Introduction to Cryptography: RSA

Introduction to Cryptography: RSA: Steven J. Miller

http://www.williams.edu/Mathematics/sjmiller/public_html

VCTAL, Burlington, June 20

RSA Description
(Rivest, Shamir, and Adleman)

Set-up: Example

Alice always sends to Bob, Charlie or Eve tries to intercept.

Bob does the following (could have b subscripts):

- Secret: $p = 15217$, $q = 17569$, $d = 80998505$.

Set-up: Example

Alice always sends to Bob, Charlie or Eve tries to intercept.

Bob does the following (could have b subscripts):

- Secret: $p = 15217$, $q = 17569$, $d = 80998505$.
- Public: $N = pq = 267347473$, $e = 3141593$.

Set-up: Example

Alice always sends to Bob, Charlie or Eve tries to intercept.

Bob does the following (could have *b* subscripts):

- Secret: $p = 15217$, $q = 17569$, $d = 80998505$.
- Public: $N = pq = 267347473$, $e = 3141593$.
- Note: $ed = 1 \bmod (p-1)(q-1)$.

Set-up: Example

Alice always sends to Bob, Charlie or Eve tries to intercept.

Bob does the following (could have b subscripts):

- Secret: $p = 15217$, $q = 17569$, $d = 80998505$.
- Public: $N = pq = 267347473$, $e = 3141593$.
- Note: $ed = 1 \bmod (p-1)(q-1)$.
- **Message:** $M = 195632041$**, send** $M^e \bmod N$ **or** $X = 121209473$**.**

Set-up: Example

Alice always sends to Bob, Charlie or Eve tries to intercept.

Bob does the following (could have b subscripts):

- Secret: $p = 15217$, $q = 17569$, $d = 80998505$.
- Public: $N = pq = 267347473$, $e = 3141593$.
- Note: $ed = 1 \bmod (p-1)(q-1)$.
- **Message:** $M = 195632041$**, send** $M^e \bmod N$ **or** $X = 121209473$**.**
- Decrypt: $X^d \bmod N$ or 195632041.

Set-up: Example

Alice always sends to Bob, Charlie or Eve tries to intercept.

Bob does the following (could have *b* subscripts):

- Secret: $p = 15217$, $q = 17569$, $d = 80998505$.
- Public: $N = pq = 267347473$, $e = 3141593$.
- Note: $ed = 1 \bmod (p-1)(q-1)$.
- **Message:** $M = 195632041$**, send** $M^e \bmod N$ **or** $X = 121209473$**.**
- Decrypt: $X^d \bmod N$ or 195632041.

Imagine receive $\widetilde{X} = 121209483$.
Message 195632041
Decrypts 121141028, only two digits are the same!

Implementation Questions

A lot of implementation issues.

- How do we find large primes? How large is large?
- How do we find e and d so that $ed = 1 \bmod (p-1)(q-1)$?
- How do we compute $M^e \bmod N$ efficiently?
- Can Eve determine d from e and N?

Fermat's little Theorem

Euler totient function

$\phi(n)$ is the number of integers from 1 to n relatively prime to n.

$\phi(p) = p - 1$ and $\phi(pq) = (p-1)(q-1)$ if p, q distinct primes.

Do not need, but $\phi(mn) = \phi(m)\phi(n)$ if $\gcd(m, n) = 1$, and $\phi(p^k) = p^k - p^{k-1}$.

A lot of group theory lurking in the background, only doing what absolutely need.

Fermat's little Theorem

Fermat's little Theorem (FlT)

Let a be relatively prime to n. Then $a^{\phi(n)} = 1 \bmod n$.

Special cases: $a^{p-1} = 1 \bmod p$, $a^{(p-1)(q-1)} = 1 \bmod pq$.

Will only prove these two cases....

Proof of Fermat's little Theorem: $n = p$

Proof: Let $n = p$, let $\gcd(a, p) = 1$.

Consider $1, 2, \ldots, p - 1$ and $a, 2a, \ldots, (p - 1)a$.

Claim both sets are all residues modulo p.

Proof of Fermat's little Theorem: $n = p$

Proof: Let $n = p$, let $\gcd(a, p) = 1$.

Consider $1, 2, \ldots, p - 1$ and $a, 2a, \ldots, (p - 1)a$.

Claim both sets are all residues modulo p.

If $ia = ja \bmod p$ then $(i - j)a = 0 \bmod p$ so $i = j \bmod p$.

Proof of Fermat's little Theorem: $n = p$

Proof: Let $n = p$, let $\gcd(a, p) = 1$.

Consider $1, 2, \ldots, p-1$ and $a, 2a, \ldots, (p-1)a$.

Claim both sets are all residues modulo p.

If $ia = ja \bmod p$ then $(i-j)a = 0 \bmod p$ so $i = j \bmod p$.
Thus $(p-1)! = (p-1)!a^{p-1} \bmod p$, so $a^{p-1} = 1 \bmod p$. $\square$

Proof of Fermat's little Theorem: $n = p$

Proof: Let $n = p$, let $\gcd(a, p) = 1$.

Consider $1, 2, \ldots, p - 1$ and $a, 2a, \ldots, (p - 1)a$.

Claim both sets are all residues modulo p.

If $ia = ja \bmod p$ then $(i - j)a = 0 \bmod p$ so $i = j \bmod p$.
Thus $(p - 1)! = (p - 1)!a^{p-1} \bmod p$, so $a^{p-1} = 1 \bmod p$. □

Note: General case: $x_1, \ldots, x_{\phi(n)}$ and $ax_1, \ldots, ax_{\phi(n)}$.

Proof of Fermat's little Theorem: $n = pq$

Proof: Let $n = pq$, let $\gcd(a, pq) = 1$.

Proof of Fermat's little Theorem: $n = pq$

Proof: Let $n = pq$, let $\gcd(a, pq) = 1$.

Apply FlT with a^{q-1} and p: $(a^{q-1})^{p-1} = 1 \bmod p$.

Apply FlT with a^{p-1} and q: $(a^{p-1})^{q-1} = 1 \bmod q$.

Proof of Fermat's little Theorem: $n = pq$

Proof: Let $n = pq$, let $\gcd(a, pq) = 1$.

Apply FlT with a^{q-1} and p: $(a^{q-1})^{p-1} = 1 \bmod p$.

Apply FlT with a^{p-1} and q: $(a^{p-1})^{q-1} = 1 \bmod q$.

Thus $a^{(p-1)(q-1)}$ is $1 \mod p$ and is $1 \bmod q$.

$a^{(p-1)(q-1)} = 1 + \alpha p = 1 + \beta q$.

Proof of Fermat's little Theorem: $n = pq$

Proof: Let $n = pq$, let $\gcd(a, pq) = 1$.

Apply FlT with a^{q-1} and p: $(a^{q-1})^{p-1} = 1 \bmod p$.

Apply FlT with a^{p-1} and q: $(a^{p-1})^{q-1} = 1 \bmod q$.

Thus $a^{(p-1)(q-1)}$ is $1 \mod p$ and is $1 \bmod q$.

$a^{(p-1)(q-1)} = 1 + \alpha p = 1 + \beta q$.

Thus $\alpha p = \beta q$ so $q|\alpha$ and $p|\beta$, so $a^{(p-1)(q-1)} = 1 \bmod pq$. $\square$

Primality Tests from FIT

If $\gcd(a, n) = 1$ and $a^{n-1} \neq 1 \bmod n$ then n cannot be prime.

If equalled 1 then n ***might be*** prime.

Primality Tests from FlT

If $\gcd(a, n) = 1$ and $a^{n-1} \neq 1 \bmod n$ then n cannot be prime.

If equalled 1 then n ***might be*** prime.

- If can take high powers, very fast!
- Can suggest candidate primes, and then use better, slower test for certainty.
- Carmichael numbers: Composites that are never rejected: 561, 1105, 1729, 2465, 2821, 6601, 8911, 10585, 15841, 29341, ... (OEIS A002997).

Fast Multiplication

Cost of Standard Polynomial Evaluation

Multiplication far more expensive than addition....

$f(x) = 3x^5 - 8x^4 + 7x^3 + 6x^2 - 9x + 2$: Cost is $5 + 4 + 3 + 2 + 1 + 0 = 15$ multiplications.

These are triangle numbers: degree d have $d(d+1)/2$.

Cost of Standard Polynomial Evaluation

Multiplication far more expensive than addition....

$f(x) = 3x^5 - 8x^4 + 7x^3 + 6x^2 - 9x + 2$: Cost is $5 + 4 + 3 + 2 + 1 + 0 = 15$ multiplications.

These are triangle numbers: degree d have $d(d+1)/2$.

$$\begin{aligned} S(d) &= 1 + 2 + \cdots + d \\ S(d) &= d + (d-1) + \cdots 1 \end{aligned}$$

Cost of Standard Polynomial Evaluation

Multiplication far more expensive than addition....

$f(x) = 3x^5 - 8x^4 + 7x^3 + 6x^2 - 9x + 2$: Cost is $5 + 4 + 3 + 2 + 1 + 0 = 15$ multiplications.

These are triangle numbers: degree d have $d(d+1)/2$.

$$\begin{aligned} S(d) &= 1 + 2 + \cdots + d \\ S(d) &= d + (d-1) + \cdots 1 \end{aligned}$$

Thus $2S(d) = d \cdot (d+1)$ and claim follows.

Horner's Algorithm

$f(x) = 3x^5 - 8x^4 + 7x^3 + 6x^2 - 9x + 2$: Cost is $5 + 4 + 3 + 2 + 1 + 0 = 15$ multiplications.

Horner's algorithm:

$$\left(\left(\left(\left(3x - 8\right)x + 7\right)x + 6\right)x - 9\right)x + 2.$$

Horner's Algorithm

$f(x) = 3x^5 - 8x^4 + 7x^3 + 6x^2 - 9x + 2$: Cost is $5 + 4 + 3 + 2 + 1 + 0 = 15$ multiplications.

Horner's algorithm:

$$\Bigg(\bigg(\Big(\big(3x - 8\big)x + 7\Big)x + 6\bigg)x - 9\Bigg)x + 2.$$

Cost is degree d multiplications!

Useful also in fractal plotting.... Shows can often do common tasks faster.

Fast Multiplication

Horner is best in general, but maybe for special polynomials can do better?

Try polynomials of the form $f(x) =$

Fast Multiplication

Horner is best in general, but maybe for special polynomials can do better?

Try polynomials of the form $f(x) = x^n$.

Write n in binary: Say $n = 100 = 64 + 32 + 4 = 1100100_2$.

Fast Multiplication

Horner is best in general, but maybe for special polynomials can do better?

Try polynomials of the form $f(x) = x^n$.

Write n in binary: Say $n = 100 = 64 + 32 + 4 = 1100100_2$.

$$\begin{aligned} x \cdot x &= x^2 \\ x^2 \cdot x^2 &= x^4 \\ x^4 \cdot x^4 &= x^8 \\ x^8 \cdot x^8 &= x^{16} \\ x^{16} \cdot x^{16} &= x^{32} \\ x^{32} \cdot x^{32} &= x^{64} \end{aligned}$$

Fast Multiplication

Horner is best in general, but maybe for special polynomials can do better?

Try polynomials of the form $f(x) = x^n$.

Write n in binary: Say $n = 100 = 64 + 32 + 4 = 1100100_2$.

$$
\begin{aligned}
x \cdot x &= x^2 \\
x^2 \cdot x^2 &= x^4 \\
x^4 \cdot x^4 &= x^8 \\
x^8 \cdot x^8 &= x^{16} \\
x^{16} \cdot x^{16} &= x^{32} \\
x^{32} \cdot x^{32} &= x^{64}
\end{aligned}
$$

Fast Multiplication

Horner is best in general, but maybe for special polynomials can do better?

Try polynomials of the form $f(x) = x^n$.

Write n in binary: Say $n = 100 = 64 + 32 + 4 = 1100100_2$.

$$
\begin{aligned}
x \cdot x &= x^2 \\
x^2 \cdot x^2 &= x^4 \\
x^4 \cdot x^4 &= x^8 \\
x^8 \cdot x^8 &= x^{16} \\
x^{16} \cdot x^{16} &= x^{32} \\
x^{32} \cdot x^{32} &= x^{64}
\end{aligned}
$$

Fast Multiplication

Horner is best in general, but maybe for special polynomials can do better?

Try polynomials of the form $f(x) = x^n$.

Write n in binary: Say $n = 100 = 64 + 32 + 4 = 1100100_2$.

$$
\begin{aligned}
x \cdot x &= x^2 \\
x^2 \cdot x^2 &= x^4 \\
x^4 \cdot x^4 &= x^8 \\
x^8 \cdot x^8 &= x^{16} \\
x^{16} \cdot x^{16} &= x^{32} \\
x^{32} \cdot x^{32} &= x^{64}
\end{aligned}
$$

Recap

Horner takes us from order d^2 to order d.

Fast multiplication takes us to order $\log_2 d$, but only for special polynomials; these though are the ones used in RSA!

Euclidean Algorithm

Preliminaries

Input x, y with $y > x$.

Goals: find $\gcd(x, y)$, find a, b so that $ax + by = \gcd(x, y)$.

Lot of ways to go: non-constructive proofs of a, b but need values; Euclidean algorithm is *very* fast.

Euclidean Algorithm

Let $r_0 = y, r_1 = x$.

$$r_0 \;=\; q_1 r_1 + r_2, \quad 0 \leq r_2 < r_1.$$

Euclidean Algorithm

Let $r_0 = y, r_1 = x$.

$$r_0 = q_1 r_1 + r_2, \quad 0 \leq r_2 < r_1.$$

$$r_1 = q_2 r_2 + r_3, \quad 0 \leq r_3 < r_2.$$

Euclidean Algorithm

Let $r_0 = y, r_1 = x$.

$$r_0 = q_1 r_1 + r_2, \quad 0 \leq r_2 < r_1.$$

$$r_1 = q_2 r_2 + r_3, \quad 0 \leq r_3 < r_2.$$

Continue until....

$$r_n = q_{n+1} r_{n+1} + r_{n+2}, \quad r_{n+2} \in \{0, 1\}.$$

Euclidean Algorithm

Let $r_0 = y, r_1 = x$.

$$r_0 = q_1 r_1 + r_2, \quad 0 \le r_2 < r_1.$$

$$r_1 = q_2 r_2 + r_3, \quad 0 \le r_3 < r_2.$$

Continue until....

$$r_n = q_{n+1} r_{n+1} + r_{n+2}, \quad r_{n+2} \in \{0, 1\}.$$

Note $\gcd(r_0, r_1) = \gcd(r_1, r_2) = \gcd(r_2, r_3), \ldots$.

Euclidean Algorithm

Let $r_0 = y, r_1 = x$.

$$r_0 = q_1 r_1 + r_2, \quad 0 \leq r_2 < r_1.$$

$$r_1 = q_2 r_2 + r_3, \quad 0 \leq r_3 < r_2.$$

Continue until....

$$r_n = q_{n+1} r_{n+1} + r_{n+2}, \quad r_{n+2} \in \{0, 1\}.$$

Note $\gcd(r_0, r_1) = \gcd(r_1, r_2) = \gcd(r_2, r_3), \ldots$.

Can 'climb upwards' to get a, b such that $ax + by = \gcd(x, y)$.

Implementing RSA

Implementing RSA

- Choose large primes p, q: Use FlT to get candidates.... If random choice is composite implement by 2 and try again.
- Use Euclidean algorithm to find e, d such that $ed = 1 \mod \phi(pq)$; choose a candidate e randomly and apply Euclidean algorithm to $x = e$ and $y = (p-1)(q-1)$. If gcd equals 1 win, else increase e by 2 and try again.
- Use fast multiplication to compute $M^e \mod pq$ efficiently, and also for that to the d^{th} power.