Standards for Efficient Cryptography

SEC 4: Elliptic Curve Cryptography

Certicom Research

Contact: René Struik (rstruik@certicom.com)

Working Draft
June 9, 2006
Version 1.1r1

Contents

1 Introduction **1**

1.1 Overview . . . 1

1.2 Aim . . . 1

1.3 Compliance . . . 1

1.4 Document Evolution . . . 1

1.5 Intellectual Property . . . 2

1.6 Organization . . . 2

2 ECQV Implicit Certificate Scheme **3**

3 Validation of Implicit Certificate Information **4**

3.1 ECQV Implicit Certificate Information Validation Primitive . . . 5

4 ECQV Implicit Certificate Generation Scheme **6**

4.1 Initiator Transformation . . . 7

4.2 Responder Transformation . . . 8

5 ECQV Implicit Certificate Processing Transformation **9**

A Glossary **11**

A.1 Terms . . . 11

A.2 Acronyms . . . 12

A.3 Notation . . . 12

B Commentary **13**

B.1 Implicit Certificate Overview . . . 13

B.1.1 Principle . . . 13

B.1.2 Properties . . . 14

B.1.3 Applications . . . 14

B.2 Commentary on Section 2 – Certificate Scheme . . . 14

B.2.1 Properties . . . 14

B.2.2 Proof of Knowledge . . . 15

B.2.3 Security Considerations . . . 16

B.3 Commentary on Appendix C – X.509 Implicit Certificates 16

C X.509 Implicit Certificates 17

C.1 Representation of ECQV Certificate Generation Messages 17

C.1.1 Representation of ECQV Certificate Request Message 17

C.1.2 Representation of ECQV Certificate Response Message 19

C.2 Representation of ECQV Implicit Certificates 20

C.2.1 ASN.1 Module 22

D References 23

List of Tables

4 The ECQV implicit certificate scheme 15

List of Figures

1 Introduction

1.1 Overview

Implicit certificates are an alternative to traditional public-key certificates and can be more efficient than traditional certificates. This document specifies implicit certificate schemes based on elliptic curve cryptography. In particular, it specifies the ECQV Implicit Certificate Scheme and how to use this scheme within the X.509 certificate format.

Implicit certificate schemes are intended for general applications within computer and communications systems, but are particularly suited in application environments where resources, such as bandwidth, computing power, and storage, are limited.

1.2 Aim

The aim of this document is to facilitate deployment of implicit certificates by completely specifying efficient and well-understood implicit certificate schemes based on ECC.

1.3 Compliance

Implementations may claim compliance with the cryptographic schemes specified in this document provided the external interface (input and output) to the schemes is identical to the interface specified here. Internal computations may be performed as specified here, or may be performed via an equivalent sequence of operations.

Note that this compliance definition implies that conformant implementations must perform all the cryptographic checks included in the scheme specifications in this document. This is important because the checks are essential to the prevention of subtle attacks.

It is intended to make a validation system available so that implementors can check compliance with this document – see the SECG website, `www.secg.org`, for further information.

1.4 Document Evolution

This document will be reviewed every five years to ensure it remains up to date with cryptographic advances. The next scheduled review will therefore take place in October 2005.

Additional intermittent reviews may also be performed from time-to-time as deemed necessary by the Standards for Efficient Cryptography Group.

External normative standards contain provisions, which, through reference in this document, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

1.5 Intellectual Property

The reader's attention is called to the possibility that compliance with this document may require use of an invention covered by patent rights. By publication of this document, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. The patent holder(s) may have filed with the SECG a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license. Additional details may be obtained from the patent holder and from the SECG web site, `www.secg.org`.

1.6 Organization

This document is organized as follows.

The main body of the document focuses on the specification of implicit certificate schemes. Section 2 specifies implicit certificate schemes based on ECC.

The appendices to the document provide additional relevant material. Appendix A gives a glossary of the acronyms and notation used as well as an explanation of the terms used. Appendix B elaborates some of the details of the main body – giving a general introduction to implicit certificates, making security remarks and attributing references. Appendix C describes the use of implicit certificates within the X.509 certificate format, and Appendix D lists the references cited in the document.

2 ECQV Implicit Certificate Scheme

This section specifies the ECQV implicit certificate scheme based on elliptic curve cryptography (ECC) used in this specification.

The implicit certificate scheme is used by three entities – a Certificate Authority CA, a certificate requester U, and a certificate processor V, where U wishes to obtain an implicit certificate from CA in order to convey U's associated public key to V.

The implicit certificate scheme is described in terms of a certificate generation scheme, a certificate processing transformation, and a certificate information validation primitive. CA, U, and V use these schemes as follows, when they wish to communicate.

Prior to use of the scheme, U, V, and CA agree on the parameters with which the scheme shall be used. In particular, this includes the CA generating its public-key pair and U and V obtaining an authentic copy of CA's public key. V uses CA's public key during execution of the implicit certificate processing transformation, while U or V use CA's public key during validation of implicit certificate information (if applicable).

U executes the implicit certificate generation scheme with CA, to interactively compute an elliptic curve public-key pair with CA and to obtain an implicit certificate IC for this public key provided by CA. V executes the implicit certificate processing transformation, to obtain U's purported static public key from U's purported implicit certificate IC presented to V. U or V may execute the implicit certificate information validation primitive, to verify that the implicit certificate IC purportedly provided by CA to U indeed originated from CA and was provided to U.

The implicit certificate processing transformation yields a static public key (and associated keying information) purportedly bound to the claimed holder; evidence that this public key is genuinely bound to this entity is only corroborated via subsequent use of the corresponding private key (e.g., via execution of an authenticated key agreement scheme or a signing transformation involving this public-key pair). This situation is slightly different from that experienced with processing of ordinary certificates (e.g., X.509 certificates), where evidence of the binding between an entity and its public key is obtained via processing of the certificate, whereas evidence that this entity has indeed access to the corresponding private key is only obtained during cryptographic usage of the public key. Thus, with implicit certificates, the binding of an entity and its public and private key can be verified in unison only, during key usage, whereas, with ordinary certificates, the binding between an entity and its public key and the binding between this entity and the corresponding private key can be verified independently. Implicit certificate information validation allows checking the binding between an entity and its public key prior to cryptographic usage. As such, implicit certificate information validation may be used in contexts where evidence similar to that provided by ordinary certificate processing is desirable (e.g., verification of authenticity of the implicit certificate after receipt by its purported holder).

The implicit certificate information validation primitive is specified in Section 3, the implicit certificate generation scheme is specified in Section 4, and the implicit certificate processing transformation is specified in Section 5.

Prerequisites: The following are the prerequisites for the use of the scheme:

1. An infrastructure shall have been established for the operation of the scheme – including a certificate format, certificate processing rules, and unique identifiers. For an example of such an infrastructure, see RFC 3280 [17].

2. Each entity has an authentic copy of the system's elliptic curve domain parameters $D{=}(p, a, b, G, n, h)$ or $D{=}(m, f(x), a, b, G, n, h)$. These parameters shall have been generated using the parameter generation primitive in Section 3.1.1.1 or the primitive specified in Section 3.1.2.1, both of SEC1 [18]. Furthermore, the parameters shall have been validated using the parameter validation primitives in Sections 3.1.1.2 or 3.1.2.2 of SEC1 [18].

3. Each entity shall be bound to a unique identifier (e.g., distinguished names). All identifiers shall be bit strings of the same length *entlen* bits. Entity U's identifier will be denoted by the bit string U. Entity V's identifier will be denoted by the bit string V. Entity CA's identifier will be denoted by the bit string CA.

4. The CA shall be bound to a static public-key pair (w_{CA}, W_{CA}) associated with the system's elliptic curve domain parameters D. The key binding shall include the unique identifier of the entity CA involved. The binding process shall include the validation of the static public key as specified in Section 3.2.2 of SEC1 [18]. Each party shall have evidence that access to the private key corresponding to the public key of CA is restricted to CA itself.

5. A cryptographic hash function *Hash* shall have been chosen for use with the ECQV implicit certificate generation scheme. Let hashlen denote the length in bits of the output value of this hash function.

6. Each entity shall have decided how to represent elliptic curve points as octet strings (i.e., compressed form, uncompressed form, or hybrid form).

7. A fixed representation of octets as binary strings shall have been chosen (e.g., most-significant-bit-first order or least-significant-bit-first order).

3 Validation of Implicit Certificate Information

Implicit certificate information validation refers to the process of checking that an implicit certificate and associated private-key reconstruction data indeed originated from the purported certification authority and that the implicit certificate is valid according to the procedures of the established infrastructure.

When an entity U is required to verify that an implicit certificate and associated private-key reconstruction data indeed originated with CA and that this implicit certificate is valid according to the procedures of the established infrastructure, the following four methods of validation of implicit certificate information are acceptable:

1. U performs explicit validation of the implicit certificate information using the technique described in Section 3.1 (take $V := U$).

2. U performs implicit validation of the implicit certificate information itself by generating the implicit certificate and associated private-key reconstruction data itself using trusted routines.

3. U receives assurance in an authentic manner that a party trusted during the execution of the implicit certificate generation scheme during which the implicit certificate information was generated has explicitly validated the implicit certificate information using the technique described in Section 3.1 (take V to be that trusted party).

4. U receives assurance in an authentic manner that a party trusted during the execution of the implicit certificate generation scheme during which the implicit certificate information was generated has implicitly validated the implicit certificate and associated private-key reconstruction data itself by generating the implicate certificate and associated private-key reconstruction data itself using trusted routines.

U shall know the type of assurance provided.

3.1 ECQV Implicit Certificate Information Validation Primitive

V shall execute the following transformation to obtain assurance that an implicit certificate IC_U and associated private-key reconstruction data s purportedly provided by CA to U indeed originated from CA and was provided to U. V shall obtain an authentic copy of U's and CA's identifiers and an authentic copy of CA's static public key W_{CA}.

Input: The authenticity-checking transformation takes as inputs:

1. An implicit certificate IC_U and associated private-key reconstruction data s (as octet string se) purportedly originating from CA and purportedly provided to U.

2. An ephemeral public key Q_U purportedly owned by U.

Ingredients: The implicit certificate information validation primitive employs the public key validation primitive in Section 3.2.2 of SEC1 [18], one of the hash functions in Section 5.6.2 of ANSI X9.63-2001 [3], and the certificate validation primitive of the established infrastructure.

Actions: V shall proceed as follows:

1. Verify that Q_U is a valid key for the parameters D as specified in Section 3.2.2 of SEC1 [18]. If the validation primitive rejects the key, output 'invalid' and stop.

2. Calculate the hash value H=$\mathit{Hash}(IC_U)$ using the established hash function. If this transformation outputs 'invalid', output 'invalid' and stop.

3. Derive the integer e from H as follows:

   3.1. Set $E = H$ if $\mathit{hashlen} \leq \lfloor \log_2 n \rfloor$; set E as the bit string resulting from H by omitting all but the leftmost $t = \lfloor \log_2 n \rfloor$ bit positions (i.e., E is the truncation of H to its leftmost t bit positions) otherwise.

3.2. Convert E to the integer e as specified in Section 2.3.8 in SEC1 [18].

3.3. If $e = 0$, output ‘invalid’ and stop.

4. Convert se to the integer s as specified in Section 2.3.8 of SEC1 [18].

5. Verify the contents of IC_U according to the established infrastructure. This includes verifying the contents of the certificate, such as the subject’s name and the validity period. If the subject’s name is unequal to U, output ‘invalid’ and stop.

6. Derive BEU and I_U from IC_U according to the procedures of the established infrastructure.

7. Convert BEU to the elliptic curve point B_U as specified in Section 2.3.4 of SEC1 [18].

8. Verify that B_U is a valid key for the parameters D as specified in Section 3.2.2 of SEC1 [18]. If the validation primitive rejects the key, output ‘invalid’ and stop.

9. Derive CA’s identifier from I_U, according to the certificate format specified during the setup procedure,[1] and obtain CA’s static key W_{CA}. If CA’s identifier or CA’s static key is unknown to V, output ‘invalid’ and stop.

10. Verify that the public key $W_U = e \cdot Q_U + s \cdot G$ is equal to the reconstructed public key $W'_U = e \cdot B_U + W_{CA}$. If this check fails, output ‘invalid’ and stop.

Output: If any of the above verifications has failed, then output invalid’ and stop; otherwise, output ‘valid’, accept CA as the source of the implicit certificate IC_U and of the private-key reconstruction data s, and accept IC_U as U’s purported implicit certificate. (V may accept IC_U as U’s genuine implicit certificate provided U evidences knowledge to V of the the private key w_U that corresponds to IC_U.)

4 ECQV Implicit Certificate Generation Scheme

This section specifies the scheme for generating implicit certificates (self-certified public keys).

The scheme is ‘asymmetric’ so two transformations are specified. U uses the transformation specified in section 4.1 to compute an elliptic curve public-key pair and to obtain an implicit certificate for this public key from CA if U is the protocols initiator, and CA uses the transformation specified in Section 4.2 to interactively assist U in computing an elliptic curve public-key pair and to provide an implicit certificate for this public key to U if CA is the protocol’s responder.

If U executes the initiator transformation and CA executes the responder transformation with the corresponding public keying material as input, then U will interactively compute an elliptic curve public-key pair with CA and will obtain an implicit certificate for this public key provided by CA.

[1] This may include the use of side information known to all parties involved (i.e., system-wide parameters implied by the certificate infrastructure). A special case hereof would be the scenario where there is only one CA.

4.1 Initiator Transformation

U shall execute the following transformation to obtain an elliptic curve public-key pair and an implicit certificate for this public key with CA if U is the protocol's initiator. U shall obtain an authentic copy of CA's identifier and an authentic copy of CA's static public key W_{CA}.

Input: This routine does not take any inputs.

Ingredients: The initiator transformation employs the key pair generation primitive in Section 3.2.1 of SEC1 [18], the public key validation primitive in Section 3.2.2 of SEC1 [18], one of the hash functions in Section 5.6.2 of ANSI X9.63-2001 [3], the implicit certificate information validation primitive as specified in Section 3.1, and the certificate validation primitive of the established infrastructure.

Actions: U shall proceed as follows:

1. Use the key pair generation primitive specified in Section 3.2.1 of SEC1 [18] to generate an ephemeral key pair (d_U,Q_U) for the parameters D. Send Q_U to CA.
2. Receive from CA the values IC_U and se. If these values are not received, output 'invalid' and stop.
3. Calculate the hash value H=$Hash(IC_U)$ using the established hash function. If this transformation outputs 'invalid', output 'invalid' and stop.
4. Derive the integer e from H as follows:
   - 4.1. Set $E = H$ if $hashlen \leq \lfloor \log_2 n \rfloor$; set E as the bit string resulting from H by omitting all but the leftmost $t = \lfloor \log_2 n \rfloor$ bit positions (i.e., E is the truncation of H to its leftmost t bit positions) otherwise.
   - 4.2. Convert E to the integer e as specified in Section 2.3.8 in SEC1 [18].
   - 4.3. If $e = 0$, output 'invalid' and stop.
5. Convert se to the integer s as specified in Section 2.3.8 of SEC1 [18].
6. Compute the private key $w_U = s + e \cdot d_U (\mod n)$.
7. Verify the contents of IC_U according to the established infrastructure. This includes verifying the contents of the certificate, such as the subject's name and the validity period. If the subject's name is unequal to U, output 'invalid' and stop.
8. Derive BEU and I_U from IC_U according to the procedures of the established infrastructure.
9. Convert BEU to the elliptic curve point B_U as specified in Section 2.3.4 of SEC1 [18].
10. Verify that B_U is a valid key for the parameters D as specified in Section 3.2.2 of SEC1 [18]. If the validation primitive rejects the key, output 'invalid' and stop.

11. Derive CA's identifier from I_U, according to the certificate format specified during the setup procedure.[2] If CA's identifier is unknown to U, output 'invalid' and stop.

12. Verify that the values IC_U and se received indeed originated from CA as specified in Section 3.

Output: If any of the above verifications has failed, then output 'invalid' and stop; otherwise, output 'valid', accept the authenticity of CA, accept CA as the source of the octet string IC_U, accept IC_U as the implicit certificate provided by CA, and accept w_U as the private key.

4.2 Responder Transformation

CA shall execute the following transformation to provide an implicit certificate to U and to provide evidence to U as to its actual involvement in a real-time communication with U if CA is the protocol's responder. CA shall obtain an authentic copy of U's identifier.

Inputs: The ephemeral public key Q_U purportedly owned by U.

Ingredients: The responder transformation employs the key pair generation primitive in Section 3.2.1 of SEC1 [18], the public key validation primitive in Section 3.2.2 of SEC1 [18], one of the hash functions in Section 5.6.2 of ANSI X9.63-2001 [3], and the certificate generation primitive of the established infrastructure.

Actions: CA shall proceed as follows:

1. Verify the authenticity of the request received from U according to the procedures of the established infrastructure. The checks performed shall include, as a minimum, checking that U is indeed the origin of the request value Q_U and checking that U is authorized to obtain a certificate.

2. Verify that Q_U is a valid key for the parameters D as specified in Section 3.2.2 of SEC1 [18]. If the validation primitive rejects the key, output 'invalid' and stop.

3. Use the key pair generation primitive specified in Section 3.2.1 of SEC1 [18] to generate an ephemeral key pair (d_{CA},Q_{CA}) for the parameters D.

4. Compute the elliptic curve point $B_U = Q_U + Q_{CA}$.

5. Convert the elliptic curve point B_U to the octet string BEU as specified in Section 2.3.3 in SEC1 [18]. (BEU is referred to as the public-key reconstruction data.)

6. Construct 'to-be-signed-certificate data' I_U. This octet string shall contain identification information according to the procedures of the established infrastructure and may also contain other information, such as the intended use of the public key, the serial number of the implicit certificate, and the validity period of the implicit certificate. The exact form of I_U depends on the certificate format specified during the setup procedure.

[2] This may include the use of side information known to all parties involved (i.e., system-wide parameters implied by the certificate infrastructure). A special case hereof would be the scenario where there is only one CA.

7. Construct U's implicit certificate IC_U, according to the procedures of the established infrastructure. This octet string shall contain the octet strings I_U and BEU encoded in a reversible manner. The exact form of IC_U depends on the certificate format specified during the setup procedure.

8. Compute the hash value $H{=}Hash(IC_U)$ using the established hash function. If this transformation outputs 'invalid', output 'invalid' and stop.

9. Derive the integer e from H as follows:

   9.1. Set $E = H$ if $hashlen \leq \lfloor \log_2 n \rfloor$; set E as the bit string resulting from H by omitting all but the leftmost $t = \lfloor \log_2 n \rfloor$ bit positions (i.e., E is the truncation of H to its leftmost t bit positions) otherwise.

   9.2. Convert E to the integer e as specified in Section 2.3.8 in SEC1 [18].

   9.3. If $e = 0$, go to Step 3.

10. Compute the integer $s = e \cdot d_{CA} + w_{CA}(\mod n)$. ($s$ is referred to as the private-key reconstruction data.)

11. Convert s to the octet string se as specified in Section 2.3.7 of SEC1 [18]. Send IC_U and se to U.

Output: If any of the above verifications has failed, then output 'invalid' and stop; otherwise, output 'valid' and accept IC_U as U's purported implicit certificate provided by CA. (CA may accept IC_U as U's genuine implicit certificate provided U evidences knowledge to CA of the corresponding private key w_U.)

5 ECQV Implicit Certificate Processing Transformation

V shall execute the following transformation to obtain U's purported static public key from U's purported implicit certificate. V shall obtain an authentic copy of U's and CA's identifiers and an authentic copy of CA's static public key W_{CA}.

Input: U's purported implicit certificate IC_U provided by CA.

Ingredients: The certificate processing transformation employs the public key validation primitive in Section 3.2.2 of SEC1 [18], one of the hash functions in Section 5.6.2 of ANSI X9.63-2001 [3], and the certificate validation primitive of the established infrastructure.

Actions: V shall proceed as follows:

1. Calculate the hash value $H{=}Hash(IC_U)$ using the established hash function. If this transformation outputs 'invalid', output 'invalid' and stop.

2. Derive the integer e from H as follows:

   2.1. Set $E = H$ if *hashlen* $\leq \lfloor \log_2 n \rfloor$; set E as the bit string resulting from H by omitting all but the leftmost $t = \lfloor \log_2 n \rfloor$ bit positions (i.e., E is the truncation of H to its leftmost t bit positions) otherwise.

   2.2. Convert E to the integer e as specified in Section 2.3.8 in SEC1 [18].

   2.3. If $e = 0$, output 'invalid' and stop.

3. Verify the contents of IC_U according to the established infrastructure. This includes verifying the contents of the certificate, such as the subject's name and the validity period. If the subject's name is unequal to U, output 'invalid' and stop.

4. Derive BEU and I_U from IC_U, according to the certificate format specified during the setup procedure.

5. Convert the octet string BEU to the elliptic curve point B_U as specified in Section 2.3.4 of SEC1 [18].

6. Verify that B_U is a valid key for the parameters D as specified in Section 3.2.2 of SEC1 [18]. If the validation primitive rejects the key, output 'invalid' and stop.

7. Derive CA's identifier from I_U, according to the certificate format specified during the setup procedure,[3] and obtain CA's static key W_{CA}. If CA's identifier or CA's static key is unknown to V, output 'invalid' and stop.

8. Compute the public key $W_U = e \cdot B_U + W_{CA}$.

Output: If any of the above verifications has failed, then output 'invalid' and stop; otherwise, output 'valid' and accept W_U as U's purported static public key. (V may accept W_U as U's genuine static public key provided U evidences knowledge to V of the corresponding private key w_U.)

[3] This may include the use of side information known to all parties involved (i.e., system-wide parameters implied by the certificate infrastructure). A special case hereof would be the scenario where there is only one CA.

A Glossary

This section provides a glossary to the terms, acronyms, and notation used in this document.

Please refer to the glossary of SEC1 [18] for any term, acronym, or notation not specified in this section.

A.1 Terms

Terms used in this document include:

certificate

Information including the public key and identity of an entity, rendered unforgeable by signing hereof by a Certificate Authority.

impersonation-prevention

Assurance that the CA cannot obtain U's private key if U's implicit certificate was created by CA and U together, using the certificate generation scheme.

implicit certificate

Information including public-key reconstruction data and the identity of an entity that, together, constitutes the certificate of that entity.

non-forgeability

Assurance that no adversary can forge an implicit certificate created by some certificate authority CA for an entity U such that it also obtains significant information about U's corresponding private key.

public-key reconstruction data

Value contained in the implicit certificate, from which any party with access to CA's public key can reconstruct the public key. This value may also be used (e.g., by the certificate requester) as an ingredient in validating the authenticity of the implicit certificate.

private-key reconstruction data

Value computed by a CA during the operation of an implicit certificate generation scheme that allows the certificate requester to compute its private key. This value may also be used (e.g., by the certificate requester) as an ingredient in validating the authenticity of the implicit certificate.

to-be-signed-certificate data

Data to be included in a certificate or implicit certificate. This data includes the identity of the certified entity, but may also include other data, such as the intended use of the public key, the serial number of the certificate, and the validity period of the certificate. The exact form of this data depends on the certificate format being used.

traditional certificate

see certificate in Appendix A.1 of SEC1 [18].

A.2 Acronyms

The acronyms used in this document denote:

ECQV	Elliptic Curve Qu-Vanstone. See Section 2.

A.3 Notation

The notation adopted in this document is:

B	Public reconstruction data.
I	To-be-signed-certificate data.
IC	Implicit certificate.

B Commentary

This section provides a commentary on the main body of this document, including implementation discussion, security discussion, and references.

The aim of this section is to supply implementers with relevant guidance. However, this section does not attempt to provide exhaustive information but rather focuses on giving basic information and including pointers to references which contain additional material. Furthermore, this section concentrates on supplying information specific to implicit certificates. Extension commentary on ECC in general – addressing issues like parameter selection and implementation of elliptic curve arithmetic – can be found in Appendix B of SEC1 [18].

The information is likely to change over time, and implementers should therefore survey the state-of-the-art at the time of implementation and carry out periodic reviews subsequent to deployment.

This section is organized as follows. Section B.1 provides an introduction to implicit certificates, including their operation and properties and compares them to traditional certificates. Section B.2 provides a commentary on Section 2 of the main body of this document.

B.1 Implicit Certificate Overview

This section provides a commentary on implicit certificates. It discusses their basic principle, properties and possible applications.

B.1.1 Principle

A traditional certificate is a mechanism to bind a public key to an entity, represented by various fields. For simplicity we will denote these fields by I_U, the certificate information data. A user – which could be also a machine – presents itself to a CA, proves its identity I_U and provides its public key W_U to the CA. The CA then binds the public key to the user information I_U. The binding is achieved through a digital signature s by the CA of I_U and the users public key. The certificate is (I_U, W_U, s). This is an explicit binding, since there is an explicit (unforgeable) signature.

An implicit certificate is another mechanism to bind a public key to an entity. However, the certificate does not consist of (I_U, W_U, s), but rather of (I_U, B_U). There is no explicit signature s by the CA, and no explicit public key W_U. Instead, there is a public reconstruction value B_U generated by the user and the CA together. It has the property that one can compute the users public key from B_U, I_U and W_{CA}.

$$W_U = f(B_U, I_U, W_{CA}), \text{ where } f() \text{ is publicly known}$$

Since any triple (I_U, B_U, W_{CA}) generates a public key Q_U, the implicit certificate (I_U, B_U) does not demonstrate that CA has bound B_U to I_U. However, if an entity also demonstrates the knowledge of the private key corresponding to the reconstructed public key, then assurance is provided that CA has bound I_U to B_U.

This is what is called *implicit binding.*

B.1.2 Properties

Forgeability Unlike regular certificates, there is no digital signature. In fact, one could simply choose an arbitrary identity I and a random value for B. Together with the public key of a CA, this generates a certified public key for the entity identified by I. However, if one constructs an implicit certificate in such a way (without interacting with a CA as described in section 2), it is believed to be infeasible to compute the private key that corresponds to the public key generated by the certificate.

Efficiency An advantage of implicit certificates is clearly, that they contain only the public reconstruction data instead of the subjects public key and the CA's signature. This means, that in theory they should be smaller than regular certificates.

Another efficiency issue is the verification process. While regular certificates have to be verified, this step is not necessary with implicit certificates. However, with implicit certificates one has to compute the public key. It depends on the particular implementation, whether computing the public key is more efficient than verifying a signature.

B.1.3 Applications

Implicit certificates achieve a binding of a public key to an entity. Implicit certificates can therefore be deployed anywhere where regular certificates can be.

However, some restrictions on key parameters and key pair generation apply (see Appendix B.2.1). It has to be decided upon a particular situation if it the efficiency advantage can make up for these restrictions.

B.2 Commentary on Section 2 – Certificate Scheme

This section provides a commentary on the ECQV implicit certificate scheme. It discusses properties specific to this scheme, certain parts of the ECQV implicit certificate scheme and some security considerations specific to this type of certificate. For convenience, the scheme is summarized in Table 4.

B.2.1 Properties

Key Pair Generation In regular certificates, key pair generation and certificate issuing are two independent processes. A user can present an arbitrary public key to a CA that the user wants to get certified. In the presented implicit certificate scheme the situation is somewhat different. When a user requests an implicit certificate for a public key from a CA, this public key (and the private key) is a random result of the computations made by the user and the CA.

This has the direct consequence that once a ECQV implicit certificate is issued, one cannot get another ECQV implicit certificate for the same public key from a different CA. However,

CA		U
		$d_U \in [1..n-1]$
		$Q_U := d_U \cdot G$
	$\xleftarrow[\text{authentic}]{Q_U}$	
$d_{CA} \in [1..n-1]$		
$Q_{CA} := d_{CA} \cdot G$		
$B_U := Q_U + Q_{CA}$		
$\overline{B_U} := B_U$ as octets		
$s := hash(I_U, \overline{B_U}) \cdot d_{CA} + w_{CA}$		
	$\xrightarrow{(I_U, B_U), s, W_{CA}}$	
		$\overline{B_U} := B_U$ as octets
		$w_U := hash(I_U, \overline{B_U}) \cdot d_U + s$
		Verify $hash(I_U, \overline{B_U}) \cdot (B_U - Q_U) + W_{CA} \overset{!}{=} sG$

Table 4: The ECQV implicit certificate scheme

it is possible to get ECQV implicit certificates from different CAs for the same public key, if all the concerning CAs are involved in the key pair generation process.

Key Parameters In order to compute the user's public key from an ECQV implicit certificate, one has to evaluate a function $Q_U = Hash(I_U, \overline{B_U}) \cdot B_U + Q_{CA}$. Q_{CA} is a parameter of this function. Since Q_{CA} is defined over a certain elliptic curve, the result of the function must relate to the same curve. But this means, that the constructed user's public key is defined over the same elliptic curve as the CA's public key. Hence, the key pair certified by an ECQV implicit certificate has to be defined over the same elliptic curve parameters as the CA's key pair is.

B.2.2 Proof of Knowledge

When an entity U requests a regular certificate for a public key, it has to prove to the CA the knowledge of the corresponding private key. This is called proof of knowledge. It is to prevent a user from choosing an arbitrary public key, that might already belong to another user, and have it certified. Such a behavior could cause confusion and must be prevented.

In implicit certificates this proof is redundant because the situation is different. There is no public key before issuing the certificate. Further, U has no control over the final value of his public key, making it impossible for U to cause the confusion described above.

B.2.3 Security Considerations

One difference between traditional certificates and implicit certificates is that when presented with a valid normal certificate, one knows that the certificate belongs to *someone*. A valid certificate containing the certificate data string D_U is a proof that the CA signed this certificate for U, and so U knows the private key of the public key included in the certificate. One does not have this guarantee with implicit certificates as described in Section 2. It is trivially possible to construct an implicit certificate (D_U, B_U) such that the private key corresponding to the public key computed as $Q_U = hash(D_U, \overline{B_U}) \cdot B_U + G$ is unknown.

This fact suggests a denial-of-service type attack, where a party is flooded with protocol requests using “faked” implicit certificates. The fact that the private key of the faked public key is unknown is revealed only after the party has performed most of the protocol. Of course, a similar attack can be launched in a system using regular certificates. In this case, though, the cheater would flood a party with various certificates belonging to other entities. The certificates are valid, but the cheater does not know the private key of the corresponding public key.

However, it is possible to prove statements about the security of implicit certificates. Especially, it is important to prove that forging an implicit certificate *and* knowing the corresponding private key is impossible without knowing the CA's private key. This fact has been proven in [4].

B.3 Commentary on Appendix C – X.509 Implicit Certificates

It may be particularly desirable to include a unique identifier for the issuer of the public key in the certificate field `issuerUniqueIdentifier`.

C X.509 Implicit Certificates

This section describes how to use the implicit certificate schemes specified in this document within the X.509 certificate format [8] and the PKIX framework [13].

It may be desirable to use implicit certificate schemes in the context of X.509 and PKIX, because these standards are the most widely deployed PKI standards. Note, however, that these standards do not offer optimal efficiency, so in many environments it may also be desirable to use implicit certificates within other certificate formats.

This section needs to be updated, to allow a wider choice of crypto primitives (besides, e.g., SHA-1).

C.1 Representation of ECQV Certificate Generation Messages

This section describes how to embed the certificate generation scheme messages within standard certificate generation protocol messages.

C.1.1 Representation of ECQV Certificate Request Message

The most widely used certificate request format is the format defined in RFC 2314 [12], which is specified as follows:

```
CertificationRequest ::= SEQUENCE {
   certificationRequestInfo     CertificationRequestInfo,
   signatureAlgorithm           AlgorithmIdentifier{{
                                         SignatureAlgorithms }},
   signature                    BIT STRING
},
```

where

```
CertificationRequestInfo ::= SEQUENCE {
   version                     INTEGER {v1(0), v2(1), v3(2)},
   subject                     Name,
   subjectPKInfo               SubjectPublicKeyInfo{{
                                           PKInfoAlgorithms }},
   attributes                  [0] Attributes{{ CRIAttributes }}
},
```

where

```
SubjectPublicKeyInfo {ALGORITHM:IOSet} ::= SEQUENCE {
   algorithm                   AlgorithmIdentifier {{IOSet},
   subjectPublicKey            BIT STRING
}.
```

An ECQV implicit certificate request message may be represented using the RFC 2314 format, where the cryptographic fields are set as follows:

- `subjectPKInfo`. This field contains the following two subfields:
  - `algorithm`. This subfield identifies the type of key being certified and any necessary associated parameters. Here, `algorithm` shall contain `ecPublicKeyType` as defined in SEC1 [18]:

    ```
    ecPublicKeyType ALGORITHM ::= {
       Parameters IDENTIFIED BY id-ecPublicKey
    },
    ```

    where

    ```
    id-ecPublicKey OBJECT IDENTIFIER ::= {
       ansi-X9-62 keyType(2) id-publicKeyType(1)
    },
    ```

    where

    ```
    ansi-X9-62 OBJECT IDENTIFIER ::= {
       iso(1) member-body(2) us(840) 10045
    }.
    ```

    Here, the parameter field may contain an explicit reference to the elliptic curve domain parameters D established by the CA or may contain the value `NULL`, which indicates that the parameters are not explicitly included, but are deemed inherited from the CA.
  - `subjectPublicKey`. This subfield contains the public request value Q_U, encoded as a value of type `BIT STRING` as follows: First, Q_U is encoded as a value of type `ECPoint` as defined in SEC1 [18]. Then, the most significant bit of the resulting octet string becomes the most significant bit of the `subjectPublicKey` field, and so on. Note that when DER-encoding is being used, this means that the raw public request value without prepended DER-encoding tag and length octets is placed in the `subjectPublicKey` field.
- `signatureAlgorithm`. This field shall contain the identifier for the ECQV implicit certificate scheme. This value may be used by the CA to identify the request as a request for an implicit certificate.

  ```
  ecqvWithSHA1 ALGORITHM ::= {
     NULL IDENTIFIED BY ecqv-with-SHA1
  },
  ```

  where

  ```
  ecqv-with-SHA1 OBJECT IDENTIFIER ::= {
     iso(1) identified-organization(3) certicom(132) algos(3) 1
  }.
  ```

- `signature`. This field shall contain one of the following two values:
  - the zero-length bit string;
  - U's ECDSA signature on the DER-encoded `certificationRequestInfo` field computed using the ephemeral private key d_U, encoded as a value of type `ECDSA-Sig-Value`:

```
ECDSA-Sig-Value ::= SEQUENCE {
   r       INTEGER,
   s       INTEGER
}
```

  Note that the entire encoding of `ECDSA-Sig-Value` is placed into the signature `BIT STRING`. Inclusion of U's signature in `signature` helps providing a proof-of-possession of d_U to CA.

C.1.2 Representation of ECQV Certificate Response Message

The certificate response format defined in RFC 2510 [14] is specified as follows:

```
CertRepMessage ::= SEQUENCE {
   caPubs                      [1] SEQUENCE SIZE (1..MAX) OF
                                  Certificate         OPTIONAL,
   response                    SEQUENCE OF CertResponse
},
```

where

```
CertResponse ::= SEQUENCE {
   certReqId                   INTEGER,
   status                      PKIStatusInfo,
   certifiedKeyPair            CertifiedKeyPair        OPTIONAL,
   rspInfo                     OCTET STRING            OPTIONAL
},
```

where

```
CertifiedKeyPair ::= SEQUENCE {
   certOrEncCert               CertOrEncCert,
   privateKey                  [0] EncryptedValue      OPTIONAL,
   publicationInfo             [1] PKIPublicationInfo  OPTIONAL
},
```

where

```
EncryptedValue ::= SEQUENCE {
   intendedAlg                 [0] AlgorithmIdentifier OPTIONAL,
   symmAlg                    [1] AlgorithmIdentifier OPTIONAL,
   encSymmKey                 [2] BIT STRING          OPTIONAL,
   keyAlg                     [3] AlgorithmIdentifier OPTIONAL,
   valueHin                   [4] OCTET STRING        OPTIONAL,
   encValue                   BIT STRING
}.
```

An ECQV implicit certificate response message may be represented using the RFC 2314 format [14, 15], where the cryptographic fields are set as follows:

- `certOrEncCert`. This field shall contain the implicit certificate IC_U, formatted as specified in Appendix C.
- `intendedAlg`. This field shall contain the `ecqvWithSHA1` algorithm identifier for the ECQV implicit certificate scheme. (This denotes that `encValue` contains the private-key reconstruction value.)
- `encValue`. This field shall contain the private-key reconstruction value s, encoded as a value of type `BIT STRING` as follows: First, s is converted to an octet string of length $\lceil log_2(n/8)\rceil$ (where n is the order of the curve) using the conversion routine specified in 2.3.7 of SEC1 [18]. Then, the most significant bit of the value of the resulting octet string becomes the most significant bit of the `encValue` field, and so on.

C.2 Representation of ECQV Implicit Certificates

This section describes how to embed implicit certificates within the X.509 certificate format.

The X.509 certificate format is specified as follows:

```
Certificate ::= SEQUENCE {
   tbsCertificate             TBSCertificate,
   signatureAlgorithm         AlgorithmIdentifier,
   signatureValue             BIT STRING
},
```

where

```
TBSCertificate ::= SEQUENCE {
   version                [0] Version DEFAULT v1,
   serialNumber               CertificateSerialNumber,
   signature                  AlgorithmIdentifier,
   issuer                     Name,
   validity                   Validity,
   subject                    Name,
   subjectPublicKeyInfo       SubjectPublicKeyInfo,
   issuerUniqueIdentifier   [1] IMPLICIT UniqueIdentifier OPTIONAL,
         -- If present, version shall be v2 or v3
   subjectUniqueIdentifier  [2] IMPLICIT UniqueIdentifier OPTIONAL,
         -- if present, version shall be v2 or v3
   extensions               [3] Extensions OPTIONAL,
         -- if present, version shall be v3
},
```

where

```
   SubjectPublicKeyInfo ::= SEQUENCE {
      algorithm                  AlgorithmIdentifier{{
                                          SupportedAlgorithms }},
      subjectPublicKey           BIT STRING
   }.
```

An ECQV implicit certificate may be represented using the X.509 certificate format [13], where the cryptographic fields are set as follows:

- signature and signatureAlgorithm fields. These fields shall contain the identifier of the algorithm used by the CA to compute the implicit certificate. For ECQV implicit certificates, this identifier shall be set to ecqvWithSHA1.
- subjectPublicKeyInfo. This field contains the following two subfields:
  - algorithm. This subfield shall contain the identifier for the certified key type. Since ECQV certifies an elliptic curve public key and uses the same parameters as the CA, it shall contain ecPublicKeyType. The parameter field may contain an explicit reference to the elliptic curve domain parameters D established by the CA or may contain the value NULL, which indicates that the parameters are not explicitly included, but are deemed inherited from the CA.
  - subjectPublicKey. This subfield shall contain the zero-length bit string.
- signatureValue. This field shall be used for conveying the public reconstruction data B_U of the implicit certificate, encoded as value of type BIT STRING as follows: First, B_U is encoded as a value of type ECPoint as defined in SEC1 [18]. Then, the most significant bit of the value of the resulting octet string becomes the most significant bit of the value of the signatureValue field, and so on. Note that when DER-encoding is being used, this means that the raw public reconstruction data without prepended DER-encoding tag and length octets is placed in the signatureValue field.

The implicit certificate IC_U is the DER-encoding of this syntax.

C.2.1 ASN.1 Module

The only new ASN.1 syntax introduced is summarized below. These lines are included in the certicom module `iso(1 identified-organization(3) certicom(132)`.

```
ecqvWithSHA1 ALGORITHM ::= {
    NULL IDENTIFIED BY ecqv-with-SHA1
}
ecqv-with-SHA1 OBJECT IDENTIFIER ::= {
   iso(1) identified-organization(3) certicom(132) algos(3) 1
}
```

D References

[1] ANSI X9.62-1998: ‘Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature Algorithm (ECDSA),’ American Bankers Association, 1999,

[2] ANSI X9.30-1993, Part 2: ‘Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry: The Secure Hash Algorithm 1 (SHA-1) (Revised)’, American Bankers Association, 1993.

[3] ANSI X9.63-2001, ‘Public Key Cryptography for the Financial Services Industry – Key Agreement and Key Transport Using Elliptic Curve Cryptography,’ American Bankers Association, November 20, 2001.

[4] D.R.L. Brown, R.P. Gallant, S.A. Vanstone, ‘Provably Secure Implicit Certificate Schemes, in *Proceedings of the Fifth International Conference on Financial Cryptography – FC 2001*, P.F. Syverson, Ed., Lecture Notes in Computer Science, Vol. 2339, pp. 156-165, Berlin: Springer, 2002.

[5] FIPS Pub 140-2, ‘Security requirements for Cryptographic Modules,’ Federal Information Processing Standards Publication 140-2, US Department of Commerce/N.I.S.T., Springfield, Virginia, June 2001 (supersedes FIPS Pub 140-1).

[6] FIPS Pub 197, ‘Advanced Encryption Standard (AES),’ Federal Information Processing Standards Publication 197, US Department of Commerce/N.I.S.T., Springfield, Virginia, November 26, 2001.

[7] FIPS Pub 198, ‘The Keyed-Hash Message Authentication Code (HMAC),’ Federal Information Processing Standards Publication 198, US Department of Commerce/N.I.S.T., Springfield, Virginia, March 6, 2002.

[8] ITU-T Recommendation X.509: ‘Information Technology - Open Systems Interconnection - The Directory: Authentication Framework,’ August 1997.

[9] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of Applied Cryptography*, Boca Raton: CRC Press, 1997.

[10] PKCS #10, ‘Certification Request Syntax Standard, v.1.7,’ RSA Laboratories, May 26, 2000.

[11] D. Pointcheval, J. Stern, ‘Security Proofs for Signature Schemes’, in *Proceedings of Advances in Cryptology – Eurocrypt ’96*, U. Maurer, Ed., Lecture Notes in Computer Science, Vol. 1070, pp. 387-398, New York: Springer Verlag, 1996.

[12] RFC 2314, ‘PKCS #10: Certification Request Syntax Version 1.5,’ Internet Request for Comments 2314, B. Kaliski, March 1998.

[13] RFC 2459, ‘Internet X.509 Public Key Infrastructure Certificate and CRL Profile,’ Internet Request for Comments 2459, R. Housley, W. Ford, W. Polk, W. Solo, January 1999.

[14] RFC 2510, ‘Internet X.509 Infrastructure: Certificate Management Protocols,’ Internet Request for Comments 2510, C. Adams, S. Farrell, March 1999.

[15] RFC 2511, ‘Internet X.509 Infrastructure: Certificate Request Message Format,’ Internet Request for Comments 2511, M. Myers, C. Adams, D. Solo, D. Kemp, March 1999.

[16] RFC 3279, ‘Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and CRL Profile,’ Internet Request for Comments 3279, L. Bassham, R. Housley, W. Polk, April 2002.

[17] RFC 3280, ‘Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,’ Internet Request for Comments 3280, R. Housley, W. Polk, W. Ford, D. Solo, April 2002.

[18] SEC 1: ‘Elliptic Curve Cryptography’, Standards for Efficient Cryptography, Version 1.0, Certicom Research, September 20, 2000.

[19] SEC 2: ‘Recommended Elliptic Curve Domain Parameters’, Standards for Efficient Cryptography, Version 1.0, Certicom Research, September 20, 2000.