LATTICE ATTACKS ON DIGITAL SIGNATURE SCHEMES

N.A. HOWGRAVE-GRAHAM AND N.P. SMART

Abstract. We describe a lattice attack on the Digital Signature Algorithm (DSA) when used to sign many messages, m_i, under the assumption that a proportion of the bits of each of the associated ephemeral keys, y_i, can be recovered by alternative techniques.

1. Introduction

Lattice attacks have recently been used to attack RSA schemes under various additional assumptions, such as low exponent versions of RSA, or factoring the modulus when a certain portion of the bits of p are known in advance. Many of these attacks have derived from ground breaking ideas of Coppersmith on how one can use the LLL algorithm [9] to solve univariate and bivariate modular polynomial equations. For more details on this and related matters the reader should consult, [2], [4], [5], [7] and [8].

ElGamal signatures, see [6], are based on the assumption that one has a finite abelian group, G, for which it is computationally infeasible to solve the discrete logarithm and Diffie-Hellman problems. ElGamal type signature schemes have been deployed and standardized in the Digital Signature Algorithm, DSA, and its elliptic curve variant, EC-DSA.

In the above mentioned protocols, based on the discrete logarithm problem, Alice publishes the group G, along with its cardinality $p = \#G$, which we assume to be a large prime of over 160 bits in length. Alice also publishes an element $g \in G$, and $h = g^x$ for some private integer x. In what follows f is any mapping from G to $\mathbb{Z}/p\mathbb{Z}$, that is almost bijective, and which is also assumed to be public knowledge.

For Alice to sign a message $m \in \mathbb{Z}/p\mathbb{Z}$ she computes b such that

$$m \equiv by - xf(g^y) \pmod{p}, \tag{1}$$

for some randomly chosen $y \in \{1, \ldots, p-1\}$, and sends Bob the triple (m, g^y, b). The integer y is usually referred to as the ephemeral key, since it needs to be different for each message and is only required for the short space of time it requires to sign the message.

Bob may verify that

$$g^{mb^{-1}} h^{f(g^y)b^{-1}} = g^y$$

without ever knowing the quantity y (clearly knowledge of y immediately leads to the discovery of x). There are various other signing/verifying equations that one could use, but they are all roughly of the same form and our attack will apply to any scheme which uses an auxiliary equation such as Equation (1).

In this paper we analyze the situation where Alice signs many messages, m_i, using her fixed private key x and the ephemeral keys y_i. The messages Alice will

Key words and phrases. digital signatures, lattices.

sign will not be chosen by the adversary. However, we do assume that a few of the bits of the random quantities y_i are also known. We do not address how these few bits of y_i are to be determined, it may be due to a weak random number generator, a timing attack or using some probe on the device used to generate the signatures.

Under the above assumption we show that the remaining bits of the y_i may be discovered in essentially polynomial time. However, we observe that when the number of known bits of each y_i is very small, the increasing size of the lattices we need to consider make the method increasingly impractical. Notice that if we manage to recover any one of the ephemeral keys then we recover the private key, x, and are so able to impersonate the valid user.

Our method resembles some of the techniques used in [3] in that it uses a polynomial time algorithm of Babai [1] to find a lattice vector which is close to a non-lattice vector.

2. Basic strategy

Assume we intercept h messages, then we have the following set of equations

$$m_i - b_i y_i + x f_i \equiv 0 \pmod{p}$$

for $1 \leq i \leq h$, where $f_i = f(g^{y_i})$ and only x and y_i are unknown.

Rearranging these equations we obtain equations of the form $y_i + C_i x + D_i \equiv 0 \pmod{p}$, for some integers C_i, D_i. If we know no information about any bits of x then we can eliminate x and obtain $h-1$ equations of the form $y_i + C_i' y_h + D_i' \equiv 0 \pmod{p}$ for some other integers C_i' and D_i'. On the other hand if we do know some information about x then we may as well use it.

In either case we obtain $n = h$ or $h - 1$ equations of the form

$$y_i + A_i y_0 + B_i \equiv 0 \pmod{p} \text{ for } i = 1, \ldots, n, \tag{2}$$

for some given integers $A_i, B_i \in [0, \ldots, p-1]$, where $y_0 = x$ or $y_0 = y_h$. It is on these equations that our attack will be mounted and not the discrete logarithm problem from which a single instance of the protocol derives its security.

Suppose that we do not know a certain set of (contiguous) bits of the y_i, for $i = 0, \ldots, n$. In other words, for $i = 0, \ldots, n$, we have

$$y_i = z_i' + 2^{\lambda_i} z_i + 2^{\mu_i} z_i''$$

where z_i', z_i'', λ_i and μ_i are known and the z_i are the only unknowns. Clearly in the above representation of y_i we are assuming

$$0 \leq z_i' < 2^{\lambda_i},\ 0 \leq z_i < X_i = 2^{\mu_i - \lambda_i},\ \lambda_i < \mu_i \text{ and } 0 \leq z_i''.$$

By rearranging the Equations (2) we obtain equations in the z_i given by

$$z_i + s_i z_0 + t_i \equiv 0 \pmod{p} \text{ for } i = 1, \ldots, n, \tag{3}$$

for some integers $s_i, t_i \in [0, \ldots, p-1]$.

A random set of equations of this form would have solutions with $z_i \approx p$. But our set is not a random set since we know there is a solution with $z_i < X_i < p$. In the examples we consider the size of the z_i could be as much as $p^{0.95}$, even so we know there is a solution which is smaller than one would expect from a random set of equations.

We have reduced our problem to finding a 'small' solution to a set of modular equations. Since one would expect such 'small' solutions to be rare, one can hope that any solution to the above congruences, which is suitably small in some sense, will be precisely the small solution we require.

To tackle this problem of finding a 'small' solution to the set of simultaneous linear equations we consider the lattice, L, generated by the rows of the following matrix:

$$A = \begin{pmatrix} -1 & s_1 & s_2 & \dots & s_n \\ 0 & p & 0 & \dots & 0 \\ 0 & 0 & p & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \dots & \dots & \dots & p \end{pmatrix} \in M_{(n+1),(n+1)}(\mathbb{Z}).$$

Hence $L = \{\mathbf{x}A : \mathbf{x} \in \mathbb{Z}^{n+1}\}$. Now consider the non-lattice vector given by

$$\mathbf{t} = (0, t_1, t_2, \dots, t_n) \in \mathbb{Z}^{n+1}.$$

By construction we know that there is a vector $\mathbf{x} \in \mathbb{Z}^{n+1}$ such that

$$\mathbf{x}A - \mathbf{t} = (z_0, z_1, \dots, z_n) \in \mathbb{Z}^{n+1}.$$

So there is a lattice vector, $\mathbf{x}A$, whose distance from the vector $\mathbf{t}$ is bounded by

$$\|\mathbf{x}A - \mathbf{t}\|^2 \leq \sum_{i=0}^{n} X_i^2$$

In [1], Babai gives a polynomial time algorithm to find a closest lattice vector to a given non-lattice point. Suppose we first transform A to an LLL-reduced basis represented by the rows of the matrix B. Let $\mathbf{b}_i^*$ denote the corresponding Gram-Schmidt basis derived from B in the usual way. Babai proves the following theorem:

Theorem 1 (Babai)**.** *In polynomial time, one can determine a lattice vector* $\mathbf{w}$ *which satisfies*

$$\|\mathbf{w} - \mathbf{t}\|^2 \leq c_1 \|\mathbf{b}_{n+1}^*\|^2,$$

for some constant c_1 *depending on* n*.*

Heuristically we believe the vector size of $\|\mathbf{b}_{n+1}^*\|$ to be slightly larger than $\Delta^{1/(n+1)}$, where Δ is the lattice determinant, say

$$\|\mathbf{b}_{n+1}^*\| \leq c_2 \Delta^{1/(n+1)}.$$

for some constant, $c_2 > 1$, depending on n. In our case we have $\Delta = p^n$, so if

$$\sum_{i=0}^{n} X_i^2 < c_1 c_2 \Delta^{2/(n+1)} = c_1 c_2 p^{2n/(n+1)}$$

then there is a good chance that Babai's algorithm will produce a lattice vector $\mathbf{w}$ such that

$$\mathbf{w} - \mathbf{t} = (z_0, z_1, \dots, z_n).$$

We are making the heuristic assumption that if Babai's algorithm finds a lattice vector which is close enough to $\mathbf{t}$, then it will be the vector which corresponds to a solution to our original problem. This heuristic seems to be born out in practice, and is common in lattice arguments. We know, after all, that there exists a lattice vector which is closer to the vector $\mathbf{t}$ than one would expect from a purely random lattice. Such close vectors should be rare, so if Babai's algorithm finds a close lattice vector then it should be the one we are after.

Notice that the above result of Babai is the theoretical bound derived from the definition of an LLL-reduced basis. It is well known that the LLL algorithm performs much better than one would expect from theory, so heuristically we hope that the constant c_1 in Babai's theorem should really be slightly larger than one and that the constant c_2 is at most n. Then, hopefully, the condition

$$\max_{0 \le i \le n} X_i < p^{n/(n+1)}$$

will be sufficient to derive the required solution to our problem.

Even if it does not, we may obtain a vector which is close enough such that the resulting ephemeral keys are indistinguishable from the correct ephemeral keys. This last case would allow us to claim, by revealing the (bogus) ephemeral keys, that it was us and not the legitimate party which signed the original messages.

On the other hand, Babai's algorithm may reveal a close vector which is not close enough for our purposes in that it may give rise to ephemeral keys which we can not use to pretend we sent the original messages. But as we argued above, if our heuristics hold, then such a situation should be very rare.

To see what all this means in practice we make the simplifying assumption that the same number of bits of the y_i are known, for all i. This is not necessary for the attack to work, but makes the following argument simpler, If the proportion of known bits is $\epsilon \in (0, \ldots, 1)$, then we have $X_i = p^{1-\epsilon}$. Our inequality then becomes

$$p^{(1-\epsilon)} < p^{n/(n+1)}.$$

Hence,

$$\epsilon > \frac{1}{n+1}.$$

So the more messages we use in our lattice attack then the smaller the number of known bits we need. However, the more messages we use, the larger the lattices and the more likely that our heuristic breaks down.

3. Experimental Results

It remains to consider whether the above heuristic simplifications are sensible and are born out in practice. We implemented the above attack using C++ and the NTL library, [10], to perform the LLL reduction. Since the DSA mandates 160 bit values of p, to agree with the output length of the SHA-1 function, we chose a prime p of 160 bits. We then generated sets of random equations such as those in Equation (3), such that the unknown values of z_i are bounded by $p^{1-\epsilon}$. Our heuristic would imply that we would require

$$n \approx 1/\epsilon$$

such equations to recover all the unknown variables.

The following table indicates the range of applicability of our heuristic and the resulting algorithm: The times are averaged over a series of runs, for a prime of 160 bits. The actual value of n is the value used which recovers the ephemeral keys, for the majority of the series of runs of the algorithm.

ϵ	$1/\epsilon$	Actual Value of n required	Time in Seconds
.500	2	2	0.0102
.250	4	4	0.0360
.100	10	11	0.4428
.050	20	30	8.6970
.025	40	–	Infeasible

The entry of 'Infeasible' means we could not find the keys with this value of ϵ with our implementation and the values of n we attempted. Notice that $\epsilon = .025$, for a prime of 160 bits, means that only four bits of each ephemeral keys are known to the attacker. As we can see our heuristic is more accurate when a higher proportion of the bits are known, and so a smaller number of equations are needed. However, when $\epsilon = 0.05$ we can mount a successful attack using very little computing resources, with only 8 bits known out of every 160 bits of ephemeral key and only 30 signed messages.

4. Non-contiguous blocks of bits

When the known bits of the ephemeral keys do not occur in one contiguous block, the lattice techniques still work with exactly the same theoretical bounds, although the time taken to find the remaining bits does increase. In this section we detail the necessary modifications to the original algorithm.

We assume there are d blocks of unknown bits in the private key x and ephemeral keys y_i, i.e.

$$x = x' + \sum_{j=1}^{d} x_j 2^{\lambda_j}, \quad \text{and} \quad y_i = y_i' + \sum_{j=1}^{d} y_{i,j} 2^{\lambda_{i,j}},$$

for some unknown positive integers $x_i, y_{i,j}$ such that

$$x_j < X_j < 2^{\lambda_{j+1}-\lambda_j} \text{ and } y_{i,j} < Y_{i,j} < 2^{\lambda_{i,j+1}-\lambda_{i,j}},$$

and for known integers x' and y_i'. We further restrict ourselves to the case when the number of unknown bits in x and the y_i is approximately the same, i.e. for all $1 \leq i \leq h$ we have the following:

$$\prod_{j=1}^{d} X_j \approx \prod_{j=1}^{d} Y_{i,j} \approx p^{1-\epsilon}.$$

Using the same transformations as in Section 2 we let $z_{i,j}$, for $i = 0, \dots, n$ and $j = 1, \dots, t$, denote our unknown quantities and write our system of equations as

$$z_{i,1} + \sum_{j=2}^{d} s_{i,j} z_{i,j} + \sum_{j=1}^{d} r_{i,j} z_{0,j} + t_i \equiv 0 \pmod{p} \text{ for } i = 1, \dots, n.$$

In terms of the unknowns $z_{i,j}$ we assume their respective bounds, $Z_{i,j} \in \mathbb{Z}$, satisfy

$$\prod_{j=1}^{d} Z_{i,j} \approx p^{1-\epsilon}$$

for each $i = 0, \dots, n$. Set $J_{i,j} = J/Z_{i,j} \in \mathbb{Z}$, for all i and j, where

$$J = \prod_{\substack{0 \leq i \leq n \\ 1 \leq j \leq d}} Z_{i,j} \ \simeq \ p^{(1-\epsilon)(n+1)}.$$

These quantities will be used to weight our lattice so as to take into account variations in the size of the $Z_{i,j}$. Let I_l denote the identity matrix of dimension l and consider the lattice, L, generated by the rows of the following matrix:

$$B = \left(\begin{array}{c|c} -I_{d(n+1)-n} & \begin{array}{c} R^t \\ S \end{array} \\ \hline 0 & -pI_n \end{array} \right) \times D,$$

where $R = (r_{i,j})$ and S denotes the matrix

$$S = \begin{pmatrix} \mathbf{s}_1 & & 0 \\ & \ddots & \\ 0 & & \mathbf{s}_n \end{pmatrix} \in M_{n(d-1),n}(\mathbb{Z}),$$

with $\mathbf{s}_i$ denoting the column vector given by $(s_{i,j})_{j=2}^{d}$. The matrix D is the diagonal matrix given by

$$\begin{aligned} D &= \operatorname{diag}(J_{0,1}, \ldots, J_{0,d}, J_{1,2}, \ldots, J_{1,d}, \ldots, J_{n,2}, \ldots, J_{n,d}, J_{1,1}, \ldots, J_{n,1}) \\ &= \operatorname{diag}(\mathbf{j}). \end{aligned}$$

If we consider the non-lattice vector given by

$$\mathbf{t} = (0, \ldots, 0, t_1 J_{1,1}, \ldots, t_n J_{n,1}),$$

then we know there exists a vector $\mathbf{x} \in \mathbb{Z}^{d(n+1)}$ such that

$$\mathbf{x}B - \mathbf{t} = (z_{0,1}, \ldots, z_{0,d}, z_{1,2}, \ldots, z_{1,d}, \ldots, z_{n,2}, \ldots, z_{n,d}, z_{1,1}, \ldots, z_{n,1}) \cdot \mathbf{j}.$$

By the choice of weights we have used, every entry in the right hand vector has size around J. We then use Babai's algorithm to find a lattice vector, $\mathbf{w}$, close to the vector $\mathbf{t}$. Hopefully we will obtain $\mathbf{w} = \mathbf{x}B$.

Since

$$\det(B) = p^n \prod_{\substack{0 \le i \le n \\ 1 \le j \le d}} J_{i,j} = p^n J^{d(n+1)} \prod_{\substack{0 \le i \le n \\ 1 \le j \le d}} Z_{i,j}^{-1} = p^n J^{d(n+1)-1},$$

in order to satisfy the criteria of Theorem 1 (under the same heuristic assumptions of Section 2) we wish the to ensure that

$$J < \det(B)^{1/d(n+1)}.$$

But $J \simeq p^{(1-\epsilon)(n+1)}$, hence we obtain

$$(1-\epsilon)(n+1) \le \frac{1}{d(n+1)} \left(n + (1-\epsilon)(n+1)(d(n+1)-1) \right).$$

In other words

$$\epsilon > \frac{1}{n+1},$$

which can be seen to be the same bound as in the contiguous case.

Even though the same theoretical bound on ϵ is reached, in practice the non-contiguous case is harder to solve. This is due to the fact that the increased dimension of the lattice to reduce both increases the time for LLL-reduction whilst decreasing the chances of the heuristics holding.

We ran some experiments, setting $Z_{i,j} = p^{(1-\epsilon)/d}$ and obtained the following results, again using a 160 bit prime number p;

d	ϵ	$1/\epsilon$	Actual Value of n	Time in Seconds
2	.500	2	2	0.067
4	.500	2	2	0.304
8	.500	2	2	1.135
16	.500	2	–	Infeasible
2	.250	4	4	0.393
4	.250	4	4	1.785
8	.250	4	–	Infeasible
2	.100	10	12	6.256
4	.100	10	–	Infeasible
2	.050	20	–	Infeasible

Hence as ϵ decreases we could only use fewer numbers of blocks to still recover the keys. This is because as ϵ decreases and d increases we obtain larger and larger matrices.

5. Conclusions

We have shown how to use lattice methods to break digital signature algorithms when small numbers of bits of many ephemeral keys are known. It goes without saying that our attack also applies when a large number of bits are known of a small number of ephemeral keys. Our attack relies on solving the many equations which arise in the multiple calls to the digital signature algorithm, rather than any underlying weaknesses of the discrete log problem or the choice of group. We have shown that in designing implementations in hardware or software of digital signature algorithms it is important that no bits of the ephemeral keys are leaked for whatever reason.

References

[1] L. Babai, On Lovász lattice reduction and the nearest point problem. *Combinatorica*, **6**, 1–13, 1986.

[2] D. Boneh and G. Durfee, Cryptanalysis of RSA with private key of less than $N^{0.292}$. *Advances in Cryptology, EUROCRYPT '99*, editor J. Stern. Springer-Verlag, LNCS 1592, 1–11, 1999.

[3] D. Boneh and R. Venkatesan, Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. *Advances in Cryptology, CRYPTO '96*, editor N. Koblitz, Springer-Verlag, LNCS 1109, 129–142, 1996.

[4] D. Coppersmith, Finding a small root of a bivariate integer equation; factoring with high bits known. *Advances in Cryptology, EUROCRYPT '96*, editor U. Maurer, Springer-Verlag, LNCS 1070, 178–189, 1996.

[5] D. Coppersmith, Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. of Cryptology*, **10**, 233–260, 1997.

[6] T. ElGamal, A public-key cryptosystem and a signature scheme based on discrete logarithms *IEEE Trans. Inform. Theory*, **31**, 469-472, 1985.

[7] N. Howgrave-Graham, Finding small roots of univariate modular equations revisited. *Proc. of Cryptography and Coding*, Springer-Verlag, LNCS 1355, 131-142, 1997.

[8] N. Howgrave-Graham, Computational mathematics inspired by RSA. *PhD. Thesis, University of Bath*, 1999.

[9] A.K. Lenstra, H.W. Lenstra and L. Lovász, Factoring polynomials with rational coefficients. *Math. Ann.*, **261**, 515–534, 1982.

[10] V. Shoup, NTL: A Library for doing Number Theory **http://www.shoup.net/**

Hewlett-Packard Laboratories, Filton Road, Stoke Gifford, Bristol, BS12 6QZ, United Kingdom
E-mail address: nichow@hplb.hpl.hp.com

Hewlett-Packard Laboratories, Filton Road, Stoke Gifford, Bristol, BS12 6QZ, United Kingdom
E-mail address: nigel_smart@hpl.hp.com