner of the public key.

- In an IBC scheme, the public identities can contain explicit semantic content, such as the e-mail address of the user, and be integrated with the protected service. Secret keys are derived by a trusted KMS who needs to ensure that the user requesting the secret key is the legitimate owner of the identity.

Key generation:

- The master key pair for a PKI is generated by the CA. The master public key needs to be distributed to all users of the service so that they can verify user certificates.

  User key pairs can either be generated by users or by the CA. If user key pairs are centrally generated, then the user secret key needs to be sent securely to the user.

- The master key pair for an IBC scheme is generated by the KMS. The master public key needs to be distributed to all users of the service so that they can use it in encryption, for an IBE scheme, or in verification, for an IBS scheme.

  User secret keys can only be derived by the KMS and need to be sent securely to the user.

Key storage:

- The master secret key for a PKI needs to be stored securely by the CA. If user key pairs are centrally generated and retained in order to provide resilience and recovery, then the user secret keys need to be stored securely by the CA.

  Users need to store their user secret keys securely. Certificates containing the user public encryption keys need to be retrievable from a central repository. Certificates containing user public verification keys can be sent along with the signed message.

- The master secret key for an IBC scheme needs to be stored securely by the KMS. User secret keys can be rederived by the KMS so do not need to be retained in order to provide resilience and recovery.

  Users need to store their user private keys securely. User public identities can be integrated with the protected service and so do not need to be retrievable from a central repository.

Key lifetimes:

- The master key pair in a PKI tends to be long lived and updated infrequently.

  Certificates containing user public keys tend to be much shorter lived and updated regularly. Certificates include explicit expiry dates for user public keys, so a new certificate needs to be issued when the old certificate expires.

- The master key pair in an IBC scheme tends to be long lived and updated infrequently.

  Public identities such as e-mail addresses do not expire, but the identity can be extended to include an expiry date. If the identity does include an expiry date, then a secret key needs to be derived for the new identity when the old identity expires.

Key compromise:

- If the master secret key for a PKI is compromised, then an attacker can forge a certificate for any user of the service.

  If the certificate contains a user public encryption key, then the attacker can impersonate the user and read any messages intended for the real user that are encrypted using the forged certificate. However, the attacker would not be able to read any messages sent to the real user before the compromise.

If the certificate contains a user public verification key, then the attacker can impersonate the user and sign messages that would verify correctly with the forged certificate. However, the signed messages would not verify correctly with a genuine certificate previously issued to the user.

- If the master secret key for an IBC scheme is compromised, then an attacker can derive the user secret key for any user of the service.

  For an IBE scheme, the attacker would be able to read any message sent to the user including messages that were sent before the compromise.

  For an IBS scheme, the attacker would be able to impersonate the user and forge signatures that are indistinguishable from a genuine signature for that user.

Key revocation:

- In a PKI, user certificates can be revoked by the CA and replaced by new certificates issued with fresh user key pairs. Short-lived user certificates can make revocation simpler as new certificates will already be needed when the old certificates automatically expire.

  Revoking the master key pair is more difficult and would require all user certificates to be reissued.

- In an IBC scheme, user public identities such as e-mail addresses are static and so cannot be revoked or reissued easily. If the public identity is extended to include an expiry date, then revocation is automatic at the end of the lifetime of the identity and a fresh user secret key can be derived for the new identity.

  Revoking the master key pair is more difficult and would require all user secret keys to be rederived.

4.5.2 Protocol model comparisons

In a conventional PKI-based encryption scheme, Alice needs a certificate containing Bob's public key before she can send him an encrypted message. This means that Bob first needs to register his public key with the CA so that the CA can issue a certificate and publish it to a repository from which Alice can then request it (see Figure 5).

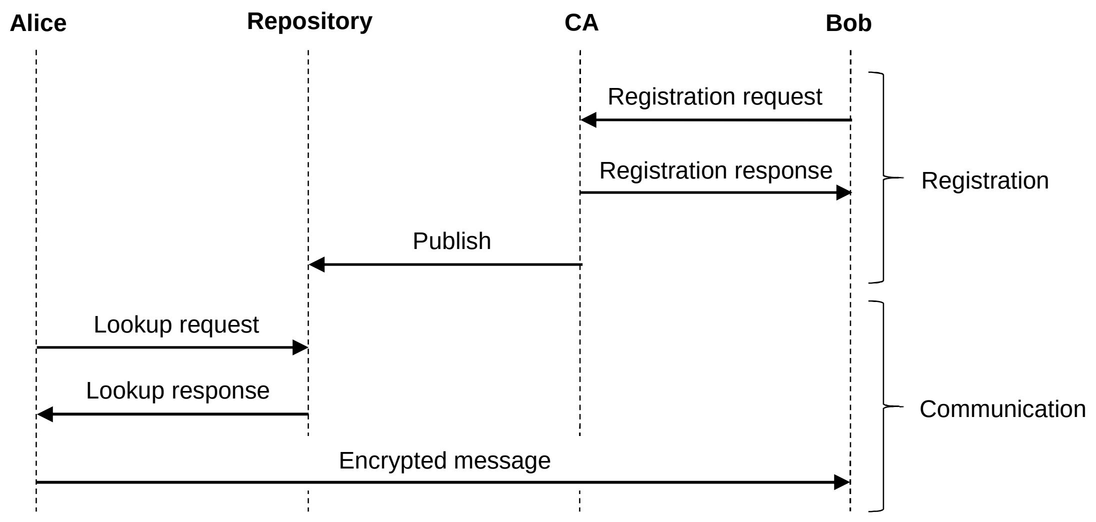


Figure 5: Encrypted communication with a PKI

If Alice wants to send Bob a message protected by an IBE scheme, she only needs to know Bob's public identifier and the master public key. This can enable simplex transmission of encrypted messages without the involvement of the KMS (Figure 6). It is possible for Alice to send Bob an encrypted message before he has registered and been given a secret key.

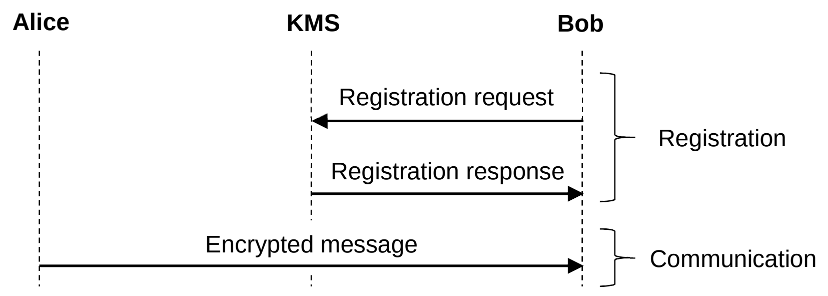


Figure 6: Encrypted communication with IBE

PKIs that handle a large number of users typically involve multiple levels of CAs. For example, in a two-tier model the root CA delegates authority to one or more issuing CAs who then sign the certificates containing user public keys. Hierarchical Identity-Based Encryption (HIBE) [i.9] is an analogous concept. A central KMS delegates the derivation of secret keys to one or more sub-KMSs. This provides more scalable and flexible user management, and still allows simplex transmission of encrypted messages without the involvement of a KMS.

5 Use cases for IBC

5.1 High-level use cases

5.1.1 Government and Enterprise

IBE, and the related Attribute Based Encryption (ABE), are particularly appropriate for government or enterprise applications where registration of employees onto a private enterprise network is likely to be handled by a human resources department and be subject to policy checks on the employee. In larger organizations it is more likely that a strict hierarchy of roles and responsibilities exist that can be associated to the identity used in IBC. It is also more likely in large organizations that there will be identity collision (e.g. two or more individuals with the same name) where strict control of e-mail and other "corporate" identities can be enforced.

IBC when allied to strict access control policies, as would be expected in any significant government or enterprise environment, is a suitable choice for file sharing and e-mail services internal to the environment.

Further, with HIBE the headquarters of a large organization could retain control of the central KMS, while allowing local provisioning of users by sub-KMSs situated in regional offices. The primary advantage of IBC is that conventional centralized information and processing models used in corporate e-mail and file sharing schemes can be straightforwardly mapped to IBC.

5.1.2 Public safety and mission critical

Public safety and mission critical applications which require secure one-to-one, group and broadcast communications are ideally suited as IBE candidates. Low-latency key management is important in order that a secure communication channel is available as soon as possible during an incident, and where there is a need to support direct communications between users if or when network infrastructure is unavailable. Existing public-safety networks such as TETRA [i.5], [i.6] typically rely on pre-configured symmetric keys, but IBE and ABE have been considered for group management by the SAFEcrypto project [i.8] and explored further in the Mission Critical Push To Talk (MCPTT) over LTE projects in 3GPP SA6 [i.29].

5.1.3 Internet of things

IBE is also suitable for some Internet of Things applications where keys can be generated centrally by the KMS and distributed to low-power embedded devices [i.9]. The primary advantage here is that the key management is lightweight and communications between devices do not need to involve the central infrastructure. In addition, the algorithmic complexity of IBC is not overly high for the encrypt/decrypt functions to be performed on a relatively constrained device.

NOTE: For IBC application in the IoT for a home environment the functionality of the home gateway or router could be extended to serve as the KMS, and in a small office environment a single multi-function server can serve as an integrated KMS.

EXAMPLE: A constrained IoT device acting as a sensor is functionally restricted to encrypt sensor readings that are forwarded to more protected devices in the network (e.g. to an address associated with processing, sensor_processor@example.org).

5.1.4 Intelligent Transport Systems

It may be possible to use IBC for some ITS applications. The model of Co-operative ITS (C-ITS) defined in ETSI TS 102 940 [i.14] uses a conventional PKI model to enable the entities to exchange signed messages (assertions of status). The difficulty in C-ITS is ensuring that the receiving party has access to the means to verify the identity of the sending party.

NOTE: C-ITS is designed not to be identity revealing but is based on proof of authority to make assertions, whereas IBC is designed to be identity revealing with additional assertions made in the complete IBC-identity in the form of policy or other meta-data.

A model of ITS using a centralized key management model as proposed in IBC should be further studied in order to establish if use cases exist for the transport sector. There may be privacy concerns if the identity of the vehicle (e.g. the globally unique registration mark) is used as the public key.

5.2 Specific functionalities

5.2.1 Access control policies towards the receiving party

5.2.1.1 Principles of access control in IBE

It is possible to implement access control in IBE where the public identity of the receiving party includes structured data representing an access control policy.

In this model, the sending party steers the ability of the receiving party to decrypt the ciphertext by setting the conditions under which the corresponding secret key can be retrieved. When the KMS receives a secret key extraction request from the receiving party for a public identity, it authenticates the receiving party, evaluates the access control policy that is embedded in the public identity, and takes an access control decision for the delivery of the secret key to the receiving party.

NOTE 1: This approach to access control requires that the KMS is not only able to authenticate the receiving party, but also knows their access rights. Access right management is an additional functionality to the normal use of IBE, where the KMS only needs to authenticate the receiving party in order to deliver the secret key that matches their public identifier.

Typical access control policies include role-based access control, mandatory access control, location-based access control, and time-based access control.

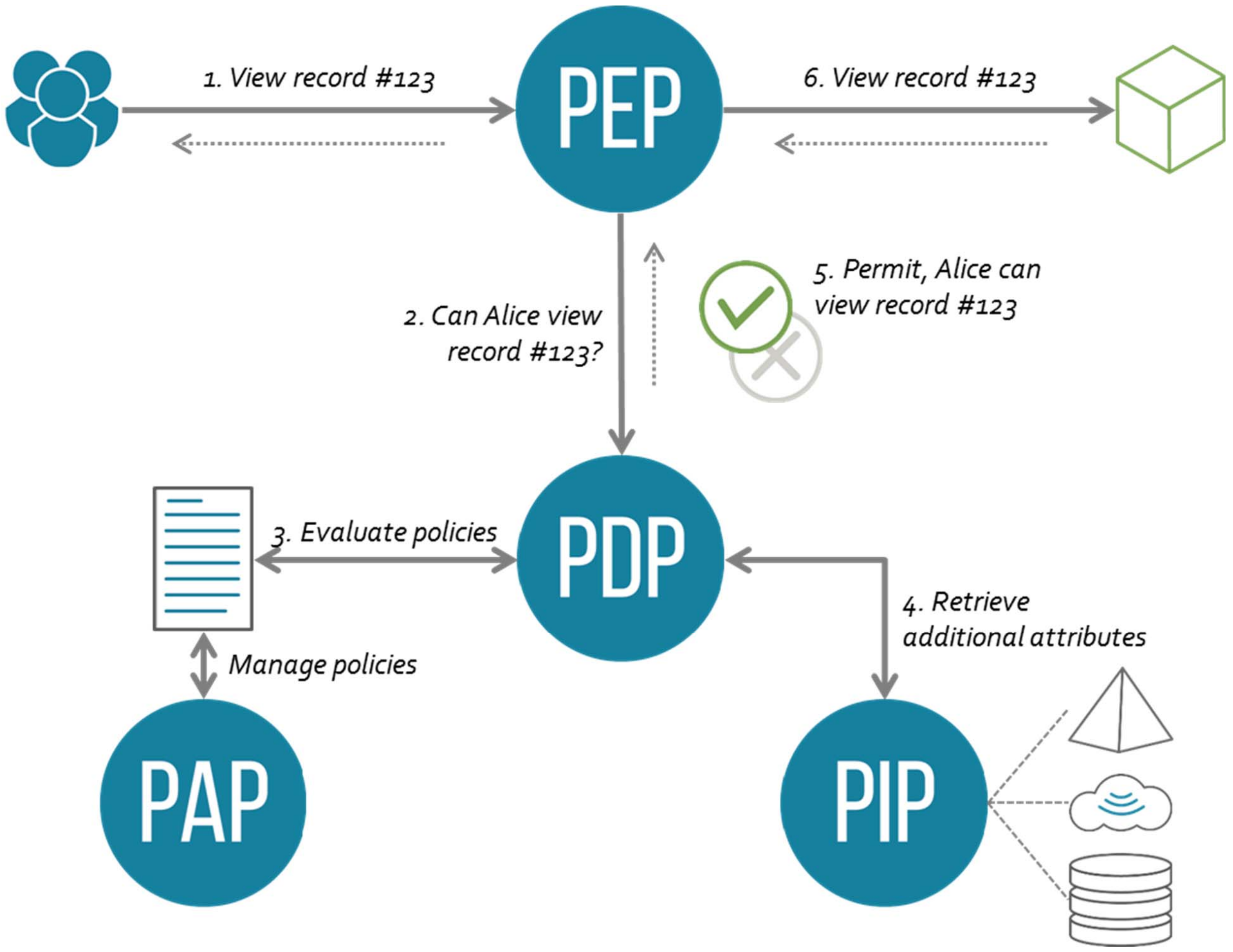
EXAMPLE 1: For a user "user1@example.org", that is a regular user and not an administrator, the KMS grants a secret key that corresponds to the embedded policy "user1@example.org || role = user", but will refuse to grant secret keys that correspond to "user1@example.org || role = admin".

EXAMPLE 2: The public identifier "user@example.org || role = financial_auditor || clearance = confidential || data-origin = accounting_headquarters || period = May_2022" allows the KMS to enforce access only if all of the conditions are met.

NOTE 2: Access control with IBE requires trust between the KMS and the sending party since the KMS cannot enforce the structure of access control policies nor the characteristics of data subjects within the IBE scheme - the sending parties act as Policy Administration Points (PAPs) as defined in [i.16]. However, the KMS can also implement its own access control policies, unrelated to what the sending party sets in the public identity.

EXAMPLE 3: A KMS policy can be to refuse secret key extraction to banned users.

NOTE 3: Efficient use of access control policies in IBE requires that the sending party and the KMS are synchronized as Policy Administration Points, i.e. regarding acceptable access control policies that can be appended to a public identifier. Otherwise, the sending party could encrypt to a public identity that the receiving party would not be able to obtain a secret key for.


1. View record #123
PEP
6. View record #123
2. Can Alice view record #123?
5. Permit, Alice can view record #123
3. Evaluate policies
PDP
4. Retrieve additional attributes
Manage policies
PAP
PIP

NOTE 1: The Policy Information Point (PIP) is the system entity that acts as a source of attribute values (i.e. a resource, subject, environment) and accessed by the PDP if needed to verify the content of access requests.
NOTE 2: The Policy Administration Point (PAP) is the system entity that manages access authorization policies.
NOTE 3: The Policy Enforcement Point (PEP) intercepts user's access request to a resource, makes a decision request to the PDP to obtain the access decision (i.e. access to the resource is approved or rejected), and acts on the received decision.
NOTE 4: The Policy Decision Point (PDP) evaluates access requests against authorization policies before issuing access decisions.
NOTE 5: The figure is from https://upload.wikimedia.org/wikipedia/commons/f/f2/XACML_Architecture_%26_Flow.png released under Creative Commons Attribution 3.0 Unported licence.

Figure 7: XACML architecture

5.2.1.2 Unicast encryption

Unicast encryption is the simplest use of IBE and relies on the sending party encrypting a message under an access control policy that resolves to a single user registered with the KMS, i.e. when the identity string is an identifier of the receiving party.

EXAMPLE: The public identity string is in the form "user@example.org".

5.2.1.3 Multicast encryption

Encryption to a known set of users (multicast encryption) is possible when the identity string provides support for the declaration of public identifiers for several receiving parties.

EXAMPLE: The public identity string is in the form "user1@example.org OR user2@example.org".

5.2.1.4 Group encryption

Group encryption differs from multicast encryption by relying on a component of the access control policy that the KMS can resolve to a group of registered users to identify recipients. The component can name a group explicitly or resolve to a group (for example, a role). However contrary to multicast encryption the sending party does not necessarily have full knowledge of which users are part of the group.

EXAMPLE: The public identity string includes meta data in the form "group = chatroom#1" or in the form "role = 'accounting department'".

5.2.1.5 Broadcast encryption

When the structure of the identity string allows, it is possible to encrypt under a string that contains neither an identifier nor role, thus achieving broadcast encryption to all users that are registered with the KMS. The KMS can still enforce access control depending on environmental parameters, such as a comparison against a date.

EXAMPLE: The public identity string includes meta data in the form "structure = broadcast" allied with data that offers any restrictions to be applied to the protected data such "not_valid_after = 01-05-2062".

5.2.2 Ephemeral public identifiers and automated revocation

Ephemeral public identifiers are a special case of access control with IBE whereby the receiving party obtains a new secret key from the KMS at regular intervals (e.g. daily) and the sending party systematically inserts a validity period in the public identifier that is compatible with the renewal interval. This approach allows for automatic revocation of secret keys and thus provides some form of mitigation in the event of a secret key compromise. In addition, it allows for automatic revocation of user rights, with granularity determined by the renewal period. This can be very useful to automatically limit exposure to sensitive data in regulated environments (such as those where health or data protection legislation apply). An interesting aspect is that the sending party does not need to query the KMS for revocation information, rather they simply need to learn the construction rules for public identities once. As long as they abide by these rules, this approach provides a simple form of data leakage prevention.

5.2.3 Encrypt into the future

Encrypt into the future is a special case of access control with IBE whereby the sending party inserts an access date in the public identity, and this date is set in the future. The receiving party will only be able to retrieve the corresponding secret key from the KMS when the date has passed. Variants exist where the access date is a limited time period in the future, or where access depends on conditions that may be verified in the future (for example, a specific achievement). Such functionality can be used, for example, to automatically unlock protected documents that have been prepared in advance.

EXAMPLE: The public identity string includes meta data in the form "valid_from = 01-03-2065", "not_valid_after = 31-03-2065", in which case the content can only be accessed in March 2065 (assuming 2065 is in the future).

5.2.4 Encrypt to future users

Since the sending party can decide on the public identity, they can encrypt messages for users or entities that are not yet registered at the KMS. In other words, they can encrypt for receiving parties that do not yet exist. This can be useful when such receiving parties are pre-registered but have not yet been fully granted access at the KMS. This can be the case for new employees of a company (where equipment and accounts are often prepared in advance of the start date in the work contract), for devices that have been pre-provisioned at time of manufacture, or for Virtual Network Functions (VNFs) in a virtualised environment.

In a case where encryption to a future user is not coordinated between the sending party and the KMS, there is a risk that the receiving party does not meet the sending party's expectation.

5.2.5 Delegation of duty

In an IBE cryptosystem where the provisions in clause 5.2.2 are applied, a user or entity can delegate duties to another user or entity by transferring secret keys and configuring forwarding within the information system. The risk of data leakage is limited by the fact that the secret keys are ephemeral, by additional access control policies (if any) in the public identity, and by the trust between the delegator and the delegatee. Typical domains where delegation of duty would be useful are e-mail domains or information management systems.

NOTE: Delegation of duty can also be performed with proxy re-encryption, described in clause 5.2.7.

5.2.6 Locally irreversible encryption

The use of an IBE cryptosystem allows for locally irreversible encryption on a user's device, if the device does not have access to an appropriate secret key and cannot obtain secret keys from the KMS (e.g. because of an access control policy related to the device identity). In such a case, the user can configure a device agent to obtain the master public key MPK from the KMS and encrypt to a public identity of their choice. In order to access the plaintext, the encrypted data is exported to a device that has access to secret keys. This allows logical zoning where devices with less physical security (e.g. in the field) can quickly encrypt sensitive data in order to limit the risk of leakage, while devices with better physical security (e.g. in an office or datacentre) can have access to the data.

EXAMPLE: A typical domain benefiting from such an approach can be constrained IoT devices acting as sensors restricted to only encrypt sensitive data sent to the more protected devices in the network.

Figure 8 illustrates one variant of this approach.

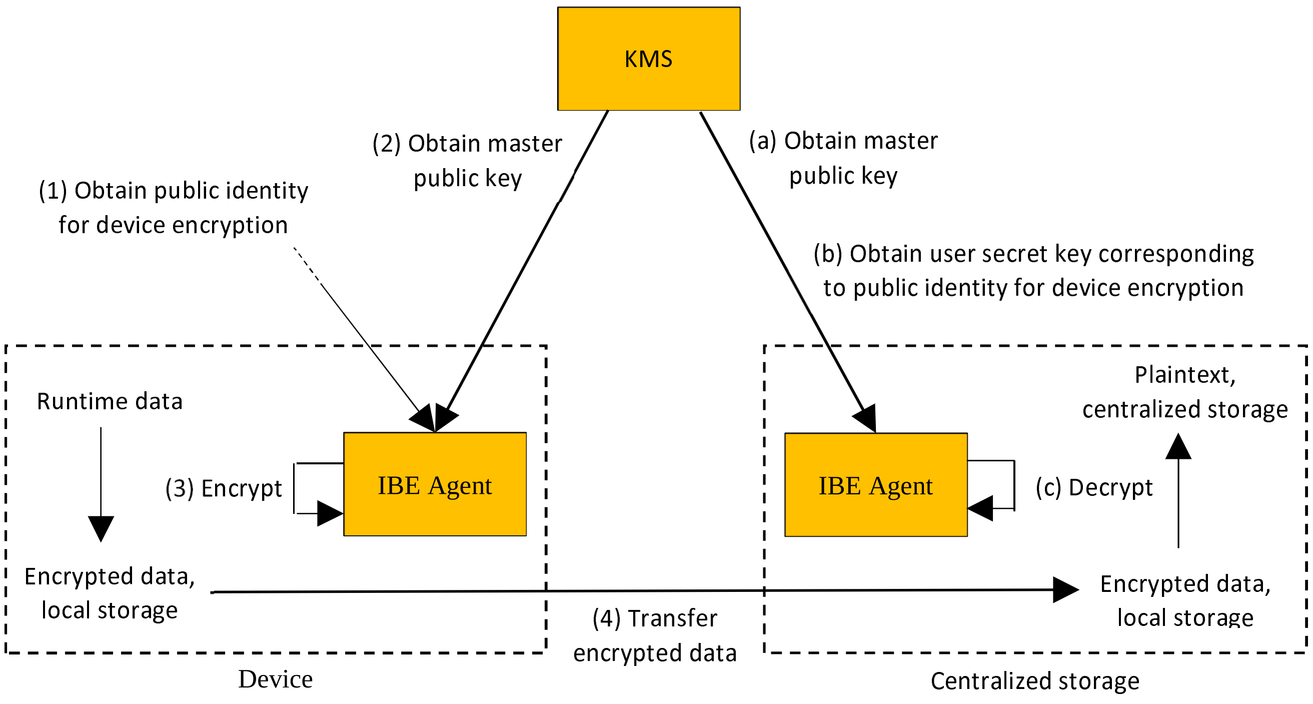

KMS
(2) Obtain master public key
(a) Obtain master public key
(1) Obtain public identity for device encryption
(b) Obtain user secret key corresponding to public identity for device encryption
Runtime data
(3) Encrypt
IBE Agent
Encrypted data, local storage
(4) Transfer encrypted data
Device
IBE Agent
(c) Decrypt
Plaintext, centralized storage
Encrypted data, local storage
Centralized storage

Figure 8: Locally irreversible encryption using IBE

5.2.7 Proxy re-encryption

Some IBE schemes support proxy re-encryption. In such a scheme a proxy can convert a ciphertext C computed for a target public identity ID into a ciphertext C' computed for a different target public identity ID'. Proxy re-encryption is useful to further automate delegation of duty, or to modify the access control policy that applies to a ciphertext.

It should be noted that in the general case the KMS is able to perform re-encryption under the same master public key and master secret key, since it can generate a secret key for any public identity. This is illustrated in Figure 9 for the case where a communication server receives a ciphertext and requests re-encryption.

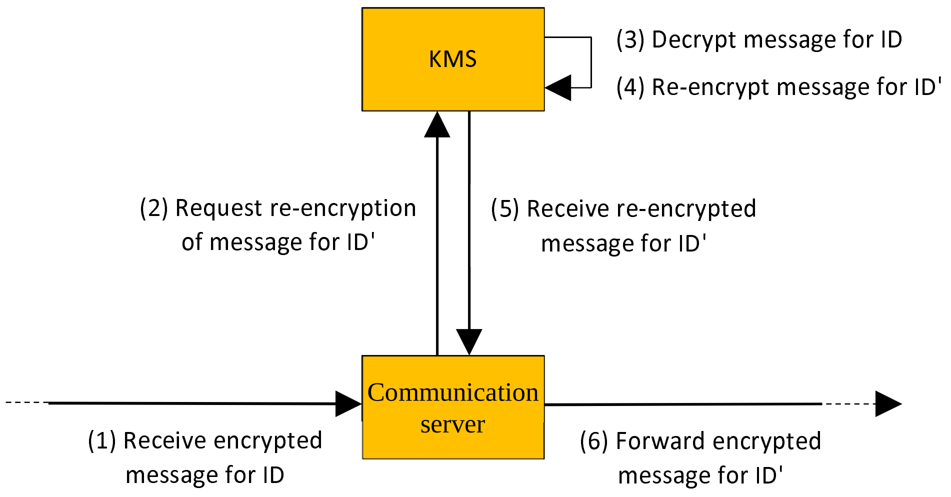

KMS
(3) Decrypt message for ID
(4) Re-encrypt message for ID'
(2) Request re-encryption of message for ID'
(5) Receive re-encrypted message for ID'
Communication server
(1) Receive encrypted message for ID
(6) Forward encrypted message for ID'

Figure 9: Proxy re-encryption using the KMS

With this approach each user can configure re-encryption for the benefit of another user directly at the KMS. However, the KMS has access to the plaintext message when performing re-encryption and this may not always be a desirable property.

With an IBE re-encryption scheme the re-encryption proxy is not fully trusted: it does not need to know any secret key and does not have access to the plaintext message. This is possible thanks to re-encryption keys, that are different from secret keys, and using two additional algorithms for which a high-level description is given below:

- **RKGen** (re-encryption key generation)**:**
  - given the master public key MPK, a secret key SK_{ID} computed by the **Extract** algorithm, and public identities ID and ID', compute the re-encryption key $RK_{ID \rightarrow ID'}$
- **Reencrypt:**
  - given the master public key MPK, ciphertext C for the public identity ID and re-encryption key $RK_{ID \rightarrow ID'}$, compute the ciphertext C' for the public identity ID'.

After the re-encryption step, the ciphertext C' can be decrypted using the secret key $SK_{ID'}$ corresponding to the public identity ID'.

NOTE: The actual input parameters for the **RKGen** and **Reencrypt** algorithms depend on the IBE re-encryption scheme.

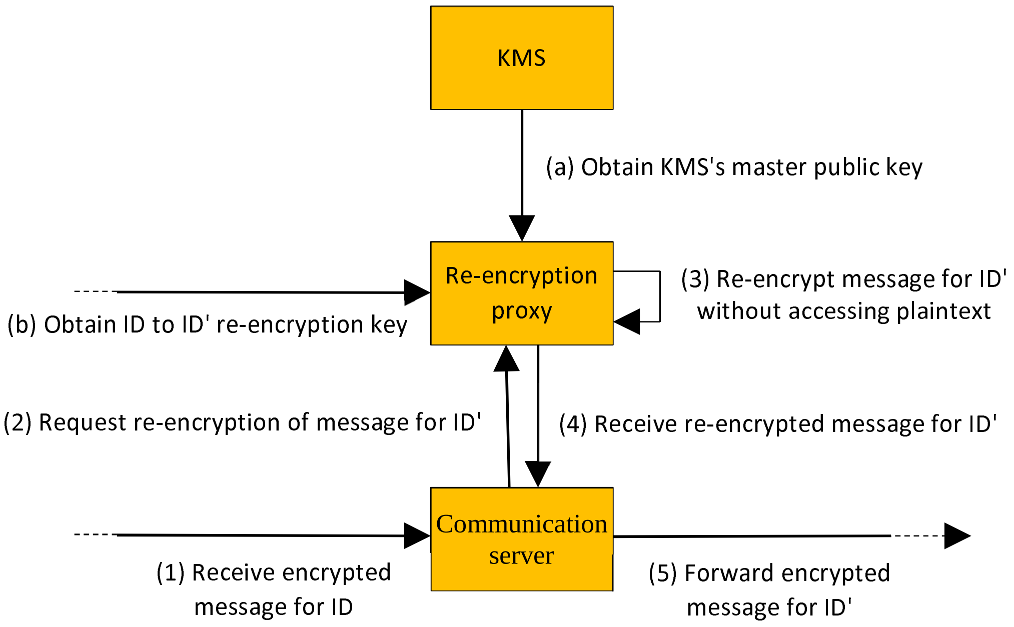

KMS
(a) Obtain KMS's master public key
Re-encryption proxy
(b) Obtain ID to ID' re-encryption key
(3) Re-encrypt message for ID' without accessing plaintext
(2) Request re-encryption of message for ID'
(4) Receive re-encrypted message for ID'
Communication server
(1) Receive encrypted message for ID
(5) Forward encrypted message for ID'

Figure 10: Proxy re-encryption using an IBE re-encryption scheme

Some desirable properties of IBE re-encryption schemes are:

- Non-interactivity, meaning that users can generate their own re-encryption keys without the help of the KMS.
- Unidirectionality, meaning that a re-encryption key from C to C' does not permit re-encryption from C' to C. Without this property it would be trivial for users to grant themselves access to a ciphertext destined for any other user.
- Multi-use capability, meaning that it is possible to re-encrypt ciphertexts consecutively (in a row) starting from an original (base) ciphertext. A multi-use scheme is often bound by an upper limit on the maximum number (depth) of re-encryptions that can be applied consecutively.

The ability to provide re-encryption without the need for the master secret key can present weaknesses, depending on the scheme, and requires careful consideration. A general risk of IBE proxy re-encryption is that the accumulation of re-encryption keys over time may allow for a re-encryption path that was not foreseen by the issuer of a re-encryption key, and another user or adversary having managed to obtain a sufficient number of re-encryption keys may be able to reconstruct such a path. This can be addressed by limiting the time validity of re-encryption keys, limiting the depth of re-encryption, and handling re-encryption keys securely.

5.2.8 Resource access control using IBE

5.2.8.1 Overview

Using IBE it is possible to implement resource access control where the access control decision is taken by the KMS, while access control enforcement remains at the resource. In this model a verifier will want proof of the identity or role of the claimant wishing to access the resource.

5.2.8.2 Challenge response authentication as pre-requisite to access control

This is achieved through a challenge-response protocol where the claimant is expected to prove ownership of a secret key SK_{ID} that matches the claimed identity or role. Access is granted when the identity or role matches the access control policy configured locally.

NOTE: This model requires that access control policies are synchronized between the KMS and the verifier.

The access control decision effectively happens when the claimant requests the SK_{ID} from the KMS. The request for SK_{ID} does not need to be synchronous to the access request. Once the claimant is provisioned, the verifier and claimant can play the challenge-response protocol offline, without requiring the KMS.

In order to mitigate the risk of SK_{ID} compromise, a time or version component can be added to access control policies in order to ensure freshness of keys. Since this goes against the benefit of offline operations, both aspects are to be balanced depending on the deployment scenario.

This model is illustrated in Figure 11 below. It is applicable both for online and physical resources alike, for example a door lock.

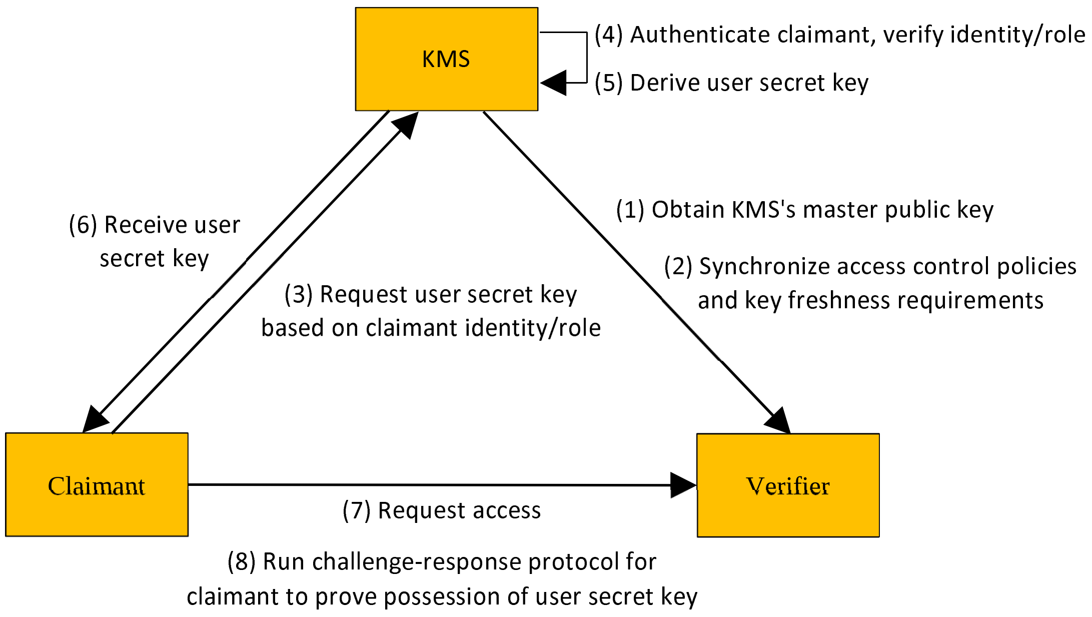

KMS
(4) Authenticate claimant, verify identity/role
(5) Derive user secret key
(6) Receive user secret key
(3) Request user secret key based on claimant identity/role
(1) Obtain KMS's master public key
(2) Synchronize access control policies and key freshness requirements
Claimant
Verifier
(7) Request access
(8) Run challenge-response protocol for claimant to prove possession of user secret key

Figure 11: Model for resource access control using IBE

5.2.8.3 Digital signature authentication as pre-requisite to access control

An alternative to the challenge response authentication is to use an identity-based signature scheme where the claimant proves possession of a signature key corresponding to their claimed identity and/or role.

NOTE: The signature-based scheme using roles or attributes is similar to that used in C-ITS [i.14].

5.3 Use of IBC for e-mail

5.3.1 Introduction

The description in this clause is for IBE applied to the storage and distribution of e-mails through a centralized server, although in the wider context of e-mail there are requirements to give assurance of the source and destination which can use IBS capabilities, and can also require confidentiality of transport in addition to using IBE in an end-to-end model.

Table 1: Classification of IBE for e-mail services deployment models

Clause	Model	Type	Remark
5.3.2	IBE secured e-mail storage under the control of the e-mail service provider	Closed	Transparent to users
5.3.3	Closed IBE cryptosystem supporting an e-mail domain and under the control of the e-mail service provider	Closed	Adapted to enterprise e-mail perimeter
5.3.4	Open IBE cryptosystem supporting an e-mail domain and under the control of the e-mail service provider	Open	Interoperable, can extend clause 5.3.3
5.3.5	Open IBE cryptosystem supporting an e-mail domain and under the control of the receiving party	Open	Interoperable, can extend clause 5.3.3

5.3.2 Secure e-mail storage under the control of the e-mail service provider

In this model, IBE is used internally within the system of the e-mail service provider. Sending parties and receiving parties under such a domain use e-mail functionalities following common practice. Notably, their e-mails go through Simple Mail Transfer Protocol (SMTP) relays that may or may not implement SMTP Secure (SMTPS) or an equivalent secure e-mail transport protocol. The e-mail service provider may be able to force SMTPS for inbound and outbound connections to and from its Message Transfer Agents (MTAs), but the service provider is not able to enforce confidentiality and integrity of messages beyond these boundaries.

Internally, however, the e-mail system supports IBE for secure **storage** of e-mails. An internal Identity Management (IdM) server assigns an identity to each user - in this case, the primary e-mail address of the user (alice@example.com say). Using an IBE proxy, inbound e-mails are encrypted to the receiving party's identity before being stored in the receiving party's mailbox (e.g. via Local Mail Transport Protocol (LMTP)). This technique can also be used when sent e-mails, drafts and sorted e-mails are managed within the e-mail service infrastructure and not locally on the user's client. When a user wishes to retrieve their e-mail (e.g. on their local Message User Agent (MUA) via Post Office Protocol version 3 (POP3) or Internet Message Access Protocol version 4 (IMAP4), or on their webmail interface), an IBE decryption proxy is used to decrypt stored e-mails on the fly. Whether the IBE decryption proxy can obtain the appropriate SK_{ID} of the user depends on the successful authentication of the user through the e-mail retrieval method or through a third-party authentication service, e.g. based on OAuth (IETF RFC 6749) [i.17].

Figure 12 illustrates this scenario.

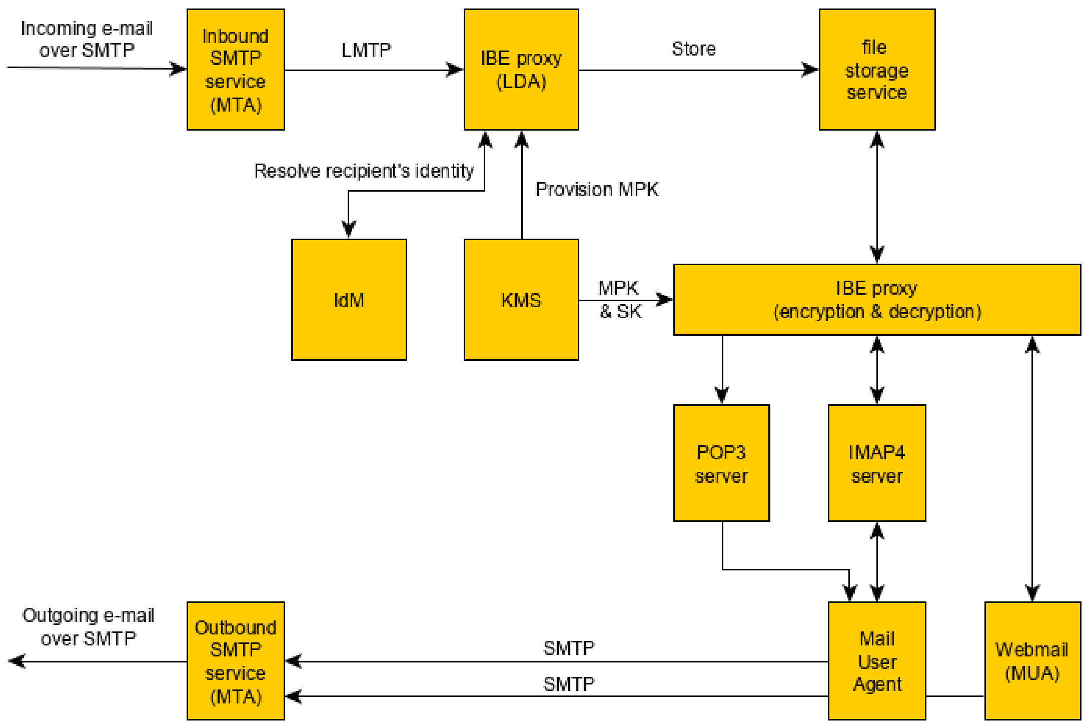

Incoming e-mail over SMTP
Inbound SMTP service (MTA)
LMTP
IBE proxy (LDA)
Store
file storage service
Resolve recipient's identity
Provision MPK
IdM
KMS
MPK & SK
IBE proxy (encryption & decryption)
POP3 server
IMAP4 server
Outgoing e-mail over SMTP
Outbound SMTP service (MTA)
SMTP
SMTP
Mail User Agent
Webmail (MUA)

Figure 12: Closed IBE cryptosystem supporting e-mail storage

In this scenario, the users are unaware of the use of IBE, therefore IBE does not provide any form of end-to-end security at the e-mail application layer - users can only benefit from hop-by-hop security at the transport layer depending on the configuration of relays and delivery methods. However, IBE provides an additional layer of security for e-mail storage within the e-mail service provider's system. By segregating the KMS, the IBE encryption proxy, and the IBE decryption proxy from the storage system and forcing a successful user authentication in order to obtain SK_{ID}, an additional layer of defence is provided against an attacker that would gain access to the e-mail storage servers.

5.3.3 Closed IBE cryptosystem supporting an e-mail domain and under the control of the e-mail service provider

In this model the MUAs of users within an e-mail domain are provisioned with IBE support and can access the KMS of the e-mail service provider. E-mail communications between users of the e-mail service provider can always be protected end-to-end at the e-mail application layer.

NOTE 1: This model forms a practical implementation of the general model of IBE introduced in clause 4 and applied to e-mail.

The e-mail provider does not advertise IBE support outside of its network but uses a transparent IBE encryption proxy connected to its inbound SMTP relay and performs on-the-fly IBE of message to the receiving party's public identity, as in the model illustrated in Figure 13.

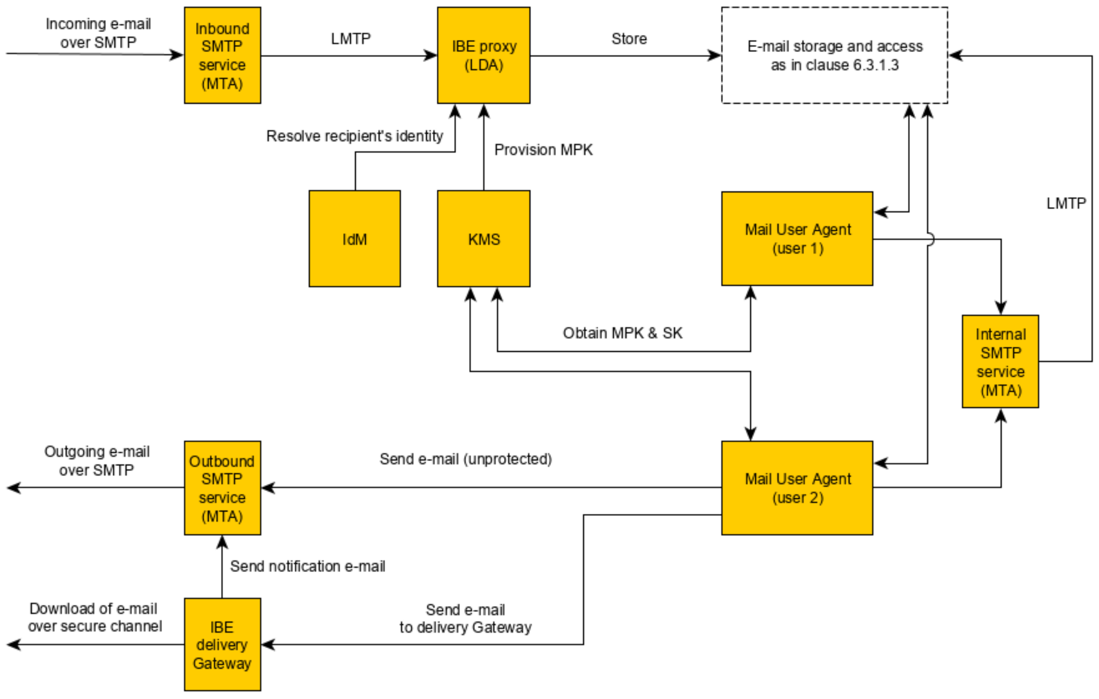

Incoming e-mail over SMTP
Inbound SMTP service (MTA)
LMTP
IBE proxy (LDA)
Store
E-mail storage and access as in clause 6.3.1.3
Resolve recipient's identity
Provision MPK
IdM
KMS
Mail User Agent (user 1)
LMTP
Obtain MPK & SK
Internal SMTP service (MTA)
Outgoing e-mail over SMTP
Outbound SMTP service (MTA)
Send e-mail (unprotected)
Mail User Agent (user 2)
Send notification e-mail
Download of e-mail over secure channel
IBE delivery Gateway
Send e-mail to delivery Gateway

Figure 13: Closed IBE cryptosystem supporting an e-mail domain, with options (a) and (b)

NOTE 2: For simplicity, IBE for e-mail storage and access is not fully described.

When users send a message to an external user (for which the e-mail address is not managed by the e-mail service provider), a copy of the message can be encrypted to a pseudonymous identity for internal storage, but the original is to be delivered to the receiving party. There are several options to address this step:

a) The original message goes through the outbound SMTP relay(s) in cleartext or encrypted using a method other than IBE and that has been negotiated out-of-band by users. This is the least intrusive method as it relies on already existing tools and infrastructure.

b) The original message is stored on an IBE delivery gateway and the receiving party is notified that an e-mail awaits. While this method helps in ensuring that delivery of the message is done over a secured transport, the security issues lie with the notification e-mail. In addition, out-of-band delivery of the e-mail message will make it hard for the receiving party to integrate the message into their usual e-mail flow.

c) The original message is encrypted using IBE to the e-mail address of the receiving party set as public identity and using a KMS of the outgoing e-mail service provider's choosing. While this method helps in spreading the use of IBE, it has the strong disadvantage of forcing the hands of receiving party.

Despite these limitations, this model closely matches the e-mail security perimeter in business scenarios.

It is therefore the model of choice for enterprise deployment.

5.3.4 Open IBE cryptosystem supporting an e-mail domain and under the control of the e-mail service provider

In this model, an e-mail service provider can advertise whether they support IBE for a specific e-mail domain or a specific e-mail address, where the IBE service provider is selected by the e-mail service provider. Such advertisement can be done through a well-known IBE advertisement service that is resolvable, for example, from a dedicated DNS record associated to the e-mail domain name, or from specific URI construction rules based on the domain name. The well-known IBE service is an interface to the KMS of the IBE service provider supporting the e-mail service provider.

Accordingly, a receiving party that wishes to use IBE would enrol through the well-known IBE service and, after authentication, register their e-mail address as public identifier for IBE. At a later stage, it is possible for the receiving party to deactivate IBE or modify other preferences. If the receiving party does not enrol to the well-known service for IBE or deactivates it through preference configuration, then it is assumed that it does not accept messages protected with IBE.

A sending party that supports IBE encryption can query the well-known IBE service and request the status of the receiving party with regard to IBE, and of other information such as the MPK and public identity to use. When the receiving party accepts IBE as announced by the well-known IBE service, then the sending party can encrypt using IBE. In the negative case, the sending party assumes that another form of e-mail encryption, or cleartext, is used when sending messages to the receiving party.

NOTE 1: The term "cleartext" is used in the present case to convey the meaning of "not encrypted", as "plaintext" has a specific meaning in the context of e-mail.

This is illustrated in Figure 14 below.

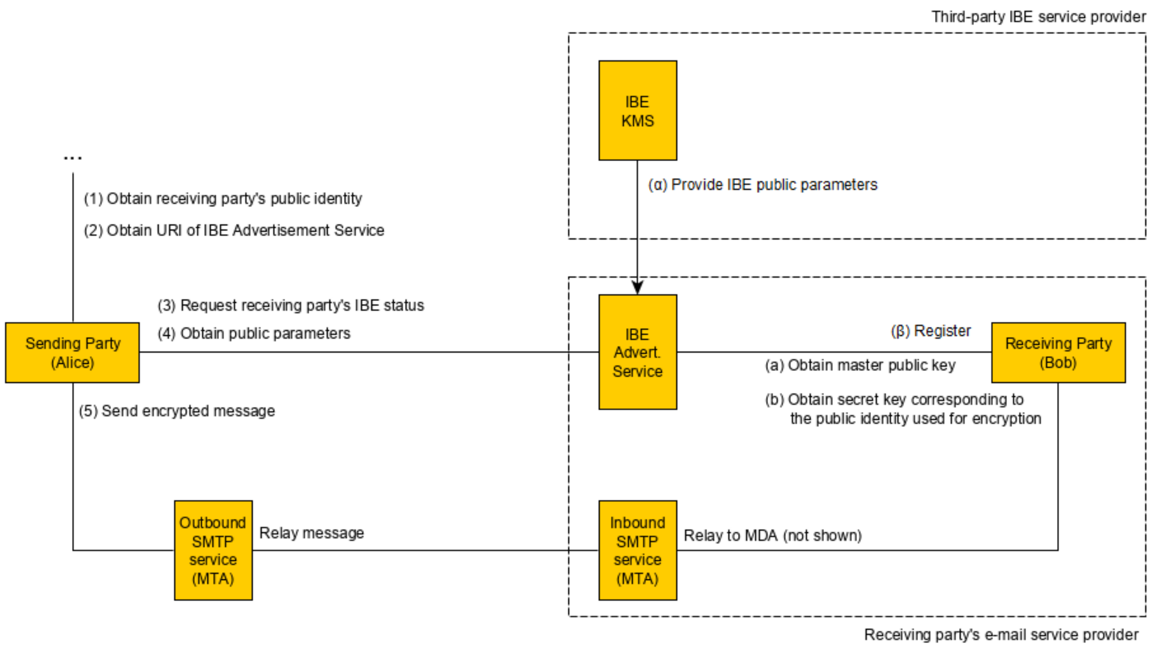

Third-party IBE service provider
IBE KMS
(α) Provide IBE public parameters
...
(1) Obtain receiving party's public identity
(2) Obtain URI of IBE Advertisement Service
(3) Request receiving party's IBE status
(4) Obtain public parameters
Sending Party (Alice)
IBE Advert. Service
(β) Register
(a) Obtain master public key
(b) Obtain secret key corresponding to the public identity used for encryption
Receiving Party (Bob)
(5) Send encrypted message
Outbound SMTP service (MTA)
Relay message
Inbound SMTP service (MTA)
Relay to MDA (not shown)
Receiving party's e-mail service provider

Figure 14: Open IBE cryptosystem under the control of the e-mail service provider

The well-known IBE service can not only be used by receiving parties to manage their IBE preferences, but also to handle key management for sending parties. For example, it can announce a new MPK or new constraints for the construction of the public identity - thus providing SK_{ID} management. A validity period can be attached to replies in order to steer the caching and key renewal mechanisms of the sending party.

It is noted that more manual methods for receiving parties to advertise acceptance of IBE exist, such as providing their MPK and other metadata on their personal website or within a vCard [i.18]. While these methods are practical on an individual basis, they are not user-friendly at scale and are therefore not studied further in the present document.

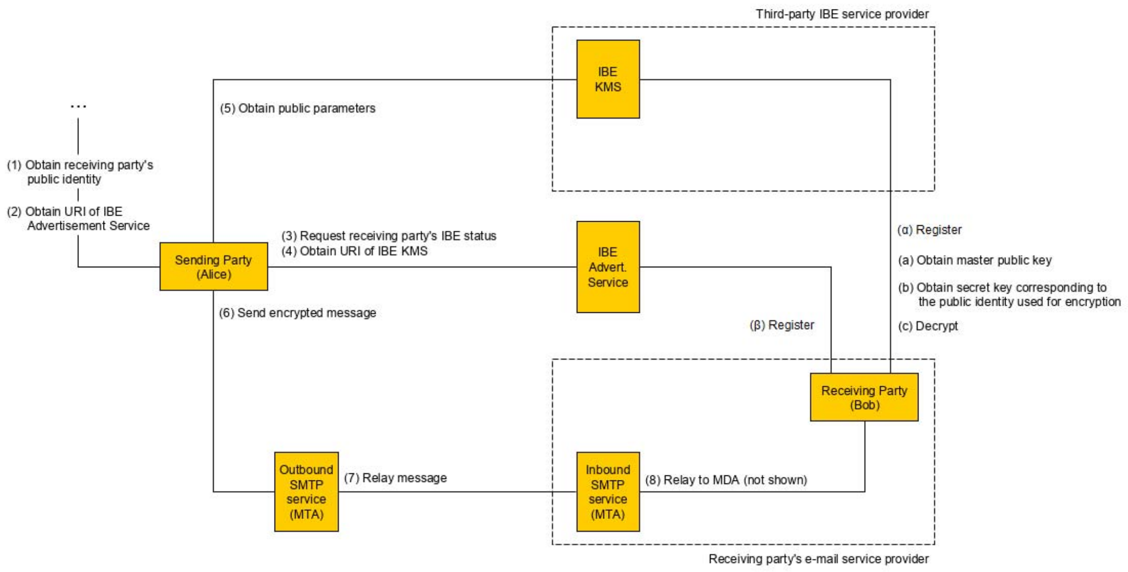
The model shown in Figure 14 is applicable both to e-mail service providers and individuals hosting their own e-mail service together with their own KMS (self-hosting).

NOTE 2: The fully open model requires native IBE support in the MUA.

5.3.5 Open IBE cryptosystem supporting an e-mail domain and under the control of the receiving party

In this model, any IBE service provider can publicly provide access to their KMS - this includes individuals that self-host a KMS. The well-known IBE advertisement service (introduced in clause 5.3.4) is an IBE service provider management front-end. It does not provide access to a KMS but allows resolution of a KMS from the receiving user's public identifier. Receiving parties can register their IBE service provider of choice and associate one or more public identifiers (e-mail addresses under a domain name of the e-mail service provider) to it.

The resolution process for the sending party is similar to that of the previous clause and is illustrated in Figure 15.



Third-party IBE service provider
IBE KMS
...
(5) Obtain public parameters
(1) Obtain receiving party's public identity
(2) Obtain URI of IBE Advertisement Service
Sending Party (Alice)
(3) Request receiving party's IBE status
(4) Obtain URI of IBE KMS
IBE Advert. Service
(6) Send encrypted message
(α) Register
(a) Obtain master public key
(b) Obtain secret key corresponding to the public identity used for encryption
(c) Decrypt
(β) Register
Receiving Party (Bob)
Outbound SMTP service (MTA)
(7) Relay message
Inbound SMTP service (MTA)
(8) Relay to MDA (not shown)
Receiving party's e-mail service provider

Figure 15: Open IBE cryptosystem under the control of the receiving party

NOTE: The KMS is normally not directly accessible but is protected by a frontend providing access control, for simplification this is not represented in the above figure.

A variant of this model is to have a set of well-known and globally available IBE resolver services whereby a receiving party can register their public identifier (e-mail address) and associated KMS.

5.3.6 Analysis

Unless all SMTP relays involved in the delivery of an e-mail support IBE, and delivery of secret keys is coordinated with a global KMS, then IBE can only provide security to the e-mail body. This is because the e-mail envelope is needed by the transport layer for routing the message, and by various middleboxes that process e-mail (such as spam detection systems). It is unrealistic to expect such global coordination for a system that was designed for the interoperability of autonomous entities. If an SMTP relay were to opportunistically apply IBE to the e-mail envelope, this would lead to serious interoperability problems as SMTP operators that do not support IBE would be unable to relay incoming messages to their users. This can also cause problems in regions where the use of cryptography is regulated.

However, once a message is received and stored, IBE can be used to provide data at rest protection for the complete message, envelope and body.

NOTE: The IBE proxy needs to support the proxied protocol in detail - for example, the IMAP4 APPEND command. This complexity is needed to enable e.g. cloud services to process e-mail by delegation of the receiving party. Otherwise, decryption happens at the MUA of the receiving party - which is often a desired feature.

The introduction of the IBE advertisement service in clause 5.3.4 raises the question of trust towards services provided in support of the IBE cryptosystem in a public infrastructure. Indeed, discovery mechanisms and retrieval of cryptosystem parameters can be targeted by an attacker causing service disruption or redirecting users to illegitimate cryptosystems.

It is noted that the concept of an IBE advertisement service can extend the architecture introduced in clause 5.3.4 and thus provide full interoperability to the enterprise model.

6 Protocols to support IBE

6.1 Message protection using identity-based cryptography

6.1.1 Message structure

An exemplary structure for a message protected with IBC is given in Figure 16 below. In this example, it is assumed that part of the message is confidentiality protected using IBE, and that the overall message is integrity protected and authenticated using IBS.

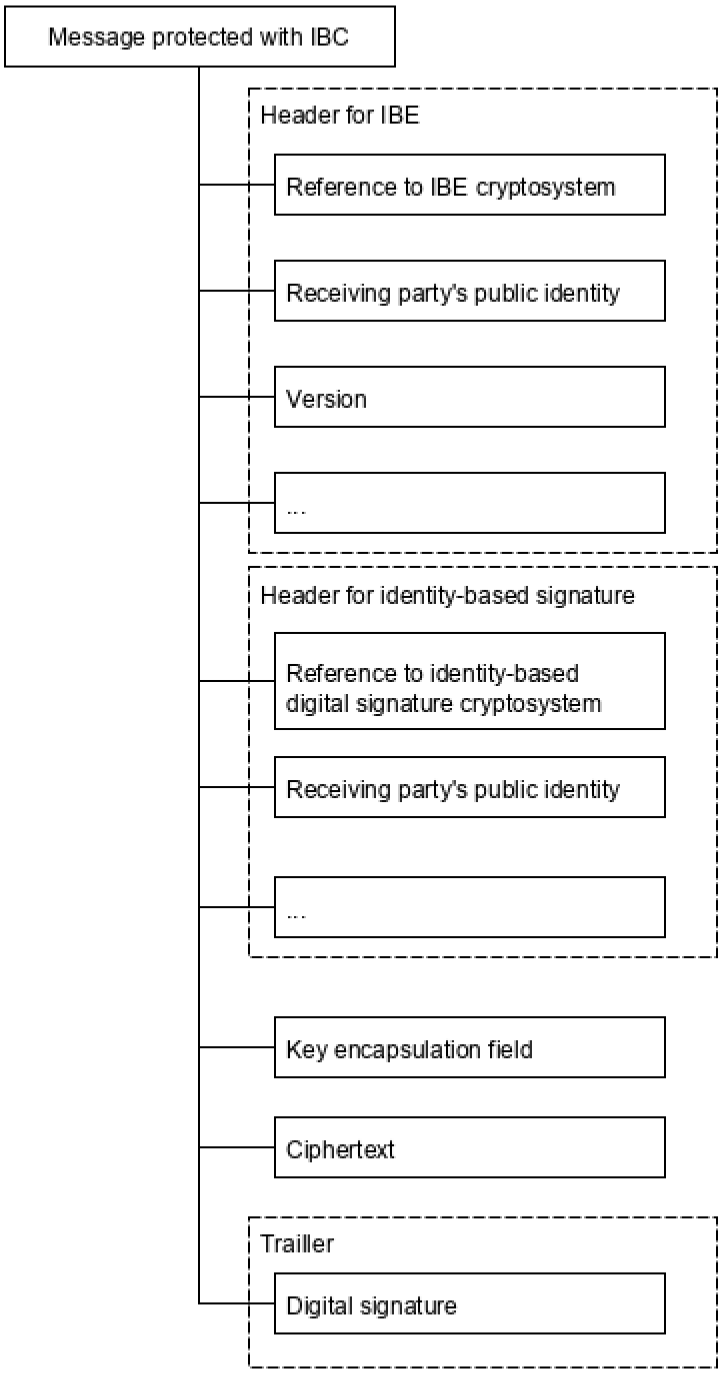

Message protected with IBC
Header for IBE
Reference to IBE cryptosystem
Receiving party's public identity
Version
...
Header for identity-based signature
Reference to identity-based digital signature cryptosystem
Receiving party's public identity
...
Key encapsulation field
Ciphertext
Trailler
Digital signature

Figure 16: Exemplary structure of a message protected with IBC

NOTE 1: While exemplary, this structure closely matches what would be expected in a practical implementation.

On a high level, a message protected with IBC can consist of the following structure:

- headers;
- an optional key encapsulation field;
- the ciphertext, containing the confidentiality protected data; and
- a trailer.

The ciphertext is a binary object which can only be decrypted by the IBE secret key provided by the KMS. For performance reason, hybrid encryption can be used: the ciphertext is produced by a symmetric cipher, and only the corresponding symmetric key is encrypted (or encapsulated) using IBE and stored in the key encapsulation field. The receiving party still has to have the IBE secret key in order to decrypt the ciphertext. This is the approach shown in Figure 16.

The header contains information needed by the receiving party to decrypt the ciphertext and verify the signature located in the trailer:

- A header for IBE, containing:
  - A reference to the IBE cryptosystem used.

    This reference is necessary for the receiving party to determine whether it can decrypt the ciphertext and request the MPK and SK_{ID} corresponding to the proper MSK. In closed systems, this can be a simple identifier that is then resolved through internal configuration. In open systems, such an identifier can be completed with a URI to the KMS (or its front-end), an identifier for the IBE scheme and other public parameters.
  - The receiving party's public identity under which IBE was applied. The public identity can contain the receiving party's public identifier, an access control policy set by the sending party or both.
  - A version field to the receiving party's public identity.
  - Any additional parameters required by the IBE scheme.
- A header for IBS, containing:
  - A reference to the IBS cryptosystem used.

    This reference is necessary for the receiving party to determine whether it can verify signatures and request the MPK and verification key corresponding to the proper MSK. In closed systems, this can be a simple identifier that is then resolved through internal configuration. In open systems, such an identifier can be completed with a URI to the KMS (or its front-end), an identifier for the digital signature scheme, and other public parameters.
  - The public identity of the sending party. The public identity can contain the sending party's public identifier, an access control policy set by the sending party or both.

NOTE 2: The possibility for the sending party to define an access control policy - i.e. to define which receiving party will be able to obtain the verification key from the KMS, and therefore be able to verify a signature - is mentioned for completeness.

  - Any additional parameters required by the IBS scheme.

6.1.2 Risk analysis and possible mitigations

6.1.2.1 Compromise of IBE secret key corresponding to a public identity

An attacker could extract a secret key from a compromised device. As a consequence, an attacker will be able to decrypt any past and future messages they are able to intercept. The probability of such an incident becomes higher as the number of involved devices increases, as is the case in group communication.

To mitigate this, it is possible to implement automated revocation as described in clause 5.2.2. Automated revocation can at least protect the confidentiality of future messages and, to an extent, that of messages in the past - when receiving devices regularly delete old secret keys.

Automated revocation can be applied to secret keys allowing decryption of the message ciphertext as well as to secret keys allowing decryption of the message header (header encryption is explained in clause 6.1.2.2).

Automated revocation can be implemented using a versioning field added to the public identity. The version field is set by the sending party and can take various form, such as a validity period (potentially under instruction from the KMS) or a random value. When random values are used, the sending party does not depend on the KMS for revocation purposes. However, random values do not provide any semantic for the secret key validity, making cache management difficult for the receiving party.

6.1.2.2 Exposure of public identities and other header information during communication

When the transport path between the sending party and the receiving party does not fully protect the confidentiality of messages there is a risk that an attacker can intercept messages, in which case they will have access to information contained in the header. This can cause security issues, for example:

- When both the sending party's and receiving party's public identities are present, it is revealed that a communication between the two parties has taken place. In certain situations, this can be considered a privacy breach or a data protection breach.
- Analysis of the public identity can reveal access control policies in use within the cryptosystem. This can, for example, reveal the access control model, or reveal groups or roles the receiving party belongs to, allowing the attacker to initiate a targeted social engineering attack.

Header encryption can be used to mitigate these risks. For example, using the same IBE cryptosystem, sensitive header fields can be encrypted to a group identity. The corresponding group secret key is distributed by the KMS to devices allowed to use the cryptosystem. As explained in clause 6.1.2.1, such a secret key can be compromised, and therefore should also be subject to automated revocation.

This mitigation is not illustrated in clause 6.1.

6.1.2.3 Modification of header and ciphertext by an attacker

An attacker successfully mounting a man-in-the-middle attack can modify the header and ciphertext in messages in transit. The usual mitigation is to append a digital signature to the ciphertext and message header - which can be done using IBS schemes.

This mitigation is illustrated in clause 6.1, with the addition of a trailer and a header for digital signature.

6.2 User registration and provisioning

Interfaces and protocols are required to enable the registration of users to the KMS and the provisioning of the MPK and domain parameters to their IBE client. This can be done by an administrator of the KMS, or directly by IBE clients provisioned to interact with the KMS. In cases where the sending party only needs to learn about the MPK and publicly available access control policies, the provisioning of the MPK and domain parameters can run over an anonymous protocol. This assumes that the sending party does not need to authenticate itself, or that authentication of the sending party is provided by non-IBE means.

For other cases, the registration and provisioning protocols require authentication of the sending party, and a process might be required to confirm the identity of the sending party before providing credentials.

6.3 Secret key request and update

Receiving parties that are registered at a KMS rely on a protocol to request and obtain secret keys. Such a protocol will usually rely on an access control management system and require that the receiving party be authenticated. The IBE client of the receiving party can also use such a protocol to periodically poll the KMS for new secret keys, possibly based on key revocation parameters set by the KMS.

Update of secret keys can also be initiated by the KMS and pushed to IBE clients through a notification protocol.

6.4 Client update

Mechanisms are required to keep IBE clients up to date on the status of the IBE cryptosystem: deactivation of secret keys, cryptoperiod of secret keys, migration to a new MPK/MSK pair and related domain parameters, and so forth. These can rely on a request-response protocol initiated by the IBE client (poll mode) or by driven by the KMS e.g. via notifications (push mode).

6.5 Receiving party resolution

This optional mechanism would allow the sending party to verify the existence of a public identifier with the KMS and obtain related access control policies before sending a message to the receiving party.

NOTE: Data protection considerations apply when the public identifier is personal data and the KMS supports such resolution protocol.

6.6 Administration interface

Interfaces and protocols are required to enable an administrator of the KMS to manage the user database, the access control policies, and the cryptosystem. Such an interface can integrate a journaling function, in particular for the generation and state transitions of secret keys.

6.7 Key authorization protocol

A mechanism is needed to inform an IBE client when a secret key transitions to the authorized state. Several methods exist for this offering various level of security assurance. For example, the IBE client can examine the public identity attached to the secret key to determine such information by itself. Transition into the authorized state can also be notified as part of the client update protocol or through a notification protocol. It is also possible to use key wrapping and deliver the secret key wrapped to the IBE client, while the decryption key will be delivered separately.

Annex A: Frequently Asked Questions

Question	Answer
In a single paragraph what is IBC?	Identity-based Cryptography is a development of asymmetric cryptography that uses a simple well-known identity (e.g. an e-mail address) as the public key, and can operate without provision of a discrete public key management infrastructure. IBC then offers the ability to integrate capabilities of encryption and signature directly with the depending function such as e-mail, within a managed domain.
Can any string be used as the public key?	Essentially yes. The idea is that the public key is a well understood identifier, often with closely associated semantic meaning. Thus, an e-mail address (bob@somedomain.com) pairs the identifier with well understood semantic context, e-mail. As there is no cryptographic value in the public key it is recommended that the public key has clear semantic content (e.g. e-mail intended for a specific person or role).
What are the significant differences between IBC and conventional asymmetric cryptography?	**1: Key binding to an identity** In conventional public key cryptography each user has a key pair consisting of matched public and private keys, where the private key is generated first via a random process and the public key is derived from the private key via a mathematical function that is hard to invert. Public keys constructed in this way are pseudo-random and have no intrinsic meaning. Consequently, it is usually necessary to bind the public key to a public identifier associated to the user. The binding is typically achieved by including the public key and identifier in a certificate that is digitally signed by a trusted third party, such as a Certification Authority (CA), during a certification process. In contrast, in identity-based cryptography the public identity is chosen first and the secret key is computed from the public identity and master public key. This means that a user's public identity can be designed to have some intrinsic semantic value. Specifically, it can be chosen to be the representation of a public identifier associated with the user (e.g. an e-mail address), and can contain additional data for steering the cryptosystem, such as imposing conditions on the recipient via access control policies. **2: Cryptographic key generation** Another important difference between traditional public key cryptography and identity-based cryptography is that the user's secret key needs to be derived from their identifier by a trusted third party such as a Key Management Service (KMS). In other words, the end user does not generate their own secret key.
What are the advantages of IBC over conventional asymmetric cryptography?	One of the main advantages of identity-based cryptography is that it offers the possibility of lightweight key management without the need for certificates or a full Public Key Infrastructure (PKI). If Alice knows one of Bob's identifiers (e-mail address, say) under an IBE cryptosystem then she would use this and the master public to encrypt a message to him. In contrast if Alice wants to send Bob a message protected by a more conventional public key encryption scheme where the public keys are managed by a PKI, she first needs to obtain Bob's public key, either directly from Bob or from a central certificate repository in the form of a Public Key Certificate (PKC). It can be argued that this conventional approach involves more work.

Annex B: Risk analysis and deployment considerations for KMS

The consideration of risk here addresses only the cryptographic and supporting properties of IBC, it does not consider wider risks in deploying file-sharing or e-mail services, or any other service in which IBC is deployed.

In cryptographic terms IBC is, as of 2022, still relatively novel, and the number of deployments is relatively small. The comparisons here are made against conventional asymmetric cryptography.

In the fully open model, the security of IBE relies on the security of the well-known IBE service. In particular, it is expected to be verifiable that:

- the well-known IBE service is legitimate for a given domain; and
- the information provided by the well-known IBE service cannot be modified by a third-party.

In the IBC protected e-mail context unless all SMTP relays involved in the delivery of an e-mail support IBE, and delivery of secret keys is coordinated with a global KMS, then IBE can only provide security to the e-mail body. This is because the e-mail envelope is needed by the transport layer for routing the message, and by various middleboxes that process e-mail (such as spam detection system). It is unrealistic to expect such global coordination for a system that was designed for the interoperability of autonomous entities. If an SMTP relay were to opportunistically apply IBE the e-mail envelope, this would lead to serious interoperability problems as SMTP operators that do not support IBE would be unable to relay incoming messages to their users. This can also cause problems in regions where the use of cryptography is regulated.

However, once a message is received and stored, IBE can be used to provide data at rest protection for the complete message, envelope and body.

An inherent property of IBC is that it has explicit key escrow (keys are maintained by a trusted 3rd party). The risk here is that non-repudiation services are not available as the necessary guarantees cannot be offered.

Annex C: Lifecycle of an IBC cryptosystem

C.1 Introduction

The present annex addresses key lifecycle using the notions defined in NIST SP 800-57 Part 1 rev 5 [i.7]. The structure of the IBC's lifecycle consists of four sequential phases.

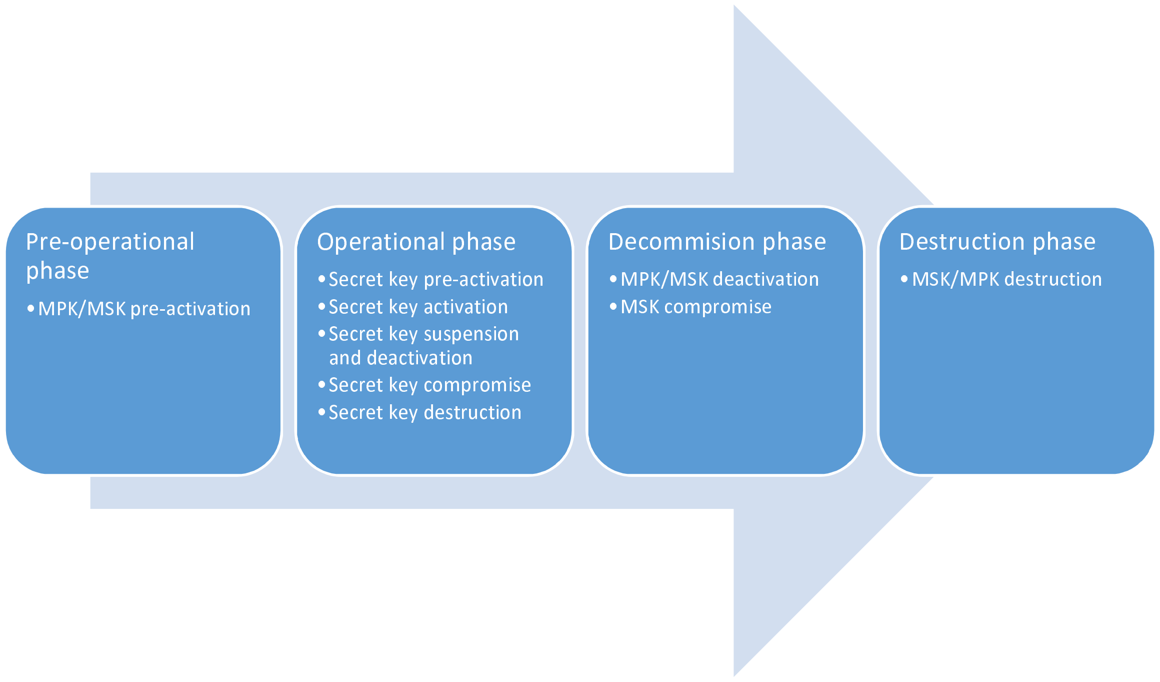


Figure C.1: IBC cryptosystem lifecycle

C.2 Pre-operational phase

C.2.1 MPK/MSK pre-activation

In this state the KMS is provisioned with a new MPK/MSK pair generated through the **Setup** algorithm of the selected IBE scheme. While the pair is not yet taken into use, metadata is prepared to identify the new MPK/MSK pair and allow future management tasks by the KMS and IBE clients (for example, decommissioning and key renewal). Metadata related to the key pair can include the following information:

- identification and versioning of the cryptosystem instance;
- identification of the IBE scheme; and
- domain parameters required for the operation of the **Extract**, **Encrypt** and **Decrypt** steps.

While the MPK and MSK have not been authorized for use at this stage, IBE clients can be provisioned with the MPK, domain parameters, as well as initial rules for the construction of public identity strings (for example, available groups and roles, access control policy versioning to ensure secret key expiration).

C.3 Operational phase

C.3.1 Introduction

In the operational phase the MSK and the MPK are authorized for use. If this event happens after the IBE clients have been provisioned, then an online procedure (e.g. a notification via a push mechanism) or a local procedure (e.g. unlocking through a security code) is necessary to authorize the MPK for use.

C.3.2 Secret key pre-activation

When a new user is registered on the KMS, the KMS can generate one or more secret keys SK_{ID} based on applicable access control policies through the **Extract** step. Metadata related to the secret can include domain parameters and the corresponding public identity (in order for the IBE client to select the appropriate secret key for the **Decrypt** step).

New secret keys can also be generated when new access control policies are defined for a user, or due to automatic key revocation mechanisms defined in clause 6.2.2.

Finally, new secret keys can be generated in advance to the validity period of the related public identity and distributed to IBE clients. These secret keys are also in the pre-activated state.

In this state, the keys are not authorized for use and are possibly not yet delivered to the IBE client of the related user.

C.3.3 Secret key activation

In this state the secret key SK_{ID} is authorized for use. This requires the provisioning of the secret key on the IBE client, and an online or local procedure to inform the IBE client about the authorization. Alternatively, time information embedded in the public identity can allow automatic (implicit) activation.

C.3.4 Secret key suspension and deactivation

In the present document the suspended and deactivated states are considered to be the same, the explanation for this follows. In IBE the suspended state translates into the fact that a public identity, which would normally be usable, is suspended. For example, the secret key is protected by the automatic expiration mechanism introduced in clause 6.2.2, but the secret key is suspended before the expiration time. Accordingly:

- the IBE client of the sending party is to be informed that it cannot use the public identity related to the secret key any longer to encrypt messages; and
- the IBE client of the receiving party is to be informed that:
  - it cannot accept messages that are sent past the time the secret key is suspended; and that
  - it can process messages that have been encrypted under the suspended public identity before the time the suspension went into effect (e.g. for data recovery purposes).

A secret key that has been suspended can be reinstated in the active state. Because in IBE the suspended state applies to the public identity of a receiving party, all potential sending parties are to be informed that the secret key is reinstated in the active state. Since this can be very resource consuming, it can be beneficial to avoid this step and instead force IBE clients into using a new public identity altogether. This can be achieved by using a public identity version number in addition to the policy component used for automatic secret key revocation. In effect, this means that the secret key transitions directly into the deactivated state.

EXAMPLE: Let a public identity be "user1@example.org || not-valid-after 2020-Dec || version 1". Suppose that the corresponding secret key is to be suspended. In order to allow encryption to the entity represented by the public identifier "user1@example.org", a new secret is generated, that correspond to the public identity "user1@example.org || not-valid-after 2020-Dec || version 2". This public identity is communicated to the sending parties, while the secret key is provisioned to the receiving party. Consequently, the first secret key can immediately transition to the deactivated state.

NOTE: A secret key can be deactivated automatically when the principles of clause 6.2.2 are followed (in the example above, when the current time passes beyond the value of the not-valid-after component). In such case there is no need to inform the IBE clients of the deactivation since the time at which the deactivation takes place is embedded in the key metadata. Informing the IBE clients remain necessary if the secret key is prematurely deactivated as above. A well-defined period of validity can efficiently reduce the need to notify IBE clients of premature key deactivation (revocation).

C.3.5 Secret key compromise

When it is apparent that a secret key of a receiving party is in the compromised state, all sending parties that are susceptible of sending protected message to said receiving party are to be notified immediately (or as soon as possible). Once a new secret key can be generated and provisioned to the IBE client of the receiving party, these sending parties are to be notified of the new public identity to use. One way to achieve this is to use the public identity versioning scheme introduced in clause C.3.4.

C.3.6 Secret key destruction

In this state a secret key is destroyed. This has the same effect on IBE clients as clause C.3.4 with the addition that the IBE client of the related receiving party (for which the secret key is destroyed) is to be informed that:

- it cannot process any message that has been encrypted under the public identity that corresponds to the destroyed secret key;
- requests for generating a secret key under the same public identity will fail; and that
- it is to destroy the key locally.

NOTE 1: Once a secret key is destroyed the IBE client has confirmation that the related public identity is not valid any longer and can cache such information.

The secret key is to be destroyed on the KMS, on the IBE client(s), and on any intermediary that has acted as a cache.

NOTE 2: The KMS retains the possibility to generate the secret key anew for data recovery purposes. This possibility remains until the related MPK/MSK pair is in the destructed state. Therefore destruction of protected data locally by the IBE client does not only depend on the destruction of the related secret key but also on the information security policy and backup policy that are relevant to the protected data. Only when the IBE client learns that the both the secret key and the master secret are in the destroyed state, can it locally delete the related protected data.

C.4 Decommission phase

C.4.1 Introduction

In this phase the KMS is preparing to completely stop operations or to migrate to a new MSK/MPK pair, this involves the deactivation of the currently active MPK/MSK pair.

NOTE: Migration to a new MSK/MPK can include a migration to a new IBE algorithm. While not detailed further, such migration implies that the IBE clients support the new IBE algorithm, either through pre-provisioning or via an update mechanism.

It is recommended that the MSK/MPK pair is refreshed periodically.

C.4.2 MPK/MSK deactivation

In this state the KMS will stop generating new secret keys except for data recovery purposes. IBE clients are to be informed that:

- they cannot use the deactivated MPK to encrypt (protect) messages under any public identity, old or new;
- they can process messages encrypted using the MPK before the deactivation went into effect; and
- they cannot request new secret keys for the deactivated MPK/MSK pair.

Deactivation of the currently active MPK/MSK pair requires that the KMS has brought a new key pair into the operational phase. The deactivation can be preceded by a grace period allowing IBE clients to synchronize to the new pair. The grace period allows IBE clients to re-encrypt data under a new secret key.

NOTE: The suspended state is not defined for the MPK/MSK pair since suspension requires the KMS and the IBE client to operate under a new MPK/MSK pair and migrate all public identity in use to new secret keys. Hence, a roll-back to a previous active state for the MPK/MSK makes little sense.

C.4.3 MSK compromise

If it is discovered that the MSK has been compromised, then the entirety of the cryptosystem is to be considered insecure since an unauthorised third-party can generate any secret key of their liking. The KMS and IBE clients are to transition to a new MPK/MSK pair immediately, following the announcement MPK/MSK pair is in the compromised state by the KMS. This requires a method whose security does not depend on that of an MSK in the compromised state.

C.5 Destruction phase

C.5.1 MSK/MPK destruction

Once an MPK/MSK pair has transitioned to the deactivated state, and it is no longer required to be able to recover protected data, the KMS can initiate the destruction of the MSK and instruct IBE clients to destroy locally stored secret keys.

Annex D: Cryptographic schemes for IBC

D.1 Introduction and overview

The present annex contains high-level descriptions of several proposals for IBC schemes. The contents of this annex are illustrative and are not intended to be an exhaustive list of all possible cryptographic approaches. As IBC becomes more widely used and embedded in products and services, new schemes will be added. As the efficiency and understanding of IBC improves, some existing schemes may be deprecated. In particular, the threat of a cryptographically relevant quantum computer will have a significant impact on many existing IBC schemes.

All IBC schemes involve the following pair of core algorithms:

- **Setup.** The KMS generates a master secret key MSK and master public key MPK for the scheme.
- **Extract.** The KMS derives the user secret key $\mathrm{SK}_{\mathrm{ID}}$ from the user's public identifier ID and the master secret key MSK.

IBE schemes consist of the four algorithms Setup, Extract, Encrypt and Decrypt where:

- **Encrypt.** The sender encrypts a fixed-length message M using the recipient's public identifier ID and the master public key MPK to produce the ciphertext C.
- **Decrypt.** The recipient decrypts the ciphertext C using their user secret key $\mathrm{SK}_{\mathrm{ID}}$ to recover the message M.

Identity-based key encapsulation mechanisms consist of the four algorithms Setup, Extract, Encapsulate and Decapsulate where:

- **Encapsulate.** The sender generates and encrypts a session key K using the recipient's public identifier ID and the master public key MPK to give the ciphertext C.
- **Decapsulate.** The recipient decrypts the ciphertext C using their user secret key $\mathrm{SK}_{\mathrm{ID}}$ to recover the session key K.

IBS schemes consist of the four algorithms Setup, Extract, Sign and Verify where:

- **Sign.** The sender signs a variable-length message M using their user secret key $\mathrm{SK}_{\mathrm{ID}}$ to give the signature S.
- **Verify.** The recipient checks the validity of the signature S for the message M using the sender's public identifier ID and the master public key MPK.

Mathematically, most IBC schemes are based on pairings on elliptic curves. These are functions which map pairs of points on an elliptic curve, or on related elliptic curves, to elements in a finite field. The security of pairing-based schemes depends on the difficulty of the discrete logarithm problem in both the elliptic curve and finite field, and the difficulty of inverting the pairing when one of the elliptic curve points is known.

Cryptographically relevant quantum computers will be able to break pairing-based schemes using Shor's algorithm. Quantum-safe IBC schemes use different constructions; for example, lattice-based cryptography. The security of lattice-based schemes depends on the difficulty of finding a lattice vector that is sufficiently short or is sufficiently close to a target vector. These problems are believed to be hard to solve using either classical or quantum algorithms.

D.2 Notation

The Boneh-Franklin (clause D.3.1), Sakai-Kasahara (clause D.3.2) and SM9 (clause D.3.3) encryption schemes are pairing-based. Similarly, the BLMQ (clause D.4.1) and SM9 (clause D.4.2) signature schemes are also pairing-based. However, ECCSI (clause D.4.3) does not use pairings.

A pairing is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ where:

- $\mathbb{G}_1$ is a cyclic group of prime order p, written additively, with fixed generator $G_1 \in \mathbb{G}_1$;
- $\mathbb{G}_2$ is a cyclic group of prime order p, written additively, with fixed generator $G_2 \in \mathbb{G}_2$; and
- $\mathbb{G}_T$ is a cyclic group of prime order p, written multiplicatively.

Pairings also satisfy the following three conditions:

- **Bilinearity.** For all $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$,

$$e(aP, bQ) = e(aP, Q)^b = e(P, bQ)^a = e(P, Q)^{ab}.$$

- **Non-degeneracy**. There exist $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$ where $e(P, Q) \in \mathbb{G}_T$ is not the identity.
- **Computability.** The value $e(P, Q)$ can be efficiently computed for all $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$.

NOTE: Non-degeneracy implies that $e(G_1, G_2)$ will be a generator for $\mathbb{G}_T$.

There are four types of pairings on elliptic curves which are categorized depending on the relationship between the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. Some types of pairing are more efficient to compute than others. Some security proofs for IBC schemes require the pairing to be of a certain type. However, the specific choice of elliptic curve and pairing is out of scope of the present document. The high-level descriptions of the schemes in clauses D.3 and D.4 are given for abstract pairings.

D.3 Identity-Based Encryption (IBE) schemes

D.3.1 Boneh-Franklin

D.3.1.1 Introduction

Boneh and Franklin's IBE scheme was first proposed in [i.1] and is included in IETF RFC 5091 [i.31], IEEE 1363.3 [i.12] and ISO/IEC 18033-5 [i.23]. The key encapsulation variant was proposed by Libert and Quisquater in [i.13].

The notation of clause D.1 is used throughout the present annex: $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of order p generated by G_1 and G_2, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing. Identifiers are represented as bit strings of arbitrary length, and messages as bit strings of fixed length ℓ.

The Boneh-Franklin scheme requires four cryptographic hash functions:

- $H_1 : \{0,1\}^* \to \mathbb{G}_2$ to hash the identifier;
- $H_2 : \{0,1\}^\ell \times \{0,1\}^\ell \to \mathbb{Z}_p$ to derive an ephemeral private value;
- $H_3 : \{0,1\}^\ell \to \{0,1\}^\ell$ to mask the message; and
- $H_4 : \mathbb{G}_T \to \{0,1\}^\ell$ to mask or derive the session key.

D.3.1.2 Setup

Input: None

Output: Master secret key MSK $\in \mathbb{Z}_p^*$
Master public key MPK $\in \mathbb{G}_1$

1) Choose a random $s \in \mathbb{Z}_p^*$.
2) Set MSK $\coloneqq s \in \mathbb{Z}_p^*$.

3) Compute $\mathrm{MPK} \coloneqq sG_1 \in \mathbb{G}_1$.

D.3.1.3 Extract

Input: Master secret key $\mathrm{MSK} \in \mathbb{Z}_p^*$
Public identifier $\mathrm{ID} \in \{0,1\}^*$

Output: User secret key $\mathrm{SK}_{\mathrm{ID}} \in \mathbb{G}_2$

1) Parse $\mathrm{MSK} \coloneqq s \in \mathbb{Z}_p^*$.

2) Compute $Q \coloneqq H_1(\mathrm{ID}) \in \mathbb{G}_2$.

3) Compute $\mathrm{SK}_{\mathrm{ID}} \coloneqq sQ \in \mathbb{G}_2$.

A user can confirm that their user secret key is valid by checking that $e(G_1, \mathrm{SK}_{\mathrm{ID}}) = e(\mathrm{MPK}, Q)$, where $Q \coloneqq H_1(\mathrm{ID})$, since:

$$e(G_1, \mathrm{SK}_{\mathrm{ID}}) = e(G_1, sQ) = e(G_1, Q)^s = e(sG_1, Q) = e(\mathrm{MPK}, Q).$$

D.3.1.4 Encrypt

Input: Message $M \in \{0,1\}^\ell$
Public identifier $\mathrm{ID} \in \{0,1\}^*$
Master public key $\mathrm{MPK} \in \mathbb{G}_1$

Output: Ciphertext $C \in \mathbb{G}_1 \times \{0,1\}^\ell \times \{0,1\}^\ell$

1) Choose a random $a \in \{0,1\}^\ell$.

2) Compute $b \coloneqq H_2(a, M) \in \mathbb{Z}_p$.

3) Compute $P \coloneqq bG_1 \in \mathbb{G}_1$.

4) Compute $Q \coloneqq H_1(\mathrm{ID}) \in \mathbb{G}_2$.

5) Compute $R \coloneqq e(\mathrm{MPK}, Q) \in \mathbb{G}_T$.

6) Compute $v \coloneqq M \oplus H_3(a) \in \{0,1\}^\ell$.

7) Compute $w \coloneqq a \oplus H_4(R^b) \in \{0,1\}^\ell$.

8) Set $C \coloneqq (P, v, w) \in \mathbb{G}_1 \times \{0,1\}^\ell \times \{0,1\}^\ell$.

D.3.1.5 Decrypt

Input: User private key $\mathrm{SK}_{\mathrm{ID}} \in \mathbb{G}_2$
Ciphertext $C \in \mathbb{G}_1 \times \{0,1\}^\ell \times \{0,1\}^\ell$

Output: Message $M \in \{0,1\}^\ell$ or decryption failure.

1) Parse $C \coloneqq (P, v, w) \in \mathbb{G}_1 \times \{0,1\}^\ell \times \{0,1\}^\ell$.

2) Compute $R' \coloneqq e(P, \mathrm{SK}_{\mathrm{ID}}) \in \mathbb{G}_T$.

3) Compute $a \coloneqq w \oplus H_4(R') \in \{0,1\}^\ell$.

4) Compute $M \coloneqq v \oplus H_3(a) \in \{0,1\}^\ell$.

5) Compute $b \coloneqq H_2(a, M) \in \mathbb{Z}_p$.

6) If $P = bG_1$, return M.

7) Otherwise, return a decryption failure.

Decryption succeeds when $P = bG_1$ because:

$$R' = e(P, \mathrm{SK_{ID}}) = e(bG_1, sQ) = e(G_1, Q)^{sb} = e(sG_1, Q)^b = e(\mathrm{MPK}, Q)^b = R^b.$$

D.3.1.6 Encapsulate

Input: Public identifier $\mathrm{ID} \in \{0,1\}^*$
Master public key $\mathrm{MPK} \in \mathbb{G}_1$

Output: Session key $K \in \{0,1\}^\ell$
Ciphertext $C \in \mathbb{G}_1$

1) Choose a random $b \in \mathbb{Z}_p$.
2) Compute $P := bG_1 \in \mathbb{G}_1$.
3) Compute $Q \coloneqq H_1(\mathrm{ID}) \in \mathbb{G}_2$.
4) Compute $R \coloneqq e(\mathrm{MPK}, Q) \in \mathbb{G}_\mathrm{T}$.
5) Compute $K = H_4(R^b) \in \{0,1\}^\ell$.
6) Set $C \coloneqq P \in \mathbb{G}_1$.

D.3.1.7 Decapsulate

Input: User secret key $\mathrm{SK_{ID}} \in \mathbb{G}_2$
Ciphertext $C \in \mathbb{G}_1$

Output: Session key $K \in \{0,1\}^\ell$

1) Parse $C \coloneqq P \in \mathbb{G}_1$.
2) Compute $R' \coloneqq e(P, \mathrm{SK_{ID}}) \in \mathbb{G}_\mathrm{T}$.
3) Compute $K \coloneqq H_4(R') \in \{0,1\}^\ell$.

Decapsulation succeeds for the same reason that decryption succeeds in clause D.3.2.5.

D.3.2 Sakai-Kasahara

D.3.2.1 Introduction

The Sakai-Kasahara IBE scheme was proposed in [i.24] and is described in IETF RFC 6508 [i.19]. The key encapsulation variant was proposed by Barbosa et al [i.25] and is included in both IEEE 1363.3 [i.12] and ISO/IEC 18033-5 [i.23].

The notation of clause D.1 is used throughout the present annex: $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of order p generated by G_1 and G_2, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing. Identifiers are represented as bit strings of arbitrary length, and messages as bit strings of fixed length ℓ.

The Sakai-Kasahara scheme requires the following five cryptographic hash functions:

- $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ to hash the identifier;
- $H_2 : \{0,1\}^\ell \times \{0,1\}^\ell \to \mathbb{Z}_p$ to derive an ephemeral private value for encryption;
- $H_2' : \{0,1\}^\ell \to \mathbb{Z}_p$ to derive an ephemeral private value for encapsulation;
- $H_3 : \{0,1\}^\ell \to \{0,1\}^\ell$ to mask the message; and
- $H_4 : \mathbb{G}_T \to \{0,1\}^\ell$ to mask or derive the session key.

NOTE: Unlike Boneh-Franklin (see clause D.3.1.1), the Sakai-Kasahara scheme does not require a hash into $\mathbb{G}_2$. This gives implementers some additional freedom in choosing curves.

D.3.2.2 Setup

Input: None

Output: Master secret key MSK $\in \mathbb{Z}_p^*$
Master public key MPK $\in \mathbb{G}_1$

1) Choose a random $s \in \mathbb{Z}_p^*$.

2) Set MSK $\coloneqq s \in \mathbb{Z}_p^*$.

3) Compute MPK $\coloneqq sG_1 \in \mathbb{G}_1$.

D.3.2.3 Extract

Input: Master secret key MSK $\in \mathbb{Z}_p^*$
Public identifier ID $\in \{0, 1\}^*$

Output: User secret key $\mathrm{SK}_{\mathrm{ID}} \in \mathbb{G}_2$

1) Parse MSK $\coloneqq s \in \mathbb{Z}_p^*$.

2) Compute $z \coloneqq H_1(\mathrm{ID}) \in \mathbb{Z}_p$.

3) Compute $\mathrm{SK}_{\mathrm{ID}} \coloneqq (s + z)^{-1} G_2 \in \mathbb{G}_2$.

NOTE: If $s + \mathrm{z} = 0$ in step 3, extraction fails and $H_1(\mathrm{ID})$ reveals the master secret key. However, this will happen with negligible probability.

A user can confirm that their user secret key is valid by checking that $e(\mathrm{MPK} + zG_1, \mathrm{SK}_{\mathrm{ID}}) = e(G_1, G_2)$, where $z \coloneqq H_1(\mathrm{ID})$, since:

$$e(\mathrm{MPK} + zG_1, \mathrm{SK}_{\mathrm{ID}}) = e((s + z)G_1, (s + z)^{-1} G_2) = e(G_1, G_2).$$

D.3.2.4 Encrypt

Input: Message $M \in \{0, 1\}^\ell$
Public identifier ID $\in \{0, 1\}^*$
Master public key MPK $\in \mathbb{G}_1$

Output: Ciphertext $C \in \mathbb{G}_1 \times \{0, 1\}^\ell \times \{0, 1\}^\ell$

1) Choose a random $a \in \{0, 1\}^\ell$.

2) Compute $b \coloneqq H_2(a, M) \in \mathbb{Z}_p$.

3) Compute $z \coloneqq H_1(\mathrm{ID}) \in \mathbb{Z}_p$.

4) Compute $P \coloneqq b(\mathrm{MPK} + zG_1) \in \mathbb{G}_1$.

5) Compute $R \coloneqq e(G_1, G_2) \in \mathbb{G}_\mathrm{T}$.

6) Compute $v \coloneqq M \oplus H_3(a) \in \{0, 1\}^\ell$.

7) Compute $w \coloneqq a \oplus H_4(R^b) \in \{0, 1\}^\ell$.

8) Set $C \coloneqq (P, v, w) \in \mathbb{G}_1 \times \{0, 1\}^\ell \times \{0, 1\}^\ell$.

NOTE: The value $R \coloneqq e(G_1, G_2) \in \mathbb{G}_\mathrm{T}$ in step 5 can be pre-computed and considered part of the public system parameters.

D.3.2.5 Decrypt

Input: User private key $\mathrm{SK}_{\mathrm{ID}} \in \mathbb{G}_2$
Ciphertext $C \in \mathbb{G}_1 \times \{0,1\}^\ell \times \{0,1\}^\ell$
Public identifier $\mathrm{ID} \in \{0,1\}^*$
Master public key $\mathrm{MPK} \in \mathbb{G}_1$

Output: Message $M \in \{0,1\}^\ell$ or decryption failure.

1) Parse $C \coloneqq (P, v, w) \in \mathbb{G}_1 \times \{0,1\}^\ell \times \{0,1\}^\ell$.
2) Compute $R' \coloneqq e(P, \mathrm{SK}_{\mathrm{ID}}) \in \mathbb{G}_T$.
3) Compute $a \coloneqq w \oplus H_4(R') \in \{0,1\}^\ell$.
4) Compute $M \coloneqq v \oplus H_3(a) \in \{0,1\}^\ell$.
5) Compute $b \coloneqq H_2(a, M) \in \mathbb{Z}_p$.
6) Compute $z \coloneqq H_1(\mathrm{ID}) \in \mathbb{Z}_p$.
7) If $P = b(\mathrm{MPK} + zG_1)$, return M.
8) Otherwise, return a decryption failure.

Decryption succeeds when $P = b(\mathrm{MPK} + zG_1)$ because:

$$R' = e(P, \mathrm{SK}_{\mathrm{ID}}) = e(b(\mathrm{MPK} + zG_1), (s+z)^{-1}G_2) = e(b(s+z)G_1, (s+z)^{-1}G_2) = e(G_1, G_2)^b = R^b.$$

D.3.2.6 Encapsulate

Input: Public identifier $\mathrm{ID} \in \{0,1\}^*$
Master public key $\mathrm{MPK} \in \mathbb{G}_1$

Output: Session key $K \in \{0,1\}^\ell$
Ciphertext $C \in \mathbb{G}_1 \times \{0,1\}^\ell$

1) Choose a random $a \in \mathbb{Z}_p$.
2) Compute $b \coloneqq H_2'(a) \in \mathbb{Z}_p$.
3) Compute $z \coloneqq H_1(\mathrm{ID}) \in \mathbb{Z}_p$.
4) Compute $P \coloneqq b(\mathrm{MPK} + zG_1) \in \mathbb{G}_1$.
5) Compute $R \coloneqq e(G_1, G_2) \in \mathbb{G}_\mathrm{T}$.
6) Compute $K \coloneqq H_3(a) \in \{0,1\}^\ell$.
7) Compute $w \coloneqq a \oplus H_4(R^b) \in \{0,1\}^\ell$.
8) Set $C \coloneqq (P, w) \in \mathbb{G}_1 \times \{0,1\}^\ell$.

NOTE: The value $R \coloneqq e(G_1, G_2) \in \mathbb{G}_\mathrm{T}$ in step 5 can be pre-computed and considered part of the public system parameters.

D.3.2.7 Decapsulate

Input: User secret key $\mathrm{SK}_{\mathrm{ID}} \in \mathbb{G}_2$
Ciphertext $C \in \mathbb{G}_1 \times \{0,1\}^\ell$
Public identifier $\mathrm{ID} \in \{0,1\}^*$
Master public key $\mathrm{MPK} \in \mathbb{G}_1$

Output: Session key K or decapsulation failure

1) Parse $C := (P, w) \in \mathbb{G}_1 \times \{0,1\}^\ell$.

2) Compute $R' := e(P, \mathrm{SK}_{\mathrm{ID}}) \in \mathbb{G}_\mathrm{T}$.

3) Compute $a := w \oplus H_4(R') \in \{0,1\}^\ell$.

4) Compute $K := H_3(a) \in \{0,1\}^\ell$.

5) Compute $b := H_2'(a) \in \mathbb{Z}_p$.

6) Compute $z := H_1(\mathrm{ID}) \in \mathbb{Z}_p$.

7) If $P = b(\mathrm{MPK} + zG_1)$, return K.

8) Otherwise, return a decapsulation failure.

Decapsulation succeeds for the same reason that decryption succeeds in clause D.3.3.5.

D.3.3 SM9

D.3.3.1 Introduction

SM9 is a Chinese national cryptography standard for Identity-based Cryptography issued by the Chinese State Cryptographic Authority. Translations of the algorithm specifications are given in [i.26] and [i.27].

The key encapsulation version of SM9 is included in ISO/IEC 18033-5:2015/Amd.1:2021 [i.23].

The notation of clause D.1 is used throughout the present annex: $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of order p generated by G_1 and G_2, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing. Identifiers are represented as bit strings of arbitrary length, and messages as bit strings of fixed length ℓ.

The SM9 encryption and encapsulation schemes require the following cryptographic hash functions:

- $H_1 : \{0,1\}^* \times \{0,1\}^8 \to \mathbb{Z}_p$ to hash the identifier;
- $H_3 : \mathbb{G}_1 \times \mathbb{G}_\mathrm{T} \times \{0,1\}^* \to \{0,1\}^\ell$ to mask the message;
- $H_4 : \mathbb{G}_1 \times \mathbb{G}_\mathrm{T} \times \{0,1\}^* \to \{0,1\}^\ell$ to derive the integrity protection key or session key; and
- $H_5 : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^{\ell'}$ to provide integrity protection.

NOTE 1: The present clause follows the description of SM9 in [i.26] and [i.27] and includes an additional 8-bit public value as an input to H_1 when hashing the identifier. This is a fixed value in [i.28].

NOTE 2: The present clause only describes SM9 in stream cipher mode. It can also be used in block cipher mode where the message or session key is encrypted using a block cipher.

D.3.3.2 Setup

Input: None

Output: Master secret key $\mathrm{MSK} \in \mathbb{Z}_p^*$
Master public key $\mathrm{MPK} \in \mathbb{G}_1$
Public value salt $\in \{0,1\}^8$

1) Choose a random $s \in \mathbb{Z}_p^*$.

2) Set $\mathrm{MSK} := s \in \mathbb{Z}_p^*$.

3) Compute $\mathrm{MPK} := sG_1 \in \mathbb{G}_1$.

4) Choose a random salt $\in \{0,1\}^8$.

D.3.3.3 Extract

Input: Master secret key MSK $\in \mathbb{Z}_p^*$
Public identifier ID $\in \{0,1\}^*$
Public value salt $\in \{0,1\}^8$

Output: User secret key $\mathrm{SK_{ID}} \in \mathbb{G}_2$

1) Parse MSK $:= s \in \mathbb{Z}_p^*$.

2) Compute $z := H_1(\mathrm{ID}, \mathrm{salt}) \in \mathbb{Z}_p$.

3) Compute $\mathrm{SK_{ID}} := s(s+z)^{-1}G_2 \in \mathbb{G}_2$

NOTE: If $s + z = 0$ in step 3, extraction fails and $H_1(\mathrm{ID}, \mathrm{salt})$ reveals the master secret key. In this case, generate a new master key pair and reissue user secret keys for all users.

A user can confirm that their user private key is valid by checking that $e(\mathrm{MPK} + zG_1, \mathrm{SK_{ID}}) = e(\mathrm{MPK}, G_2)$, where $z := H_1(\mathrm{ID}, \mathrm{salt})$, since:

$$e(\mathrm{MPK} + zG_1, \mathrm{SK_{ID}}) = e\big((s+z)G_1, s(s+z)^{-1}G_2\big) = e(G_1, G_2)^s = e(sG_1, G_2) = e(\mathrm{MPK}, G_2).$$

D.3.3.4 Encrypt

Input: Message $M \in \{0,1\}^\ell$
Public identifier ID $\in \{0,1\}^*$
Master public key MPK $\in \mathbb{G}_1$
Public salt value salt $\in \{0,1\}^8$

Output: Ciphertext $C \in \mathbb{G}_1 \times \{0,1\}^\ell \times \{0,1\}^{\ell'}$

1) Choose a random $b \in \mathbb{Z}_p$.

2) Compute $z := H_1(\mathrm{ID}, \mathrm{salt}) \in \mathbb{Z}_p$.

3) Compute $P := b(\mathrm{MPK} + zG_1) \in \mathbb{G}_1$.

4) Compute $R := e(\mathrm{MPK}, G_2) \in \mathbb{G}_T$.

5) Compute $v := M \oplus H_3(P, R^b, \mathrm{ID}) \in \{0,1\}^\ell$.

6) Compute $k := H_4(P, R^b, \mathrm{ID}) \in \{0,1\}^\ell$.

7) Compute $w := H_5(k, v) \in \{0,1\}^{\ell'}$.

8) Set $C := (P, v, w)$

NOTE: The value $R := e(\mathrm{MPK}, G_2) \in \mathbb{G}_\mathrm{T}$ in step 5 can be pre-computed and considered part of the master public key.

D.3.3.5 Decrypt

Input: User private key $\mathrm{SK_{ID}} \in \mathbb{G}_2$
Public identifier ID $\in \{0,1\}^*$
Ciphertext $C \in \mathbb{G}_1 \times \{0,1\}^\ell \times \{0,1\}^{\ell'}$

Output: Message $M \in \{0,1\}^l$ or decryption failure.

1) Parse $C := (P, v, w)$

2) Compute $R' := e(P, \mathrm{SK_{ID}}) \in \mathbb{G}_T$.

3) Compute $M \coloneqq w \oplus H_3(P, R', \mathrm{ID}) \in \{0,1\}^{\ell}$.

4) Compute $k' \coloneqq H_4(P, R', \mathrm{ID}) \in \{0,1\}^{\ell}$.

5) Compute $w' \coloneqq H_5(k', v) \in \{0,1\}^{\ell'}$.

6) If $w = w'$, return M.

7) Otherwise, return a decryption failure.

Decryption succeeds if $P = b(\mathrm{MPK} + zG_1)$ because:

$$\begin{aligned}
R' &= e(P, \mathrm{SK}_{\mathrm{ID}}) \\
&= e(b(\mathrm{MPK} + zG_1), s(s+z)^{-1}G_2) \\
&= e(b(s+z)G_1, s(s+z)^{-1}G_2) \\
&= e(G_1, G_2)^{bs} \\
&= e(sG_1, G_2)^{b} \\
&= e(\mathrm{MPK}, G_2)^{b} \\
&= R^{b}.
\end{aligned}$$

D.3.3.6 Encapsulate

Input: Public identifier $\mathrm{ID} \in \{0,1\}^*$
Master public key $\mathrm{MPK} \in \mathbb{G}_1$
Public value $\mathrm{salt} \in \{0,1\}^8$

Output: Session key $K \in \{0,1\}^{\ell}$
Ciphertext $C \in \mathbb{G}_1$

1) Choose a random $b \in \mathbb{Z}_p$.

2) Compute $z \coloneqq H_1(\mathrm{ID}, \mathrm{salt}) \in \mathbb{Z}_p$.

3) Compute $P \coloneqq b(\mathrm{MPK} + zG_1) \in \mathbb{G}_1$.

4) Compute $R \coloneqq e(\mathrm{MPK}, G_2) \in \mathbb{G}_T$.

5) Compute $K \coloneqq H_4(P, R^b, \mathrm{ID}) \in \{0,1\}^{\ell}$.

6) Set $C \coloneqq P$.

NOTE: The value $R \coloneqq e(\mathrm{MPK}, G_2) \in \mathbb{G}_\mathrm{T}$ in step 5 can be pre-computed and considered part of the master public key.

D.3.3.7 Decapsulate

Input: User secret key $\mathrm{SK}_{\mathrm{ID}} \in \mathbb{G}_2$
Public identifier $\mathrm{ID} \in \{0,1\}^*$
Ciphertext $C \in \mathbb{G}_1$

Output: Session key $K \in \{0,1\}^{\ell}$

1) Parse $C \coloneqq P \in \mathbb{G}_1$

2) Compute $R' \coloneqq e(P, \mathrm{SK}_{\mathrm{ID}}) \in \mathbb{G}_T$.

3) Compute $K \coloneqq H_4(P, R', \mathrm{ID}) \in \{0,1\}^{\ell}$.

Decapsulation succeeds for the same reason that decryption succeeds in clause D.3.3.5.

D.4 Identity-Based Signature (IBS) Schemes

D.4.1 BLMQ

D.4.1.1 Introduction

Barreto et al. first presented the BLMQ algorithm in [i.15] as a signcryption algorithm. The present annex describes the signature part of the algorithm. The BLMQ signature scheme is included in IEEE 1363.3 [i.12].

The notation of clause D.1 is used throughout the present annex: $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of order p generated by G_1 and G_2, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing. Identifiers and messages are represented as bit strings of arbitrary length.

The BLMQ algorithm uses the following pair of cryptographic hash functions:

- $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ to hash the identifier; and
- $H_2 : \{0,1\}^* \times \mathbb{G}_T \to \mathbb{Z}_p$ to hash the message.

D.4.1.2 Setup

Input: None

Output: Master secret key $\mathrm{MSK} \in \mathbb{Z}_p^*$
Master public key $\mathrm{MPK} \in \mathbb{G}_2$

1) Choose a random $s \in \mathbb{Z}_p^*$.
2) Set $\mathrm{MSK} \coloneqq s \in \mathbb{Z}_p^*$.
3) Compute $\mathrm{MPK} \coloneqq sG_2 \in \mathbb{G}_2$.

D.4.1.3 Extract

Input: Master secret key $\mathrm{MSK} \in \mathbb{Z}_p^*$
Public identifier $\mathrm{ID} \in \{0,1\}^*$

Output: User secret key $\mathrm{SK}_{\mathrm{ID}} \in \mathbb{G}_1$

1) Parse $\mathrm{MSK} \coloneqq s \in \mathbb{Z}_p^*$.
2) Set $z \coloneqq H_1(\mathrm{ID}) \in \mathbb{Z}_p$.
3) Compute $\mathrm{SK}_{\mathrm{ID}} \coloneqq (s+z)^{-1}G_1 \in \mathbb{G}_1$.

NOTE: If $s + z = 0$ in step 3, extraction fails and $H_1(\mathrm{ID})$ reveals the master secret key. However, this will happen with negligible probability.

A user can confirm that their user secret key is valid by checking that $e(\mathrm{SK}_{\mathrm{ID}}, \mathrm{MPK} + zG_2) = e(G_1, G_2)$, where $z \coloneqq H_1(\mathrm{ID})$, since:

$$e(\mathrm{SK}_{\mathrm{ID}}, \mathrm{MPK} + zG_2) = e((s+z)^{-1}G_1, (s+z)G_2) = e(G_1, G_2).$$

D.4.1.4 Sign

Input: User secret key $\mathrm{SK}_{\mathrm{ID}} \in \mathbb{G}_1$
Message $M \in \{0,1\}^*$

Output: Signature $S \in \mathbb{G}_1 \times \mathbb{Z}_p$

1) Choose a random $b \in \mathbb{Z}_p$.

2) Compute $R \coloneqq e(G_1, G_2) \in \mathbb{G}_T$.

3) Compute $v \coloneqq H_2(M, R^b) \in \mathbb{Z}_p$.

4) Compute $P \coloneqq (b + v)\mathrm{SK}_{\mathrm{ID}} \in \mathbb{G}_1$.

5) Set $S \coloneqq (P, v) \in \mathbb{G}_1 \times \mathbb{Z}_p$.

NOTE: The value $R \coloneqq e(G_1, G_2) \in \mathbb{G}_\mathrm{T}$ in step 2 can be pre-computed and considered part of the public system parameters.

D.4.1.5 Verify

Input: Message $M \in \{0,1\}^*$
Signature $S \in \mathbb{G}_1 \times \mathbb{Z}_p$
Public identifier $\mathrm{ID} \in \{0,1\}^*$
Master public key $\mathrm{MPK} \in \mathbb{G}_2$

Output: Accept or reject

1) Parse $S \coloneqq (P, v) \in \mathbb{G}_1 \times \mathbb{Z}_p$.

2) Compute $z \coloneqq H_1(\mathrm{ID}) \in \mathbb{Z}_p$.

3) Compute $R' \coloneqq e(P, \mathrm{MPK} + zG_2) \in \mathbb{G}_\mathrm{T}$.

4) Compute $R'' \coloneqq e(G_1, G_2)^{-v} \in \mathbb{G}_\mathrm{T}$.

5) Compute $v' \coloneqq H_2(M, R'R'') \in \mathbb{Z}_p$.

6) If $v = v'$, accept.

7) Otherwise, reject.

NOTE: The value $R \coloneqq e(G_1, G_2) \in \mathbb{G}_\mathrm{T}$ in step 5 can be pre-computed and considered part of the public system parameters.

Verification succeeds when $P = (b + v)\mathrm{SK}_{\mathrm{ID}}$ because:

$$\begin{aligned} R' &= e(P, \mathrm{MPK} + zG_2) \\ &= e\big((b + v)\mathrm{SK}_{\mathrm{ID}}, sG_2 + zG_2\big) \\ &= e((b + v)(s + z)^{-1}G_1, (s + z)G_2) \\ &= e(G_1, G_2)^{b+v} \end{aligned}$$

so:

$$R'R'' = e(G_1, G_2)^{b+v} e(G_1, G_2)^{-v} = e(G_1, G_2)^b = R^b.$$

D.4.2 SM9

D.4.2.1 Introduction

SM9 is a Chinese national cryptography standard for Identity-based Cryptography issued by the Chinese State Cryptographic Authority. Translations of the algorithm specifications are given in [i.26] and [i.27].

The SM9 signature scheme is included in ISO/IEC 14888-3:2018 [i.13].

The notation of clause D.1 is used throughout the present annex: $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of order p generated by G_1 and G_2, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing. Identifiers and messages are represented as bit strings of arbitrary length.

The SM9 signature scheme uses the following cryptographic functions:

- H_1: $\{0,1\}^* \times \{0,1\}^8 \to \mathbb{Z}_p^*$ to hash the identifier; and
- $H_2 : \{0,1\}^* \times \mathbb{G}_T \to \mathbb{Z}_p^*$ to hash the message.

D.4.2.2 Setup

Input: None

Output: Master secret key MSK $\in \mathbb{Z}_p^*$
Master public key MPK $\in \mathbb{G}_2$
Public value salt $\in \{0,1\}^8$

1) Choose a random $s \in \mathbb{Z}_p^*$.
2) Set MSK $:= s \in \mathbb{Z}_p^*$.
3) Compute MPK $:= sG_2 \in \mathbb{G}_2$.
4) Choose a random salt $\in \{0,1\}^8$.

D.4.2.3 Extract

Input: Master secret key MSK $\in \mathbb{Z}_p^*$
Public value salt $\in \{0,1\}^8$
Public identifier ID $\in \{0,1\}^*$

Output: User secret key $\mathrm{SK}_{\mathrm{ID}} \in \mathbb{G}_1$

1) Parse MSK $:= s \in \mathbb{Z}_p^*$.
2) Compute $z := H_1(\mathrm{ID}, \mathrm{salt}) \in \mathbb{Z}_p$.
3) Compute $\mathrm{SK}_{\mathrm{ID}} := s(s+z)^{-1}G_1 \in \mathbb{G}_1$

NOTE: If $s + z = 0$ in step 3, extraction fails and $H_1(\mathrm{ID}, \mathrm{salt})$ reveals the master secret key. In this case, generate a new master key pair and reissue user secret keys for all users.

A user can confirm that their user private key is valid by checking that $e(\mathrm{SK}_{\mathrm{ID}}, \mathrm{MPK} + zG_2) = e(G_1, \mathrm{MPK})$, where $z := H_1(\mathrm{ID}, \mathrm{salt})$, since:

$$e(\mathrm{SK}_{\mathrm{ID}}, \mathrm{MPK} + zG_2) = e(s(s+z)^{-1}G_1, (s+z)G_2) = e(G_1, G_2)^s = e(G_1, sG_2) = e(G_1, \mathrm{MPK}).$$

D.4.2.4 Sign

Input: User secret key $\mathrm{SK}_{\mathrm{ID}} \in \mathbb{G}_1$
Public value salt $\in \{0,1\}^8$
Message $M \in \{0,1\}^*$

Output: Signature $S \in \mathbb{G}_1 \times \mathbb{Z}_p$

1) Choose a random $b \in \mathbb{Z}_p^*$.
2) Compute $R := e(G_1, \mathrm{MPK}) \in \mathbb{G}_T$.
3) Compute $v := H_2(M, R^b) \in \mathbb{Z}_p$.
4) Compute $P := (b - v)\mathrm{SK}_{\mathrm{ID}} \in \mathbb{G}_1$.

5) Set $S \coloneqq (P, v) \in \mathbb{G}_1 \times \mathbb{Z}_p$.

NOTE: The value $R \coloneqq e(G_1, \mathrm{MPK}) \in \mathbb{G}_\mathrm{T}$ in step 2 can be pre-computed and considered part of the master public key.

D.4.2.5 Verify

Input: Message $M \in \{0,1\}^*$
Signature $S \in \mathbb{G}_1 \times \mathbb{Z}_p$
Public identifier $\mathrm{ID} \in \{0,1\}^*$
Master public key $\mathrm{MPK} \in \mathbb{G}_2$
Public value $\mathrm{salt} \in \{0,1\}^8$

Output: Accept or reject

1) Parse $S \coloneqq (P, v) \in \mathbb{G}_1 \times \mathbb{Z}_p$.

2) Compute $z \coloneqq H_1(\mathrm{ID}, \mathrm{salt}) \in \mathbb{Z}_p$.

3) Compute $R' \coloneqq e(P, \mathrm{MPK} + zG_2) \in \mathbb{G}_T$.

4) Compute $R'' \coloneqq e(G_1, \mathrm{MPK})^v \in \mathbb{G}_T$.

5) Compute $v' \coloneqq H_2(M, R'R'') \in \mathbb{Z}_p$.

6) If $v = v'$, accept.

7) Otherwise, reject.

NOTE: The value $R \coloneqq e(G_1, \mathrm{MPK}) \in \mathbb{G}_\mathrm{T}$ in step 4 can be pre-computed and considered part of the master public key.

Verification succeeds when $P = (b - v)\ \mathrm{SK}_\mathrm{ID}$ since:

$$\begin{aligned} R' &= e(P, \mathrm{MPK} + zG_2) \\ &= e\big((b - v)\mathrm{SK}_\mathrm{ID}, sG_2 + zG_2\big) \\ &= e((b - v)s(s + z)^{-1}G_1, (s + z)G_2) \\ &= e(G_1, G_2)^{(b-v)s} \\ &= e(G_1, sG_2)^{b-v} \\ &= e(G_1, \mathrm{MPK})^{b-v} \end{aligned}$$

so:

$$R'R'' = e(G_1, \mathrm{MPK})^{b-v} e(G_1, \mathrm{MPK})^v = e(G_1, \mathrm{MPK})^b = R^b.$$

D.4.3 ECCSI

D.4.3.1 Introduction

The Elliptic Curve-Based Certificateless Signatures for Identity-Based Encryption (ECCSI) described in IETF RFC 6507 [i.20] is an identity-based signature scheme which removes the need for a pairing at the cost of sending a Public Validation Token PVT as part of the signature.

In the present clause, $\mathbb{G}$ is a cyclic group of order p, written additively, generated by G. Identifiers and messages are represented as bit strings of arbitrary length.

This ECCSI scheme uses the following pair of cryptographic hash function:

- H_1: $\{0,1\}^* \times \mathbb{G} \to \mathbb{Z}_p$ to hash the identifier; and

- $H_2 \colon \mathbb{Z}_p \times \mathbb{Z}_p \times \{0,1\}^* \to \mathbb{Z}_p$ to hash the message.

NOTE: ECCSI [i.20] includes the generator G and master public key MPK as inputs to H_1 when hashing the identifier. These have been omitted from the present clause to simplify the description.

D.4.3.2 Setup

Input: None

Output: Master secret key $\mathrm{MSK} \in \mathbb{Z}_p^*$
Master public key $\mathrm{MPK} \in \mathbb{G}$

1) Choose a random $s \in \mathbb{Z}_p^*$.

2) Set $\mathrm{MSK} \coloneqq s \in \mathbb{Z}_p^*$.

3) Compute $\mathrm{MPK} \coloneqq sG \in \mathbb{G}$.

D.4.3.3 Extract

Input: Public identifier $\mathrm{ID} \in \{0,1\}^*$
Master secret key $\mathrm{MSK} \in \mathbb{Z}_p^*$

Output: User secret key $\mathrm{SK}_{\mathrm{ID}} \in \mathbb{Z}_p$
Public validation token $\mathrm{PVT} \in \mathbb{G}$

1) Parse $\mathrm{MSK} \coloneqq s \in \mathbb{Z}_p^*$.

2) Choose a random $t \in \mathbb{Z}_p^*$.

3) Compute $\mathrm{PVT} \coloneqq tG \in \mathbb{G}$.

4) Compute $z \coloneqq H_1(\mathrm{ID}, \mathrm{PVT}) \in \mathbb{Z}_p$.

5) Compute $\mathrm{SK}_{\mathrm{ID}} \coloneqq s + tz \in \mathbb{Z}_p$.

A user can confirm that their user secret key $\mathrm{SK}_{\mathrm{ID}}$ and public validation token PVT are valid by checking that $\mathrm{SK}_{\mathrm{ID}}G - z\mathrm{PVT} = \mathrm{MPK}$, where $z \coloneqq H_1(\mathrm{ID}, \mathrm{PVT})$, since:

$$\mathrm{SK}_{\mathrm{ID}}G - z\mathrm{PVT} = (s + tz)G - tzG = sG = \mathrm{MPK}.$$

D.4.3.4 Sign

Input: Message $M \in \{0,1\}^*$
User secret key $\mathrm{SK}_{\mathrm{ID}} \in \mathbb{Z}_p$
Public identifier $\mathrm{ID} \in \{0,1\}^*$
Public validation token $\mathrm{PVT} \in \mathbb{G}$

Output: Signature $S \in \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{G}$

1) Choose a random $a \in \mathbb{Z}_p^*$.

2) Compute $P \coloneqq aG \in \mathbb{G}$.

3) Set $v \coloneqq P_x \in \mathbb{Z}_p$, the x-coordinate of the point P.

4) Compute $z \coloneqq H_1(\mathrm{ID}, \mathrm{PVT}) \in \mathbb{Z}_p$.

5) Compute $b \coloneqq H_2(z, v, M) \in \mathbb{Z}_p$.

6) Compute $w \coloneqq a(b + v\mathrm{SK}_{\mathrm{ID}})^{-1} \in \mathbb{Z}_p$.

7) Set $S \coloneqq (v, w, \mathrm{PVT}) \in \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{G}$.

NOTE: If $b + v\mathrm{SK}_{\mathrm{ID}} = 0$ in step 6, signature generation fails. In this case, restart signature generation with a different choice of $a \in \mathbb{Z}_p^*$ in step 1.

D.4.3.5 Verify

Input: Message $M \in \{0,1\}^*$
Signature $S \in \mathbb{G} \times \mathbb{Z}_p$
Public identifier $\mathrm{ID} \in \{0,1\}^*$
Master public key $\mathrm{MPK} \in \mathbb{G}$

Output: Accept or reject

1) Parse $S \coloneqq (v, w, \mathrm{PVT}) \in \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{G}$.

2) Compute $z \coloneqq H_1(\mathrm{ID}, \mathrm{PVT}) \in \mathbb{Z}_p$.

3) Compute $b \coloneqq H_2(z, v, M) \in \mathbb{Z}_p$.

4) Compute $Q \coloneqq \mathrm{MPK} + z\mathrm{PVT} \in \mathbb{G}$.

5) Compute $R \coloneqq t(bG + vQ) \in \mathbb{G}$.

6) Compute $v' \coloneqq R_x \in \mathbb{Z}_p$, the x-coordinate of the point R.

7) If $v = v'$, accept.

8) Otherwise, reject.

Verification succeeds when $\mathrm{PVT} = tG$ since:

$$Q = \mathrm{MPK} + z\mathrm{PVT} = sG + tzG = (s + tz)G = \mathrm{SK}_{\mathrm{ID}}G$$

and:

$$R = w(bG + vQ) = a(b + v\mathrm{SK}_{\mathrm{ID}})^{-1}(b + v\mathrm{SK}_{\mathrm{ID}})G = aG = P.$$

D.5 Quantum-safe IBC

The risk from quantum computing to asymmetric cryptography described in [i.21] applies to many forms of IBC. Therefore in like manner to considerations to replace algorithms that are susceptible to attack by quantum computers the same risk analysis applies to IBC algorithms.

ETSI TR 103 618 [i.22] addresses the application of a quantum-safe hierarchical Identity-based Encryption scheme (the LATTE scheme which is a Ring-Learning With Errors based Hierarchical Identity-based Encryption scheme that adapts the Ducas, Lyubashevsky and Prest IBE scheme). The conclusions from ETSI TR 103 618 [i.22] are repeated here for ease of reading:

- encryption and decryption are straightforward to implement and are fast, even on constrained devices;
- key generation, extraction and delegation can be optimized using the structure in cyclotomic rings; and
- the security proof provides a level of reassurance and the practical security is increasingly well understood.

On the other hand:

- extraction and delegation require higher-precision arithmetic and can be slow, even on desktops or servers;
- ciphertext sizes can be large and do not scale well as the number of levels in the hierarchy increases; and
- side-channel and other implementation vulnerabilities are not yet well understood.

LATTE is a relatively new scheme and will benefit from further research to improve performance, reduce bandwidth requirements and develop effective side-channel protections. Similarly, to mitigate some of the issues around the use of identity-based and hierarchical Identity-based Encryption it is also worth investigating whether LATTE can support distributed key extraction, more efficient revocation mechanisms or mediated decryption.

Thus Quantum Safe IBC is summarized as a relatively new topic that is being actively studied in academia and industry but not yet mature on the market.

Annex E: Generalizing IBE towards Identity-based Cryptography and Attribute Based Cryptography

Functional Encryption (FE) [i.2] is a generalization of the notion of public-key encryption, wherein the public key is used for encryption or decryption of material where the inverse function is performed by the private key. In the generalized form of FE the means by which the public-key, private-key, pair is determined is specific to each class of FE.

Unlike standard encryption schemes, FE allows for a more fine-grained control of the decryption capabilities of third parties. More precisely, let F describe the class of functions associated with the functional encryption scheme and let f be a particular function in F. A functional encryption for F is then described by a set of four algorithms. The setup algorithm takes a security parameter and generates a public key for the system together with a master secret key. The key generation algorithm takes as input an index for the particular function f in F and generates a token sk[f] for f. To encrypt a message x, the sender can simply run the encryption algorithm on input x and the public key to obtain a ciphertext. Then, given the encryption of a message x, the holder of the token sk[f] for the function f, should be able to compute the value of f(x) but nothing else about the encrypted data.

NOTE 1: The 4 phases or algorithm steps described above are shown for IBC in clause 4.1 of the present document.

As the following example by Boneh et al. in [i.2] illustrates, functional encryption can be used to express how one wishes to share the data in the encryption process itself. More specifically, let x = (P, m) where m is the data one wishes to share and P is the access policy that describes how they want to share it. Then a user's secret token sk[f] can check whether a user's credentials or attributes match this policy and only reveal m in this case. If P describes a general access policy, then the corresponding system would be equivalent to the concept of *attribute-based encryption* [i.6]. Likewise, if P describes the identity of the receiver, then the corresponding system would be equivalent to the concept of *identity-based encryption* [i.3].

NOTE 2: The concept above of embedding access control policies into the encryption process is described for IBC in clause 5.2.8 of the present document.

Functional encryption encompasses and unifies many of the advanced encryption schemes in use today, such as *identity-based encryption* [i.3], *searchable public-key encryption* [i.4], *identity-based encryption with wildcards* [i.5], and *attribute-based encryption* [i.6].

History

Document history		
V1.1.1	March 2022	Publication