Algoritmos criptográficos assimétricos

- **Histórico -**

A descoberta em 1976 por Diffie, Hellman e Merkle do primeiro "proto" algoritmo criptográfico assimétrico, que prescinde de transmissão sigilosa prévia, abriu caminho para a descoberta de algoritmos criptográficos assimétricos "plenos", com propriedades de robustez baseadas em controle de custo para

1. Deduzir a mensagem ou uma das chaves a partir do criptograma;
2. Deduzir uma das chaves de cifra a partir da chave inversa desta;

permitindo o desenvolvimento da criptografia moderna, onde os mecanismos de distribuição de uma das chaves do par, autenticação de mensagens, e provas de identidade alcançaram novos níveis de versatilidade (essenciais em redes abertas), ao prescindirem de transmissão sigilosa prévia para habilitar seu uso eficaz.

Esquemas e protocolos que fazem uso de algoritmos assimétricos podem, assim, dispensar a premissa de sigilo para uma das chaves do par. Os que usam esta opção são chamados protocolos ou (cripto)sistemas *de chave pública***.**

- **Sistemas de chave pública -**

Esses sistemas são projetados para resistir a ataques de texto pleno escolhido, mas podem ser sensíveis a ataques de criptograma escolhido. Portanto, nos sistemas onde as funções de cifra indexadas pelo par de chaves assimétricas são comutativas, os dois serviços possíveis (de assinatura e de cifragem) devem ser usados com pares distintos de chaves pública/privada.

Dos algoritmos assimétricos até hoje propostos, apenas três – com variantes de um deles – tem se mostrados seguros e práticos para ambos serviços: Estes são o **RSA** (páginas C-2 a 6), **ElGamal** (e sua generalização conhecida como "esquema meta-ElGamal", páginas C-7 e 8), **Rabin** (Cap. 5). Na "família **ECC**" (*Eliptic Curve Criptography*) **–** que são variantes do El-Gamal em aritméticas específicas (esboço da aritmética de Curvas Elípticas no Anexo A-12) – existem algoritmos não comutativos mais eficientes para um dos serviços (assinatura digital). Outros pouco práticos, por serem frágeis ou lentos, não são citados aqui.

RSA

O mais usado e fácil de implementar dos algoritmos assimétricos, conhecido pela sigla das iniciais dos descobridores, Rivest, Shamir & Adleman. Resiste a mais de vinte anos de criptoanálise, com sua robustez baseada no custo computacional de se fatorar números inteiros (o módulo **n**).

Geração de parâmetros e par de chaves do sistema: {t = tamanho}

p = geraprimo(t,j) {j = precisão}
q = geraprimo(t,j)
ϕ = (p-1)*(q-1) **{p, q, ϕ secretos}**
n = p*q; e = rand(t)
e = e / mdc(e,ϕ) [>1] $\mathbf{E_A = (e,n)}$
d = euclext(e,ϕ,1) $\mathbf{D_A = (d,n)}$

Cifragem (começa com E_A pública)

$c = m^e \bmod n$ **{encripta bloco}**
$m = c^d \bmod n$ **{decripta bloco}**

Assinatura (começa com D_A privada)

$s = h(m)^d \bmod n$ **{assina hash}**
$h(m) = s^e \bmod n$? **{verifica hash}**

$\mathbf{d = e^{-1} \bmod \phi}$: *A outra chave de um par é formada pelo mesmo módulo e a inversa do expoente da primeira no anel $Z_{\phi(n)}$, calculada pelo algoritmo Euclides estendido*:

Algoritmo de Euclides extendido recursivo: Dados a, b, c onde mdc(a,b) divide c, o algoritmo retorna o menor x>0 satisfazendo

/* a*x = c mod b */

```
euclext(a, b, c)  begin
    r = b mod a
    se r == 0
       então retorne( (c div a) mod (b div a) )
       senão retorne( (euclext(r,a,-c)*b+c) div a mod b)
end
```

Fermat: *O algoritmo funciona como cifra devido ao Teorema de Euler-Fermat*:

$$c^d = (m^e)^d = m^{1+r(p-1)(q-1)} = m * m^{r(p-1)(q-1)} = m * 1 \bmod n = m$$

ou seja, as funções de cifra $\mathbf{E_A}( )$ e $\mathbf{D_A}( )$ (indexadas pelo par de chaves do titular) são inversas uma da outra supondo que **m** tem menos bits que **n**, o que sempre ocorre nos esquemas de envelope e de assinatura digital (e protocolos onde estes são úteis).

Análise do RSA

• Premissas para a robustez do algoritmo -

1. Qualquer dos parâmetros **p**, **q** e **ϕ(n)** permite o cálculo trivial de **d** em $\mathbf{D_A}$ a partir de $\mathbf{E_A}$, devendo portanto serem protegidos (sigilo) juntamente com $\mathbf{D_A}$.
2. O ataque mais eficiente (conhecido) ao algoritmo consiste em tentar fatorar **n** para se obter **ϕ(n)** e saber em que anel inverter **e** (divulgado em $\mathbf{E_A}$). Pode-se também tentar adivinhar **ϕ(n)**, mas o custo deste ataque é tão alto quanto o de fatorar **n,** sendo maior ainda o custo de se tentar adivinhar $\mathbf{E_A^{-1}}$.
3. Em princípio, poderia existir um método de ataque mais eficiente ao RSA. Porém tal método serviria também para fatoração de **n**, e o problema da fatoração vem sendo extensamente estudado desde 340 A.C., para o qual o algoritmo mais eficiente conhecido tem complexidade $O(e^{c+|n|^{1/3}\ln(|n|)^{2/3}})$, exponencial.
4. Números randômicos são selecionados como primos, para comporem o módulo **n**, por algum algoritmo de *Monte Carlo* (ex: Leehman). Como existem pseudo-primos – números de Carmichael – que, sem serem primos, passam no teste de Monte Carlo independentemente do número **j** de amostragens pela equação do teorema de Fermat, o teste é feito não com a equação do teorema, mas com a da raiz quadrada desta (teste com símbolos de Jacobi), que têm frequências iguais para resíduos de qualquer módulo não-primo. Um teste inicial com o par de chaves também é importante, pois em qualquer caso o algoritmo é probabilístico.

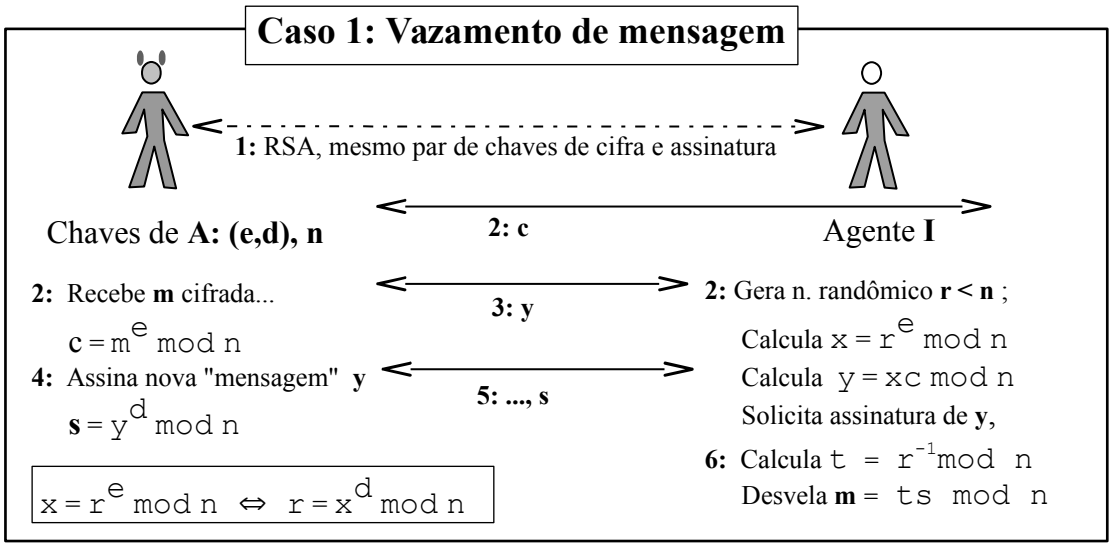
• Ataques a protocolos que usam o RSA -

Métodos conhecidos exploram falhas nos protocolos (não diretamente no algoritmo), devido à exponenciação preservar estruturas multiplicativas através das primitivas computacionais do algoritmo:

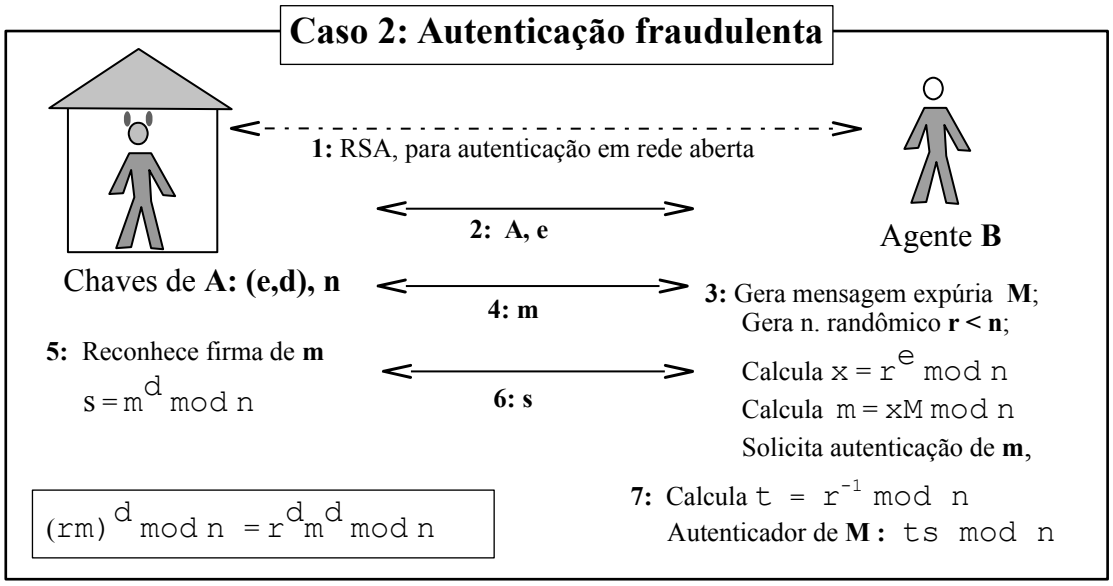
- *Criptograma escolhido contra assinatura;*
- *Módulo comum;*
- *Expoente pequeno para encriptação;*
- *Ordem de operações de cifra e assinatura*.

- **Ataque por criptograma escolhido contra assinatura -**

Este ataque é possível contra protocolos que assinam a mensagem por extenso (e não um hash da mensagem), e prescinde da conivência ou negligência do agente a ser fraudado em assinar mensagens "opacas".



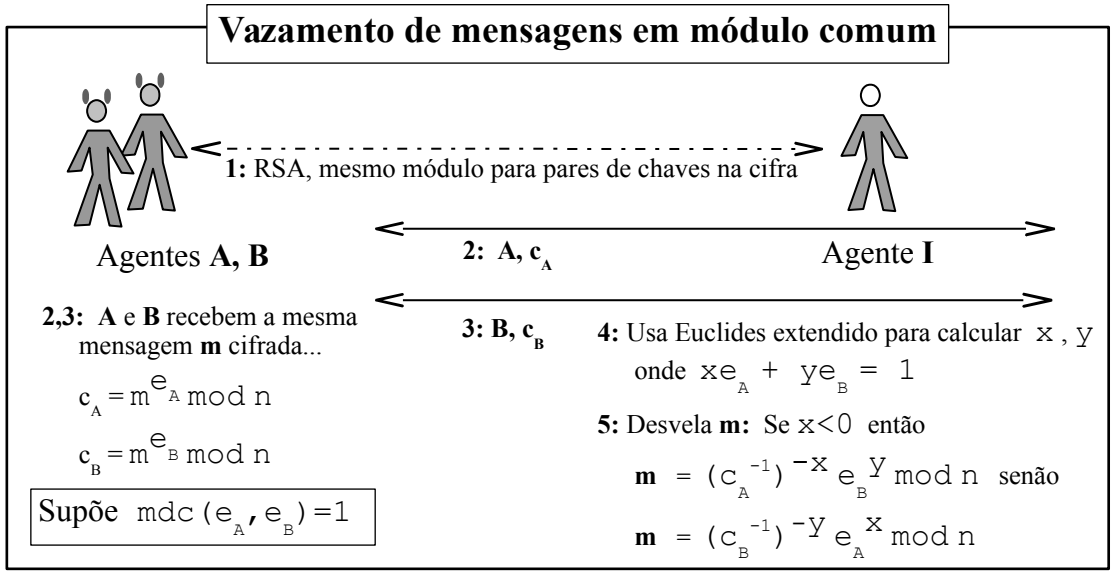
$$ts \bmod n = r^{-1}y^{d} \bmod n = r^{-1}x^{d}c^{d} \bmod n = r^{-1}rc^{d} \bmod n = c^{d} \bmod n = m$$



Conclusão: Para autenticação oponível a terceiros (p.exemplo, em rede aberta), a assinatura deve ser feita sobre um hash da mensagem, e não sobre a mensagem.

- **Ataque em módulo comum -**

Este ataque é possível se a distribuição de chaves para a cifra que usa o RSA atribui chaves com o mesmo módulo a usuários distintos. Qualquer mensagem encriptada por mais de um usuário pode ser facilmente vazada.

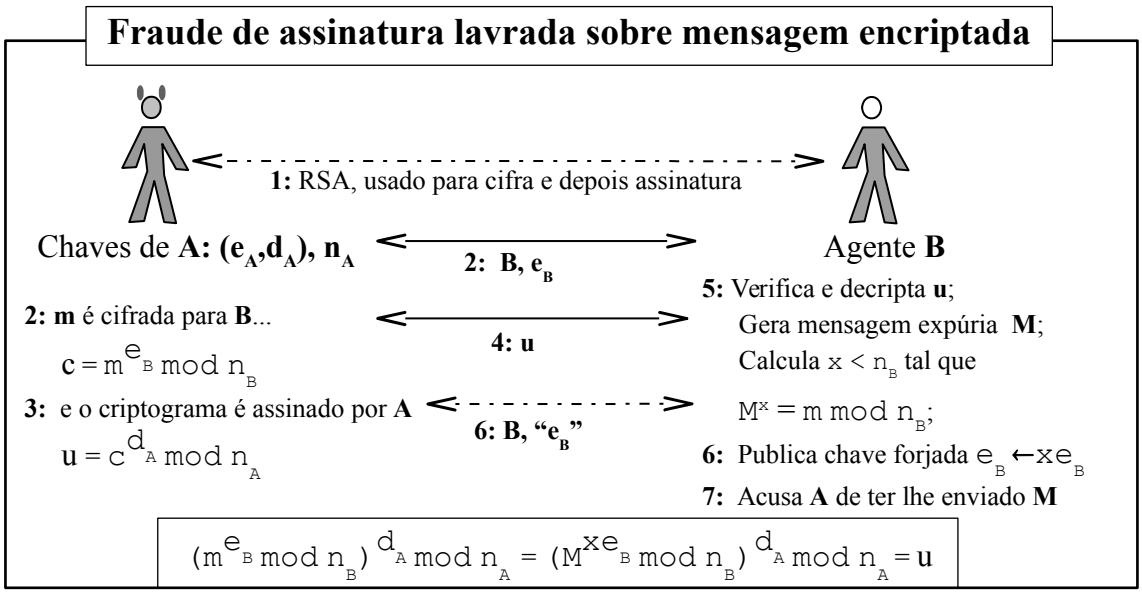


- **Ataque com expoentes pequenos de encriptação -**

Encriptação/verificação de assinatura no RSA é mais rápido quanto menor for a chave pública. Porém, esse tipo de ataque, que usa o algoritmo Poligh-Hellman para fatorar **n**, torna-se possível com a encriptação de $e(e+1)/2$ mensagens linearmente independentes, possivelmente eficiente com e pequeno. **Conclusão**: gerar exponente de chave pública pequeno só deve ocorrer para par de chaves destinados apenas para assinatura digital.

- **Ataque com assinatura sobre mensagem encriptada -**

As operações de assinatura e encriptação devem ser executadas nessa ordem, para evitar fraudes decorrentes deste tipo de ataque, onde nem mesmo o uso de função de hash para indexar o documento na assinatura pode evitar:



Este ataque é possível quando **B** puder "bancar" a solução do problema do logaritmo discreto para encontrar **x**, conhecendo a fatoração de n_B. Se a assinatura antecedesse a cifragem, **B** teria que buscar **x** sem saber fatorar n_A

- **Prevenção contra ataques conhecidos ao RSA -**

1. Conhecimento de um par de expoentes **(e,d)** permite a fatoração do módulo **n**.
2. Conhecimento de um par **(e,d)** permite encontrar outros para mesmo **n**
3. Módulo comum não deve ser usado em rede aberta.
4. Mensagens a serem cifradas devem ser "acolchoadas" (padded) com bits randômicos até completar o tamanho de **n** menos 1 ($m < n$)
5. O expoente público deve ser grande, e a assinatura anteceder a cifragem.

- **Padronização e patentes -**

O RSA é um padrão *de facto* para criptografia assimétrica: Anexo da norma ISO 9796, *draft* de uma norma ANSI, padrão bancário na França e Austrália. Não é padrão nos EUA por problemas de disputa sobre direitos de patente. Esta patente, que é válida somente nos EUA, expirou em 20/9/2000.

ElGamal

Algoritmo assimétrico cuja segurança é derivada da dificuldade de se extrair logaritmos discretos em corpos finitos. (T. ElGamal, 1984).

Geração de parâmetros, chaves assimétricas e chave de sessão:

p = geraprimo(t,j)
g = rand(|p|)
d_A = x= rand(|p|) **{chave privada}**
$y = g^x \bmod p$
e_A = (p,g,y) **{chave pública}**
k_m = rand(|p|)
k = k_m/mdc(k_m,p-1) **{ch. sessão}**
$a = g^k \bmod p$

Assinatura:

b = euclext(k,p-1,m-x∗a)
{(a,b) = assinatura de m }
$(y^a * a^b) \bmod p \stackrel{?}{=} g^m \bmod p$

Cifragem:

$b = (y^k * m) \bmod p$
{(a,b) = criptograma de m }
$m = (b * a^{-x}) \bmod p$

- ## Análise do algoritmo de ElGamal -

Cada assinatura ou encriptação requer um valor randômico para **k**. O conhecimento de pelo menos duas mensagens encriptadas ou assinadas como o mesmo **k** permite a recuperaração da chave privada **x**.

Este algoritmo não é patenteado, mas sua versão para cifragem é uma variante do algoritmo de Diffie-Helmann. A detentora de patente para o D&H (PKP Inc.) reclama direitos para licenciar seu uso (até abril de 1997).

- ## Variantes e generalizações do algoritmo de ElGamal -

Prova de identidade (T. Beth, EUROCRIPT 88);
Derivação de chaves (W. Jaburek, EUROCRIPT 89);
Autentição de senhas (C. Chang & S. Huang, IEEE Carnahan Conf. 91);

Esquema Meta-ElGamal

O Meta-algoritmo de ElGamal é um esquema para se derivar vários algoritmos assimétricos para assinatura digital (*Horster, Petersen & Michels: 1994 ACM computer conference on communications security*).

Geração de parâmetros, chaves assimétricas e chave de sessão:

p=geraprimo(rand( ))
repeat
  q = rand( )
until q | (p-1)
repeat
  g = rand( ) mod p
until $g^q \bmod p = 1$
x = rand( ) mod q **{chave privada}**
$y = g^x \bmod p$
e_A = (p,q,g,y) **{chave pública}**

Assinatura de m = (r,s):

repeat
  k = rand( ) mod q **{ch. sessão}**
until mdc(k,q) = 1 $r = g^k \bmod p$
t = r mod q

equação de assinatura

ak = b + cx mod q

equação de verificação:

$r^a =? g^b y^c \bmod q$

- **Possíveis coeficentes no esquema meta-ElGamal -**

Tabela de possíveis valores dos coeficientes a, b, c		
±t	±s	±m
±tm	±s	1
±tm	±ms	1
±tm	±ts	1
±sm	±ts	1

Exemplos: algoritmos derivados da 1ª linha (+) da tabela acima	
Equação de assinatura	**Equação de verificação**
(1) $mk = s+tx \bmod q$	$r^t = g^s * y^m \bmod q$
(2) $tk = m+sx \bmod q$	$r^t = g^m * y^s \bmod q$
(3) $sk = t+mx \bmod q$	$r^s = g^t * y^m \bmod q$
(4) $sk = m+tx \bmod q$	$r^s = g^m * y^t \bmod q$
(5) $mk = s+tx \bmod q$	$r^m = g^s * y^t \bmod q$
(6) $mk = t+sx \bmod q$	$r^m = g^t * y^s \bmod q$