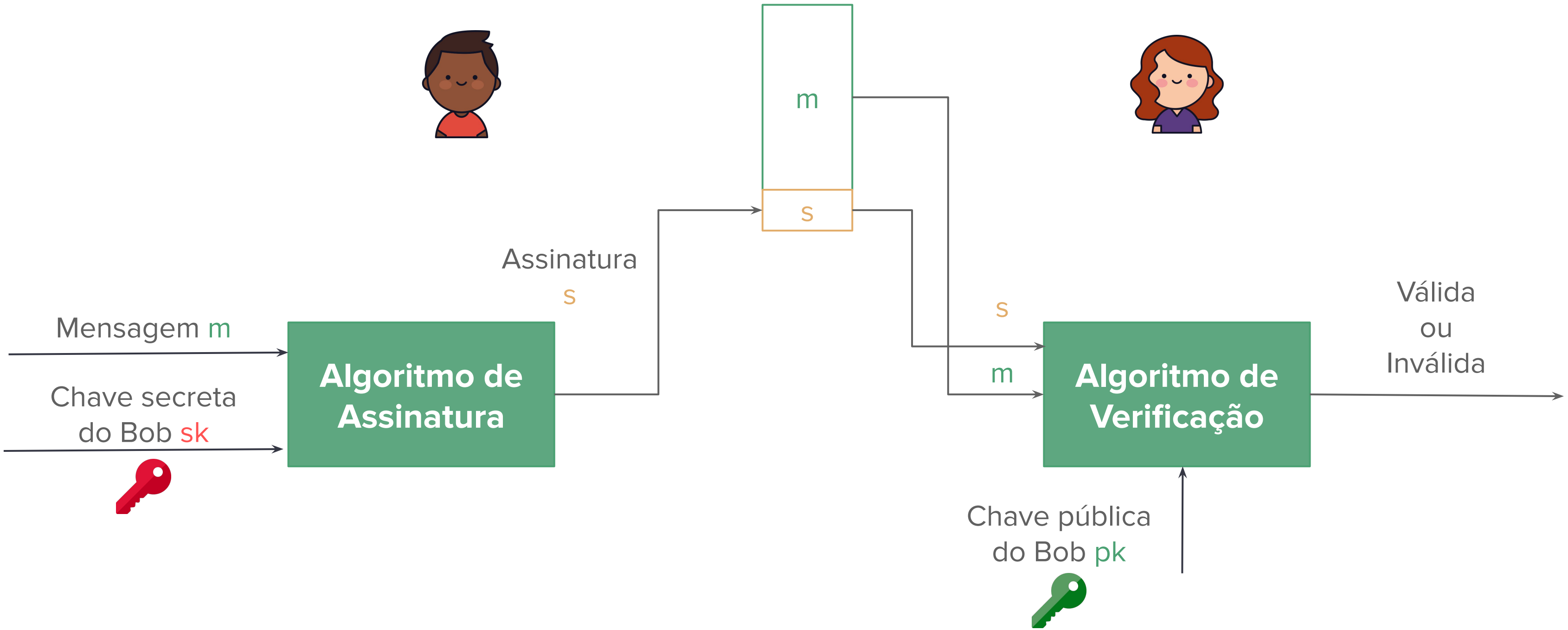
Criptografia Aplicada

Assinaturas digitais - DSA e ECDSA

LabSEC
Laboratório de Segurança em Computação

Assinatura Digital



LabSEC
Laboratório de Segurança em Computação

cpqd

Sumário

- Assinatura digital com DSA
- Assinatura digital com ECDSA
- Assinatura ECDSA na prática

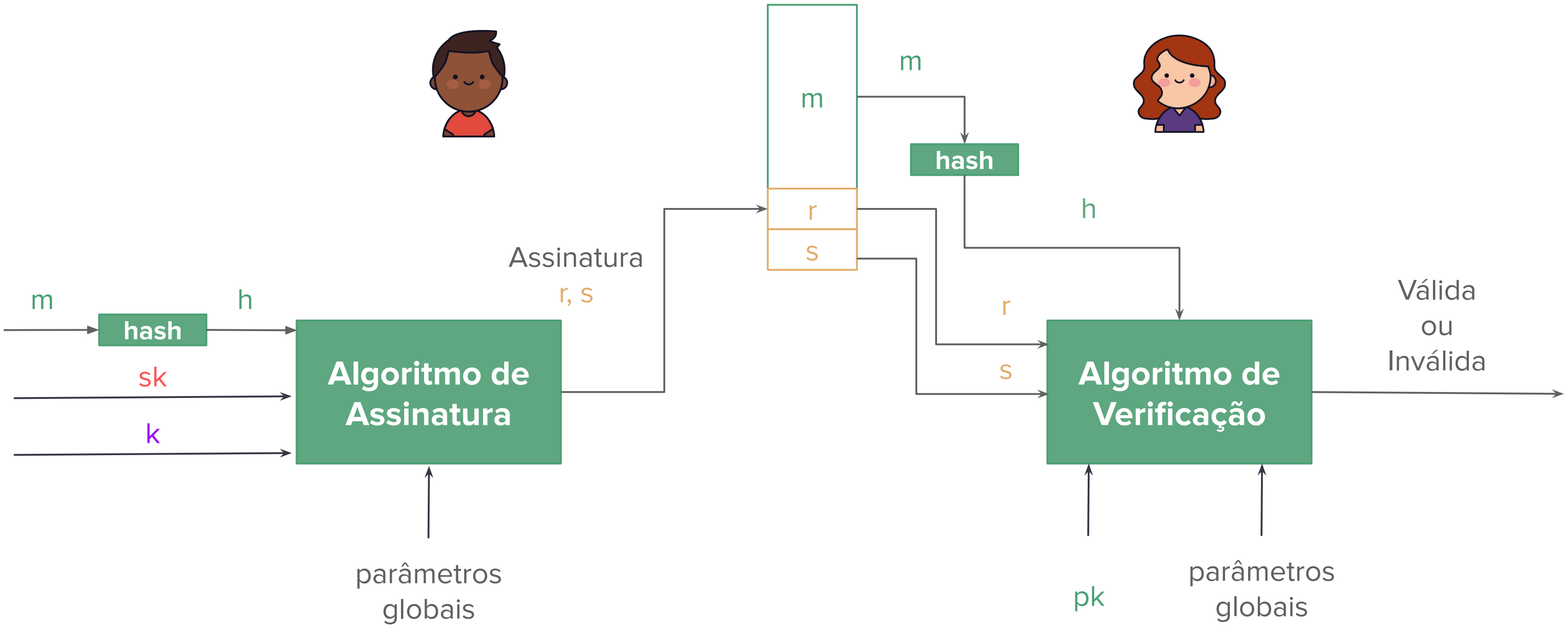
Digital Signature Algorithm (DSA)

- Originalmente proposto em 1991
- Variação do ElGamal Signature Scheme
- Segurança baseada no problema do logaritmo discreto
  - Dado a, b $\in \mathbb{Z}_n$, encontrar x tal que $a^x = b \ mod\ n$
- Hoje em dia, não é mais um algoritmo aprovado para assinaturas digitais
  - porém, ainda pode ser utilizado para verificação de assinaturas antigas
  - ver mais em: Digital Signature Standard (FIPS 186-5)

Digital Signature Algorithm (DSA)

- Desenvolvido para prover somente assinatura
    - Não pode ser usado para cifragem de dados e nem troca de chaves
- A assinatura contêm um valor aleatório k
    - Ou seja, assinar uma mensagem duas vezes provê duas assinaturas diferentes
- A assinatura é formada por um par de valores
    - Sign(m, k) = (r, s)

DSA


m
hash
h
sk
k
Algoritmo de Assinatura
parâmetros globais
Assinatura
r, s
m
r
s
m
hash
h
r
s
Algoritmo de Verificação
pk
parâmetros globais
Válida ou Inválida

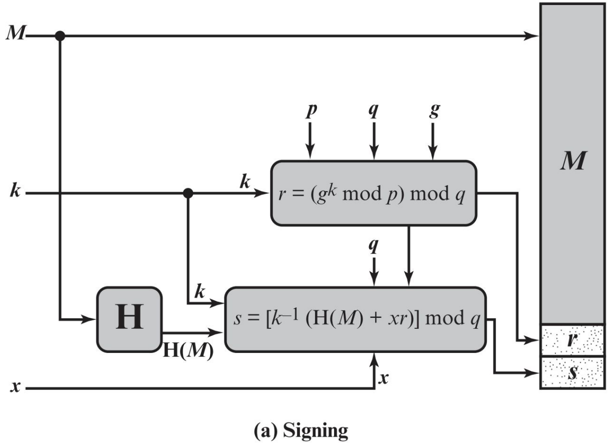
DSA - KeyGen

- **Parâmetros globais**
  - primo *p* de 2048 bits tal que o problema do logaritmo discreto seja difícil em $\mathbb{Z}_p$
  - primo *q* de 224 bits que divide p-1
  - $g = h^{(p-1)/q} \bmod p$, onde h é um inteiro $1 < h < (p-1)$ e g deve ser > 1
- **KeyGen**
  - Chave privada: valor aleatório x, $0 < x < q$
  - Chave pública: $y = g^x \bmod p$
- **Valor secreto**
  - k, $0 < k < q$
  - escolhido de forma que exista inversa multiplicativa k^{-1} módulo q
  - calcular novo valor a cada assinatura

DSA - Sign e Verify

- **Assinatura: (r, s) = Sign(m, x, k)**
  - r = (g^k mod p) mod q
  - s = [k^{-1} (H(m) + xr)] mod q

m: mensagem a ser assinada
H(m): hash de *m*



(a) Signing

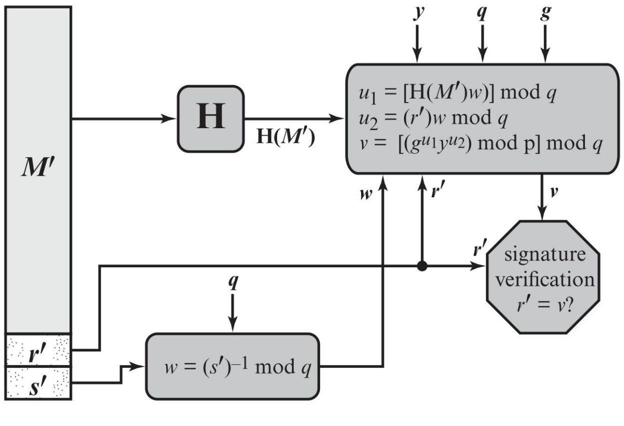
Imagem: W. Stallings. *Cryptography and network security*. Cap 13.4

DSA - Sign e Verify

- **Assinatura: (r, s) = Sign(m, x, k)**
  - $r = (g^k \bmod p) \bmod q$
  - $s = [k^{-1}(H(m) + xr)] \bmod q$

- **Verificação: Verify(m, (r, s), y, g, p, q)**
  - $w = s^{-1} \bmod q$
  - $u1 = [H(m)\ w] \bmod q$
  - $u2 = (r\ w) \bmod q$
  - $v = [(g^{u1}\ y^{u2}) \bmod p] \bmod q$
  - se v = r
    - retorne "válida"
  - senão:
    - retorne "inválida"

m: mensagem a ser assinada
H(m): hash de *m*



(b) Verifying

Imagem: W. Stallings. *Cryptography and network security*. Cap 13.4

LabSEC Laboratório de Segurança em Computação

cpqd

Considerações finais

Corretude:

- Não é óbvio verificar que o valor calculado de v é igual ao valor recebido r
    - existe uma prova de que v pode ser derivado de r
    - Ver em: W. Stallings. *Cryptography and network security, apêndice K.*

Segurança:

- Graças à dificuldade de resolver o logaritmo discreto, é computacionalmente difícil recuperar k a partir de r e x a partir de y
    - ou seja, é difícil falsificar uma assinatura

Eficiência computacional:

- g^k mod p é única parte intensiva no cálculo da assinatura
    - como não depende de *m*, pode ser pré-calculada
    - mais eficiente que RSA

LabSEC Laboratório de Segurança em Computação

Sumário

LabSEC
Laboratório de Segurança em Computação

Relembrando...

- **Curva elíptica**: conjunto de pontos que satisfaz a equação $\mathbf{y^2 = x^3 + ax + b}$
    - Chamamos esse conjunto de pontos de E(a,b)
    - Se nos restringirmos a valores em $\mathbb{Z}_p$, então chamamos de $E_p(a,b)$
- Sejam Q e P pontos em $E_p(a,b)$ e $k < p$
    - é fácil calcular Q dado k e P
    - é difícil determinar k dado Q e P
- Esse é o **problema do logaritmo discreto em curvas elípticas**
    - k é muito grande, tornando força-bruta inviável

LabSEC Laboratório de Segurança em Computação

Elliptic Curve Digital Signature Algorithm (ECDSA)

- Esquema de assinatura digital baseado em curvas elípticas
- Variação do DSA para as curvas elípticas
- Mais recente que o RSA e DSA
  - proposto em 2009 pelo NIST na FIPS 186
- Vantagem de ter chaves muito menores

ECDSA - overview

1. Todos os participantes do esquema precisam utilizar os mesmos parâmetros globais, que definem a curva a ser utilizada.
2. O assinante deve gerar um par de chaves. A chave privada é um número aleatório, enquanto a chave pública é um ponto da curva.
3. O hash da mensagem é assinado usando a chave privada e os parâmetros públicos. A assinatura consiste em dois inteiros r e s.
4. Para verificar a assinatura, o verificador utiliza a chave pública, os parâmetros globais e o valor s. Ele gera um valor v que é então comparado com r.

ECDSA - KeyGen

- **Parâmetros globais**
  - número primo q
  - inteiros a e b que definem a curva $E_q(a,b)$ com equação $y^2 = x^3 + ax + b$
  - ponto $G = (x_g, y_g)$ em $E_q(a,b)$ cuja ordem é um número grande n
    - ordem: menor inteiro *n* tal que $nG = O$
- **KeyGen**
  - Chave privada: valor aleatório d, $0 < d < n$
  - Chave pública: $Q = dG$, onde Q é um ponto em $E_q(a,b)$

ECDSA - Sign e Verify

- **Assinatura: (r, s) = Sign(m, d)**
  - Selecione um inteiro aleatório k, 0 < k < n
  - P = (x,y) = kG
  - r = x mod n.
    - se r = 0, volte ao primeiro passo.
  - s = [k^{-1}(H(m) + dr)] mod n
    - se s = 0, volte ao primeiro passo.
  - Retorne a assinatura (r, s).

- **Verificação: Verify(m, (r, s), a, b, q, n, G, Q)**
  - Verifica que r e s são inteiros entre 1 e n-1
  - $w = s^{-1} \bmod n$
  - $u_1 = H(m)\, w \bmod n$
  - $u_2 = r\, w \bmod n$
  - $X = (x_1, x_2) = u_1 G + u_2 Q$
    - se X = O, rejeite a assinatura
  - $v = x_1 \bmod n$
  - se v = r
    - retorne "válida"
  - senão:
    - retorne "inválida"

Exemplo

- **Parâmetros globais**
  - Considere $q = 11$, $(a,b) = (1, 6)$
  - ponto $G = (2, 7)$ com ordem $n = 13$
- **KeyGen**
  - Chave privada: valor aleatório $d = 7$
  - Chave pública: $Q = dG = (7, 2)$
- **Sign(m, d = 7)**
  - Considere $H(m) = 4$
  - $k = 3$, $P = (x,y) = kG = (8,3)$
  - $r = x \bmod n = 8 \bmod 13 = 8$
  - $s = [k^{-1}(H(m) + dr)] \bmod n$
  - $= 3^{-1}(4 + 7 \times 8) \bmod 13 = 7$
  - $(r, s) = (8, 7)$
- **Verify**(m, (r, s) = (8, 7), a = 1, b = 6, q=11, n=13, G=(2,7), Q=(7,2))
  - $w = 7^{-1} \bmod 13 = 2$
  - $u_1 = H(m)\, w = 4 \times 2 \bmod 13 = 8$
  - $u_2 = r\, w = 8 \times 2 \bmod 13 = 3$
  - $X = (x_1, x_2) = u_1 G + u_2 Q = 8G + 3Q = (8,3)$
  - $v = x_1 \bmod 13 = 8 = r$

Considerações finais

Corretude:

- Não é óbvio verificar que o valor calculado de v é igual ao valor recebido r
    - Ver detalhes em: W. Stallings. *Cryptography and network security, capítulo 13.5.*

Segurança:

- Graças à dificuldade de resolver o logaritmo discreto em curvas elípticas (ECDLP), é inviável recuperar k a partir de r e d a partir de s
    - ou seja, é inviável falsificar uma assinatura
- Um novo valor k precisa ser gerado a cada nova assinatura, que precisa ser mantido em segredo
- Os valores de k e k^{-1} podem ser calculados previamente, desde que mantidos em local seguro assim como a chave privada

Ver mais em: FIPS 186-5

LabSEC Laboratório de Segurança em Computação

Comparação

Table 10.3 Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis (NIST SP-800-57)

Symmetric Key Algorithms	Diffie–Hellman, Digital Signature Algorithm	RSA (size of n in bits)	ECC (modulus size in bits)
80	$L = 1024$ $N = 160$	1024	160–223
112	$L = 2048$ $N = 224$	2048	224–255
128	$L = 3072$ $N = 256$	3072	256–383
192	$L = 7680$ $N = 384$	7680	384–511
256	$L = 15{,}360$ $N = 512$	15,360	512+

Note: L = size of public key, N = size of private key.

Imagem: W. Stallings. *Cryptography and network security*. Cap 10.4

NIST SP 800-57

LabSEC Laboratório de Segurança em Computação

Sumário

- Assinatura digital com DSA
- Assinatura digital com ECDSA
- **Assinatura ECDSA na prática**

Aplicações

- Utilizadas em ambientes com capacidades limitadas de comunicação
- Uso na prática
    - TLS
    - Bitcoin
    - iMessage

Atividade: Gerando chaves com o openssl

- Os comandos abaixo já foram executados na aula de ECC!
  - caso tenha feito, utilize as chaves já geradas

- Verifique as curvas disponíveis no openssl:

```
openssl ecparam -list_curves
```

- Gere a chave privada com a curva escolhida:

```
openssl ecparam -genkey -name <nome_da_curva> -out <nome_chave_privada>.pem
```

- Extraia a chave pública:

```
openssl ec -in <nome_chave_privada>.pem -pubout -out <nome_chave_pública>.pem
```

- Visualize as chaves:

```
openssl ec -in <nome_chave_privada>.pem -text -noout
openssl ec -in <nome_chave_pública>.pem -text -noout -pubin
```

Atividade: assinando com ECDSA

- Crie um arquivo de texto qualquer com uma mensagem
  - ```
    echo "Mensagem autentica" > msg.txt
    ```
- Assine a mensagem usando a chave privada
  - ```
    openssl dgst -sha256 -sign <nome_chave_privada>.pem -out assinatura.bin msg.txt
    ```
- Verifique a assinatura
  - ```
    openssl dgst -sha256 -verify <nome_chave_pública>.pem -signature assinatura.bin msg.txt
    ```
- Modifique a mensagem e verifique a assinatura novamente
    ```

Resumo

- Princípios básicos de assinatura digital
  - requisitos, algoritmos, uso do hash, ataques
- Assinatura digital com RSA
  - primeiro esquema de assinatura digital
  - baseado no problema de fatoração
- Assinatura digital com DSA
  - baseado no problema do logaritmo discreto
  - mais complexo matematicamente, porém mais eficiente que o RSA
- Assinatura em curvas elípticas
  - variação do DSA com curvas elípticas
  - chaves menores, mais eficiente

Referências

- W. Stallings. *Cryptography and network security*. 7a edição.
  - DSA: 13.4
  - ECDSA: 13.5
- D. Stinson e M. Paterson. *Cryptography: Theory and Practice*. 4a edição.
  - DSA: 8.4.2
  - ECDSA: 8.4.3
- Recomendações para gerenciamento de chaves: NIST SP 800-57
- Digital Signature Standard: FIPS 186-5
- imagem: Flaticon.com