Curvas Elípticas: Aplicação em Criptografia Assimétrica

Pedro Carlos da Silva Lara[1], **Fábio Borges de Oliveira**[1]

[1]Laboratório Nacional de Computação Científica – LNCC
Coordenação de Sistemas e Redes – CSR
Av. Getúlio Vargas, 333 - Quitandinha 25.651-075 Petrópolis, RJ

{pcslara,borges}@lncc.br

Abstract. *Since the introduction of asymmetric cryptography by W. Diffie and M. Hellman in 1976, the Discrete Logarithm Problem (DLP) has been used in many different ways. A variation of the use of the (DLP) is in elliptic curves, which one this paper is allowed to discuss some relevant topics. In the end, we will use ElGamal and Menezes-Vanstone algorithms and with some examples, we will show the elliptic curves application in asymmetric cryptography.*

Resumo. *Desde a introdução da criptografia assimétrica por W. Diffie e M. Hellman em 1976, o Problema do Logaritmo Discreto (PLD) tem sido usado sob várias formas. Uma variação do uso do PLD está em curvas elípticas, sobre a qual este trabalho se prontifica a descrever alguns tópicos relevantes. Ao final, usaremos os algoritmos ElGamal e Menezes-Vanstone e, por intermédio de exemplos, mostraremos a aplicação de curvas elípticas em criptografia assimétrica.*

1. Introdução

Whitfield Diffie e Martin E. Hellman [Diffie and Hellman 1976] propuseram uma interessante solução para o problema de dois usuários estabelecerem uma chave secreta em um canal de comunicação inseguro, que é considerada a primeira prática de criptografia de chave pública. Suponha que dois usuários, Alice e Bob, queiram estabelecer uma chave secreta compartilhada. Alice seleciona aleatoriamente um inteiro secreto a tal que $a \in \mathbb{Z}_p$ e envia para Bob $P_A = g^a \mod p$ ($a = 0,\ a = 1$ e $a = p - 1$ devem ser evitados [Terada 2000]), sendo p um primo e g um gerador do grupo cíclico $\mathbb{Z}_p^*$. Analogamente, Bob escolhe um inteiro secreto $b \in \mathbb{Z}_p$ e envia à Alice $P_B = g^b$. Agora ambos usam sua chave secreta para obter uma chave secreta compartilhada $K = (P_A)^b = (P_B)^a = g^{ab}$, observe que a segurança deste protocolo está baseada na dificuldade computacional de calcular a dado g e p. Quando estas variáveis são relativamente grandes este problema se torna inviável, e é conhecido como Problema do Logaritmo Discreto (PLD). Taher ElGamal [ElGamal 1985] propôs um algoritmo também baseado na dificuldade de resolver o PLD. Vamos supor que Bob queira enviar uma mensagem $m \in \mathbb{Z}_p^*$ para Alice. Inicialmente, Alice escolhe $s \in \mathbb{Z}_p$, e logo após computa $y = g^s \mod p$. Alice publica y junto com p e g, que será a sua chave pública, e s será a sua chave privada. Bob escolhe aleatoriamente $z \in \mathbb{Z}_p$ e calcula $c_1 = g^z \mod p$ e $c_2 = m \cdot y^z \mod p$, depois envia para Alice o texto ilegível (c_1, c_2). Para decifrar m, Alice computa $c_2 \cdot (c_1^s)^{-1} = m \cdot y^z \cdot (g^{sz})^{-1} = m \cdot (g^{sz}) \cdot (g^{sz})^{-1} = m \mod p$, recuperando assim a mensagem m. Desde então muitos algoritmos de criptografia assimétrica usam o PLD como base de sua segurança. Em 1985, V. Miller [Miller 1986] e N. Koblitz [Koblitz 1987],

propuseram independentemente a aplicação de curvas elípticas em criptografia assimétrica. A segurança deste método está, mais uma vez, baseada no PLD. Este criptossistema exige uma chave de comprimento consideravelmente menor que a chave usada em alguns clássicos da criptografia assimétrica, tais como o RSA, no entanto com segurança equivalente, ficando acessível a sistemas com restrições computacionais.

2. Conceitos Algébricos

Nesta seção iremos apresentar uma breve introdução à teoria das curvas elípticas de modo mais geral, mais a frente iremos discutir estas curvas sobre estruturas mais específicas.
Definição. Uma curva elíptica sobre um corpo $\mathbb{F}$ (assumiremos sempre que $\mathbb{F}$ é um corpo de característica maior que 3) é o lugar geométrico dos pontos $(x, y) \in \mathbb{F} \times \mathbb{F}$ que satisfazem a equação

$$y^2 + axy + by = x^3 + cx^2 + dx + e \tag{1}$$

mais um ponto, chamado de ponto no infinito, que será denotado por ∞. Podemos simplificar a equação (1) deixando na forma

$$y^2 = x^3 + ax + b \tag{2}$$

com $a, b \in \mathbb{F}$. Esta curva deve ser uma curva não-singular, ou seja, não possui raízes múltiplas, para tanto precisamos ter:

$$\Delta = 4a^3 + 27b^2 \neq 0$$

Uma das propriedades mais interessantes das curvas elípticas, é o fato de seus pontos formarem uma estrutura de grupo, para isso, vamos definir a "soma" entre seus pontos. Como foi visto, nós contamos com um ponto chamado de ponto no infinito ∞; este ponto desempenhará um importante papel, pois será o elemento neutro da soma. Logo, se Ω é uma curva elíptica sobre um corpo $\mathbb{F}$, e temos $P \in \Omega$, então:

$$P + \infty = P = \infty + P$$

o simétrico de um ponto $P = (x, y)$ é o ponto $-P = (x, -y)$. Se somarmos um ponto $P \in \Omega$ com o seu simétrico $-P$ iremos, naturalmente, obter o ponto no infinito, ou seja

$$P + (-P) = \infty = (-P) + P$$

Sejam P e Q dois pontos de uma curva elíptica sobre o corpo $\mathbb{R}$ dos números reais, com $P \neq Q$, considere a reta determinada pelo segmento $\overline{PQ}$. Esta reta interceptará a curva em um ponto $-R$. O simétrico de $-R$, que é dado por R, será a soma de P e Q. Logo

$$R = P + Q$$

A figura 1 representa graficamente a soma entre dois pontos distintos $P, Q \in \Omega$ em uma curva elíptica sobre o corpo $\mathbb{R}$. Em criptografia trabalhamos com curvas elípticas sobre corpos finitos, neste caso $\mathbb{F}_p$ e $\mathbb{F}_{2^m}$, em outras palavras, trabalhamos com pontos discretos. O gráfico abaixo é apenas um apelo geométrico para a operação de soma entre os pontos formados por uma curva elíptica.

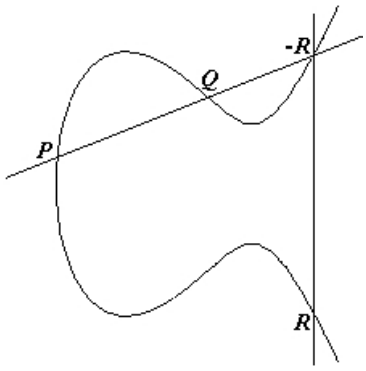

Figura 1. $P + Q = R$

Se quisermos dobrar um ponto $P \in \Omega$ o procedimento segue da seguinte maneira: passamos uma reta tangente ao ponto P; esta reta interceptará a curva Ω em outro ponto, que denotaremos por $-R$. O simétrico de $-R$ será a soma de P e P, ou seja

$$R = P + P = 2P$$

Um caso específico é configurado se tivermos P um ponto da forma $P = (x, 0)$. Neste caso a reta tangente à curva no ponto P será vertical, contudo, não interceptará a curva em um outro ponto. Neste caso temos que

$$P + P = 2P = \infty$$

2.1. Curvas Elípticas Sobre $\mathbb{F}_p$

Seja o corpo $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$, sendo p um primo qualquer. Todas as operações admitidas a seguir devem ser calculadas em função do resto da divisão por p, ou seja, todos os parâmetros e variáveis pertencem ao conjunto $\mathbb{F}_p$. Uma curva elíptica definida sobre o corpo finito primo $\mathbb{F}_p$ tem equação do tipo (2). Todos os pares $(x, y) \in \mathbb{F}_p$ que satisfazem a equação (2) pertencem à curva elíptica. Como vimos anteriormente, o ponto no infinito ∞ é essencial para que os pontos de uma curva elíptica formem um grupo. Também devemos garantir que esta curva não possua pontos repetidos, para tanto devemos ter $\Delta \neq 0$. Iremos definir uma fórmula para soma entre dois pontos pertencentes a uma curva elíptica [Koblitz et al. 2000], que será muito útil mais a frente. Se quisermos somar $P = (x_1, y_1)$ com $Q = (x_2, y_2)$ e seja $R = (x_3, y_3)$ o resultado desta soma, e se $P \neq -Q$, temos

$$x_3 = \lambda^2 - x_1 - x_2 \mod p$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \mod p$$

onde

$$\lambda = \begin{cases} \frac{(y_2 - y_1)}{(x_2 - x_1)} & \mod p, \text{se } x_1 \neq x_2 \\ \frac{(3x_1^2 + a)}{(2y_1)} & \mod p, \text{se } x_1 = x_2 \text{ e } y_1 \neq 0 \end{cases}$$

Se quiséssemos multiplicar um ponto P de uma curva elíptica Ω por um número inteiro n, o procedimento seria somá-lo n vezes. Para ilustrar esta situação, seja $P \in \Omega$, queremos obter $7P \in \Omega$. Bastaria computar:

$$7P = P + 2(P + 2P)$$

O número total de pontos que satisfaz uma curva elíptica Ω mais o ponto no infinito é denominado ordem, que denotaremos por $\#\Omega$. Um resultado importante que diz respeito à ordem de uma curva elíptica se deve a Hasse:

Teorema 1 (Hasse) *Se Ω uma curva elíptica sobre $\mathbb{F}_q$, temos*

$$q + 1 - 2\sqrt{q} \leq \#\Omega \leq q + 1 + 2\sqrt{q}$$

Existe um algoritmo de R. Scoof para obter $\#\Omega$ (ver [Schoof 1995] e [Schoof 1985]) que possui tempo de execução $O((\log p)^8)$.

2.2. Curvas Elípticas Sobre $\mathbb{F}_{2^m}$

Os elementos de um corpo finito $\mathbb{F}_{2^m}$ são representados de duas maneiras diferentes: representação polinomial e representação de base ótima. Por ser mais simples, iremos trabalhar com a representação polinomial, que é a preferida em publicações de caráter tutorial. Na representação polinomial, os elementos de $\mathbb{F}_{2^m}$ são do tipo

$$\mathbb{F}_{2^m} = \{a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_1x + a_0 : a_i \in \{0, 1\}\}$$

para $i \in \mathbb{N}$, ou seja, são polinômios de coeficientes binários. Podemos também escrever o polinômio na forma $(a_{m-1}\ a_{m-2} \ldots a_1\ a_0)$
Exemplo: Vamos descrever o conjunto $\mathbb{F}_{2^3}$

$$\mathbb{F}_{2^3} = \{(0), (1), (x), (x+1), (x^2), (x^2+1), (x^2+x), (x^2+x+1)\}$$

ou

$$\mathbb{F}_{2^3} = \{(000), (001), (010), (011), (100), (101), (110), (111)\}$$

Assim como em $\mathbb{F}_p$, onde todas as operações estão fechadas módulo p, em $\mathbb{F}_{2^m}$ estas operações estão fechadas módulo um polinômio irredutível de grau m, ou seja, um polinômio o qual é impossível fatorar em outros polinômios de graus menores. As operações de adição e multiplicação são de grande importância em $\mathbb{F}_{2^m}$, pois delas derivamos as operações de subtração e exponenciação, respectivamente. É importante ressaltar a presença de um elemento gerador de $\mathbb{F}_{2^m}$, ou seja, se g é um gerador de $\mathbb{F}_{2^m}$, obtemos todos elementos de $\mathbb{F}_{2^m} - \{0\}$ a partir das potências de g. Seja g um gerador de $\mathbb{F}_{2^m}$. Os elementos de $\mathbb{F}_{2^m}$ são

$$\mathbb{F}_{2^m} = \{0, g, g^2, \ldots, g^{2^m-2}, g^{2^m-1}\}$$

Se temos $i, j \in \mathbb{N}$ então $g^i g^j = g^{i+j} = g^{i+j \mod 2^m-1}$ e $g^{ij} = g^{ij \mod 2^m-1}$ em $\mathbb{F}_{2^m}$. A equação padrão de uma curva elíptica sobre $\mathbb{F}_{2^m}$ é ligeiramente diferente das curvas sobre $\mathbb{F}_p$:

$$\Omega : y^2 + xy = x^3 + ax^2 + b \text{ em } \mathbb{F}_{2^m}$$

Seja $P = (x, y)$ um ponto de uma curva elíptica qualquer sobre $\mathbb{F}_{2^m}$. O inverso deste ponto é dado por $-P = (x, y + x)$. Observe a diferença: o inverso de $P = (x, y)$, em $\mathbb{F}_p$, é $-P = (x, -y)$. As equações que definem a soma também são um pouco diferentes das equações em $\mathbb{F}_p$. Se quisermos somar $P = (x_1, y_1)$ com $Q = (x_2, y_2)$ e seja $R = (x_3, y_3)$ o resultado desta soma, e se $P \neq -Q$, então

$$x_3 = \begin{cases} \left(\frac{y_2+y_1}{x_2+x_1}\right)^2 + \frac{y_2+y_1}{x_2+x_1} + x_1 + x_2 + a, \text{ se } P \neq Q \\ x_1^2 + \frac{b}{x_1^2}, \text{ se } P = Q \end{cases}$$

e

$$y_3 = \begin{cases} \left(\frac{y_2+y_1}{x_2+x_1}\right)(x_1+x_3)+x_3+y_1, \text{ se } P \neq Q \\ x_1^2+\left(x_1+\frac{y_1}{x_1}\right)x_3+x_3, \text{ se } P = Q \end{cases}$$

3. Segurança

Se um esquema criptográfico está baseado em um problema matemático, a única forma de quebrar esse sistema é por intermédio de algoritmos que tentem resolver, da forma mais eficiente possível, o problema proposto. Tais algoritmos se dividem em dois tipos básicos: algoritmos específicos e algoritmos genéricos [Barbosa 2003]. Algoritmos específicos são utilizados em casos bem definidos, baseiam-se em determinados aspectos, ou seja, onde se conhece um tipo de parâmetro, ficando o seu uso restrito a esses tipos de situações. Já os algoritmos genéricos não se preocupam com qualquer tipo de restrição. Dois tipos de curvas elípticas apresentam uma certa "fraqueza" contra algoritmos específicos, são elas: supersingulares (supersingular elliptic curve) e anômalas (anomalous elliptic curve). Seja $\Omega(\mathbb{F}_p)$ uma curva elíptica, definimos esta curva como anômala, se a ordem desta curva for p, ou seja, $\#\Omega = p$. Pelo teorema de Hasse, sabemos que $\#\Omega(\mathbb{F}_q) = q + 1 - t$, onde $|t| \leq 2\sqrt{q}$. Uma curva elíptica é chamada de supersingular se, e somente se, $t = 0 \mod p$. Estas duas curvas devem ser evitadas, pois existem algoritmos eficientes contra ambas. É fácil observar que o número de curvas supersingulares ou anômalas é extremamente menor em relação ao número total de curvas possíveis sobre um corpo finito qualquer [Barreto 1999]. Para tanto, o NIST (National Institute of Standards and Technology) recomenda em seu site curvas e parâmetros de aplicação segura. Os algoritmos genéricos, como era de se esperar, possuem o tempo de execução superior ao específico. Este tempo de execução permitirá avaliar quanto o sistema criptográfico é seguro. Existem algoritmos para resolver o PLD e o Problema da Fatoração de Inteiros (PFI) (ver [Semaev 1998], [Adleman and DeMarrais 1994]), porém nenhum possui tempo de execução polinomial. Sendo assim, esses problemas são considerados de difícil resolução. O algoritmo genérico mais conhecido para o PLD sobre curvas elípticas é o Pollard-ρ (Pollard's rho [Pollard 1978]) que tem tempo de execução de $O(\sqrt{n})$ [Gordon and McCurley 1993] onde n é a ordem da curva (ver [Teske 1998], [Kuhn and Struik 2001]). Observe que este algoritmo é exponencial em relação ao número de bits necessários para representar a ordem do grupo. Já para o PLD e o PFI o tempo de execução é $O(\exp(c+O(1))(\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}})$ onde c é uma constante e $n = pq$; este é considerado sub-exponcial. Logo, é mais difícil de se resolver o PLD sobre curvas elípticas do que o PLD original e o PFI. Por estes motivos, um criptossistema baseado em curvas elípticas necessita de uma chave de aproximadamente 160 *bits*, e o RSA, por exemplo, utiliza uma chave de 1024 *bits*, mantendo segurança equivalente em ambos. Na tabela 1 podemos ver o tamanho das chaves em *bits*, com o mesmo grau de segurança.

Como era de se esperar, os algoritmos simétricos necessitam de chaves muito menores que os assimétricos. O mais relevante na tabela 1 é a diferença considerável entre os algoritmos assimétricos.

4. O PLD Sobre Curvas Elípticas

Nesta seção iremos tratar do PLD sobre curvas elípticas. Antes de entrar em curvas elípticas, vamos falar de forma mais geral. Seja $(G, *)$ um grupo munido com a operação $*$,

Tabela 1. Tamanho das chaves em *bits*.

Modelo de Criptografia			
Simétrico	ECC	RSA	Razão ECC:RSA
80	163	1024	1:6
128	256	3072	1:12
192	384	7680	1:20
256	512	15360	1:30

e dado $\alpha \in G$ um gerador de um subgrupo J de G, ou seja, $J = \{\alpha^i : i \geq 0\}$. Sendo $\beta \in J$, o problema consiste em calcular s tal que $\alpha^s = \beta$ onde $\alpha^s = \alpha * \alpha * \ldots * \alpha$ s vezes [Terada 2000]. No caso de uma curva elíptica, o conjunto G é formado pelos pontos desta curva mais o ponto no infinito ∞, e a operação binária $*$ é representada pela soma de dois pontos na curva, ou seja, a dificuldade que era na exponenciação agora está na multiplicação de um inteiro por um ponto de uma curva elíptica. Isso significa que o problema consiste em encontrar um inteiro que foi multiplicado por um ponto da curva elíptica, isto é, seja Ω uma curva elíptica, dados os pontos $P, Q \in \Omega$ encontrar s tal que $P = sQ$.

5. Aplicação de Curvas Elípticas em Criptografia

Nesta seção iremos expor dois métodos usados para criptografar mensagens usando curvas elípticas. Ilustraremos com exemplos numéricos tais algoritmos, tendo em vista facilitar o entendimento. Iremos trabalhar com ambos os corpos $\mathbb{F}_p$ e $\mathbb{F}_{2^m}$ para descrever o algoritmo ElGamal sobre curvas elípticas, no entanto o outro algoritmo (Menezes-Vanstone) que mostraremos é aplicado apenas em corpos primos $\mathbb{F}_p$.

5.1. O Algoritmo ElGamal Sobre Curvas Elípticas

Como o algoritmo ElGamal trabalha com um grupo, podemos usá-lo sobre curvas elípticas. Vamos supor que Alice deseja enviar uma mensagem para Bob, seja m esta mensagem mapeada num ponto de uma curva elíptica, para tanto vamos ao algoritmo [Hankerson et al. 2003].

Algoritmo:

1. Bob escolhe, e mantém em segredo, um inteiro $b \in \mathbb{N}^*$ e envia à Alice $K = bP$, sendo P um ponto da curva elíptica conhecido publicamente.
2. Alice escolhe inteiro $a \in \mathbb{N}^*$ (também o guarda para si) e computa $c_1 = aP$ e $c_2 = m + aK$.
3. Para decifrar a mensagem m Bob calcula $c_2 - bc_1 = m + abP - baP = m$.

Para facilitar o entendimento, vamos a um exemplo.

Exemplo: Iremos criptografar a palavra LNCC, neste caso iremos cifrar em blocos de duas letras, primeiro mapeamos as letras LN e depois CC em pontos de uma curva elíptica. Neste exemplo iremos mapear da seguinte maneira: cada letra do alfabeto estará associada a um número entre 01 e 26 de forma que A$= 01$, B$= 02$, ..., Z$= 26$, porém, precisamos mapear esta mensagem em pontos de uma curva elíptica. Para isso iremos multiplicar o número associado à letra por um ponto P, logo iremos concatenar os números associados e depois multiplicar por P. Para ilustrar esta situação, vamos supor que iremos mapear

AB em uma curva elíptica $y^2 = x^3 + 373x + 402$ sobre $\mathbb{F}_{3697}$, como A= 01 e B= 02 concatenamos 01 e 02, obtemos 0102 e multiplicamos por P, digamos $P = (551, 1946)$, logo $102P = 102(551, 1946) = (3108, 1065)$. Logo, AB=$(3108, 1065)$. Vamos supor que Alice enviará a palavra LNCC para Bob, para tanto

1. Bob escolhe um número inteiro b, digamos $b = 919$, e envia à Alice $P = (551, 1946)$ e $K = bP = 919(551, 1446) = (301, 3454)$ (observe que é um problema inviável descobrir b a partir de K e P). Observe que K pertence a curva elíptica $y^2 = x^3 + 373x + 402$ sobre $\mathbb{F}_{3697}$.
2. Alice escolhe um inteiro $a = 815$, e multiplica $c_1 = 815P = 815(551, 1946) = (958, 14)$. Agora Alice precisa mapear a *string* LN, procedendo do modo que fizemos acima. A *string* foi mapeada no ponto $m = \text{LN} = (2309, 2502)$. Precisamos somar este ponto ao ponto obtido por $aK = 815K = 815(301, 3454) = (837, 2461)$. Agora Alice mascara o ponto $m = \text{LN} = (2309, 2502)$, somando-o ao ponto aK. Logo, $c_2 = m + aK = (2309, 2502) + (837, 2461) = (1518, 14)$. Alice transmite a Bob o par de pontos cifrados $(c_1, c_2) = ((958, 14), (1518, 14))$
3. Para descriptar Bob deve calcular $m = c_2 - bc_1$. Primeiro Bob calcula o valor de $bc_1 = 919c_1 = 919(958, 14) = (837, 2461)$. Prosseguindo, temos que $m = c_2 - bc_1 = (1518, 14) - (837, 2461) = (1518, 14) + (-(837, 2461)) = (1518, 14) + (837, -2461) = (2309, 2502)$, repare que LN $= (2309, 2502)$.
4. Para criptografar as letras CC o procedimento é o mesmo, agora Alice só precisa calcular o ponto $c_2 = m + aK$ CC $= 0303P = 303(551, 1946) = (3023, 762) = m$, $c_2 = (3023, 762) + (837, 2461) = (3084, 2426)$ e envia $c_2 = (3084, 2426)$ para Bob.
5. Bob novamente calcula $m = c_2 - bc_1 = (3084, 2426) - (837, 2461) = (3084, 2426) + (-(837, 2461)) = (3084, 2426) + (837, -2461) = (3023, 762)$, recuperando a *string* CC, e assim encerra o processo.

Neste trabalho, expomos dois corpos para serem usados sobre curvas elípticas, $\mathbb{F}_p$ e $\mathbb{F}_{2^m}$. No exemplo anterior usamos o corpo $\mathbb{F}_p$, agora usaremos como exemplo um corpo finito do tipo $\mathbb{F}_{2^m}$, usando também o algoritmo ElGamal para criptografar a palavra LNCC. A conveniência em usar o corpo $\mathbb{F}_{2^m}$ é que podemos representá-lo por *strings* de m *bits*, ou seja, cada elemento de $\mathbb{F}_{2^m}$ pode ser representado por um número binário entre 0 e $2^m - 1$. Usaremos a curva elíptica $\Omega : y^2 + xy = x^3 + g^{140}x^2 + g^{97}$. Observe que $a = g^{140} = x^6 + x^5 + x^3 + x^2 = (01101100)$ e $b = g^{97} = x^6 + x^2 + x = (01000110)$, onde $g = x^7 + x^3 + x^2 + x$ é um polinômio gerador de $\mathbb{F}_{2^8}$. Portanto, os pontos desta curva serão um par de polinômios pertencentes a $\mathbb{F}_{2^8}$. Por exemplo, o ponto (g^{63}, g^{81}) pertence a Ω, visto que

$$(g^{81})^2 + g^{63}g^{81} = (g^{63})^3 + g^{140}(g^{63})^2 + g^{97}$$

$$g^{162} + g^{144} = g^{189} + g^{11} + g^{97}$$

$$(10001100) \oplus (10110110) = (11010001) \oplus (10101101) \oplus (01000110)$$

$$(00111010) = (00111010)$$

Exemplo: Novamente, é Alice quem enviará uma mensagem para Bob. Vamos supor que, depois de mapeado na curva elíptica Ω, LN $= (g^{137}, g^{40})$ e CC $= (g^{82}, g^{102})$.

1. Bob escolhe um inteiro, $b = 77$, e envia para Alice $P = (g^{58}, g^{27}) = ((10011000), (10011110))$ (a partir de agora, para a notação não ficar muito carregada, iremos usar (g^{115}, g^{49}) no lugar de $((00110011), (10110010)$ por exemplo) e $K = bP = 77(g^{58}, g^{27}) = (g^{23}, g^{13})$, sendo P um ponto da curva elíptica $\Omega : y^2 + xy = x^3 + g^{140}x^2 + g^{97}$.
2. Alice escolhe um inteiro $a = 52$, e multiplica $c_1 = 52P = 52(g^{58}, g^{27}) = (g^{182}, g^{99})$. Como a *string* já foi mapeada, $m =$ LN $= (g^{137}, g^{40})$. Precisamos somar este ponto ao ponto obtido por $aK = 52K = 52(g^{23}, g^{13}) = (g^2, g^{14})$. Agora Alice esconde o ponto $m =$ LN $= (g^{137}, g^{40})$ somando-o ao ponto aK, logo $c_2 = m + aK = (g^{137}, g^{40}) + (g^2, g^{14}) = (g^{174}, g^7)$, e envia os pontos cifrados $(c_1, c_2) = ((g^{182}, g^{99}), (g^{174}, g^7))$.
3. Bob agora precisa descriptografar o texto ilegível $(c_1, c_2) = ((g^{182}, g^{99}), (g^{174}, g^7))$ que recebeu de Alice, para isso deve calcular $m = c_2 - bc_1$. Primeiro Bob calcula o valor de $bc_1 = 77c_1 = 77(g^{182}, g^{99}) = (g^2, g^{14})$. Vimos que, se $P = (x, y)$, então $-P(x, x + y)$. Fazendo isso temos que $-bc_1 = (g^2, g^2 + g^{14}) = (g^2, g^{77})$. Prosseguindo temos que $m = c_2 - bc_1 = (g^{174}, g^7) - (g^2, g^{14}) = (g^{174}, g^7) + (-(g^2, g^{14})) = (g^{174}, g^7) + (g^2, g^{77}) = (g^{137}, g^{40})$; observe que LN $= (g^{137}, g^{40})$ recuperado assim a primeira parte da mensagem.
4. Alice repete o mesmo procedimento, só que agora já temos o ponto $c_1 = (g^{182}, g^{99})$. A mensagem CC foi mapeada no ponto CC $= (g^{87}, g^{102})$, como foi dito anteriormente. Para obter o ponto c_2, ela computa $c_2 = m + aK = (g^{87}, g^{102}) + (g^2, g^{14}) = (g^{66}, g^{135})$, e envia para Bob o ponto $c_2 = (g^{66}, g^{135})$.
5. Bob agora calcula $m = (g^{66}, g^{135}) + (g^2, g^{77}) = (g^{87}, g^{102})$. Lembre-se que Bob já havia calculado o valor de $-bc_1 = (g^2, g^{77})$, recuperando a *string* CC e, assim, finalizando o algoritmo.

5.2. O Criptossistema Menezes-Vanstone

Uma outra técnica muito usada para criptografar dados usando curvas elípticas é o criptossistema Menezes-Vanstone [Menezes and Vanstone 1993]. Neste sistema, o texto legível é um par ordenado $m = (x_1, x_2)$, com $x_1, x_2 \in \mathbb{F}_p^*$, sendo que m não é um ponto da curva elíptica em questão, diferentemente do criptossistema anterior. O texto ilegível será uma tripla ordenada $r = (y_0, y_1, y_2)$, onde $y_1, y_2 \in \mathbb{F}_p^*$ e y_0 é um ponto da curva elíptica. Segue abaixo o algoritmo usado neste método.

Algoritmo:

Para criptografar $m = (x_1, x_2)$.

1. Bob escolhe um inteiro $k \in \mathbb{F}_p^*$ e calcula $y_0 = kP$ (lembre-se que Bob conhece publicamente o ponto $P \in \Omega$).
2. Bob computa $(c_1, c_2) = kQ$, $y_1 = c_1x_1 \mod p$ e $y_2 = c_2x_2 \mod p$. Envia para Alice a tripla $r = (y_0, y_1, y_2)$.

Para descriptografar $r = (y_0, y_1, y_2)$.

1. Alice calcula $sy_0 = skP = kQ = (c_1, c_2)$, onde s é um inteiro selecionado por Alice. Observe que $Q = sP$.
2. Logo após, Alice calcula $x_1 = y_1(c_1)^{-1} \mod p$ e $x_2 = y_2(c_2)^{-1} \mod p$, recuperando a mensagem $m = (x_1, x_2)$.

Vamos a um exemplo da aplicação deste criptossistema.
Exemplo: Bob irá enviar a mensagem MCT para Alice, tal mensagem será codificada na tabela ASCII. Então temos que M $= 77$, C $= 67$ e T $= 84$. Como a mensagem legível será o par $m = (x_1, x_2)$, vamos separar em dois caracteres. Logo, a mensagem legível ficará $m = (7767, 84)$. Para Bob criptografar tal palavra ele precisa conhecer os pontos $P = (1355793, 621792) \in \Omega$ e $Q = (949594, 812871) \in \Omega$, onde $\Omega : y^2 = x^3 + 67110x + 262147$ está sobre $\mathbb{F}_{2097421}$ e $Q = 78771 \cdot P$, ou seja, Alice escolheu o inteiro $s = 78771$.

1. Bob escolhe $k = 23358$, e logo após computa $y_0 = kP = (1390038, 1344654)$.
2. Bob agora calcula $kQ = (647014, 449701) = (c_1, c_2)$, $c_1 \cdot x_1 = 647014 \cdot 7767 = 2034443 \mod p$ e $c_2 \cdot x_2 = 449701 \cdot 84 = 21306 \mod p$. Bob envia $r = (y_0, y_1, y_2)$ para Alice. Repare que y_0 é um ponto da curva elíptica e y_1 e y_2 são inteiros pertencentes a $\mathbb{F}_{2097421}$.
3. Para descriptografar Alice calcula $s \cdot y_0 = (647014, 449701) = (c_1, c_2)$. Para finalizar calcula $x_1 = y_1(c_1)^{-1} = 7767 \mod p$ e $x_2 = y_2(c_2)^{-1} = 84 \mod p$, recuperando a mensagem MCT.

6. Considerações Finais

Concluímos que os métodos de criptografia sobre grupos de curvas elípticas possuem segurança equivalente ao RSA, entretanto, com quantidade de bits consideravelmente menor na chave criptográfica, o que possibilita um ganho substancial de processamento. Uma vez que a chave criptográfica pode ser reduzida, podemos ter criptografia com curvas elípticas em dispositivos com pouco poder computacional, por exemplo, cartões de banco. Isto se deve ao fato de que o PLD sobre pontos de uma curva elíptica é mais difícil de ser resolvido do que sobre um corpo primo, como foi mostrado na seção 3 . Além disto, os portais de bancos na Internet poderão processar muito mais requisições sem a necessidade de aumentar sua capacidade de processamento. Assim, concluímos que deve ser recomendada a substituição do RSA por métodos de criptografia baseados em curvas elípticas. Também fica recomendado o uso do algoritmo de Menezes-Vanstone para troca de chaves. Estas recomendações possibilitam a existência de segurança com maior velocidade e capacidade de transmissão de dados.

7. Agradecimentos

Gostaríamos de agradecer ao PIBIC/CNPq, pelo apoio financeiro para este trabalho.

Referências

Adleman, L. M. and DeMarrais, J. (1994). A subexponential algorithm for discrete logarithms over all finite fields. In *CRYPTO '93: Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, pages 147–158, London, UK. Springer-Verlag.

Barbosa, J. C. (2003). Criptografia de chave pública baseada em curvas elípticas. Dissertação de Mestrado, COPPE–UFRJ.

Barreto, P. (1999). Curvas elípticas e criptografia - conceitos e algoritmos.

Diffie, W. and Hellman, M. E. (1976). New directions in cryptography. *IEEE Trans. Information Theory*, IT-22(6):644–654.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31(4):469–472.

Gordon, D. M. and McCurley, K. S. (1993). Massively parallel computation of discrete logarithms. *Lecture Notes in Computer Science*, 740:312–323.

Hankerson, D., Menezes, A. J., and Vanstone, S. (2003). *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209.

Koblitz, N., Menezes, A., and Vanstone, S. (2000). The state of elliptic curve cryptography. *Des. Codes Cryptography*, 19(2-3):173–193.

Kuhn, F. and Struik, R. (2001). Extensions of pollard's rho algorithm for computing multiple discrete logarithms. In *SAC '01: Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography*, pages 212–229, London, UK. Springer-Verlag.

Menezes, A. and Vanstone, S. A. (1993). Elliptic curve cryptosystems and their implementations. *J. Cryptology*, 6(4):209–224.

Miller, V. S. (1986). Use of elliptic curves in cryptography. In *Advances in cryptology—CRYPTO '85 (Santa Barbara, Calif., 1985)*, volume 218 of *Lecture Notes in Comput. Sci.*, pages 417–426. Springer, Berlin.

Pollard, J. M. (1978). Monte Carlo methods for index computation mod p. *Mathematics of Computation*, 32:918–924.

Schoof, R. (1985). Elliptic curves over finite fields and the computation of square roots mod p. *Mathematics of Computation*, 44:483–494.

Schoof, R. (1995). Counting points on elliptic curves over finite fields. *J. Th'eor. Nombres Bordeaux (219–254)*.

Semaev, I. A. (1998). An algorithm for evaluation of discrete logarithms in some nonprime finite fields. *Math. Comput.*, 67(224):1679–1689.

Terada, R. (2000). *Segurança de Dados: Criptografia em Redes de Computadores*. Edgard Blucher, 1 edition.

Teske, E. (1998). Speeding up pollard's rho method for computing discrete logarithms. In *ANTS-III: Proceedings of the Third International Symposium on Algorithmic Number Theory*, pages 541–554, London, UK. Springer-Verlag.