A Comparative analysis of the attacks on public key RSA cryptosystem

Varun Shukla

Research Scholar,RKDF-IST, RGPV,BHOPAL,Varun.shuklaa@gmail.com

Abhishek Choubey

Head of Department of Electronics and Communication,RKDF-IST,RGPV, BHOPAL
abhishekchoubey84@gmail.com

***Abstract*— The RSA is widely used public-key cryptosystem. RSA is intensively used for encryption methods and digital signature. It is purposely used in securing e-commerce and e-mail, providing authenticity of e- documents. It can be said that Internet security depends prominently on the security of the RSA cryptosystem. Since its discovery, the RSA cryptosystem has been extensively analysed for vulnerabilities. Years of cryptanalysis of RSA have given us a broad insight into its properties and provided valuable guidelines to us for proper use and implementation in different aspects. In this paper we propose an analysis of the main methods used in attacks on the RSA cryptosystem together with the new possible attack. Our focus is based on the underlying mathematical function.**

Keywords: Plaintext, RSA, Secure Sockets Layer (SSL), Chinese remainder theorem (CRT), Elliptic Curve

I THE RSA CRYPTOSYSTEM:

The RSA cryptosystem was discovered in 1977 [1] and named after its inventors. Ronald Rivest, Adi Shamir and Leonard Adleman. It is widely used to secure communication in the Internet, ensure confidentiality and authenticity of e-mail, and it has become fundamental to e-commerce. RSA is deployed in the most popular security protocols, including SET, SSH, S/MIME, PGP, DNSSEC, and SSL (Secure Sockets Layer) and TLS. It is implemented in most Web servers and browsers, and present in most commercially available security products. In fact, the implementation of RSA has placed it at the heart of modern information security. Many standards cover the use of RSA for encryption, digital signatures and key establishment. In fact, RSA is usually present wherever security of digital data does matters.

The mathematical structure of the RSA function is relatively easy and this might be paramount reason for its popularity; people do feel more familiar on working with an algorithm one can understand. It is based on basic algebraic operations on large integers.

II ATTACKS ON THE RSA FUNCTION:

(1) Trial Division:

Trial method is an attempt to divide n by all successive primes until one divisor is found. Since a composite number n must have a prime divisor $\leq n^{1/2}$ one has only to check for all primes up to the square root of n. It then follows from the Prime number theorem that the number of attempts is bounded by $(2\, n^{1/2})/(\log(n))$. Although this method is very effective when trying to factor a randomly selected composite integer or relatively small numbers [2] but it is useless against the kind of numbers currently used with RSA

(2) Low Private Exponent Attack:

The RSA decryption and signing are very calculation-intensive operations, which takes time linear to the length of the private exponent d. Thus some low-power devices may want to use a small d instead of a random one, in order to improve performance. However an attack due to small d can lead to a total break of the system. More accurately, if n is the modulus and d is the private exponent,

with $d < 1/3(n^{1/4})$, then given the public key (e, n), an attacker can efficiently recover d.

Actually, this means that for a typical 1024-bit RSA modulus, the private exponent d should be at least 300 bits in length. If the public exponent e is chosen to be 65,537 (the most commonly used value), and we calculate d as $de \equiv 1 \bmod (n)$, then we are guaranteed to have d nearly as long as n, and this attack should not pose a danger.

(3) Partial Key Exposure Attack:

A well-known principle in cryptography says that the security of any cryptographic system should rely mainly on the security of the private key. An attack on the RSA cryptosystem due to Boneh, Durfee and Frankel shows the importance of protection of the entire private exponent d [3].

They have shown that, if the modulus n is k bits long, given the (k/4) least significant bits of d, an attacker can rebuilt all of d in time linear to ($e \log(e)$), where e is the public exponent. This means that if e is small, the exposure of a quarter of bits of d can lead to the recovery of the whole private key d.

The agenda of safeguarding of RSA private keys is a crucial one, but it is a concern that is often overlooked. In Secure socket layer (SSL)-enabled servers for example, it is not unusual to have the private key stored in the computer's hard disk, in plaintext form so that the server can be re-started without human interference because an encrypted file would require the encryption key in order to start the server[3],[4].

when e is small the RSA system always leaks half the most significant bits of d, plus a intelligent technique in which one searches for randomness in order to locate private keys in large volumes of data, such as the hard disk filing system (or memory), it should be clear how important the safe storage of the RSA private key is. The best solution is the use of tamper-resistant hardware modules or tokens, in which the private key is securely stored and the private operation is performed.

(4) Broadcast and Related Message Attacks:

If someone broadcasts the encryption of the same message M to a sufficient large number of recipients (which have different public keys (ei, ni)), Hastad has described an attack in which a malicious eavesdropper can efficiently recover M if all the public exponents are small. This attack can be extended to the case in which instead of sending $(M)^{ei} \bmod ni$ to each recipient, someone sends $(fi(M))^{ei} \bmod ni$, where fi are known polynomials (for example, some kind of known padding).This is done by solving a system of univariate equations modulo relatively prime composites, which can be efficiently done if sufficiently many equations are given. There is also an attack when one sends to other related encrypted messages using the same modulus. If M1 and M2 are 2 messages such that M1 = f(M2), where f is again a known polynomial function, and one sends $C1 \equiv (M1)^e \bmod n$ and $C2 \equiv (M2)^e \bmod n$, then a malicious eavesdropper can again efficiently recover both messages if e is small. An example of such scenario is one where one sends the first encrypted message to another, which is intercepts by somebody else.

Both of these attacks are more effective if the public exponent e is 3, although they can be prevented if we remove the relation between the messages, usually by adding some kind of random padding.

(5) Short Pad Attack:

Related to the attacks above, Coppersmith has shown that an adversary can still be successful if the random padding Bob adds to the messages is not large enough. For example, if e is 3, the length of the random padding must be at least 1/9 of the message length. A variant of the attack is successful in decrypting a single ciphertext when a large fraction (2/3) of the message is known. For example, PKCS#1 standard recommends [5] the use of its simpler padding scheme (v1.5) only for encryption of relatively short messages which are typically a 128-bit symmetric key. If a long message is to be encrypted, or if part of a message is known, then the attack above may be a concern, in which case an alternative padding method should be used.

(6) Implementation Attacks:

All the attacks against the RSA as we have discussed so far, are applicable to the underlying cryptographic primitive and parameters. On the other hand, implementation attacks (side-channel attacks) always target specific implementation details.

In this case, an attacker typically uses some additional information leaked by the implementation of the RSA function or exploit faults in the implementation. The attacks are usually applied against smart cards and security tokens, and are more effective when the attacker is in possession of the cryptographic module.

Defense against side-channel attacks is hard; one usually tries to reduce the amount of information leaked or make it irrelevant to the adversary.

(7) Timing Attack:

Timing attacks [6] enjoy the advantage of the correlation between the private key and the runtime of the cryptographic operation. Recall that the RSA private operation consists of a modular exponentiation, using the private key d as exponent.

Modular exponentiations are usually implemented using an algorithm called repeated squaring algorithm. If the private key is k bits long, this consists of a loop running through the bits of d, with at most $2k$ modular multiplications. In each step the data is squared, with the execution of a modular multiplication if the current bit of the exponent is 1.

By calculating the runtime of the private operation on a large number of random messages, an attacker can recover bits of d one at a time, starting with the least significant bit (LSB). Note that in view of the partial key information attack described earlier, if a low public exponent is used, the attacker needs only to find the first $k/4$ bits using this method; the remaining bits can be found using the previous method.

To defend against timing attacks, one must understand the correlation between the runtime and the private exponent. One solution is to add a delay, so that every modular operation takes the same fixed time. This naturally affects the performance of the operation, which many times may not be a concern.

(8) Fault Analysis:

Fault Analysis attacks work by exploiting errors on key-dependent cryptographic operations. These errors can be random, latent (e.g. due to bugs in the implementation) or induced. There are a number of fault analysis attacks against public-key and symmetric-key cryptographic devices. In 1997 Boneh, DeMillo and Lipton [7] introduced an attack against RSA, which exploits possible errors on the RSA private operation in cryptographic devices.

As we have mentioned, the RSA private operation is a very calculation-intensive operation, consisting of a modular exponentiation using numbers typically in the range of 300 decimal digits. Many realizations of RSA decryption and signing use a technique known as the Chinese Remainder Theorem (CRT), which by working modulo p and q (instead of module $n = pq$), can give a four-fold enhancement in the performance. Boneh, DeMillo and Lipton explained a technique, which by exploiting an error occurring during the decryption or signing and analyzing the output, an adversary could factor the modulus n and therefore recover the device's private key. Both the output and the input of the operation are vital for the attack to succeed (making it more effective against signing devices). To perform this kind of attack, one needs only to induce an error into the device during the private operation (for example, by voltage or clock speed variation). We should also note that unlike many of the attacks described before, the difficulty of this one is independent of the key length.

This fault attack against RSA can be easily prevented by requiring the device to verify the operation with the public key before outputting the result. This should not particularly degrade the performance, as the RSA public operation is very fast. Also we note that the attacker needs to have full knowledge of the actual string being signed or decrypted; the addition of random padding that is unknown to the attacker by the device prior to the operation would also prevent this type of attack.

Nevertheless the existence of fault analysis attacks stresses the need for special care when implementing secure cryptographic applications.

(9) Key Size –RSA keys should be long enough so as to make attacks against the system infeasible. The selection should take into consideration a number of factors such as the value of data being protected, the expected lifetime of the data, the threat model, and the best possible attacks. Many of the current standards require a minimum length of 1024 bits for RSA keys. Since the solution of the RSA-155 challenge [10], there is a general consensus that 512-bit keys are too short and should not be used.

(10) Strong Primes - Apart from a minimal length, some cryptographic standards also need that the primes used for RSA have some special properties [6]. In particular, that $p-1$ needs to have a large prime factor. These so-called strong primes are required in order to prevent Pollard's $p-1$ factorization method. In spite of that, Rivest and Silverman [8] find these requirements unnecessary. In view of the Elliptic Curve and Number Field Sieve methods (which have better runtime than Pollard's $p-1$ method), they believe that strong primes offer a negligible increase in security when compared with random primes. The real security comes from actually choosing large enough primes p and q.

(11) Multi-prime RSA – To speed up the RSA private operation, there are proposals of using more than two primes to generate the modulus. Using CRT, the speedup of an RSA system using k primes over the standard RSA is of around $k^2/4$. For illustration, a system deployed with 1024-bit modulus n, which is the product of 4 distinct (256-bit) primes, would perform around 4 times faster than the standard two-prime system. While the Number Field Sieve method cannot take advantage of this special form of n, 256-bit primes are currently considered within the bounds of the Elliptic Curve method (which is quite effective in finding small primes). Therefore, it is not recommended that 1024-bit modulo use more than three factors.

(12) Public Exponent – Mostly in RSA implementations, the public exponent e is chosen to be either 3 or $2^{16}+1$ (= 65,537), with which the public operation takes 2 and 17 modular multiplications (instead of ~ 1000 multiplications expected for a randomly chosen e). Particularly, 65,537 is more secure and can be used.

(13) Private Exponent – Wiener's low private exponent attack is quite effective, and if successful, can result in total recovery of the private exponent d. For the typical 1024-bit key, the private exponent should be at least 300 bits long. One should also pay special concern to the safety of the private key. The knowledge of a fraction of the key might allow the recovery of the entire key. The hardware solution is the most secure and should be considered whenever possible.

III THE NEW PROPOSED ATTACK ALGORITHM:

Here we address the million dollar question: is there a possible attack on the RSA cryptosystem other than factoring n? The answer is yes [9], there are few methods that attack the RSA scheme that does not involve finding the factoring of the modulus n but most of them carrying some deficiencies [3].

We will now prove the very interesting result that, as long as the exponent key e is known, then n can be factored in polynomial time by means of a randomized algorithm. Therefore we can say that computing this method is no easier than factoring n. However, this does not rule out the possibility of breaking the RSA cryptosystem without involving e . Notice that this result is of much more than theoretical interest.

In this paper we proposed a method that breaking the RSA scheme based on the knowing public key (e, n) . This method will work efficiently if the exponent key e . It is possible to recover the entire private exponent *d* and therefore factor the modulus n .

Algorithm: The steps are in this manner

1. Find entity public key A (e,n)
2. Change the modules n into its binary equivalent
3. Number of bits in n is equal to b.
4. Calculate $d=\lceil b/4 \rceil$
5. Find $ed \equiv 1+k(n-s-1) \bmod 2^b$

6. Repeat k from 1 to e until

$P^2 - s*p+n \equiv 0 \bmod 2^b$ is true

And calculate $ed \equiv 1+k(n-s+1) \bmod 2^d$

Also calculate $p^2 - s*p+n \equiv 0 \bmod 2^d$

7. Find $p_0 \equiv p \bmod 2^d$

8. Find $q_0 * p_0 \equiv n \bmod 2^d$

9. Find θ(n) by computing:

$n \equiv (2^d * x+p_0)*(2^d * y+q_0)$

$p=(2^d * x+p_0)$, $q=(2^d * x+q_0)$

So θ(n)= (p-1)(q-1)

10. Finally d=e*d-k* θ(n)=1

IV AN EXAMPLE TO ILLUSTRATE:

1. Suppose that the public key (e=23, n=1633)
2. Convert n into its binary equivalent i.e. $(11001100001)_2$
3. b=11
4. $d=\lceil 11/4 \rceil = 3$
5. $(e= 23*d=d) \equiv 1+k(n=1633-s+1) \bmod (2^b=8)$

   $69 \equiv 1+k(1634-s) \pmod 8$

   $69 \bmod 8=5$

   Now, $5 \equiv 1+k(1634-s) \pmod 8$

   $4 \equiv k(1634-s) \pmod 8$

6. For k=1 to 23 do
   (a) $4 \equiv 1(1634-s) \pmod 8$
   $s \equiv (1634-4) \pmod 8$

   $s=1630 \bmod 8$

   $s=6$

   (b) $p^2-(s=6)*p+(n=1633) \equiv 0 \bmod (2^d=8)$
   $p^2-6p+1633 \equiv 0 \bmod 8$

   $p^2-6p \equiv -1633 \bmod 8$

   $p^2-6p \equiv 7 \bmod 8$

   $7^2-6*7 \equiv 7 \bmod 8$

   $49-42 \equiv 7 \bmod 8$

   $7 \bmod 8 \equiv 7 \bmod 8$

   So p=7

   It means $p^2-(s=6)*p+(n=1633) \equiv (0 \bmod 2^b=8)$ holds true

   So as a result, loop must be stopped.

7. $p_0 \equiv (p=7) \pmod{2^d \equiv 8}$
   $p_0 \equiv 7$

8. $q_0 *(p0=7) \equiv (n=1633 \bmod 2^d=8)$

   $7q_0 \equiv 1633 \bmod 8$

   $7q_0 \equiv 1 \bmod 8$, inverse of 7 mod 8 is 7

   $q_0 \equiv 7 \bmod 8$

   So $q_0 \equiv 7$

9. Find θ(n)

   $n \equiv (2^d * x+p_0)*(2^d * y+q_0)$

   $1633 \equiv (8*x+7)(8y+7)$

   $1633 \equiv (8*2+7)(8*8+7)$

   $1633 \equiv (23)(71)$

1633□1633

S0 x=2 and y=8

That means p=23, q=71

θ(n)=(23-1) (71-1)

θ(n)=1540

10. (e=23*d-(k=1)*(θ(n)=1540) □ 1
23d□1541

d= 67 (By multiplicative inverse method)

V CONCLUSION:

Since RSA is applicable in e-commerce and digital signatures, its security is a million dollar question. There are so many attacks already defined on RSA based on factoring. Methods like trial division are time taking and low private attack contains a limitation that the private exponent d should be at least 300 bits in length. The new proposed method is essentially based on public key (e,n) and does not have all these limitations. So we can say that the new algorithm is suitable for implementation aspects.

References:

(*1) R. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital*
Signatures and Public Key Cryptosystems. Communications of ACM 21
(1978) 120-125
(2) B. Canvel. Password Interception in a SSL/TLS Channel.
(3) D. Boneh. Twenty Years of Attacks on the RSA Cryptosystem.
Notices of the American Mathematical Society, 46(2):203--213, 1999.
(4) A.Shamir and N. van Someren. Playing hide and seek with stored keys.
Lectures in Computer Science, 1998
(5) RSA Laboratories. PKCS #1 v2.1 - RSA Encryption Standard. June 2002.
(6) B. Kaliski. Timing Attacks on Cryptosystems. RSA Laboratories' Bulletin
No. 2, January 1996
(7) B. Kaliski and M. Robshaw. Comments on Some New Attacks on
Cryptographic Devices. RSA Laboratories' Bulletin No. 5, July 1997.
(8) R. Rivest and R. Silverman. Are Strong Primes Needed for RSA?
RSA Laboratories Technical Notes and Reports
(9)C. KAUFMAN, R. PERLMAN, "Network Security –private communication in a public world", 2nd edition, Prince Hall PTR, 2002.
(10)RSA Laboratories. Factorization of RSA-155.