O algoritmo RSA

Gustavo Zambonin
Segurança em Computação (UFSC-INE5429)

- O algoritmo conhecido como RSA, publicado em 1978 por Rivest, Shamir e Adleman, é uma das implementações de criptografia assimétrica mais utilizadas até o presente momento. É baseado na dificuldade de fatorar o produto de dois grandes números primos. Pares de chaves gerados de acordo com estes fatores primos são facilmente computáveis, onde a chave *pública* $K_{Pu} = (e, n)$ é compartilhada e a chave *privada* $K_{\text{Pr}} = (d, n)$ é guardada. Revelar a chave pública não faz com que seja fácil computar a chave privada, e assim, apenas uma entidade pode decodificar mensagens codificadas com sua própria chave pública e enviadas a ela.

  Para gerar um par de chaves, escolhe-se um par de primos p e q aleatórios, com um grande número de dígitos, e sua multiplicação $n = p \cdot q$ é calculada. A parte d da chave privada K_{Pr} é um número igualmente aleatório e grande, mas relativamente primo a $\phi(n) = (p-1) \cdot (q-1)$, o que significa mdc[1]$(d, \phi(n)) = 1$. A parte e da chave pública obtém-se a partir da inversa multiplicativa[2] de d, ou seja, $e \cdot d \equiv 1 \pmod{\phi(n)}$. Por fim, para codificar uma mensagem M, transforma-se esta em um inteiro no intervalo $[0, n-1]$ (talvez seja necessário quebrá-la em blocos). Então, o texto cifrado C é obtido a partir de $M^e \pmod{n}$. Para decifrá-lo, o cálculo $C^d \pmod{n}$ deve ser realizado. A prova para $M^{e \cdot d} \equiv M$ módulo n pode ser encontrada na publicação original [3].

- Seja $p = 211$ e $q = 257$, então $n = 211 \cdot 257 = 54227$ e $\phi(n) = 210 \cdot 256 = 53760$. Um inteiro d relativamente primo a $\phi(n)$ é 46937, e $46937^{-1} \pmod{53760} = e = 15593$ (e claramente, $46937 \cdot 15593 \equiv 1 \pmod{53760}$). Assim, $K_{\text{Pr}} = (46937, 54227)$ e $K_{Pu} = (15593, 54227)$. Para uma mensagem $M = 26706$, $C = 26706^{15593} \pmod{54227} = 46137$. Decodificando-a, temos $46137^{46937} \pmod{54227} = 26706 = M$.

- O arquivo `rsa.py` contém a definição do algoritmo, bem como métodos para codificar e decodificar uma mensagem numérica ou alfabética. É possível criar um objeto desta classe a partir de dois números primos. Demonstra-se que o exemplo numérico acima funciona:

```
$ python
>>> from rsa import RSA
>>> x = RSA(211, 257)
>>> msg = 26706
>>> enc_m = x.encrypt(msg)
>>> dec_m = x.decrypt(enc_m)
>>> m == dec_m
True
```

  O arquivo `gen_keys.py` gera chaves aleatórias de tamanho 64 a 4096 bits, e sua saída é mostrada abaixo. As chaves podem demorar para serem geradas, visto que o teste de primalidade utilizado não é otimizado, e os números aleatórios escolhidos podem ser regularmente compostos.

```
$ python gen_keys.py
Time: 0:00:00.154982    Bits: 64
Your public key is (65537, 15125487135336563173896939309783231757 9).
Your private key is (13402616673379387162790407546599967283 3,
                     15125487135336563173896939309783231757 9).

Time: 0:00:00.769706    Bits: 128
Your public key is (65537,
193238576067611345845586741484208360597346905905911275008112614709534597103 57).
Your private key is
(108724525900134286993761775874524300601865521962162977939776972606908486683 53,
193238576067611345845586741484208360597346905905911275008112614709534597103 57).

Time: 0:00:04.451425    Bits: 256
```

gustavo.zambonin@grad.ufsc.br src/

[1] "mdc" quer dizer "máximo divisor comum".

[2] a inversa multiplicativa modular de um inteiro a módulo m é um inteiro x tal que $a \cdot x \equiv 1 \pmod{m}$, ou seja, a inversa multiplicativa no anel de inteiros $\mathbb{Z}_m$.

```
Your public key is (65537,
34344635189650785136064165266960865782400826451839667081439595973954430930004695672626387716704128891225332714502830644826776879531592704071229681193121 27).
Your private key is
(33947929682249383723915294047541463377693906305601013679839739798781879482515605079314541279266429303113525647623604365896107803486002449286869061973091 3,
34344635189650785136064165266960865782400826451839667081439595973954430930004695672626387716704128891225332714502830644826776879531592704071229681193121 27).
...
```

Questionário

- A função totiente de Euler, representada por $\phi(n)$, pode ser definida como o número de inteiros k tal que $mdc(n, k) = 1$ em $1 \leq k \leq n$, ou seja, a contagem de todos os inteiros nesse intervalo que são primos entre si.
- Como deve ser teoricamente impossível obter uma chave privada de uma chave pública, o uso de n como módulo para os expoentes habilitaria qualquer entidade a obter a chave privada a partir da pública com o cálculo da inversa multiplicativa. Supondo que e é a chave pública e d é a chave privada, então caso o módulo n fosse utilizado, $ed \equiv 1 \pmod{n} \longrightarrow d \equiv e^{-1} \pmod{n}$. O módulo $\phi(n)$ é utilizado pela dificuldade de ser computado dado apenas n, mas trivialmente descoberto se os fatores $p \cdot q = n$ forem conhecidos, pois $\phi(n) = \phi(pq) = \phi(p) \cdot \phi(q) = (p-1) \cdot (q-1)$.
- PKCS#1 (*Public-Key Cryptography Standards* nº 1) contém definições básicas e recomendações para implementar o RSA, bem como propriedades matemáticas de chaves públicas e privadas, sintaxe de estruturas relativas à criptografia, e algoritmos para codificação/decodificação de texto, e produção e verificação de assinaturas.
- Embora não seja recomendado cifrar grandes mensagens com o RSA por conta de sua velocidade de cifragem, tamanho da mensagem cifrada final, e simplesmente falta de uso desta estratégia, é possível realizar este procedimento a partir de encadeamento de blocos (CBC — *cipher block chaining*). O uso mais difundido do RSA acontece junto a um algoritmo de criptografia assimétrica como o AES, deste modo: seja uma mensagem M, uma chave K_{AES} aleatoriamente gerada, e $K_{RSA_{Pu}}$ a chave pública do destinatário. Então, o texto cifrado C_M contém a mensagem cifrada com K_{AES}, e C_K contém K_{AES} cifrado com $K_{RSA_{Pu}}$. Enviando C_M e C_K por um canal seguro, o destinatário pode decifrar C_K com sua chave privada $K_{RSA_{\mathrm{Pr}}}$, e obter K_{AES} para decifrar C_M.
- Existem diversos possíveis ataques ao RSA, que podem ser classificados em quatro categorias; ataques elementares, que exploram o mau uso do algoritmo, ataques a pequenos expoentes privados, ataques a pequenos expoentes públicos, e ataques à implementação do RSA [1]. O ataque mais direto que pode ser executado, até o momento, é a fatoração do módulo $n = p \cdot q$, a partir do algoritmo GNFS (*General Number Field Sieve*). Também é possível, por exemplo, descobrir chaves públicas e privadas que utilizam o mesmo módulo — o ataque do módulo em comum, realizado aplicando o Teorema Chinês do Resto.
- O uso de números aleatórios, gerados a partir de um gerador criptograficamente seguro, é extremamente importante para que p e q sejam escolhidos de modo que não possam ser fatorados; por exemplo, caso duas chaves públicas contenham um fator em comum, torna-se trivial encontrar o outro fator.
- O tamanho desejável para uma chave é de, no mínimo, 2048 bits. No final de 2009, uma chave de 768 bits foi fatorada [2], e demonstrou-se que na próxima década, será possível fatorar chaves de 1024 bits por conta do poder computacional crescente. Assim, chaves de 2048 e 4096 serão seguras por muitos anos em computadores clássicos, dado esforço apenas acadêmico; o mesmo não pode ser afirmado para computadores quânticos.

Referências

[1] Dan Boneh. Twenty years of attacks on the rsa cryptosystem. *Notices of the AMS*, 46:203–213, February 1999.

[2] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thomé, Joppe W. Bos, Pierrick Gaudry, Alexander Kruppa, Peter L. Montgomery, Dag Arne Osvik, Herman Te Riele, Andrey Timofeev, and Paul Zimmermann. Factorization of a 768-bit rsa modulus. In *Proceedings of the 30th Annual Conference on Advances in Cryptology*, CRYPTO'10, pages 333–350, Berlin, Heidelberg, 2010. Springer-Verlag.

[3] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.