
Review of
SEC 1: Elliptic Curve Cryptography
(Version 0.4)

Phillip Rogaway
University of California at Davis
September 1999

1 Summary

This note represents my review of *Standards for Efficient Cryptography — SEC 1: Elliptic Curve Cryptography* (Draft 0.4 — August 1999). I carried out this review at the request of Certicom Corporation.

This is an excellent document. Indeed I find very little to criticize. It nicely spells out all of the details for doing encryption, digital signatures, and key distribution using elliptic curve cryptography. It accomplishes its stated purposes well. It is highly readable, with simple and clear algorithm descriptions. It is long but also accurate and almost self-contained (beyond the fact that two related documents will give test vectors and specific domain parameters). The document gives almost everything needed to implement the covered techniques, all the way down to teaching the basic mechanics of doing arithmetic in EC groups. Appendices give nice security discussions, and there is a good reference list and glossary.

The problem addressed is a timely one, as there is strong current interest in extending the set of methods which have been standardized. Standards for EC methods are now emerging from the ANSI, IEEE, and ISO. This draft is well-aligned with these other efforts.

EC techniques have been somewhat slow to get standardized. Some reasons for this may include the perceived sufficiency of RSA for most public-key applications, patent issues, the difficulty of understanding EC mathematics, and the fact that there are so many details (including so many groups) to choose among. The benefit of overcoming these obstacles is efficiency: at present, EC cryptography seems to be substantially more efficient than public-key cryptography based on other techniques.

Detailed comments can be found in Section 2. The comments are all quite minor.

The vast majority of my comments are just to list typos or give minor suggestions about style. These comments are really directed at the editor, who will probably decide that I should find something better to do than to criticize every missing comma.

There are a few minor editorial suggestions of a cryptographic nature. The most significant is probably to remove any mention of plaintext awareness (which is mentioned in connection with ECAES). While reference [11] claimed plaintext awareness (under stated assumptions) for the cryptographic scheme from which ECAES is derived, the full version of paper [11] (reference [1]) actually backed off of that claim, and directly proved the desired security results. For a history, see the more detailed comments in Section 2.

I was at first concerned that an optimization made to [11,1] for the ECAES scheme—the removal of the ephemeral private key from the domain of the KDF—would negatively impact provable security. However, I and my co-authors Michel Abdalla and Mihir Bellare have looked at this, and we concur that, because of the *particular* groups in which this standard is operating (groups of prime order), this optimization does not cause any problems. Somebody was pretty clever—or pretty lucky—to make this optimization and get away with it.

I thought that the key exchange portion of this document seemed less mature than the rest (though I understand that this is largely the nature of the beast). For Sections 4 and 5 I know exactly what the goal is—the document tells me quite clearly. But for key agreement, the document is more vague about what are the goals. It says that the DH scheme and the MQV scheme are designed to meet a variety of goals, and then some words are listed. But the document never tells me how I get what goals. The result is that I am unsure exactly what is intended for the DH and MQV schemes to be doing, and under what circumstances. The MQV method, in particular, is very complicated, and I do not yet know what can be, or has been, proven about it. (I have not yet read references [55]; perhaps that will help.)

One could try to include a little chart: if someone does this with that scheme, then they will get such-and-such. I understand that this may be difficult to say—this is not a domain with a great deal of maturity. But should I, for example, attack the MQV scheme, as instantiated in this standard, in some very specific sense, I'd like it to be clearer if I've broken the intent of the standard or not.

Validation of EC parameters and public keys is something that this document pays more attention to than I had anticipated. This is excellent, but one should warn there are real limits on the utility of parameter validation. Even if the documented checks in 3.1.1.2.1 and 3.1.2.2.1 are quite comprehensive for what we know now (and I assume that they are—you guys are the elliptic curve experts!), still, I doubt that this list will be comprehensive five or ten years from now. The value of parameter validation is pretty unclear.

2 Detailed Comments

1. p.5. Table 1. Concrete, efficient, but plenty of different fields to choose; very good.
2. p.9, Section 2.3. It appears that you are regarding the five types of objects very much as abstract data types. For example, in your view, I infer, an

octet string is *not* a bit string; the operations are different as, for example, the “extraction” operation, M_i , has different semantics if M is a bit string or if M is an octet string. I had to infer that thinking from the way that you were doing things.

If you’re going to regard types as abstract data types (which I suppose is indeed the right way to think of them), then maybe you should start out by saying what the types *are* (ie., what, abstractly, is the data, and what are the supported operations) *before* jumping in and describing your type conversions.

3. p.10, Section 2.3.1. I suppose $blen \geq 0$, but I doubt you ever use this for $blen = 0$. Anyway, how your algorithm behaves on input of the empty string doesn’t seem well-defined. Clearly the empty bit string should convert to the empty octet string, and conversely.
4. p.9, Section 2.3. You might try something like BitString, OctetString, EllipticCurvePoint, Integer, FieldPoint instead of your hyphenated versions. This would make the names of the conversion routines easier to parse, eg., BitString-to-OctetString would seem clearer than Bit-String-to-Octet-String.
5. p.9, Section 2.3. These conversions should probably be re-ordered so as to eliminate forward references—eg., it would be more logical to have Field Point to Octet String conversion proceed Elliptic Curve to Octet String conversion, which uses it.
6. p.11, line 2.3, sounds a bit awkward, how about: “If . . . then let $Y = 02_{16}$. If . . . then let $Y = 03_{16}$.”
7. p.12. Have you described an algorithm for performing steps 2.4.1 and 2.4.3?? An algorithm or reference would be good!
8. p.13. 2.3.6, “Input:”, typo, you have *log* in the wrong font.
9. p.14. I believe you mean to view M_i as an integer in range $[0, 255]$, not $[1, 256]$.
10. p.14. And I suppose there could be some question about how to view an octet as an integer in this set—insofar as the little-endian representation could conceivably be used. I found this spelled out in the appendix, under “octet”, but I doubt anyone would know to look there.
11. p.16, line 3, “such that taking logarithms of the associated elliptic curve requires about 2^t operations.” If you could really justify this statement you’d have the most impressive result in theoretical computer science of the century! Better replace “requires” with “would seem to require”, or something like that.

12. p.16, and then throughout the document. You frequently speak of the need for a party “to receive *an* assurance” (that domain parameters are valid, or that a public key is valid, or that a public key is partially valid.) At first, “to receive an assurance” didn’t sound like very good English to me, because assurance isn’t a “thing” that takes an article. You can receive a certificate, or a promise, a letter of assurance—but can you receive “an assurance”?

Maybe saying that you receive “assurance” or “some assurance” would read better.

(But now I’ve read “receive an assurance” so many times that it starts to sound just fine!)

13. p.16, and then throughout the document. I’ve heard people say that it’s bad style to start a sentence with a mathematical symbol (and it does indeed look ugly). You have lots of sentences like:

1. U does something.
2. V does something.
- V is trusted with respect to something.

You might wish to say “Party U ” and “Party V ” instead.

14. Continuing my focus on minutia: you might wish to hyphenate the following, to more clearly show what is most closely bound to what: *hash-function construct*, *key-deployment procedure*, *key-derivation function*, *public-key partial validation*, *small-subgroup attack*, *tag-checking operation*, and probably more that I’ve forgotten. (You can find a note on hyphenation of such noun clusters on my web page).
15. pp. 16–17, pp. 21–22. The lists of how U might get some assurance about the validity of the parameters might seem more logically ordered as 2, 4, 1, 3.
16. p.17. I am left wondering, at the end of 3.1.1.2.1, how much I’ve really bought from doing this test. Maybe the important thing to test for isn’t known, and will be published tomorrow. The same comment applies at the end of 3.1.2.2.1.
17. p.21. 3.2.2., and all over the place, really. Add a comma after “that is”.
Actually, there are lots of other missing commas after introductory phrases that are normally terminated with a comma. These include: *Equivalently*, *Furthermore*, *Loosely speaking*, and *Similarly*.
18. p.24, 3.3, and p.25, 3.4: Why do you say “simultaneously”? There need be nothing simultaneous about when U and V execute the primitive.

19. p.28, 3.6.1. This is more a comment for X9.63-KDF than your own effort (please pass on this comment if you attend those meetings), but it seems to me undesirable for a general KDF that you can easily get a K_i value for one shared key Z which you know to be identical to a K_j value for some different shared key Z' . Conceptually, we are trying to imitate a random oracle, and this is a simple sense in which we do not. Recall that

$$K_i = \text{Hash}(Z \parallel \text{Counter} \parallel [\text{SharedInfo}]) .$$

So, for example, the K_1 value generated with $Z = 00_{16}$ and $\text{SharedInfo} = 0000_{16}$ is identical to the the K_{16} value you get with $Z' = 0000_{16}$ and $\text{SharedInfo}' = 00_{16}$. There are simple ways to fix this.

20. p.30, 3.7.1, step 1. The parenthetical comment is somewhat misleading, insofar as the only supported MACs requires no “initial values.” And if a MAC did require such initial values they might as well be regarded as part of the key, no? Can’t you kill the notion of an “initial value” associated to a MAC?
21. p.32, lines 3–4. You might re-word this sentence—no need to perpetuate the myth that all an encryption scheme should do is hide the (entire) plaintext. *Roughly speaking, symmetric encryption schemes are designed so that it is hard for an adversary to recover information about messages (other than their length) from their ciphertexts. Thus they provide data confidentiality.*
22. p.32, lines 3–4, and again repeated, for parallel structure. This sentence is a run-on; it should be split up.
23. p.32. Your DES-CBC IV of 00000000_{16} has half the number of nibbles it is supposed to.
24. p.32. Usually it is *not* acceptable to used a fixed IV of 0000000000000000_{16} , since this reveals partial information about the plaintexts. It is fine in your setting since keys are used to encrypt only one message, resulting in a very weak security requirement being required for the underlying encryption scheme. Thus the paragraph beginning “3-key TDES in CBC mode is designed” is *not* true with respect to the scheme that you have specified (with zero IV)—it does *not* achieve semantic security under a chosen-plaintext attack. Instead, the comment you make about the XOR scheme applies to both encryption schemes.
25. p.32. “Chosen-message attack” should be “chosen-plaintext attack” according to the descriptions in your glossary. (But, actually, I have never heard this distinction, and call both of these “chosen-plaintext attack.”)
26. p.32, paragraph beginning with “The XOR encryption scheme.” To preserve parallel structure, you might re-word this sentence to say *The XOR scheme is commonly used either with truly random keys, when it is known*

as a “one time pad”, or with pseudorandom keys, when it is known as a “Vernam cipher.”

27. p.32. Not knowing what is done for padding in X9.52, I still do not know how padding is intended to work in 3-key TDES: zero-pad, if necessary, to a 64-bit boundary; or $10 \cdots 0_2$ pad to the next 64-bit boundary; or it is the caller’s business to ensure that the primitive is called with a message that is a multiple of 64 bits. You should probably clarify this, to make the document more self contained.
28. p.33, first sentence, same comment as earlier about the parenthetical statement — there are no initial values in the permitted encryption scheme, and if an encryption scheme needed an initial value I would want to consider it as part of the key.
29. p.35, 39. Run-on sentences beginning “Loosely speaking”—should be reworded.
30. p.39. “or to encrypt information data directly”. Information data is not a standard phrase. How about replacing this clause with: “or to directly encrypt user data.” (The phrase “information data” occurs one more time in this paragraph, and could again be replaced by “user data”.)
31. p.39. Does it make sense for the A in ECAES to stand for “augmented,” when here there is no scheme you are augmenting?! Actually, my recollection was that the A in DHAES stood for “authenticated” [11]. (Or perhaps “augmented” comes from P1363 pre-history, where, if I remember, there was once a DHES as well as a DHAES, so there it would have made sense to be augmenting something.)
32. p.39, second sentence of 5.1. While our first paper that described DHAES claimed plaintext awareness (reference [11]), Mihir and I subsequently retracted that claim (reference [1]), and directly proved in [1] that DHAES achieves indistinguishability under a chosen-ciphertext attack (under appropriate assumptions). Here is the background.

Reference [11], a conference paper, contained the DHAES scheme (and one other scheme) but it contained no proofs. When we were doing the full version of the paper, we ran into technical difficulties trying to prove that our scheme possessed the strong-form of plaintext awareness that we desired. We could prove a weak form of plaintext awareness, but not the desired definition. (The weaker definition formalizes that for any ciphertext C that the adversary produces, she knows that the underlying plaintext is one of $L = \{M_1, \dots, M_t\}$, for some small set L that she can enumerate. In contrast, in the desired definition, this list is just a single plaintext—the adversary knows exactly what is the plaintext of a ciphertext she produces.) So we had to back off of trying to prove plaintext awareness.

This difficulty seems inherent to any DH-based scheme.

We also decided that we had rather mis-positioned plaintext awareness, making it seem more like a “goal” than a tool, when, in fact, the real goal was usually something like non-malleability or indistinguishability under a chosen-ciphertext attack. Indeed we knew of no application in which PA is an actual goal in itself. Rather, showing PA was typically just a strategy for showing NM/IND under CCA. (Namely, demonstrate PA and IND-CPA and you have shown IND-CCA and NM-CCA, in the RO model, as proven in the CRYPTO '98 paper of Bellare, Desai, Pointcheval, and me. Here CCA refers to the strong form of CCA, dubbed CCA2 in [BDPR].) But since it seemed that DH-based schemes could not be proven to be plaintext-aware, plaintext awareness was just not a useful tool in this domain. Instead, one needs to directly argue goals like IND-CCA.

The result is that you should strike all mention of PA in your spec, and refer specifically to the goals of IND or NM under CCA. In 5.1, sentence 2, change “It is designed to be both semantically secure and plaintext aware, in the presence of an adversary capable of launching chosen-plaintext and chosen-ciphertext attacks” to something like “It is designed to be semantically secure in the presence of an adversary who can perform a chosen-ciphertext attack.”

(Note: I would say that a chosen-plaintext attack needn't be mentioned in the context of a PK encryption scheme, since it is *always* assumed to be possible.)

You might wish to mention somewhere that semantic security (or indistinguishability) under a CCA (meaning CCA2) implies non-malleability, a notion due to Dolev, Dwork and Naor. This was proven in [BDPR], mentioned above.

Reference [11] should have a pointer to see [1].

33. p.45, Steps 4 and 5. You're worrying about checking the validity of the public key when there is nothing in the protocol which mandates checking that the purported public key is the public key of the entity who claims it?! The one seems more basic than the other.
34. p.44. You might add “unilateral implicit key authentication” (or “unilateral authentication”) to the glossary. And you might add “mutual implicit key authentication” (or “mutual authentication”) to the glossary.
35. p.46. What you are calling the “Key Agreement Operation” is the non-interactive portion of key agreement in which each party defines the key he shall use based on flows from the protocol already received. This stage isn't really “key agreement”—something more like “key definition.”
36. p.46, Section 6.1. I am left, after reading this section, with no real understanding of what I should put in *SharedInfo* or what else I need to be doing in order to get what security properties. This is a little too open-ended.

37. p.47, Section 6.2.2, item 7. “partially valid” should be “at least partially valid”?
38. p.48, Section 6.2, same comment, that this section does not make clear what should be done to achieve what security properties. And the glossary didn’t really help.
39. p.49, Glossary — Terms.
 - (a) certificate. The definition has a logical problem. How about: “The public key and identity of an entity, together with some other information, rendered unforgeable by signing this information with the secret key of a Certification Authority.”
 - (b) chosen-message attack. “Similarly” → “Or, similarly,”
 - (c) elliptic curve. Comma after “two parameters”
 - (d) forward secrecy. Typo, add *of* before some.
 - (e) hash function (cryptographic hash function). The first clause following the colon is commonly said, but pretty close to meaningless short of interpreting “pre-specified” as “a randomly given,” which is not, I believe, what people intend. The second clause also has “foundational” problems. I’d suggest: “The function has properties such as: nobody can exhibit a pair of distinct inputs which map to the same output.”
 - (f) key agreement scheme. I have never heard the distinction you are making between a key agreement scheme and a key establishment scheme. I don’t know that who contributes to the key is particularly important (though I’ve heard others speak of it as though it is).
 - (g) key establishment scheme. How about: “. . . which reveals only to the legitimate users keying data suitable for subsequent cryptographic purposes.”
 - (h) keying data. How about changing to the singular, “Data suitable for use as a cryptographic key.”
 - (i) It is mildly strange that “cryptographic scheme” is listed under “cryptographic” while “cryptographic primitive” is listed under “primitive”.
 - (j) public-key cryptographic scheme. Missing the article “an”.
 - (k) secret key. Missing the article “a”.
 - (l) x -coordinate, y -coordinate, eliminate commas.
40. p.55, Appendix A.3, Notation.
 - (a) Typo: $[5.3] = 6$. I don’t think so!
 - (b) Typographical oddity in line 2 of page 56.
 - (c) $\gcd(x, y)$ might say that x and y are positive integers (what’s $\gcd(0, 0)$?)

- (d) Some of the alphabetization is odd, eg. having $\|X\|$ under X but $[X]$ at the beginning.
41. p.59, last line -4. Both *of* these weak classes of curves . . .
 42. p.60, first paragraph. How about “are described, for example, in [47,58,62].” and “like precomputation and switching to projective co-ordinates.”
 43. p.61. This table is potentially quite “biased”, and rather debatable. Some disclaimer?
 44. p.61. Also, its first column is redundant, and should probably be eliminated.
 45. p.61. Backwards opening quote on ‘complex multiplication’ [54]. Actually, there were quite a few more backwards opening quotes, too, like ‘implementation attacks’
 46. p.62, line 5, grammar, “user themselves’
 47. p.63. “, for example” can be eliminated from line one. Paragraph 6 ends with “first.)” instead of “first).”
 48. p.63, Section B.2.4. Since the MQV primitive is one of the more mysterious things in this document, I was hoping for a real discussion about it. What properties is this supposed to have? What has been show about it?
 49. p.64, B.3.1, paragraph 2, line 2: It’s → Its
 50. p.64. one-wayness is not in the glossary
 51. p.65, Section B.4.1. Replace the first two sentences with: “The ECAES is a public-key encryption scheme based on ECC. It is designed to be both semantically secure and non-malleable in the presence of an adversary capable of launching a chosen-ciphertext attack.”
 52. p.66, first paragraph. It is not (the equivalent of) R that was fed to the key derivation function, but the equivalent of the ephemeral private key, that is, (the bit representation of) k .
 53. p.66. Anyway, I do buy that it’s OK to omit k from the scope of the hash function. This isn’t in [1]. We will add it to the next version of [1].
 54. p.67, last paragraph. I don’t think the characterization that HMAC-SHA-1-160 is more secure than HMAC-SHA-1-80 is valid. While a random tag is obviously less likely to work for the former than the latter, the adversary’s ability to see the entire output of the SHA-1 means that more information is being given away when the adversary launches her attack in the former case. Personally, I’d say you’re better off dropping some bits. Anyway, the security of these primitives is, from what we know, incomparable.

55. p.67, last line –1. You say that the public key R could be included for alignment with [1]. But this should have been the octet string version of the secret key k . As such, it is *not* information shared between U and V , and it therefore could *not* have been put inside $SharedInfo_1$.
56. p.67, last line. Putting a counter value into $SharedInfo_1$ or $SharedInfo_2$ would not seem to be relevant for mitigating against replay attacks. You are making a new MAC key and encryption key with every encryption, so replayed messages are encrypted completely differently each time. The only way that I can think of replayed messages being a point of concern would be for an implementation failure so severe that the process which makes the ephemeral public key / private key pair got “stuck” at some fixed value.
57. p.68. Spelling error — “Abdullah” should be “Abdalla”.
58. p.68, third bullet. As I indicated before, I wanted more. The adversary could be passive or active. Party U might (due to out-of-band means, like a certificate) or might not know if Q_V is bound to entity V ; likewise, party V might or might not know if Q_U is bound to entity U . Party U might (due to out-of-band means) or might not know if Q_V is “fresh”; and analogously for party V . Party U might (or might not) be using his own, freshly minted, ephemeral key; or he may be using his long lived key. Given some choices, what, do you believe, is achieved? This is the sort of information that the security architect needs to understand.
59. p.69, third bullet. You say that the scheme is particularly vulnerable to invalid public keys. If you’re using an ephemeral key, that’s probably not true.
60. p.69. But certainly the “basic” scheme, where nothing is checked, is highly vulnerable to “bogus” public keys—public keys which don’t belong to their purported originator.
61. p.69. The scheme, in some versions, is vulnerable to replay attacks. But there are so many potential “versions” / “trust models” it is hard to be concrete.
62. p.69, Section B.5.2. Most of the comments just made about the DH scheme apply, (all the more so) to the MQV scheme.
63. p.74, Appendix C, ASN.1. You have earned my sympathy and respect, Simon.
64. p.83. Reference [1], spelling error, it’s Abdalla.
65. References. I noticed inconsistencies in formats, or typos, in references [24], [25], [51], [53], [61], [68], [78], [82]. Some of these are just the inclusion of a middle initial. On [79]: I’ve never seen “EuroCrypt”—Eurocrypt or EUROCRYPT.