COMP 8920: Cryptography	Winter 2025

Lecture 22 — March 27, 2025

Prof. Curtis Bright *Scribe: Rajat Yadav*

RSA and Continued Fractions

Next, we will show that if the decryption exponent a is revealed, the n can be factored in expected polynomial time. So, if a is leaked, a new modulus n must be chosen in addition to a new (a, b) pair. The algorithm is of Las Vegas type, meaning that it may fail to work (output fail) with probability ε. It can therefore be shown that you expect to run the algorithm $1/(1-\varepsilon)$ times until a success. The algorithm is somewhat similar to the Miller-Rabin primality test in that a random base $w \in \mathbb{Z}_n^*$ is chosen and exponentiated in $\mathbb{Z}_n$. Recall that we write $n - 1 = 2^k \cdot m$ (with m odd) and compute $w^m, w^{2m}, w^{4m}, \ldots, w^{2^k m}$ all in $\mathbb{Z}_n^*$.

This algorithm will do something similar except based on writing $ab - 1 = 2^s \cdot r$ (where (a, b) is the decryption/encryption exponent pair).

Algorithm

Input: a, b, n where $ab \equiv 1 \mod (\phi(n))$

1. Write $ab - 1 = 2^s \cdot r$ for $s, r \in \mathbb{Z}, r$ odd.
2. Choose a random w in $[2, n-1]$
3. If $\gcd(w, n) > 1$: return $\gcd(w, n)$ (either p or q)
4. $v := w^r \bmod n$
5. If $v = 1$: return failure
6. For i from 1 to s:
   (a) $v_{\text{prev}} := v$
   (b) $v := v^2 \bmod n$
   (c) If $v = -1$: return failure
   (d) If $v = 1$: return $\gcd(v_{\text{prev}} + 1, n)$

In the final return, v_{prev} is a non-trivial square root of 1. We must eventually reach a return because $w^{ab-1} \equiv 1 \bmod n$ as $ab = 1 + k\phi(n)$ for $k \in \mathbb{Z}$ and $w^{\phi(n)} \equiv 1 \bmod n$.

We already saw $\gcd(v_{\text{prev}} \pm 1, n)$ will be a non-trivial factor (p or q) in the previous class. However, the algorithm fails when either:

1. $w^r \equiv 1 \bmod n$ (this doesn't help find a root of 1)
2. $w^{2^i r} \equiv -1 \bmod n$ (as this only finds a trivial root of 1)

One can show there are at most $n/4$ values of w in case (1) and at most $n/4$ values of w in case (2), so there are at most $n/2$ values of w which cause failure, with probability at most $1/2$.

Small Decryption Exponents

Can we take the decryption exponent to be small? This would be nice in order to speed up decryption, but to be secure we'll need a to be at least $3n^{1/4}$ as we'll show. We'll show that n can be factored in polynomial time when:

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

The second inequality says that if n has l bits then p and q each have $l/2$ bits (± 1 bit) which is typical, and the first inequality says that a has at most $l/4 - 1$ bits.

So for RSA to be secure, we always ensure $3a > n^{1/4}$, even though this increases the cost of decryption slightly.

The attack is based on computing an approximation to the fraction b/n (a publicly known quantity) that has a smaller denominator than n. Since $ab \equiv 1 \bmod \phi(n)$ or $ab = 1 + t \cdot \phi(n)$ for $t \in \mathbb{Z}$.

Since $n = pq > q^2$ so $q < \sqrt{n}$, and $0 < n - \phi(n) = p + q - 1 < 2q + q - 1 = 3q - 1 < 3\sqrt{n}$.

$$\left|\frac{b}{n} - \frac{t}{a}\right| = \left|\frac{ba - tn}{an}\right| = \left|\frac{1 + t\phi(n) - tn}{an}\right| = \left|\frac{t(n - \phi(n)) - 1}{an}\right| \tag{2}$$

$$< \frac{3\sqrt{n}t}{an} = \frac{3t}{a\sqrt{n}} \tag{3}$$

Note $t = \frac{ab-1}{\phi(n)} < \frac{ab}{\phi(n)} < a < \frac{n^{1/4}}{3}$ so the above is

$$< \frac{n^{1/4}}{a\sqrt{n}} = \frac{1}{an^{1/4}} \tag{4}$$

and $\frac{1}{n^{1/4}} < \frac{1}{3a}$ by assumption.

The final bound is $\left|\frac{b}{n} - \frac{t}{a}\right| < \frac{1}{3a^2}$. Since $\frac{1}{3a^2}$ is very small, this means $\frac{t}{a}$ is a very good approximation to $\frac{b}{n}$. In fact, $\frac{t}{a}$ can be computed directly from $\frac{b}{n}$ by the following:

Theorem 1. *If $\frac{a}{b}$ and $\frac{c}{d}$ are in lowest terms and $\left|\frac{a}{b} - \frac{c}{d}\right| < \frac{1}{2d^2}$, then $\frac{c}{d}$ is a convergent in the continued fraction (CF) expansion of $\frac{a}{b}$.*

A continued fraction is of the form $q_1 + \frac{1}{q_2 + \frac{1}{q_3 + \frac{1}{q_4 + \dots}}}$ when the q_i's are positive integers. In fact, the CF expansion of $\frac{a}{b}$ has a surprising connection to the Euclidean algorithm. When running the Euclidean algorithm on (a, b), the quotients produced are exactly the q_i in the CF expansion of $\frac{a}{b}$.