Cryptanalysis of Short RSA Secret Exponents

MICHAEL J. WIENER

Abstract—**A cryptanalytic attack on the use of short RSA secret exponents is described. The attack makes use of an algorithm based on continued fractions that finds the numerator and denominator of a fraction in polynomial time when a close enough estimate of the fraction is known. The public exponent e and the modulus pq can be used to create an estimate of a fraction that involves the secret exponent d. The algorithm based on continued fractions uses this estimate to discover sufficiently short secret exponents. For a typical case where $e < pq$, $GCD(p-1, q-1)$ is small, and p and q have approximately the same number of bits, this attack will discover secret exponents with up to approximately one-quarter as many bits as the modulus. Ways to combat this attack, ways to improve it, and two open problems are described. This attack poses no threat to the normal case of RSA where the secret exponent is approximately the same size as the modulus. This is because this attack uses information provided by the public exponent and, in the normal case, the public exponent can be chosen almost independently of the modulus.**

I. Introduction

FROM THE SET of all key pairs for the RSA public-key cryptosystem [5], some key pairs have properties that can be exploited by various cryptanalytic attacks. Some attacks exploit weaknesses in the modulus, and others exploit weaknesses in the public exponent or the secret exponent. The weaknesses discussed here are those that allow an attack on RSA to be completed in a length of time that is polynomial in the length of the modulus.

Attacks on the RSA modulus are aimed at discovering the two prime factors (p and q) of the modulus. One such attack can be used to factor the modulus when the prime factors of either $p-1$ or $q-1$ are all small [3]. The modulus can also be factored when the prime factors of either $p+1$ or $q+1$ are all small [6]. There is a simple algorithm for factoring the modulus when the difference between the primes is bounded by $\sqrt{p}(\log p)^k$ for some constant k. This algorithm is based upon the following identity:

$$\left(\frac{p+q}{2}\right)^2 - pq = \left(\frac{p-q}{2}\right)^2.$$

The modulus can be factored by finding $(p+q)/2$ and $(p-q)/2$. $((p+q)/2)^2$ can be found in a linear search through the perfect squares starting from the modulus. The correct square is found when the difference between the square and the modulus is itself a perfect square.

There are various attacks on RSA that require, among other conditions, either the public or secret exponent to be short. In some cases it may be desirable to use a shorter public or secret exponent because this reduces the encryption or decryption execution time. This is because, for a fixed modulus size, the RSA encryption or decryption time is roughly proportional to the number of bits in the exponent. One situation where the use of short exponents is particularly advantageous is when there is a large difference in computing power between two communicating devices. An example of this is when RSA is used in communications between a smart card and a larger computer. In this case, it would be desirable for the smart card to have a short secret exponent, and for the larger computer to have a short public exponent in order to reduce the processing required in the smart card. However, one must be wary of short exponent attacks on RSA.

Short public exponents can be exploited when the same message is broadcast to many parties [1]. To illustrate this attack, suppose that a message m is broadcast to three parties in which the public exponents are $e_1 = e_2 = e_3 = 3$, and in which the moduli are n_1, n_2, and n_3. The encrypted messages are

$$m^3 \bmod n_1,\ m^3 \bmod n_2, \text{ and } m^3 \bmod n_3.$$

Using the Chinese remainder theorem, one can find $m^3 \bmod n_1 n_2 n_3$. However, $m^3 < n_1 n_2 n_3$ because $m < n_1, n_2, n_3$. Therefore, m^3 is not affected by being reduced modulo $n_1 n_2 n_3$, and the message can be recovered by taking the cube root of m^3. In this paper, an attack on short secret exponents is described. This attack is based upon continued fractions.

II. Continued Fractions Background

Continued fractions can be used to find the numerator and denominator of a fraction when a close enough estimate of the fraction is known. This will be related to RSA in Section IV where the public exponent and modulus will be used to construct an estimate of a fraction involving the secret exponent.

The algorithm for using continued fractions to find the numerator and denominator of a fraction given an estimate will be referred to here as the continued fraction

Manuscript received July 15, 1989. This work was presented at Eurocrypt '89, Houthalen, Belgium, April 12, 1989.

The author is with Bell-Northern Research, Ltd., P.O. Box 3511, Station C, Ottawa, Ontario, Canada, K1Y 4H7.

IEEE Log Number 8932929.

algorithm. This algorithm will be described in Section III. A background in continued fractions for a discussion of the continued fraction algorithm is presented in this section. Further discussion of continued fractions can be found in [2].

A continued fraction is an expression of the form

$$\cfrac{a_1}{q_1+\cfrac{a_2}{q_2+\cfrac{a_3}{\cdots\cfrac{}{q_{m-1}+\cfrac{a_m}{q_m}}}}}$$

$$=a_1/(q_1+a_2/(q_2+a_3/(\cdots/(q_{m-1}+a_m/q_m)\cdots))). \tag{1}$$

We are interested in continued fractions that have all the a_i's in (1) equal to one. For convenience, let us define

$$\langle q_0,q_1,\cdots,q_m\rangle$$
$$=q_0+1/(q_1+1/(q_2+1/(\cdots/(q_{m-1}+1/q_m)\cdots))). \tag{2}$$

For example, $\langle 0,2,1,3\rangle = 0+1/(2+1/(1+1/3)) = 4/11$. $\langle 0,2,1,3\rangle$ is called the continued fraction expansion of $4/11$. The continued fraction expansion of a positive rational number f is formed by subtracting away the integer part of f and repeatedly inverting the remainder and subtracting away the integer part until the remainder is zero. Let q_i be the integer quotient and r_i be the remainder at step i, and let m be the number of inversion steps:

$$q_0=\lfloor f\rfloor, \qquad r_0=f-q_0, \quad \text{and}$$
$$q_i=\left\lfloor\frac{1}{r_{i-1}}\right\rfloor, \qquad r_i=\frac{1}{r_{i-1}}-q_i, \qquad \text{for } i=1,2,\cdots,m. \tag{3}$$

Because $r_m=0$, we have $f=\langle q_0,q_1,\cdots,q_m\rangle$. There are two observations which can be made at this point that will be useful later on. The first is that $q_m\geq 2$. This is true because $q_m=1$ implies that $r_{m-1}=1$ which is impossible. The second is that for any $x>0$,

$$\langle q_0,q_1,\cdots,q_m\rangle<\langle q_0,q_1,\cdots,q_{m-1},q_m+x\rangle, \quad \text{if } m \text{ is even},$$
$$\langle q_0,q_1,\cdots,q_m\rangle>\langle q_0,q_1,\cdots,q_{m-1},q_m+x\rangle, \quad \text{if } m \text{ is odd}. \tag{4}$$

This can be seen by looking at the number of levels of fraction nesting in (2).

We will now consider how one would go about reconstructing f from its continued fraction expansion. Using (2), f can be reconstructed by starting from q_m and adding and inverting at each step back to q_0. However, it is useful to be able to reconstruct f starting from q_0. Let n_i and d_i, $i=0,1,\cdots,m$ be a sequence of numerators and denominators defined as follows:

$$\frac{n_i}{d_i}=\langle q_0,q_1,\cdots,q_i\rangle, \qquad GCD(n_i,d_i)=1, \quad \text{for } i=0,1,\cdots,m. \tag{5}$$

It can be shown that

$$n_0=q_0, \qquad d_0=1,$$
$$n_1=q_0q_1+1, \qquad d_1=q_1,$$
$$n_i=q_in_{i-1}+n_{i-2}, \qquad d_i=q_id_{i-1}+d_{i-2}, \quad \text{for } i=2,3,\cdots,m. \tag{6}$$

In this way, the fraction $f=n_m/d_m$ can be reconstructed.

There is a relationship between the numerators and denominators that will be useful later on. It can be shown that

$$n_id_{i-1}-n_{i-1}d_i=-(-1)^i, \qquad \text{for } i=1,2,\cdots,m. \tag{7}$$

Sufficient background in continued fractions has been presented for a discussion of the continued fraction algorithm.

III. Continued Fraction Algorithm

Let f' be an underestimate of f:

$$f'=f(1-\delta), \qquad \text{for some } \delta\geq 0. \tag{8}$$

Let q_i, r_i and q_i', r_i' be the ith quotients and remainders of f and f' respectively. If δ is small enough, then the numerator and denominator of f can be found using the following algorithm. Repeat the following until f is found.

- Generate the next quotient (q_i') of the continued fraction expansion of f'.
- Use (6) to construct the fraction equal to
  $\langle q_0',q_1',\cdots,q_{i-1}',q_i'+1\rangle$, if i is even,
  $\langle q_0',q_1',\cdots,q_{i-1}',q_i'\rangle$, if i is odd.
- Check whether the constructed fraction is equal to f.

The reason for adding one to even quotient values is that the guess of f should be larger than f', because $f\geq f'$, and it can be seen from (4) that $\langle q_0',q_1',\cdots,q_{i-1}',q_i'\rangle$ is less than $f'=\langle q_0',q_1',\cdots,q_{i-1}',q_i'+r_i'\rangle$. Note that a test must exist to determine whether a guess of f is correct.

The continued fraction algorithm will succeed if

$$\langle q_0,q_1,\cdots,q_{m-1},q_m-1\rangle<f'\leq\langle q_0,q_1,\cdots,q_m\rangle, \quad \text{if } m \text{ is even},$$
$$\langle q_0,q_1,\cdots,q_{m-1},q_m+1\rangle<f'\leq\langle q_0,q_1,\cdots,q_m\rangle, \quad \text{if } m \text{ is odd}. \tag{9}$$

We will now consider the implications of (9) on the size of δ. Solving (8) for δ yields

$$\delta=1-\frac{f'}{f}. \tag{10}$$

Separate analyses will be done for the following cases: $m = 0$, $m = 1$, m even and $m \geq 2$, and m odd and $m \geq 3$.

Case 1: $m = 0$.

Using (9) to substitute for f' in (10) yields

$$\delta < 1 - \langle q_0 - 1 \rangle / \langle q_0 \rangle. \tag{11}$$

Using (2), this reduces to $\delta < 1/q_0$ that can be rewritten as (recall that $n_0 = q_0$ and $d_0 = 1$)

$$\delta < \frac{1}{n_0 d_0}. \tag{12}$$

Case 2: $m = 1$.

Using (9) to substitute for f' in (10) yields

$$\delta < 1 - \langle q_0, q_1 + 1 \rangle / \langle q_0, q_1 \rangle. \tag{13}$$

Using (2), this reduces to

$$\delta < \frac{1}{(q_0 q_1 + 1)(q_1 + 1)}. \tag{14}$$

It was shown earlier that $q_m \geq 2$. This implies that for this case, $(3/2)q_1 \geq q_1 + 1$. Combining this with (14) and the expressions for n_1 and d_1 in (6) yields that

$$\delta < \frac{1}{\frac{3}{2} n_1 d_1} \tag{15}$$

is sufficient to guarantee the success of the continued fraction algorithm.

Case 3: m even and $m \geq 2$.

Using (9) to substitute for f' in (10) yields

$$\delta < 1 - \langle q_0, q_1, \cdots, q_{m-1}, q_m - 1 \rangle / \langle q_0, q_1, \cdots, q_m \rangle. \tag{16}$$

Using (6), we have

$$\langle q_0, q_1, \cdots, q_{m-1}, q_m - 1 \rangle = \frac{(q_m - 1) n_{m-1} + n_{m-2}}{(q_m - 1) d_{m-1} + d_{m-2}} \quad \text{and}$$

$$\langle q_0, q_1, \cdots, q_m \rangle = \frac{q_m n_{m-1} + n_{m-2}}{q_m d_{m-1} + d_{m-2}}. \tag{17}$$

Substituting these expressions into (16) yields

$$\delta < \frac{n_{m-1} d_{m-2} - n_{m-2} d_{m-1}}{(q_m n_{m-1} + n_{m-2})(q_m d_{m-1} + d_{m-2} - d_{m-1})}. \tag{18}$$

Using (7) and the expressions for n_m and d_m in (6) yields

$$\delta < \frac{1}{n_m (d_m - d_{m-1})}. \tag{19}$$

Therefore,

$$\delta < \frac{1}{n_m d_m} \tag{20}$$

is sufficient to guarantee the success of the continued fraction algorithm.

Case 4: m odd and $m \geq 3$.

Performing a similar analysis to the one in case 3 yields

$$\delta < \frac{1}{n_m (d_m + d_{m-1})}. \tag{21}$$

Because $d_m = q_m d_{m-1} + d_{m-2}$ and $q_m \geq 2$, we have $d_m + d_{m-1} \leq (3/2) d_m$. Therefore,

$$\delta < \frac{1}{\frac{3}{2} n_m d_m} \tag{22}$$

is sufficient to guarantee the success of the continued fraction algorithm.

Taking into account the results of all four cases,

$$\delta < \frac{1}{\frac{3}{2} n_m d_m} \tag{23}$$

is sufficient to guarantee the success of the continued fraction algorithm. Recall that n_m and d_m are the numerator and denominator of f.

Let us now consider the execution time of this algorithm. Let $x = \max(n_m, d_m)$. The number of quotients in the continued fraction expansion of f can be shown to be $O(\log x)$. For each quotient, a guess of f is generated and tested. The calculations required to generate each guess of f is polynomial in $\log x$. Therefore, assuming that the test of whether the guess of f is correct in polynomial in $\log x$, the continued fraction algorithm execution time is polynomial in $\log x$.

IV. Continued Fraction Algorithm Applied to RSA

The following relationship between the public exponent e and the secret exponent d is given in [5]:

$$ed \equiv 1 \pmod{LCM(p - 1, q - 1)}. \tag{24}$$

This relationship is necessary for exponentiation with the public exponent and secret exponent to be inverses of each other. From (24), there must exist an integer K such that

$$ed = K \cdot LCM(p - 1, q - 1) + 1. \tag{25}$$

If we let $G = GCD(p - 1, q - 1)$ and use the fact that $LCM(p - 1, q - 1) = (p - 1)(q - 1)/G$, we get

$$ed = \frac{K}{G}(p - 1)(q - 1) + 1. \tag{26}$$

It is possible for K and G to have common factors. Let us define $k = K / GCD(K, G)$ and $g = G / GCD(K, G)$. Then $k/g = K/G$, and $GCD(k, g) = 1$. We now have

$$ed = \frac{k}{g}(p - 1)(q - 1) + 1. \tag{27}$$

Dividing through by dpq in (27) gives

$$\frac{e}{pq} = \frac{k}{dg}(1 - \delta), \qquad \text{where } \delta = \frac{p + q - 1 - \frac{g}{k}}{pq}. \tag{28}$$

Note that e/pq consists entirely of public information and is a close underestimate of k/dg. Before invoking the continued fraction algorithm, we must remember that

this algorithm always finds fractions in lowest terms. From (25), we see that $GCD(K,d)=1$. Because k divides K, we have $GCD(k,d)=1$. Also, $GCD(k,g)=1$ by definition. Therefore, $GCD(k,dg)=1$, and the continued fraction algorithm can be used to find k and dg as long as δ is small enough.

Using the expression for δ in (28) and the restriction on δ in (23), it can be shown that

$$kdg < \frac{pq}{\frac{3}{2}(p+q)} \tag{29}$$

is sufficient to allow k and dg to be found. Note that $(-1-g/k)$ in the expression of δ was dropped because it is small compared to $(p+q)$. This does not affect the validity of (29) because $(-1-g/k)$ serves to reduce the size of δ.

We will now consider how one could test whether a guess of k and dg is correct. In order to simplify this test, we will assume that $ed > pq$. This is not a particularly restrictive assumption because when either e or d is fixed, the expected value of the other is approximately pq/G (recall that $G = GCD(p-1, q-1)$). Unless G is chosen to be large, it is very likely that $ed > pq$. From (27), a consequence of $ed > pq$ is that $k > g$. By rewriting (27) as

$$edg = k(p-1)(q-1) + g \tag{30}$$

we see that dividing edg by k yields a quotient of $(p-1)(q-1)$ and a remainder of g as long as $k > g$. This provides a guess of $(p-1)(q-1)$ and of g. If the guess of $(p-1)(q-1)$ is zero, then k and dg are wrong. This case must be filtered out at this point or the remainder of this test will succeed in factoring pq into 1 and pq. The guess of $(p-1)(q-1)$ can be used to create a guess of $(p+q)/2$ using the following identity:

$$\frac{pq-(p-1)(q-1)+1}{2} = \frac{p+q}{2}. \tag{31}$$

If the guess of $(p+q)/2$ is not an integer, then the guess of k and dg is wrong. The guess of $(p+q)/2$ can be used to create a guess of $((p-q)/2)^2$ using the following identity:

$$\left(\frac{p+q}{2}\right)^2 - pq = \left(\frac{p-q}{2}\right)^2. \tag{32}$$

If the guess of $((p-q)/2)^2$ is a perfect square, then the original guess of k and dg is correct. The secret exponent d can be found by dividing dg by g. Recall that g was the remainder when edg was divided by k. We can also recover p and q easily from $(p+q)/2$ and $(p-q)/2$.

If nothing special is done to combat this continued fraction attack on RSA, then one can expect g to be small, and $k < dg$. Under these conditions, we can see from (29) that secret exponents with up to approximately one-quarter as many bits as the modulus can be found in polynomial time. This attack cannot be extended to the normal case where the secret exponent is approximately the same size as the modulus because it relies on the public exponent providing information to help factor the modulus and, in the normal case, the public exponent can be chosen almost independently of the modulus.

V. An Example

In this section, the continued fraction algorithm will be applied to a small RSA key pair. For this example

$$pq = 8927 \quad \text{and} \quad e = 2621.$$

A continued fraction expansion is performed on $e/pq = 2621/8927$ in Table I. The continued fraction attack on RSA for this example yields

$$d=5, \quad p=113, \quad q=79, \quad k=3, \quad \text{and} \quad g=2.$$

One can verify that (27) is satisfied for these values to see that $d=5$ is the secret exponent corresponding to $e=2621$. One can also verify that the sufficient condition for the success of this algorithm (29) is satisfied.

This example illustrates the details of the continued fraction attack on RSA, but it is useful to consider a more realistic case. Suppose that a 1024-bit modulus is used for RSA. Then p and q are approximately 2^{512}. Suppose that $g=2$, and that $e \approx pq$ so that $k \approx dg$ (see (28)). Then

TABLE I

Calculated Quantity	How it is Derived	$i=0$	$i=1$	$i=2$
q_i'	See (3)	0	3	2
r_i'	See (3)	$\frac{2621}{8927}$	$\frac{1064}{2621}$	$\frac{493}{1064}$
$\frac{n_i'}{d_i'}$	See (6)	$\frac{0}{1}$	$\frac{1}{3}$	$\frac{2}{7}$
$=\langle q_0', q_1', \cdots, q_i' \rangle$				
Guess of k/dg	$\langle q_0', q_1', \cdots, q_{i-1}', q_i'+1 \rangle$ (i even) $\langle q_0', q_1', \cdots, q_i' \rangle$ (i odd)	$\frac{1}{1}$	$\frac{1}{3}$	$\frac{3}{10}$
Guess of edg	$e \cdot dg$	2621	7863	26210
Guess of $(p-1)(q-1)$	$\lfloor edg/k \rfloor$	2621	7863	8736
Guess of g	$edg \bmod k$	0	0	2
Guess of $(p+q)/2$	See (31)	3153.5 (quit)	532.5 (quit)	96
Guess of $((p-q)/2)^2$	See (32)			$289 = 17^2$
d	dg/g			5

using (29), we see that the continued fraction attack will find secret exponents up to a size of approximately 2^{255}.

VI. Combatting the Continued Fraction Attack on RSA

There are two ways of reducing the maximum size of secret exponent that can be found using the continued fraction attack on RSA. From (29), we can see that these are to make k larger and to make g larger.

To make k larger, one must make the public exponent e larger (see (27)). This can be done by adding a multiple of $LCM(p-1, q-1)$ to e. Suppose that $e > (pq)^{1.5}$. This implies that $k/dg > (pq)^{0.5}$ (see (28)). Substituting $k = dg(pq)^{0.5}$ into (29) leads to $d < 1$. Therefore, if $e > (pq)^{1.5}$, the continued fraction algorithm is not guaranteed to work for any size of secret exponent. Increasing the size of e has the disadvantage that it increases the execution time of public key encryption. But this may be acceptable in some systems.

To make g larger, p and q must be chosen such that $GCD(p-1, q-1)$ is large. However, we will see later that there are ways to find g or factors of g under certain conditions.

VII. Improvements to the Attack on RSA

In this section, four possible improvements to the attack on short secret exponents will be discussed. The first improvement is to allow the continued fraction algorithm to continue searching for d slightly beyond the limit of (29). The algorithm is only guaranteed to work up to this limit, but it may work slightly beyond the limit. This may add a bit or so to the size of secret exponent that can be found.

The second improvement is based upon the observation that the denominator of e/pq (which is the underestimate of k/dg) is simply an overestimate of $(p-1)(q-1)$. A closer estimate of $(p-1)(q-1)$ is

$$\left\lfloor \left(\sqrt{pq} - 1\right)^2 \right\rfloor.$$

Using this estimate, (29) becomes

$$kdg < \frac{2}{3}\left(\frac{\sqrt{pq} - 1}{\sqrt{p} - \sqrt{q}}\right)^2.$$

This increases the size of secret exponents that can be found. The amount of improvement increases as $|p-q|$ decreases.

The third improvement to the continued fraction attack on RSA is to perform the algorithm on many guesses of k/dg. One might start at some initial guess and then try successively larger guesses. In this way, one would be performing a linear search for k/dg. For secret exponents up to the limit of (29), the algorithm takes polynomial time. As the secret exponent increases in size beyond this limit, the number of times that the algorithm must be performed increases exponentially.

The fourth improvement is to attempt to find g or factors of g. Suppose that t is known to be a factor of g. Then one could use

$$t\left(\frac{e}{pq}\right) \text{ as an underestimate of } \frac{k}{d\left(\frac{g}{t}\right)}.$$

In this case (29) becomes

$$kd\left(\frac{g}{t}\right) < \frac{pq}{\frac{3}{2}(p+q)}.$$

This increases the size of d that can be found by a factor of t. We now need a way to find factors of g. Because g divides $GCD(p-1, q-1)$, g divides both $p-1$ and $q-1$. This means that g also divides $pq-1$ because

$$pq - 1 = (p-1)(q-1) + (p-1) + (q-1).$$

One may be able to find factors of g by factoring $pq-1$. If g is chosen to be large and all of the prime factors of g are large, then it may be difficult to find factors of g by factoring $pq-1$. However, if g is so large that $(p-1)/g$ and $(q-1)/g$ are small, then one could find g by searching through possible values of $(p-1)/g$ and $(q-1)/g$.

VIII. Open Problems

The main motivation for using short secret exponents is to reduce the secret key exponentiation time. A useful technique for reducing the secret key exponentiation time is to take advantage of the knowledge of p and q (rather than just the product pq) [4]. Using this technique, two half-sized exponentiations are performed. The first exponentiation gives the result modulo p using exponent $d_p = d \bmod (p-1)$, and the second gives the result modulo q using exponent $d_q = d \bmod (q-1)$. These two results can be combined easily using the Chinese remainder theorem to obtain the final result modulo pq. One could reduce the secret key exponentiation time further by choosing d so that d_p and d_q are short. An interesting open problem is whether there is an attack on RSA when d_p and d_q are short, but not equal.

There is another open problem related to the size of the public exponent. Recall that the attack described in this paper is defeated if the public exponent is chosen to be at least 50% longer than the modulus pq. For some systems, this may be a small price to pay in order to have fast secret key exponentiations. An interesting question is whether there is an attack on RSA when the secret exponent is short, and the public exponent is larger than the modulus.

IX. Conclusion

The continued fraction algorithm can be used to find sufficiently short RSA secret exponents in polynomial time. For a typical case where $e < pq$, $GCD(p-1, q-1)$ is small, and p and q have approximately the same

number of bits, this algorithm will find secret exponents with up to approximately one-quarter as many bits as the modulus.

There are ways to combat the continued fraction attack on RSA. If $e > (pq)^{1.5}$, then the continued fraction algorithm is not guaranteed to work for any size of secret exponent. Also, one might choose $GCD(p-1, q-1)$ to be large because the size of secret exponent that can be found is inversely proportional to $GCD(p-1, q-1)$. However, choosing $GCD(p-1, q-1)$ to be large may cause other problems.

A number of improvements to the continued fraction attack on RSA were discussed. However, they only add a few more bits to the maximum size of secret exponent that can be found in polynomial time. As the secret exponent increases in size beyond this maximum, the time required to find the secret exponent increases exponentially. This attack cannot be extended to the normal case where the secret exponent is approximately the same size as the modulus.

REFERENCES

[1] J. Hastad, "On using RSA with low exponent in a public key network," *Lecture Notes in Computer Science: Advances in Cryptology —CRYPTO '85 Proceedings*. New York: Springer-Verlag, pp. 403–408.

[2] D. E. Knuth, *Art of Computer Programming Vol. 2 / Seminumerical algorithms*. New York: Addison Wesley, 1969.

[3] J. M. Pollard, "Theorems on factorization and primality testing," *Proc. Cambridge Philos. Soc.*, vol. 76, 1974, pp. 521–528.

[4] J. J. Quisquater and C. Couvreur, "Fast decipherment algorithm for RSA public-key cryptosystem," *Electron. Lett.*, vol. 18, no. 21, pp. 905–907, Oct. 1982.

[5] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 158–164, Feb. 1978.

[6] H. C. Williams, "A $p+1$ method of factoring," *Mathematics of Computation*, vol. 39, no. 159, pp. 225–234, July 1982.