IEEE P1363 / D13 (Draft Version 13). Standard Specifications for Public Key Cryptography

Annex C (Informative). Rationale.

Comments and suggestions are welcome. Please contact the chair, Ari Singer, at singerar@pb.com.

ANNEX C (Informative) Rationale

This annex is presented in the form of questions and answers.

C.1 General

C.1.1 Why are there three families of cryptographic techniques?

All three of the families (DL, EC, and IF) are established in the marketplace. They have different advantages and disadvantages, such as performance, key size, code size, availability of cryptanalytic research, patent coverage and use in other standards. The goal of this standard is not to restrict users to a single choice, but rather to provide a framework that would allow selection of methods appropriate for particular applications. Since implementers of this standard need not implement it in its entirety, they have the option to choose the techniques that best suit their needs.

C.1.2 Why are primitives and schemes separated?

In this standard, primitives are presented as mathematical operations, while schemes are presented in a general form based on certain primitives and additional techniques such as encoding methods. Historically, primitives were discovered based on number-theoretic one-way functions. Encoding methods and other components of the scheme were added later to achieve particular security attributes, and tend to focus more on bit-string manipulation. The general form of the scheme lays down a good framework to allow alternative techniques to be included in a given scheme in future versions of the standard. An implementation can conform to either a primitive or a scheme (see Annex B). Hardware components, in particular, may find it advantageous to conform to a primitive, because primitives tend to be less flexible and change less than encoding methods and other components of a scheme. Note, however, that primitives alone are not intended to provide security (see Section 4).

C.1.3 How were the decisions made regarding the inclusion of individual schemes?

Three types of schemes are defined in this standard: key agreement, digital signatures, and public-key encryption. In practice, these are the most basic yet important functions in a public-key cryptosystem. Other types of schemes (e.g., identification schemes) may be included in future versions of the standard.

The main purpose for the current version of the standard is to specify the basic public-key techniques in a common way. Additional techniques remain to be specified, which could eventually be added to the standard. However, many of those additional techniques require further development before being standardized, whereas the basic techniques in the current version are relatively more established. To facilitate the completion of the work on basic techniques while also providing a forum for discussing additional techniques, the working group decided to have two separate projects: P1363 and P1363a. P1363a will supplement the more established techniques already covered in the P1363 document, and it is intended that the two documents will be merged during future revisions.

The selection of schemes in the current version of the standard was based on the above decision. Some reasons for the selection of each individual scheme can be found in Section C.3.

C.1.4 Why are constraints on key sizes not specified?

Key sizes are chosen in accordance with security requirements, and these are application-dependent. As the focus of the standard is specifications of public-key techniques, not security certification, no constraints are given on the key size for any of the schemes. However, recommendations on key sizes for each family of techniques are included in Annex D.

C.1.5 Why are message encoding methods for encryption and signature needed?

The mathematical structure of certain "raw" cryptographic algorithms can lead to a number of delicate attacks, such as chosen ciphertext attacks (where one obtains the decryption of a ciphertext by asking for the decryption of an apparently unrelated ciphertext) or chosen message attacks (where one obtains a signature of a message after requesting signatures for apparently unrelated messages). An appropriate message encoding method provides a systematic way of thwarting all such attacks. Quite a few encoding methods for encryption and signature have appeared in academic literature and other standards. The methods included in this standard are well established techniques. It is highly recommended to use these methods when implementing the schemes specified in the standard. More discussions on security properties of message encoding methods are given in Annex D.

C.1.6 Why are key derivation functions needed?

A key derivation function is needed in the DL/EC key agreement schemes. One reason is that the output from a DL/EC secret value derivation primitive may not be appropriate to use directly as a session key due to a bias in some of the bits of the derived secret value. A good key derivation function helps to remove such bias and to use all the entropy in the output of the secret value derivation primitive. Another reason is that in many cases it is useful to be able to derive more than one key from a secret value. In addition, key derivation functions as defined in the standard also take as input key derivation parameters, which allow one to specify additional information related to the derived key as well as the parties in the key agreement scheme.

C.1.7 Why are data formats for input/output, keys, and domain parameters not normative?

Some data formats for input/output, keys and domain parameters are specified in the informative Annex E. A choice of a particular data format does not affect security, as long as the choice is unambiguous and known to all parties involved and provides for adequate and unambiguous representation of the relevant data. Two implementations with different choices of data formats may not be interoperable. They will need to convert between each other's formats to achieve interoperability. The data formats are not normative because some applications may not require interoperable formats.

C.2 Keys and domain parameters

C.2.1 Why have two types of fields for the DL and EC families?

Modular arithmetic is already implemented in many systems and may have performance advantages in software. On the other hand, arithmetic in finite fields of characteristic 2 may have performance advantages in hardware. There are also security issues related to the two types of fields (see Annex D for more discussion). In addition, certain techniques in both cases are covered by patents, and a particular application will have to weigh these tradeoffs when deciding what type of field to use.

C.2.2 Why allow multiple representations for $GF(2^m)$?

While there is a single most commonly used representation for $GF(p)$, there are multiple natural commonly used representations for $GF(2^m)$. There are performance tradeoffs in the use of each representation. When choosing a representation, one needs to consider whether hardware or software performance is more important as well as time complexity, memory complexity, and relevant patents.

While it would have been possible to standardize one representation and require that applications that want to do computations in a different representation convert from one to the other, the expense of basis conversion may be unacceptable in certain applications. The aim instead was to provide a flexible framework within which applications will decide how to interoperate. Therefore, recommendations on

two particular representations are provided. It is possible that other representations for $GF(2^m)$ as well as $GF(p)$ will be provided in P1363a.

C.3 Schemes

C.3.1 For the DL and EC families, why have three key agreement schemes (-DH1, -DH2, and -MQV)?

The schemes DL/ECKAS-DH1 are based on the traditional Diffie-Hellman key agreement schemes in which each party contributes one key pair. The scheme DL/ECKAS-DH2 is based on the so-called "unified model" for key agreement in which each party contributes two key pairs. The schemes DL/ECKAS-MQV also utilize two key pairs from each party, but are more computationally efficient than DL/ECKAS-DH2. Certain security attributes can be achieved in different ways by the schemes, as further discussed in Annex D.

The reason all three schemes are in the standard is that there are tradeoffs in efficiency, depending on how the security attributes are achieved with each of the schemes. There may also be differences in patent coverage. When choosing a scheme, one needs to consider the desired security attributes, communication, time and memory complexity, and relevant patents.

C.3.2 For the DL and EC families, why have the "compatibility" option for the DHC and MQVC primitives?

The "compatibility" option provides flexibility to implementers. The "compatible" versions of DHC and MQVC are likely to achieve greater interoperability with existing DH and MQV implementations, but the "incompatible" versions may have some implementation advantages as they may be easier to layer on top of existing DH and MQV implementations. (In particular, it may be convenient in some architectures to view the cofactor multiplication as an additional step applied after ordinary DH and MQV.) Concerns about interoperability can be addressed in profiles and other standards based on this standard.

C.3.3 For the EC and DL families, why have both DSA and NR signature schemes with appendix?

The scheme DLSSA-DSA is a generalization of the U.S. Government Digital Signature Standard (DSS) specified in FIPS-186 [FIP94b] and ANSI X9.30 [ANS97a], and it has withstood extensive cryptanalysis. The scheme ECSSA-DSA is the elliptic curve analog of DLSSA-DSA, and a similar scheme called ECDSA is specified in the ANSI X9.62 draft standard [ANS98e].

Note that FIPS-186/ANSI X9.30 (and ANSI X9.62) mandate the use of the hash function SHA-1 with DSS (and ECDSA), and they have restrictions on the sizes of the DL and EC parameters. However, this standard attempts to provide a more flexible specification by allowing other suitable hash functions and a larger range of key sizes. A particular application may find that its key sizes need to be shorter or longer than DSS allows, and that a different hash function better suits its needs. Implementations of DLSSA-DSA and ECSSA-DSA can be compliant with FIPS-186/ANSI X9.30 and ANSI X9.62, respectively.

The schemes DL/ECSSA-NR have the advantages of better performance and smaller code size (in particular because they do not require computation of a multiplicative inverse in a finite field). Also, DL/ECSP-NR and DL/ECVP-NR, the basic primitives for the schemes, allow for message recovery, whereas DL/ECSP-DSA and DL/ECVP-DSA do not. Should a signature scheme with message recovery be necessary for the DL or EC system, it can be constructed based on DL/ECSP-NR and DL/ECVP-NR, although such a scheme is not currently specified in the standard. See next question C.3.4.

C.3.4 For the DL and EC families, why are there no signature schemes with message recovery?

A signature scheme with message recovery requires the use of an encoding method, which, in particular, adds redundancy to a message. The working group was not aware of an encoding method that would be acceptable for standardization for the DL and EC family. Should an application require a DL or EC signature scheme with message recovery, it may use an encoding method of its choice with the NR signature and verification primitives (see Annex B). It is also possible a suitable encoding method will be established during the P1363a effort (see Question C.1.3). Note that ISO/IEC 9796-4 [ISO98g] (currently in draft stage) defines a signature scheme with message recovery based on the Nyberg-Rueppel primitive.

C.3.5 For the DL and EC families, why are there no encryption schemes?

A DL or EC encryption scheme in general requires the use of an encoding method that provides the security properties as described in D.5.3. In addition, since EC keys commonly used in practice are fairly short and hence can be shorter than the data to be encrypted; an encoding method should be designed to also handle this size difference. Several proposals on DL and EC encryption schemes were presented to the P1363 working group to address these issues. However, none of the proposals was considered mature enough, and it was decided that they would benefit from further study before being standardized. It is likely that a suitable encryption scheme for the DL and EC families will be established during the P1363a effort (see Question C.1.3). Note that ANSI X9.63 [ANS98f] (currently in draft stage) defines encryption schemes for the EC family.

C.3.6 For the IF family, why have both RSA and RW signature schemes?

Both RSA and Rabin-Williams (RW) are established signature schemes based on the integer factorization problem. The RSA signature schemes are well understood and widely used. The Rabin-Williams signature verification is generally faster than the RSA signature verification if the public exponent of 2 is used; however, the RSA signature generation has a code-size and speed advantage since there is no need to compute the Jacobi symbol. Both RSA and Rabin-Williams signatures are described in the appendix to ISO/IEC 9796-1 [ISO91].

Note that there are two signature and verification primitives (IFSP/IFVP) for RSA, namely, RSA1 and RSA2. RSA1, which specifies just the "raw" RSA operation, has been widely used in various implementations and protocols, such as PKCS #1 ([PKC93], [PKC98]), SET ([MV97]) and S/MIME ([DHR98a], [DHR98b]). RSA2 has an extra simple step to adjust the most significant bit of the signature to zero, and hence can save one bit (and potentially one byte for some key sizes) compared with RSA1. RSA2 is a component in ANSI X9.31 [ANS98a] and ISO/IEC 9796-1.

C.3.7 For the IF family, why are there no signature schemes with message recovery?

Unlike the DL and EC cases, there is a well-established encoding method for the IF case of signatures with message recovery, specified in ISO/IEC 9796-1 [ISO91]. However, work by Coron, Naccache and Stern [CNS99] and Coppersmith, Halevi and Jutla [CHJ99] demonstrated practical chosen-message attacks against this method. The working group decided that the attacks warranted exclusion of this method from the standard. The working group was not aware of another encoding method that would have been ready for standardization for the IF family. Should an application require an IF signature scheme with message recovery, it may use an encoding method of its choice with the IF signature and verification primitives (see Annex B). It is also possible a suitable encoding method will be established during the P1363a effort (see Question C.1.3). Note that ISO/IEC JTC1 SC27 (the subcommittee responsible for the ISO/IEC 9796 series of standards) is currently considering modifications to the encoding method of ISO/IEC 9796-1, as well as other methods, which will be issued in the ISO/IEC 9796 series.

C.3.8 For the IF family, why are there no key agreement schemes?

In general, key agreement can be achieved using techniques based on integer factorization. For example, each party can transport a secret value using an IF encryption scheme to the other party and a shared secret key can then be derived using the two secret values. However, such methods are better described as protocols rather than as schemes in the model given in Section 4. This may be a semantic distinction between 'protocol' and 'scheme'. Under the model in Section 4, a key agreement scheme is a collection of related operations by which parties can agree on one or more secret keys in some protocol. While one can use IF encryption and decryption operations to agree upon secret keys, those operations are more naturally described in an encryption scheme. It is possible that key agreement techniques based on the IF family will be specified during the P1363a effort.