RSA (sistema criptográfico)

Nota: Se procura por a empresa, veja RSA Data Security, Inc..

RSA (**Rivest-Shamir-Adleman**) é um dos primeiros sistemas de criptografia de chave pública e é amplamente utilizado para transmissão segura de dados. Neste sistema de criptografia, a chave de encriptação é pública e é diferente da chave de decriptação que é secreta (privada). No RSA, esta assimetria é baseada na dificuldade prática da fatorização do produto de dois números primos grandes, o "problema de fatoração". O acrônimo RSA é composto das letras iniciais dos sobrenomes de Ron Rivest, Adi Shamir e Leonard Adleman, fundadores da atual empresa RSA Data Security, Inc., os quais foram os primeiros a descrever o algoritmo em 1978. Clifford Cocks, um matemático Inglês que trabalhava para a agência de inteligência britânica Government Communications Headquarters (GCHQ), desenvolveu um sistema equivalente em 1973, mas ele não foi revelado até 1997.[1]

Adi Shamir, um dos criadores do RSA

É considerado dos mais seguros, já que mandou por terra todas as tentativas de quebrá-lo. Foi também o primeiro algoritmo a possibilitar criptografia e assinatura digital, e uma das grandes inovações em criptografia de chave pública.

Um usuário do RSA cria e publica uma chave (**chave pública**) baseada em dois números primos grandes, junto com um valor auxiliar. Os números primos devem ser mantidos secretos. Qualquer um pode usar a chave pública para encriptar a mensagem, mas com métodos atualmente publicados, e se a chave pública for muito grande, apenas alguém com o conhecimento dos números primos pode decodificar a mensagem de forma viável. Quebrar a encriptação RSA é conhecido como problema RSA. Se ele for tão difícil quanto o problema de fatoramento, ele permanece como uma questão em aberto.

O RSA é um algoritmo relativamente lento e, por isso, é menos usado para criptografar diretamente os dados do usuário. Mais frequentemente, o RSA passa chaves criptografadas compartilhadas para criptografia de chave simétrica que, por sua vez, pode executar operações de criptografia-descriptografia em massa a uma velocidade muito maior.

Funcionamento

O RSA envolve um par de chaves, uma chave pública que pode ser conhecida por todos e uma chave privada que deve ser mantida em sigilo. Toda mensagem cifrada usando uma chave pública só pode ser decifrada usando a respectiva chave privada. A criptografia RSA atua diretamente na internet, por exemplo, em mensagens de emails, em compras on-line e o que você imaginar; tudo isso é codificado e recodificado pela criptografia RSA.

Geração das chaves

No RSA as chaves são geradas desta maneira:

1. Escolha de forma aleatória dois números primos grandes p e q, da ordem de 10^{100} no mínimo.
2. Calcule $n = pq$
3. Calcule a função Função totiente de Euler em n: $\phi(n) = (p-1)(q-1)$.[2]
4. Escolha um inteiro e tal que 1 < e < $\phi(n)$, de forma que e e $\phi(n)$ sejam relativamente primos entre si.
5. Calcule d de forma que $de \equiv 1 \pmod{\phi(n)}$, ou seja, d seja o inverso multiplicativo de e em $\pmod{\phi(n)}$.

- No passo 1 os números podem ser testados probabilisticamente para primalidade
- No passo 5 é usado o algoritmo de Euclides estendido, e o conceito de inverso multiplicativo que vem da aritmética modular

Por final temos:

A chave pública: o par (n, e).

A chave privada: a tripla (p, q, d). (De fato, para desencriptar, basta guardar d como chave privada, mas os primos p e q são usados para acelerar os cálculos)

Cifragem

Para transformar uma mensagem m, onde $1 < m < n-1$, numa mensagem c cifrada usando a chave pública do destinatário n e e basta fazer uma potenciação modular:

$$m^e \equiv c \mod n$$

A mensagem então pode ser transmitida em canal inseguro para o receptor. Há um algoritmo para realizar esta potência rapidamente.

Decifragem

Para recuperar a mensagem m da mensagem cifrada c usando a respectiva chave privada do receptor n e d, basta fazer outra potenciação modular:

$$c^d \equiv m \mod n$$

Implementação

Em traços gerais, são gerados dois pares de números – as chaves – de tal forma que uma mensagem criptografada com o primeiro par possa ser apenas decriptada com o segundo par; mas, o segundo número não pode ser derivado do primeiro. Esta propriedade assegura que o primeiro número possa ser divulgado a alguém que pretenda enviar uma mensagem criptografada ao detentor do segundo número, já que apenas essa pessoa pode decriptar a mensagem. O primeiro par é designado como chave pública, e o segundo como chave secreta.

RSA baseia-se no fato de que, embora seja fácil encontrar dois números primos de grandes dimensões (p.e. 100 dígitos), conseguir factorizar o produto de tais dois números é considerado computacionalmente complexo (em outras palavras, o tempo estimado para o conseguir ronda os milhares de anos). De fato, este algoritmo mostra-se computacionalmente inquebrável com números de tais dimensões, e a sua força é geralmente quantificada com o número de bits utilizados para descrever tais números. Para um número de 100 dígitos são necessários 330 bits, e as implementações atuais superam os 1024 e mesmo os 2048 bits (a conversão do sistema decimal para o sistema binário é feito de forma clássica).

RSA é usado comumente para transferir senhas RC4 por ser mais rápido. A senha, geralmente, tem apenas 128 bits (16 bytes) o que facilita o manuseio, já que os processadores modernos tem tipos de 16 bytes embora restringido pelo número de operações. Geralmente o servidor, como por exemplo o servidor HTTPS, gera um par de chaves, uma chave pública e uma chave privada, transmite a chave pública para o cliente, e este gera uma senha RC4 (ou de qualquer outro padrão), criptografa com a chave pública do servidor e envia de volta para o servidor. Assim, tanto o receptor quanto o servidor podem usar a senha RC4 de forma segura para criptografar e descriptografar.

Em Java

```
import java.math.BigInteger;
import java.security.SecureRandom;

class RSATest {

  public static void main(String args[]) {
    String msg = "Paz e felicidade a todos!";
    String msgcifrada = null;
    String msgdecifrada = null;
    BigInteger n, d, e;
    int bitlen = 2048;

    //Escolha de forma aleatória dois números primos grandes p e q
    SecureRandom r = new SecureRandom();
    BigInteger p = new BigInteger(bitlen / 2, 100, r);
    BigInteger q = new BigInteger(bitlen / 2, 100, r);

    //Compute n = p * q
    n = p.multiply(q);

    //Compute a função totiente phi(n) = (p -1) (q -1)
    BigInteger m =
(p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));

    //Escolha um inteiro  "e"  , 1 < "e" < phi(n) ,  "e" e phi(n)
sejam primos entre si.
    e = new BigInteger("3");
    while(m.gcd(e).intValue() > 1) e = e.add(new BigInteger("2"));

    // d seja inverso multiplicativo de "e"
    d = e.modInverse(m);

    System.out.println("p:"+p);
    System.out.println("q:"+q);
    System.out.println("n:"+n);
    System.out.println("e:"+e);
    System.out.println("d:"+d);

    //mensagem cifrada - RSA_encrypt()
```

```
    msgcifrada = new BigInteger(msg.getBytes()).modPow(e,
n).toString();
    System.out.println("msg cifrada: "+ msgcifrada);

    //mensagem decifrada - RSA_decrypt()
    msgdecifrada = new String(new BigInteger(msgcifrada).modPow(d,
n).toByteArray());
    System.out.println("msg decifrada: " +msgdecifrada);
  }
}
```

Correio Anônimo

Correio Anônimo é uma técnica utilizada para enviar mensagens anonimamente utilizando criptografia RSA, de forma que seja necessário, vários computadores, usando aplicativos para a estragar o anonimato, para que isto seja possível. Envio a requisição de chave publica, para vários computadores selecionados aleatoriamente. Criptografo na ordem inversa a que eu vou enviar(usando as chaves publicas que recebi), ou seja, primeiro com a senha do ultimo que irá receber a mensagem, e por ultimo com a senha do primeiro. Envio para o primeiro de forma que ele só saiba o segundo destinatário quando descriptografar. Sendo que se todos os destinatários, estiverem sem criptografia, o segundo pode notificar o ultimo, estragando o sigilo. Seguindo está sequência, descriptografo e envio para o próximo. Como ninguém sabe a rota, só sabem o destinatário atual, e como também não sabem o que há no pacote, dificilmente, alguém poderá encontrar o remetente. Trabalhar usando RSA permite, que, nem mesmo gravando todas as conexões de um determinado computador na rede(inclusive o remetente), seja possível ler os pacotes transmitidos.

Assinatura digital

🔍Ver artigo principal: Assinatura digital

O algoritmo RSA é extensível a este contexto, pelas suas propriedades. Para implementar um sistema de assinaturas digitais com RSA, o utilizador que possua uma chave privada *d* poderá assinar uma dada mensagem (em blocos) *m* com a seguinte expressão:

$$s = m^d \mod n$$

Como se pode deduzir, é difícil descobrir *s* sem o conhecimento de *d*. Portanto, uma assinatura digital definida conforme esta equação é difícil de forjar. Mas o emissor de *m* não pode negar têla emitido, já que mais ninguém poderia ter criado tal assinatura. O receptor recupera a mensagem utilizando a chave pública *e* do emissor:

$$s^e = (m^d)^e \mod n = m \mod n$$

Como tal, o receptor consegue validar a assinatura do emissor calculando s^e *mod n*. Podemos verificar então que o algoritmo RSA satisfaz os três requisitos necessários de uma assinatura digital.

É fácil deduzir que a assinatura varia dependentemente da mensagem em si, e que operando sobre mensagens longas o tamanho da assinatura seria proporcional. Para colmatar esta situação, faz-se operar o algoritmo sobre um resumo (*digest*) da mensagem, que identifique essa mensagem como única – geralmente o *digest* de uma mensagem varia alterando um único byte –, o que mantém, como consequência, que uma assinatura varia de mensagem para mensagem, para um mesmo emissor. Exemplo de função geradora do *digest* é o Secure Hash (SHA-1).

Vulnerabilidades do RSA

Por ser um sistema de criptografia muito utilizado, o RSA vem tendo suas vulnerabilidades pesquisadas e analisadas praticamente desde sua publicação inicial. Uma lista bem detalhada de ataques ao sistema pode ser encontrada no artigo de Dan Boneh.[3] Segue um resumo de alguns tipos de ataque discutidos no artigo.

Fatorando números inteiros muito grandes

Uma primeira abordagem de ataque ao RSA teria como objetivo a chave pública por meio da fatoração do módulo n , ou seja, dada uma fatoração de n , pode-se chegar ao expoente de descriptografia. Este é um exemplo de ataque de força bruta ao RSA.

Apesar da melhora constante dos algoritmos de fatoração de números inteiros, esta ainda é uma ameaça considerada distante da realidade, caso o sistema RSA seja corretamente implementado, devido à dificuldade para a fatoração de números inteiros com a tecnologia atual.

A título de ilustração Christof Paar[4] apresenta a discussão sobre o módulo de 129 dígitos publicada em um artigo da revista *Scientific American* por Martin Gardner em 1997. Fazendo uso dos métodos de fatoração disponíveis e poder computacional da época foi estimado que seriam necessários 40 trilhões de anos para fatorar um número desta magnitude porém a tarefa levou apenas 30 anos para ser realizada.

A discussão então recai em qual é o tamanho seguro para o módulo usado no RSA? Ainda de acordo com Christof Paar muitas aplicações utilizam módulos de 1024 bits porém hoje em dia acredita-se que será possível fatorar números desta magnitude em 10-15 anos e a agências de inteligência podem ser capazes de fazê-lo ainda mais rápido, já que tipicamente empregam as maiores autoridades em criptografia do mundo. Portanto, já é aconselhado utilizar parâmetros de RSA da ordem de 2048-4096 bits para uma segurança mais duradoura.

Módulo comum

Para evitar a geração de módulos n diferentes poderia ser considerado o uso do mesmo módulo n para todos os usuários emitido, por exemplo, por uma autoridade central confiável. Apesar de parecer eficiente em uma primeira análise, um usuário poderia usar seus próprios expoentes para fatorar o módulo n de outros usuários. Devido a este fato, um módulo RSA nunca deve ser utilizado por mais de uma entidade.

Este é considerado um ataque elementar pois ilustra o uso errôneo do sistema RSA.

Pequeno expoente da chave privada

Para reduzir o tempo necessário para decriptar uma mensagem ou o tempo necessário para gerar uma assinatura pode-se tentar usar um valor de d pequeno no lugar de um d aleatório. Usando um d pequeno pode-se alcançar uma melhora no desempenho em torno de fatores de 10 para um módulo de 1024 bits porém Michael Wiener[5] mostra que a escolha de um d pequeno pode quebrar completamente o sistema RSA.

Pequeno expoente da chave pública

Pelos mesmos motivos do item anterior, é comum utilizar um expoente público e pequeno. O menor e possível é três porém para o sistema ficar protegido de alguns tipos de ataque sugere-se o valor $e = 2^{16} + 1 = 65537$ e a especificação do sistema RSA menciona a geração aleatória de e para tornar o sistema ainda mais seguro.

Ataques desta natureza, ao contrário dos baseados no método anterior, não levam a uma quebra total do sistema RSA, sendo portanto menos perigosos.

Os ataques mais poderosos relacionados ao pequeno expoente da chave pública utilizam resultados baseados no *Teorema de Coppersmith*.[6]

Exposição parcial da chave privada

Imaginemos que um usuário teve acesso a uma fração da chave privada, de tamanho d bits. Seria este usuário capaz de reproduzir o restante da chave d a partir desta fração? Surpreendentemente a resposta é positiva caso a chave d seja pequena o suficiente.

Um artigo de 1998[7] mostra que sendo $e < \sqrt{n}$ é possível reconstruir toda a chave a partir de uma fração da mesma. Este resultado mostra a importância de proteger a chave privada RSA de forma eficiente e completa.

Ataques temporais

Alguns ataques não são decorrentes de falhas ou artifícios matemáticos. Estes ataques são chamados de *ataques de implementação* e buscam vulnerabilidades na implementação computacional do sistema RSA, segue um exemplo de ataques desta natureza.

Imaginemos que a chave privada do RSA está resguardada pelo armazenamento em um *smartcard* devidamente protegido. Um potencial ataque não consegue examinar o conteúdo do cartão e portanto não chega a expor a chave. Porém, um artigo de 1996[8] mostra que por meio de uma medição precisa do tempo que o *smartcard* demora para executar uma decriptação ou assinatura do RSA é possível descobrir o expoente d da chave privada e desta forma deixar exposto todo o sistema.

Referências

1. Smart, Nigel (19 de fevereiro de 2008). «Dr Clifford Cocks CB» . Bristol University. Consultado em 14 de agosto de 2011
2. David Lowry (30 de setembro de 2013). «Using Carmichael function in RSA» (em inglês). Mathematics Stack Exchange. Consultado em 1 de junho de 2018
3. Boneh, Dan (1999). «Twenty Years of Attacks on the RSA Cryptosystem». *Notices of the AMS*. **46**: 203–213
4. Paar, Christof (2009). *Understanding cryptography a textbook for students and practitioners*. Berlin London: Springer. ISBN 9783642041013
5. Wiener, Michael J. (1990). «Cryptanalysis of short RSA secret exponents». *IEEE Transactions on Information Theory*. **36**: 553–558
6. Coppersmith, Don (1997). «Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities». *Journal of Cryptology*. 10(4): 233–260
7. Boneh, Dan; et al. (1998). «Exposing an RSA Private Key Given a Small Fraction of its Bits»

8. Kocher, Paul C. (1996). «Timing Attacks on Implementations of Diffie-Hellman RSA DSS and Other Systems». *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*: 104-113

Ver também

- Criptografia
- Assinatura digital
- Complexidade computacional
- Criptografia Quântica

Ligações externas

- PKCS #1: RSA Cryptography Standard (RSA Laboratories website)
  - O standard *PKCS #1 "tece algumas recomendações para a implementação de criptografia de chave pública, abrangendo os seguintes temas: primitivas criptográficas; métodos de criptografia;".*
- A Method for Obtaining Digital Signatures and Public-Key Cryptosystems , R. Rivest, A. Shamir, L. Adleman, Communications of the ACM, Vol. 21 (2), 1978, pages 120—126. Publicado pelo Instituto MIT como um "Memorando Técnico" em Abril de 1977.
  - Publicação inicial do esquema *RSA*.
- http://alexm.unetvale.com.br/blog/2009/07/criptografia-rsa-em-simples-passos/ RSA em scripts PHP.
- http://www.java2s.com/Tutorial/Java/0490__Security/BasicRSAexample.htm RSA em Java tutorial em inglês