Evaluation of Security Level of Cryptography

HDEF-ECDH

Dan Boneh **Phillip Rogaway**

January 16, 2001

Contents

This document analyzes HDEF-ECDH (Higher Degree Extension Field — Elliptic Curve Diffie-Hellman Key-Agrement), as described in documents [1] and [2].

1 Summary

Scheme. The heart of the proposal is an interesting technique for generating elliptic curves appropriate for cryptographic applications. None of the known attacks on the Elliptic Curve Discrete Log Problem (ECDLP) apply to these curves. Consequently, these curves can be used for all of: confidentiality, authentication, signatures, and key sharing. The curves generated by HDEF-ECDH are defined over $\mathbb{F}_p$ for a prime p and their trace is 3 (i.e. the curves contain $p-2$ points). The authors suggest choosing primes p so that both p and $p-2$ are prime.

As an application for their curves the authors chose to describe a key sharing technique. This key sharing technique is a direct implementation of the Diffie-Hellman protocol. The key sharing application seems to be added on as an afterthought and its description is incomplete.

It is best to view the HDEF-ECDH proposal as a technique for generating ECC domain parameters rather than a specific cryptographic technique.

Performance claims. The motivation behind HDEF-ECDH is that it enables users to quickly generate elliptic curves appropriate for cryptographic applications. The cost of generating a curve is roughly equivalent to finding two 160-bit twin primes. Although this is a non-trivial task, it can be done in a few minutes on a modern processor.

Once the curve $E/\mathbb{F}_p$ is generated one can use it for confidentiality, authentication, signatures, and key sharing. The performance of the these systems depends on the time it takes to compute a multiplication on the curve. More precisely, given a point $P \in E/\mathbb{F}_p$ efficiency depends on the time to compute xP for some integer x on the order of p. Since HDEF-ECDH curves are defined over $\mathbb{F}_p$ one cannot take advantage of standard acceleration techniques for curves defined over fields of characteristic 2. For example, one cannot use the popular Koblitz φ-expansion technique [3]. Hence, HDEF-ECDH cryptosystems are slower than systems using curves over $\mathbb{F}_{2^m}$.

Overall, the performance of systems based on HDEF-ECDH curves is comparable to the performance of systems based on the $\mathbb{F}_p$ curves defined in SEC2. The performance of HDEF-ECDH is likely to be worse than some of the $\mathbb{F}_{2^m}$ curves in SEC2.

Security claims. The curves generated by the HDEF-ECDH have $p-2$ points

over $\mathbb{F}_p$. Consequently, the discrete log problem on these curves is not susceptible to any of the known attacks: (1) the SSSA attack only applies to curves with p points. (2) the MOV and FR attacks only apply when the number of point r divides $p^k - 1$ for some small k. The authors show correctly that the smallest possible k is $k = \log_2 p$, which defeats both the MOV and FR attacks. The authors recommend that the prime p be a minimum of 160 bits. This is sufficient for defeating generic discrete log algorithms such as Pollard's Rho method. Overall, these curves are well designed to avoid all known attacks.

There are two concerns regarding this family of curves. First, all curves in the HDEF-ECDH family are designed to have trace 3. Current attacks show that curves with trace 0,1,2 and insecure. None of the current attacks apply to curves of trace 3. Nevertheless, restricting the set of curves to such a narrow family introduces some risk. It is possible that future attacks will specifically target this family of curves.

The second concern is that the HDEF-ECDH curves are a very small subset of the curves with trace 3. This follows from the fact that all HDEF-ECDH curves have CM fields with low discriminant. For example, the authors suggest using curves with discriminant -403. Non of the current attacks can exploit this, but future attacks might target this specific set.

Standards generally try to avoid picking curves from a narrow family, unless there is a compelling reason. Indeed, standards frequently pick curves over $\mathbb{F}_p$ by picking random curves and ensuring that these curves have a large prime order sub-group. Random curves over $\mathbb{F}_p$ have a CM field with a large discriminant (on the order of p).

Critiques. The proposal has the following weaknesses.

1. The proposal focuses on generating elliptic curve domain parameters rather than building cryptosystems. Domain parameters for ECC based cryptography have already been established in SEC2. The proposal does not make it immediately clear what is the advantage of the HDEF-ECDH method over using the pre-established curves in SEC-2. After all, the curves in SEC2 were also constructed so as to avoid all known attacks. The authors may wish to explain in more detail why the HDEF-ECDH method is superior to the curves already provided in SEC2.

2. The key exchange mechanism described at the end of the proposal is an immediate application of the Diffie-Hellman protocol. There is very little discussion of authentication, forward secrecy, and other desirable properties of a key exchange algorithm. The authors may wish to remove this section and present HDEF-ECDH as an alternative way for generating ECC domain parameters. HDEF-ECDH curves can be used for many cryptographic applications beyond basic key exchange.

2 Detailed Evaluation

We now provide a full evaluation of the performance and security of the proposed system.

2.1 Misclassification

HDEF-ECDH is currently classified as an Assymetric key sharing technique. This is inaccurate and ignores many of the applications for HDEF-ECDH. It is better to view HDEF-ECDH as a technique for generating domain parameters for ECC. The HDEF-ECDH curves can be used as follows:

1. Combining HDEF-ECDH with the ECMQV key-exchange algorithm leads to a key exchange mechanism with the same security properties as claimed for ECMQV [4].

2. Combining HDEF-ECDH curves with ECDSA leads to a signature scheme.

3. Combining HDEF-ECDH curves with elliptic curve ElGamal encryption leads to a semantically secure public key encryption scheme. One can further use Cramer-Shoup over HDEF-ECDH curves to obtain a chosen-ciphertext-secure system.

In the current proposal the authors view HDEF-ECDH curves as useful only for key exchange. They describe a key-exchange protocol which is directly based on Diffie-Hellman key exchange. There is no security analysis given for the key-exchange protocol, and its description appears to be orthogonal to the main purpose of the proposal.

We refer the reader to our report on ECDHS (from SEC 1) for some analysis of Diffie-Hellman authenticated key exchange.

2.2 Security

The authors claim that the ECDLP problem over the HDEF-ECDH curves is sufficiently hard so as to provide adequate security for cryptosystems using these curves. The authors correctly argue as follows:

- The SSSA attack only applies to curves over $\mathbb{F}_p$ whose trace is 1. Since the HDEF-ECDH curves never have trace 1 this attack does not apply.

- Let $E/\mathbb{F}_p$ be a curve with n points. The MOV and FR attacks reduce the ECDLP problem on $E/\mathbb{F}_p$ to the problem of computing discrete log in $\mathbb{F}_{p^k}^*$ where n divides $p^k - 1$. Hence, if the smallest k satisfying $n|p^k - 1$ is large, e.g. $k > \log p$, then these attacks are ineffective. The authors show that for curves of size $n = p - 2$ we must have $k > \log_2 p$ which proves that both attacks do not apply to the HDEF-ECDH curves.

- In light of the above two points, the only known way to solve the ECDLP on HDEF-ECDH curves is by using generic discrete log algorithms such as Pollard's Rho method. Since the authors choose p so that $n = p - 2$ is prime, it follows that Pollard's algorithm will run in time $O(\sqrt{p})$. Hence, when p is more than 160-bits long the ECDLP on HDEF-ECDH curves is sufficiently hard for cryptographic applications.

Several other techniques can generate curves that resist the three attacks described above. The most common is to pick a random curve, count the number of points on the curve, and then verify that the curve defeats the attacks above. One repeatedly picks new random curves until an adequate curve is found. Counting points on a random curve can be avoided by using one of the following methods: (1) use the CM method to generate curves with a known number of points, or (2) lift random curves over a small extension. Either way, generating curves for use in cryptography requires repeatedly picking curves until an adequate curve is found

The interesting fact about the HDEF-ECDH method is that there is no need to repeat the generation process multiple times. HDEF-ECDH only generates curves that defeat the attacks above. We note that HDEF-ECDH curve selection cannot be done in real time since it involves finding two 160-bit twin primes. This could take several minutes on a modern processor. Nevertheless, with HDEF-ECDH it is possible for each person to be using a different curve. This is somewhat different from the current SEC1 standard where all parties use one of a number of preselected curves. It is not immediately clear why allowing parties to select curves for themselves is preferable to using curves from a predefined list. The authors may wish to explain this further in the proposal.

The family of HDEF-ECDH curves. All HDEF-ECDH curves have trace 3. Furthermore, the discriminant of their CM field is small. Both these conditions impose extra structure on the resulting elliptic curve. Although none of the attacks today can exploit this structure, future attacks might specifically target this family. We note that it is unlikely that an attack can target curves of trace 3 specifically. However, it is quite possible that an attack could target curves whose CM field has a small discriminant.

The example of Algorithm 3. In Algorithm 3 the authors show how to construct curves of trace 3 whose CM field discriminant is -403. The authors choose this example since the Hilbert polynomial in this case is quadratic, and can be easily solved. The authors could have just as easily used a larger discriminant, -427, which would also lead to a quadratic Hilbert polynomial. Larger discriminants appear to be better to use from a security standpoint.

Key exchange protocol The last section of the cryptographic techniques specifications describes a key exchange mechanism. The proposed protocol is a direct implementation of the Diffie-Hellman protocol. No security analysis is offered. The authors might want to remove this section and present HDEF-ECDH for what it is: an algorithm for generating ECC domain parameters. However, if

the authors insist on including a key-exchange protocol in the proposal, then they should base their protocol on established key-exchange protocols, such as SKEME [5] or MQV [4].

2.3 Parameter choice

The authors suggest using prime modulus p that are at least 160 bits long. With our current state of knowledge, the ECDLP problem defined over the HDEF-ECDH curves is sufficiently hard so as to provide adequate levels of security. The security provided by a curve who order is 160 bits long is considered to be comparable to the security provided by 1024-bit RSA.

In Section 3.1.3 of the Cryptographic Techniques specification document the authors suggest picking a prime p such that $p = 2^u - c$ for some small c. However, for HDEF-ECDH the prime must lie in the sequence $p = d\ell^2 + d\ell + \frac{d+9}{4}$ for some small value of d (the authors suggest using $d = 403$). When choosing a 160-bit prime, the value ℓ is on the order of 2^{80}. It is not clear how to pick a prime in the desired sequence so that the prime is close to a power of 2. The authors may wish to explain this in more detail.

2.4 Editorial comments

Both the specification of HDEF-ECDH and the self evaluation report are written clearly. Below are a few comments the might help in improving the writeup.

- In Algorithm 3 the authors may wish to further explain why Step 4 is needed. Step 4 twists the curve resulting in a curve with $p + 4$ points on it. Why is this necessary?
- Neither Algorithm 1 nor Algorithm 3 explain what to do when the Hilbert polynomial does not have a root in $\mathbb{F}_p$. It seems that one would have to choose a new p and start over.
- Section 3.1.3. Explain how to generate primes $p = 2^u - c$ for small c when the prime p is of the form $p = d\ell^2 + d\ell + \frac{d+9}{4}$.
- The Self Evaluation Report is very vague in Section 2.2.2. Especially at the end where the authors give a brief explanation of the security of the key-exchange protocol. It is better to use existing key-exchange protocols and refer to the analysis given in those papers [5, 4].
- In Section 3 the authors only give running times for the key exchange protocol. There is no running time given for curve generation. The proposal would be much strengthened if the authors state how long it takes to generate HDEF-ECDH curves in practice.

References

[1] No authors stated, but proposal from JAIST and Technology Matsushita Electric Industrial Co. Cryptographic Techniques Specifications (for HDEF-ECDH). Undated CRYPTREC submission, 1999.

[2] No authors stated, but proposal from JAIST and Technology Matsushita Electric Industrial Co. Self Evaluation Report (for HDEF-ECDH). Undated CRYPTREC submission, 1999.

[3] N. Koblitz. CM-curves with good cryptographic properties, Advances in Cryptology - Crypto '91, Springer-Verlag, 1992, 279-287.

[4] L. Law, M. Qu, A. Menezes, J. Solinas and S. Vanstone, An efficient protocol for authenticated key agreement, Technical report CORR 98-05, Department of C&O, University of Waterloo.

[5] H. Krawczyk, SKEME: A Versatile Secure Key Exchange Mechanism for Internet. In Proceedings of the Internet Society Symposium on Network and Distributed System Security, February 1996.