N USING RSA WITH LOW EXPONENT IN A PUBLIC KEY NETWORK

by Johan Hastad*
MIT

Abstract: We consider the problem of solving systems of equations $P_i(x) \equiv 0 \pmod{n_i}$ $i = 1 \ldots k$ 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 $n_i >> 2^k$. This shows that RSA with low exponent is not a good alternative to use as a public key cryptosystem in a large network. It also shows that a protocol by Broder and Dolev [4] is insecure if RSA with low exponent is used.

1. Introduction

Let us start with some cryptographic motivation. The famous RSA function [8] is defined as $f(x) \equiv x^d \pmod{n}$. Here n is usually taken of the form $n = pq$ where p and q are two large primes and d 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 d it is easy to compute $f(x)$ and given $f(x)$ it is also easy to recover x provided you have some secret information but otherwise it is infeasible. 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 d_B which is relatively prime to $(p-1)(q-1)$. He keeps the factorization 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 $y \equiv m^{d_B} \pmod{n_B}$ and sends y to B. B now obtains the original message using his secret information while somebody else presumably faces an intractable computational task.

However PKC are different and more complex objects than trapdoor functions. 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 (at least Blum, Lieberherr and Williams) have observed the following 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 $y_i \equiv m^3 \pmod{n_i}$ $i = 1, 2, 3$. But using the fact that n_1, n_2 and n_3 are relatively prime a listener who know the values of y_1,y_2 and y_3 can combine the messages by chinese remaindering to get $m^3 \pmod{n_1 n_2 n_3}$ and since $m^3 < n_1 n_2 n_3$ he can recover m. In general if the exponent is d the number of messages needed is d.

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

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 $2^{|t|}m + t$ where $2^{|t|}m$ is the shifted message and t is the time (which will be different for the different receivers). The previous attack fails and we are led to the following computational problem (for $d = 3$).

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 see in section 3 that the answer is YES if the number of similar messages is at least 7. In fact we will prove that given a set of equations

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

where we have k polynomial equations of degree $\leq d$ it is possible to recover the solution in time polynomial in both k and $\log n_i$ if $k > d(d+1)/2$ provided $n_i >> 2^d$. Therefore we conclude that if RSA is to be used as a PKC we should use a large exponent or even better use a probabilistic encryption scheme [3],[6] based on RSA. By [1],[3] this can be done with as much efficiency as in the deterministic case.

2. The insecurity of a protocol by Broder and Dolev.

Broder and Dolev proposed a protocol for flipping a coin in a distributed system [4]. Some of their essential ingredients were Shamir's method of sharing a secret and the use of a deterministic PKC. They proposed to use the RSA. In [2] it is shown that what they really need from the security of the cryptosystem is:

Given the encryption of $a_i x + b_i$ with different keys it should be infeasible to decide the parity of x with a better probability than flipping a coin. The analysis in the next section shows that given this information we can, not only find the parity of x, but the exact value of x if the PKC is RSA with a small exponent. In the case of a large exponent the protocol is not known to be insecure but on the other hand there is no proof of correctness. A provably secure protocol has been designed by Awerbuch et.al. [2].

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 $\sum_{j=0}^{d} a_{ij} x^j \equiv 0 \pmod{n_i}$, $i = 1, \ldots, k$. *Suppose that the system have a solution $x < n$. Can we find such a solution efficiently?*

Before we give the theorem 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.

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

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

Lemma 1: If $|c_j| < \frac{N}{(d+1)n^j}$ we can find x in time polynomial in d, k and $\log n_i$.

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 in polynomial time. This follows from [7] but there are more efficient algorithms.

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:

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 $x < n$ and $gcd(\langle a_{ij} \rangle_{j=0}^{d}, n_i) = 1$ for all i. *Then it is possible to recover x in time polynomial in d, k and $\log n_i$ if*

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

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.

Proof: The idea is to use lemma 1. However as we remarked it is quite unlikely that it will apply to our equations directly. To get more possibilities we will multiply the i-th equation by a constant s_i before we combine them using chinese remaindering. If we have chosen the s_i carefully enough the resulting equation will have the desired small coefficients. We get

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

Let c_j denote the coefficient of x^j in this equation. To apply lemma 1 we want $|c_j| n^j < \frac{N}{d+1}$. The main tool for achieving this will be the use of lattices. We first start by recalling some background from the geometry of numbers.

3.1 Background from geometry of numbers.

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 wellknown facts:

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

γ_n is not known explicitly but we have an upper bound $\gamma_n \leq n$ [5].

Theorem: We can find a vector $\vec{b}$ **in polynomial time which satisfies** $\|\vec{b}\|/\lambda_1 \leq 2^{\frac{n}{2}}$.

This is bound you get from the famous algorithm in the paper by Lenstra,Lenstra and Lovasz [7]. By a result by Schnorr it is possible to replace the constant 2 by any number greater than 1 [9] but this is not important to us. Armed with this information we return to the original problem.

3.2 Continuation of Proof.

Define the following lattice L of dimension $k+d+1$ by its base vectors:

$$\vec{b}_1 = (a_{10}u_1, na_{11}u_1, n^2a_{12}u_1, \ldots, n^d a_{1d}u_1, \tfrac{N}{n_1(d+1)}, 0, \ldots, 0)$$
$$\vec{b}_2 = (a_{20}u_2, na_{21}u_2, n^2a_{22}u_2, \ldots, n^d a_{2d}u_2, 0, \tfrac{N}{n_2(d+1)}, \ldots, 0)$$

$$\vec{b}_k = (a_{k0}u_k, na_{k1}u_k, n^2a_{k2}u_k, \ldots, n^d a_{kd}u_k, 0, 0, \ldots, \tfrac{N}{n_k(d+1)})$$
$$\vec{b}_{k+1} = (N, 0, 0, \ldots, 0, 0, 0, \ldots, 0)$$
$$\vec{b}_{k+2} = (0, nN, 0, \ldots, 0, 0, 0, \ldots, 0)$$

$$\vec{b}_{k+d+1} = (0, 0, 0, \ldots, n^d N, 0, 0, \ldots 0)$$

Observe that

$$\sum_{i=1}^{k} s_i\vec{b}_i = (\sum_{i=1}^{k} s_i a_{i0}u_i + s_{k+1}N, n\sum_{i=1}^{k} s_i a_{i1}u_i + s_{k+2}N, \ldots, \frac{Ns_1}{n_1(d+1)}, \ldots, \frac{Ns_k}{n_k(d+1)}) \equiv$$

$$\equiv (c_0, nc_1, n^2c_2, \ldots, \tfrac{Ns_1}{n_1(d+1)}, \ldots, \tfrac{Ns_k}{n_k(d+1)}) \pmod{N}$$

Observe that for $1 \leq i \leq k+1$ the i'th coefficient is divisible by n^i. We multiply the different coefficients by the corresponding powers of n since we want $|c_j|n^j < \frac{N}{d+1}$. The last k coordinates are there to make the multipliers s_i small in a short vector in the lattice and the last $d+1$ vectors reflect the fact that we have a modular equation.

The only term in the expansion of the determinant is the diagonal term and we get

$$Det(L) = n^{\frac{d(d+1)}{2}} N^{d+k+1}(d+1)^{-k} \prod_{i=1}^{k} n_i^{-1} = n^{\frac{d(d+1)}{2}} N^{d+k}(d+1)^{-k}$$

This also shows that the vectors are independent. Combining the two theorems in section 3.1 we know that we can find a vector $\vec{b}$ in L that satisfies

$$\|\vec{b}\| < (k+d+1)^{\frac{1}{2}} 2^{\frac{k+d+1}{2}} Det(L)^{\frac{1}{k+d+1}}$$

Observe that to get the desired bounds for the c_i's we need

$$\|\vec{b}\| < \frac{N}{d+1}$$

A simple calculation shows that to get this we need exactly the bound from the theorem to get this.

To finish the proof we need to prove that we get a nontrivial equation. Since $\|\vec{b}\| < \frac{N}{d+1}$ we know by the expressions for the last k coordinates that $|s_i| < n_i$. $\vec{b}$ is also nonzero. This together with the bound for its length imply that there is at least one $s_i \neq 0$. Look at the equation (mod n_i) for the same i. Using that $0 \neq |s_i| < n_i$ and $gcd(\langle a_{ij} \rangle_{j=0}^{d}, n_i) = 1$ we see that this is a nontrivial equation.

The proof is complete.

4. Cryptographic Corollaries

We get some immediate corollaries of the main theorem

Corollary 1: Sending linearly related messages using RSA with low exponent is insecure. *Sending more than $\frac{d(d+1)}{2}$ messages enables an adversary to recover the messages.*

This follows directly from our main the main theorem assuming that the constants depending on the dimension is small compared to the moduli. In the same spirit we get

Corollary 2: Sending linearly related messages using the Rabin encryption function is insecure. *If 4 such messages are sent it is possible to retrieve the message.*

If one does a bit of extra work it is possible to say something about the cases of equality ($\frac{d(d+1)}{2}$ and 3 messages respectively) but we omit the details.

Corollary 3: The protocol by Broder and Dolev is insecure if RSA with low exponent is used.

Follows from the analysis in [2] and the main theorem.

The theorem also proves that we should not encode messages that are small known polynomials in some unknown but this seems quite farfetched.

5. Open questions

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

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

There does not seem to be any way to extend the above attack to RSA with large exponent. The reason being 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 used.

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 flaws in the argument of Broder and Dolev.

References:

[1] Alexi W., Chor B., Goldreich O. and Schnorr C.P. "RSA/Rabin Bits are $\frac{1}{2} + \frac{1}{poly(logN)}$ Secure"
FOCS 1984 pp 449-457

[2] Awerbuch B., Chor B., Goldwasser S. and Micali S. "Provably Secure 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" Presented in Crypto 1984

[4] Broder A.Z. and Dolev D. "Flipping Coins in Many Pockets"
FOCS 1984 pp 157-170

[5] Cassels J.W.S. "Geometry of Numbers" Springer 1959

[6] Goldwasser S. and Micali S. "Probabilistic Encryption"
JSCC **28** 270-299

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

[8] Rivest R.L., Shamir A. and Adleman L. "A Method for Obtaining Digital Signatures
and Public Key Cryptosystems" CACM 21-2 February 1978.

[9] Schnorr C.P. "A Hierarchy of Polynomial Basis Reduction Algorithms" , manuscript