NASA/CR—2005-213578

An Interoperability Consideration in Selecting Domain Parameters for Elliptic Curve Cryptography

Wesley M. Eddy
RS Information Systems, Inc., Cleveland, Ohio

March 2005

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at 301–621–0134

- Telephone the NASA Access Help Desk at 301–621–0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076

NASA/CR—2005-213578

An Interoperability Consideration in Selecting Domain Parameters for Elliptic Curve Cryptography

Wesley M. Eddy
RS Information Systems, Inc., Cleveland, Ohio

Prepared under Contract NAS3–03100

National Aeronautics and
Space Administration

Glenn Research Center

March 2005

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100

Available electronically at http://gltrs.grc.nasa.gov

An Interoperability Consideration in Selecting Domain Parameters for Elliptic Curve Cryptography

Wesley M. Eddy
RS Information Systems, Inc.
Cleveland, Ohio 44135

Abstract– Elliptic curve cryptography (ECC) will be an important technology for electronic privacy and authentication in the near future. There are many published specifications for elliptic curve cryptosystems, most of which contain detailed descriptions of the process for the selection of domain parameters. Selecting strong domain parameters ensures that the cryptosystem is robust to attacks. Due to a limitation in several published algorithms for doubling points on elliptic curves, some ECC implementations may produce incorrect, inconsistent, and incompatible results if domain parameters are not carefully chosen under a criterion that we describe. Few documents specify the addition or doubling of points in such a manner as to avoid this problematic situation. The safety criterion we present is not listed in any ECC specification we are aware of, although several other guidelines for domain selection are discussed in the literature. We provide a simple example of how a set of domain parameters not meeting this criterion can produce catastrophic results, and outline a simple means of testing curve parameters for interoperable safety over doubling.

1 Introduction

Elliptic curve cryptography (ECC) appears to be emerging as a technology of choice for key transport and agreement in cryptosystems. ECC can be used to implement public key sytems that offer advantages in both smaller key length and faster computation time over traditional public key systems like RSA and DSA. ECC techniques can also be used to exchange shared secret keys for traditional symmetric cryptographic algorithms like AES or triple-DES. Government and financial industries in the United States have begun the slow process of adopting ECC in their systems.

The major advantage that ECC has over RSA and DSA is that the lengths of ECC keys scale linearly with those of AES keys of equivalent strength. For example the security of 256 bit AES keys is equivalent to that of 512 bit ECC keys, while it takes RSA keys of 15,360 bits to rival this strength. Keys of this length take up a considerably disproportionate amount of storage space and capacity for transmission in comparison to the security level they provide. Operations using keys of that length are also terribly expensive for smaller mobile and wireless hosts of limited power and computational resources.

The basic building blocks of ECC systems are points on curves whose coordinates are members of a number-theoretic field. The two fields typically employed are F_p, and F_{2^m}. Elements of F_p are integers from 0 to some prime p and the field operations of addition and multiplication are performed modulo p. Elements of F_{2^m} are binary strings of length m and operations are performed bitwise across the field elements.

Several standards bodies have produced exact descriptions of the fields F_p and F_{2^m}, along with techniques for working with curves over points whose coordinates are field members (for example ANSI [X9.62, X9.63], IEEE [P1363], NIST [FIPS 186-2], and SECG [SEC 1]). These base standards are implemented in commercial products and used in communications protocols (by the IETF, for example [GBWM$^+$04]) to provide cryptographic functions.

We describe a glaring ommision from many ECC-explanatory documents and some of the standards, that affects the selection of domain parameters for ECC. Many of the algorithms presented for doubling points are not able to function for certain points on the axes and thus implementations may either fail or produce incorrect results if those points are members of a selected domain and are encountered in some calculation. We present one example of parameters that can generate an error condition, and show how it results in a failed Diffie-Hellman key exchange. In addition to revising some of the standards to properly specify the doubling operation, we make the recommendation that those selecting curves for use in practice make explicit efforts to avoid using domain parameters that can create possible error situations.

2 Problem Description

Equation 1 is used to specify the points on an elliptic curve over field F_p with domain parameters a and b chosen from F_p, where each point's coordinates, x and y, are also elements of F_p.

$$y^2 = x^3 + ax + b \tag{1}$$

Geometrically, addition of two points, (x_1, y_1) and (x_2, y_2), on such a curve is accomplished by drawing a line through both points. This line will intersect the curve at some other point with coordinates (x_3, y_3). The additive inverse of this point, $(x_3, -y_3)$, is the desired sum. To double a point on a curve (add it to itself), the same proceedure is performed, except since it is not possible to define a line with a single point, instead a line tangent to the curve at the point to be doubled is used. With curves over F_p and F_{2^m} where coordinates are discrete field elements and the shape of a curve is not easily visualized, a numeric method is employed for adding and doubling points. The rules for adding and doubling points are clearly specified in many of the ECC standards.

For points (x_1, y_1) and (x_2, y_2) on a curve over F_p, their sum, (x_3, y_3) is given by equation 2, where all operations are done modulo p.

$$(x_3, y_3) = (\lambda^2 - x_1 - x_2, \lambda(x_1 - x_3) - y_1) \tag{2}$$

For two distinct points, λ simply represents the slope between them and is given by equation 3, while when doubling a point ($x_1 = x_2$ and $y_1 = y_2$), λ represents the slope of a tangent to the curve at that point and is given by equation 4. Using these equations to compute λ, fractions are evaluated using multiplication modulo p of of the numerator by the multiplicative inverse of the denomator in F_p.

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \tag{3}$$

$$\lambda = \frac{3x_1^2 + a}{2y_1} \tag{4}$$

These definitions of λ are always accompanied in the literature by two restrictions. Since equation 1 only contains the term y^2 in the y-coordinate, no more than two points exist with the same x-coordinate. Thus, the first restriction is that if adding two distinct points such that $x_1 = x_2$, then $y_1 \equiv -y_2 (\text{mod}\, p)$, so the points are additive inverses and their sum is $(0, 0)$. This intuitively makes since, as a line between the points would lie perpendicular to the x-axis and never be able to intersect another point on the curve. In this case, the normal addition rules are short-circuited and the point $(0, 0)$, which is, by definition, present on every curve, is returned. The second restriction is that for doubling a point, the y-coordinate must not be 0. In many stadards, no algorithm is given for doubling points that lie on the x-axis. At least one standard [P1363] specifies that the double of such a point yeilds $(0, 0)$, however several other documents ignore this situation, and present no information on how it is to be handled.

The situation is similar for elliptic curves over F_{2^m}. Such curves are defined by the parameters m, a, and b, where m is the length in bits of each field element, and a and b set the curve's shape according to equation 5.

$$y^2 + xy = x^3 + ax^2 + b \tag{5}$$

In the case of such curves over F_{2^m}, doubling a point (x_1, y_1) into the point (x_3, y_3) is accomplished via equation 6 with λ given by equation 7. Field addition is done using exclusive-or on the bit strings and field multiplication yielding an m bit string whose value depends on the basis used[1].

$$\begin{aligned} (x_3, y_3) &= (\lambda^2 + \lambda + a, x_1^2 + (\lambda + 1)x_3) && (6) \\ \lambda &= x_1 + \frac{y_1}{x_1} && (7) \end{aligned}$$

In the case of equation 7, to avoid a denominator of 0, the point may not lie on the y-axis, or $x_1 \neq 0^m$. In some documents, the algorithm's output is unspecified if this condition is not met, although at least one [P1363] properly returns $(0, 0)$ in such cases.

Both curves in F_p and those in F_{2^m} may contain points for which the doubling operation is unspecified. This is evident, yet not addressed in some of the ECC standards. One of the standards [X9.63] even uses an example curve containing such a point. The curve over F_p with $p = 23$, $a = 1$, and $b = 1$ contains the point (4, 0). This is clearly on the curve as $0^2 = 0 \equiv 0(\text{mod}23)$, and $4^3 + 1(4) + 1 = 69 \equiv 0(\text{mod}23)$, yet the standard does not specify how it is to be doubled.

With the doubling operation inconsistently specified, implementations may not always perform in either correct or compatible ways; in fact we have seen software that does this incorrectly. Leaving point doubling unspecified for certain points on elliptic curves would pose no problem if point doubling was not an integral part of ECC systems. Unfortunately, this is not the case. Multiplication of points by scalars is a crucial operation needed for elliptic curve Diffie-Hellman (ECDH) key exchanges. ECDH key exchanges are used to establish shared secret keys for encryption and authentication between two parties over an insecure channel. The mechanism is explained in section 3. The problem is that all of the common algorithms for finding scalar multiples of a point involve doubling it *and* doubling points which are generated as intermediate values during the computation of the final answer. This means that the problem may not be hedged by simply avoiding the use of certain forbidden points, as these points may be generated as intermediate results during a computation. Instead, we should prevent the use of curves containing such points, which is only slightly more difficult.

3 Example Failed Key Exchange

ECDH is an adaptation of the traditional Diffie-Hellman key exchange using points on elliptic curves as the insecurely transmitted information. Each party generates a random scalar, k, and agrees on a set of elliptic curve domain parameters along with an initial point, P, on the chosen curve. Each party then uses P and its value of k to generate a new point $Q = kP$ using scalar multiplication. These values of Q are public keys, while the k values are private keys. The Q values are exchanged and each side multiples its received Q by its k to obtain the shared secret key $S = kQ$.

For example, users i and j agree to use some curve with point P. They select k_i and k_j respectively and generate $Q_i = k_iP$ and $Q_j = k_jP$, then transmit Q_i and Q_j over non-secured public channels. User i computes $S = k_iQ_j = k_ik_jP$ and user j computes $S = k_jk_iP$. Scalar multiplication being commutitive,

[1] Whether the basis is polynomial or normal is irrelevent for our purposes, the formula for doubling a point is the same.

these values of the shared secret key are identical. Both users may then safely use S in symmetric key encryption algorithms, for example.

The ECDH algorithm relies heavily on scalar multiplication of points. There are several algorithms of varying efficiency for computing this function. The simplest is merely to add P upon itself $k-1$ times. For $k > 1$ however, this will involve doubling P as the initial step[2]. Many other more efficient (and commonly implemented) methods also rely on doubling, including the binary method, the addition-subtraction method, the window method, and Lim and Lee's Comb method (all these methods are well described by Lopez and Dahab [LD00]). More recent and even faster methods also rely on repeated doublings, and clearly do not avoid the problem we have described, if doubling certain points is undefined. All of the listed scalar multiplication algorithms simply assume a well-defined doubling algorithm. As we have shown, this is simply not the case for at least certain points on some curves.

None of the standards documents that fail to include full doubling algorithms, specify amongst the requirements for acceptable curves that they do not contain any points that would violate their algorithms' constraints on doubling. Thus naive implementers may allow such curves to be used, and if a point whose doubling is undefined is generated as an intermediate result in a scalar multiplication, bad things may happen. If implementers are not ECC experts and do not realize $(0, 0)$ is the proper result of some doublings, they may consider the result incomputable and throw an exception or return an error value. This could render an ECDH exchange a failure. In an implementation that ignores the restriction and attempts to double such points with an incomplete algorithm, the result can be a point not on the curve. This will clearly be troublesome and either lead to an error condition if checking is performed, or (as we shall demonstrate) inconsistent shared secret keys may be generated, making symmetric encryption and decryption operations unable to decode encoded data. These are both problematic situations, which could have been avoided in the specifications.

As an example of how doubling of an intermediate point not on the curve can lead to inconsistent keys, we present an example using the previously discussed curve over F_p, with $p = 23$, $a = 1$, and $b = 1$. The point $(4, 0)$ will be generated as an intermediate result of the addition-subtraction method of scalar multiplication, and a point not on the curve will be generated as the final result due to the addition function attempting to compute the double of a point on the x-axis without the proper algorithm.

Take the point $P = (19, 18)$, which by equation 1 can be verified as on the example curve. Multiply this point by the scalar $k = 427$. Following the addition-subtraction method, we first convert 427 into its non-adjacent form (NAF), and initialize a return value Q to (0, 0). The chosen scalar multiplication algorithm then iterates through the values in the NAF form of k. For each entry in the NAF, Q is doubled, and if the entry is a 1, then the initial point $(19, 18)$ is also added to Q, while if the entry is a -1, then the additive inverse of the initial point $(19, -18) = (19, 5)$ is added to Q.

We take the NAF of 427, $u = [-1, 0, -1, 0, -1, 0, -1, 0, 0, 1]$, and list out the intermediate values of $Q = kP$ with respect to each NAF entry Subscripts referring to the elements of the NAF start with 0 as the rightmost element of u, increasing to the left. Our point addition algorithm is that listed by Lopez and Dahab [LD00], which (incorrectly) specifies no precondition that points not lie on the x-axis, and does not check for this condition as part of the algorithm, and does not properly return $(0, 0)$ in such cases.

$$\begin{array}{rcl} \text{initialize} & \rightarrow & Q = (0,0) \\ u[0] = 1 & \rightarrow & Q = (19,18) \\ u[1] = 0 & \rightarrow & Q = (12,19) \\ u[2] = 0 & \rightarrow & Q = (5,19) \end{array}$$

[2]Using $k \in \{0, 1\}$ makes no sense as the result will be $(0, 0)$ or P, either of which an eavesdropper could identify from the domain parameters and deduce k.

$$\begin{aligned} u[3] = -1 &\rightarrow Q = (11, 3) \\ u[4] = 0 &\rightarrow Q = (4, 0) \end{aligned}$$

Up to this point in the execution, there have not been any problems with doubling Q nor performing the subsequent possible addition of P or $-P$. All of the generated points have been on the curve. However, the latest value of Q generated is $(4, 0)$, which (while on the curve) lies on the x-axis. The next step of the algorithm, since there are more elements of the NAF, is to double Q. Since neither the addition nor multiplication algorithms we are using here make checks on Q before following the flawed doubling proceedure, it is interesting to see the result.

$$\begin{aligned} u[5] = -1 &\rightarrow Q = (15, 0) \\ u[6] = 0 &\rightarrow Q = (16, 0) \\ u[7] = -1 &\rightarrow Q = (14, 0) \\ u[8] = 0 &\rightarrow Q = (18, 0) \\ u[9] = -1 &\rightarrow Q = (21, 22) \end{aligned}$$

Doubling $(4, 0)$ and adding $-P$ to it gives $(15, 0)$, which from equation 1 does not lie on the curve specified. Using this value of Q and finishing through the elements of the NAF of 427, the final result returned by the scalar multiplication is $(21, 22)$, which is also clearly not on the curve, as it does not satisfy equation 1.

To see why this is a problem, consider the case where two parties decide to use an ECDH key exchange to establish a shared secret key for a session. They agree to use the example curve that we have discussed, and the point $P = (19, 18)$. One of the parties draws $k = 427$ as its random bits. The point it generates as $Q = kP$ will not lie on the selected curve. An error occurs in the exchange whether the generating party checks Q and decides it is invalid, or transmits it and lets the other party notice that it is invalid. The results are worse yet if the other party receives the bogus Q and generates a session key from it. This key will be unusable as it is generated from a point that is not on the curve and the session keys generated by each party will not agree, preventing them from communicating.

4 Avoidance

Since on-axis points (which are by some definitions undoublable) may be generated as intermediate results of scalar multiplication, avoiding the doubling problem is more difficult than simply not using such points. In an ECDH key exchange, if such a point is encountered and must be doubled during a scalar multiplication, it is also insufficient to merely start over using another multiplication algorithm, another k, or another set of curve parameters. Using another algorithm is a poor attempt at a solution, as even if the problem is avoided locally, there is generally no way to ensure the other party will not run into it. Using another value of k is also a bad idea, as the curve obviously contains troublesome points which may be encountered again, and generating more random bits for another k may be relatively expensive on some systems. Curve domain paramenters will have typically been agreed upon ahead of time and renegotiating them is likely to be a bad idea, as schemes that allow this will be open to many forms of attack.

The most foolproof solution is to simply not allow curve domain parameters that specify curves with disallowed points under certain poorly-specified doubling operations. This involves checking the domain parameters in the curve generating function over the elements of the underlying field, F_p or F_{2^m}. For instance, we could clearly and easily predict problems with F_p, where $p = 23$, $a = 1$, and $b = 1$, by solving for x in equation 1 with $y = 0$, in the range $0 \leq x < 23$, and seeing that $x = 4$ satisies the relation. In general, verifying that no disallowed doublings can occur may be accomplished via two means. The brute force approach is to iterate through the members of the underlying fi eld, testing that no members allow the restricted coordinate to be zero.

A more elegant approach would be to solve for the roots of the point generating functions (equations 1 and 5) with the restricted variable set to 0 and the chosen domain parameters. If none of the roots are members of the selected fi eld, then the set of elliptic curve domain parameters is safe to use, otherwise it is not. For curves over F_p, the roots of equation 8 are used and for curves over F_{2^m}, equation 9 is used.

$$x^3 + ax + b = 0 \tag{8}$$

$$y^2 = b \tag{9}$$

Equation 8 can be easily solved using the domain parameters a and b in the general proceedure for solving cubic equations. Since at least two of the obtained roots are complex, if the other does not lie in the prime fi eld selected, then the domain parameters are safe. Equation 9 is derived from equation 5 by setting $x = 0$. Obtaining a square root function for the appropriate basis in the underlying fi eld is all else needed for F_{2^m} curves.

As an example, take the curve over F_p specifi ed by $p = 23$, $a = 1$, $b = 4$, which only differs from the curve we have already seen to be poor by the b parameter. We easily see that there are no zeros of x^3+x+4 in F_{23}, and thus that the curve generated by those parameters is safe for use. This test would have easily caught the example curve we showed to be a bad choice with $b = 1$, as we would fi nd a non-complex root of $x^3 + x + 1$ in F_{23} at $x = 4$ using the cubic formula.

5 Conclusion

We have described a problem with the current specifi cations for elliptic curve cryptography in that they do not disallow the use of curves containing points on the axes which may not be doublable using certain ECC specifi cations. We have shown that the accidental use of such curves may lead to failed key exchanges, and be confusing to debug. We have outlined a simple means of verifying that a given set of domain parameters is safe with regards to this condition for curves over both F_p and F_{2^m}. Adding this verifi cation check to the process for selecting curves for cryptography will allow use of the current point addition and doubling algorithms and scalar multiplication algorithms (and running code) without modifi cation, and prevent doubling-related problems from arising. Restricting the usable curve domain parameters in this way does not influence the degree of security provided by ECC techniques. Ideally, we would fi x all insuffi cient standards documents to fully specify doubling, however we have no effective way of making previously written code compliant, and so must select curves so as to avoid possible problems.

References

[SEC 1] Standards for Efficient Cryptography Group. SEC 1: Elliptic Curve Cryptography, September 2000.

[GBWM$^+$04] V. Gupta, S. Blake-Wilson, B. Moeller, C. Hawk, and N. Bolyard. ECC Cipher Suites for TLS, January 2004. Internet Draft (work in progress).

[P1363] IEEE. Standard Specifications for Public-Key Cryptography, November 1999. P1363 (working draft).

[X9.62] American National Standards Institute. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), September 1998. X9.62 (working draft).

[X9.63] American National Standards Institute. Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography, January 1999. X9.63 (working draft).

[LD00] J. Lopez and R. Dahab. An Overview of Elliptic Curve Cryptography, May 2000.

[FIPS 186-2] U.S. Department of Commerce National Institute of Standards and Technology. Digital Signature Standard (DSS), January 2000. FIPS 186-2.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (*Leave blank*)	**2. REPORT DATE** March 2005	**3. REPORT TYPE AND DATES COVERED** Final Contractor Report

4. TITLE AND SUBTITLE An Interoperability Consideration in Selecting Domain Parameters for Elliptic Curve Cryptography	**5. FUNDING NUMBERS** WBS–22–184–10–07 NAS3–03100
6. AUTHOR(S) Wesley M. Eddy	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) RS Information Systems, Inc. 21000 Brookpark Road Cleveland, Ohio 44135	**8. PERFORMING ORGANIZATION REPORT NUMBER** E–15045
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546–0001	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** NASA CR—2005-213578

11. SUPPLEMENTARY NOTES

Wesley M. Eddy, RS Information Systems, Inc., 21000 Brookpark Road, Cleveland Ohio, 44135, e-mail Wesley.M.Eddy@grc.nasa.gov, 216–433–6682. Project Manager, Will Ivancic, Satellite Networks and Architectures Branch, NASA Glenn Research Center, organization code RCN, 216–433–3494.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category: 62 Available electronically at http://gltrs.grc.nasa.gov This publication is available from the NASA Center for AeroSpace Information, 301–621–0390.	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(Maximum 200 words)***

Elliptic curve cryptography (ECC) will be an important technology for electronic privacy and authentication in the near future. There are many published specifications for elliptic curve cryptosystems, most of which contain detailed descriptions of the process for the selection of domain parameters. Selecting strong domain parameters ensures that the cryptosystem is robust to attacks. Due to a limitation in several published algorithms for doubling points on elliptic curves, some ECC implementations may produce incorrect, inconsistent, and incompatible results if domain parameters are not carefully chosen under a criterion that we describe. Few documents specify the addition or doubling of points in such a manner as to avoid this problematic situation. The safety criterion we present is not listed in any ECC specification we are aware of, although several other guidelines for domain selection are discussed in the literature. We provide a simple example of how a set of domain parameters not meeting this criterion can produce catastrophic results, and outline a simple means of testing curve parameters for interoperable safety over doubling.

14. SUBJECT TERMS Cryptography	**15. NUMBER OF PAGES** 13
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT**

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102