THE REQUIREMENT FOR STRONG PRIMES IN RSA ENCRYPTION

Robert D. Silverman

RSA Laboratories

§1. Introduction.

The current EMV standard for generating RSA keys $N = pq$ has three requirements. The first is that p and q be selected as *strong* primes. A strong prime p is one for which $p \pm 1$ has at least one large prime factor, in particular one which exceeds 2^{128}. The historical reason for this requirement has been to guard against the Pollard $P-1$ and Williams $P+1$ factoring algorithms.

The second requirement is that when one does an encryption of a message M into ciphertext C by means of the congruence $C = M^e \bmod N$, that the encrypting exponent e has large order mod N.

The third requirement is that both primes p, q be congruent to $-1 \bmod 4$. The reason for this requirement is unclear to this author. The requirement is irrelevant to any factoring attacks on the modulus and adds no security to the key.

This paper shall argue that the more recent Elliptic Curve Factoring Algorithm (ECM) obsoletes the requirement for strong primes in the sense that generating an RSA key with strong primes does *not* add any security to the key over using random primes. This is because ECM majorizes the $P \pm 1$ algorithms. Therefore, it is cryptographically secure for BSAFE to ignore this requirement.

Furthermore, a more recent paper by Bach and Shallit has shown that algorithms exist for finding a factor p dividing a larger number N provided that any one of a number of other cyclotomic polynomials in p, e.g. $p^2+1, p^2 \pm p+1$, etc. has only small prime factors. The current

Typeset by $\mathcal{AMS}$-TEX

standard does nothing to guard against these attacks. It should be noted that ECM obsoletes these methods as well.

This paper shall also argue that even a randomly generated prime of at least 256 bits has a vanishingly small chance of being found by ECM, let alone by $P \pm 1$. This theoretical estimate shall be backed by a summary of practical experience with these methods.

Finally, we shall use a little bit of elementary group theory, combined with the fact that the probability that $(p-1)(q-1)$ has only small prime factors is vanishingly small, to show that the probability that e has small order is also vanishingly small.

§2. **Dominance of ECM over older Methods**

We start by defining some terminology. A number all of whose prime factors are small is said to be *smooth*. In particular, an integer is said to be k-smooth if all of its prime factors are less than k.

There are currently two classes of factoring algorithms that are applied to large integers. The first class attempts to find a factor $p|N$ by computing a multiple of a random element in some specially selected group such that the order of that group is divisible only by small primes. The second class includes algorithms such as the Multiple Polynomial Quadratic Sieve (MPQS) and the Numbner Field Sieve (NFS) and their variations, and will not be discussed here.

The $P-1$ method uses the group $\mathbb{Z}/N\mathbb{Z}^*$ in the hope that the order of that group, $p-1$, will be divisible only by small primes. The $P+1$ method works in the multiplicative sub-group of order $p+1$ in the finite field $GF(p^2)$. These two algorithms can be quite effective at quickly finding small prime factors (5 to 25 digits) of large integers. However, in over 15 years of experience by the entire factoring research community, the largest factor ever found with these methods has had 34 decimal digits. This was found with an FFT version of $P-1$ [2].

By Mordell's Theorem, addition of points on an Elliptic Curve forms a finitely generated Abelian group over Q. According to a theorem of Hasse, if one takes the same curve over the finite field $GF(p)$, the order of the curve shall be an integer in the range $[p-2\sqrt{p}+1, p+2\sqrt{p}+1]$.

ECM works by selecting a random Elliptic Curve over $\mathbb{Z}/N\mathbb{Z}$ and a random point on the curve.

One then hopes to find a multiple of that point which is the identity element mod p, but is not the identity element in $\mathbb{Z}/N\mathbb{Z}$. This will reveal p. This is easy to do if the order of the curve is divisible only by small primes .

The advantage that ECM has over $P \pm 1$ is that if one curve fails it is possible to choose a different curve and hope that its group order is divisible only by small primes. With $P \pm 1$ there is only one group to work with. If that one fails, you are out of luck.

Suppose we choose our primes for our RSA key such that $p \pm 1, q \pm 1$ have no small factors and are thus inaccessible to $P \pm 1$. This does not guard against the existence of a small value of $k, k \neq 1$, such that $p \pm k$ is divisible by only small primes. And if such a k exists, ECM can succeed where $P \pm 1$ fails. A theorem of Deuring states that for every integer in the range $[p - 2\sqrt{p} + 1, p + 2\sqrt{p} + 1]$, there exists some Elliptic Curve over $GF(p)$ which has that group order. While $p \pm 1$ may not be smooth, $p \pm k$ will be smooth for some k and hence accessible to ECM. It is impossible to guard against all such possible values of k.

§3. **Probability of Succes with ECM,** $P \pm 1$

References [1] and [2] discuss in detail the practical computing limits of ECM and $P \pm 1$. This section summarizes those results. We shall assume two cases; case 1 is for 256-bit p, case 2 is for 384-bit p.

The $P \pm 1$ methods and ECM are two step algorithms. In the first step one hopes that the order of the group is divisible only by small primes up to some bound, $B1$. In the second step one allows for the possiblity that the group order might be smooth up to $B1$ with the exception of a single, larger prime greater than $B1$ and less than a second bound, $B2$. It takes about 1 hour to execute $P - 1$ on a 100-digit N to a step 1 limit of 10^7 on a large, fast, modern processor. This step is virtually impossible to run in parallel. While some small degree of parallelism may be achieved in performing the multiple precision arithmetic, this degree is limited to at most the number of bits in the modulus divided by the word size of the machine. It takes an additional hour to run step 2 to about 10^{13}, but this step does run well in parallel. The algorithm scales approximately quadratically with the size of the number being factored, so the corresponding

times for a 512-bit key will increase by a factor of about 2.4 and by a factor of 5.3 for a 768-bit key. The probability of succeeding with ECM or $P-1$ can be approximated by a theorem of Canfield, Erdös, and Pomerance. The probability that an integer Y has all of its prime factors less than X is approximately $\mu^{-\mu}$ where $\mu = \log Y / \log X$. This probability can also be computed somewhat more accurately using Dickman's function [1]. Furthermore, Dickman's function can be used to accurately compute the effect of Step 2 of both the $P \pm 1$ and ECM algorithms. Step 2 of these algorithms is asymptotically negligible in increasing the probability of success, but for numbers within practical computer range it has the effect of increasing the chance of success by a small factor, typically 2 to 3. The table below gives the step1/step2 limits for $P-1$, the corresponding computation time, and the probabilities of finding a 256-bit factor of a 512-bit modulus and a 384-bit factor of a 768-bit modulus with $P-1$ using Step 1 only. Step 2 will increase the probabilities by a small factor. Times (T) are given in hours.

Step 1 Limit	Step 2 Limit	256-Bit T	768-Bit T	P(256-bit success)	P(384-bit success)
10^7	10^{14}	5	13	2×10^{-12}	6×10^{-21}
10^8	10^{16}	50	130	2.2×10^{-10}	1.4×10^{-17}
10^9	10^{18}	500	1300	$7.4 \times 10^{-}9$	4.9×10^{-15}
10^{10}	10^{20}	5000	13000	$1.1 \times 10^{-}7$	4.5×10^{-13}
10^{11}	10^{22}	50000	130000	$9.3 \times 10^{-}7$	1.6×10^{-11}
10^{12}	10^{24}	500000	1.3×10^6	$5.2 \times 10^{-}6$	3×10^{-10}

To summarize this table, even if one were to spend about .5 million hours (57 years) running $P-1$ on a 512-bit key, the chance of success is still less than 1 in 100,000. Once could succeed with certainty in substantially less time by using the Number Field Sieve.

It takes about .5 hours to run step 1 of ECM to 10^7 on a MIPS-R10000 (with a 100-digit composite) using the fastest implementation available today (by P. Montgomery). It takes about

.33 hours to run step 2 to 10^{10}. Reference [1] describes how to select optimally the step 1 and step 2 parameters when looking for a factor of given size. It also includes a table giving the expected amount of work to find a factor with probability $1 - 1/e \sim .63$. This probability is the one attained with optimal selection of the ECM parameters.

ECM performs approximately like an $O(p^{1/5})$ algorithm for factors that are within reach of even extraordinary computer facilities. The cost to find a 128-bit number (with probability $1-1/e$) is about 4900 curves with step1/step 2 limits of $3.3\times10^6/1.3\times10^8$. Montgomery's program will take about 1/4 hour to run one curve with these limits on a MIPS-R10000. One can run independent curves in parallel. Doubling the number of curves (and the total time) will increase the chance of success to about .86. A 256-bit number will take approximately $2^{(128/5)} \sim 5 \times 10^7$ as much work to find with the same probability. A 384-bit number will take about $2^{(256/5)} \sim 2.5 \times 10^{15}$ times as much work. These times are considerably longer than it would take to factor N with the Number Field Sieve.

§4. **Practical Experience with $P-1$ and ECM**

$P-1$ has been in use since about 1975. ECM has been in use since 1985. Since then, the largest prime ever found by $P-1$ was 34-decimal digits. It is hard to know how much total effort has been expended with this method, but it is safe to say that during this period several tens-of-thousands of $P-1$ trials have been run.

The largest factor ever found by ECM was 47 decimal digits. This is thought to have been an extraordinary bit of luck. The second largest factor ever found was 43 digits. This experience has been with millions of elliptic curves trials by the entire factoring research community over the last 12 years. Only about a dozen factors greater than 2^{128} have been found in that time. Finding a 50-digit factor should take about 4 times this total effort, and finding a 256-bit factor should take about 50 million times as this total effort.

§5. **Order of the Encrypting Exponent**

Even in the extremely unlikely event that one of $p-1$ or $q-1$ has only small factors, it is much

more unlikely that *both* have only small factors. Suppose r is a moderately large prime exactly dividing $LCM(p-1, q-1)$. Suppose r does *not* divide $ord(e)$ mod N. It follows immediately that e must be an r^{th} power mod$N$. There are no more than r such integers. It is easy to show that r must be less than $\sqrt{N/4}$, so the chance that e is an r^{th} power is no more than $1/(2\sqrt{N})$, which for 512-bit N is about 1 in 10^{78}. Therefore, either r divides the order of e, and hence the order of e is large, or e is a perfect power mod N with vanishingly small probability.

Futhermore, suppose $GCD(p-1, q-1) = s$. If p, q are randomly chosen, then s will be very small. The probability that a randomly chosen integer mod N has order k is at most $ks/LCM(p-1, q-1) = ks^2/((p-1)(q-1)) = ks^2/N + \epsilon$, with ϵ being very small. If k is small, then this probability will be correspondingly small.

§6. **Summary**

If one expends the same level of effort with $P-1$ as was spent in factoring RSA-130 with the number field sieve (roughly 50 years on a single Sparc-10), one should be able to take step 1 to about 10^{12} and step 2 to about 10^{24}. This would give a probability of 9×10^{-7} of finding a 256-bit prime factor and 2×10^{-11} of finding a 384-bit prime factor.

Please note that bounding the time one spends with ECM does not allow one to choose its parameters optimally [2]. It is therefore very difficult to estimate the chance of success within an *a priori* fixed amount of time. In a fixed time one can vary the step 1 and step 2 parameters as well as the number of curves. The response surface is highly non-linear. Extensive computations would have to be performed to determine the optimal ECM parameters for a given level of event when looking for 256-bit and 384-bit primes. A rough estimate suggests 2 to 3 orders of magnitude greater chance than with $P-1$. One still only has about a 1 in 10,000 chance of finding a 256-bit prime, whereas NFS can succeed with certainty.

CONCLUSION: Randomly chosen 256-bit and 384-bit primes are safe from attacks via $P \pm 1$ and ECM. The probability that randomly chosen primes will yield an encrypting exponent with small order is negligible. Therefore it is cryptographically secure for BSAFE to ignore the EMV standard. Indeed, following the EMV standard can give a false sense of security. Keys generated

according to the requirements are no more secure than keys that are not.

References

1. P.L. Montgomery & R.D. Silverman, *An FFT Extension to the $P-1$ Factoring Algorithm*, Math. Comp. **54** (1990), 839-854.
2. R.D. Silverman & S.S. Wagstaff Jr., *A Practical Analysis of the Elliptic Curve Factoring Algorithm*, Math. Comp. **61** (1993), 445-462.
3. P.L. Montgomery, *Speeding the Pollard and Elliptic Curve Methods of Factorization*, Math. Comp. **48** (1987), 243-264.
4. E. Bach & J. Shallit, *Factoring with Cyclotomic Polynomials*, Math. Comp..
5. P.L. Montgomery, doctoral thesis, UCLA (1992).

20 Crosby Drive

Bedford, MA 01730