

Segurança da Informação

Curvas Elípticas
Emparelhamentos Bilineares

Curvas Elípticas

Grupos

- Criptossistemas baseados no PLD operam numa estrutura algébrica de *grupo*.
- Um grupo (abeliano) G é um conjunto com uma operação de composição:
 - $P + O = P$ (elemento neutro)
 - $P + (-P) = O$ (elemento inverso)
 - $P + (Q + R) = (P + Q) + R$ (associatividade)
 - $P + Q = Q + P$ (comutatividade)

Grupos

- Multiplicação por escalar:
 - $n \cdot P = P + \dots + P$ (n vezes),
 - $0 \cdot P = O$,
 - $(-n) \cdot P = n \cdot (-P)$.
- Propriedades de espaço vetorial:
 - $(m+n) \cdot P = m \cdot P + n \cdot P$,
 - $m \cdot (n \cdot P) = n \cdot (m \cdot P) = (mn) \cdot P$.

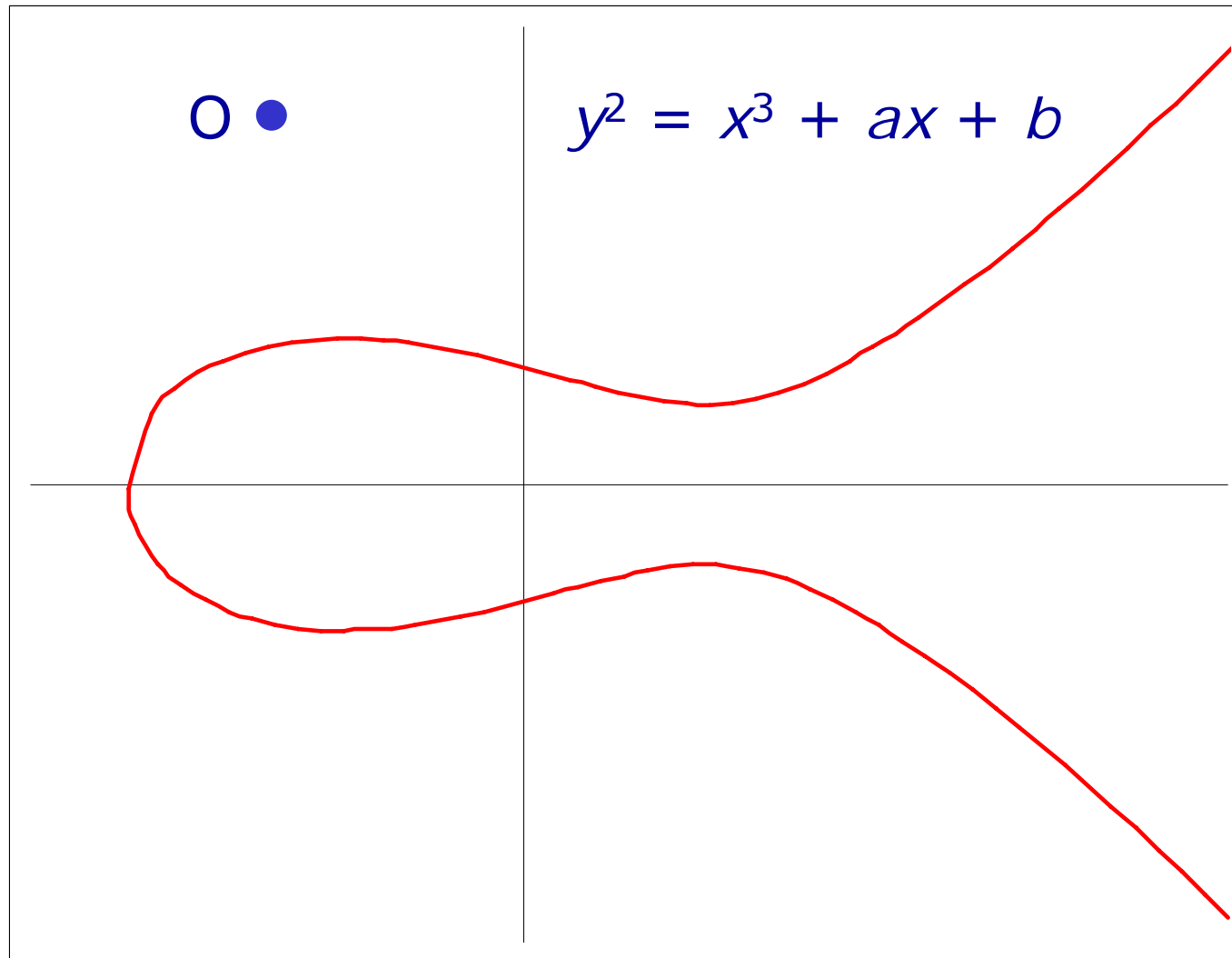
Grupos

- Ordem de P : menor $r > 0$ tal que $r \cdot P = O$.
- Subgrupo gerado por P :
 - $\langle P \rangle = \{m \cdot P \mid m = 0, \dots, r-1\} = \{O, P, 2 \cdot P, \dots, (r-1) \cdot P\}$.
- Logaritmo (ou índice) discreto de $Y \in \langle P \rangle$:
 - $\log_P Y = m \Leftrightarrow m \cdot P = Y$.
- Notação multiplicativa:
 - $a \cdot 1 = a$ (elemento neutro)
 - $a \cdot a^{-1} = 1$ (elemento inverso)
 - $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ (associatividade)
 - $a \cdot b = b \cdot a$ (comutatividade)
 - $a^n = a \cdot \dots \cdot a$ (n vezes), $a^0 = 1$, $a^{-n} = (a^{-1})^n$.

Exemplos de grupos

- Inteiros mod p (\mathbb{Z}_p), operação de adição.
- Inteiros não nulos mod p (\mathbb{Z}_p^*), operação de multiplicação.
- **Curvas elípticas**: conjuntos de pontos no plano (x, y) que satisfazem uma equação da forma $y^2 = x^3 + ax + b$, com um ponto adicional O (ponto no infinito).

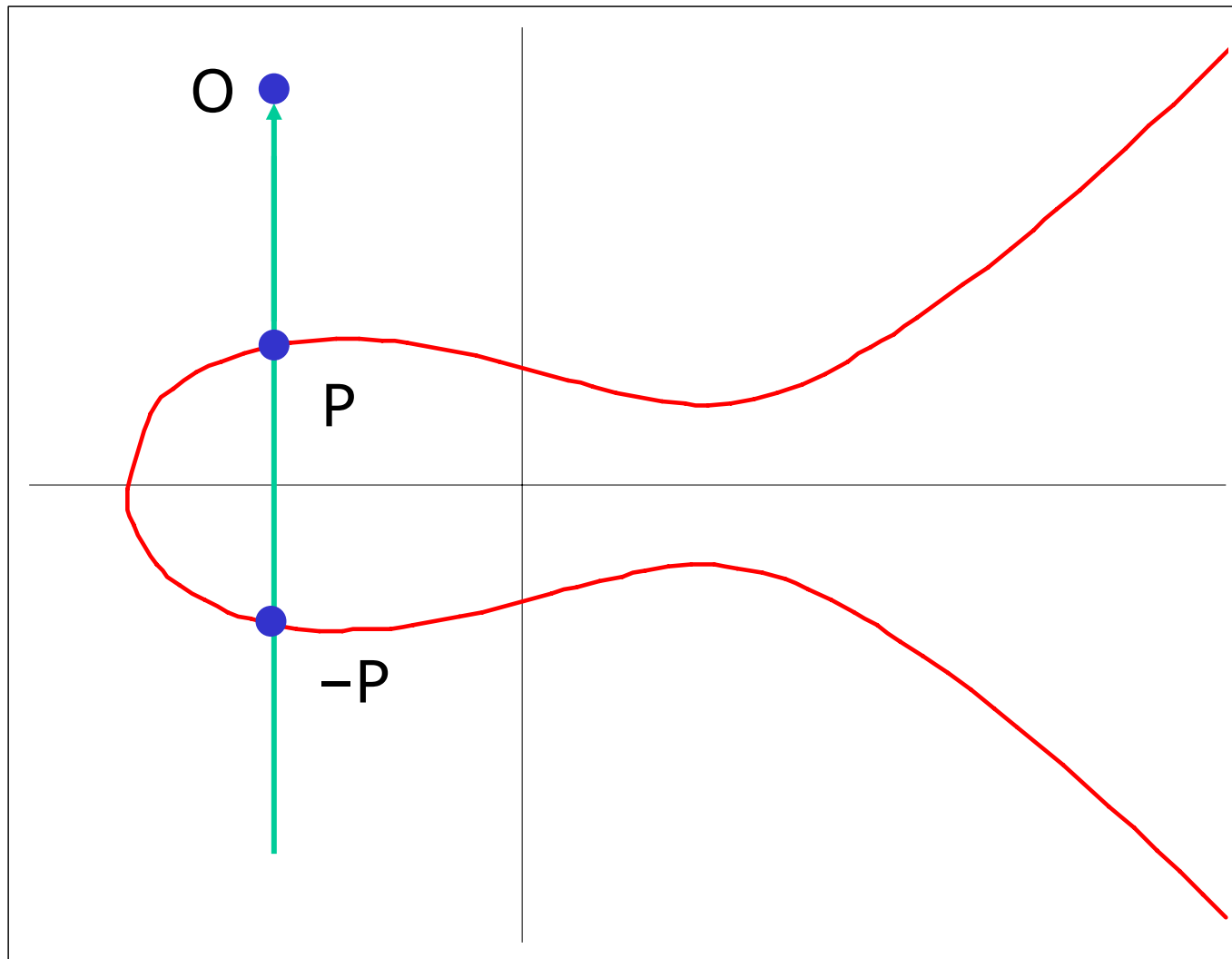
Curvas elípticas



Curvas elípticas

- Dados dois pontos (x_1, y_1) e (x_2, y_2) de uma curva elíptica, em geral o ponto $(x_1 + x_2, y_1 + y_2)$ **não** está sobre a curva.
- É possível definir uma operação de soma de pontos através de uma construção geométrica (secantes e tangentes).
- Fórmulas derivadas dessa construção calculam diretamente as coordenadas do ponto $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$.

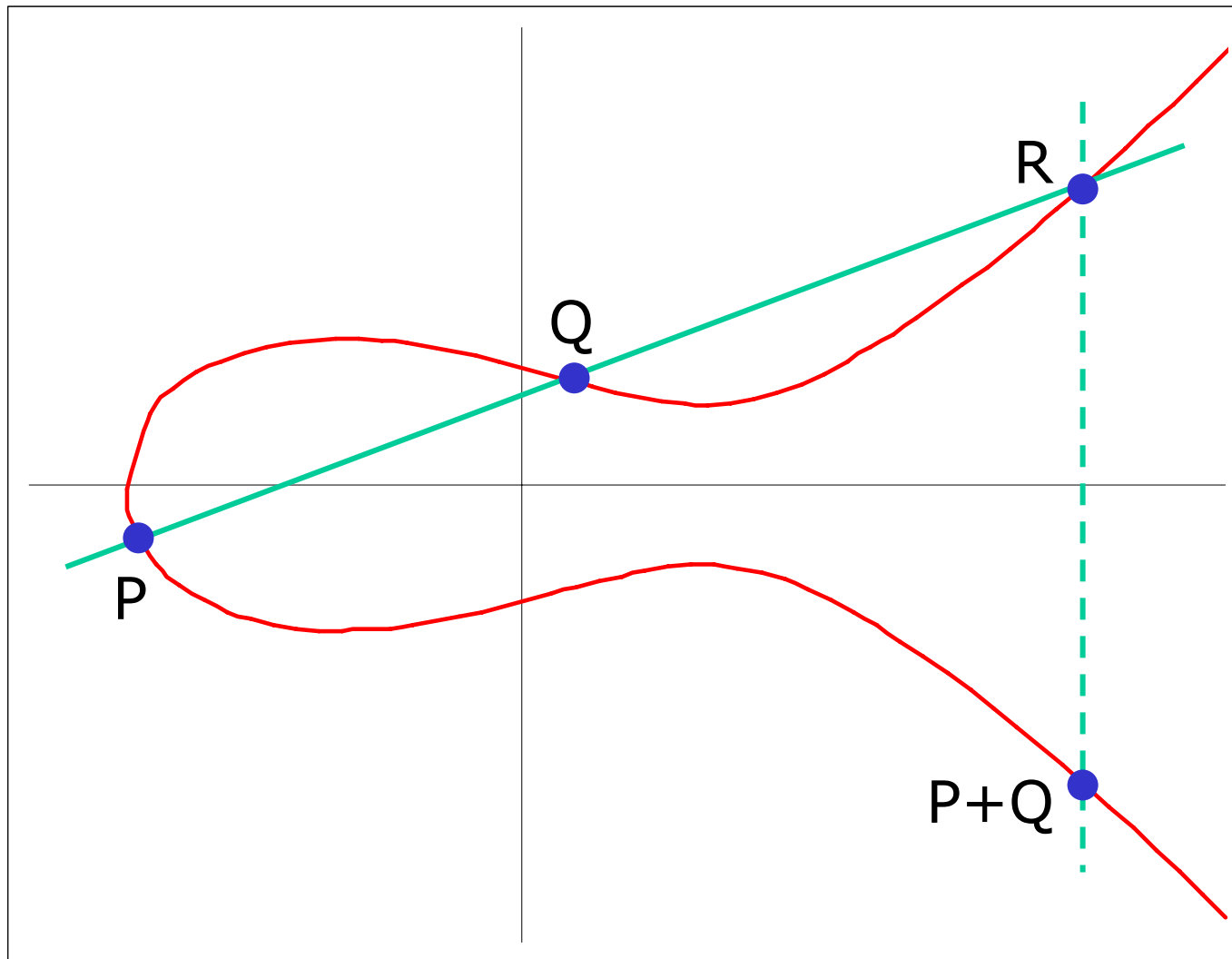
Curvas elípticas



Curvas Elípticas

- $P = (x_1, y_1)$, $-P = (x_1, -y_1)$.
- NB: se $y_1 = 0$, $P = -P$.
- Ponto no infinito:
 - $O = (0, 0)$ se $b \neq 0$
 - $O = (0, 1)$ se $b = 0$.
- NB: estes pontos *não* satisfazem a equação da curva!
 - Artifícios convencionais para representar o ponto O .

Curvas elípticas



Curvas Elípticas

- $P = (x_1, y_1)$, $Q = (x_2, y_2)$:

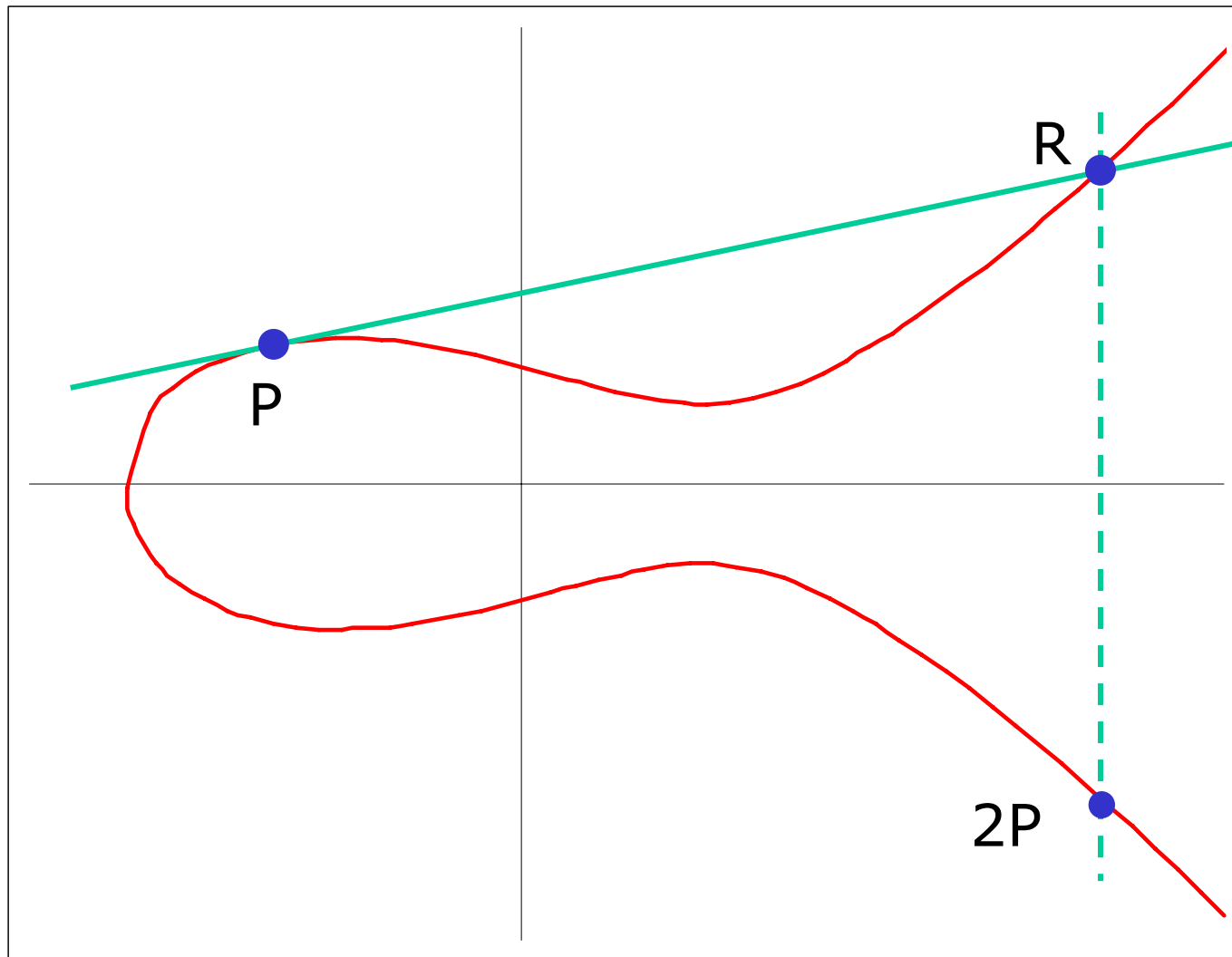
$$\lambda \leftarrow \frac{y_2 - y_1}{x_2 - x_1}$$

$$x_3 \leftarrow \lambda^2 - (x_1 + x_2)$$

$$y_3 \leftarrow \lambda(x_1 - x_3) - y_1$$

- Se $x_1 = x_2$, usa-se a fórmula de duplicação.

Curvas elípticas



Curvas Elípticas

- $P = (x_1, y_1)$, $2P = (x_3, y_3)$:

$$\lambda \leftarrow \frac{3x_1^2 + a}{2y_1}$$

$$x_3 \leftarrow \lambda^2 - 2x_1$$

$$y_3 \leftarrow \lambda(x_1 - x_3) - y_1$$

- Se $y_1 = 0$, $P = -P$ e portanto $2P = O$.

Algoritmo de "exponenciação"

// $x = (x_m x_{m-1} \dots x_1 x_0)_2$, $x_m = 1$.

$Y \leftarrow P$

for $i = m-1, \dots, 0$ {

$Y \leftarrow 2Y$

if $x_i = 1$ {

$Y \leftarrow Y + P$

 }

}

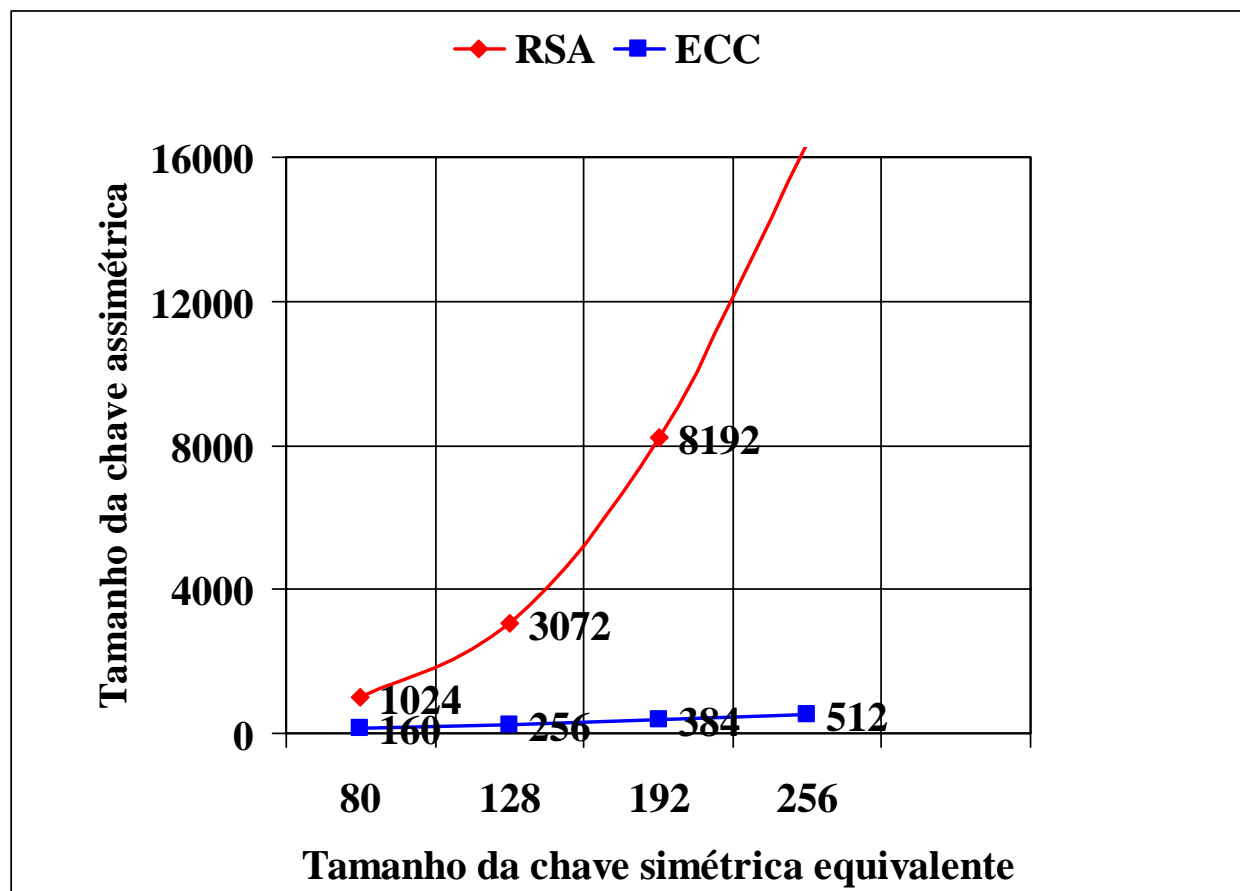
return Y

// $Y = x \cdot P$

Problema do Logaritmo Discreto

- Dado um escalar x e um elemento $P \in G$, é “fácil” calcular $Y = x \cdot P$.
- Dados P e Y , é “difícil” calcular x .
- Com inteiros de k bits, para calcular Y a partir de x e P são necessários $O(k^3)$ passos (esforço *polinomial*).
- Em certos grupos, para calcular x a partir de Y e P são necessários $O(2^k)$ passos (esforço *exponencial*).

Comparação: ECC vs. RSA



Diffie-Hellman

- Objetivo: negociação de chave através de um canal potencialmente inseguro.
- Parâmetro público:
 - Grupo $\langle P \rangle$ de ordem r .
- Chave privada:
 - x : número aleatório mod r .
- Chave pública:
 - $Y = x \cdot P$.

Protocolo Diffie-Hellman

- Alice (A) e Beto (B) desejam estabelecer comunicação segura.
- Possibilidade: usar um algoritmo simétrico para cifrar as mensagens trocadas entre A e B .
- Problema: A e B precisam compartilhar uma chave simétrica.

Protocolo Diffie-Hellman

- Pares de chaves:

$$x_A, Y_A = x_A \cdot P$$

$$x_B, Y_B = x_B \cdot P$$

- A obtém a chave pública de B e vice-versa a partir de uma base de chaves.

- A calcula $x_A \cdot Y_B$
 $= x_A \cdot (x_B P)$
 $= (x_A x_B) \cdot P$

- B calcula $x_B \cdot Y_A$
 $= x_B \cdot (x_A P)$
 $= (x_A x_B) \cdot P$

Curvas Hiperelípticas


- Uma curva hiperelíptica de gênero g é definida por uma equação da forma

$$y^2 + h(x)y = f(x)$$

onde $\deg h(x) \leq g$ e $\deg f(x) = 2g+1$.

- Curvas elípticas são curvas hiperelípticas de gênero $g = 1$.
- Lei de grupo: seqüências de g pontos.

Curvas Hiperelípticas

- Aritmética geralmente implementada com *divisores* em representação de Mumford (pares de polinômios).
- Segurança diminui com o gênero:
 - $g = 1, 2$: OK
 - $g = 3, 4$: aplicações *muito* especiais.
 - $g \geq 5$: 

Emparelhamentos Bilineares

Corpos Finitos

- Corpo $(\mathbb{K}, +, 0, \cdot, 1)$: conjunto com duas operações algébricas. Notação: \mathbb{F}_q , $\text{GF}(q)$.
 - $(\mathbb{K}, +, 0)$ é um grupo abeliano:
 - $a+0 = a$, $a+(-a) = 0$, $a+(b+c) = (a+b)+c$, $a+b = b+a$.
 - $(\mathbb{K}^*, \cdot, 1)$ é um grupo abeliano:
 - $a \cdot 1 = a$, $a \cdot a^{-1} = 1$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$, $a \cdot b = b \cdot a$.
 - $a \cdot (b + c) = a \cdot b + a \cdot c$ (distributividade).

Corpos Finitos

- Propriedade: o número de elementos de um corpo finito, $q = \#\mathbb{K}$, é sempre uma potência de um número primo: $q = p^m$.
- Propriedade (Pequeno Teorema de Fermat): $a^q = a$ (ou $a^{q-1} = 1$).

Exemplos de Corpos Finitos

- \mathbb{F}_p : Inteiros módulo um primo p ($\mathbb{F}_p = \mathbb{Z}_p$).
- \mathbb{F}_{p^m} : Polinômios com m coeficientes em \mathbb{F}_p , módulo um polinômio irreduzível de grau m .
- Notação: $\mathbb{F}_{p^m} = \mathbb{F}_p[X]/v(X)$, $v(X)$ polinômio irreduzível, $\deg(v) = m$.
- AES utiliza aritmética no corpo finito $\text{GF}(2^8) = \mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X + 1)$.

Emparelhamentos

- \mathbb{G}_0 e \mathbb{G}_1 : grupos aditivos, \mathbb{G}_T : grupo multiplicativo, todos de ordem r .
- Um *emparelhamento* é uma função
$$e: \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$$
satisfazendo as propriedades:
 - $e(P, Q) \neq 1$;
 - $e(\alpha P, \beta Q) = e(P, Q)^{\alpha\beta}$, $\forall \alpha, \beta \in \mathbb{Z}$;
 - eficientemente computável.

Assinaturas BLS

- Par de chaves: $(s, V = sQ)$.
- Assinatura:
 - calcular $\Sigma = sH(m)$;
 - mensagem assinada: (m, Σ) .
- Verificação:
 - aceitar $(m, \Sigma) \Leftrightarrow e(\Sigma, Q) = e(H(m), V)$.
- Explicação:
 - $e(\Sigma, Q) = e(sH(m), Q) = e(H(m), Q)^s$.
 - $e(H(m), V) = e(H(m), sQ) = e(H(m), Q)^s$.

Cálculo do emparelhamento

- Só se conhecem emparelhamentos adequados quando \mathbb{G}_0 e \mathbb{G}_1 são curvas elípticas sobre $\text{GF}(q)$ e \mathbb{G}_T é um corpo finito $\text{GF}(q^k)$.
- Em geral, k é enorme \Rightarrow curvas especiais são necessárias:
 - Curvas supersingulares;
 - Curvas MNT;
 - Curvas BN.

Cálculo do emparelhamento

- Cálculo através do algoritmo de Miller ou variantes aperfeiçoadas.
- Idéia: operações adicionais intercaladas no algoritmo de exponenciação.
- “Pequeno” problema...

Algoritmo de Miller

```

//  $r = (r_m \ r_{m-1} \ \dots \ r_1 \ r_0)_2$ ,  $r_m = 1$ .
 $f \leftarrow \text{const}$ ,  $R \leftarrow \text{random}(E)$ ,
 $Y \leftarrow P$ 
for  $i = m-1, \dots, 0$  {
     $f \leftarrow f^2 \cdot g_{V,V}(Q+R) \cdot g_{2V,-2V}(R) / g_{2V,-2V}(Q+R) \cdot g_{V,V}(R)$ ,
     $Y \leftarrow 2Y$ 
    if  $r_i = 1$  {
         $f \leftarrow f \cdot g_{V,P}(Q+R) \cdot g_{V+P,-V-P}(R) / g_{V+P,-V-P}(Q+R) \cdot g_{V,P}(R)$ ,
         $Y \leftarrow Y + P$ 
    }
}
 $z \leftarrow (p^k - 1)/r$ ,  $f \leftarrow f^z$ 
return  $f$  //  $e(P, Q)$ 

```

Algoritmo BKLS/GHS

// $r = (r_m r_{m-1} \dots r_1 r_0)_2$, $r_m = 1$.

$f \leftarrow 1$,

$Y \leftarrow P$

for $i = m-1, \dots, 0$ {

$f \leftarrow f^2 \cdot g_{V,V}(Q)$,

$Y \leftarrow 2Y$

if $r_i = 1$ {

$f \leftarrow f \cdot g_{V,P}(Q)$,

$Y \leftarrow Y + P$

}

}

$z \leftarrow (p^k - 1)/r$, $f \leftarrow f^z$

return f // $e(P, Q)$

Criptografia baseada em identidades

- Algoritmo Boneh-Franklin (2001): primeira instância prática de um criptossistema baseado em identidades (problema proposto por Shamir em 1984).
- Gerador de chaves privadas (GCP): $(s, T = sP)$.
- Função de hash:
 $H: \{0,1\}^* \rightarrow \langle Q \rangle$.
- Cifra simétrica:
 $\varepsilon: GF(q^k) \times \{0,1\}^* \rightarrow \{0,1\}^*$.

Criptografia baseada em identidades

- Extração de chave:
 - $Q_{id} = H(id), D_{id} = sQ_{id}$.
- Encriptação da mensagem M :
 - escolher $u \in \mathbb{Z}_r^*$ aleatório;
 - calcular $N = uP, K = e(T, Q_{id})^u, C = \varepsilon_K(M)$.
 - texto cifrado: (N, C) .
- Decriptação do criptograma (N, C) :
 - calcular $K = e(N, D_{id})$;
 - recuperar $M = \varepsilon_K^{-1}(C)$.
- Explicação:
$$e(T, Q_{id})^u = e(sP, Q_{id})^u = e(uP, sQ_{id}) = e(N, D_{id}).$$

Outros esquemas

- Existem inúmeros tipos de assinaturas digitais baseadas em identidade, esquemas para acordo de chaves, cifrassinatura, quotização de segredo, ...
- Há também sistemas convencionais com propriedades exóticas, incluindo assinaturas de vários tipos, sistemas hierárquicos, controle de acesso, PKC sem certificados e auto-certificada, ...
- Inúmeros problemas em aberto!

Apêndice

Cifrassinatura McCullagh-Barreto
baseada em identidades

Estrutura do esquema

- Composição universal – algoritmos separados para assinatura e cifração.
- Propriedade: emparelhamentos *não* são usados para cifrar e assinar.
- Resultado: algoritmo de cifrassinatura (baseado em identidades) mais eficiente conhecido.
- Segurança “demonstrável” em determinadas parametrizações.

Parametrização

- O Gerador de Chaves Privadas (GCP) escolhe três grupos (e respectivos geradores)

$$\mathbb{G}_0 = \langle P \rangle, \mathbb{G}_1 = \langle Q \rangle, \mathbb{G}_T = \langle g \rangle$$

de mesma ordem prima r que admitam um emparelhamento eficientemente calculável

$$e: \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$$

satisfazendo $g = e(P, Q)$, e três oráculos aleatórios

$$H_0: \{0,1\}^* \rightarrow \mathbb{Z}_r^*,$$

$$H_1: \mathbb{G}_T \rightarrow \{0,1\}^*,$$

$$H_2: \{0,1\}^* \times \mathbb{G}_T \rightarrow \mathbb{Z}_r^*.$$

Setup

- O GCP escolhe uma chave privada

$$s \leftarrow \mathbb{Z}_r^*$$

e calcula duas chaves públicas

$$P_{\text{pub}} = s \cdot P,$$

$$Q_{\text{pub}} = s \cdot Q.$$

Keygen

- O GCP calcula duas chaves privadas para cada identidade ID:

$$K_{ID} = (H_0(ID) + s)^{-1} \cdot Q,$$

$$S_{ID} = (H_0(ID) + s)^{-1} \cdot P.$$

- Duas chaves públicas estão associadas às chaves acima, respectivamente:

$$P_{ID} = H_0(ID) \cdot P + P_{pub},$$

$$Q_{ID} = H_0(ID) \cdot Q + Q_{pub}.$$

Essas chaves públicas podem ser calculadas sob demanda ou pré-calculadas e armazenadas.

Keygen

- Propriedade: $e(P_{ID}, K_{ID}) = e(S_{ID}, Q_{ID}) = g$.
- Obviamente, se $\mathbb{G}_0 = \mathbb{G}_1$ ($P = Q$) apenas um par de chaves poderia ser calculado e usado:
 - $P_{pub} = Q_{pub}$,
 - $P_{ID} = S_{ID}$,
 - $K_{ID} = Q_{ID}$.
- Isto acontece (único caso conhecido) com o uso de curvas *supersingulares*, que admitem emparelhamentos da forma $\hat{e}: \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_T$.

Encrypt

- Para cifrar uma mensagem $M \in \{0, 1\}^*$, o remetente:
 - escolhe $x \leftarrow \mathbb{Z}_r^*$ e calcula $z \leftarrow g^x$,
 - calcula $C \leftarrow H_1(z) \oplus M$,
 - obtém P_{ID} e calcula $T \leftarrow x \cdot P_{ID}$.
- O texto cifrado de M é o par (C, T) .

Decrypt

- Para decifrar uma mensagem (C, T) , o usuário identificado por ID:
 - calcula $z' \leftarrow e(T, K_{ID})$,
 - calcula $M \leftarrow H_1(z') \oplus C$.

Sign

- Para assinar $M \in \{0, 1\}^*$, o signatário associado à identidade ID:
 - escolhe $x \leftarrow \mathbb{Z}_r^*$ e calcula $z \leftarrow g^x$,
 - calcula $h \leftarrow H_2(M, z)$,
 - calcula $S \leftarrow (x + h) \cdot S_{ID}$.
- A assinatura de M é o par $\sigma = (h, S)$.

Verify

- Para verificar uma mensagem M dotada de uma assinatura $\sigma = (h, S)$ supostamente gerada pelo signatário ID, o destinatário:
 - obtém Q_{ID} e calcula $z' \leftarrow e(S, Q_{ID}) \cdot g^{-h}$,
 - calcula $h' \leftarrow H_2(M, z')$.
- O destinatário aceita a mensagem M se, e somente se, $h = h'$.

Composição

- Os processos de cifração e assinatura usam *nonces* aleatórios com a mesma forma:
 - escolhe $x \leftarrow \mathbb{Z}_r^*$ e calcula $z \leftarrow g^x$.
- Esse passo pode ser combinado no processo de geração de cifrassinatura, tornando-o mais eficiente que a mera agregação de algoritmos.
- Assinatura intransferível (verificável somente pelo destinatário legítimo).
- Ordem normal de aplicação: cifrar, assinar (ordem reversa equivalente).

Signcrypt

- Para cifrassinar $M \in \{0, 1\}^*$, o remetente:
 - escolhe $x \leftarrow \mathbb{Z}_r^*$ e calcula $z \leftarrow g^x$.
 - calcula $C \leftarrow H_1(z) \oplus M$,
 - calcula $h \leftarrow H_2(C, z)$,
 - calcula $S \leftarrow (x + h) \cdot S_{ID}$.
 - obtém P_{ID} e calcula $T \leftarrow x \cdot P_{ID}$.
- Texto cifrassinado de M : (C, S, T) .

Unsigncrypt

- Para decifrar uma tripla (C, S, T) supostamente gerada pelo signatário ID, o destinatário:
 - calcula $z' \leftarrow e(T, K_{ID})$,
 - calcula $h' \leftarrow H_2(C, z')$.
 - obtém Q_{ID} e calcula $z'' \leftarrow e(S, Q_{ID}) \cdot g^{h'}$,
 - aceita a mensagem se, e somente se, $z' = z''$.
 - calcula $M \leftarrow H_1(z') \oplus C$.