Crypto Course Project

Variants of RSA

By ,
Lokananda D.M. (09c043)
Nishanth Prakash (09co63)
Soumya Mathew (09co94)

Batch RSA

- When using small public exponents e1 and e2 for the same modulus N, it is possible to decrypt two ciphertexts for approximately the price of one.
- This batching technique is only worthwhile when the public exponents e1 and e2 are small (e.g., 3 and 5).
- Batch RSA can lead to a significant improvement in RSA decryption time
- Also, one can only batch-decrypt ciphertexts encrypted using the same modulus and distinct public exponents
- Batch RSA within the Apache web server to improve the performance of the SSL handshake.
- Here the Key generation and encryption is same as standard RSA, only decryption is different.

$$M_i = C_i^{1/e_i} = \frac{A^{\alpha_i}}{C_i^{(\alpha_i - 1)/e_i} \cdot \prod_{j \neq i} C_j^{\alpha_i / e_j}} \quad \text{where} \quad \alpha_i = \begin{cases} 1 \bmod e_i \\ 0 \bmod e_j \text{ (for } j \neq i) \end{cases}$$

Multi-prime RSA

- Mprime RSA was introduced by Collins et al. [Collins et al. 1997]. It differs from plain RSA by constructing moduli with k prime factors ($n = p1p2 \cdot \cdot \cdot pk$) instead of only two.
- The key generation, encryption and decryption algorithms are as follows
- **Key generation:** The key generation algorithm takes as input a security parameter n and an additional parameter b. It generates an RSA public/private key pair as follows:
  - Step 1: Generate b distinct primes p1.. pb each n/b bits long.
    - Set N = product of all p's
  - Step 2: Pick the same e used in standard RSA public keys, namely e = 65537.
  - Then compute $d = e^{-1} \bmod '(N)$
  - For $1 < i < k$, compute $di = d \pmod{pi - 1}$; The public key is (n, e) while the private key is (n, d1, . . . , dki)

Multi-prime RSA

- **Encryption:** Given a public key <N,e> the encrypter encrypts exactly as in standard RSA.
- **Decryption:** Decryption is done using the Chinese Remainder Theorem (CRT).
  - To decrypt a ciphertext C, first compute Mi = C^di mod pi for 1 i k. Next, apply the CRT to the Mi's to obtain $M = C^d \bmod n$.
- The CRT step takes negligible time compared to the b exponentiations**.**
- **Performance**: standard RSA decryption using CRT requires two full exponentiations modulo n=2-bit numbers. In multi-prime RSA decryption requires b full exponentiations modulo n=b bit numbers. Using basic algorithms computing xd mod p takes time O(log d log2 p). When d is on the order of p the running time is O(log3 p). Therefore, the asymptotic speedup of multi-prime RSA over standard RSA is simply: $2*(n/2)^3/b*(n/b)^3 = b^2/4$
- **Security.** The security of multi-factor RSA depends on the difficulty of factoring integers of the form N = p1*..*pb for b > 2

Multi-power RSA

- Here speeding up of RSA decryption is done using a modulus of the form $N = p^{b-1}q$ where p and q are n/b bits each. When N is 1024-bits long we can use at most $b = 3$, i.e., $N = p^2q$.
- The two primes p; q are then each 341 bits long
- **Key generation:** The key generation algorithm takes as input a security parameter n and an
- additional parameter b. It generates an RSA public/private key pair as follows:
  - Step 1: Generate two distinct n/b bit primes, p and q, and compute N p^(b -1) q.
  - Step 2: Use the same public exponent e used in standard RSA public keys, namely e = 65537.
  - Compute $d = e^{-1} \bmod (phi)$.
  - Step 3: Compute r1 =d mod p-1 and r2= d mod q-1.
  - The public key is (N, e); the private key is (p, q, r1, r2i)
- **Encryption:** Same as in standard RSA.

Multi-power RSA

- **Decryption:** To decrypt a ciphertext C using the private key (p; q; r1; r2) one does:
  - Step 1: Compute M1= C^r1 mod p and M2= C^r2 mod q.
  - Step 2: Using Hensel lifting construct an M0 such that (M0)^e = C mod p^(b-1).
  - Step 3: Using CRT, compute an M such that M = M0mod p^b-1 and M = M2 mod q. Then M = Cd mod N is a proper decryption of C.
- **Performance**. For multi-power RSA, decryption takes two full exponentiations modulo (n/b)- bit numbers, and b-2 Hensel liftings. Since the Hensel-lifting is much faster than exponentiation, we focus on the time for the two exponentiations. As noted before, a full exponentiation using basic modular arithmetic algorithms takes cubic time in the size of the modulus. So, the speedup of multi-power RSA over standard RSA is approximately: $(n/2)^3/(n/b)^3 = b^3/8$
- **Security**. The security of multi-power RSA depends on the difficulty of factoring integers of the form $N = p^{b-1}q$.

Rebalanced RSA

- RSA that enables us to rebalance the difficulty of encryption and decryption: we speed up RSA decryption by shifting the work to the encrypter.
- Instead of speeding up RSA decryption by using a small value of d, d is chosen such that d is large (on the order of N), but d mod p-1 and d mod q-1 are small numbers
- **Key generation:**

Step 1: Generate two distinct (n/2)-bit primes p and q with gcd(p-1; q-1) = 2. Compute N=pq.

Step 2: Pick two random k-bit values r1 and r2 such that

gcd(r1; p - 1) = 1; and gcd(r2; q - 1) = 1; and r1 = r2 mod 2

Step 3: Find a d such that d = r1 mod p - 1 and d = r2 mod q - 1.

Step 4: Compute e d-1 mod '(N). The public key is (N,e); the private key is (p,q, r1,r2).

- **Encryption :** This is similar to standard RSA

Rebalanced RSA

- **Decryption :** To decrypt a ciphertext C using the private key hp; q; r1; r2i one does:
  - Step 1: Compute M1= C^r1 mod p and M2 =C^r2 mod q.
  - Step 2: Using the CRT compute an M 2 such that M = M1 mod p and M = M2 mod q. Note that M = C^d mod N. Hence, the resulting M is a proper decryption of C.
- **Performance:** Since modular exponentiation takes time linear in the exponent's bit-length,
- we get a speedup of (n/2)=k over standard RSA. For a 1024-bit modulus and 160-bit exponent
- (k = 160), this gives a theoretical speedup of about 3.20 over standard RSA decryption.
- **Security.** It is an open research problem whether RSA using values of d as above is secure. Since
- d is large, the usual small-d attacks do not apply.

Thank You