SOLVING SIMULTANEOUS MODULAR EQUATIONS OF LOW DEGREE

Johan Hastad*
MIT

Abstract: We consider the problem of solving systems of equations $P_i(x) \equiv 0 \pmod{n_i}$ $i = 1 \dots k$ where P_i are polynomials of degree d and the n_i are distinct relatively prime numbers and $x < \min(n_i)$. We prove that if $k > \frac{d(d+1)}{2}$ we can recover x in polynomial time provided $\min(n_i) > 2^{d^2}$. As a consequence the RSA cryptosystem used with a small exponent is not a good choice to use as a public key cryptosystem in a large network. We also show that a protocol by Broder and Dolev [4] is insecure if RSA with a small exponent is used.

1. Introduction

Let us start with some cryptographic motivation. The RSA function [10] is defined as $f(x) \equiv x^e \pmod{n}$. Here n is usually taken of the form $n = pq$ where p and q are two large primes and e is an integer relatively prime to $(p-1)(q-1)$. Using these parameters the function is $1-1$ when restricted to $1 \leq x \leq n, (x,n) = 1$. Furthermore the function is widely believed to be a *trapdoor function* i.e. given n and e it is easy to compute $f(x)$ and given $f(x)$ it is also easy to recover x provided one has some secret information but otherwise it is difficult to compute x. In this case the secret information is the factorization of n.

The RSA function can be used to construct a deterministic Public Key Cryptosystem (PKC) in the following way:

Each user B in a communication network chooses two large primes p and q and multiplies them together and publishes the result n_B together with a number e_B which is relatively prime to $(p-1)(q-1)$. He keeps the factorization of n_B as his private secret information. If any user A in the system wants to send a secret message m to another user B she retrieves B's published information computes $y \equiv m^{e_B} \pmod{n_B}$ and sends y to B. B now obtains the original message using his secret information, while somebody who does not know the secret information presumably faces an intractable computational task.

Public Key Cryptosystems are different and more complex objects than trapdoor functions. The reason is that a PKC involves a protocol consisting of several steps. For example the use of RSA in a PKC may present obstacles that did not occur when we considered it as a trapdoor function. Several people (including Blum, Lieberherr and Williams) have observed the following possible attack. Assume that 3 is chosen as the exponent and that A wants to send the same message m to users U_1, U_2 and U_3. She will compute and send $y_i \equiv m^3 \pmod{n_i}$ $i = 1, 2, 3$. If someone gains access to y_1,y_2 and y_3 then by using the fact that n_1, n_2 and n_3 will be relatively prime he can combine the

* Supported by an IBM fellowship, partially supported by NSF grant DCR-8509905

messages by chinese remaindering to get $m^3 \pmod{n_1n_2n_3}$ and since $m^3 < n_1n_2n_3$ he can recover m. In general if the exponent is e the number of messages needed is e.

A natural question is therefore: Is there a better way to send the same message to many people using this PKC?

A common heuristic tells us to use a "time stamp". Instead of sending the same message m to everybody one attaches the time and thus sends the encryption of $2^{|t_i|}m + t_i$ where $2^{|t_i|}m$ is the shifted message and t_i is the time when the message is sent to user U_i. This time will be different for the different receivers. The previous attack then fails.

If we assume that the times t_i are known to the cryptanalyst we are led to consider the following computational problem (for $e = 3$).

Given $(a_i m + b_i)^3 \pmod{n_i}$ where all the a_i and b_i are known is it possible to recover m in polynomial time?

We will prove in section 3 that the answer is YES if the number of similar messages is at least 7. In fact we will prove this as a special case of our main result, which is: Given a set of k polynomial equations

$$P_i(x) \equiv 0 \pmod{n_i} \quad i = 1, \ldots, k$$

each of degree $\leq d$, it is possible to recover all solutions in time polynomial in both k and $\log n_i$ if $k > d(d+1)/2$ provided $\min(n_i) > 2^{d^2}$.

Observe that the described attack does not work if the values of the t_i are not known to the cryptanalyst. Thus if for instance a random padding was used or if the time stamp was unknown then the present attack will not work. However, this weakness seems severe enough that if one uses RSA as a PKC then as a matter of prudence one should use a large exponent or even better one should use a probabilistic encryption scheme [3],[7] based on RSA. By [1],[3] this can be done with as much efficiency as in the deterministic case.

The outline of the paper is as follows. In section 2 we state some results from geometry of numbers which will be needed in later sections. In section 3 we state and prove our main result and in section 4 we derive some cryptographic applications.

2. Background from geometry of numbers.

The main tool in our algorithm will be the use of lattices and in this section we will gather the relevant background information. A *lattice* L is defined to be the set of points

$$L = \{y \mid y = \sum_{i=1}^{n} a_i \vec{b_i}, a_i \in \mathbf{Z} \}$$

where $\vec{b_i}$ are linearly independent vectors in $\mathbf{R}^n$. The set $\vec{b_i}$ is called a *basis* for the lattice and n is the dimension. The *determinant* of a lattice is defined to be the absolute value of the determinant of the matrix with rows $\vec{b_i}$. It is not hard to see that the determinant is independent of the choice of basis. The length of the shortest nonzero vector in the lattice is denoted by λ_1. Let us recall the following well known fact:

Theorem: *(Minkowski)* $\lambda_1 \leq \gamma_n^{\frac{1}{2}} (det(L))^{\frac{1}{n}}$ *where* γ_n *is Hermite's constant.*

Hermite's constant is not known exactly for $n > 8$ but Minkowski,s convex body theorem ([5], ix.7) implies that $\gamma_n \leq n$. Lenstra et al. showed in [8] that it was possible to find a vector in L of length at most $2^{\frac{n-1}{2}}\lambda_1$ in polynomial time. From their proof we can however derive a slightly better bound in the present case.

Theorem(LLL): *Given a lattice L as a basis of integer vectors of length at most B we can find a vector $\vec{b}$ in time $O(n^6(\log B)^3)$ which satisfies $||\vec{b}|| \leq 2^{\frac{n-1}{4}}(det(L))^{\frac{1}{n}}$.*

This gives an effective variant of Minkowski's theorem. Here $||\vec{b}||$ is the euclidean length of the vector $\vec{b}$. The bound on the running time assumes that multiplication of r bit numbers are done by classical arithmetic taking $O(r^2)$ steps. Using faster multiplication routines the bounds can be improved by a factor close to $n \log B$. Armed with this information we return to the original problem.

3. Main Theorem

Let us start by fixing some notation. Let $N = \prod_{i=1}^{k} n_i$ and $n = \min n_i$. Now we can state the problem formally:

Problem: *Given a set of k equations $\sum_{j=0}^{d} a_{ij}x^j \equiv 0 \ (mod\ n_i)$, $i = 1, \ldots, k$. Suppose that the system have a solution $x < n$ and the numbers n_i are pairwise relatively prime. Can we find such a solution efficiently?*

Before we state our main result let us give the basic ideas. Define $u_j < N$ to be the chinese remaindering coefficients i.e. $u_j \equiv \delta_{ij} \pmod{n_i}$ ($\delta_{ij} = 1$ if $i = j$ and 0 otherwise). We can combine the equations to a single equation using the chinese remainder theorem.

$$0 \equiv \textstyle\sum_{j=0}^{d} x^j \sum_{i=1}^{k} u_i a_{ij} \equiv \sum_{j=0}^{d} x^j c_j \pmod{N}$$

One of the important parts of the entire paper is the following simple lemma.

Lemma 1: *If $|c_j| < \frac{N}{(d+1)n^j}$ and we have at least one nonzero c_j then we can find all x satisfying $x < n$ and $\sum_{j=0}^{d} c_j x^j \equiv 0 \ (mod\ N)$ in time $O((d \log N)^3)$.*

Proof: If $|c_j| < \frac{N}{(d+1)n^j}$ then

$$|\sum_{j=0}^{d} c_j x^j| \leq \sum_{j=0}^{d} |c_j| n^j < N.$$

Thus the condition $\sum_{j=0}^{d} c_j x^j \equiv 0 \pmod{N}$ implies $\sum_{j=0}^{d} c_j x^j = 0$. In other words x solves the equation over the integers and to prove the lemma we just need the fact that we can solve polynomial equations over the integers quickly. Since we are in the special case that we are looking for an integer solution we can proceed as follows. Find all linear factors modulo a small prime. Now apply Hensel lifting to obtain these factors modulo a large power of the prime and finally check if any of the roots is a root over the integers. The estimate for the running time in the lemma is correct but not the best possible. ∎

The condition of Lemma 1 is quite unlikely to be fulfilled when we start with a general set of equations. In spite of this Lemma 1 will be one of our main tools for proving our main result, which is as follows.

Theorem: *Given a set of equations $\sum_{j=0}^{d} a_{ij}x^j \equiv 0 \pmod{n_i}$, $i = 1, 2, \ldots, k$ where the moduli n_i are pairwise relatively prime and $gcd(\langle a_{ij}\rangle_{j=0}^{d}, n_i) = 1$ for all i. Then we can find all $x < n$ satisfying the equations in time $O(d^6(\log N)^3)$ if*

$$N > n^{\frac{d(d+1)}{2}} 2^{\frac{(d+2)(d+1)}{4}} (d+1)^{(d+1)}.$$

As before $N = \prod_{i=1}^{k} n_i$, $n = \min n_i$, d is the degree of the equations and k is the number of equations. By $gcd(\langle a_{ij}\rangle_{j=0}^{d}, n_i)$ we mean the greatest common divisor of all $d+2$ numbers.

Proof: The idea is to use Lemma 1. However as we remarked it is quite unlikely that it will apply to our equations directly. However we have an extra degree of freedom. We can multiply the equation by an arbitrary constant S and we still get a valid equation. Using this trick we will be able to make the coefficients small. Thus we want to make $Sc_i \pmod{N}$ less than $\frac{N}{n^i(d+1)}$ in absolute value. Set up the following lattice L of dimension $d+2$:

$$\begin{aligned}
\vec{b}_1 &= (c_0, nc_1, n^2c_2 \ldots, n^d c_d, \tfrac{1}{(d+1)}) \\
\vec{b}_2 &= (N, \quad 0, \quad 0, \ldots, \quad 0, \quad 0) \\
\vec{b}_3 &= (\ 0, nN, \quad 0, \ldots, \quad 0, \quad 0) \\
\vec{b}_4 &= (\ 0, \quad 0, n^2N, \ldots, \quad 0, \quad 0) \\
&\ \ \vdots \\
\vec{b}_{d+2} &= (\ 0, \quad 0, \quad 0, \ldots, n^dN, \quad 0)
\end{aligned}$$

Let us see why this lattice is relevant to our purposes. Look at a generic vector $S\vec{b}_1 + \sum_{i=2}^{d+2} s_i\vec{b}_i$. Call the ith coordinate d_i. From the definition it follows that d_i is divisible by n^{i-1} and $\frac{d_i}{n^{i-1}} \equiv Sc_{i-1} \pmod{N}$. Thus if we find a vector $\vec{b} \in L$ satisfying $\|\vec{b}\| < \frac{N}{d+1}$ we know that $|d_i| < \frac{N}{d+1}$ and we get the desired bound for $Sc_i \pmod{N}$. The last coordinate is there to prevent $S = N$ which would make $d_i = 0$ for $i = 1, \ldots, d+1$.

We have only one term in the expansion of the determinant and we get

$$Det(L) = \frac{n^{\frac{d(d+1)}{2}} N^{d+1}}{(d+1)}.$$

Using the theorem of LLL in section 2 we know that we can find a vector $\vec{b}$ in L that satisfies

$$\|\vec{b}\| \leq 2^{\frac{d+1}{4}} \left(\frac{n^{\frac{d(d+1)}{2}} N^{d+1}}{d+1}\right)^{\frac{1}{d+2}}$$

As observed above to get the desired bound for the coefficients we need $\|\vec{b}\| < \frac{N}{d+1}$ and thus we need

$$2^{\frac{d+1}{4}} \left(\frac{n^{\frac{d(d+1)}{2}} N^{d+1}}{d+1}\right)^{\frac{1}{d+2}} < \frac{N}{d+1}$$

Raising both sides to the $d+2$ power and rearranging we see that this is equivalent to the condition in the theorem.

To finish the proof we need to prove that we have at least one nonzero coefficient. Since $\|\vec{b}\| < \frac{N}{d+1}$ we see by looking at the last coordinate that the coefficient S multiplying $\vec{b}_1$ satisfies $|S| < N$. Further we know that $S \neq 0$ since all nonzero vectors with $S = 0$ are of length at least N. This means that there is an n_i such that $S \not\equiv 0 \pmod{n_i}$. Look at the equation modulo this n_i. Using that $gcd(\langle a_{ij}\rangle_{j=0}^{d}, n_i) = 1$ we see that the equation is nontrivial. The bottleneck in the computation is the lattice computation and this gives the running time of the algorithm. ∎

Remark: One interesting open question is whether we can solve the problem with fewer equations. It does not seem possible to use this line of attack with substantially fewer equations. To see this one might argue as follows:

The probability that $|c_j| < \frac{N}{(d+1)n^j}$ for $j = 0, 1, \ldots, d$ for a fixed S is approximately $n^{-d(d+1)/2}$ and this would indicate that we should have $n^{d(d+1)/2}$ different S to choose between and therefore need at least $d(d+1)/2$ equations.

4. **Cryptographic Applications**

We get some immediate applications of the main theorem.

Application 1: *Sending linearly related messages using RSA with low exponent e is insecure. Sending more than $\frac{e(e+1)}{2}$ messages enables an adversary to recover the messages provided that the moduli n_i satisfy $n_i > 2^{\frac{(e+2)(e+1)}{4}}(e+1)^{(e+1)}$.*

Proof: Suppose we are given the encryption of k linearly dependent messages. We expand the eth power and we get k equations of degree e with the different moduli used n_i. We now apply the main theorem. We need to verify that the conditions of the main theorem are satisfied. If one of the gcd conditions is not satisfied we can factor one of the n_i and that way obtain the message. Finally

$$N = \prod_{i=1}^{k} n_i \geq n_1 \prod_{i=2}^{\frac{d(d+1)}{2}+1} n_i > 2^{\frac{(e+1)(e+2)}{4}}(e+1)^{(e+1)} n^{\frac{d(d+1)}{2}}$$

and hence we can apply our main theorem. ∎

If one is prepared to do computation which is exponential in the number of equations one can attack the cryptosystem also given exactly $\frac{e(e+1)}{2}$ messages. The way to proceed is to use almost the same lattice. The only difference is to replace the last coordinate of the first vector by $2^{\frac{e+2}{4}}$. Now the algorithm of LLL finds a vector in the lattice of length at most $N2^{\frac{e+2}{4}}$. This implies in particular that $c_i < N2^{\frac{e+2}{4}}n^{-i}$. Now it is no longer possible to conclude that x solves the equation over the integers but we know that the right hand side is a multiple of N not exceeding $e2^{\frac{e+2}{4}}N$ and hence we try all $e2^{\frac{e+2}{4}}$ possibilities.

Another way of encrypting messages was proposed by Rabin [9]. He uses $f(x) \equiv x^2 \pmod{n}$ where also here n is chosen to be a specific composite number for each user. Using the same methods we get:

Application 2: *Sending linearly related messages using the Rabin encryption function is insecure. If 3 such messages are sent it is possible to retrieve the message in polynomial time.*

Broder and Dolev proposed a protocol for flipping a coin in a distributed system [4]. Two of their essential ingredients were Shamir's method of sharing a secret [11] and the use of a deterministic PKC. The secret they use is the constant coefficient of a polynomial of degree t over a finite field. The secret is distributed by evaluating the polynomial at a given set of points. It is easy to see that $t + 1$ pieces each consisting of the value of the polynomial at a point are enough to get the secret back while t pieces are not sufficient to determine the polynomial. Broder and Dolev claim that t pieces are insufficient to find the secret even in the presence of the encryption of other pieces. This is not correct if the cryptosystem used is RSA with small exponent. This is because when knowing t pieces the secret enters linearly in the remaining pieces and hence we can use Application 1.

Application 3: *The protocol by Broder and Dolev is insecure if RSA with small exponent is used.*

Of course if other cryptosystems are used then this attack does not work. A different type of attack on the Broder-Dolev protocol has been proposed by Benny Chor [6]. This attack just relies on the protocol and not on the cryptosystem. A provably secure protocol has been designed by Awerbuch et.al. [2]. For further discussion of coin flipping protocols see [2] and [4].

Finally we remark that there does not seem to be any way to extend the above attack to RSA with large exponent. The reason is that the integers involved are too big even to write down. There is still a large amount of structure present and it would be interesting to investigate whether this structure could be exploited to yield a successful cryptanalytic attack on RSA with large exponent.

Acknowledgments: I would like to thank Silvio Micali, Shafi Goldwasser and Benny Chor for suggesting the problem, listening to early solutions and suggesting improvements and simplifications. They also pointed out the incorrectness in the proof of Broder and Dolev. I am also very grateful to Ron Rivest for greatly simplifying the proof and finally I would like to thank Jeff Lagarias for many helpful comments.

References:

[1] Alexi W., Chor B., Goldreich O. and Schnorr C.P. "RSA/Rabin Bits are $\frac{1}{2} + \frac{1}{poly(logN)}$ Secure" *Proceedings of 25th Annual IEEE Symposium on Foundations of Computer Science*, 1984, 449-457. Also this volume.

[2] Awerbuch B., Chor B., Blum M., Goldwasser S., and Micali S. "Fair Coin Flip in a Byzantine Environment" , manuscript in preparation.

[3] Blum M. and Goldwasser S. "An efficient Probabilistic Public Key Encryption Scheme which Hides all Partial Information" *Advances in Cryptology: Proceedings of CRYPTO 84 (Blakeley G.R. and Chaum D. eds.) Lecture Notes in Computer Science 196, Springer Verlag*, 1985 pp 289-299.

[4] Broder A.Z. and Dolev D. "Flipping Coins in Many Pockets" *Proceedings of 25th Annual IEEE Symposium on Foundations of Computer Science*, 1984, 157-170.

[5] Cassels J.W.S. "An Introduction to the Geometry of Numbers" Springer Verlag, Heidelberg 1971.

[6] Chor B. , Personal Communication 1985.

[7] Goldwasser S. and Micali S. "Probabilistic Encryption" *Journal of Computer and System Sciences* **28**, 1984, 270-299.

[8] Lenstra A.K. ,Lenstra H.W. and Lovasz L. "Factoring Polynomials with Integer Coefficients" *Matematische Annalen* **261** (1982) 513-534.

[9] Rabin M.O, "Digital Signatures and Public Key Functions as Intractable as Factorization" MIT/LCS/TR-212, 1979.

[10] Rivest R.L., Shamir A. and Adleman L. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems" *Communications of ACM* 21-2 February 1978.

[11] Shamir A. " How to Share a Secret", *Communications of ACM* 22 November 1979, 612-613.