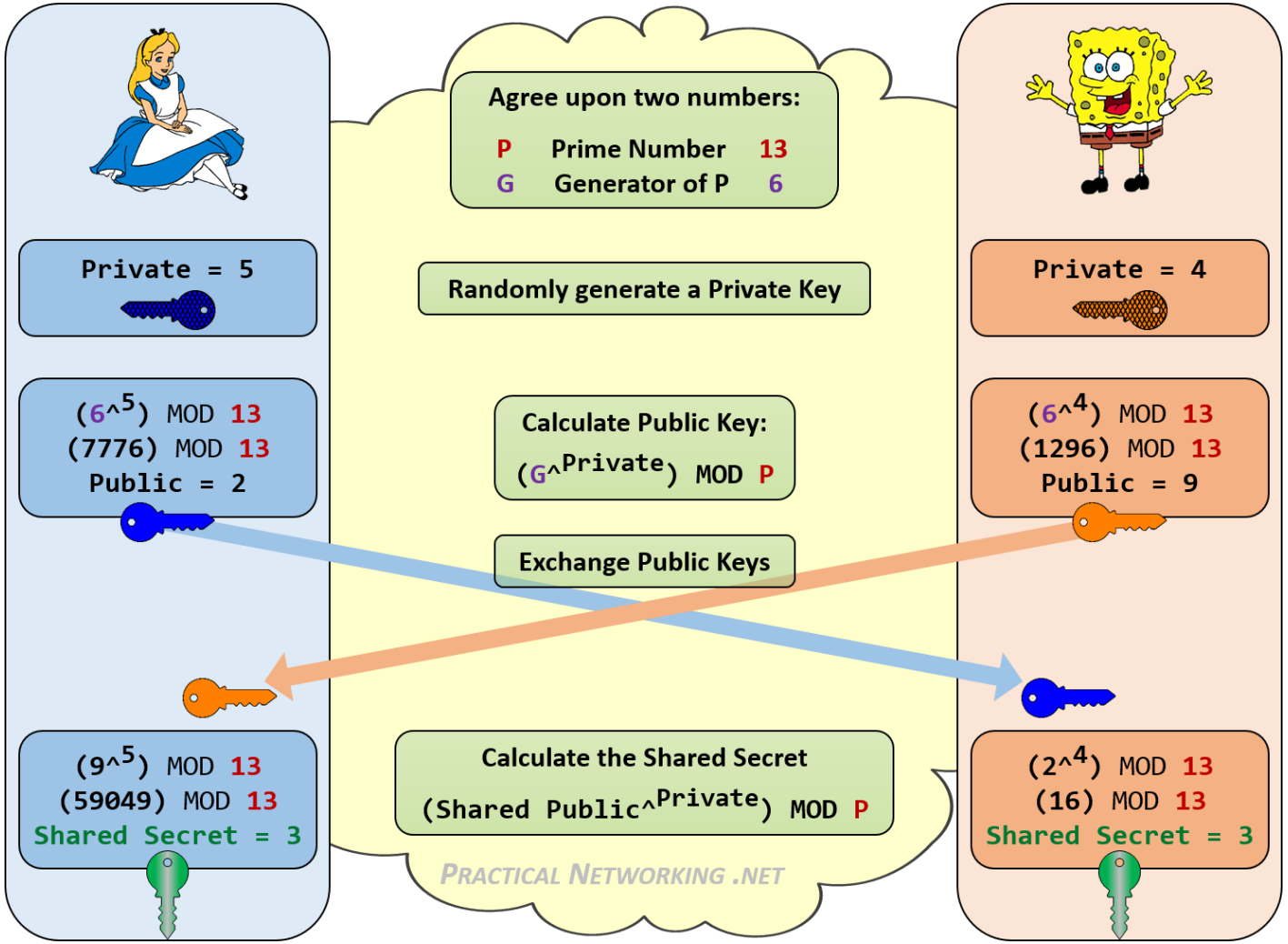
Segurança da Informação– GBC083

Prof. Rodrigo Sanches Miani – FACOM/UFU

Aulas passadas

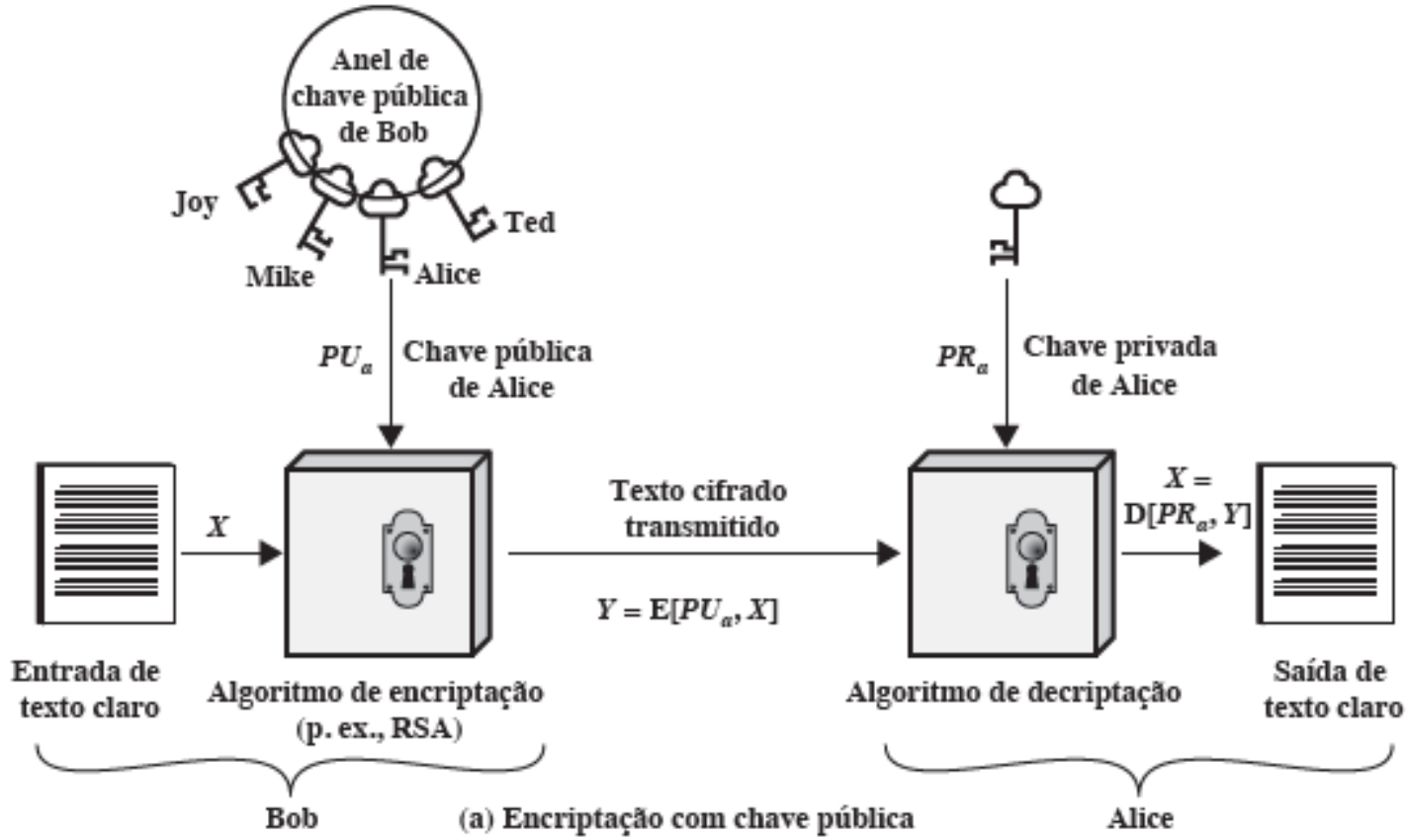
Segurança da Informação– GBC083

Diffie-Hellman (Ideia)



Retirado de: https://www.practicalnetworking.net/series/cryptography/diffie-hellman/

Elementos


Anel de chave pública de Bob
Joy
Ted
Mike
Alice
PU_a
Chave pública de Alice
PR_a
Chave privada de Alice
X
Texto cifrado transmitido
$Y = E[PU_a, X]$
$X = D[PR_a, Y]$
Entrada de texto claro
Algoritmo de encriptação (p. ex., RSA)
Algoritmo de decriptação
Saída de texto claro
Bob
(a) Encriptação com chave pública
Alice

Requisitos

A ideia é construir uma função que tenha a seguinte propriedade:

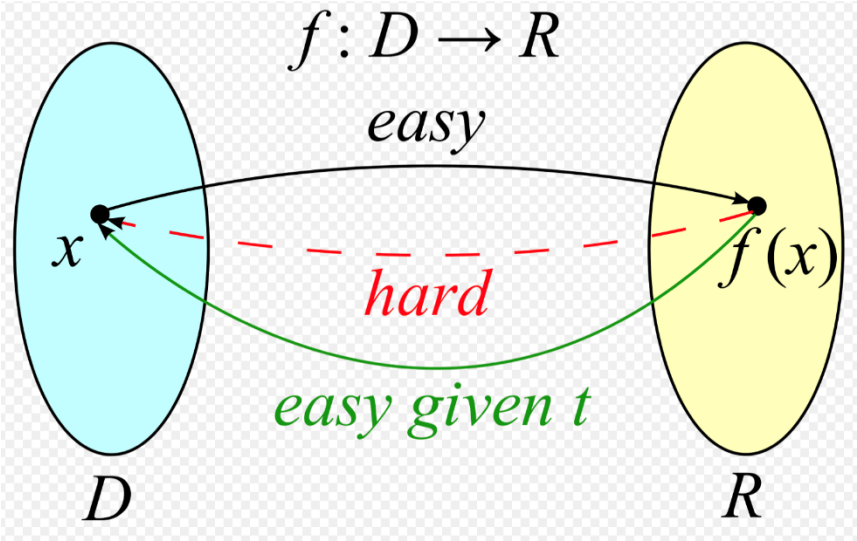
- $C = E(PU_b, M)$ – fácil de calcular
- $M = D(PR_b, C)$ – fácil de calcular, se eu souber PR_b
  - Note que D é a inversa de E. A notação poderia ser E=f e $D = f^{-1}$

- $M = D(?, C)$ – inviável se eu não souber PR_b

Requisitos

- A função f deve ser uma *trapdoor function*;

- Fácil de se calcular em uma direção e inviável na outra, a menos que certa informação adicional seja conhecida.



Tópicos da aula

Segurança da Informação– GBC083

Tópicos da aula – Criptografia de chave pública

1. Ideia
2. Descrição do algoritmo
3. Por que funciona?

Ideia

Segurança da Informação– GBC083

Histórico

- Em 1978, três professores do MIT (Ron Rivest, Adi Shamir e Len Adleman) publicaram uma proposta de um algoritmo de criptografia de chave pública;

- O algoritmo, teoricamente, satisfazia as cinco condições de funcionamento de criptografia de chave pública.

Requisitos

1. É computacionalmente fácil* para uma entidade B gerar um par (PU_b PR_b);

2. É computacionalmente fácil* que um emissor A, conhecendo a chave pública e a mensagem a ser cifrada, *M*, gere o texto cifrado: $C = E(PU_b, M)$;

3. É computacionalmente fácil* que o receptor B decifre o texto cifrado C usando a sua chave privada;

* Fácil significa um problema que pode ser resolvido em tempo polinomial como função do tamanho da entrada.

Requisitos

4. É computacionalmente inviável* que um atacante, conhecendo a chave pública, PU_b, determine a chave privada PR_b;

5. É computacionalmente inviável* que um atacante, conhecendo a chave pública, PU_b e um texto cifrado C recupere a mensagem original M.

* Um problema é inviável se o esforço para solucioná-lo aumentar mais rapidamente do que o tempo polinomial em função do tamanho da entrada

Ideia

- A função f, *trapdoor*, escolhida foi baseada em uma série de argumentos de uma área da matemática conhecida como *teoria dos números*;

- Tal área estuda propriedades de números inteiros como divisibilidade, primalidade e fatoração.

Ideia

Relembrando... A função f, *trapdoor:*

- $C = E(PU_b, M)$ – fácil de calcular
- $M = D(PR_b, C)$ – fácil de calcular, se eu souber PR_b
  - Note que D é a inversa de E. A notação poderia ser E=f e $D = f^{-1}$

A cifragem e decifragem no RSA devem ser feitas da seguinte forma:

- $C = M^e \bmod n$
- $M = C^d \bmod n$
- Ou seja, e seria a chave pública e *d* a chave privada;
- O grande "truque" é definir uma função tal que i) e,*d* sejam inversos e ii) dado somente e, é inviável para um atacante descobrir *d*.

Descrição do algoritmo

Segurança da Informação– GBC083

Características

- RSA é uma cifra de bloco em que o texto claro e o cifrado são inteiros entre 0 e n-1 para algum n;

- Até 2015, um tamanho típico para n era 1024 bits, ou 309 dígitos decimais (1+(1024)*log(2));
  - Desde 2015, o NIST recomenda que o tamanho de n seja 2048 bits.
  - 1+(2048)*log(2) = aproximadamente 617 dígitos!

Descrição do algoritmo

- O RSA pode ser descrito em três etapas:
  1. Criação das chaves;
  2. Cifrar;
  3. Decifrar.

Descrição do algoritmo – Geração de chaves

Lembrem-se que precisamos encontrar *e*,*d* tal que:

$C = M^e \bmod n$
$M = C^d \bmod n$

A grande tarefa aqui será entender as características de *e*,*d*.

Descrição do algoritmo – Geração de chaves – Alguns comentários…

$C = M^e \bmod n$
$M = C^d \bmod n$

Na prática isso significa que se eu cifrar um número elevando a *e*-ésima potência módulo n, eu consigo recuperar tal número elevando o resultado a *d*-ésima potência módulo n.

Descrição do algoritmo – Geração de chaves – Alguns comentários…

- 2^6 mod 10 = 4 (resto da divisão de 64 por 10 é 4)
- Se M = 2 e C = 4 será que não existe algum número tal que:
- 4^x mod 10 = 2?
- Ou seja, será que eu consigo recuperar M à partir do texto cifrado? Quem deveria ser x?

Descrição do algoritmo – Geração de chaves

Geração de chave por Alice

Selecione p, q	p e q são primos, $p \neq q$
Calcule $n = p \times q$	
Calcule $\phi(n) = (p - 1)(q - 1)$	
Selecione o inteiro e	$\text{mdc}(\phi(n), e) = 1; 1 < e < \phi(n)$
Calcule d	$d \equiv e^{-1} \pmod{\phi(n)}$
Chave pública	$PU = \{e, n\}$
Chave privada	$PR = \{d, n\}$

Descrição do algoritmo – Geração de chaves – Alguns comentários…

Geração de chave por Alice

Selecione p, q	p e q são primos, $p \neq q$
Calcule $n = p \times q$	
Calcule $\phi(n) = (p-1)(q-1)$	
Selecione o inteiro e	$\text{mdc}(\phi(n), e) = 1; 1 < e < \phi(n)$
Calcule d	$d \equiv e^{-1} \pmod{\phi(n)}$
Chave pública	$PU = \{e, n\}$
Chave privada	$PR = \{d, n\}$

Quem é $\phi(n)$? É a ***função totiente de Euler***, que é definida como o número de inteiros positivos menores que *n* e relativamente primos de *n*. Ela vai ajudar a encontrar a relação entre e, *d*.

Quem é $\phi(n)$ se n for um primo p?

$\phi(p) = p - 1$

É possível demonstrar que se n=p*q, $\phi(n) = \phi(pq) = \phi(p) * \phi(q) = (p-1) * (q-1)$

Descrição do algoritmo – Geração de chaves – Alguns comentários...

Geração de chave por Alice

Selecione p, q	p e q são primos, $p \neq q$
Calcule $n = p \times q$	
Calcule $\phi(n) = (p-1)(q-1)$	
Selecione o inteiro e	$\text{mdc}(\phi(n), e) = 1; 1 < e < \phi(n)$
Calcule d	$d \equiv e^{-1} \pmod{\phi(n)}$
Chave pública	$PU = \{e, n\}$
Chave privada	$PR = \{d, n\}$

Lembrando que:

$C = M^e \bmod n$
$M = C^d \bmod n$

Para que a relação acima seja satisfeita, é preciso que e,d sejam inversos multiplicativos. É possível demonstrar que se

$$ed \equiv 1 \ (mod\ \phi(n))$$

então a relação acima é verdadeira. A demonstração envolve propriedades de aritmética modular e dois teoremas. Um deles é conhecido como o Pequeno Teorema de Fermat e o outro como Teorema de Euler.

Descrição do algoritmo – Geração de chaves – Alguns comentários...

Geração de chave por Alice	
Selecione p, q	p e q são primos, $p \neq q$
Calcule $n = p \times q$	
Calcule $\phi(n) = (p-1)(q-1)$	
Selecione o inteiro e	$\text{mdc}(\phi(n), e) = 1; 1 < e < \phi(n)$
Calcule d	$d \equiv e^{-1} \pmod{\phi(n)}$
Chave pública	$PU = \{e, n\}$
Chave privada	$PR = \{d, n\}$

$$ed \equiv 1 \ (mod\ \phi(n))$$

Para que a relação acima seja verdadeira, e,d devem ser co-primos com $\phi(n)$.

Por exemplo, considere e=3 e $\phi(n) = 32$. Qual seria o inverso multiplicativo de e=3? Ou seja, qual deve ser o valor de d para que e*d seja congruente a 1 mod 32?

Descrição do algoritmo – Geração de chaves – Alguns comentários...

Geração de chave por Alice

Selecione p, q	p e q são primos, $p \neq q$
Calcule $n = p \times q$	
Calcule $\phi(n) = (p-1)(q-1)$	
Selecione o inteiro e	$\text{mdc}(\phi(n), e) = 1; 1 < e < \phi(n)$
Calcule d	$d \equiv e^{-1} \pmod{\phi(n)}$
Chave pública	$PU = \{e, n\}$
Chave privada	$PR = \{d, n\}$

$$ed \equiv 1 \ (mod\ \phi(n))$$

Para que a relação acima seja verdadeira, e,d devem ser co-primos com $\phi(n)$.

Por exemplo, considere e=3 e $\phi(n) = 32$. Qual seria o inverso multiplicativo de e=3? Ou seja, qual deve ser o valor de d para que e*d seja congruente a 1 mod 32?

Se d = 11, teremos que 3*11 = 33.
33 mod 32 = 1... Achamos o inverso de e! Note que para isso acontecer, e=3 e d=11 precisam ser co-primos com 32.

Descrição do algoritmo – Cifrar/Decifrar

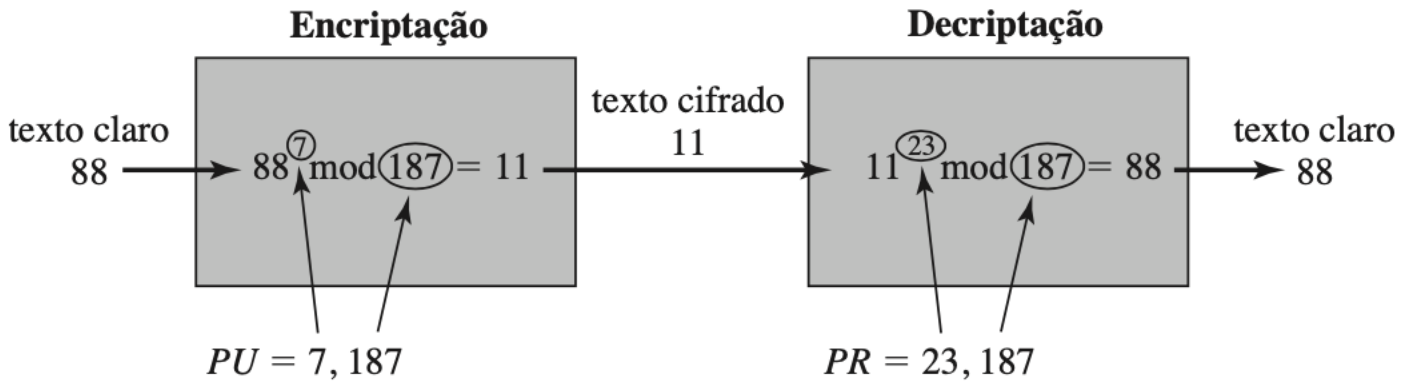
Encriptação por Bob com chave pública de Alice	
Texto claro:	$M < n$
Texto cifrado:	$C = M^e \bmod n$

Decriptação por Alice com a chave privada de Alice	
Texto cifrado:	C
Texto claro:	$M = C^d \bmod n$

RSA - Exemplo

1. Selecione dois números primos, $p = 17$ e $q = 11$.
2. Calcule $n = pq = 17 \times 11 = 187$.
3. Calcule $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Selecione e, tal que e seja relativamente primo de $\phi(n) = 160$ e menor que f(n); escolhemos e = 7.
5. Determine d, tal que $de \equiv 1 \pmod{160}$ e $d < 160$. O valor correto é $d = 23$, pois $23 \times 7 = 161 = (1 \times 160) + 1$; d pode ser calculado usando o algoritmo de Euclides estendido.
6. Ou seja, PU = (7,187) e PR = (23,187).

RSA - Exemplo


Encriptação
Decriptação
texto claro
88
$88^{7} \bmod 187 = 11$
texto cifrado
11
$11^{23} \bmod 187 = 88$
texto claro
88
$PU = 7, 187$
$PR = 23, 187$

Por que funciona?

Segurança da Informação– GBC083

Funcionamento e segurança do RSA

- Por que o RSA funciona?
- Onde está o segurança do RSA?
- Como deve ser feita a escolha dos parâmetros?
- Até quando podemos confiar em tal algoritmo?

Por que o RSA funciona?

- O par (*n*,e) é acessível a qualquer usuário ou seja, o RSA só será seguro se for difícil calcular *d* quando apenas (*n*,e) são conhecidos;

- Para encontrar d, precisamos aplicar o algoritmo euclidiano estendido a $\phi(n)$e e, porém para encontrar $\phi(n)$ precisamos conhecer *p* e *q*, ou seja, só podemos "quebrar" o RSA se soubermos fatorar *n*.
  - O que acontece se n for muito grande?

Por que o RSA funciona?

- O par (*n*,*e*) é acessível a qualquer usuário ou seja, o RSA só será seguro se for difícil calcular *d* quando apenas (*n*,*e*) são conhecidos;

- Para encontrar d, precisamos aplicar o *algoritmo euclidiano estendido* a $\phi(n)$ e e, porém para encontrar $\phi(n)$ precisamos conhecer *p* e *q*, ou seja, só podemos quebrar o RSA se soubermos fatorar *n*.
  - O que acontece se n for muito grande?
  - Nenhum algoritmo atualmente disponível/publicado consegue fatorar números em tempo polinomial (acredita-se que esse seja um problema NP-completo);
  - Um número de 768 bits formado por dois fatores primos, foi fatorado com sucesso no fim de 2009 – pesquisadores levaram 2 anos para se fazer isso!

Escolha dos parâmetros

- Além do tamanho n, outros parâmetros precisam ser escolhidos com cuidado:

1. *p* e *q* deverão diferir em tamanho por apenas alguns dígitos;
2. Tanto (p – 1) quanto (q – 1) deverão conter um fator primo grande;
3. mdc(p – 1, q – 1) deverá ser pequeno;
4. se e < n e *d* < $n^{1/4}$, então *d* pode ser determinado com facilidade.

Escolha dos parâmetros

Como escolher p e q?

Escolha dos parâmetros

Como escolher p e q?

Existe alguma técnica para "gerar" números primos?

Escolha dos parâmetros

Como escolher p e q?

Existe alguma técnica eficiente para "gerar" números primos?

Resposta: não! A ideia é escolher um número ímpar e testar se ele é primo...

Existem diversos testes de primalidade que podem ser usados para escolher p e q.

A grande maioria dos testes implementados na prática são probabilísticos*... Alguém sabe o porquê disso?

* Algoritmo de Miller-Rabin - https://www.geeksforgeeks.org/primality-test-set-3-miller-rabin/

Escolha dos parâmetros

Será que existe algum algoritmo determinístico eficiente para mostrar que um dado número é primo?

Escolha dos parâmetros

Será que existe algum algoritmo determinístico eficiente para mostrar que um dado número é primo?

Sim! Em 2004* três pesquisadores indianos propuseram um algoritmo determinístico que define com eficiência se um dado número é primo. Contudo, outras técnicas probabilísticas continuam sendo usadas...

* Agrawal, Manindra, Neeraj Kayal, and Nitin Saxena. "PRIMES is in P." *Annals of mathematics* (2004): 781-793.

Até quando podemos confiar no RSA?

- A maioria das discussões da criptoanálise do RSA tem focado na tarefa de fatorar n em seus dois fatores primos;
  - Com os algoritmos atualmente conhecidos, econtrar *d*, dados e e *n*, parece ser pelo menos tão demorado quanto o problema de fatoração.

- Logo, podemos usar o **desempenho da fatoração** como um *benchmark* contra o qual avaliaremos a segurança do RSA.

Até quando podemos confiar no RSA?

NÚMERO DE DÍGITOS DECIMAIS	NÚMERO DE BITS	DATA EM QUE FOI ALCANÇADO
100	332	abril de 1991
110	365	abril de 1992
120	398	junho de 1993
129	428	abril de 1994
130	431	abril de 1996
140	465	fevereiro de 1999
155	512	agosto de 1999
160	530	abril de 2003
174	576	dezembro de 2003
200	663	maio de 2005
193	640	novembro de 2005
232	768	dezembro de 2009

A segurança pelo tamanho das chaves (Recomendação Ecrypt/2011)

Date	Security Strength	Symmetric Algorithms	Factoring Modulus	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash (A)	Hash (B)
Legacy [1]	80	2TDEA	1024	160	1024	160	SHA-1 [2]	
2019 - 2030	112	(3TDEA) [3] AES-128	2048	224	2048	224	SHA-224 SHA-512/224 SHA3-224	
2019 - 2030 & beyond	128	AES-128	3072	256	3072	256	SHA-256 SHA-512/256 SHA3-256	SHA-1 KMAC128
2019 - 2030 & beyond	192	AES-192	7680	384	7680	384	SHA-384 SHA3-384	SHA-224 SHA-512/224 SHA3-224
2019 - 2030 & beyond	256	AES-256	15360	512	15360	512	SHA-512 SHA3-512	SHA-256 SHA-512/256 SHA-384 SHA-512 SHA3-256 SHA3-384 SHA3-512 KMAC256

- https://www.keylength.com/en/4/

Roteiro de estudos

1. Leitura da seção 9.2 do livro "Criptografia e segurança de redes. Princípios e práticas". William Stallings;
2. Estudo da vídeo-aula referente ao tópico 10;
3. Resolução do TP-5.
4. Referências complementares:
   - https://medium.com/@jryancanty/understanding-cryptography-with-rsa-7472135033 1f
   - https://medium.com/dataseries/rsa-cryptography-behind-the-scenes-fee1e1389f7f