Constructing Large random primes for RSA using the deterministic Maurer’s Method

Paul LaFond

Project Objectives

• Implement Maurer’s Algorithm in C++ and measure times necessary to generate prime ranging from 2 and 2048 bits.

• Extra: Implement Miller-Rabin in C++ and compare times necessary to generate primes with Maurer’s Algorithm.
Brief Theory

• Pocklington’s theory: \( n \) is prime:
  – If: \( n-1 = F*R \)
   » \( F = q_1 \cdot q_2 \cdot \ldots \cdot q_t \) where \( q_i \)'s are distinct primes
   » \( R \) is a random number \( \{R < q\} \)
  – And: there exists \( “a” \) such that:
    – \( a^{n-1} \mod n = 1 \)
    – \( \gcd(a^{(n-1)/q_i} - 1, n) = 1 \)

Project implementation

• Platforms:
  – Intel 864 MHz Processor, 128 MB RAM
  – Intel Celeron 897 MHz processor, 128 MB RAM

• Primes:
  – Range: \([2, 2^{2048}]\) (1 decimal digit: 615 decimal digits)
  – Step Size: 32 bits (10 decimal digits)

• Code Library: MIRACL
  – (Multi Integer and Rational Arithmetic C/C++ Library)

• Random Number Generator: MIRACL
  – Based on C/C++ function rand()
Maurer’s Algorithm vs. Miller-Rabin

Maurer’s Algorithm: 3rd order polynomial approximation
(roughly the same curves for 864 MHz platform)

Processor Speed comparison

4% faster Processor results in 50% decrease in time
• Maurer’s Algorithm: 2nd order polynomial
• Miller-Rabin: nearly Linear

Why such a Big Difference !!!

• Intel Pentium III (864 MHz Processor)
  – Designed for programs that optimize its “enhanced instruction set”
  – Separate chip for cache
    • Time delay for cache hit is large

• Intel Celeron (897 MHz Processor)
  – ¼ Cache size of Pentium, but it is onboard
  – Cache delay is less than half that of the Pentium
  – optimal for Maurer’s Algorithm
    • Numerous recent memory hits in the cache
Comparison to Ueli Maurer’s results (1)

- Maurer Claimed (circa 1989)
  - “expected time for generating primes is only slightly greater than the expected time for generating pseudo-prime of the same size that passes the Miller-Rabin test for only one base”.
  - Only 40% increase in time to produce a prime compare to a pseudo prime

- Maurer’s Assumption:
  - Special Performance Hardware
    - Designed to calculate exponentiation
  - Primes of Maximum length 664 bits

Comparison to Ueli Maurer’s results (2)

- Primes <= 600 bits fall within the 40% zone
- Primes > 600 bits increase by a factor of three

- Possible Reasons:
  - Optimization of C++ code
Comparison of Algorithms

• Maurer’s Algorithm
  – Provable prime
  – ~60 seconds for 1024 bit prime (897 MHz)
  – ~60 seconds for 2 – 768 bit primes (897 MHz)

• Miller-Rabin
  – Probable prime
  – Only a few seconds for a 1024 bit pseudo-prime
    with a probability of error of $2^{-100}$.

• Difference in time to produce primes
  increase significantly (nearly a factor of 4)

Comparison (2)
When most efficient to use

• Miller-Rabin:
  – Frequent generation is necessary
  – Only slow processor is available
  – If risk of probable prime is acceptable

• Maurer’s Algorithm:
  – Higher security is required
  – Frequent generation of large primes is not
    necessary
  – Faster processors will shorten time to produce
    prime significantly
Conclusion

Use Maurer’s Algorithm

- RSA only requires 1024 bit key maximum
- No reason to continually generate new keys
- Key may be generated during off-hours
- Unless you don’t have 2 extra minutes a day why risk it (even though the risk is very small)

Questions?