numbers in the new RSA challenge: RSA-576, RSA-640, RSA-704, RSA-768, RSA-896, RSA-1024, RSA-1536 and RSA-2048. There are prizes for factoring these numbers, ranging from $10,000 to $200,000. The factorization of RSA-576 was announced in December, 2003 by Jens Franke; the factorization was accomplished using the GENERAL NUMBER FIELD SIEVE.

Extrapolating current trends in factoring into the future, it has been suggested that 768-bit moduli may be factored by 2010, and 1024-bit moduli may be factored by the year 2018.

5.7 Other Attacks on RSA

In this section, we address the following question: are there possible attacks on the *RSA Cryptosystem* other than factoring n? For example, it is at least conceivable that there could exist a method of decrypting RSA ciphertexts that does not involve finding the factorization of the modulus n.

5.7.1 Computing $\phi(n)$

We first observe that computing $\phi(n)$ is no easier than factoring n. For, if n and $\phi(n)$ are known, and n is the product of two primes p, q, then n can be easily factored, by solving the two equations

$$n = pq$$
$$\phi(n) = (p-1)(q-1)$$

for the two "unknowns" p and q. This is easily accomplished, as follows. If we substitute $q = n/p$ into the second equation, we obtain a quadratic equation in the unknown value p:

$$p^2 - (n - \phi(n) + 1)p + n = 0. \tag{5.1}$$

The two roots of equation (5.1) will be p and q, the factors of n. Hence, if a cryptanalyst can learn the value of $\phi(n)$, then he can factor n and break the system. In other words, computing $\phi(n)$ is no easier than factoring n.

Here is an example to illustrate.

Example 5.13 Suppose $n = 84773093$, and the adversary has learned that $\phi(n) = 84754668$. This information gives rise to the following quadratic equation:

$$p^2 - 18426p + 84773093 = 0.$$

This can be solved by the quadratic formula, yielding the two roots 9539 and 8887. These are the two factors of n. ∎

5.7.2 The Decryption Exponent

We will now prove the very interesting result that, if the decryption exponent a is known, then n can be factored in polynomial time by means of a randomized algorithm. Therefore we can say that computing a is (essentially) no easier than factoring n. (However, this does not rule out the possibility of breaking the *RSA Cryptosystem* without computing a.) Notice that this result is of much more than theoretical interest. It tells us that if a is revealed (accidently or otherwise), then it is not sufficient for Bob to choose a new encryption exponent; he must also choose a new modulus n.

The algorithm we are going to describe is a randomized algorithm of the Las Vegas type (see Section 4.2.2 for the definition). Here, we consider Las Vegas algorithms having worst-case success probability at least $1 - \epsilon$. Therefore, for any problem instance, the algorithm may fail to give an answer with probability at most ϵ.

If we have such a Las Vegas algorithm, then we simply run the algorithm over and over again until it finds an answer. The probability that the algorithm will return "no answer" m times in succession is ϵ^m. The average (i.e., expected) number of times the algorithm must be run in order to obtain an answer is $1/(1-\epsilon)$ (see the Exercises).

We will describe a Las Vegas algorithm that will factor n with probability at least $1/2$ when given the values a, b and n as input. Hence, if the algorithm is run m times, then n will be factored with probability at least $1 - 1/2^m$.

The algorithm is based on certain facts concerning square roots of 1 modulo n, where $n = pq$ is the product of two distinct odd primes. $x^2 \equiv 1 \pmod{p}$ and Theorem 5.13 asserts that there are four square roots of 1 modulo n. Two of these square roots are $\pm 1 \bmod n$; these are called the *trivial* square roots of 1 modulo n. The other two square roots are called *non-trivial*; they are also negatives of each other modulo n.

Here is a small example to illustrate.

Example 5.14 Suppose $n = 403 = 13 \times 31$. The four square roots of 1 modulo 403 are $1, 92, 311$ and 402. The square root 92 is obtained by solving the system

$$\begin{aligned} x &\equiv 1 \pmod{13}, \\ x &\equiv -1 \pmod{31}. \end{aligned}$$

using the Chinese remainder theorem. The other non-trivial square root is $403 - 92 = 311$. It is the solution to the system

$$\begin{aligned} x &\equiv -1 \pmod{13}, \\ x &\equiv 1 \pmod{31}. \end{aligned}$$

∎

Suppose x is a non-trivial square root of 1 modulo n. Then

$$x^2 \equiv 1^2 \pmod{n}$$

but

$$x \not\equiv \pm 1 \pmod{n}.$$

Then, as in the Random squares factoring algorithm, we can find the factors of n by computing $\gcd(x+1, n)$ and $\gcd(x-1, n)$. In Example 5.14 above,

$$\gcd(93, 403) = 31$$

and

$$\gcd(312, 403) = 13.$$

Algorithm 5.10 attempts to factor n by finding a non-trivial square root of 1 modulo n. Before analyzing the algorithm, we first do an example to illustrate its application.

Example 5.15 Suppose $n = 89855713$, $b = 34986517$ and $a = 82330933$, and the random value $w = 5$. We have

$$ab - 1 = 2^3 \times 360059073378795.$$

Then

$$w^r \bmod n = 85877701.$$

It happens that

$$85877701^2 \equiv 1 \pmod{n}.$$

Therefore the algorithm will return the value

$$x = \gcd(85877702, n) = 9103.$$

This is one factor of n; the other is $n/9103 = 9871$. □

Let's now proceed to the analysis of Algorithm 5.10. First, observe that if we are lucky enough to choose w to be a multiple of p or q, then we can factor n immediately. If w is relatively prime to n, then we compute $w^r, w^{2r}, w^{4r}, \ldots$, by successive squaring, until

$$w^{2^t r} \equiv 1 \pmod{n}$$

for some t. Since

$$ab - 1 = 2^s r \equiv 0 \pmod{\phi(n)},$$

we know that $w^{2^s r} \equiv 1 \pmod{n}$. Hence, the **while** loop terminates after at most s iterations. At the end of the **while** loop, we have found a value v_0 such

Algorithm 5.10: RSA-FACTOR(n, a, b)

comment: we are assuming that $ab \equiv 1 \pmod{\phi(n)}$

write $ab - 1 = 2^s r, r$ odd
choose w at random such that $1 \leq w \leq n-1$
$x \leftarrow \gcd(w, n)$
if $1 < x < n$
  then return (x)
comment: x is a factor of n

$v \leftarrow w^r \bmod n$
if $v \equiv 1 \pmod{n}$
  then return ("failure")
while $v \not\equiv 1 \pmod{n}$
  do $\begin{cases} v_0 \leftarrow v \\ v \leftarrow v^2 \bmod n \end{cases}$
if $v_0 \equiv -1 \pmod{n}$
  then return ("failure")
  else $\begin{cases} x \leftarrow \gcd(v_0 + 1, n) \\ \textbf{return } (x) \end{cases}$
comment: x is a factor of n

that $(v_0)^2 \equiv 1 \pmod{n}$ but $v_0 \not\equiv 1 \pmod{n}$. If $v_0 \equiv -1 \pmod{n}$, then the algorithm fails; otherwise, v_0 is a non-trivial square root of 1 modulo n and we are able to factor n.

The main task facing us now is to prove that the algorithm succeeds with probability at least $1/2$. There are two ways in which the algorithm can fail to factor n:

1. $w^r \equiv 1 \pmod{n}$, or
2. $w^{2^t r} \equiv -1 \pmod{n}$ for some t, $0 \leq t \leq s-1$.

We have $s+1$ congruences to consider. If a random value w is a solution to at least one of these $s+1$ congruences, then it is a "bad" choice, and the algorithm fails. So we proceed by counting the number of solutions to each of these congruences.

First, consider the congruence $w^r \equiv 1 \pmod{n}$. The way to analyze a congruence such as this is to consider solutions modulo p and modulo q separately, and then combine them using the Chinese remainder theorem. Observe that $x \equiv 1 \pmod{n}$ if and only if $x \equiv 1 \pmod{p}$ and $x \equiv 1 \pmod{q}$.

So, we first consider $w^r \equiv 1 \pmod{p}$. Since p is prime, ${\mathbb{Z}_p}^*$ is a cyclic group by Theorem 5.7. Let g be a primitive element modulo p. We can write $w = g^u$

for a unique integer u, $0 \leq u \leq p-2$. Then we have

$$w^r \equiv 1 \pmod{p},$$
$$g^{ur} \equiv 1 \pmod{p}, \text{ and hence}$$
$$(p-1) \mid ur.$$

Let us write

$$p - 1 = 2^i p_1$$

where p_1 is odd, and

$$q - 1 = 2^j q_1$$

where q_1 is odd. Since

$$\phi(n) = (p-1)(q-1) \mid (ab-1) = 2^s r,$$

we have that

$$2^{i+j} p_1 q_1 \mid 2^s r.$$

Hence

$$i + j \leq s$$

and

$$p_1 q_1 \mid r.$$

Now, the condition $(p-1) \mid ur$ becomes $2^i p_1 \mid ur$. Since $p_1 \mid r$ and r is odd, it is necessary and sufficient that $2^i \mid u$. Hence, $u = k2^i$, $0 \leq k \leq p_1 - 1$, and the number of solutions to the congruence $w^r \equiv 1 \pmod{p}$ is p_1.

By an identical argument, the congruence $w^r \equiv 1 \pmod{q}$ has exactly q_1 solutions. We can combine any solution modulo p with any solution modulo q to obtain a unique solution modulo n, using the Chinese remainder theorem. Consequently, the number of solutions to the congruence $w^r \equiv 1 \pmod{n}$ is $p_1 q_1$.

The next step is to consider a congruence $w^{2^t r} \equiv -1 \pmod{n}$ for a fixed value t (where $0 \leq t \leq s-1$). Again, we first look at the congruence modulo p and then modulo q (note that $w^{2^t r} \equiv -1 \pmod{n}$ if and only if $w^{2^t r} \equiv -1 \pmod{p}$ and $w^{2^t r} \equiv -1 \pmod{q}$). First, consider $w^{2^t r} \equiv -1 \pmod{p}$. Writing $w = g^u$, as above, we get

$$g^{u2^t r} \equiv -1 \pmod{p}.$$

Since $g^{(p-1)/2} \equiv -1 \pmod{p}$, we have that

$$u2^t r \equiv \frac{p-1}{2} \pmod{p-1}$$
$$(p-1) \mid \left(u2^t r - \frac{p-1}{2} \right)$$
$$2(p-1) \mid (u2^{t+1} r - (p-1)).$$

Since $p-1=2^i p_1$, we get

$$2^{i+1}p_1 \mid (u2^{t+1}r - 2^i p_1).$$

Taking out a common factor of p_1, this becomes

$$2^{i+1} \mid \left(\frac{u2^{t+1}r}{p_1} - 2^i\right).$$

Now, if $t \geq i$, then there can be no solutions since $2^{i+1} \mid 2^{t+1}$ but $2^{i+1} \nmid 2^i$. On the other hand, if $t \leq i-1$, then u is a solution if and only if u is an odd multiple of 2^{i-t-1} (note that r/p_1 is an odd integer). So, the number of solutions in this case is

$$\frac{p-1}{2^{i-t-1}} \times \frac{1}{2} = 2^t p_1.$$

By similar reasoning, the congruence $w^{2^t r} \equiv -1 \pmod{q}$ has no solutions if $t \geq j$, and $2^t q_1$ solutions if $t \leq j-1$. Using the Chinese remainder theorem, we see that the number of solutions of $w^{2^t r} \equiv -1 \pmod{n}$ is

$$\begin{array}{rl} 0 & \text{if } t \geq \min\{i,j\} \\ 2^{2t}p_1q_1 & \text{if } t \leq \min\{i,j\}-1. \end{array}$$

Now, t can range from 0 to $s-1$. Without loss of generality, suppose $i \leq j$; then the number of solutions is 0 if $t \geq i$. The total number of "bad" choices for w is at most

$$\begin{aligned} p_1q_1 + p_1q_1(1+2^2+2^4+\cdots+2^{2i-2}) &= p_1q_1\left(1+\frac{2^{2i}-1}{3}\right) \\ &= p_1q_1\left(\frac{2}{3}+\frac{2^{2i}}{3}\right). \end{aligned}$$

Recall that $p-1=2^i p_1$ and $q-1=2^j q_1$. Now, $j \geq i \geq 1$, so $p_1q_1 < n/4$. We also have that

$$2^{2i}p_1q_1 \leq 2^{i+j}p_1q_1 = (p-1)(q-1) < n.$$

Hence, we obtain

$$\begin{aligned} p_1q_1\left(\frac{2}{3}+\frac{2^{2i}}{3}\right) &< \frac{n}{6}+\frac{n}{3} \\ &= \frac{n}{2}. \end{aligned}$$

Since at most $(n-1)/2$ choices for w are "bad," it follows that at least $(n-1)/2$ choices are "good" and hence the probability of success of the algorithm is at least $1/2$.

5.7.3 Wiener's Low Decryption Exponent Attack

As always, suppose that $n = pq$ where p and q are prime; then $\phi(n) = (p-1)(q-1)$. In this section, we present an attack, due to M. Wiener, that succeeds in computing the secret decryption exponent, a, whenever the following hypotheses are satisfied:

$$3a < n^{1/4} \quad \text{and} \quad q < p < 2q. \tag{5.2}$$

If n has ℓ bits in its binary representation, then the attack will work when a has fewer than $\ell/4 - 1$ bits in its binary representation and p and q are not too far apart.

Note that Bob might be tempted to choose his decryption exponent to be small in order to speed up decryption. If he uses Algorithm 5.5 to compute $y^a \bmod n$, then the running time of decryption will be reduced by roughly 75% if he chooses a value of a that satisfies (5.2). The results we prove in this section show that this method of reducing decryption time should be avoided.

Since $ab \equiv 1 \pmod{\phi(n)}$, it follows that there is an integer t such that

$$ab - t\phi(n) = 1.$$

Since $n = pq > q^2$, we have that $q < \sqrt{n}$. Hence,

$$0 < n - \phi(n) = p + q - 1 < 2q + q - 1 < 3q < 3\sqrt{n}.$$

Now, we see that

$$\begin{aligned}\left|\frac{b}{n} - \frac{t}{a}\right| &= \left|\frac{ba - tn}{an}\right| \\ &= \left|\frac{1 + t(\phi(n) - n)}{an}\right| \\ &< \frac{3t\sqrt{n}}{an} \\ &= \frac{3t}{a\sqrt{n}}.\end{aligned}$$

Since $t < a$, we have that $3t < 3a < n^{1/4}$, and hence

$$\left|\frac{b}{n} - \frac{t}{a}\right| < \frac{1}{an^{1/4}}.$$

Finally, since $3a < n^{1/4}$, we have that

$$\left|\frac{b}{n} - \frac{t}{a}\right| < \frac{1}{3a^2}.$$

Therefore the fraction t/a is a very close approximation to the fraction b/n. From the theory of continued fractions, it is known that any approximation of b/n that

is this close must be one of the convergents of the continued fraction expansion of b/n (see Theorem 5.14). This expansion can be obtained from the EUCLIDEAN ALGORITHM, as we describe now.

A (finite) *continued fraction* is an m-tuple of non-negative integers, say

$$[q_1, \ldots, q_m],$$

which is shorthand for the following expression:

$$q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cdots + \cfrac{1}{q_m}}}.$$

Suppose a and b are positive integers such that $\gcd(a, b) = 1$, and suppose that the output of Algorithm 5.1 is the m-tuple $(q_1, \ldots, q_m)$. Then it is not hard to see that $a/b = [q_1, \ldots, q_m]$. We say that $[q_1, \ldots, q_m]$ is the *continued fraction expansion* of a/b in this case. Now, for $1 \leq j \leq m$, define $C_j = [q_1, \ldots, q_j]$. C_j is said to be the jth *convergent* of $[q_1, \ldots, q_m]$. Each C_j can be written as a rational number c_j/d_j, where the c_j's and d_j's satisfy the following recurrences:

$$c_j = \begin{cases} 1 & \text{if } j = 0 \\ q_1 & \text{if } j = 1 \\ q_j c_{j-1} + c_{j-2} & \text{if } j \geq 2 \end{cases}$$

and

$$d_j = \begin{cases} 0 & \text{if } j = 0 \\ 1 & \text{if } j = 1 \\ q_j d_{j-1} + d_{j-2} & \text{if } j \geq 2. \end{cases}$$

Example 5.16 We compute the continued fraction expansion of $34/99$. The EUCLIDEAN ALGORITHM proceeds as follows:

$$\begin{aligned} 34 &= 0 \times 99 + 34 \\ 99 &= 2 \times 34 + 31 \\ 34 &= 1 \times 31 + 3 \\ 31 &= 10 \times 3 + 1 \\ 3 &= 3 \times 1. \end{aligned}$$

Hence, the continued fraction expansion of $34/99$ is $[0, 2, 1, 10, 3]$, i.e.,

$$\frac{34}{99} = 0 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{10 + \cfrac{1}{3}}}}.$$

The convergents of this continued fraction are as follows:

$$\begin{aligned}[0] &= 0\\ [0,2] &= 1/2\\ [0,2,1] &= 1/3\\ [0,2,1,10] &= 11/32, \quad \text{and}\\ [0,2,1,10,3] &= 34/99.\end{aligned}$$

The reader can verify that these convergents can be computed using the recurrence relations given above. □

The convergents of a continued fraction expansion of a rational number satisfy many interesting properties. For our purposes, the most important property is the following.

THEOREM 5.14 *Suppose that* $\gcd(a,b) = \gcd(c,d) = 1$ *and*

$$\left|\frac{a}{b} - \frac{c}{d}\right| < \frac{1}{2d^2}.$$

Then c/d *is one of the convergents of the continued fraction expansion of* a/b.

Now we can apply this result to the *RSA Cryptosystem.* We already observed that, if condition (5.2) holds, then the unknown fraction t/a is a close approximation to b/n. Theorem 5.14 tells us that t/a must be one of the convergents of the continued fraction expansion of b/n. Since the value of b/n is public information, it is a simple matter to compute its convergents. All we need is a method to test each convergent to see if it is the "right" one.

But this is also not difficult to do. If t/a is a convergent of b/n, then we can compute the value of $\phi(n)$ to be $\phi(n) = (ab-1)/t$. Once n and $\phi(n)$ are known, we can factor n by solving the quadratic equation (5.1) for p. We do not know ahead of time which convergent of b/n will yield the factorization of n, so we try each one in turn, until the factorization of n is found. If we do not succeed in factoring n by this method, then it must be the case that the hypotheses (5.2) are not satisfied.

A pseudocode description of WIENER'S ALGORITHM is presented as Algorithm 5.11.

We present an example to illustrate.

Example 5.17 Suppose that $n = 160523347$ and $b = 60728973$. The continued fraction expansion of b/n is

$$[0,2,1,1,1,4,12,102,1,1,2,3,2,2,36].$$

Algorithm 5.11: WIENER'S ALGORITHM(n, b)

$(q_1, \ldots, q_m; r_m) \leftarrow$ EUCLIDEAN ALGORITHM(b, n)
$c_0 \leftarrow 1$
$c_1 \leftarrow q_1$
$d_0 \leftarrow 0$
$d_1 \leftarrow 1$
for $j \leftarrow 2$ **to** m
do
- $c_j \leftarrow q_j c_{j-1} + c_{j-2}$
- $d_j \leftarrow q_j d_{j-1} + d_{j-2}$
- $n' \leftarrow (d_j b - 1)/c_j$
- **comment:** $n' = \phi(n)$ if c_j/d_j is the correct convergent
- **if** n' is an integer
  - **then**
    - let p and q be the roots of the equation $x^2 - (n - n' + 1)x + n = 0$
    - **if** p and q are positive integers less than n
      - **then return** (p, q)

return ("failure")

The first few convergents are

$$0, \frac{1}{2}, \frac{1}{3}, \frac{2}{5}, \frac{3}{8}, \frac{14}{37}.$$

The reader can verify that the first five convergents do not produce a factorization of n. However, the convergent $14/37$ yields

$$n' = \frac{37 \times 60728973 - 1}{14} = 160498000.$$

Now, if we solve the equation

$$x^2 - 25348x + 160523347 = 0,$$

then we find the roots $x = 12347, 13001$. Therefore we have discovered the factorization

$$160523347 = 12347 \times 13001.$$

Notice that, for the modulus $n = 160523347$, WIENER'S ALGORITHM will work for

$$a < \frac{n^{1/4}}{3} \approx 37.52.$$

∎

Cryptosystem 5.2: *Rabin Cryptosystem*

Let $n = pq$, where p and q are primes and $p, q \equiv 3 \pmod 4$. Let $\mathcal{P} = \mathcal{C} = {\mathbb{Z}_n}^*$, and define

$$\mathcal{K} = \{(n, p, q)\}.$$

For $K = (n, p, q)$, define

$$e_K(x) = x^2 \bmod n$$

and

$$d_K(y) = \sqrt{y} \bmod n.$$

The value n is the public key, while p and q are the private key.

5.8 The Rabin Cryptosystem

In this section, we describe the *Rabin Cryptosystem*, which is computationally secure against a chosen-plaintext attack provided that the modulus $n = pq$ cannot be factored. Therefore, the *Rabin Cryptosystem* provides an example of a provably secure cryptosystem: assuming that the problem of factoring is computationally infeasible, the *Rabin Cryptosystem* is secure. We present the *Rabin Cryptosystem* as Cryptosystem 5.2.

REMARK The requirement that $p, q \equiv 3 \pmod 4$ can be omitted. As well, the cryptosystem still "works" if we take $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$ instead of ${\mathbb{Z}_n}^*$. However, the more restrictive description we use simplifies some aspects of computation and analysis of the cryptosystem. ∎

One drawback of the *Rabin Cryptosystem* is that the encryption function e_K is not an injection, so decryption cannot be done in an unambiguous fashion. We prove this as follows. Suppose that y is a valid ciphertext; this means that $y = x^2 \bmod n$ for some $x \in {\mathbb{Z}_n}^*$. Theorem 5.13 proves that there are four square roots of y modulo n, which are the four possible plaintexts that encrypt to y. In general, there will be no way for Bob to distinguish which of these four possible plaintexts is the "right" plaintext, unless the plaintext contains sufficient redundancy to eliminate three of these four possible values.

Let us look at the decryption problem from Bob's point of view. He is given a ciphertext y and wants to determine x such that

$$x^2 \equiv y \pmod n.$$