Designing and Detecting Trapdoors for Discrete Log Cryptosystems

Daniel M. Gordon[⋆]

Department of Computer Science, University of Georgia, Athens, GA 30602

Abstract. Using a number field sieve, discrete logarithms modulo primes of special forms can be found faster than standard primes. This has raised concerns about trapdoors in discrete log cryptosystems, such as the Digital Signature Standard. This paper discusses the practical impact of these trapdoors, and how to avoid them.

1 Introduction

The National Institute of Standards and Technology (NIST) recently announced a proposal for a federal digital signature standard, DSS [21]. This proposal gives an algorithm for electronically signing documents, to guarantee the integrity of the message and the identity of the sender. The Digital Signature Algorithm (DSA) given in the proposal is based on the difficulty of solving discrete logarithms modulo large primes. It has already excited a great deal of discussion regarding its efficiency and security.

In the DSA, the public key consists of a prime p of 512 bits, a prime q dividing $p-1$ of 160 bits, and a number g which is a $((p-1)/q)$th power mod p. The private key is a number x, and $y = g^x$ is also made public. Then to sign a message m, the sender calculates

$$r \equiv (g^k \bmod p) \bmod q$$

and

$$s \equiv (k^{-1}(H(m) + xr) \bmod q.$$

Here H is any one-way hash function, m is the message, and k is a random number less than q. To authenticate a message, a recipient computes:

$$w = s^{-1} \bmod q,$$

$$u_1 = (H(m)w) \bmod q,$$

$$u_2 = (rw) \bmod q,$$

[⋆] Current address: Center for Communications Research, 4320 Westerra Court, San Diego, CA 92121

$$v = (g^{u_1} y^{u_2} \bmod p) \bmod q.$$

The signature is correct if $v = r$.

The only known way to break this system is to find x from g and y (i.e. find the discrete logarithm $\log_g y \bmod p$). Several other schemes (e.g. [3], [6], [18]) also depend on the difficulty of discrete logarithms. Subexponential algorithms are known for finding discrete logarithms modulo large primes, but the largest prime for which the problem has been solved is 224 bits in length, by LaMacchia and Odlyzko [9], using the Gaussian integer method of Coppersmith, Odlyzko and Schroppel [5].

In [7], an algorithm is given for finding discrete logarithms using a number field sieve, which is asymptotically faster than other known methods. The general number field sieve is impractical, but a variant of the algorithm for primes of special forms is practical. The idea of using the number field sieve to make trapdoor primes is mentioned in [1], page 50.

In Sect. 2, we give a brief description of how the special number field sieve for discrete logarithms works. Estimates for the time to break the DSS with regular versus various trapdoor primes are given in Sect. 3. The rest of the paper deals with how to detect trapdoors, how to construct trapdoors to avoid detection, and how one or more people can choose primes for which the probability of a trapdoor existing is negligible.

2 The Number Field Sieve

Here we give a short presentation of the special number field sieve. For a more complete description of the algorithm, and the heuristic assumptions involved, see [7].

Let p be a prime and f be an irreducible monic polynomial of degree k with reasonably small coefficients, such that for some integers X and Y near $p^{1/k}$ we have $Y^k f(X/Y) \equiv 0 \pmod{p}$. Let $\alpha \in \mathbb{C}$ denote a root of f, and $K = \mathbb{Q}(\alpha)$. For constructing trapdoor primes, it is convenient to pick f so that $\mathcal{O}_K = \mathbb{Z}[\alpha]$ is a unique factorization domain.

We may define a homomorphism φ from $\mathbb{Z}[\alpha]$ to $\mathbb{Z}/p\mathbb{Z}$ by sending α to $X/Y \bmod p$, so that for any integers c and d,

$$cY + dX = Y(c + dX/Y) \equiv Y\varphi(c + d\alpha) \pmod{p}.$$

The factor base $\mathcal{B}$ will consist of rational primes less than a bound B ($\mathcal{B}_{\mathbb{Q}}$), first-degree primes in $\mathcal{O}_K$ with norm less than B ($\mathcal{B}_K$), a fundamental set of units in $\mathcal{O}_K$, and Y. Calculating the primes and units for the field is not difficult when f is, say, $x^5 - 2$ (see [14]), but will be more difficult for polynomials with larger coefficients. We will discuss this problem in the next section.

Call a rational or algebraic integer *smooth* if its prime factors are all in the factor base. We will need to find many pairs of coprime integers c, d such that $cY + dX$ and $c + d\alpha$ are both smooth. This can be accomplished efficiently by sieving $cY + dX$ and the norm

$$|N(c + d\alpha)| = |(-d)^k f(-c/d)|$$

for fixed c and large range of d. The smoothness of $c + d\alpha$ and $N(c + d\alpha)$ are related by the following (see [7], Proposition 2):

Theorem 1. *If c and d are relatively prime and $r^l \parallel N(c + d\alpha)$ for a prime r, then $(r, \alpha - c_r)^l \parallel (c + d\alpha)$ in $\mathcal{O}_K$, for $c_r \equiv -c/d \pmod r$.*

We will choose g, the base for the discrete logarithm to be smooth and a primitive root modulo p. Note that this cannot be the same as the base g for the DSA, since that g is a $(p-1)/q$th power. Thus, the first step in breaking the DSS would be to find the log of its base.

The precomputation step involves sieving through small c and d, looking for pairs with $cY + dX$ and $N(c + d\alpha)$ both smooth. Each hit gives us an equation involving logarithms of the factor base. Suppose that we find a c and d for which both are smooth, say

$$cY + dX = \prod_{s \in \mathcal{B}_{\mathbb{Q}}} s^{w_s(c,d)},$$

and

$$|N(c + d\alpha)| = \prod_{s \in \mathcal{B}_{\mathbb{Q}}} s^{v_s(c,d)},$$

for $v_s, w_s \in \mathbb{Z}_{\geq 0}$. Then

$$(c + d\alpha) = \prod_{s \in \mathcal{B}_K} s^{v_s(c,d)}$$

by Theorem 1. Since $\mathcal{O}_K$ is a UFD, this equation involving ideals can be replaced with one involving algebraic integers, by replacing each s in the above equation by a generator for the ideal. Then $c + d\alpha$ divided by the generators is a unit, which can be explicitly computed in terms of a fundamental set of units, using Theorem 5 of [7].

From this, we obtain:

$$(cY + dX)(Y\ \varphi(c + d\alpha))^{-1} \equiv \prod_{s \in \mathcal{B}} s^{u_s(c,d)} \equiv 1 \pmod p,$$

which gives us an equation for the logs of the factor base:

$$\sum_{s \in \mathcal{B}} u_s(c,d) \log_g s \equiv 0 \pmod{p-1}.$$

Once we have more than $|\mathcal{B}|$ hits, we solve the resulting matrix equation over $\mathbb{Z}/(p-1)\mathbb{Z}$ using structured Gaussian elimination to reduce the size of the matrix, and then solving a smaller, dense matrix using the conjugate gradient method or Wiedemann's algorithm (see [10]). This completes the precomputation.

To find an individual logarithm, we reduce the problem to finding the logs of medium-sized primes. Choose random values of s and attempt to factor $g^s y \pmod p$ using the elliptic curve method (ECM) until one is found for which

$$g^s y \equiv q_1 q_2 \cdots q_r \pmod p,$$

with each q_i less than a bound Q. (This can be improved as in [9], by finding $z_1, z_2 = O\left(\sqrt{p}\right)$ such that $g^s y \equiv z_1/z_2 \pmod p$, and testing whether z_1 and z_2 are both Q-smooth.)

For each q_i, we will sieve c and d for which $q_i|(cY + dX)$, say fixing d and taking $c = c_0 + eq_i$, to find one value for which $(cY + dX)/q_i$ and $N(c + d\alpha)$ are both smooth. Once this happens we are done, since from the precomputation we know the logs of the whole factor base.

The choices for the size of the factor base and q_i's depend on how time is to be divided between the two stages. Enlarging the factor base reduces the time needed to find individual logarithms, but at the cost of increasing the precomputation time. Let

$$L_n[v; c] = \exp\{(c + o(1))(\log n)^v(\log\log n)^{1-v}\},$$

for $n \to \infty$. Assuming some reasonable heuristics (see [7]), the optimal choice of parameters is

$$k = \left\lceil 10^{1/5}\left(\frac{\log p}{\log\log p}\right)^{1/5}\right\rceil,$$

$$B = L_p[2/5; (4/125)^{1/5}],$$

and

$$Q = L_p[3/5; (1/100)^{1/5}],$$

which results in both the precomputation and individual logarithms taking expected time

$$L_p[2/5; \left(\frac{128}{125}\right)^{1/5}] \approx L_p[2/5; 1.00475].$$

If many instances are to be done for one p, more time could be spent on the precomputation by taking a larger factor base. For $\mu \geq (128/125)^{1/5}$, if we spend $L_p[2/5; \mu]$ time on the precomputation, each logarithm can be found in time

$$L_p\left[2/5; \left(\frac{128}{125\mu^2}\right)^{1/3}\right].$$

The Gaussian integer method is a special number field sieve with $k = 2$ and K a complex quadratic field. For any $c \geq 1$, the Gaussian integer method can find logarithms in time $L_p[1/2; 1/(2c)]$ if $L_p[1/2; c]$ is spent on the precomputation. Even for fairly small primes with good polynomials, the special number field sieve is faster than the Gaussian integer method.

For primes which cannot be represented by good polynomials, a similar procedure called the general number field sieve can be done. The difference is that the polynomial f will have large coefficients, so operations in the resulting field will be impractical. To avoid them, the equations must be solved over the rationals instead of modulo $p - 1$, to eliminate ideals and units.

The better asymptotic time for the general number field sieve comes from using different fields for finding individual logarithms. Instead of sieving through

c and d such that $q_i|(cY+dX)$, we search through polynomials for which $q_i|\varphi(\alpha)$. This allows us to take Q as big as X and Y, which asymptotically speeds up the algorithm. The time for the general number field sieve is

$$L_p[1/3; 3^{2/3}] \approx L_p[1/3; 2.08].$$

Oliver Schirokauer [17] has developed a method to avoid solving equations over the rationals, so that the time can be improved to $L_p[1/3, 1.902]$. The larger constant and $o(1)$ terms make the general number field sieve impractical for numbers we are interested in.

3 Complexity Estimates

There are four parts of the algorithm which dominate the timing estimates. For the precomputation, there is the sieve to gather equations, and then the linear algebra modulo $p-1$ to solve the equations. For finding individual logarithms, the medium-sized primes are found by repeated trials of the ECM, and then another sieve must be done for each q_i.

How much time is devoted to each part depends on the choice of parameters: the degree k of the polynomial f, the polynomial chosen (and the resulting field), the size of the factor base, and the size of a medium-sized prime.

For $k = 2$, we can take $f = x^2 + r$, for r a small positive integer for which $-r$ is a quadratic residue modulo p. Then the resulting field is just a complex quadratic field $\mathbb{Q}(\sqrt{-r})$, and we have the Gaussian integer method. This can be applied to any prime, but is impractical for 512-bit primes. Breaking the DSS using the Gaussian integer method using $B = 50,000,000$ would require sieving 10^{20} numbers. Even if this could be accomplished, the resulting matrix would have over $5,000,000$ columns, and the linear algebra problem would be a major hurdle.

The numbers in Table 1 show the difficulty of finding discrete logarithms for 512-bit primes using the special number field sieve with polynomials of degree 2–5 with small coefficients. They assume that the large prime variation described in [14] is being used. They are intended as rough estimates only, but serve to give an idea of the time required. For comparison, the factorization of F_9 required sieving about 10^{14} numbers, and solving a matrix with 199,203 columns modulo 2 [14]. For larger k, X and Y are smaller, so a smaller factor base can be used, speeding the precomputation. But then for individual logarithms $N(c+d\alpha) \approx Q^k$ is larger, so we need to take Q smaller and do more ECM trials.

Table 1 indicates that the ideal polynomial for a trapdoor would have degree four. Its coefficients should be small, to keep down the size of $N(c + d\alpha)$. The field generated by a root of the polynomial should have small discriminant and regulator, class number one and index one, so that field operations can be done efficiently. If the polynomial has four complex roots, then the unit group will have rank one.

For example, the polynomial x^4+x+1 satisfies all the above conditions. The problem is that the polynomial could only be used with primes p for which there

Table 1. Statistics for 512-bit primes with good polynomials.

k	2	3	4	5
B	5×10^7	5×10^6	3×10^6	2×10^6
sieve range	10^{20}	2×10^{16}	2×10^{14}	10^{14}
matrix size	$5,600,000$	$650,000$	$400,000$	$280,000$
Q	10^{20}	10^{19}	10^{15}	10^{13}
# ECM trials	$29,000$	$78,000$	2×10^7	2×10^9
second sieve	2.5×10^{14}	2.4×10^{15}	1.5×10^{14}	3×10^{14}

exist $X, Y \approx p^{1/4}$ such that $X^4 + XY^3 + Y^4 \equiv 0 \pmod{p}$. This is a thin set of primes, which can easily be detected (see the next section).

For polynomials with larger coefficients, the special number field sieve is more complicated. The sieving stage takes slightly longer, since the norms being tested for smoothness are larger. For polynomials with coefficients of, say, up to 100 in absolute value, the sieving range must be increased by roughly a factor of ten.

Another difficulty is dealing with a field of larger discriminant. The problem is finding generators for the unit group and prime ideals in the factor base. In [13], these are found by searching through algebraic integers of the form $\sum_{i=0}^{4} h_i \alpha^i$, for α a generator of K and small values of h_i. For fields generated by polynomials with larger coefficients, this will be impractical.

There have been several papers on efficient algorithms to find units and algebraic integers of given norms in general number fields, (see [4], [16]). The computations are involved, but they only need to be done once for a given f.

The matrix equation resulting from the sieving may be solved using intelligent Gaussian elimination to greatly reduce the size of the matrix, and then the conjugate gradient algorithm to solve the reduced equation. In [10] these methods were used to solve matrices with up to 96,321 columns.

4 Trapdoor Primes and Polynomials

From Table 1, we see that some 512-bit primes may not be safe, but general ones (at least for the moment) are. We want to ensure that for a given prime p there is a no polynomial f which can be used for the special number field sieve. Currently, the only way to check for this is to check one polynomial at a time.

Let p be a 512-bit prime and f be a polynomial of degree $k \geq 3$. We will say that X and Y are a trapdoor for p and f if they are both less than (say) $1,000\, p^{1/k}$ in absolute value, and $Y^k f(X/Y) \equiv 0 \pmod{p}$.

Theorem 2. *If a trapdoor X, Y exists for p and f, then for a root c_p of f mod p, (X, Y) is a short vector in the lattice $\mathcal{L} = \langle (p, 0), (c_p, 1) \rangle$.*

Proof. Let c_p be the root of f mod p congruent to X/Y mod p. The lattice $\mathcal{L}$ contains (X, Y), since $(c_p, 1)Y = (c_p Y, Y) \equiv (X, Y) \pmod{p}$.

The shortest vector in $\mathcal{L}$ has length at most $O(\sqrt{p})$, and for most choices of p, f and c_p, the short vector will be $\Theta(\sqrt{p})$. If such an X and Y do exist, (X, Y) has length $< \sqrt{2}\ 1,000\ p^{1/k}$.

Conversely, all vectors $(X, Y) \in \mathcal{L}$ satisfy $Y^k f(X/Y) \equiv 0 \pmod{p}$, so any such short vector is a trapdoor for f and p. □

This gives an efficient algorithm for testing whether a trapdoor exists for a given f and p. One may find linear factors of f mod p efficiently by eliminating square factors (dividing by the greatest common divisor of f and f' mod p), and then taking the gcd of $(x^p - x)$ and f mod p (see [8]). Then X and Y, if they exist, can be found using lattice reduction.

The main problem with this is that every polynomial f needs to be considered separately, so a limited range of polynomials can be searched. On a Sparcstation 1, one fourth-degree polynomial can be tested for a 512-bit prime in about a minute. On a parallel machine many polynomials could be searched at once, and a fairly large range of polynomials could be tested. With this test, a trapdoor with a good polynomial could be found. This forces an adversary to choose a polynomial from a set too large to be exhaustively searched, say an f of degree fourth with coefficients chosen randomly between -100 and 100.

Note that in the special case $Y = 1$, where $p = f(X)$, the polynomial can be found much faster. In this case $(p/a_k)^{1/k}$ is close to X, where a_k is the leading coefficient of f. All polynomials with a given a_k could be tested at once, very efficiently, so such a trapdoor would be much easier to discover.

Similar techniques can be used to construct a trapdoor prime. Suppose we wish to compute q and p for the DSA such that for a given polynomial $f(x) = x^3 + bx^2 + cx + d$ and some $X, Y \approx p^{1/3}$, $p = Y^3 f(X/Y)$. Begin by finding a 160-bit prime q, and choosing any $Y_0 \approx 2^{170}$. Let $g(x) = Y_0^3 f(x/Y_0)$. Then we may find an a mod q such that $g(a) \equiv 1 \pmod{q}$, by looking for linear factors of $g(x) - 1 \pmod{q}$. If none exists, then another Y_0 may be tried.

For any $X \equiv a \pmod{q}$ and $Y \equiv Y_0 \pmod{q}$, we have $Y^3 f(X/Y) \equiv 1 \pmod{q}$. Taking $X = a + l_1 q$ and $Y = Y_0 + l_2 q$, with l_1 and l_2 chosen so that $Y^3 f(X/Y) \approx 2^{512}$. we expect to soon find a pair for which $p = Y^3 f(X/Y)$ is prime. This p and q could be used as a trapdoor for the DSS.

This is not an ideal trapdoor, since from Table 1, a degree four polynomial would work better. The problem with constructing a better trapdoor using the above method is that a is usually a 160-bit number, which is bigger than $p^{1/4}$, so X would be too large. The revised DSS will allow primes p up to 1024 bits [19]. For primes with 640 or more bits, the above method can be used to make a trapdoor with a degree four polynomial. For primes with 800 or more bits, a degree five polynomial can be used.

Another way to generate a trapdoor would be to choose a polynomial f, and try random values of X and Y until $p = Y^k f(X/Y)$ is prime and divisible by a 160-bit prime q. To find such a value, one could sieve by small primes or use the ECM factoring method to find an X, Y pair for which $p - 1$ is smooth except for one 160-bit prime factor. This has the drawback that $(p-1)/q$ would be smooth, which while it is not known to weaken the system, does seem undesirable.

5 Protocols for Choosing a Prime

The ideal way to avoid worries about a trapdoor would be to come up with a way of generating primes for which one can guarantee that no such polynomial exists. An alternative is to use a random prime, which is almost certain to be safe. Call a prime p *unsafe* if an f exists with $Y^k f(X/Y) \equiv 0 \pmod{p}$, where k is between 3 and 10, X and Y are less than $1,000\, p^{1/k}$, and the absolute values of the coefficients of f are less than 500. Then the fraction of 512-bit primes which are unsafe is at most

$$\frac{1}{\pi(2^{512}) - \pi(2^{511})} \sum_{k=3}^{10} \left(1,000 \cdot 2^{512/k}\right)^2 \cdot 1000^{k+1} < 2^{-100}.$$

Suppose two people wish to agree on a safe key for the DSS. They can choose a random seed for the random number generator, using a protocol due to Blum [2]. From this they can use the method of Appendix 2 of [21] to create a key which is as likely to be safe as any random key.

On the other hand, a central authority might want to announce a key for general use, so that everyone is convinced there is no trapdoor. To do this, the authority must have a pseudo-random number generator and algorithm for constructing keys so that

1. Any user can verify that a key was generated using the approved method.
2. Keys produced by this method should be no more likely than random keys to contain trapdoors.
3. The choice of seed used for the random number generator should not allow the authority to create a particular key.

With a few modifications, the random number generator mentioned in Appendix 3 of [21] can be made to satisfy the above criteria. That method uses DES with a 64-bit seed, DES key and 64-bit date/time-stamp. To satisfy the above conditions, the DES key used should be fixed as part of the algorithm, the seed should be made public with the DSS key, and the time-stamp format should be specified.

It could be argued that the 64-bit seed gives too much freedom, putting the third condition at risk. This can be remedied by restricting the choice of seeds, or eliminating the seed entirely and just using the time-stamp.

For an example of a "trustable" key, consider:

$$q = 1147860701762054730346201299935827782113538756127$$

and

$$\begin{aligned} p = {} & 71561947643978020492787877919336180873773390583792476383 \\ & 44062581902861059517171507927020818420231820214082169894 \\ & 37333407873531412629727277892752481262741 1 \end{aligned}$$

These numbers were generated using the binary expansions of π and e. The prime q is the smallest prime greater than $\left[2^{158}\pi\right]$, and $(p-1)/q$ is the smallest number larger than $\left[2^{350}e\right]$ for which p is prime. There is no reason to suspect this number of being any more likely than a random number to have a trapdoor, and tests of p by many polynomials have not found any.

6 Conclusion

In this paper, we have tried to quantify the threat of trapdoors for discrete logarithm-based cryptosystems, in particular DSS. While trapdoors do give a definite advantage over standard keys, with a few easy precautions in the choosing of p and q it is possible to prevent them, and they do not seem to pose a major problem for such systems.

In [15], Maurer and Yacobi present a public key distribution system, based on computing discrete logarithms modulo a composite number n. The factorization of n is a trapdoor which allows a trusted authority to compute secret keys. Unlike the DSS, their system relies on the trapdoor, and they ask if a similar trapdoor can be made for primes. The special number field sieve does provide a trapdoor which could be used to construct a similar system with a prime modulus, but such a system would be impractical.

References

1. T. Beth, M. Frisch and G.J. Simmons, eds., *Public-key Cryptography, State of the Art and Future Directions*, LNCS #578, Springer, 1992.
2. M. Blum, Coin flipping by telephone: A protocol for solving impossible problems, (*Proceedings of the 24th IEEE Computer Conference*, 1982, pp. 133-137.
3. E.F. Brickell and K.S. McCurley, An Interactive Identification Scheme Based on Discrete Logarithms and Factoring, *Journal of Cryptology*, to appear.
4. J. Buchmann and A. Pethö, Computation of independent units in number fields by Dirichlet's method, *Math. Comp.*, **52** (1989), pp. 149-159.
5. D. Coppersmith, A.M. Odlyzko and R. Schroeppel, Discrete logarithms in GF(p), *Algorithmica*, **1** (1986), pp. 1-15.
6. W. Diffie and M.E. Hellman, New directions in cryptography, *IEEE Trans. Info. Theory*, **22** (1976), pp. 644-654.
7. D. Gordon, Discrete logarithms in $GF(p)$ using the number field sieve, *SIAM Journal on Discrete Math.*, to appear.
8. D.E. Knuth, *The Art of Computer Programming*, Vol. 2, *Seminumerical Algorithms*, Second Edition, Addison-Wesley, Massachusetts, 1981.
9. B. LaMacchia and A.M. Odlyzko, Computation of discrete logarithms in prime fields, *Designs, Codes and Cryptography*, **1** (1991), pp. 47-62.
10. B. LaMacchia and A.M. Odlyzko, Solving large sparse linear systems over finite fields, Advances in Cryptology: Proceedings of Crypto '90, (A. Menezes, S. Vanstone, eds.), *Lecture Notes in Computer Science*, Springer-Verlag, New York, 1991.
11. H.W. Lenstra, Jr., Factoring integers with elliptic curves, *Ann. of Math.*, **126** (1987), pp. 649-673.

12. A.K. Lenstra, H.W. Lenstra, Jr., and L. Lovász, Factoring polynomials with rational coefficients, *Math. Ann.* **261** (1982), pp. 515-534.
13. A.K. Lenstra, H.W. Lenstra, Jr., M.S. Manasse and J.M. Pollard, The number field sieve, *Proc. 22nd ACM Symposium on Theory of Computing* (1990) pp. 564-572.
14. A.K. Lenstra, H.W. Lenstra, Jr., M.S. Manasse and J.M. Pollard, The factorization of the ninth Fermat number, preprint, 1991.
15. U.M. Maurer and Y. Yacobi, A non-interactive public-key distribution system, Advances in Cryptology: Proceedings of Eurocrypt '91, (D.W. Davies, ed.), *Lecture Notes in Computer Science*, Springer-Verlag, New York, 1991, pp. 498-507.
16. M. Pohst and H. Zassenhaus, *Algorithmic Algebraic Number Theory*, Cambridge University Press, Cambridge, 1989.
17. O. Schirokauer, *On pro-finite groups and on discrete logarithms,* Ph.D. thesis, University of California, Berkeley, May 1992.
18. C.P. Schnorr, Efficient signature generation by smart cards, *Journal of Cryptology*, to appear.
19. M.E. Smid and D.K. Branstad, Response to comments on the NIST Proposed Digital Signature Standard, *Advances in Cryptology: Proceedings of Crypto '92*, to appear.
20. D.H. Wiedemann, Solving sparse linear equations over finite fields, *IEEE Trans. Info. Theory*, **32** (1986), pp. 54-62.
21. Specifications for a digital signature standard, National Institute for Standards and Technology, *Federal Information Processing Standard Publication XX*, draft, August 1991.

This article was processed using the LaTeX macro package with LLNCS style