DANIELLE PAES BARRETTO DE ARRUDA CAMARA.

CRIPTOGRAFIA DE CHAVE PÚBLICA BASEADA EM CURVAS ELÍPTICAS COM APLICAÇÕES.

RECIFE
2001

UNIVERSIDADE FEDERAL DE PERNAMBUCO

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

CRIPTOGRAFIA DE CHAVE PÚBLICA BASEADA EM CURVAS ELÍPTICAS COM APLICAÇÕES.

Dissertação submetida à
Universidade Federal de Pernambuco
como parte dos requisitos para a
obtenção do grau de Mestre em Engenharia Elétrica

DANIELLE PAES BARRETTO DE ARRUDA CAMARA

Recife, Agosto de 2001.

CRIPTOGRAFIA DE CHAVE PÚBLICA BASEADA EM CURVAS ELÍPTICAS COM APLICAÇÕES.

Danielle Paes Barretto de Arruda Camara

'Esta dissertação foi julgada adequada para o Título de Mestre em Engenharia Elétrica, Área de Concentração em Comunicações, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco.'

_____

Prof. Dr. Ricardo Menezes Campello de Souza, Ph. D., Manchester.

_____

Prof. Dr. Eduardo Fontana; Ph. D., Stanford.

Banca Examinadora:

_____

Prof. Dr. Cecílio Pimentel, Ph. D., Waterloo.

_____

Prof. Dr. Ricardo Menezes Campello de Souza, Ph. D., Manchester.

_____

Prof. Dr. Manoel José Machado Soares Lemos; Ph. D., Oxford.

_____

Prof. Dr. Valdemar C. Rocha Junior; Ph. D., Kent.

À minha mãe, Eunice Paes Barretto de Arruda Camara.

AGRADECIMENTOS

Em primeiro lugar gostaria de agradecer a Deus por mais essa etapa concluída e por todas as experiências vividas durante esses anos. Aproveito para agradecer de modo bastante especial, a uma mulher extremamente forte e corajosa, mãe dedicada e amorosa, que sempre me apoiou de maneira irrestrita sendo de fundamental importância em mais esse momento e a quem dedico esta dissertação.

Agradeço ao Prof. Ricardo Campello, meu orientador, por todos esses anos de rica convivência, não só como mestre, mas também como amigo, compartilhando seu conhecimento e experiências, estando sempre aberto a troca de idéias e mostrando sempre o prazer de se aprender e passar esse aprendizado a outros.

A Benoit agradeço pelo companheirismo e carinho, entendendo os momentos em que estive ocupada demais, procurando muitas vezes me fazer esquecer o estresse desses momentos finais.

Agradeço aos meus familiares e amigos pelo estímulo e por compreenderem as minhas ausências.

Gostaria de aproveitar este momento para prestar meus sinceros agradecimentos aos professores do DES por, apesar de todas as dificuldades enfrentadas pela educação no nosso país, não desistirem e por a cada ano procurarem passar seus conhecimentos a vários estudantes como eu.

Agradeço também aos colegas do mestrado, aos funcionários do DES e a todos aqueles que de alguma forma participaram deste momento da minha vida.

Por fim, agradeço o apoio financeiro prestado pela CAPES.

Resumo da Dissertação apresentada à UFPE como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica.

CRIPTOGRAFIA DE CHAVE PÚBLICA BASEADA EM CURVAS ELÍPTICAS COM APLICAÇÕES.

DANIELLE PAES BARRETTO DE ARRUDA CAMARA.

Agosto 2001

Orientador: Prof. Ricardo Menezes Campello de Souza, Ph. D., Manchester.

Área de Concentração: Comunicações

Palavras-chave: criptografia, criptografia simétrica, criptografia assimétrica, curvas elípticas, Problema do Logaritmo Discreto, IP móvel.

Número de Páginas: 169

O presente trabalho teve como objetivo principal o estudo do mais novo tipo de cripto-sistema assimétrico, aquele baseado no Problema do Logaritmo Discreto sobre Curvas Elípticas (PLDCE). Após uma breve apresentação das principais ferramentas matemáticas empregadas no trabalho, alguns aspectos gerais da área de Criptografia são abordados, destacando-se as diferenças entre cripto-sistemas simétricos e assimétricos, fazendo-se uma análise comparativa dos mesmos. Nesse contexto, uma maior ênfase foi dada à criptografia baseada em curvas elípticas (CCE), tema central da dissertação, uma vez que esta apresenta a tendência atual no que diz respeito às técnicas de Criptografia assimétrica empregadas com o objetivo de se obter sigilo e autenticidade em ambientes inseguros de comunicação e processamento de dados, tais como a Internet e a telefonia móvel. Assim, os principais cripto-sistemas baseados no PLDCE foram abordados e ilustrados através de algoritmos criptográficos práticos, o que inclui uma investigação de aspectos ligados à sua implementação e ao nível de segurança proporcionado pelos mesmos. Foi feita ainda uma análise comparativa em relação aos outros cripto-sistemas assimétricos que têm como base os problemas de Fatoração de Inteiros ou do Logaritmo Discreto sobre Corpos Finitos. Por fim foi discutido um contexto atual de aplicação prática das ferramentas de segurança, o IP móvel. Esse sistema tem no IPSec, parte responsável pela segurança do protocolo, a presença do Protocolo para Troca de Chave de Diffie-Hellman baseado no PLDCE.

Abstract of Dissertation presented to UFPE as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

ELLIPTIC CURVE BASED PUBLIC-KEY CRYPTOGRAPHY WITH APPLICATIONS.

DANIELLE PAES BARRETTO DE ARRUDA CAMARA.

August 2001

Advisor: Prof. Ricardo Menezes Campello de Souza, Ph. D., Manchester.
Area of Concentration: Communications
Key-words: cryptography, symmetric and asymmetric cryptosystems, Elliptic curves, Discrete Logarithm Problem, mobile IP.
Number of Pages: 169

The main objective of this dissertation is the study of the newest type of asymmetric cryptosystem which is based on the Discrete Logarithm Problem over Elliptic Curves (DLPEC). After a brief review of the most relevant mathematical tools used in the text, some general aspects of the field of cryptography are described, with emphasis being given to the main diferences between symmetric and asymmetric cryptosystems. In this context, the major focus was on Elliptic Curve Cryptography (ECC), which represents the current mainstream of research and development of asymmetric cryptography techniques for data security. The most important cryptosystems based on the (DLPEC) are described and aspects related to its security and implementation are discussed. These elliptic curve cryptosystems are then compared with the asymmetric systems which are based on the integer factoring problem (RSA) and on the discrete logarithm problem over finite fields. Finally, a pratical aplication context was considered, namely mobile IP. Such a system, has in IPSec (which is responsible for the security of the protocol) the presence of the elliptic curve Difife-Hellman (ECDH) Key Exchange Protocol.

SUMÁRIO

LISTA DE FIGURAS

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 4

CAPÍTULO 5

APÊNDICE A

APÊNDICE B

LISTA DE TABELAS

CAPÍTULO 2

Capítulo 3

Capítulo 4

Capítulo 5

Apêndice B

Apêndice D

1

Introdução

Grandes transformações ocorreram nos últimos anos na área de telecomunicações, fazendo com que muitas coisas que eram feitas pessoalmente, pudessem agora ser feitas através de meios digitais. Tais meios trazem muitas vantagens sobre o meio impresso, como por exemplo: armazenamento mais compacto, transferência quase que instantânea e acesso via base de dados facilitada.

Isso promoveu, por exemplo, uma significante economia para grandes empresas que atualmente não precisam enviar seus funcionários para outras localidades a fim de fechar grandes negócios. Reuniões podem ser feitas através de vídeo conferências, propostas de negócio podem ser enviadas pela rede e contratos podem ser feitos através do meio digital. Mas tudo isso tem um preço. Não se pode transmitir informações sigilosas via rede sem que as mesmas estejam protegidas de possíveis competidores, que tanto podem apenas ficar a par dos planos da empresa como podem alterar dados a fim de prejudicá-la. Desta forma, a fim de aproveitar a economia e rapidez proporcionada por essa nova era globalizada, onde as formas de comunicação se tornam cada vez mais rápidas e acessíveis; e onde a Internet atinge cada vez mais pessoas, torna-se necessária a utilização de meios que protejam tais dados, tanto apenas da leitura dos mesmos por pessoas não autorizadas, promovendo desta forma *privacidade*, como também inviabilizando a alteração dos mesmos, promovendo assim a *integridade* dos dados.

Tudo isto é possível uma vez que, diferentemente do meio impresso, a informação na forma digital pode ser facilmente roubada a partir de uma localidade remota e também ser interceptada e alterada.

Um dos meios utilizados a fim de promover privacidade e integridade dos dados é a criptografia.

O objetivo principal desta dissertação é o estudo do mais novo tipo de cripto-sistemas assimétrico, aquele baseado no Problema do Logaritmo Discreto sobre Curvas Elípticas (PLDCE).

Tais cripto-sistemas surgiram com vantagens sobre os outros cripto-sistemas de mesma classe, como por exemplo, são capazes de obter um mesmo nível de segurança com comprimentos de chave menores, proporcionando desta forma economia no espaço para armazenamento e menor gasto de energia no processamento. Desta forma mecanismos com limitação de memória e potência, como por exemplo os *smart cards*, podem ter alto nível de segurança.

Estes cripto-sistemas são construídos levando em conta entidades matemáticas conhecidas como Curvas Elípticas. Tais entidades são conhecidas a mais de um século, porém seu uso em criptografia é relativamente recente, datando de década de 80 quando Neil Koblitz e Victor Miller chegaram, de forma independente, à conclusão que esta rica estrutura matemática seria uma fonte de problemas matemáticos de difícil solução e, portanto, poderia ser utilizada como base para a construção de cripto-sistemas assimétricos.

Devido a dificuldade inerente deste problema (PLDCE) e a facilidade de entendimento dos problemas utilizados nos outros cripto-sistemas assimétricos conhecidos (e.g., problema de fatoração de inteiros utilizado pelo RSA[RSA78]) ainda não são muitos os pesquisadores a investigar no estudo de tais cripto-sistemas. Porém as vantagens demonstradas sobre os outros cripto-sistemas assimétricos tem mostrado que o esforço em procurar aprofundar o conhecimento nos cripto-sistemas baseados no PLDCE é compessador e isto tem feito com que cada vez mais cientistas se interessem e passem a investigá-los.

Apesar do pouco tempo de estudo e ainda certa dúvida por parte de alguns cientistas em relação às alegadas vantagens, principalmente por causa do pouco tempo de investigação dos mesmos, tais cripto-sistemas tem mostrado sua força sendo utilizados em alguns padrões, como é o caso do protocolo para comunicações em redes móveis, o IP móvel. Este protocolo que vem em substituição ao IP para tornar possível a comunicação via Internet mesmo que o usuário se encontre em movimento utiliza, em seus mecanismos de segurança o protocolo para troca de chave Diffie-Hellman baseado no PLDCE.

Baseada nestes fatos esta dissertação foi construída com o intuito de prover conhecimento básico sobre os cripto-sistemas baseados no PLDCE, analisando-os e por fim mostrando uma das suas aplicações práticas, o IP móvel.

A dissertação em questão é constituída de mais 5 capítulos e 5 apêndices. A seguir é feita uma breve descrição de cada um deles.

Capítulo 2: Este capítulo foi construído no intuito de promover informações básicas sobre estruturas algébricas como grupos, anéis, corpos e em especial corpos finitos (Campos de Galois). O capítulo também inclui alguns aspectos da Teoria de Curvas Elípticas sobre Corpos Finitos.

Capítulo 3: Neste capítulo são introduzidos alguns conceitos básicos de Criptografia assim como é feita a descrição de alguns cripto-sistemas utilizados no IPsec (utilizado como protocolo padrão de segurança em redes móveis): o HMAC-MD5-96 discutido na seção 3.4 e o substituto do DES como padrão de cifragem para dados do governo americano, o Rijndael, apresentado na seção 3.5.

Capítulo 4: Aqui é introduzido o PLDCE e alguns cripto-sistemas baseados em tal problema são mostrados, como por exemplo, o protocolo para troca de chave de Diffe-Hellman utilizado como uma das ferramentas criptográficas no IP móvel.

Capítulo 5: Neste capítulo será dada uma breve explicação sobre o funcionamento do IP móvel e aspectos de segurança do mesmo serão analisados.

Capítulo 6: As conclusões sobre este trabalho são apresentadas e sugestões para futuras pesquisas.

Apêndice A: Neste apêndice conceitos básicos sobre geometria projetiva são fornecidos com o intuito de facilitar o entendimento de alguns aspectos ligados as curvas elípticas como por exemplo, o ponto no infinito.

Apêndice B: Devido a existência de algumas expressões sobre complexidade dos algoritmos criptográfico durante esta dissertação e com intuito de promover um maior entendimento das mesmas, este apêndice foi inserido.

Apêndice C: Este apêndice consiste em um glossário de alguns termos utilizados no capítulo 5 sobre IP móvel.

Apêndice D: Este capítulo analisa alguns aspectos ligados a implementação dos cripto-sistemas baseados no PLDCE, como por exemplo, aspectos envolvendo a seleção da representação dos elementos sobre o corpo escolhido e a seleção da curva.

Apêndice E: Para finalizar, um pouco da biografia de John Wallis, um importante matemático envolvido com cônicas assim como muitos outros assuntos diferentes, será mostrada.

2

ESTRUTURAS ALGÉBRICAS

Este capítulo foi construído no intuito de prover informações básicas sobre estruturas algébricas de importância em várias áreas da Engenharia Elétrica, tais como Códigos Corretores de Erro, Processamento Digital de Sinais e Criptografia.

Além de estruturas algébricas como grupos, anéis, corpos e em especial corpos finitos (Campos de Galois), o capítulo também inclui alguns aspectos da Teoria de Curvas Elípticas sobre Corpos Finitos, tudo isto visando um melhor entendimento dos capítulos subsequentes.

Informações adicionais sobre outras ferramentas relevantes no contexto desta dissertação, tais como noções de Geometria Projetiva e Complexidade Computacional, são apresentadas nos apêndices.

2.1 – Estruturas Algébricas

2.1.1 - Grupos

Definição 2.1 – Um *grupo* $\langle G,* \rangle$ é uma estrutura algébrica, onde G é um conjunto não-vazio e $*$ é uma operação neste conjunto, de modo que os seguintes axiomas são válidos

i) Fechamento: $\forall\ a, b \in G,\ a * b \in G$.

ii) Associatividade: $a * (b * c) = (a * b) * c = a * b * c$, $\forall\ a, b, c \in G$.

iii) Elemento Identidade: Existe um elemento $e \in G$, chamado elemento identidade, tal que $e * g = g * e = g$, $\forall\ g \in G$; se $*$ é a adição usual então $e = 0$; se é a multiplicação usual, então $e = 1$.

iv) Inversos: Para cada elemento $a \in G$ existe um elemento $a^{-1} \in G$, chamado inverso de a, de modo que $a * a^{-1} = a^{-1} * a = e$. A notação a^{-1} denota usualmente o inverso do elemento a em um grupo multiplicativo, ou seja, o grupo no qual $*$ é a operação de multiplicação. No caso de um grupo aditivo, ou seja o grupo no qual $*$ é a operação de adição, o inverso de a será denotado por $-a$.

Um grupo é dito *Abeliano* (ou Comutativo), se além destes quatros axiomas obedecer o axioma da comutatividade, ou seja:

v) Comutatividade: $a * b = b * a$, $\forall\ a, b \in G$.

Definição 2.2 – Um grupo $\langle G,* \rangle$ é dito *finito* se $|G|$ é finita, onde $|G|$ denota a *ordem* ou *cardinalidade do grupo*, ou seja, o número de elementos de G.

Exemplo 2.1: O conjunto Z_n[1], com a operação de adição módulo n, forma um grupo de ordem n. Já o conjunto Z_n juntamente com a operação de multiplicação não forma um grupo já que nem todos os elementos possuem inverso. Porém o conjunto Z_p^* juntamente com a operação de multiplicação módulo p, onde p é primo, forma um grupo.

□

Definição 2.3 – A *ordem de um elemento g* em um grupo é a menor quantidade de vezes que g é operado consigo mesmo, resultando na identidade do grupo. Por exemplo, no caso de um grupo multiplicativo, é o menor expoente que resulta no elemento 1, ou seja, o menor inteiro positivo t tal que $g^t = 1$. Se tal inteiro t não existir, então a ordem de g é definida como ∞.

Definição 2.4 – Um grupo com n elementos e que possui um elemento g de ordem n é dito um *grupo cíclico*; g é chamado elemento gerador do grupo.

[1] Z_n denota o conjunto dos inteiros $\{0, 1, ..., n\text{-}1\}$ e $Z_n^* = Z_n - \{0\}$.

Definição 2.5 – Um subconjunto H não vazio de um conjunto G forma um *subgrupo* de $\langle G,* \rangle$ se H forma um grupo com respeito a operação $*$ de $\langle G,* \rangle$. Se $\langle H,* \rangle$ é um subgrupo de $\langle G,* \rangle$ e $H \neq G$, então $\langle H,* \rangle$ é dito *subgrupo próprio* de $\langle G,* \rangle$.

A prova dos teoremas 2.1 a 2.4 mostrados a seguir pode ser encontrada em [D92].

Teorema 2.1 – Seja $\langle G,* \rangle$ um grupo, e g um elemento de ordem finita t. Então $|\langle g \rangle|$, a ordem do subgrupo gerado por g, é igual a t.

Teorema 2.2 – (*Teorema de Lagrange*) Se $\langle G,* \rangle$ é um grupo finito e $\langle H,* \rangle$ é um subgrupo de $\langle G,* \rangle$, então $|H|$ divide $|G|$. Portanto, se $g \in G$, a ordem de g divide $|G|$.

Teorema 2.3 – Todo subgrupo de um grupo cíclico $\langle G,* \rangle$ também é cíclico. De fato, se $\langle G,* \rangle$ é um grupo cíclico de ordem n, então para cada divisor d de n, G contém exatamente um subgrupo de ordem d.

Teorema 2.4 – Seja $\langle G,* \rangle$ um grupo

i) Se a ordem de $a \in G$ é t, então a ordem de a^k será t/mdc (t,k)

ii) Se $\langle G,* \rangle$ é um grupo cíclico de ordem n e $d|n$, então G tem exatamente $\phi(d)$ elementos de ordem d. Em particular, G tem $\phi(n)$ geradores[2].

2.1.2 – Anéis

Definição 2.6 – Um *anel* $\langle R,+,\cdot \rangle$ é uma estrutura algébrica formada por um conjunto R e duas operações, de modo que apenas uma possui inversa. Portanto, se "+" e "·" são operações do anel, então "-" também é uma operação válida, mas "÷" não é. O anel segue os seguintes axiomas:

i) $\langle R,+ \rangle$ é um grupo abeliano com identidade 0.

ii) A operação $\cdot$ é associativa, ou seja, $a \cdot (b \cdot c) = (a \cdot b) \cdot c = a \cdot b \cdot c$, $\forall\ a, b, c \in R$

[2] Para $n \geq 1$, $\phi(n)$ denota o número de inteiros positivos que não excedem n e são relativamente primos com o mesmo. $\phi(n)$ é a função de Euler.

iii) A operação $\cdot$ é distributiva sobre +, isto é, $a\cdot(b+c) = (a\cdot b)+(a\cdot c)$ e $(b+c)\cdot a=(b\cdot a) + (c\cdot a)$, $\forall\ a, b, c \in R$.

O anel é dito um *anel comutativo* se $a\cdot b = b\cdot a$, $\forall\ a, b \in R$; um anel com identidade se existe uma identidade multiplicativa, denotada por 1, com $1 \neq 0$, tal que $1\cdot a=a\cdot 1=a$, $\forall a \in R$.

Exemplo 2.2 – O conjunto dos inteiros Z juntamente com as operações de adição e multiplicação é um anel comutativo com identidade.

□

2.1.3 – Corpos

Definição 2.7 – Um corpo $\langle F,+,\cdot\rangle$ é uma estrutura algébrica formada por um conjunto F juntamente com duas operações, ambas possuindo inversas. Portanto, se "+" e "$\cdot$" são duas operações válidas no corpo, então "-" e "$\div$" também são permitidas neste corpo. Os seguintes axiomas são obedecidos, $\forall\ a, b, c \in F$:

Axioma	Adição	Multiplicação
Associatividade	$(a+b)+c = a+(b+c)=a+b+c$	$(a\cdot b)\cdot c = a\cdot (b\cdot c)=a.b.c$
Comutatividade	$a+b= b+a$	$a\cdot b=b\cdot a$
Identidade	$a + 0 = a = 0 + a$	$a\cdot 1= a =1\cdot a$
Inversos	$a + (-a) = 0 = (-a) + a$	$a\cdot a^{-1} = 1 =a^{-1}\cdot a$ se $a \neq 0$

Além disso, **.** é distributiva sobre +.

Definição 2.8 – Por definição, a *característica de um corpo* é 0 se 1+1+1+...+1 (m vezes) nunca é igual a 0 para qualquer $m \geq 1$. Caso contrário, a característica de um corpo é o menor inteiro positivo m tal que $\sum_{i=1}^{m} 1$ igual a 0.

Teorema 2.5 – Se a característica m de um corpo não é 0, então m é um número primo.

Definição 2.9 – Se um subconjunto S dos elementos do corpo $<F,+, \bullet>$ satisfaz os axiomas do corpo ,então S é dito um *subcorpo* de F e F é chamado *extensão* de S.

2.1.4 – Anéis Polinomiais

Definição 2.10 – Se R é um anel comutativo, então um polinômio em x sobre o anel R é uma expressão da forma

$$f(x) = a_n x^n + \ldots + a_2 x^2 + a_1 x + a_0, \qquad \textbf{(2.1)}$$

onde cada $a_i \in R$ e $n \geq 0$. O elemento a_i é chamado *coeficiente* de x^i em $f(x)$. O maior inteiro m para o qual $a_m \neq 0$ é chamado *grau* de $f(x)$, denotado por grau($f(x)$); a_m é chamado de coeficiente principal de $f(x)$. Se $f(x) = a_0$ (um polinômio constante) e $a_0 \neq 0$, então $f(x)$ tem grau 0. Se todos os coeficientes de $f(x)$ são 0, então $f(x)$ é chamado polinômio nulo e seu grau, por conveniência matemática, é definido como $-\infty$. O polinômio $f(x)$ é dito ser *mônico* se seu coeficiente principal é igual a 1.

Definição 2.11 – Se R é um anel comutativo, o *anel polinomial* $R[x]$ é o anel formado pelo conjunto de polinômios em x com coeficientes em R. As duas operações utilizadas são a adição e a multiplicação de polinômios, com a aritmética dos coeficientes feitas sobre o anel R.

Deste momento em diante será considerado o anel polinomial sobre um corpo arbitrário F, denotado por $F[x]$.

Definição 2.12 – Seja $f(x) \in F[x]$ um polinômio de grau pelo menos 1. Então $f(x)$ é dito um *polinômio irredutível sobre F* se não pode ser escrito como produto de dois polinômios de grau positivo menor que grau de $f(x)$.

Comparativamente, um polinômio irredutível é como um número primo, não possuindo fatores não triviais. Qualquer polinômio pode ser escrito unicamente (sem considerar fatores constantes) como um produto de polinômios irredutíveis (assim como qualquer número pode ser escrito unicamente como um produto de primos) [D92].

Assim como para os inteiros, pode–se definir congruências para os polinômios em $F[x]$ baseado na divisão por um certo $f(x) \in F[x]$.

Definição 2.13 – Se $g(x)$, $h(x) \in F[x]$, então $g(x)$ é dito *congruente a* $h(x)$ módulo $f(x)$ se $f(x)$ divide $[g(x) - h(x)]$(Denotando-se por $g(x) \equiv h(x) (\mathrm{mod}\ f(x))$).

Seja $f(x)$ um polinômio fixo em $F[x]$. A classe de equivalência de um polinômio $g(x) \in F[x]$ é o conjunto de todos os polinômios em $F[x]$ congruentes a $g(x)$ mod $f(x)$. A relação de congruência mod $f(x)$ particiona $F[x]$ em classes de equivalência [K94].

Definição 2.14 – $F[x]/(f(x))$ denota o conjunto (classes de equivalência) de polinômios em $F[x]$ de grau $n =$ grau($f(x)$). Adição e multiplicação são feitas módulo $f(x)$.

Teorema 2.6 – $F[x]/f(x)$ é um anel comutativo.

Teorema 2.7 – Se f(x) é irredutível sobre F, então $F[x]/f(x)$ é um corpo.

2.1.5 – Corpos Finitos

Definição 2.15 – Um *corpo F* é dito *finito* quando contém um número finito de elementos. A ordem de F é o número de elementos em F.

Teorema 2.8 – A ordem q de um corpo finito F, é uma potência de um número primo p, ou seja, $q=p^m$ com $m \geq 1$.

Este corpo é denotado por F_q ou GF(q) (Campo de Galois, em homenagem ao matemático francês Evariste Galois que viveu entre 1811-1832).

Teorema 2.9 –Seja F_q um corpo finito com q elementos, e seja t um inteiro positivo. Se t não divide $q - 1$, não há elementos de ordem t em F_q; por outro lado, se t divide q-1, então há exatamente $\phi(t)$ elementos de ordem t em F_q.

Corolário – Em todo corpo finito, existe pelo menos um elemento (de fato $\phi(q\text{-}1)$ elementos) de ordem $q - 1$. Daí, portanto, o grupo multiplicativo de qualquer corpo finito é cíclico.

Definição 2.16 – Um elemento de ordem multiplicativa $q - 1$, i. e., um gerador do grupo cíclico $F_q^* = F_q - \{0\}$, é chamado uma *raiz primitiva* (ou *elemento primitivo*) do corpo finito F_q.

Teorema 2.10 – Suponha F_q um corpo finito com $q = p^m$ elementos. Associado com cada $\alpha \in F$, há um único polinômio mônico $p(x) \in F_p(x)$, com as seguintes propriedades.

a) $p(\alpha) = 0$

b) grau(p) $\leq m$.

c) Se $f(x)$ é outro polinômio em $F_p(x)$ com $f(\alpha) = 0$, então $p(x)|f(x)$.

Teorema 2.11 – Se existe um corpo F_q com q elementos, $\forall\ n \geq 1$ existe pelo menos um polinômio irredutível de grau n sobre F.

Do Teorema 2.11, conclui-se que para qualquer inteiro positivo q da forma $q = p^m$, onde p é um primo, existe um corpo finito de ordem q.

Teorema 2.12 – Para qualquer potência de primo p^m, há (sem considerar isomorfismos[3]) um e apenas um corpo finito.

Construção de um corpo finito

Foi visto no Teorema 2.8 que o número q de elementos de um corpo finito é da forma p^m, p primo e $m \geq 1$.

No caso em que $m = 1$, a construção de F_p, consiste simplesmente em aplicar as operações de adição e multiplicação módulo p aos elementos do conjunto Z_p={0, 1,..., p-1}, lembrando que na operação de multiplicação desconsidera-se o elemento 0, a fim de manter a propriedade de inversão.

Exemplo 2.3 – Construção do corpo $GF(5) \equiv \langle \{0,1,2,3,4\}, +_5, \bullet_5 \rangle$

$+_5$	**0**	**1**	**2**	**3**	**4**
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Tabela 2. 1 - Adição dos elementos de GF(5)

$\bullet_5$	**1**	**2**	**3**	**4**
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

Tabela 2. 2 - Multiplicação do elementos de GF(5)

□

[3] Um mapeamento entre dois grupos que preserva a identidade é chamado um *homomorfismo*. Se um homomorfismo possui um inverso que também é um homormofismo, então é chamado um *isomorfismo* e os dois grupos são chamados *isomórficos*. Informalmente falando, dois corpos são ditos isomórficos se eles são estruturalmente os mesmos, embora a representação dos seus elementos seja diferente.

Já no caso em que $m > 1$, os elementos do corpo não são mais números, são polinômios; as operações de adição e multiplicação são feitas módulo um certo polinômio, ao invés de módulo um certo primo.

Para a construção de tais corpos, escolhe-se inicialmente um polinômio de grau m sobre F_p (com coeficientes em F_p), o qual é irredutível sobre F_p. Tal polinômio será denotado por $\Pi(x)$. Neste caso, os elementos de F_p são polinômios sobre F_p com grau $< m$ e as operações são adição e multiplicação módulo $\Pi(x)$.

Exemplo 2.4 – Construção do corpo GF(8) = GF(2^3)

Considera-se neste exemplo o polinômio irredutível de grau 3 sobre GF(2), $\Pi(x) = x^3 + x + 1$. Os elementos do corpo são formados por todos os polinômios com grau menor que 3 sobre GF(2), ou seja, os 8 polinômios 0, 1, x, x+1, x^2, $x^2 + 1$, $x^2 + x$, $x^2 + x + 1$. A seguir são mostradas as tabelas de + e • em GF(8)

Adição mod $\Pi(x)$	0	1	x	x+1	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
0	0	1	x	x+1	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
1	1	0	x+1	X	$x^2 + 1$	x^2	$x^2 + x + 1$	$x^2 + x$
x	x	x+1	0	1	$x^2 + x$	$x^2 + x + 1$	x^2	x^2 +1
x+1	x+1	x	1	0	$x^2 + x + 1$	$x^2 + x$	$x^2 + 1$	x^2
x^2	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$	0	1	x	$x + 1$
$x^2 + 1$	$x^2 + 1$	x^2	$x^2 + x + 1$	$x^2 + x$	1	0	$x + 1$	x
$x^2 + x$	$x^2 + x$	$x^2 + x + 1$	x^2	$x^2 + 1$	x	$x + 1$	0	1
$x^2 + x + 1$	$x^2 + x + 1$	$x^2 + x$	$x^2 + 1$	x^2	$x + 1$	x	1	0

Tabela 2. 3 - Adição mod $\Pi(x) = x^3+x+1$ dos elementos de GF(8)

Multiplicação mod $\Pi(x)$	1	x	x+1	x^2	$X^2 + 1$	$x^2 + x$	$x^2 + x + 1$
1	1	x	x^2	$x + 1$	$X^2 + 1$	$x^2 + x$	$x^2 + x + 1$
x	x	x^2	$x + 1$	$x^2 + x$	1	$x^2 + x + 1$	$x^2 + 1$
$x + 1$	$x + 1$	$x^2 + x$	$x^2 + x + 1$	$x^2 + 1$	x^2	1	x
x^2	x^2	$x + 1$	$x^2 + x$	$x^2 + x + 1$	x	$x^2 + 1$	1
$x^2 + 1$	$X^2 + 1$	1	x	x^2	$x^2 + x + 1$	$x + 1$	$x^2 + x$
$x^2 + x$	$x^2 + x$	$x^2 + x + 1$	$x^2 + 1$	1	$x + 1$	x	x^2
$x^2 + x + 1$	$x^2 + x + 1$	$x^2 + 1$	1	x	$x^2 + x$	x^2	$x + 1$

Tabela 2. 4 - Multiplicação mod $\Pi(x) = x^3+x+1$ dos elementos de GF(8).

□

Definição 2.17 – Seja Π (x) um polinômio irredutível de grau m sobre GF(p). Diz-se que $\Pi(x)$ pertence ao expoente e se $\Pi(x) | (x^e-1)$, e não divide (x^n-1) para $n<e$. Além disso, se $e = p^m - 1$, $\Pi(x)$ é chamado *polinômio primitivo*.

Exemplo 2.5 –Pela definição 2.17, um polinômio de grau *m* sobre GF(2) é primitivo se o expoente ao qual ele pertence é 2^m-1. Considerando o corpo GF(16), o polinômio primitivo $\Pi(x)$ deve pertencer ao expoente 15 e não dividir (x^n-1) para $n < 15$. Um exemplo de polinômio primitivo neste corpo é $\Pi(x) = x^4 + x + 1$.

Os elementos de GF(16) podem ser representados como polinômios (0, 1, x, $x+1$, x^2, x^2+1, x^2+x, x^2+x+1, x^3, x^3+1, x^3+x, x^3+x+1, x^3+x^2, x^3+x^2+1, x^3+x^2+x, x^3+x^2+x+1), bem como através de *m*-uplas binárias.

Uma outra representação é possível considerando as potências de um elemento α, raiz do polinômio primitivo $\Pi(x)$, i. e., $\Pi(\alpha) = \alpha^4 + \alpha + 1 = 0$.

α^1	$\alpha^5 = \alpha$	$\alpha^9 = \alpha^3 + \alpha$	$\alpha^{13} = \alpha^3$
α^4	$\alpha^6 = \alpha$	$\alpha^{10} = \alpha^2 + \alpha$	$\alpha^{14} = \alpha^3$
α^3	$\alpha^7 = \alpha$	$\alpha^{11} = \alpha^3 + \alpha$	$\alpha^{15} = 1$
α^4	$\alpha^8 = \alpha$	$\alpha^{12} = \alpha^3 + \alpha$	

Então, observa-se que α é um gerador do grupo multiplicativo de GF(16) (ou elemento primitivo de GF(16)) e as potências de α geram os elementos de GF(16). A tabela 2.5 mostra as três representações dos elementos de GF(16).

Polinômio	-upla	Potência de α
0	000	$\alpha^{-\infty}$
1	001	α^{15}
x	010	α^1
x		α^4

	011	
x^1	100	α^2
x^1	101	α^8
x^1	110	α^5
x^1	111	α^{10}
x^1	000	α^5
x^1	001	α^{14}
x^1	010	α^9
x^1	011	α^7
x^1	100	α^6
x^1	101	α^{13}
x^1	110	α^{11}
x^3	111	α^{12}

Tabela 2. 5 - Representações dos elementos do corpo GF(16)

2.2 – Curvas Elípticas sobre Corpos Finitos

A Teoria de Curvas Elípticas sobre Corpos Finitos, tópico comum a Teoria dos Números e Geometria Algébrica, vem sendo estudada a mais de um século. Tais estruturas

algébricas tiveram um grande destaque a partir da recente prova do Último Teorema de Fermat por Andrew Wiles [www2.1].

Recentemente, a teoria de Curvas Elípticas sobre Corpos Finitos vem sendo usada de forma prática na construção de algoritmos para fatoração de inteiros, teste de primalidade e em Criptografia de Chave Pública. A primeira proposta para o uso de Curvas Elípticas em Criptografia se deu em meados dos anos 80 de forma independente por Neal Koblitz [K87] e Victor Miller [M86]. Tais aplicações relacionadas à área de Criptografia se devem principalmente ao fato de que Curvas Elípticas sobre Corpos Finitos proporcionam uma fonte ilimitada de grupos abelianos finitos que, mesmo quando grandes, são facilmente manipulados devido a sua rica estrutura. Desta forma, as ferramentas criptográficas que antes utilizavam comumente grupos multiplicativos de inteiros módulo *n* em sua construção, passaram a utilizar Curvas Elípticas sobre Corpos Finitos, pois a partir da análise de tais estruturas notou-se que as mesmas são análogas naturais dos grupos multiplicativos de corpos finitos.

As vantagens obtidas ao se utilizar curvas elípticas no lugar de grupos multiplicativos abelianos em cripto-sistemas serão abordados com mais atenção no capítulo 3, o qual é dedicado ao estudo de cripto-sistemas baseados em curvas elípticas.

2.2.1 – Cônicas Racionais

A fim de compreender a estrutura de uma curva elíptica e como seus pontos formam um grupo abeliano, serão introduzidos inicialmente alguns conceitos básicos sobre cônicas visando facilitar o entendimento das cúbicas e em especial das curvas elípticas.

Pontos Racionais sobre Cônicas

Considere inicialmente pontos racionais sobre cônicas. Sabe-se que os *números racionais* são aqueles que podem ser representados pelo quociente de dois inteiros. Diz-se que um ponto (*x*, *y*) é um *ponto racional* se tanto *x* quanto *y* são números racionais. Uma reta é dita racional se os coeficientes de sua equação são números racionais, i. e., se em

$$ax + by + c = 0 \qquad \textbf{(2.2)}$$

a , *b*, *c* são números racionais. A partir de (2.2) verifica-se que através de dois pontos racionais passa uma reta racional e que quando duas retas racionais se cruzam, o ponto de interseção das duas é um ponto racional.

Antes das cúbicas que são o ponto principal desta seção, mais especificamente as curvas elípticas, serão consideradas as cônicas racionais que possuem como equação geral

$$ax^2 + bxy + cy^2 + dx + ey + f = 0 \qquad \textbf{(2.3)}$$

com os coeficientes *a*, *b*, *c*, *d*, *e* e *f* racionais. Pode-se observar utilizando (2.2) e (2.3) que a interseção de uma reta racional e uma cônica racional, resultará em dois pontos racionais, se e só se a equação quadrática obtida a partir de (2.2) e (2.3) possuir raízes racionais. Porém se um desses pontos for racional o outro também o será, uma vez que a equação quadrática mônica obtida terá seus coeficientes racionais e o coeficiente do termo em *x* é a soma das raízes da equação quadrática. Daí, surge uma idéia que faz com que seja possível, partindo-se de um ponto conhecido da cônica racional obter outros pontos racionais sobre a mesma. Considerando que um certo ponto O racional sobre a cônica racional é conhecido, o procedimento consiste em se fazer a projeção do ponto O sobre uma reta que não intercepta a curva, passando uma reta racional pelo ponto O em questão e interceptando a reta externa a cônica racional. Assim, a reta passa por O, intercepta a cônica em um ponto *P* e a outra reta num ponto *Q* (Figura 2.1).

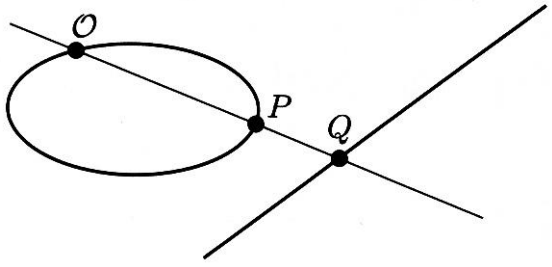


Figura 2. 19 - Projeção de cônica sobre uma reta.

Observa-se desta forma que a cada ponto *P* sobre a cônica obtém-se um novo ponto *Q* sobre a reta. Por outro lado, passando uma reta por O e *Q* obtém-se para cada novo *Q* na reta um novo *P* na curva. Conclui-se a partir daí que existe uma relação de um para um entre o ponto *Q* da reta e o ponto *P* da cônica. Assim, usando este procedimento é possível,

tendo um ponto conhecido sobre a cônica, obter-se outros pontos, como será visto mais claramente no decorrer desta seção.

Do que foi comentado anteriormente sobre retas racionais e cônicas racionais, conclui-se que *P* é um ponto racional, uma vez que o ponto o que pertence tanto a reta racional quanto a cônica racional, é também racional. Por outro lado, conclui-se que *Q* também é um ponto racional uma vez que a reta que intercepta a cônica é racional e o ponto o é racional.

Exemplo 2.6 : Considere o círculo (Figura 2.2).

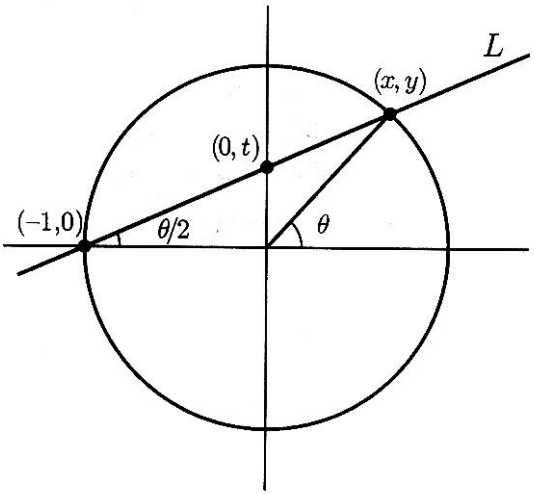


Figura 2. 20 - Projeção do círculo $x^2+y^2 = 1$ sobre a reta $x=0$.

Projetando o ponto (-1,0) sobre o eixo *y* obtém-se o ponto (0,*t*). Se o ponto (*x*, *y*) também pertence à reta e à cônica e se este ponto é conhecido, então *t* pode ser obtido facilmente. Ou seja, considerando (i) a equação da reta *L* que conecta (-1,0) a (0,*t*), i. e., $y = t(1+x)$ e (ii) que o ponto (*x*, *y*) está tanto na reta quanto no círculo, tem-se que

$$1-x^2 = y^2 = t^2(x+1)^2 \Rightarrow (1-x)(1+x) = t^2(x+1)(x+1) \Rightarrow 1-t^2 = x(1+t^2) \text{ para } x \neq -1$$

$$\Rightarrow x = \frac{1-t^2}{1+t^2} \qquad \textbf{(2.4a)}$$

$$y = t(1+x) = t\left(1+\frac{1+t^2}{1-t^2}\right) = t\left(\frac{1-t^2+1+t^2}{1-t^2}\right) = t\left(\frac{2}{1-t^2}\right)$$

$$\Rightarrow y = \frac{2t}{1-t^2} \qquad \textbf{(2.4b)}$$

Portanto, das expressões (2.4a e 2.4b) obtidas, conclui-se que sendo (*x*,*y*) um ponto racional, (0,*t*) também será um ponto racional e vice-versa.

□

2.2.2 - Cúbicas

Neste ponto é iniciado o estudo das cúbicas. Seja

$$ax^3 + bx^2y + cxy^2 + dy^3 + ex^2 + fxy + gy^2 + hx + iy + j = 0 \qquad \textbf{(2.5)}$$

a equação geral de uma cúbica. Diz-se que tal cúbica é racional se os coeficientes da equação (2.5) são racionais.

Uma reta geralmente intercepta uma cúbica em três pontos, portanto, o princípio que foi usado nas cônicas não funcionará aqui, uma vez que, neste caso, o ponto na reta corresponderá a dois pontos na cúbica. Porém, existe um princípio geométrico que pode ser usado neste caso, que diz que se for possível encontrar dois pontos sobre a cúbica, o terceiro ponto poderá ser facilmente obtido. O procedimento consiste em conectar dois pontos conhecidos da cúbica racional. A reta racional resultante neste procedimento interceptará um terceiro ponto (Figura 2.3*a*).

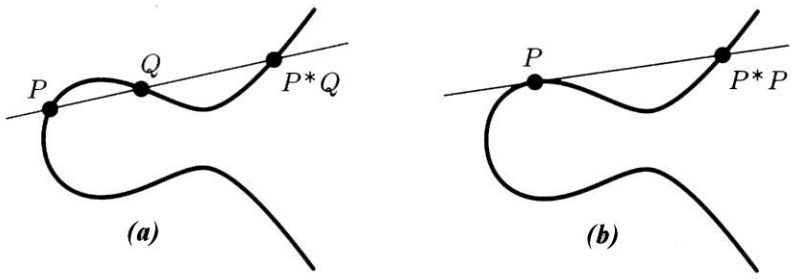


Figura 2. 21 - Composição dos pontos de uma cúbica.

Ao proceder com a interseção da cúbica racional (2.5) com a reta racional (2.2), obtém-se uma equação cúbica com coeficientes racionais. Supondo que dois pontos racionais que pertencem tanto a cúbica quanto a reta são conhecidos, o terceiro ponto racional será obtido facilmente uma vez que o coeficiente de x^2 é a soma das raízes do polinômio cúbico mônico. Fica claro, portanto, que o terceiro ponto obtido será também racional.

Na figura 2.3*a*, observa-se a composição dos pontos *P* e *Q*. Como já foi descrito, a composição consiste em passar uma reta por esses pontos resultando no ponto $P * Q$. No caso de apenas um ponto ser conhecido, seguindo a mesma regra a reta ligará *P* a *P*, o que se resume a ter uma reta tangente à cúbica no ponto *P*. O outro ponto obtido será denominado $P * P$ (Figura 2.3*b*). Seguindo com tal procedimento, muitos outros pontos podem ser obtidos. No caso de uma cúbica racional não-singular, segundo o Teorema de Mordell[4] (1921), se a curva tem um ponto racional, então o grupo infinito de seus pontos racionais é finitamente gerado [ST92].

A partir do que foi colocado sobre essa regra de composição e do Teorema de Mordell, fica uma pergunta: "Os pontos de uma cúbica racional não-singular formam um grupo?" Infelizmente, a resposta é não, devido a falta de um elemento identidade. Porém, essa situação pode ser mudada considerando um certo ponto racional ○ que funcionará como elemento identidade do grupo. Denotando por + a operação do grupo, a regra para se adicionar dois pontos, *P* e *Q*, sobre a cúbica racional, é a seguinte:

$$P+Q = \circ * (P * Q) \qquad \textbf{(2.6)}$$

Esta lei de composição do grupo é ilustrada na figura 2.4.

[4] Teorema de Mordell: Se uma cúbica plana não-singular tem um ponto racional, então o grupo de pontos racionais é finitamente gerado.

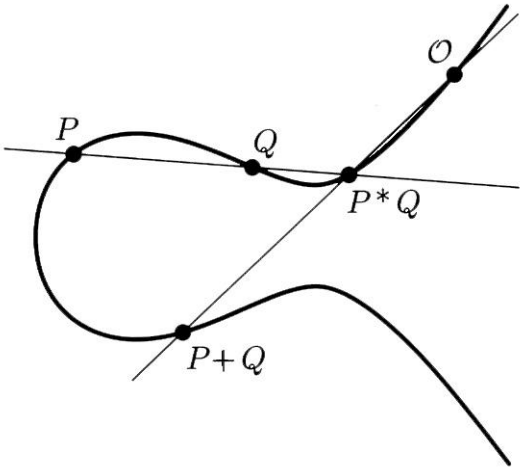


Figura 2. 22 - A lei do grupo sobre uma cúbica.

Observa-se facilmente que tal operação é comutativa uma vez que *P*+*Q* = *Q*+*P*. Na figura a seguir, pode se observar que o elemento o é realmente o elemento identidade, utilizando o procedimento já descrito, pois *P*+ o = o + *P* = *P*.

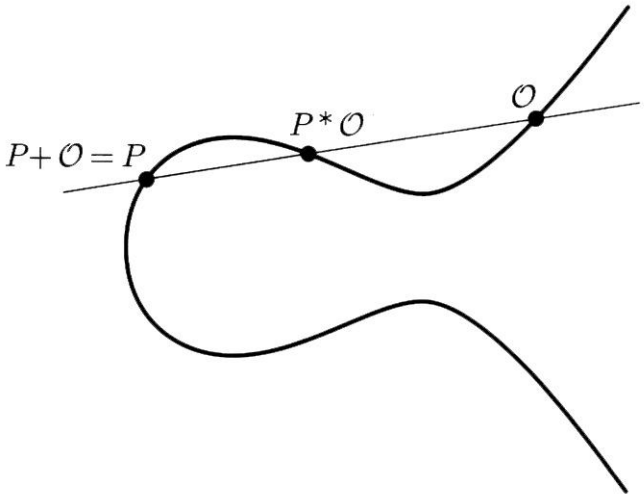


Figura 2. 23 - O elemento identidade.

O elemento inverso não é obtido tão facilmente. Para se obter o negativo de um certo ponto, por exemplo o ponto *Q*, passa-se uma reta tangente ao elemento o; tal reta conectará a curva em mais um ponto, que será denotado por *S*. Obtido este ponto, o mesmo é conectado ao ponto ao qual se deseja obter o negativo, neste caso o ponto *Q*. O terceiro ponto de interseção entre a reta que passa por *S* e *Q* será o negativo de *Q*, denotado por –*Q* (Figura 2.6).

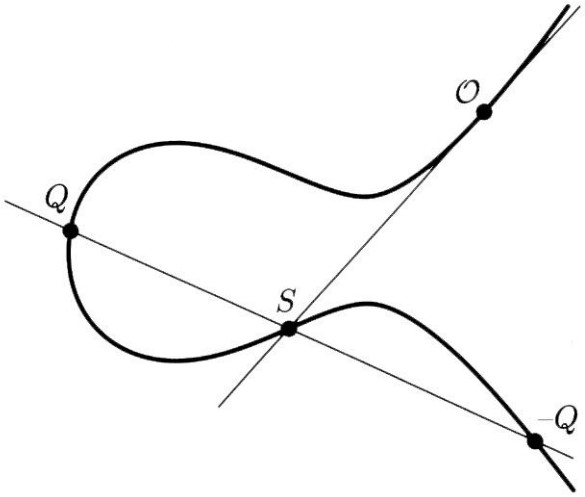


Figura 2. 24 - O negativo de um ponto.

Do que foi posto até o momento, observa-se que a única propriedade que não foi demonstrada e que definitivamente mostra que os pontos de uma cúbica racional não-singular juntamente com a operação já descrita forma um grupo foi a associatividade, a qual é mostrada a seguir.

Sejam *P*, *Q* e *R* três pontos na curva. Deseja-se provar que $(P+Q)+R=P+(Q+R)$. Para isso será usado um procedimento gráfico no qual a operação de adição de pontos de uma cúbica já definida será aplicada.

Para se obter $P+Q$, toma-se o terceiro ponto de interseção da reta que passa por *P* e *Q*, i.e., o ponto $P*Q$ e faz-se a conexão do mesmo ao ponto o. A partir daí $(P+Q)+R$ é obtido conectando-se o terceiro ponto de intercessão da reta que passa por $P+Q$ e *R*, ou seja o ponto $(P+Q)*R$, com a cúbica ao ponto o.

De forma semelhante obtém-se $P+(Q+R)$, desta vez adicionando-se inicialmente *Q* e *R* e o resultado desta adição sendo adicionado ao ponto *P*.

Observa-se na Figura 2.7 que para mostrar que $(P+Q)+R=P+(Q+R)$ basta mostrar que $(P+Q)*R=P*(Q+R)$. Nota-se, na figura em questão, que a reta pontilhada contendo os pontos *P*+*Q* e *R*, e a reta sólida que contém os pontos *P* e $Q+R$, tem a sua interseção sobre a cúbica. Portanto, está provado que $(P+Q)*R=P*(Q+R)$ e consequentemente que $(P+Q)+R=P+(Q+R)$, sendo portanto esta uma operação associativa.

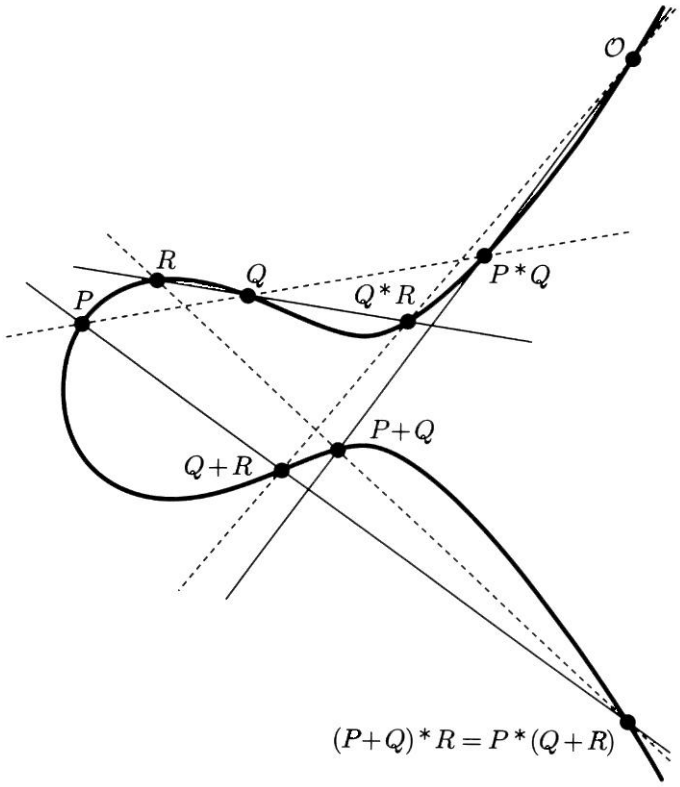


Figura 2. 25 - Verificação da associatividade.

Concluímos desta forma que os pontos da cúbica racional juntamente com a operação descrita acima, tendo como elemento identidade o ponto O, formam um grupo abeliano.

A seguir será introduzida a forma normal de Weierstrass, que possibilitará entre outras a obtenção de fórmulas analíticas para a adição de pontos sobre a curva. A partir dessas fórmulas também é possível demonstrar a associatividade desta operação e também as outras propriedades que foram ilustradas apenas de forma gráfica nesta seção.

Forma Normal de Weierstrass

Na próxima seção serão mostradas as fórmulas analíticas para a adição de pontos numa cúbica racional. Para tornar essas fórmulas as mais simples possíveis, é importante saber que qualquer cúbica com um ponto racional pode ser transformada na forma normal

de Weierstrass. Será necessário um conhecimento básico sobre geometria projetiva (apêndice A) para introduzir esta forma de se representar as cúbicas.

A forma clássica para a equação de uma cúbica na forma normal de Weierstrass é

$$y^2 = 4x^3 - g_2 x - g_3 \qquad \textbf{(2.7)}$$

ou ainda, de forma mais geral,

$$y^2 = x^3 + ax^2 + bx + c \qquad \textbf{(2.8)}$$

Torna-se necessário neste momento, a fim de se chegar à forma normal de Weierstrass mostrar que qualquer cúbica é equivalente de forma *bi-racional* às cúbicas deste tipo.

Suponha uma cúbica qualquer num plano projetivo. A idéia principal consiste em escolher os eixos do plano projetivo de modo a simplificar a forma da curva. Considere um ponto conhecido o sobre a curva *C*, tome inicialmente *Z* = 0, tangente à curva *C* no ponto o. Esta reta intercepta a curva em um outro ponto como pode ser observado na Figura 2.8. Neste ponto passa-se uma nova reta tangente, *X* = 0. Finalmente, escolhe-se Y = 0 como outra reta (diferente de *Z* = 0) que intercepta o (Assume-se que não é um ponto de inflexão; de outra forma X = 0 seria qualquer reta contendo o).

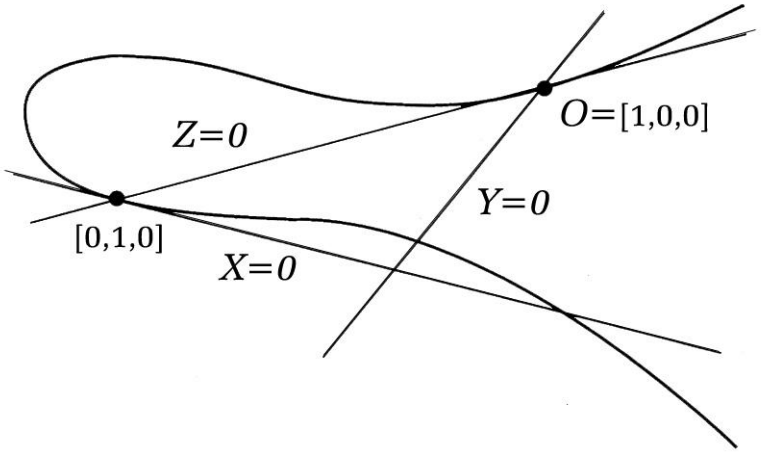


Figura 2. 26 - Escolha de eixos para colocar *C* na formal normal de Weiertrass.

Escolhendo os eixos dessa forma e fazendo $x = \frac{X}{Z}$ e $y = \frac{Y}{Z}$, serão conseguidas algumas condições lineares sobre a forma que a equação tomará nessas coordenadas. Isto é

chamada transformação projetiva. Aqui não será trabalhada a álgebra, mas o resultado obtido após toda a manipulação matemática sobre a equação da curva *C* será [K84].

$$xy^2 + (ax + b)y = cx^2 + dx + e \qquad (\mathbf{2.9})$$

Multiplicando por *x*,

$$(xy)^2 + (ax + b)xy = cx^3 + dx^2 + ex \qquad (\mathbf{2.10})$$

Substituindo *xy* por *y*, tem-se

$$y^2 + (ax + b)y = cx^3 + dx^2 + ex \qquad (\mathbf{2.11})$$

Colocando no lugar da variável y a expressão $y - \frac{1}{2}(ax + b)$, obtém-se

$$y^2 = cx^3 + dx^2 + ex \qquad (\mathbf{2.12})$$

Desta forma chegou-se a equação na forma normal de Weierstrass.

Observando-se todas as transformações a partir das coordenadas originais até a obtenção da equação na forma normal de Weierstrass, nota-se que as transformações não foram lineares mas sim racionais, i. e., as novas coordenadas são dadas por razões dos polinômios nas antigas coordenadas. Portanto, pontos racionais na curva original correspondem a pontos racionais na nova curva.

2.2.3 - Curvas Elípticas

Uma equação cúbica em sua forma normal é

$$y^2 = f(x) = x^3 + ax^2 + bx + c \qquad (\mathbf{2.13})$$

Se o polinômio cúbico de (2.13) tiver raízes distintas então tal cúbica é chamada *curva elíptica*. Essa nomenclatura não vem da forma da curva que nada parece com uma elipse, mas do fato que tais curvas surgiram no estudo de como calcular o comprimento do arco de uma elipse.

Considerando os coeficientes de (2.13), números racionais (particularmente números reais) o polinômio *f*(*x*) de grau 3 tem pelo menos uma raiz real. Então, considerando coeficientes reais, o polinômio *f*(*x*) é fatorado da seguinte forma

$$f(x) = (x - \alpha)(x^2 + \beta x + \gamma) \qquad (\mathbf{2.14})$$

com α, β, γ reais. O polinômio acima pode ter as três raízes reais ou apenas uma raiz real. No caso de apenas uma raiz real, a curva será semelhante a da figura 2.9, pois $y = 0$ e $x = \alpha$.

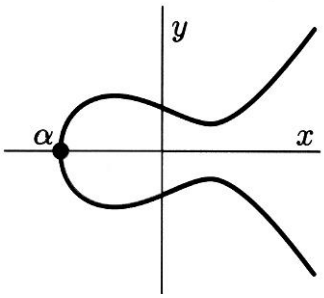


Figura 2. 27 - Curva elíptica com um componente real.

Se $f(x)$ tem as três raízes reais, então a curva será similar à curva da figura 2.10; neste caso os pontos reais formam duas componentes.

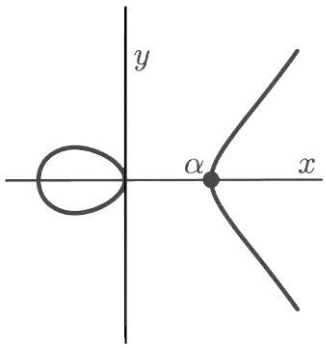


Figura 2. 28 - Curva Elíptica com dois componentes reais.

Tudo isto é válido se as raízes de *f*(*x*) forem distintas. A importância deste fato será explicada agora. Desde o início assumiu-se que a curva cúbica era não-singular. Escrevendo a equação (2.13) como $F(x, y) = y^2 - f(x) = 0$ e tomando as suas derivadas parciais

$$\frac{\partial F}{\partial x} = -f'(x), \qquad \frac{\partial F}{\partial y} = 2y, \qquad \textbf{(2.15)}$$

diz-se que uma curva é não-singular quando nenhum dos pontos sobre a curva tem ambas as derivadas parciais nulas. Em outras palavras, isto significa que cada ponto da curva possui uma reta tangente bem definida.

Considere o ponto (x_0, y_0). Se as derivadas parciais (2.15) forem ambas nulas em (x_0, y_0), então $y_0 = 0$ e portanto $f(x_0) = 0$ e $f'(x_0) = 0$. Portanto tanto $f(x_0)$ como $f'(x_0)$ tem como raiz x_0. Logo x_0 é uma raiz dupla de f, i. e., o ponto $(x_0,0)$ é um ponto singular da curva.

Existem dois tipos de singularidade:

1) Se $f(x)$ tem uma raiz dupla.

Neste caso uma equação típica é

$$y^2 = x^2(x+1) \qquad (\mathbf{2.16})$$

e a curva possui duas tangentes distintas na origem (Figura 2.11).

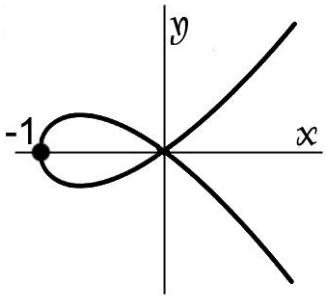


Figura 2. 29 - Exemplo de cúbica singular com raiz dupla.

2) Se $f(x)$ tem uma raiz tripla, então depois de se transladar x obtém-se a equação

$$y^2 = x^3 \qquad (\mathbf{2.17})$$

que é uma parábola semicúbica com um cúspide na origem (Figura 2.12).

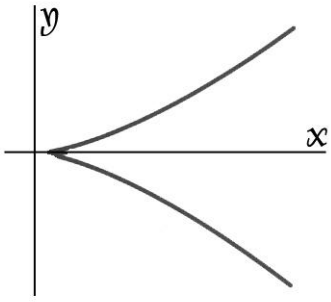


Figura 2. 30 - Exemplo de cúbica singular com raiz tripla.

Estes ((2.16) e (2.17)) são exemplos de cúbicas singulares na forma normal de Weierstrass, e o caso geral será semelhante a esses exemplos depois de uma mudança nas coordenadas.

As cúbicas singulares e não-singulares se comportam de modo diferente, por essa razão e devido ao propósito de formar um grupo com os pontos da curva, o que só ocorre com curvas não-singulares, se priorizou o estudo destas.

O estudo das cúbicas singulares é tão simples quanto o estudo das cônicas. É possível se usar o método de projeção usado no caso das cônicas, obtendo uma relação de um para um entre os pontos da cúbica racional e da reta racional. De fato, isto é feito facilmente de maneira analítica

Exemplo 2.7: Considere a curva (2.16) mostrada na figura 2.11. Considerando $r = \frac{y}{x}$, então (2.16) será transformada em

$$r^2 = x + 1, \qquad (\mathbf{2.18})$$

portanto $x = r^2 - 1$ e $y = r^3 - r$ ao tomar r como um número racional e considerando a relação do mesmo com x e y, será obtido um ponto racional sobre a cúbica; por outro lado, considerando um ponto racional (x, y) sobre a cúbica, obtém-se um número racional r. Essas operações são inversas umas das outras, e estão definidas em todos os pontos com exceção do ponto de singularidade (0,0) sobre a curva.

□

Exemplo 2.8: Considere a curva (2.17) mostrada na figura 2.12. Este caso é ainda mais simples. A transformação consiste apenas em fazer

$$x = t^2 \text{ e } y = t^3 \qquad (\mathbf{2.19})$$

□

Conclui-se portanto que as curvas singulares são tão simples quanto as cônicas porém o Teorema de Mordell não é obedecido neste caso. Na verdade, ainda não se obteve uma operação entre os pontos deste tipo de curva a fim de se caracterizar um grupo, mas devido ao Teorema de Mordell, evitando tais curvas e trabalhando-se com curvas não-singulares, forma-se um grupo com os pontos desta curva juntamente com a operação descrita na seção 2.2.2. A demonstração do Teorema de Mordell está fora do

escopo deste trabalho, para os que desejarem maiores detalhes, a prova se encontra no capítulo 3 de [ST92].

Fórmulas Analíticas para a Adição de pontos em Curvas Elípticas.

Neste momento os pontos de uma curva elíptica serão analisados mais cuidadosamente chegando-se inclusive a fórmulas analíticas para a adição de pontos sobre a curva. A teoria expressa nesta seção necessita de uma certa base de Geometria Projetiva que pode ser obtida no apêndice A.

Considere a seguinte equação na forma normal de Weierstrass

$$y^2 = x^3 + ax^2 + bx + c \qquad (\mathbf{2.20})$$

e torne a mesma homogênea fazendo $x = \frac{X}{Z}$ e $y = \frac{Y}{Z}$. A partir dessas substituições em (2.20) chega-se a

$$Y^2Z = X^3 + aX^2Z + BXZ^2 + CZ^3 \qquad (\mathbf{2.21})$$

A fim de se obter a interseção da cúbica em questão com a reta no infinito em $Z = 0$, faz-se $Z = 0$ em (2.21). Daí obtém-se $X^3 = 0$, ou seja uma raiz tripla em $X = 0$, o que significa que a cúbica intercepta a reta no infinito em três pontos, porém esses pontos são os mesmos, logo a cúbica tem exatamente um ponto no infinito com multiplicidade tripla, sendo este o ponto onde as linhas verticais (X=constante) se encontram. O ponto no infinito é um ponto de inflexão da cúbica, e a reta tangente neste ponto é a reta no infinito, que o toca com multiplicidade três. Observando-se as derivadas parciais em O nota-se que o mesmo é um ponto não-singular. O ponto O em questão, chamado ponto no infinito, é considerado o elemento identidade o qual juntamente com os pontos da curva elíptica e com a operação de adição de pontos definida anteriormente, forma um grupo. Para que isto realmente ocorra algumas considerações devem ser feitas; os pontos na curva elíptica devem ser pontos no plano afim *xy* juntamente com o ponto O no infinito. Desta forma, nota-se que realmente toda reta intercepta a curva em três pontos. A reta no infinito, como foi visto intercepta o ponto no infinito, O, três vezes. Uma reta vertical intercepta a curva em dois pontos no plano *xy* e o ponto no infinito. E uma reta não vertical intercepta a cúbica em três pontos no plano *xy*.

Depois de tudo que foi dito até agora, o estudo da estrutura do grupo formado pelos pontos da curva elíptica e do ponto O pode ser iniciado, obtendo-se por conseqüência as fórmulas analíticas da adição de pontos sobre a curva. A adição dos pontos P e Q sobre a curva elíptica colocada na forma normal de Weierstrass, ocorre da seguinte forma.

a) Traça-se uma reta que passa por P e Q e obtém-se a terceira interseção $P*Q$. Traça-se uma nova reta que conecta o ponto $P*Q$ e o ponto O, ou seja uma reta vertical passando por $P*Q$, obtendo-se como terceiro ponto de interseção o ponto $P*Q$ refletido em relação ao eixo dos x, sendo este o ponto $P+Q$ desejado (Figura 2.13). Isso porque uma cúbica na forma de Weierstrass é simétrica em relação ao eixo dos x.

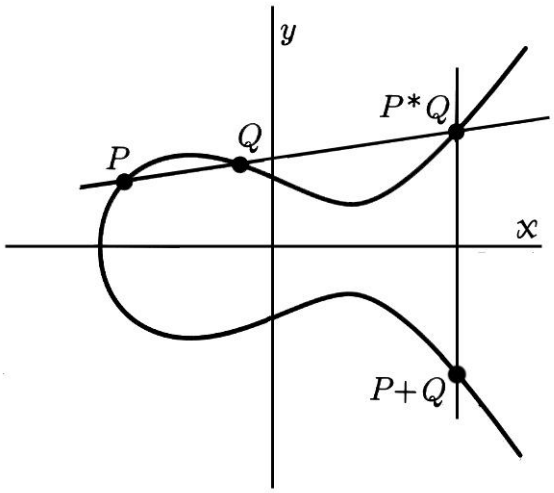


Figura 2. 31 - Adicionando pontos sobre uma cúbica na forma de Weierstrass.

O negativo do ponto Q será simplesmente a reflexão do mesmo com relação ao eixo x, i. e., se $Q = (x, y)$ então, $-Q=(x,-y)$ (Figura 2.12). Isso por que $Q+(-Q) = O$ e, como foi visto, o ponto O é o ponto onde as retas verticais se encontram e, como a curva é simétrica, o negativo de Q será a sua reflexão com relação ao eixo das abscissas. Isso só não é válido para o caso em que $Q= O$, mas neste caso é obvio que $O = - O$.

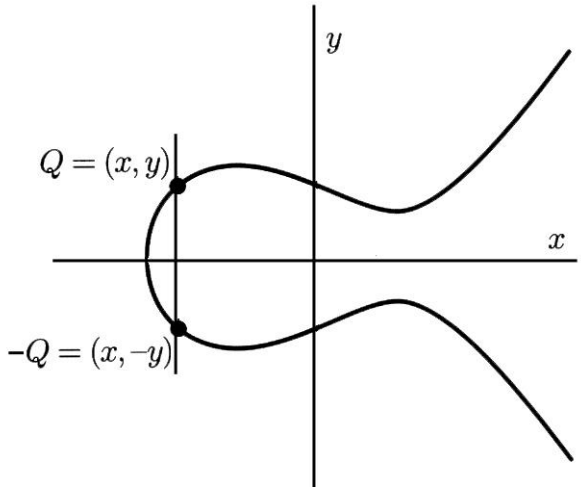


Figura 2. 32 - Negativo de um ponto sobre uma curva elíptica.

Para a obtenção das fórmulas analíticas da soma dos pontos considere $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, $P_1 * P_2 = (x_3, y_3)$ **e** $P_1 + P_2 = (x_3, -y_3)$**. Assume-se que** $P_1 = (x_1, y_1)$ **e** $P_2 = (x_2, y_2)$ **são dados, e deseja-se computar** $P_1 * P_2 = (x_3, y_3)$ **(figura 2.15). A reta que intercepta P_1 e P_2 possui a seguinte equação.**

$$y = \alpha x + \beta, \tag{2.22}$$

onde

$$\alpha = \frac{y_2 - y_1}{x_2 - x_1} \tag{2.23}$$

e

$$\beta = y_1 - \alpha x_1 = y_2 - \alpha x_2 \tag{2.24}$$

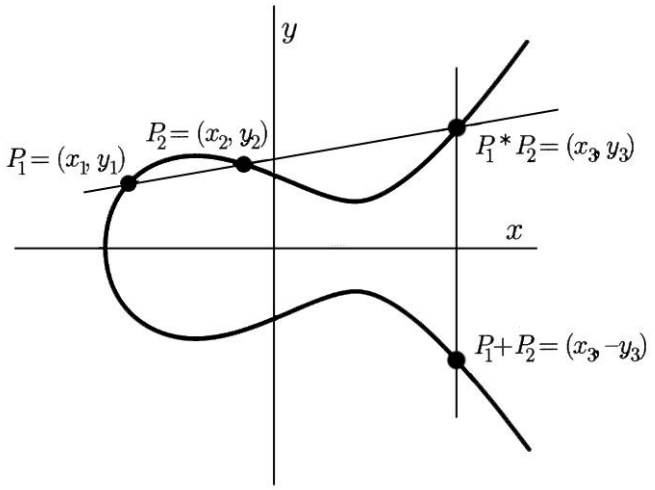


Figura 2. 33 - Derivação da fórmula de adição.

Por construção, a reta em questão intercepta três pontos, P_1, P_2 e P_1*P_2. A fim de obter-se P_1*P_2 substitui-se (2.22) em (2.20) obtendo-se

$$y^2 = (\alpha x + \beta)^2 = x^3 + ax^2 + bx + c \qquad (\mathbf{2.25})$$

e

$$x^3 + (a - \alpha^2)x^2 + (b - 2\alpha\beta)x + (c - \beta^2) = (x - x_1)(x - x_2)(x - x_3)$$
$$= x^3 - (x_1 + x_2 + x_3)x^2 + (x_1x_2 + x_1x_3 + x_2x_3)x - x_1x_2x_3$$

Da expressão acima observa-se que $(\alpha^2 - a) = (x_1 + x_2 + x_3)$, logo

$$x_3 = \alpha^2 - a - x_1 - x_2 \qquad (\mathbf{2.26a})$$

e

$$y_3 = \alpha x_3 + \beta = \alpha x_3 + y_1 - \alpha x_1 = y_1 + \alpha(x_3 - x_1) \qquad (\mathbf{2.26b})$$

Assim, obtém-se

$$P_1 + P_2 = \left(\alpha^2 - a - x_1 - x_2, -y_1 + \alpha(x_1 - x_3)\right) \qquad (\mathbf{2.27})$$

No caso considerado $x_1 \neq x_2$, o que implica em (2.23), mas no caso da duplicação de pontos, i. e., P_1+P_1 = 2P_1, considera-se, como mencionado anteriormente, uma reta tangente à curva no ponto a ser duplicado, tendo-se neste caso, a partir da relação $y^2 = f(x)$,

$$\alpha = \frac{dy}{dx} = \frac{f'(x)}{2y} \qquad (\mathbf{2.28})$$

Exemplo 2.9: Seja $y^2 = x^3 + 17$. Adicione os pontos P_1=(-1,4) e P_2=(2,5) pertencentes a curva e duplique o ponto P_1.

$\alpha = \frac{5-4}{2-(-1)} = \frac{1}{3}$. Considerando os pontos dados e o valor de α calculado, utilizando (2.27) obtém-se

$$x_3 = \alpha^2 - x_1 - x_2 = \left(\frac{1}{3}\right)^2 - (-1) - 2 = \left(\frac{1}{9}\right) - 1 = -\frac{8}{9} \ ,$$

$$-y_3 = -y_1 + \alpha(x_1 - x_3) = -4 + \left(\frac{1}{3}\right)\left((-1) - \left(-\frac{8}{9}\right)\right) = -\frac{109}{27}. \text{ Logo, } P_1 + P_2 = \left(-\frac{8}{9}, -\frac{109}{27}\right).$$

Para duplicar P_1, um novo valor de α será calculado já que x_1=x_2 neste caso. Assim,

$$\alpha = \left.\frac{dy}{dx}\right|_{P_1} = \frac{f'(x_1)}{2y_1} = \frac{3x_1^2}{2y_1} = \frac{3(-1)^2}{2.4} = \frac{3}{8}$$

$$x_3 = \alpha^2 - 2x_1 = \left(\frac{3}{8}\right)^2 - 2(-1) = \left(\frac{9}{64}\right) + 2 = \frac{137}{64}$$

$$-y_3 = -y_1 + \alpha(x_1 - x_3) = -4 + \left(\frac{3}{8}\right)\left((-1) - \left(\frac{137}{64}\right)\right) = -\frac{2651}{512}$$

Logo,

$$2P_1 = \left(\frac{137}{64}, -\frac{2651}{512}\right)$$

□

Resumindo, existem cinco possibilidades:

1) Se o ponto P é o ponto no infinito O, então define-se $-P$ como O. Uma vez que O é o elemento identidade do grupo formado pelos pontos de uma curva elíptica.

Supondo dois pontos P e Q, ambos diferentes de O, segue-se

2) Se $P = (x,y)$, então o negativo de P, denotado por $-P = (x, -y)$.

3) Se P e Q tem coordenadas x diferentes, $P+Q$ será a reflexão em relação ao eixo x do terceiro ponto de interseção da reta com a cúbica, que passa por P e Q (exemplo na figura 2.16). Analiticamente, $P+Q$ é obtido através das equações (2.27) e (2.23).

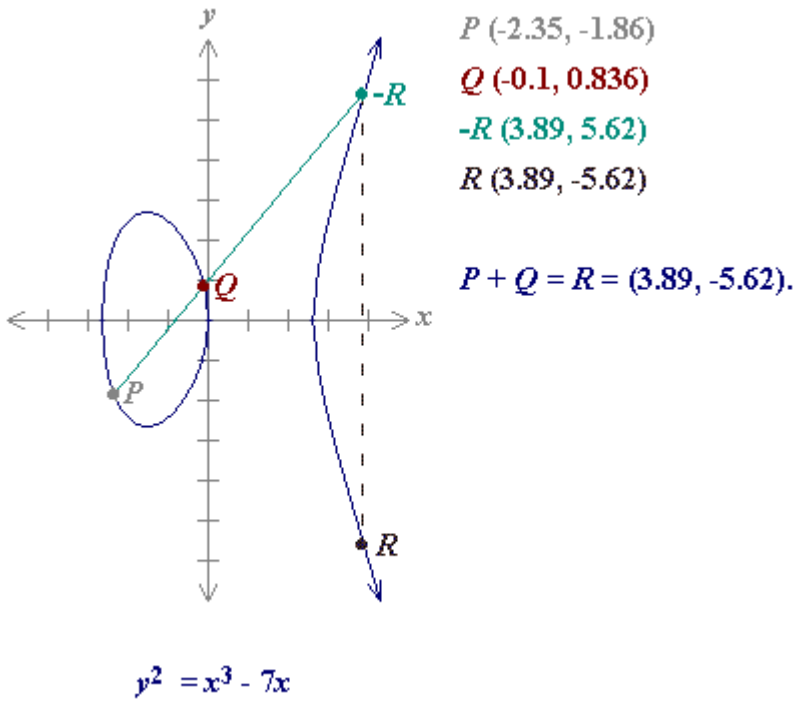


Figura 2. 34 - Exemplo de adição dos pontos P e Q pertencentes a curva elíptica onde P e Q.

4) Se $Q = -P$, então define-se $P+Q= O$ (figura 2.17).

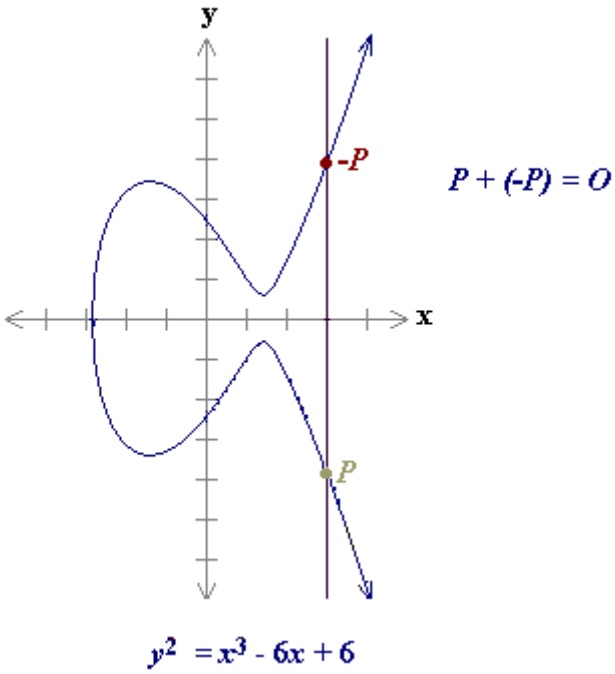


Figura 2. 35 - Exemplo de adição dos pontos P e Q pertencentes a curva elíptica onde $P = -P$.

5) Se $P = Q$, então $P+Q$ este caso será obtido traçando-se uma reta tangente a curva em P, o qual terá multiplicidade dupla e portanto a reta interceptará a curva em apenas mais um ponto, que será rebatido com relação ao eixo x, obtendo-se assim $2P$ (exemplo na figura 2.18). Analiticamente, $P+Q = 2P$ é obtido através das equações (2.28) e (2.27).

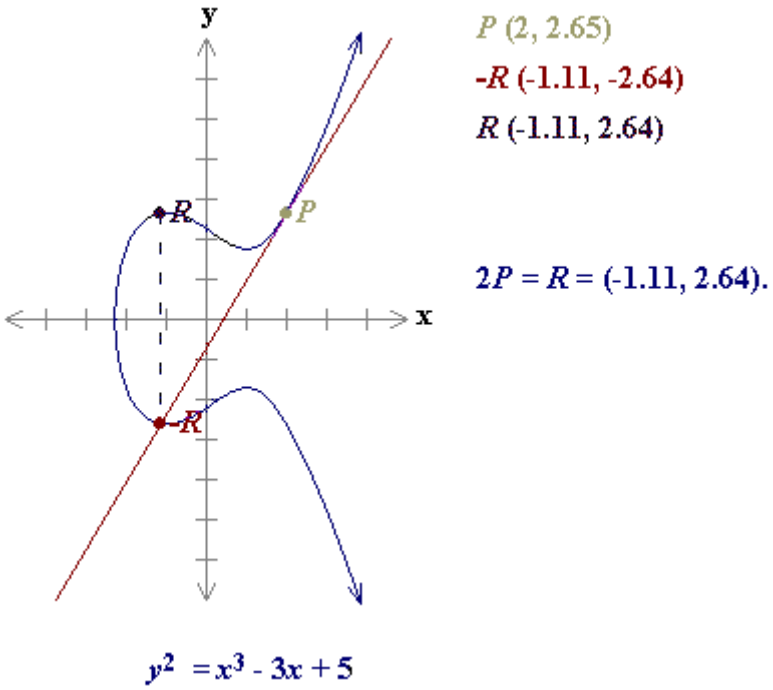


Figura 2. 36 - Exemplo de duplicação de um ponto P pertencente a uma curva elíptica.

Dado que é possível adicionar um ponto P a si mesmo um número arbitrário k de vezes, é natural definir multiplicação escalar como $kP \equiv P + P + P + \ldots + P$ (k termos). Daí defini-se também $0.P \equiv \mathcal{O}$, e para coeficientes negativos $(-k)P \equiv k(-P)$ [B99]. A multiplicação escalar assim definida satisfaz as seguintes propriedades, semelhantes a de um espaço vetorial: $\forall\ P, Q \in E, \forall\ m, n \in Z$,

1. Identidade: $1 \bullet P = P$;
2. Distributividade vetorial: $m \bullet (P + Q) = m \bullet P + m \bullet Q$;
3. Distributividade escalar: $(m + n) \bullet P = m \bullet P + n \bullet P$;
4. Associatividade: $m \bullet (n \bullet P) = (mn) \bullet P$;
5. Comutatividade: $m \bullet (n \bullet P) = n \bullet (m \bullet P)$

Para cada ponto P de um grupo finito G formado pelos pontos da curva elíptica, sempre é possível encontrar um inteiro positivo r tal que $r \bullet P = \mathcal{O}$. A razão é simples: se o número de elementos de G é t, uma seqüência qualquer de $t+1$ ou mais múltiplos de P necessariamente contém algum valor repetido, isto é, dois coeficientes a e b ($a < b$) tais que $a \bullet P = b \bullet P$, implicando $r \bullet P = \mathcal{O}$ para $r = b - a > 0$.

Definição 2.18: O menor inteiro positivo *r* tal que *r*•*P*= $\mathcal{O}$ é chamado a ordem de *P*.

Um dos métodos utilizados para executar a multiplicação de pontos de numa curva elíptica de modo eficiente é o método da duplicação repetida [R98]. Dados o inteiro *k* e o ponto *P* pertencente a curva elíptica, desejando-se calcular o produto $Q = k \cdot P$, o método consiste em, expressar *k* na sua forma binária, digamos $(a_t, a_{t-1}, \ldots, a_1, a_0)$ e seguir o seguinte algoritmo:

```
Begin
     Q:= P;
          From i:= t-1 downto 0 do
               begin
                    Q:=2*Q;
                    If   a_i = 1
                         then Q:=Q+P;
               end;
     Writeln('Q = k · P =', Q);
End.
```

A fim de ilustrar o método considere o exemplo a seguir.

Exemplo 2.10: Deseja-se calcular $Q = k \cdot P$, onde $k = 75$.

Em binário, tem-se 75 = $(1001011)_2$, assim *t*=6 e portanto

$$\begin{array}{ll} a_6 = 1 & a_2 = 0 \\ a_5 = 0 & a_1 = 1 \\ a_4 = 0 & a_0 = 1 \\ a_3 = 1 & \end{array}$$

Seguindo o algoritmo tem-se,

$Q := P$;

Para $i = t\text{-}1 = 5$, $a_5 = 0$, logo $Q := 2P$;

para $i = 4$, $a_4 = 0$ logo $Q := 2(2P)$;

para $i = 3$, $a_3 = 1$ logo $Q := 2(2(2P)) + P$;

para $i = 2$, $a_2 = 0$ logo $Q := 2(2(2(2P)) + P)$;

para $i = 1$, $a_1 = 1$ logo $Q := 2(2(2(2(2P)) + P)) + P$;

para $i = 1$, $a_0 = 1$ logo $Q := 2(2(2(2(2P)) + P)) + P) + P$.

Assim, $Q = 2(2(2(2(2P)) + P)) + P) + P = 75P$, sendo necessárias apenas 6 duplicações e 3 adições, um total de 9 operações ao invés de 75.

2.2.4 - Curvas Elípticas sobre Corpos Finitos

Até este momento foram consideradas curvas elípticas sobre o corpo dos reais, porém deste momento em diante serão estudadas as curvas elípticas sobre Corpos Finitos, uma vez que essas famílias de curvas elípticas são as únicas empregadas em Criptografia, destacando-se a família de curvas elípticas sobre F_{2^m} .

De um modo geral define-se uma curva elíptica da seguinte forma

Definição 2.19 – Seja K um corpo e $a, b, c, d, e \in K$. Uma curva elíptica sobre K, denotada por $E(K)$, é o conjunto de pares $(x, y) \in K \times K$ que satisfazem uma equação da forma

$$y^2 + axy + by = x^3 + cx^2 + dx + e, \qquad (\mathbf{2.29})$$

juntamente com o ponto no infinito, O.

Considerando a curva sobre corpos finitos, $K = F_p$, onde p denota um número primo, procuram-se soluções (x, y) em (2.29) tal que $x, y \in F_p$. Pode-se generalizar procurando-se soluções em F_q, onde F_q é um corpo de extensão de F_p contendo $q = p^m$ elementos. Tal solução (x, y) é um ponto sobre a curva.

Há alguns diferenças importantes a serem consideradas entre as curvas elípticas sobre os reais e as curvas elípticas sobre corpos finitos. As curvas elípticas sobre corpos finitos tem um número finito de pontos, o que é uma característica desejada para propósitos criptográficos. Como essas curvas são representadas por pontos, não fica claro como conectá-los a fim de obter um gráfico em formato de curva, assim não é claro a utilização de relações geométricas como as usadas nas curvas sobre os reais. Porém, as regras algébricas para a aritmética podem ser adaptadas para este caso, com a vantagem que diferentemente das curvas elípticas sobre os reais a aritmética sobre os corpos finitos não possui erros de arredondamento, uma característica requerida nas aplicações criptográficas. Desta forma, assim como nos reais, tendo-se uma curva não-singular, uma regra de adição pode ser definida e os pontos formarão um grupo comutativo.

Os grupos formados pelos pontos de uma curva elíptica são cíclicos. Neste caso, existe um ponto que gera todos os outros pontos pertencentes a curva, chamado *ponto gerador*.

Uma das vantagens de se usar o ponto gerador é a facilidade de se calcular a soma de pontos pertencentes a curva, uma vez que dado um gerador *G* pertencente a uma curva de ordem *n*, a soma dos pontos $P = k_P \cdot G$ e $Q = k_Q \cdot G$ é $P + Q = k_S \cdot G$ onde $k_S = (k_P + k_Q) \bmod n$.

Para aplicações criptográficas, consideram-se curvas que contenham subgrupos cíclicos de ordem prima, a fim de evitar certos ataques. O melhor caso ocorre quando a própria ordem da curva é prima, o que nem sempre é possível.

Maiores detalhes sobre o uso de curvas elípticas em criptografia serão fornecidos no capítulo 4.

Em corpos finitos, nota-se que o uso do termo "curva" é, na verdade, um abuso de linguagem, pois embora realmente o gráfico da equação sobre os reais tenha realmente o aspecto de uma curva, a mesma sobre um corpo finito passa a ser representada por um conjunto de pontos.

O número de pontos de uma curva elíptica $E(F_q)$ (i.e., o número de soluções para a equação da curva mais o ponto no infinito) é denominado ordem da curva, denotada por $\#E(F_q)$. O número de pontos de uma curva elíptica sobre um corpo finito é sempre um valor próximo ao número de elementos desse corpo. Este resultado é conhecido como *Teorema de Hasse* [B99]:

Teorema 2.13 (Teorema de Hasse): A ordem , $\#E(F_q)$, de uma curva elíptica $E(F_q)$ satisfaz as relações:

$$q + 1 - 2\sqrt{q} \leq \#E(F_q) \leq q + 1 + 2\sqrt{q} \qquad \textbf{(2.30)}$$

A seguir são mostradas curvas elípticas sobre corpos de características $p = 2$, $p = 3$ e $p > 3$, obtidas a partir da modificação de (2.29) para a forma normal de Weiertrass, com suas respectivas fórmulas analíticas de adição. As regras para operação de adição aqui são exatamente as mesmas aplicadas às curvas sobre os reais, apenas os cálculos são feitos usando-se aritmética modulo *p*.

A tabela com os discriminantes de cada um dos tipos de curvas é definido na tabela 2.6, pois como foi visto anteriormente a lei de construção realmente define um grupo sobre os pontos da curva, desde que $\frac{df(x)}{dx} \neq 0$, i.e. se o discriminante da curva não for nulo.

Para corpos finitos binários, F_{2^m}, tem-se as seguintes possíveis equações na forma normal de Weierstrass

$$y^2 + cy = x^3 + ax + b \qquad (\mathbf{2.31}\boldsymbol{a})$$

onde $a, b, c \in F_{2^m}$ e $c \neq 0$.

e

$$y^2 + xy = x^3 + ax^2 + b \qquad (\mathbf{2.31}\boldsymbol{b})$$

onde $a, b \in F_{2^m}$ e $b \neq 0$.

Para corpos finitos cuja característica é um primo $p = 3$, a equação na forma normal de Weierstrass é

$$y^2 = x^3 + ax^2 + bx + c \qquad (\mathbf{2.31}\boldsymbol{c})$$

onde $a, b, c \in F_p$, tal que $a^2(b^2 - ac) - b^3 \neq 0 \bmod p$. Em corpos finitos cuja característica é um primo $p > 3$, a equação na forma normal de Weierstrass é

$$y^2 = x^3 + ax + b \qquad (\mathbf{2.31}\boldsymbol{d})$$

onde a e $b \in F_p$, tal que $4a^3 + 27b^2 \neq 0 \bmod p$.

Curva	**Discriminante**
2.32*a*	b
2.32*b*	c^4
2.32*c*	$a^2(b^2 - ac) - b^3$
2.32*d*	$-16(4a^3 + 27b^2)$

Tabela 2. 6 - Discriminantes de curvas elípticas.

A fim de definir as fórmulas analíticas de adição para cada uma das curvas considere os pontos $P = (x_1, y_1)$, $Q = (x_2, y_2)$ e $P + Q = (x_3, y_3)$. Para curvas do tipo (2.31*a*) tem-se:

$$-P = (x_1, y_1 + c); \qquad \mathbf{(2.33}\boldsymbol{a}\mathbf{)}$$

$$x_3 = \begin{cases} \left(\left(\frac{y_1 + y_2}{x_1 + x_2}\right)^2 + x_1 + x_2\right) \bmod p & P \neq Q \\ \left(\frac{x_1^4 + a^2}{c^2}\right) \bmod p & P = Q \end{cases} \quad \textbf{(2.33}b\textbf{)}$$

e

$$y_3 = \begin{cases} \left(\left(\frac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + y_1 + c\right) \bmod p & P \neq Q \\ \left(\left(\frac{x_1^2 + a}{c}\right)(x_1 + x_3) + y_1 + c\right) \bmod p & P = Q \end{cases} \quad \textbf{(2.33}c\textbf{)}$$

Para as do tipo (2.31*b*) tem-se:

$$-P = (x_1, y_1 + x_1); \quad \textbf{(2.34}a\textbf{)}$$

$$x_3 = \begin{cases} \left(\left(\frac{y_1 + y_2}{x_1 + x_2}\right)^2 + \left(\frac{y_1 + y_2}{x_1 + x_2}\right) + x_1 + x_2 + a\right) \bmod p & P \neq Q \\ \left(x_1^2 + \frac{b}{x_1^2}\right) \bmod p & P = Q \end{cases} \quad \textbf{(2.34}b\textbf{)}$$

e

$$y_3 = \begin{cases} \left(\left(\frac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + x_3 + y_1\right) \bmod p & P \neq Q \\ \left(x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right)x_3 + x_3\right) \bmod p & P = Q \end{cases} \quad \textbf{(2.34}c\textbf{)}$$

Para as curvas do tipo (2.31*c*), tem-se

$$-P = (x_1, -y_1); \quad \textbf{(2.35}a\textbf{)}$$

$$x_3 = \begin{cases} \left(\left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2\right) \bmod p & P \neq Q \\ \left(\left(\frac{3x_1^{\,2} + ax + b}{2y_1}\right) - x_1 - x_2\right) \bmod p & P = Q \end{cases} \qquad \textbf{(2.35}b\textbf{)}$$

e

$$y_3 = \begin{cases} \left(\left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1\right) \bmod p & P \neq Q \\ \left(\left(\frac{3x_1^{\,2} + ax + b}{2y_1}\right)(x_1 - x_3) - y_1\right) \bmod p & P = Q, \end{cases} \qquad \textbf{(2.35}c\textbf{)}$$

e, por fim, para as curvas (2.31*d*)

$$-P = (x_1, -y_1); \qquad \textbf{(2.36}a\textbf{)}$$

$$x_3 = \begin{cases} \left(\left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2\right) \bmod p & P \neq Q \\ \left(\left(\frac{3x_1^{\,2} + a}{2y_1}\right) - x_1 - x_2\right) \bmod p & P = Q \end{cases} \qquad \textbf{(2.36}b\textbf{)}$$

e

$$y_3 = \begin{cases} \left(\left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1\right) \bmod p & P \neq Q \\ \left(\left(\frac{3x_1^{\,2} + a}{2y_1}\right)(x_1 - x_3) - y_1\right) \bmod p & P = Q \end{cases} \qquad \textbf{(2.36}c\textbf{)}$$

Será visto no capítulo 4 que dentre as equações sobre corpos de características 2 só a (2.31*b*) tem utilidade criptográfica uma vez que a curva com equação (2.31*a*) é supersingular, sendo mais vulnerável criptoanaliticamente [MOV93].

2.2.5 - Exemplos

Nesta seção alguns exemplos envolvendo operações aritméticas em curvas elípticas sobre corpos finitos serão mostrados.

Exemplo 2.11: Seja *p*=23 e a curva elíptica $E: y^2 = x^3 + x + 4$ definida sobre F_{23} (Em (2.31*d*), *a*=1 e *b*=4). Nota-se que o discriminante é não nulo uma vez que $4a^3 + 27b^2 = 4 + 432 = 436 \equiv 22 (\mathrm{mod}\, 23)$, então *E* é de fato uma curva elíptica. Os pontos pertencentes a esta curva juntamente com o ponto O são:

(0,2)	(0,21)	(1,11)	(1,12)	(4,7)	(4,16)	(7,3)
(7,20)	(8,8)	(8,15)	(9,11)	(9,12)	(10,5)	(10,18)
(11,9)	(11,14)	(13,11)	(13,12)	(14,5)	(14,18)	(15,6)
(15,17)	(17,9)	(17,14)	(18,9)	(18,14)	(22,5)	(22,19)

□

Exemplo 2.12: Considere a curva elíptica do exemplo anterior e as equações para adição de pontos (2.36*b*) e (2.36*c*), quando $P \neq Q$. Seja $P = (4,7)$ e $Q = (13,11)$, então $P + Q = (x_3, y_3)$ é dado por:

$$x_3 = \left(\frac{11-7}{13-4}\right)^2 - 4 - 13 = 3^2 - 4 - 13 = -8 \equiv 15 (\mathrm{mod}\, 23)$$

e $y_3 = 3(4-15) - 7 = -40 \equiv 6 (\mathrm{mod}\, 23)$. Assim, $P + Q = (15,6)$.

A duplicação do ponto $P = (4,7)$, $2P = P + P = (x_3, y_3)$, obtida através das equações (2.36*b*) e (2.36*c*) quando $P = Q$, é dada por:

$$x_3 = \left(\frac{3(4)^2 + 1}{14}\right)^2 - 8 = 15^2 - 8 = 217 \equiv 10 (\mathrm{mod}\, 23)$$

e $y_3 = 15(4-10) - 7 = -97 \equiv 18 (\mathrm{mod}\, 23)$. Assim, $2P = (10,18)$.

□

Exemplo 2.13: Considere F_{2^4} representado pelo polinômio irredutível $f(x) = x^4 + x + 1$ (exemplo 2.4, seção 2.1.5). Considere também a curva elíptica $E: y^2 + xy = x^3 + \alpha^4 x^2 + 1$ sobre F_{2^4} (em (2.31*b*), $a = \alpha^4$ e $b = 1$). Nota-se que o discriminante é não nulo, i.e. $b \neq 0$, então $E(F_{2^4})$ é de fato uma curva elíptica. Os pontos pertencentes a esta curva juntamente com o ponto O são:

(0,1)	(1,α^6)	(1,α^{13})	(α^3,α^8)	(α^3,α^{13})
(α^5,α^3)	(α^5,α^{11})	(α^6,α^8)	(α^6,α^{14})	(α^9,α^{10})
(α^9,α^{13})	(α^{10},α)	(α^{10},α^8)	(α^{12},0)	(α^{12},α^{12})

□

Exemplo 2.14: Considere a curva elíptica do exemplo anterior e as equações para adição de pontos (2.34*b*) e (2.34*c*), quando $P \neq Q$. Seja $P = (\alpha^6, \alpha^8)$ e $Q = (\alpha^3, \alpha^{13})$, então $P + Q = (x_3, y_3)$ é dado por:

$$x_3 = \left(\frac{\alpha^8 + \alpha^{13}}{\alpha^6 + \alpha^3}\right)^2 + \frac{\alpha^8 + \alpha^{13}}{\alpha^6 + \alpha^3} + \alpha^6 + \alpha^3 + \alpha^4 = \left(\frac{\alpha^3}{\alpha^2}\right)^2 + \frac{\alpha^3}{\alpha^2} + \alpha^6 + \alpha^3 + \alpha^4 = 1$$

e $y_3 = \left(\frac{\alpha^8 + \alpha^{13}}{\alpha^6 + \alpha^3}\right)(\alpha^6 + 1) + 1 + \alpha^8 = \left(\frac{\alpha^3}{\alpha^2}\right)(\alpha^{13}) + \alpha^2 = \alpha^{13}$. Assim, $P + Q = (1, \alpha^{13})$.

A duplicação do ponto $P = (\alpha^6, \alpha^8)$, $2P = P + P = (x_3, y_3)$, obtida através das equações (2.34*b*) e (2.34*c*) quando $P = Q$, é dada por:

$$x_3 = (\alpha^6)^2 + \frac{1}{(\alpha^6)^2} = \alpha^{12} + \alpha^3 = \alpha^{10}$$

e $y_3 = (\alpha^6)^2 + \left(\alpha^6 + \frac{\alpha^8}{\alpha^6}\right)\alpha^{10} + \alpha^{10} = \alpha^{12} + \alpha^{13} + \alpha^{10} = \alpha^8$. Assim, $2P = (\alpha^{10}, \alpha^8)$.

□

Exemplo 2.15: Considere a curva elíptica $E(F_{23})$ do exemplo 2.11. Como $\#E(F_{23}) = 29$, i.e., um número primo, então $E(F_{23})$ é cíclica e qualquer ponto diferente de O é gerador de $E(F_{23})$. Considerando-se, por exemplo o ponto $P = (0,2)$, obtém-se:

1P=(0,2)	2P=(13,12)	3P=(11,9)	4P=(1,12)	5P=(7,20)
6P=(9,11)	7P=(15,6)	8P=(14,5)	9P=(4,7)	10P=(22,5)
11P=(10,5)	12P=(17,9)	13P=(8,15)	14P=(18,9)	15P=(18,14)
16P=(8,8)	17P=(17,14)	18P=(10,18)	19P=(22,18)	20P=(4,16)
21P=(14,18)	22P=(15,17)	23P=(9,12)	24P=(7,3)	25P=(1,11)
26P=(11,14)	27P=(13,11)	28P=(0,21)	29P= O	

Referências:

[B98] – D. M. Burton, *Elementary Number Theory*, 4ª Edição, Allyn and Bacon, Inc., 1998.

[B99] – P. S. L. M. Barreto, *Curvas Elípticas e Criptografia: Conceitos e Algoritmos*, Junho 1999.

[D92] – J. R. Durbin, *Modern Algebra – An Introduction*, 3ª Edição, John Wisley & Sons, 1992.

[K84] – N. Koblitz, *Introduction to Elliptic Curves and Modular Forms*, Springer- Verlag, 1984.

[K87] – N. Koblitz, *Elliptic Curve Cryptosystems*, Mathematics of Computation, 48: pp. 203-209, 1987.

[K94]- N. Koblitz, *A Course in Number Theory and Cryptography*, 2ª Edição., Springer-Verlag, 1994.

[M86] – V. S. Miller, *Use of Elliptic Curves in Cryptography*, In Advances in Cryptology Crypto *?5*, pp. 417-426, Springer-Verlag, 1986.

[M87] – R. J. McEliece, *Finite Fields for Computer Scientists and Engineers*, Kluwer Academic Publishers, 1987.

[MOV93] – A. Menezes, T. Okamoto e S. Vanstone, *Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field*, IEEE Transactions in Information Theory, vol. IT-39, pp. 1639- 1646, 1993.

[MOV96] – A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[MS86] – F. J. Macwilliams e N.J.A. Sloane, *The Theory of Error Correcting Codes*, North Holland Library, 1986.

[ST92] – J. H. Silverman e J. Tate, *Rational Points on Elliptic Curves*, Springer-Verlag, 1992.

Páginas na Web:

[www2.1] - http://www.pbs.org/wgbh/nova/proof/

3

CRIPTOGRAFIA

Neste capítulo serão introduzidos alguns aspectos básicos sobre criptografia, sendo dada uma maior ênfase aos cripto-sistemas assimétricos. Na seção 3.3 o Problema do Logaritmo Discreto, no qual alguns dos cripto-sistemas assimétricos atuais são baseados, é explicado. O algoritmo *hash* MD5 é discutido na seção 3.4. A última seção trata do cripto-sistema Rijndael (pronuncia-se "rainandol"), padrão atual para cifragem de dados do governo americano, que substitui o DES (Data Encryption Standard) a partir de Janeiro de 2001. Ambos os sistemas, MD5 e Rijndael, fazem parte do padrão proposto para segurança em redes móveis.

3.1 – Introdução

A arte e ciência que pode manter mensagens seguras é a *criptografia*, a qual é praticada por *criptógrafos*. A *criptoanálise*, arte e ciência de recuperar a mensagem sem a necessidade do conhecimento da chave[5], é praticada pelos criptoanalistas. O ramo da matemática que engloba tanto a criptografia quanto a criptoanálise é chamado *criptologia* e é praticado por *criptologistas* [S96].

A criptografia é uma das ferramentas utilizadas para a proteção dos dados, proporcionando privacidade e integridade aos mesmos, evitando que tais informações sejam reveladas, alteradas, substituídas ou destruídas por pessoas não autorizadas. Diferentemente de outras ferramentas utilizadas para a segurança de dados, os cripto-sistemas são aqueles que se mostram mais completos até o momento, proporcionando alto nível de segurança com mais flexibilidade [CW97].

Os *cripto-sistemas* usam transformações a fim de evitar a alteração dos dados (Integridade) tanto quanto tornar os dados não inteligíveis (Privacidade) para pessoas não autorizadas.

[5] Uma chave pode ser definida como um elemento de informação utilizado na troca de mensagens de forma sigilosa. Num cripto-sistema simétrico uma única chave é usada por cada um dos usuários e a mesma é mantida secreta para os que não estão envolvidos na troca de mensagem. Já num cripto-sistema assimétrico duas chaves são utilizadas por cada usuário, uma secreta e uma pública.

Na construção de um cripto-sistema as características desejadas são levadas em conta, para que em sua construção as mesmas estejam presentes. A fim de proporcionar um maior entendimento e visando facilitar o projeto de um cripto-sistema faz-se uma subdivisão do conceito de integridade, pois como será observado nas próximas seções, para cada característica um tipo de cripto-sistema é utilizado.

As subdivisões em questão são:

- *Autenticação do usuário*: Assegura que as partes envolvidas numa comunicação são quem elas realmente dizem ser.
- *Autenticação da origem dos dados*: Assegura a fonte dos dados.
- *Integridade dos dados*: Assegura que os dados não foram alterados por um usuário não autorizado.
- *Não repudiação*: Torna possível a não repudiação de uma transação, i. e., aquele que receber a assinatura poderá usá-la para provar para uma terceira parte neutra que a assinatura foi de fato gerada pelo assinante, o qual não pode repudiar a assinatura.

A criptografia é conhecida classicamente como uma arte muito antiga que utilizava procedimentos visando tornar certas informações não legíveis para pessoas não autorizadas que as interceptassem.

A palavra criptografia é de origem grega significando "Escrita Escondida" (Kriptos (escondido) + graphos (escrita)). Seu primeiro registro é de 400 A.C. na utilização pelos espartanos de um mecanismo conhecido como Cítala Espartana [B86]. O processo de *cifragem*[6] consistia em enrolar uma tira de couro ou papiro num tubo e escrever a mensagem no sentido vertical; ao se desenrolar a tira a mensagem parecia não ter nenhum sentido. Para se obter a mensagem original, ou seja *decifrar*[7] a mensagem cifrada recebida, bastava enrolar novamente a tira de couro ou papiro num tubo com as mesmas dimensões do tubo usado no processo de cifragem. Neste caso, observa-se que a informação secreta compartilhada entre o remetente e receptor, que possibilitava a troca de mensagens, dificultando o acesso de uma pessoa não autorizada à mesma, corresponde às dimensões do tubo, o que pode se chamar de *chave secreta* do *cripto-sistema*.

[6] Processo pelo qual através de alguma informação secreta transforma-se o texto claro (dados originais, legíveis) em texto cifrado (dados não inteligíveis, sem sentido).
[7] Processo oposto ao processo de cifragem, permitindo a obtenção do texto claro a partir do texto cifrado.

Outro cripto-sistema clássico muito conhecido é o cripto-sistema de César. Este cripto-sistema foi criado e utilizado pelo imperador romano Júlio César em suas conquistas. O cripto-sistema consistia num deslocamento cíclico de três letras do alfabeto, i. e., a letra *a* era substituída pela letra *d*, a letra *b* era substituída pela letra *e* e assim por diante. Conhecido o deslocamento usado no processo de cifragem a decifragem é feita facilmente, olhando as substituições de forma invertida, i. e., substituindo-se *d* por *a*, *e* por *b* e assim por diante.

Inicialmente, a criptografia era usada apenas para fins militares e diplomáticos. Porém, devido ao grande desenvolvimento nos meios de comunicação, o que era um privilégio de militares e diplomatas passou a se disseminar para outras áreas. Durante a Segunda Guerra Mundial houve um grande desenvolvimento na área, novas técnicas foram criadas e máquinas foram usadas no processo de cifragem e decifragem [S87]. Atualmente um cidadão comum tem a possibilidade de utilizar tais técnicas para se comunicar e armazenar seus dados de forma segura.

Existem basicamente dois tipos de criptografia, o clássico, que surgiu desde a origem da escrita conhecido com criptografia de chave secreta ou criptografia simétrica, assim chamada por ter apenas uma chave compartilhada pelos usuários que desejam se comunicar de forma segura, e a criptografia de chave pública ou assimétrica, assim chamada devido a existência de duas chaves, surgida na década de 70 e revolucionando a área.

3.2 – Tipos de Criptografia

3.2.1 – Criptografia de Chave Secreta

Nota-se que classicamente os cripto-sistemas compartilhavam apenas uma informação que tornava possível tanto a cifragem quanto a decifragem e que esta informação deveria ser mantida secreta uma vez que a segurança do cripto-sistema dependia da mesma.

Essa informação compartilhada entre o remetente e o receptor do texto cifrado (criptograma) é chamada *chave secreta* e, segundo o Princípio de Kerckhoff[8] é nela que deve residir toda a segurança do cripto-sistema [S96].

[8] Princípio de Kerckhoff : O criptoanalista sabe todos os detalhes do processo de cifragem e decifragem com exceção do valor da chave secreta. Portanto é nela que reside a segurança do cripto-sistema.

Este tipo de cripto-sistema também é conhecido como cripto-sistema simétrico devido a sua simetria, uma vez que a mesma chave é usada tanto no processo de cifragem quanto no de decifragem.

Uma desvantagem desta classe de cripto-sistemas esta relacionada ao gerenciamento de chaves. Como em um tal cripto-sistema a chave deve permanecer secreta, sua distribuição deve ser feita de modo seguro, o que pode provocar um alto custo. Além disso, seu armazenamento e distribuição torna-se bastante problemático em grandes redes, uma vez que cada usuário do sistema deve possuir uma chave distinta para se comunicar com cada um dos outros usuários, ou seja, numa rede com *n* usuários será necessário gerar $\frac{n(n-1)}{2}$ chaves. Assim, por exemplo, num sistema com 1000 usuários se faz necessário 499.500 chaves, que devem ser trocadas e mantidas seguras.

Foi pensando neste problema que em 1976, dois pesquisadores da Universidade de Stanford, Whitfield Diffie e Martin E. Hellman, publicaram um trabalho [DH76] no qual introduziam uma idéia inovadora no campo da criptografia, a *criptografia de chave pública*.

3.2.2 – Criptografia de Chave Pública

A criptografia de chave pública introduzida na década de 70 veio não só resolver o problema da distribuição de chaves como também da autenticidade, permitindo a utilização de um processo equivalente a assinatura escrita, conhecido como *assinatura digital*, promovendo a partir do mesmo a integridade, não repudiação e autenticação da origem dos dados.

Esta classe de cripto-sistemas é caracterizada pela existência de duas chaves para cada usuário, sendo uma pública (*E*) e outra secreta (*D*). Desta forma, tal cripto-sistema também chamado de *cripto-sistema assimétrico*. Cada uma das chaves é utilizada em um dos processos sem que a chave *D* possa ser obtida a partir da chave *E*, não havendo assim a necessidade de uma troca de chave como no cripto-sistema simétrico.

Para se obter privacidade utilizando um cripto-sistema de chave pública, a chave *E* é utilizada no processo de cifragem e a chave *D* no decifragem. Para um melhor entendimento considere que o usuário **A** deseja enviar uma mensagem para o usuário **B**, sem que a mesma possa ser lida por um outro usuário em caso de interceptação. O processo de troca de mensagem com privacidade consiste em:

1) O usuário **A** obtém a chave pública de **B**, E_B, (por exemplo, em um catálogo público) e cifra a mensagem M com a mesma, obtendo $C = E_B(M)$. (*processo de cifragem*)

2) O usuário **B** ao receber C utiliza sua chave secreta e decifra C, i. e., calcula $D_B(C)=D_B(E_B(M))=M$. (*processo de decifragem*)

Nota-se claramente aí que, diferentemente de um cripto-sistema de chave secreta, não houve necessidade da troca de uma chave através de um canal seguro, e que além disso reduz-se bastante o número de chaves a serem gerenciadas, uma vez que cada usuário deve possuir apenas duas chaves, e portanto naquele sistema com 1000 usuários só 2000 chaves seriam gerenciadas.

Devido ao grande volume de dados compartilhados a distância e a todas as possibilidades existentes descritas na seção 3.1, tornou-se necessário uma maneira de proporcionar a não repudiação, integridade e autenticação da origem dos dados. Deste modo vislumbrou-se a possibilidade de se obter um equivalente no meio eletrônico à assinatura escrita. Porém, como a informação no formato eletrônico é facilmente copiada, a assinatura digital não pode ser formada da mesma maneira que a escrita, ou seja, associando a cada usuário um padrão a ser inserido na mensagem.

A fim de evitar que a assinatura seja forjada é usado um cripto-sistema de chave pública sobre o dado a ser assinado digitalmente.

A principal diferença entre usar um cripto-sistema assimétrico para obter privacidade e construir uma assinatura digital reside na ordem em que são utilizadas as chaves E e D. Desta vez um usuário assina a mensagem usando sua chave secreta D e um outro usuário verifica a assinatura usando a chave publica E do usuário que assinou a mensagem.

Considere, a fim de ilustrar como o processo ocorre, que o usuário **A** deseja autenticar a mensagem M. Então

1) O usuário **A** inicialmente transforma M usando uma *função hash*; a saída denotada por $H(M)$ é chamada sumário de mensagem e é função da mensagem M, servindo desta forma como uma impressão digital de M.

2) **A** então assina M utilizando D_A, isto é, **A** calcula $S=D_A(H(M))$ e envia para o usuário **B** a mensagem M juntamente com a assinatura S.

3) Para o usuário **B** verificar a assinatura inicialmente ele computa $H(M)$.

4) Então **B** computa $E_A(S)$ e compara com o resultado anterior.

   Se $E_A(S) = H(M)$, então a assinatura é valida.

   Se $E_A(S) \neq H(M)$, então a assinatura não é valida.

A origem dos dados é comprovada uma vez que apenas o usuário **A** conhece D_A e portanto apenas ele poderia transformar $H(M)$ em S; logo, é proporcionada a *autenticação da origem dos dados*.

Uma vez que o usuário **A** assina a mensagem *M*, esta não pode ser alterada uma vez que alterando *M*, *H*(*M*) será alterado e consequentemente *S* também sofrerá alteração. Portanto, a *integridade* de *M* é comprovada.

Ao assinar *M*, **A** não poderá repudiar este ato, pois ao se verificar a assinatura através de *M* e *S*, está demonstrado que **A** gerou a assinatura.

A assinatura digital é criada a partir do sumário da mensagem ao invés da própria mensagem, a fim de haver uma redução de tempo na operação, devido ao menor comprimento do sumário com relação a mensagem original. Pode-se pensar no sumário como uma impressão digital da mensagem original. Porém a utilização da função *hash* traz consigo uma pequena insegurança. Tal insegurança diz respeito ao fato que diferentes entradas podem produzir saídas iguais. Quando duas mensagens diferentes resultam num mesmo sumário diz-se que ocorreu uma *colisão*.

Uma característica de uma função *hash* é a de transformar uma entrada de comprimento arbitrário numa saída de comprimento fixo. Para uso em criptografia outras características adicionais são necessárias. Os requerimentos básicos para uma função *hash* criptográfica são:

- Entrada de qualquer comprimento;
- Saída de comprimento fixo;
- $H(x)$ é relativamente fácil de calcular dado qualquer x;
- $H(x)$ é unidirecional[9];
- $H(x)$ é livre de colisão[10].

Exemplos de funções *hash* bastante conhecidas são as MD2 [K92], MD5 [R92a] e SHA [NIST00][11].

Suponha agora um processo de comunicação em tempo real entre **A** e **B**, que **A** deseja se assegurar da identidade de **B**, i. e., deseja fazer a *autenticação do usuário* **B**. Os procedimentos usados anteriormente para se obter autenticação não podem ser usados agora, pois se o usuário autenticar a mensagem "Aqui fala **B**", a mesma poderá ser usada num outro momento por um outro usuário que a intercepte, e que desta forma poderia se passar por **B**. Assim, uma simples assinatura digital não resolve o problema de autenticação do usuário. Neste caso é inserido uma espécie de desafio que se modifica a cada vez que os usuários iniciam uma comunicação. A autenticação do usuário utiliza o seguinte processo:

1) A gera um número aleatório R_A e envia para **B**. R_A é chamado desafio.
2) Ao invés de assinar uma mensagem dizendo "Aqui fala **B**", **B** assina a mensagem "Aqui fala **B**, e você acabou de me enviar R_A"

[9] Diz-se que uma função é unidirecional se a mesma é dificil de ser invertida. Entenda-se por "difícil de ser invertida" se dado o resultado $f = F(x)$ é computacionalmente inviável encontrar x dado f.
[10] Uma função *hash* H é livre de colisão se é computacionalmente inviável achar duas mensagens diferentes x e y tais que $H(x) = H(y)$.
[11] MD = *Message Digest*, SHA = *Secure Hash Algorithm*.

3) **A** verifica a assinatura da mensagem; sendo esta válida, o valor R_A será o mesmo enviado por **A** que, portanto, tem certeza de estar se comunicando com **B** em tempo real.

Assim, combinando o processo de assinatura digital com um desafio aleatório por parte de **A,** foi possível autenticar o usuário **B**.

Todos os cripto-sistemas de chave pública práticos são baseados em funções unidirecionais com *trapdoor*, que são funções unidirecionais cujo inverso é encontrado facilmente por aqueles que possuem uma certa informação secreta (*trapdoor*). Neste tipo de função unidirecional a chave pública dá uma informação geral sobre a função, a qual pode ser de conhecimento público, enquanto que a chave secreta é o *trapdoor*. Assim, quem possui a chave secreta pode calcular a função em ambas as direções (direta e inversa) facilmente, enquanto os possuidores apenas da chave pública só terão facilidade no calculo na direção direta. Logo, a direção direta é usada para cifrar e verificar a assinatura digital e a inversa para decifrar e gerar a assinatura digital.

No decorrer dos anos, alguns cripto-sistemas de chave pública foram quebrados, e outros foram provados não práticos. Atualmente, apenas três tipos de cripto-sistemas assimétricos podem ser considerados seguros e eficientes. Esses cripto-sistemas se baseiam nos seguintes problemas:

a) Fatoração de Inteiros : A complexidade computacional de se multiplicar dois primos grandes p e q é bem menor comparada com a complexidade de se fatorar n nesses dois primos. O problema da multiplicação dos números p e q é um problema cuja complexidade é polinomial. Usando a transformada rápida de Fourier por exemplo, pode-se projetar um algoritmo para obter o produto dos dois inteiros p e q em um tempo linear como função do tamanho l do problema (neste caso l é o número de dígitos decimais necessários para escrever p e q) [SP89]. Para fatorar n nos primos p e q não há algoritmo com complexidade polinomial. Os algoritmos para fatoração são divididos em dois tipos: os de propósito especial e os de propósito geral [RSA00]. Os de *propósito especial* exploram alguma característica particular do número a ser fatorado, dentre eles o mais eficiente é um método baseado em curvas elípticas [L87], tendo complexidade $O(e^{\sqrt{(2 \ln p \ln \ln p)}})$. Já os de propósito geral funcionam com qualquer número, sem precisar explorar uma característica específica como ocorre nos de propósito especial, possuindo porém menor eficiência que estes. Dentre os algoritmos do propósito geral, o mais eficiente é o crivo numérico (*Number Field Sieve* - NFS) [BLP94], [BLZ94], tendo complexidade $O(e^{1.9223(\ln n^{1/3})(\ln \ln n^{2/3})})$. O principal exemplo de cripto-sistema que se baseia neste problema é o RSA criado em 77 por Rivest, Shamir e Adleman [RSA78].

b) Problema do Logaritmo Discreto sobre Corpos Finitos (PLD): Esse problema será abordado com mais atenção na próxima seção. Um exemplo de cripto-sistema assimétrico que utiliza como base este problema é o cripto-sistema de ElGamal [E85a]. Uma outra aplicação no contexto de criptografia que se baseia no PLD é o protocolo para troca de chave introduzido por Diffie-Hellman em 1976, no mesmo trabalho que também introduziu a idéia de criptografia de chave pública [DH76].

c) Problema do Logaritmo Discreto sobre Curvas Elípticas (PLDCE): Este problema será analisado cuidadosamente no próximo capítulo, que tratará sobre cripto-sistemas baseados no mesmo. Um dos exemplos de cripto-sistema baseados neste problema é o EC-DSA [JM99].

3.3 – Problema do Logaritmo Discreto

O problema do logaritmo discreto sobre um grupo finito $\langle G,* \rangle$ de ordem t consiste em, dado

$$y = \alpha * \alpha * \alpha * \ldots * \alpha \qquad \textbf{(3.1)}$$

com $\alpha \in G$, encontrar a quantidade x de vezes que α é operado consigo mesmo, $1 \le x \le t-1$, para produzir y. Dados x e α, y é encontrado facilmente. Porém, dados α e y, encontrar x é uma tarefa árdua. Por essa razão o PLD pode ser considerada uma função unidirecional.

Em criptografia, trabalhou-se inicialmente o PLD sobre corpos finitos. Neste caso, o grupo multiplicativo e o elemento primitivo α de um corpo finito são considerados, a fim de aumentar a segurança proporcionada por este problema matemático aos cripto-sistemas assimétricos.

A partir deste momento, será analisado o PLD sobre corpos finitos, mostrando-se alguns aspectos ligados à criptografia.

Considere a expressão da exponencial modular $y \equiv \alpha^x \bmod p$ onde x e y são inteiros com $1 \le x, y \le p-1$, e α é um elemento primitivo de $GF(p)$. Considere inicialmente o caso em que são dados x, α e p e deseja-se calcular y. Usando o método de quadrado sucessivo [K98], tal exponenciação é feita

de maneira rápida tendo que se executar, no máximo, $2\log_2 p$ multiplicações. Por outro lado, se forem fornecidos os valores de *y*, α e *p* a fim de se obter *x* na equação (3.2), o algoritmo mais efetivo para resolução de tal problema (PLD) possui complexidade exponencial [P78].

Atualmente os melhores algoritmos para resolver o PLD são classificados de duas formas: métodos do índice (*index-calculus methods*) e métodos das colisões (*collision search methods*). A diferença básica entre os dois métodos é com relação a sua utilização. O *método do índice* só é bem sucedido se certas propriedades algébricas estiverem presentes, enquanto que o *método das colisões* pode ser usado de forma mais geral. Apesar de ser mais geral, o *método das colisões* é mais lento; o algoritmo mais eficiente desta classe possui complexidade puramente exponencial. Já na classe dos *métodos do índice*, os melhores algoritmos possuem complexidade subexponencial [RSA00].

O *método do índice* é similar aos métodos atuais mais rápidos de fatoração (e.g., crivo quadrático, crivo numérico). Como exemplos de algoritmos eficientes nesta classe podemos citar o Pohlig-Hellman [PH78], que é bem sucedido se os fatores de *p*-1 são primos pequenos, o Taher ElGamal [E85b] baseado no crivo quadrático e o de Odlyzko [LO91] baseado no crivo numérico. O maior PLD resolvido até o momento [RSA00] foi sobre GF (2^{503}).

O melhor algoritmo na classe dos *métodos das colisões* é o algoritmo de Pollard-rho. Usando este método conseguiu-se resolver o PLD para $p \sim 2^{97}$ (*p* é a ordem do grupo).

Desta forma nota-se que a exponenciação discreta pode ser considerada uma função unidirecional, já que é facilmente calculada em uma direção (exponenciação discreta) e dificilmente calculada na outra (logaritmo discreto).

Definindo formalmente o logaritmo discreto, tem-se

Definição 3.1: Dados um elemento primitivo $\alpha \in GF(p)$ e qualquer $y \in GF^*(p) = GF(p) - \{0\}$, o logaritmo discreto de *y* na base α é o inteiro *x*, $0 \le x \le p-1$, para o qual

$$y = \alpha^x \bmod p \qquad \textbf{(3.2)}$$

ou seja,

$$x = \log_\alpha y \text{ em } GF(p)\,. \qquad \textbf{(3.3)}$$

Uma das primeiras aplicações envolvendo a exponenciação discreta como uma função unidirecional foi na segurança de senhas de sistemas multiusuário, sugerida por John Gill [W88]. Tal algoritmo consistia no armazenamento em um arquivo do par $(i, f(p(i)))$ onde *i* denotava o login do usuário e $f(p(i))$ a imagem da senha do usuário cujo login era *i*. Desta forma, contrário ao método antigo que necessitava da proteção do arquivo no qual estavam armazenadas as senhas, no novo método o arquivo poderia ser público. Nota-se que no método anterior, se o arquivo ficasse disponível a algum intruso o mesmo poderia facilmente se passar por um usuário legítimo do sistema, o que depois da utilização da função unidirecional, tornou-se computacionalmente inviável devido a dificuldade de resolver, no caso proposto, o PLD.

Além desta conhecida aplicação em autenticação, temos também aplicações em outras áreas da criptografia, como por exemplo, o protocolo para troca de chaves de Diffie-Hellman [DH76], o algoritmo de Massey-Omura [WW84] que foi baseado numa idéia de Shamir [K81] e o de T. ElGamal para transmissão de informação e ainda o algoritmo de ElGamal [E85a] para assinatura digital que foi tomado como base para o DSA (Digital Signature Algorithm) [NIST00], sendo usado atualmente como padrão.

Sente-se agora claramente, devido a ampla gama de aplicações em criptografia, o quão importante se tornou o estudo do PLD, já que ao resolver tal problema todos esses instrumentos de segurança se tornam vulneráveis.

3.3.1 – Aplicações do Problema do Logaritmo Discreto

Nesta seção o PLD será ilustrado através de quatro aplicações em criptografia: o protocolo para troca de chave de Diffie-Hellman, o cripto-sistema e o sistema de assinatura de T. ElGamal e o DSA (Digital Signature Algorithm), que faz parte do padrão de assinatura digital DSS (Digital Signature Standard) definido pelo NIST (National Institute of Standards and Technology), que é uma divisão do departamento do comércio americano.

a) Protocolo para Troca de Chave de Diffie-Hellman

O protocolo para troca de chave de Diffie-Hellman foi proposto em 1976 [DH76].Este algoritmo, que tem sua segurança baseada no PLD, é descrito a seguir.

Inicialmente cada usuário do sistema gera um número aleatório independente, X*i*, escolhido uniformemente no conjunto de inteiros {1,2, ..., *p*-1}, onde *p* é um número primo grande (com pelo menos 100 digitos decimais). Escolhido X_i, o qual é mantido secreto, cada um dos usuários calculará $Y_i = \alpha^{X_i} \bmod p$, deixando tal valor a disposição num arquivo público. Suponha agora que dois usuários, Alice e Bob, desejam compartilhar uma chave. Inicialmente, Alice escolhe aleatoriamente X_A e Bob X_B, ambos mantidos secretos. Com X_A, Alice calcula $Y_A = \alpha^{X_A} \bmod p$ que é colocado em um arquivo público ou enviado a Bob por uma canal que pode ser inseguro. Bob por sua vez utiliza X_B para calcular $Y_B = \alpha^{X_B} \bmod p$ que também é colocado em um arquivo público ou mandado para Alice. Ao receber Y_A de Alice, Bob calcula $K_{AB} = (Y_A)^{X_B} \bmod p = (\alpha^{X_A})^{X_B} \bmod p = \alpha^{X_A X_B} \bmod p$; Alice por sua vez, ao receber Y_B de Bob calcula $K'_{AB} = (Y_B)^{X_A} \bmod p = (\alpha^{X_B})^{X_A} \bmod p = \alpha^{X_A X_B} \bmod p$. Nota-se que $K_{AB} = K'_{AB}$, portanto $K_{AB} = \alpha^{X_A X_B} \bmod p$ é a chave compartilhada por Alice e Bob.

Se um outro usuário desejar calcular K_{AB} através de Y_A e Y_B ele deverá resolver o seguinte problema

$$K_{AB} = Y_A^{\log_\alpha^{Y_B}} \bmod q \qquad \textbf{(3.4)}$$

ou seja, deverá resolver o PLD.

b) Cripto-sistema de Chave Pública e Esquema de Assinatura de Taher ElGamal

Em 1985, T. ElGamal publicou um artigo [E85a] que mostrou duas aplicações do PLD; um cripto-sistema de chave pública e um algoritmo para assinatura digital.

i) O Cripto-sistema de Chave Pública de ElGamal

Suponha desta vez que Alice deseja mandar uma mensagem *m* para Bob, tal que $0 \le m \le p-1$, onde *p* é um número primo grande escolhido de modo que *p*-1 tenha pelo menos um primo grande como fator (mais de 150 dígitos decimais) [PH78]. Inicialmente Alice escolhe uniformemente um número *k*, $0 \le k \le p-1$, que é mantido secreto e calcula

$$K = y_B^k \bmod p \qquad \textbf{(3.5)}$$

onde y_B é público. A mensagem cifrada será dada pelo par (c_1,c_2), tal que

$$c_1 = \alpha^k \bmod p\,, \qquad \textbf{(3.6a)}$$

$$c_2 = Km \bmod p\,. \qquad \textbf{(3.6b)}$$

Nota-se que o protocolo usado para o envio da chave *K* foi o protocolo para troca de chave de Diffie-Helman, já explicado na seção anterior. Outro fato importante a ser notado é que o texto cifrado (c_1,c_2) tem o dobro do comprimento do texto claro (*m*).

Recebido o texto cifrado (c_1,c_2), Bob usa o processo de decifragem a fim de obter a mensagem *m* de Alice. O processo consiste inicialmente no calculo de *K*, através de c_1, ou seja,

$$c_1^{X_B} = (\alpha^k)^{x_B} = (\alpha^{x_B})^k \equiv y_B^k \bmod p = K\,. \qquad \textbf{(3.7)}$$

A partir de *K*, Bob obtém o seu inverso, K^{-1} e o utiliza em c_2, obtendo *m*, como é observado na expressão abaixo,

$$c_2 \cdot K^{-1} = K \cdot K^{-1} \cdot m \bmod p = m \bmod p = m\,. \qquad \textbf{(3.8)}$$

Nota-se que não é aconselhável usar o mesmo *k* para cifrar mais de um bloco de mensagem, uma vez que o conhecimento de um bloco de mensagem, permite que o invasor calcule os outros blocos. A fim de ilustrar, suponha que o invasor conseguiu interceptar o bloco de mensagem m_1 e que o bloco m_2 (desconhecido) foi cifrado com mesmo *k* usado na

cifragem de m_1. Sendo ($c_{1,1}$, $c_{2,1}$) o texto cifrado obtido a partir de m_1 e ($c_{1,2}$, $c_{2,2}$) o texto cifrado obtido a partir de m_2,

$$\begin{aligned} c_{1,1} &\equiv \alpha^k \bmod p \quad ; \quad c_{2,1} \equiv m_1 \cdot K \bmod p \\ c_{1,2} &\equiv \alpha^k \bmod p \quad ; \quad c_{2,2} = m_2 \cdot K \bmod p \end{aligned} \tag{3.9}$$

Assim

$$\frac{m_1}{m_2} = \frac{c_{2,1}}{c_{2,2}} \bmod p \Rightarrow m_2 = m_1 \cdot \frac{c_{2,1}}{c_{2,2}} \bmod p\,. \tag{3.10}$$

O que deixa claro a fácil obtenção de m_2 a partir de m_1 por k ter sido utilizado mais de uma vez.

Do exposto acima, conclui-se que quebrar tal cripto-sistema é equivalente a quebrar o protocolo de chave de Diffie-Hellman, ou seja, é equivalente a resolver o PLD.

Propriedades do Cripto-sistema

Observa-se que há aleatoriedade na operação de cifragem, uma vez que cada cifragem de m resulta num texto cifrado (c_1,c_2) diferente, já que para cada cifragem é escolhido um número aleatório k. Tal propriedade previne ataques como um "ataque ao texto provável" (*probable text attack*). Neste ataque, se o criptoanalista desconfia que m é o texto claro ele o cifra a fim de confirmar se o texto claro é realmente m.

Devido a estrutura do cripto-sistema não há relação obvia entre m_1 , m_2 e m_1m_2 ou qualquer combinação simples de m_1 e m_2.

ii) O Esquema de Assinatura Digital

Considere que Alice deseja assinar um certo documento m, de modo que não apenas Bob como também qualquer um que tenha acesso a chave pública de Alice possa verificar a autenticidade da assinatura, porém apenas Alice possa gerá-la. No processo de assinatura será usada a chave secreta x de Alice e a chave pública $y = \alpha^x \bmod p$ (eq. 3.3). A assinatura de m será o par (r , s), $0 \le r$, $s \le p - 1$, escolhidos de forma que a equação abaixo seja satisfeita

$$\alpha^m \equiv y^r r^s \bmod p \tag{3.12}$$

onde α é um elemento primitivo de GF(p), p é um número primo grande e $p - 1$ possui pelo menos um fator primo grande q.

Processo de Assinatura

Inicialmente, Alice escolhe um número aleatório k, uniformemente distribuído no intervalo $0 \leq k \leq p-1$, tal que $mdc(k, p-1) = 1$. Selecionado k, Alice calcula

$$r \equiv \alpha^k \bmod p . \quad \textbf{(3.13)}$$

Desta forma, com (3.13) e (3.12), tem-se

$$\alpha^m \equiv \alpha^{xr}\alpha^{ks} \bmod p . \quad \textbf{(3.14)}$$

Então,

$$m \equiv (xr + ks) \bmod (p-1), \quad \textbf{(3.15)}$$

que pode ser resolvida a fim de se obter s.

Processo de Verificação de Assinatura

O processo de verificação é muito simples; é necessário apenas usar m, r e s para verificar (3.13).

c) DSA (Algoritmo de Assinatura Digital)

O Algoritmo de Assinatura Digital (DSA – do inglês Digital Signature Algorithm) faz parte do padrão DSS (Digital Signature Standard) [NIST00] e é usado para geração e verificação de assinaturas digitais. Este padrão é aplicável a todos os departamentos e agências Federais dos EUA para a proteção de informações sensíveis não classificadas (De acordo com a seção 2315 do Titulo 10 ou da seção 3502(2) do Titulo 44 do Código dos EUA [NIST00]). O padrão também pode ser adotado para organizações privadas e comerciais.

O algoritmo foi feito com o objetivo de ser usado em correio eletrônico, transferência de fundos, troca eletrônica de dados, distribuição de software, armazenamento de dados, e outras aplicações que necessitam da garantia da integridade dos dados e da autenticidade da origem dos dados.

Uso do Algoritmo para Assinatura Digital

O DSA é usado pelo *assinante* para gerar a assinatura digital e pelo *verificador* para verificar a autenticidade da assinatura. Cada assinante possui uma chave secreta e uma chave pública que são usadas, respectivamente, na geração e na verificação da assinatura. Tanto para o processo de assinatura quanto para o processo de verificação, a mensagem M (dados) é compactada usando o SHA-1 [NIST00]. Sem o conhecimento da chave secreta, o

adversário não pode gerar a assinatura correta, ou seja, a assinatura não pode ser forjada. Qualquer um que tenha acesso a chave pública do assinante pode verificar a assinatura corretamente.

Uma maneira de associar o par chave pública/chave secreta com o respectivo usuário é necessária. Para isso requer-se a presença de uma terceira parte confiável que possa assinar credenciais, contendo a chave pública e a identidade do usuário possuidor de tal par de chaves, a fim de se formar um certificado digital.

Parâmetros do DSA

O DSA utiliza os seguintes parâmetros:

1. p, um modulo primo, onde $2^{L-1} \le p \le 2^{L}$ para $512 \le L \le 1024$ e L é um múltiplo de 64.

2. q, primo, divisor de p–1, onde $2^{159} \le q \le 2^{160}$.

3. $g = h^{(p-1)/q} \bmod p$, onde h é qualquer inteiro com $1 < h < p$–1 tal que $h^{(p-1)/q} \bmod p > 1$ (g tem ordem q mod p).

4. x, inteiro aleatório ou pseudo-aleatório gerado com $0 < x < q$.

5. $y = g^{x} \bmod p$.

6. k, inteiro aleatório ou pseudo-aleatório ($0 < k < q$).

Os inteiros p, q e g podem ser públicos e também podem ser comuns a um grupo de usuários. As chaves secreta e pública do usuário são respectivamente x e y. Estas chaves são normalmente mantidas fixas por um período de tempo. Os parâmetros x e k são usados apenas na geração da assinatura digital e têm que ser mantidos secretos. O parâmetro k é gerado para cada assinatura.

Geração de Assinatura no DSA

A assinatura da mensagem M é dada por r e s, a qual é gerada a partir das equações abaixo:

$$r = (g^{k} \bmod p) \bmod q, \qquad \textbf{(3.16a)}$$

$$s = (k^{-1}(SHA-1(M) + xr)) \bmod q, \qquad \textbf{(3.16b)}$$

onde k^{-1} é o inverso multiplicativo de k modulo q, isto é $kk^{-1} \equiv 1 \bmod q$, $0 < k^{-1} < q$. O valor de SHA–1(M) é uma seqüência de 160 bits resultante da saída da função *hash*, especificada no FIPS PUB 180-1, cuja entrada é M.

Verificação da Assinatura

A priori o usuário que vai verificar a mensagem tem em mãos p, q e g e ainda a chave pública e a identidade do assinante, as quais são disponibilizadas de forma autenticada. Sejam M', r' e s' as versões recebidas de M, r e s, respectivamente, e seja y a chave pública do assinante. Inicialmente, para verificar a assinatura, observa-se se $0 < r' < q$ e $0 < s' < q$. Se pelo menos uma das condições for violada, a assinatura deve ser rejeitada. Se ambas as condições forem obedecidas, computa-se

$$w = (s')^{-1} \bmod q, \qquad \textbf{(3.17a)}$$

$$u1 = ((SHA-1(M'))w) \bmod q, \qquad \textbf{(3.17b)}$$

$$u2 = ((r')w) \bmod q\,, \qquad \textbf{(3.17c)}$$

$$v = (((g)^{u1}(y)^{u2}) \bmod p) \bmod q\,. \qquad \textbf{(3.17d)}$$

Se $v = r$', então a assinatura está verificada de maneira altamente confiável, garantindo que a mensagem foi assinada pelo usuário possuidor da chave secreta *x* e sua correspondente chave pública *y*. Se v ≠ r' ou a mensagem foi modificada ou a mensagem foi assinada de forma incorreta ou ainda, a mensagem foi assinada por um impostor. De qualquer forma, a assinatura deve ser invalidada.

3.4 – HMAC-MD5-96

O MD5 [R92b] é utilizado juntamente com o HMAC (*Hash Message Authentication Code*) [MG98] como uma ferramenta de autenticação no EPS e no AH. Tal protocolo é denominado HMAC-MD5-96. A meta da utilização destes mecanismos é assegurar que o pacote enviado é autentico e que não foi modificado em trânsito. Nesta seção tal protocolo, utilizado pelo IPSec, será descrito.

3.4.1 – MD5

O algoritmo *hash* MD5 é uma função *hash* que transforma uma entrada de comprimento arbitrário numa saída de comprimento 128 bits chamada *sumário de mensagem*. Conjectura-se que é computacionalmente inviável conseguir dois sumários de mensagem idênticos para entradas diferentes, ou produzir uma mensagem que tenha um sumário de mensagem pré-especificado.

O MD5 é uma extensão do MD4, resultado de sugestões feitas por vários revisores e contendo otimizações adicionais. Apesar de perder um pouco na velocidade com relação ao MD4, o MD5 possui um nível de segurança mais elevado. Além disso, o mesmo foi projetado a fim de ser mais rápido em máquinas de 32 bits.

Descrição do Algoritmo

Suponha uma entrada de comprimento *b* bits, onde *b* é não negativo, podendo ser não necessariamente múltiplo de 8 e arbitrariamente grande. Considere a representação da entrada como sendo: $m_0 \quad m_1 \quad \cdots \quad m_{b-1}$.

O calculo do sumário da mensagem é constituído de cinco passos descritos a seguir:

1) Adicionando bits

A mensagem é estendida de modo que seu comprimento (em bits) seja congruente a 448 mod 512, i. e., tenha um comprimento que seja apenas 64 bits menor que um múltiplo de 512. A extensão é feita adicionando-se apenas um bit "1" ao fim da mensagem e mais outros bits "0" até que o comprimento seja congruente a 448 mod 512.

2) Comprimento Adicionado

Uma representação em 64 bits do comprimento da mensagem (antes da extensão) é adicionada ao resultado obtido no passo 1). Se a mensagem possuir comprimento maior que 64 bits, apenas os 64 bits de menor ordem serão usados. Neste ponto a mensagem resultante (depois de estendida e adicionada aos bits da mensagem de entrada) terá um comprimento que será exatamente um múltiplo de 512. De modo equivalente, a mensagem resultante terá uma mensagem com comprimento de 16 palavras de 32 bits. Denota-se a mensagem resultante por $M[0 \ldots N-1]$, onde N é um multiplo de 16.

3) Inicializando o *buffer* MD

Um buffer de quatro palavras (*A*, *B*, *C*, *D*) é utilizado para calcular o sumário de mensagem. Cada uma das variáveis *A*, *B*, *C*, *D* é um registro de 32 bits. Esses registros são inicializados com os seguinte valores hexadecimais:

A = 01 23 45 67
B = 89 ab cd ef
C = fe dc ba 98
D = 76 54 32 10

4) Processamento da mensagem em 16 blocos de 32 bits

Neste momento o enlace principal do algoritmo começa e continua até que o ultimo bloco de 512 bits seja processado. As quatro variáveis mostradas no passo (4) são copiadas em diferentes variáveis: *a* recebe *A*, *b* recebe *B*, *c* recebe *C* e *d* recebe *D*. O enlace principal tem quatro rodadas (Figura 3.1), todas muito similares. Cada rodada usa uma operação diferente 16 vezes.

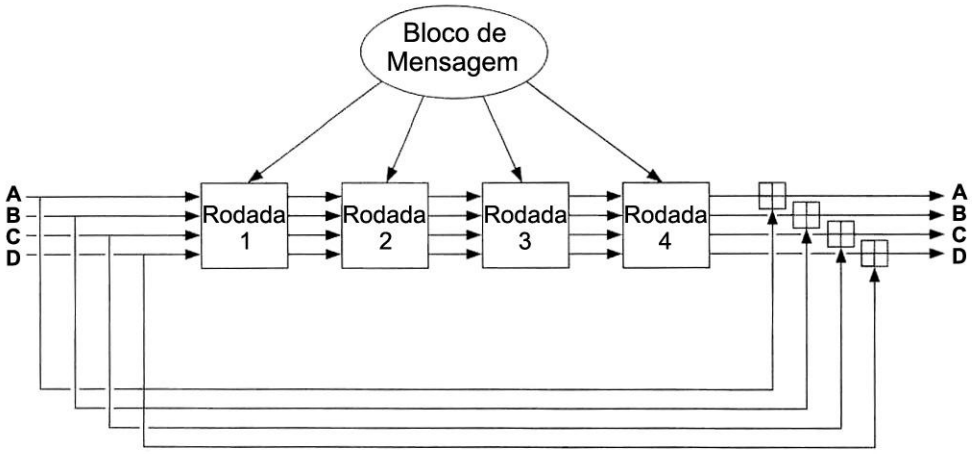


Figura 3. 13 - Algoritmo MD5.

Cada operação executa uma função não-linear sobre três das quatro variáveis *a*, *b*, *c* e *d*. O resultado desta função não-linear é adicionado a quarta variável, a um sub-bloco da mensagem e a uma constante; o resultado obtido é rotacionado à esquerda uma quantidade variável de bits e adicionado a uma das variáveis *a*, *b*, *c* e *d* (Figura 3.2).

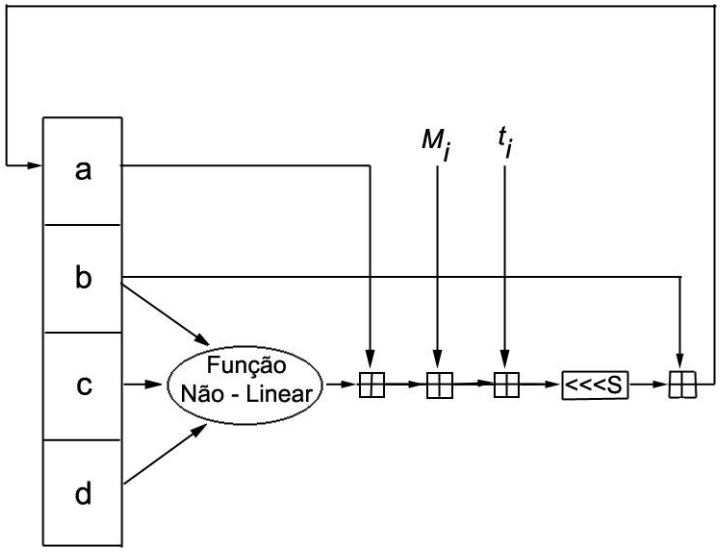


Figura 3. 14 - Uma operação do MD5.

As quatro funções não-lineares em questão tem como entrada três palavras de 32 bits e como saída uma palavra de 32 bits e são diferentes para cada uma das rodadas.

$F(X, Y, Z) = (X \text{ AND } Y) \text{ OR } (\text{NOT}(X) \text{ AND } Z)$;

$G(X, Y, Z) = (X \text{ AND } Z) \text{ OR } (Y \text{ AND NOT}(Z))$;

$H(X, Y, Z) = X \text{ XOR } Y \text{ XOR } Z$;

$I(X, Y, Z) = Y$ XOR $(X$ OR NOT$(Z))$.

Estas funções são projetadas de modo que os bits correspondentes de *X*, *Y* e *Z* são independentes e não polarizados, desta forma tendo cada bit do resultado também independentes e não polarizados. A função *F* é conditional bit a bit: Se *X* então *Y*, caso contrário *Z*. A função *H* é um operador de paridade bit a bit.

Se M_j representa o *j*-ésimo sub-bloco da mensagem $(0 \le j \le 15)$, e <<<*s* representa um deslocamento circular de *s* bits à esquerda, as quatro operações são dadas por:

$FF(a,b,c,d,M_j,s,t_i)$ denota $a = b + ((a + F(b,c,d) + M_j + t_i) <<< s)$;

$GG(a,b,c,d,M_j,s,t_i)$ denota $a = b + ((a + G(b,c,d) + M_j + t_i) <<< s)$;

$HH(a,b,c,d,M_j,s,t_i)$ denota $a = b + ((a + H(b,c,d) + M_j + t_i) <<< s)$;

$II(a,b,c,d,M_j,s,t_i)$ denota $a = b + ((a + I(b,c,d) + M_j + t_i) <<< s)$.

Nas quatro rodadas tem-se:

Rodada 1:

$FF(a,b,c,d,M_0,7,0xd76aa478)$
$FF(a,b,c,d,M_1,12,0xe8c7b756)$
$FF(a,b,c,d,M_2,17,0x242070db)$
$FF(a,b,c,d,M_3,22,0xc1bdceee)$
$FF(a,b,c,d,M_4,7,0xf57c0faf)$
$FF(a,b,c,d,M_5,12,0x4787c62a)$
$FF(a,b,c,d,M_6,17,0xa8304613)$
$FF(a,b,c,d,M_7,22,0xfd469501)$
$FF(a,b,c,d,M_8,7,0x698098d8)$
$FF(a,b,c,d,M_9,12,0x8b44f7af)$
$FF(a,b,c,d,M_{10},17,0xffff5bb1)$
$FF(a,b,c,d,M_{11},22,0x895cd7be)$
$FF(a,b,c,d,M_{12},7,0x6b901122)$
$FF(a,b,c,d,M_{13},12,0xfd987193)$
$FF(a,b,c,d,M_{14},17,0xa679438e)$
$FF(a,b,c,d,M_{15},22,0x49b40821)$

Rodada 2:

$GG(a,b,c,d,M_{0},5,0xf\,61e2562)$
$GG(a,b,c,d,M_{1},9,0xc040b340)$
$GG(a,b,c,d,M_{2},14,0x265e5a51)$
$GG(a,b,c,d,M_{3},20,0xce9b6c7aa)$
$GG(a,b,c,d,M_{4},5,0xd62\,f105d)$
$GG(a,b,c,d,M_{5},9,0x02441453)$
$GG(a,b,c,d,M_{6},14,0xd8a1e681)$
$GG(a,b,c,d,M_{7},20,0xe7d3\,fbc8)$
$GG(a,b,c,d,M_{8},5,0x21e1cde6)$
$GG(a,b,c,d,M_{9},9,0xc33707d6)$
$GG(a,b,c,d,M_{10},14,0xf\,4d50d87)$
$GG(a,b,c,d,M_{11},20,0x455a14ed)$
$GG(a,b,c,d,M_{12},5,0xa9e3e905)$
$GG(a,b,c,d,M_{13},9,0xfcefa3\,f8)$
$GG(a,b,c,d,M_{14},14,0x676\,f02d9)$
$GG(a,b,c,d,M_{15},20,0x8d2a4c8a)$

Rodada 3:

$HH(a,b,c,d,M_{0},4,0xfffa3942)$
$HH(a,b,c,d,M_{1},11,0x8771\,f681)$
$HH(a,b,c,d,M_{2},16,0x6d9d6122)$
$HH(a,b,c,d,M_{3},23,0xfde5380c)$
$HH(a,b,c,d,M_{4},4,0xa4beea44)$
$HH(a,b,c,d,M_{5},11,0x4bdecfa9)$
$HH(a,b,c,d,M_{6},16,0xf\,6bb4b60)$
$HH(a,b,c,d,M_{7},23,0xfbebfbc70)$
$HH(a,b,c,d,M_{8},4,0x289b7ec6)$
$HH(a,b,c,d,M_{9},11,0xeaa127\,fa)$
$HH(a,b,c,d,M_{10},16,0xd4ef\,3085)$
$HH(a,b,c,d,M_{11},23,0x04881d05)$
$HH(a,b,c,d,M_{12},4,0xd9d4d039)$
$HH(a,b,c,d,M_{13},11,0xe6db99e5)$
$HH(a,b,c,d,M_{14},16,0x1\,fa27cf\,8)$
$HH(a,b,c,d,M_{15},23,0xc4ac5665)$

Rodada 4:

$$II(a,b,c,d,M_0,6,0xf\,4292244)$$
$$II(a,b,c,d,M_1,10,0x432aff\,97)$$
$$II(a,b,c,d,M_2,15,0xab9423a7)$$
$$II(a,b,c,d,M_3,21,0xfc93a039)$$
$$II(a,b,c,d,M_4,6,0x655b59c3)$$
$$II(a,b,c,d,M_5,10,0x8f0ccc92)$$
$$II(a,b,c,d,M_6,15,0xffeff\ 47d)$$
$$II(a,b,c,d,M_7,21,0x85845dd1)$$
$$II(a,b,c,d,M_8,6,0x6\,fa87e4\,f)$$
$$II(a,b,c,d,M_9,10,0xfe2ce6e0)$$
$$II(a,b,c,d,M_{10},15,0xa3014314)$$
$$II(a,b,c,d,M_{11},21,0x4e0811a1)$$
$$II(a,b,c,d,M_{12},6,0xf\,7537e82)$$
$$II(a,b,c,d,M_{13},10,0xbd3af\,235)$$
$$II(a,b,c,d,M_{14},15,0x2ad7d2bb)$$
$$II(a,b,c,d,M_{15},21,0xeb86d391)$$

As constantes t_i são escolhidas da seguinte forma: No passo i, i é a parte inteira de $2^{32} * abs(\text{sen}(i))$, onde i é dado em radianos. Em seguida *a*, *b*, *c* e *d* são adicionadas a *A*, *B*, *C* e *D*, respectivamente, e o algoritmo continua com o próximo sub-bloco de mensagem. A saída final é a concatenação de *A*, *B*, *C* e *D*.

Segurança

Com relação ao MD4, o MD5 teve adicionadas características a fim de aumentar o nível de segurança com relação ao anterior, como por exemplo:

1. A função *G* da segunda rodada foi mudada de ((*X* AND *Y*) OR (*X* AND *Z*) OR (*Y* AND *Z*)) para ((*X* AND *Z*) OR (*Y* AND NOT(*Z*)) a fim de diminuir a simetria de *G*. Isto promove um efeito avalanche mais rápido.

2. A ordem na qual os sub-blocos de mensagem são acessados nas rodadas 2 e 3 foi mudada, a fim de fazer estes padrões menos parecidos.

3. As quantidades deslocadas circularmente à esquerda foram otimizadas a fim de produzir um efeito avalanche mais rápido. Os quatro deslocamentos usados em cada rodada são diferentes daqueles usados em outras rodadas.

Tom Berson tentou atacar o MD5 usando criptoanálise diferencial [B92], mas tal ataque não é efetivo em todas as quatro rodadas. O ataque mais bem sucedido foi feito por den Boer e Bosselaers produzindo colisões usando a função de compressão no MD5[BB94, R93, R94].

3.4.2 – HMAC

Um MAC (*Message Authentication Code*) é uma função *hash* unidirecional chave-dependente que possui as mesmas propriedades das funções *hash* unidirecionais (e.g. SHA-1 [NIST95], MD4 [R92a], MD5, etc), e além disso inclui uma chave. Desta forma apenas o usuário com uma chave idêntica será capaz de verificar o *hash*.

Tipicamente, MACs são utilizados por duas partes afim de validar a informação transmitida entre as mesmas. Uma das maneiras mais simples de se construir um MAC é cifrar a mensagem com uma cifra de bloco no modo CBC ou CFB [S96, pp. 446 – 456].

Neste trabalho serão apenas analisadas os HMACs, i. e., MACs baseados em funções *hash* [BCK96]. Uma maneira simples de tornar uma função *hash* num MAC é cifrar o valor *hash* com um criptosistema simétrico.

Um HMAC é um algoritmo de autenticação de chave privada e pode ser usado em combinação com qualquer função *hash* interativa (MD5, SHA-1). Integridade e autenticação da origem dos dados proporcionada pelo HMAC depende de uma distribuição segura das chaves. No caso do ESP e AH é utilizada a função *hash* MD5 no HMAC, formando o HMAC-MD5-96.

A chave secreta utilizada pelo HMAC pode ter qualquer comprimento, se o comprimento da chave for maior que o comprimento de bloco da função *hash*, inicialmente a chave passará pela função *hash* e o resultado será usado como chave do HMAC. De qualquer forma o menor valor recomendado para o comprimento da chave é o comprimento da saída da função *hash* [KA98].

3.4.3 – HMAC-MD5-96

Algoritmo

Os algoritmos MD5 e HMAC já foram descritos nas seções anteriores. O HMAC-MD5-96 opera sobre blocos de dados de 64 bytes e produz como saída um valor de autenticação de 128 bits, o qual pode ser truncado [KBC97]. Para ser utilizado tanto no AH

quanto no ESP, um valor truncado utilizando os primeiros 96 bits deve ser permitido. Ao ser enviado, tal valor é armazenado no campo de autenticação. Na recepção, todos os 128 bits são computados e os primeiros 96 bits são comparados com o valor armazenado no campo de autenticação. Nenhum outro comprimento para o autenticador é permitido pelo HMAC-MD5-96. O comprimento de 96 bits foi selecionado por ser o valor *default* especificado para o AH e além disso obedece aos requerimentos de segurança descrito no [KBC97].

Chaves

A chave secreta utilizada pelo HMAC-MD5-96 permite qualquer comprimento, mas para o uso tanto no ESP quanto no AH é especificado que a chave deve ter comprimento 128 bits; outros comprimentos não devem ser permitidos. Chaves de comprimentos menores comprometem a segurança e de comprimentos maiores não tem muita influência sobre a mesma. Mais discussões envolvendo o comprimento da chave estão disponíveis em [KBC97].

A fim de proporcionar autenticação da origem dos dados, o mecanismo de distribuição de chave deve assegurar que chaves únicas são alocadas e distribuídas apenas para as partes participantes da comunicação.

A troca periódica da chave é fundamental na segurança prática pois ajuda contra fraquezas potenciais da função e das chaves, reduz a informação disponível para o criptoanalista e limita o dano causado por uma chave exposta.

Segurança

A segurança do HMAC-MD5-96 é baseada na robustez proporcionada pelo HMAC, e em menor grau pelo MD5. Tal segurança não depende criticamente da forte resistência a colisões, considerada no caso do MD5, o qual recentemente se mostrou não tão resistente a colisões quanto esperado [MG98]. Como em qualquer algoritmo criptográfico parte da segurança se encontra na implementação correta do algoritmo, na segurança do mecanismo de gerenciamento de chaves e sua correta implementação, na segurança da chave associada, e na correta implementação de todos os sistemas participantes. Em [CG97] podem ser encontrados vetores teste e código exemplo para auxiliar na verificação da exatidão do código do HMAC-MD5-96. Ataques práticos contra o HMAC-MD5-96 não

foram obtidos até o momento. Mais comentários sobre a segurança de tal protocolo podem ser encontrados em [KBC97].

3.5 – Rijndael

Sendo a intenção do grupo de trabalho em segurança IP da IETF adotar em breve o AES como padrão na cifra ESP do IPsec, uma descrição do AES é feita neste capítulo. Neste momento a utilização do AES como padrão no IPsec ainda se encontra em fase de teste a fim de determinar como o mesmo deve ser usado da melhor maneira nas implementações do IPsec [KA98].

3.5.1 - Introdução

Após quase quatro anos de competições, em 02 de Outubro de 2000 o NIST (*National Institute of Standards and Technology*) anunciou o AES (*Advanced Encryption Standard*) [AES01], cifra substituta do DES. O processo de seleção contou inicialmente com 15 algoritmos que foram eliminados gradualmente através de participação e comentários públicos [AES02].

O AES, formalmente conhecido como Rijndael [DR98], foi escolhido na fase final entre cinco finalistas. Os outros quatro finalistas, MARS [MARS99], RC6 [RRSY98], Serpent [ABK98] e Twofish [Twofish98], foram considerados suficientemente seguros cabendo a resolução final a dois fatores adicionais:

1) Eficiência computacional e requerimento de memória numa grande variedade de softwares e hardwares, incluindo smart cards;

2) Flexibilidade, simplicidade e facilidade de implementação.

O Rijndael será adotado como cifra padrão do governo dos EUA e espera-se que ainda este ano (Agosto ou Outubro de 2001) seja descrito num FIPS (*Federal Information Processing Standard*) [AES03]. O mesmo deve ser suficiente para proteger informações sensíveis (não classificadas) do governo americano até pelo menos o próximo século. Também espera-se que o AES seja utilizado amplamente pelo comércio e instituições financeiras. O AES será público em todo um mundo numa base Royalty-free.

3.5.2 – Rijndael

Rijndael é uma cifra de bloco iterativa com comprimentos variáveis de bloco e de chave. Seu nome vem da junção do nome dos seu criadores Vicent RIJman e Joan DAEmen. A seguir será iniciada a descrição desta cifra que substitui o DES [S96, pp. 265-301].

Descrição do Algoritmo

Na Cifra os comprimentos de bloco e de chave são variáveis, podendo ser definidos independentemente entre os seguinte valores : 128, 192 ou 256 bits. A Cifra é constituída de:

- Uma rodada inicial de adição de chave;
- **Nr**-1 rodadas;
- Uma rodada final.

Em pseudo C, tem-se:

```
Rijndael(State, CipherKey)
{
KeyExpasion(CipherKey, ExpandedKey);
AddRoundKey(State, ExpandedKey);
For (i = 1; i < Nr; i ++)
Round(State, ExpandedKey + Nb*i ;
FinalRound(State, ExpandedKey + Nb*Nr);
}
```

A expansão da chave pode ser feita antes e Rijndael pode ser especificada em termos da Chave Expandida[12].

```
Rijndael(State, ExpandedKey)
{
AddRoundKey(State, ExpandedKey);
For (i = 1; i < Nr; i ++)
Round(State, ExpandedKey + Nb*i ;
FinalRound(State, ExpandedKey + Nb*Nr);
}
```

Maiores detalhes sobre a cifra assim como as funções e variáveis utilizadas na descrição geral da mesma utilizando uma linguagem pseudo C serão dados na subseção a seguir.

Diferentemente das outras cifras de bloco, o Rijndael não possui suas rodadas baseadas na estrutura de Feiestel [S96]. Ao invés disto, a transformação da rodada é constituída de três transformações uniformes e inversíveis distintas, chamadas *camadas*. Uniforme neste caso significa que cada bit do estado é tratado de modo similar. A escolha por diferentes camadas é baseada, principalmente, na aplicação da estratégia *Wide Trail* [D95], a qual é um método de projeto que proporciona resistência contra criptoanalise linear e diferencial. Neste método cada camada tem a sua própria função.

A camada linear: garante alta difusão sobre múltiplas rodadas;

A camada não-linear: aplicação paralela de S-boxes que tem propriedades ótimas de não-linearidade;

A camada da adição de chave: simples operação de ou-exclusivo entre a chave de rodada e o Estado.

Estado, Chave da Cifra e Número de Rodadas

As diferentes transformações operam sobre resultados intermediários da cifra, os quais são chamados de *Estado*. O estado pode ser ilustrado como uma matriz retangular de bytes, com quatro linhas e **Nb** colunas, onde **Nb** é dado pelo comprimento do bloco dividido por 32. A *chave da cifra* também pode ser ilustrada como uma matriz retangular com quatro linhas e **Nk** colunas, onde **Nk** é igual ao comprimento da chave dividido por 32.

Exemplo 7.1:

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

[12] A Chave Expandida deve SEMPRE ser derivada a partir da chave da cifra e nunca especificada diretamente. Porém não há restrições quanto a escolha da chave da cifra.

Tabela 3. 1 - Exemplo de Estado com Nb=6.

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

Tabela 3. 2 - Layout da Chave de Cifra com Nk=4.

Em alguns casos esses blocos (estado e chave da cifra) podem ser considerados matrizes unidimensionais com vetores de 4 bytes, onde cada vetor corresponde a uma coluna da matriz retangular. Assim essas matrizes terão comprimentos 4 (128 bits), 6 (192 bits) ou 8 (256 bits) e consequentemente índices 0.. 3, 0 .. 5, 0 .. 7 respectivamente. Neste capítulo, algumas vezes esses vetores de 4 bytes serão chamados *palavras*.

A entrada e a saída usada por Rijndael na sua interface externa são consideradas matrizes unidimensionais de bytes numeradas de 0 a 4***Nb**-1. Esses blocos possuem comprimentos 16, 24 ou 32 bytes e respectivamente índices da matriz entre 0 .. 15, 0 .. 23 ou 0.. 31. A chave de cifra também é considerada uma matriz unidimensional de bytes numeradas de 0 a 4***Nk**-1 tendo também comprimentos 16, 24 ou 32 bytes e respectivamente índices da matriz entre 0 .. 15, 0 .. 23 ou 0.. 31.

Os bytes de entrada da cifra (o "texto claro" se considerado o modo de cifragem ECB [pp. 189-191, S96]) são mapeados nos bytes de estado na seguinte ordem $a_{0,0}$,$a_{1,0}$, $a_{2,0}$, $a_{3,0}$, $a_{0,1}$, $a_{1,1}$, $a_{2,1}$, $a_{3,1}$,..., e os bytes de chave da cifra são mapeadas na matriz na ordem $k_{0,0}$,$k_{1,0}$, $k_{2,0}$, $k_{3,0}$, $k_{0,1}$, $k_{1,1}$, $k_{2,1}$, $k_{3,1}$,.... No fim da operação de cifragem, a saída é extraída do estado tomando os bytes do estado na mesma ordem.

Daí, numa matriz unidimensional se o índice de um byte é *n* e os índices da matriz bidimensional são dados por (*i*,*j*) , tem-se:

$$i = n \bmod 4; \qquad j = \left\lfloor n/4 \right\rfloor; \qquad n = i + 4 * j .$$

Além disso, o índice *i* indica o byte dentro do vetor de 4 bytes e o *j* o vetor ou palavra dentro do bloco.

O *número de rodadas* é dado por **Nr** e depende de **Nb** e **Nk** (Tabela 3.3).

Nr	**Nb=4**	**Nb=6**	**Nb=8**
Nk=4	10	12	14
Nk=6	12	12	14
Nk=8	14	14	14

Tabela 3. 3 - Número de rodadas como função dos comprimentos de bloco e de chave.

Transformação `Round`

A transformação da Rodada é constituída por quatro transformações diferentes. Em notação pseudo C, tem-se:

```
Round(State, RoundKey)
{
```

```
ByteSub(State);
ShiftRow(State);
MixColumn(State);
AddRoundKey(State, RoundKey);
}
```

A rodada final é um pouco diferente para que a estrutura inversa (decifragem) não seja muito diferente da direta (cifragem):

```
FinalRound(State, RoundKey)
{
ByteSub(State);
ShiftRow(State);
AddRoundKey(State, RoundKey);
}
```

Nesta notação, as funções (`Round`, `ByteSub`, `ShiftRow`, ...) operam sobre matrizes nas quais os apontadores (`State`, `RoundKey`) são dados. As funções são especificadas a seguir:

Transformação `ByteSub`:

Esta é uma substituição não-linear de bytes, operando sobre cada um dos bytes do Estado independentemente. A tabela de substuição (ou S-Box) é inversível e construída pela composição de duas transformações:

1. Calcular o inverso multiplicativo em GF(2^8).

2. Aplicar uma transformação afim sobre GF(2) definida por:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

A aplicação desta transformação sobre os bytes do estado é denotada por: `ByteSub(State)`

A figura 3.3 ilustra o efeito desta transformação sobre os bytes do Estado.

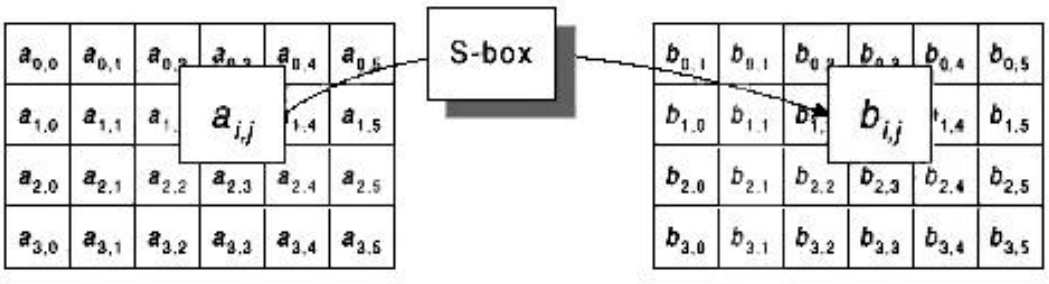


Figura 3. 15 - A função ByteSub agindo sobre cada um dos bytes do estado individualmente.

O inverso desta operação é obtido através da aplicação do inverso do mapeamento afim tomando em seguida o inverso multiplicativo sobre GF(2^8).

Transformação `ShiftRow`

Nesta transformação as linhas do Estado (matriz bidimensional) são deslocadas ciclicamente por quantidades diferentes. A linha 0 não é deslocada, a linha 1 é deslocada C1 bytes, a linha 2 é deslocada C2 bytes e a linha 3 é deslocada C3 bytes. As quantidades C1, C2 e C3 de posições deslocadas dependem do comprimento de bloco e consequentemente de **Nb** (Tabela 3.4).

Nb	**C1**	**C2**	**C3**
4	1	2	3
6	1	2	3
8	1	3	4

Tabela 3. 4 - Deslocamentos para diferentes Nb.

A operação de deslocamento de linhas do Estado sobre deslocamentos específicos é denotada por: `ShiftRow(State)`

A figura 3.4 ilustra o efeito desta operação sobre o Estado.

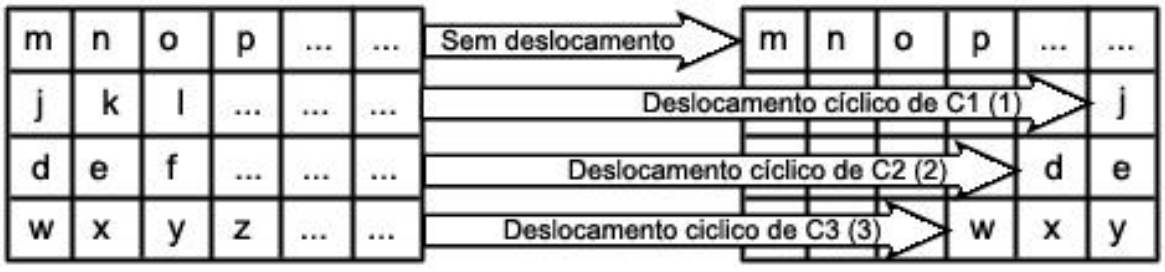


Figura 3. 16 - ShiftRow opera sobre as linhas do estado.

O inverso da `ShiftRow` é dado por um deslocamento cíclico das 3 últimas linhas de **Nb**-C1, **Nb**-C2 e **Nb**-C3 bytes, respectivamente, de modo que o byte da posição *j* na linha *i* seja deslocado para a posição (*j* + **Nb** - C*i*) mod **Nb**.

Transformação `MixColumn`

Na `MixColumn` as colunas do Estado são consideradas polinômios sobre GF(2^8) e multiplicados $\mathrm{mod}(x^4+1)$ com um polinômio fixo, $c(x)$, dado por

$$c(x) = '03'x^3 + '02'x^2 + '01'x + '02'.^{13} \quad \textbf{(3.18)}$$

Esse polinomio é coprimo com x^4+1, sendo portanto inversível. Assim, da seção 2.2 de [DR99], pode-se escrever a expressão (3.18) como uma matriz multiplicação. Seja $b(x) = c(x) \otimes a(x)$,

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

A aplicação dessa operação em todas as colunas do Estado é denotada por: `MixColumn(State)`. A figura 3.5 ilustra o efeito desta transformação sobre o Estado.

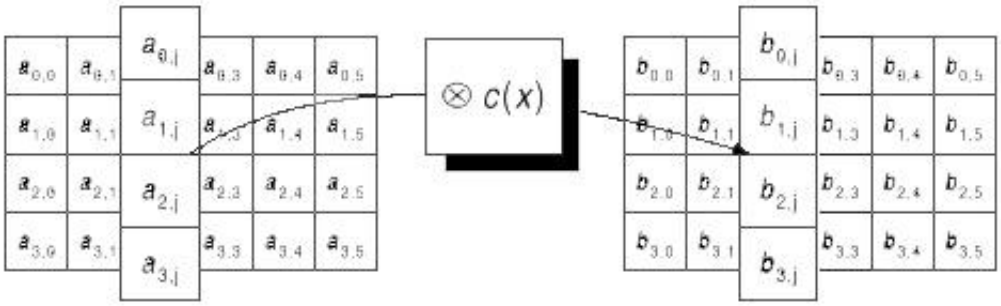


Figura 3. 17 - MixColumn opera sobre as colunas do estado.

O inverso da `MixColumn` é similar a `MixColumn` consistindo da multiplicação de cada coluna pelo polinômio específico $d(x)$, dado por:

$$('03'x^3 + '02'x^2 + '01'x + '02') \otimes d(x) = '01' \quad \textbf{(3.19)}$$

onde $d(x) = '0B'x^3 + '0D'x^2 + '09'x + '0B'$.

Adição da chave de rodada

Esta operação consiste apenas de um ou-exclusivo bit a bit do Estado com a chave de rodada. A *chave de rodada* é gerada a partir da chave da cifra num processo descrito na próxima subseção. O comprimento da chave de rodada é igual ao comprimento do bloco. Essa transformação que consiste na aplicação do ou-exclusivo entre os bits da chave de rodada e do Estado é denotada por `AddRoundKey(State, RoundKey)`. Sua transformação é ilustrada na figura 3.6.

[13] O número da forma '*xy*' é um número hexadecimal.

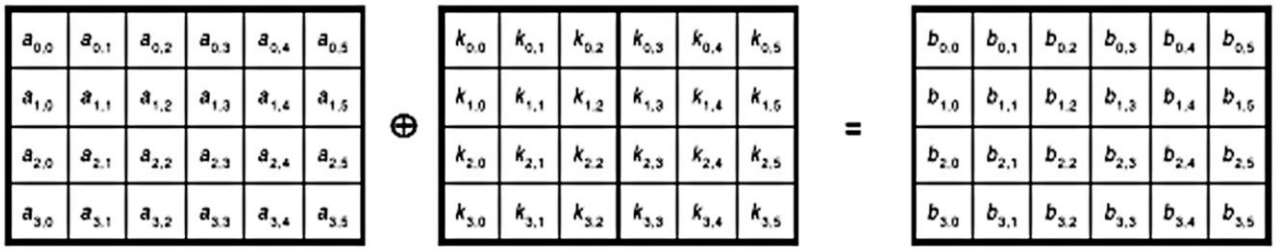


Figura 3. 18 - Na adição da chave cada bit da Chave de Rodada é operada ou-exclusivo com cada bit do Estado.

O inverso desta operação é a própria `AddRoundKey`.

Geração de Chaves

As chaves de rodadas são geradas a partir da chave de cifra utilizando um procedimento constituído de duas partes: a *expansão da chave* e a *seleção da chave de rodada*. O princípio básico é:

- O comprimento da chave expandida em bits é dado pelo comprimento do bloco em bits multiplicado pelo número de rodadas adicionado de 1, i. e., 32x**Nb**x(**Nr**+1) (e.g. para um comprimento de bloco de 128 bits (**Nb**=4) e 10 rodadas (**Nr**=10) serão necessários 1408 bits de chave de rodada).
- A chave de cifra é expandida numa *Chave Expandida*.
- As chaves de rodada são obtidas a partir da Chave Expandida da seguinte maneira: a chave da primeira rodada consiste **Nb** palavras, a segunda das **Nb** palavras seguintes, e assim por diante.

Expansão da Chave

A chave expandida é uma matriz linear de palavras de 4 bytes denotada por `W[Nb*(Nr+1)]`. As primeiras **Nk** palavras contém a chave de cifra. Todas as outras são definidas recursivamente em termos de palavras com índices menores. A função de expansão da chave depende no valor de **Nk**, possuindo uma versão para valores de **Nk** ≤ 6 e outra para **Nk** > 6.

Para **Nk** ≤ 6, tem-se:

```
KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)])
{
     For (i = 0; i < Nk; i++)
     W[i] = (Key[4*i], Key[4*i+1], Key[4*i+2], Key[4*i+3]);

     For (i = Nk; i < Nb * (Nr + 1); i++)
     {
          temp = W[i - 1];
          if   (i % Nk == 0)
```

```
            temp = SubByte(RotByte(temp)) ^ Rcon[o / Nk];
        W[i] = W[i - Nk] ^ temp;
    }
}
```

Nesta descrição , `SubByte(W)` é uma função que retorna uma palavra de 4 bytes e cada byte é resultado da aplicação da S-Box ao byte na posição correspondente na palavra de entrada. A função `RotByte(W)` retorna uma palavra onde os bytes são permutações daqueles da entrada tais que a palavra de entrada (*a*, *b*, *c*, *d*) produz a palavra (*b*, *c*, *d*, *a*).

Nota-se que as primeiras **Nk** palavras são preenchidas com a chave de cifra. Cada palavra seguinte, `W[i]` é igual ao ou-exclusivo entre a palavra anterior, `W[i-1]`, e a palavra **Nk** posições atrás, `W[i-Nk]`. Para palavras em posições múltiplas de **Nk**, a transformação consiste de um deslocamento cíclico dos bytes numa palavra (`RotByte`), seguido da aplicação da substituição de todos os quatros bytes da palavra (`SubByte`).

Para **Nk** > 6, tem-se:

```
KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)])
{
     For (i = 0; i < Nk; i++)
     W[i] = (Key[4*i], Key[4*i+1], Key[4*i+2], Key[4*i+3]);

     For (i = Nk; i < Nb * (Nr + 1); i++)
     {
          temp = W[i - 1];
          if   (i % Nk == 0)
               temp = SubByte(RotByte(temp)) ^ Rcon[o / Nk];
          else if (i % Nk == 4)
               temp = SubByte(temp);
          W[i] = W[i - Nk] ^ temp;
     }
}
```

A diferença com relação ao caso **Nk** ≤ 6 é que para i – 4 multiplo de **Nk**, SubByte é aplicado a W[i-1] antes do EXOR.

As constantes das rodadas são independentes de **Nk** e definidas por :

`Rcon[i]` = (`RC[i]`, ‘00’, ‘00’, ‘00’)

com RC[I] representando um elemento em GF(2^8) com um valor de $x^{(i-1)}$ tal que:

`RC[1]` = 1 (i.e., ‘01’)

`RC[i]` = x (i.e., ‘02’) • `RC[i-1]` = $x \bullet x^{(i-1)}$

Seleção da Chave de Rodada

A Chave de Rodada `i` é dada por `W[Nb*i]` to `W[Nb*(i+1)]` (Figura 3.7)

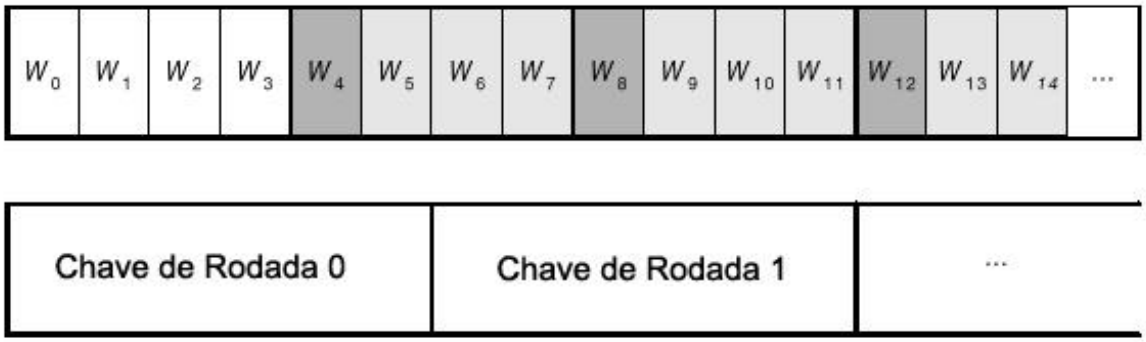


Figura 3. 19 - Expansão e Seleção da chave de rodada para Nb = 6 e Nk = 4.

A geração de chave pode ser implementado sem o uso explicito da matriz `W[Nb*(Nr+1)]`. Para implementações onde a RAM é escassa, as chaves de rodada podem ser computadas em tempo real usando um *buffer* de **Nk** palavras quase sem sobrecarga computacional.

Segurança

O Rijndael parece proporcionar um margem de segurança adequada. A margem de segurança é um pouco dificil de medir porque o número de rodadas muda com o comprimento da chave. Duas críticas foram feitas com relação a segurança: que a sua margem de segurança está num nível menor com relação aos outros finalistas, e que sua estrutura matemática poderá permitir alguns ataques. Porém, sua estrutura é bastante simples, facilitando a analise de segurança no tempo determinado para a escolha do AES [AES02].

3.5.3 – Especificações do IPsec[14]

Modo CBC (*Cipher Block Chaining*)

Os modos de operação para o AES ainda se encontram em fase de definição pelo NIST. Porém, tal modo de operação é bem conhecido e entendido de maneira geral para cifras simétricas como o AES. Desta forma e devido a necessidade de se utilizar a cifra neste modo nas cifras ESP, será especificado aqui, baseado na minuta atual do IETF IPsec sobre a utilização do AES no IPsec [AES01], o modo CBC do AES a ser utilizado no ESP.

Este modo requer um *vetor inicialização* (VI) que tem o mesmo tamanho do bloco de mensagem. A fim de evitar textos cifrados idênticos obtidos a partir de blocos de dados idênticos usa-se vetores inicialização gerados aleatoriamente. O VI é operado ou-exclusivo com o primeiro bloco de texto claro antes de ser cifrado. Para os blocos sucessivos, o bloco de texto cifrado previamente é operado ou-exclusivo com o atual bloco de texto claro, antes da sua cifragem (Figura 3.8).

[14] Esta seção foi escrita baseada num trabalho ainda em progresso (Documento válido até Maio 2001).

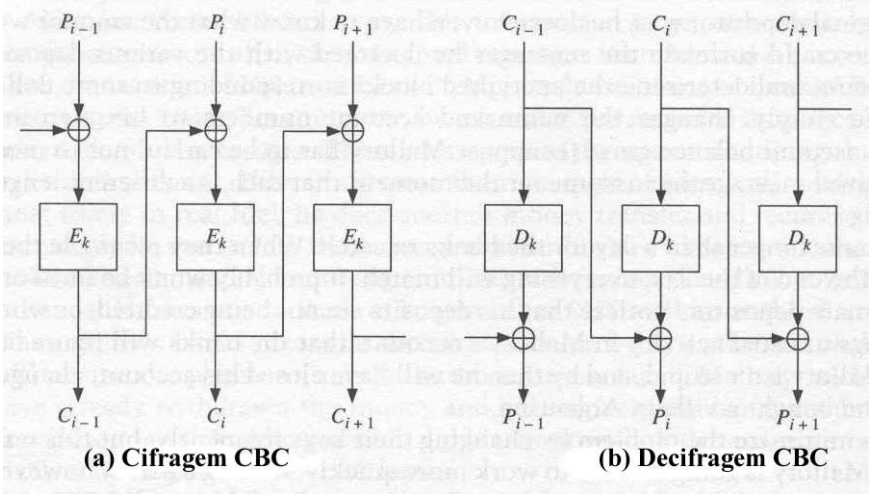


Figura 3. 20 - Modo CBC (Cipher Block Chaining).

Maiores explicações sobre o modo de operação CBC podem ser encontradas em [S96]. O uso do modo CBC em ESP com cifras de 64 bits é descrito em [PR98].

Comprimento da Chave

No caso do AES, a chave pode ter comprimento de 128, 192 ou 256 bits. Com relação ao IPsec a chave deve ter comprimento múltiplo de 8, e por *default* o tamanho da chave usado nas implementações do IPsec deve ser de 128 bits, apesar da mesma permitir outros comprimentos de chave como foi visto.

Até o momento não foram encontradas chaves fracas. Caso tais chaves sejam descobertas recomenda-se que as mesmas sejam verificadas e eliminadas quando utilizando gerenciamento manual de chaves. No caso de mecanismos dinâmicos de gerenciamento de chaves como por exemplo o IKE [BF01], a verificação de chaves fracas não deve ser feita se resultar num aumento desnecessário de complexidade.

Tamanho do Bloco e *Padding*

O comprimento do bloco é de 16 octetos (128 bits). apesar do AES, como foi visto na seção anterior, suportar também blocos maiores.

A fim de que o algoritmo mantenha blocos de comprimento 128 bits, e os dados a serem cifrados (incluindo o comprimento *Pad* e os campos *Next Header*) tenham um comprimento multiplo de 128 bits, é necessário que bits sejam acrescentados (*padding*).

Rodadas

Para o IPsec define-se o valor padrão como sendo de 10 rodadas para a chave já definida de comprimento 128 bits. Como foi visto, o AES tem como possíveis números de rodadas 10, 12 e 14 dependendo do comprimento da chave e do bloco (Tabela 3.3). Daí portanto a escolha de 10 rodadas.

3.5.4 – Conclusão

Este capítulo teve como intuito introduzir o AES/Rijndael e como o mesmo tem sido definido nesta fase de testes no IPsec. Embora neste trabalho só tenham sido colocadas informações sobre o Rijndael, os outros finalistas também foram considerados para uso no IPsec [FKG00].

3.6 – Considerações Finais

Observa-se que os cripto-sistemas de chave pública possuem algumas vantagens sobre os cripto-sistemas de chave secreta, como por exemplo, a não necessidade de transmissão por um canal seguro da chave secreta e a autenticação sem possibilidade de repudiação; porém, os cripto-sistemas simétricos são mais rápidos que os assimétricos e

por esta razão geralmente utiliza-se ambos os tipos de cripto-sistemas de modo combinado, aproveitando as vantagens que os mesmos oferecem.

Referências:

[AES01] – *AES home page*,
http://www.nist.gov/aes.

[AES02] – J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, E. Roback, *Report on the Development of the Advanced Encryption Standard (AES)*, Computer Security Division, Information Thecnology Laboratory, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, Outubro 2000.
http://csrc.nist.gov/encryption/aes/round2/r2report.pdf

[AES03] – Advanced Encryption Standard (AES) – Questions and Answers.
http://csrc.nist.gov/encryption/aes/aesfact.html

[ABK98] – R. Anderson, E. Biham e L. Knudsen, *Serpent: A Proposal for the Advanced Encryption Standard*, NIST AES Proposal, Junho 1998.
http://www.cl.cam.ac.uk/~rja14/serpent.html

[B86] – New Enciclopaedia Britannica, vol. 16, 15th edition, pp. 913-924B, 1986.

[B92] – T. Berson, *Differencial Cryptoanalysis mod* 2^{32} *with Applications to MD5*, Advances in Cryptology – EUROCRYPT '92 Proceedings, pp. 71-80, 1992.

[BB94] – B. den Boer e A. Bosselaers, Collisions for the Compression Function of MD5, Advances in Cryptology – EUROCRYPT '93 Proceedings, Springer-Verlag, pp. 293-304, 1994.

[BCK96] M. Bellare, R. Canetti, and H. Krawczyk, *Keyed Hash Functions and Message Authentication*, Proceedings of Crypto'96, LNCS 1109, pp. 1-15.
http://www.research.ibm.com/security/keyed-md5.html

[BF01] – S. Blake-Wilson e P. Fahn, , *IKE Authentication Using ECDSA*, IPSec Working Group INTERNET DRAFT, Março 2001.

[BLP94] – Buhler, J. P., Lenstra, H. W. e Pomerance, C., *The development of Number Field Sieve*, Lectures Notes in Computer Science,Volume 1554, Springer-Verlag, 1994.

[BLZ94] – Buchmann, J., Loho, J. e Zayer, J., *An Implementation of the General Number Field Sieve*, Advances in Cryptology Crypto 93', pp. 159-166, 1984.

[CW97] – *An Introdution to Information Security*, A Certicom White Paper, 1997.
http://www.certicom.com

[D95] – J. Daemen, *Cipher and Hash Fuction Design Strategies Basead on Linear and Differential Cryptanalysis*, Doctoral Dissertation, K. U Leuven, Março 1995.

[DH76] – Diffie, Whitfield e E. Hellman, Martin, *New Directions in Cryptography*, IEEE Transactions on Information Theory, Vol. IT.22, No. 6, pp. 644-654, Novembro 1976.

[DR99] – J. Daemen e V. Rijman, *AES Proposal: Rijndael*, NIST AES Proposal, versão 2, Setembro 1999.

[FKG00] – S. Frankel, S. Kelly e R. Glenn - IPsec Work Group, *The AES Cipher Algorithm and Its use with IPsec*, Network Working Group, Internet Draft, Novembro 2000.

http://ietf.org/internet-drafts/draft-ietf-ipsec-ciph-aes-cbc-01.txt

[E85a] –ElGamal, Taher, *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE Transactions on Information Theory, Vol. IT.31, No. 4, pp. 469-481, Julho 1985.

[E85b] - ElGamal, T., *A Subexponencial-Time Algorithm for Computing Discrete Logarithms over GF(p^2)*, IEEE Trans. Inform. Theory, vol. IT-31, pp. 473 – 479, Julho 1985.

[K81] - Konheim, A. G. , *Cryptography: A Primer*, Wiley, 1981.

[K92] – Kaliski Jr., B. S., *RFC 1319: The MD2 Message-Digest Algorithm*, RSA Laboratories, Abril 1992.

[K98] – Knuth, D., *The Art of Computer Programing*, Vol.2, Semi-Numerical Algorithms, Reading, MA. : Addison – Wesley, 1998.

[KA98] – S. Kent e R. Atkinson, *Security Architeture for the Internet Protocol*, Network Working Group Request for Comments 2401, Novembro 98.

[KBC97] – H. Krawczyk, M. Bellare e R. Canetti, *HMAC: Keyed-Hashing for Message Authentication*, Network Working Group Request for Comments 2104, Fevereiro 1997.

[L87] – Lenstra Jr., H. W., *Factoring Integers with Elliptic Curves*, Annuals of Mathematics, 126, pp. 649-673, 1987.

[LO91] – B. A. LaMacchia, A. M. Odlyzko, *Computation of Discrete Logarithms in Prime Fields*, 1991.

[MARS] – C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. Matyas Jr., L. O'Connor, M. Peyravian, D. Safford, N. Zunic, *MARS – a candidate for AES*, IBM Corporation, Revised, 22 de Setembro 1999.
http://www.research.ibm.com/security/mars.pdf

[MG98] – C. Madson e R. Glenn, *The Use of HMAC-MD5-96 within ESP and AH*, Network Working Group Request for Comments 2403, Novembro 1998.

[NIST95] - National Institute of Standards and Technology (NIST). *FIPS Publication 180-1: Secure Hash Standard (SHS)*, 17 de Abril de 1995. http://www.itl.nist.gov/fipspubs/fip180-1.htm

[NIST00] – National Institute of Standards and Technology (NIST). *FIPS Publication 186-2: Digital Signature Standard (DSS)*, 27 de Janeiro de 2000. http://csrc.nist.gov/fips/fips186-2.pdf

[O85] – A. M., Odlyzko, *Discrete Logarithms in Finite Fields and their Cryptographic Significance*, Advances in Cryptology: Proceedings of Eurocrypt '84, T. Beth, N. Cot, I. Ingemarsson, eds., Lecture Notes in Computer Science 209, Springer-Verlag, NY (1985), pp. 224-314.

[P78] – Pollard, J. M., *Monte Carlo Methods for Index Computation (mod p)*, Mathematics of Computation, vol. 32, nº 143, pp. 918-924, Julho 1978.

[PH78] - Pohlig, S. C. e Hellman, M. E., *An Improved Algorithm for Computing Logarithms over GF(p) and Its Cryptographic Significance*, IEEE Trans. Inform. Theory, vol. IT-24, No. 1, pp. 106 – 110, Janeiro 1978.

[PR98] – R. Pereira e R. Atkinson, *The ESP CBC-mode Cipher Algorithms*, RFC 2451, Novembro 1998.

[RSA78] – Rivest, R. L., Shamir, A. e Adleman, L. M., *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, Communications of the ACM, 21(2), pp. 120-126, Fevereiro 1978.

[R92a] – R. L. Rivest, *The MD5 Message Digest Algorithm*, Network Working Group Request for Comments 1321, Abril 1992.

http://www.faqs.org/rfcs/rfc1321.html

[R92b] – R. L. Rivest, The MD4 Message Digest Algorithm, Network Working Group Request for Comments 1320, Abril 1992.

http://www.faqs.org/rfcs/rfc1320.html

[R93] - M. J. B. Robshaw, Implementation of the Search for Pseudo-Colisions in MD5, Technical Report TR-103, Version 2.0, RSA Laboratories, Novembro 1993.

[R94] - M. J. B. Robshaw, On Pseudo-Colisions in MD5, Technical Report TR-102, Version 1.1, RSA Laboratories, Julho 1994.

[RSA00] - RSA Laboratories'Frequently Asked Questions About Today's Cryptography, versão 4.1, 2000.
http://www.rsalabs.com/faq/index.html

[RRSY98] – R. Rivest, M. Robshaw, R. Sidney e Y. Yin, *The RC6 Block Cipher*, Agosto 1998.
ftp://ftp.rsasecurity.com/pub/rsalabs/rc6/rc6v11.pdf

[S87] – *Breaking the Enemy's Code*, IEEE Spectrum, pp. 47-51, Setembro 1987.

[SP89] – Seberry, Jennifer e Pieprzyk, Josef, *Cryptography: An Introduction to Computer Security*, Advances in Computer Science Series, Prentice Hall, 1989.

[S96] – Scheneier, Bruce, *Applied Cryptography Second Edition: Protocols, Algorithms and Source Code in C*, John Wiley & Sons, Inc., 1996.

[Twofish98] – B. Scheneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson, *Twofish: A 128-Bit Block Cipher*, 15 Junho de 1998.
http://www.counterpane.com/twofish.pdf

[WW84] - Wah , P. K. S. e Wang, M. Z., *Realization and Application of the Massey-Omura Lock*, Proc. Intern. Zurich Seminar, pp. 175-182, Março 1984.

4

CRIPTOGRAFIA EM CURVAS ELÍPTICAS

Neste capítulo serão estudados os cripto-sistemas baseados no Problema do Logaritmo Discreto sobre Curvas Elípticas (PLDCE). Inicialmente será dada uma breve explicação sobre o PLDCE, seguido pela descrição de alguns cripto-sistemas baseados neste problema.

As seções subsequentes são constituídas de uma analise da eficiência de tais cripto-sistemas com relação aos cripto-sistemas baseados em outros problemas matemáticos; da citação de algumas aplicações e informações sobre alguns processos de padronização em andamento.

4.1 - Introdução

No capítulo anterior os cripto-sistemas assimétricos foram introduzidos. Observou-se que a segurança de tais cripto-sistemas é baseada na dificuldade de se resolver problemas matemáticos.

Desde a introdução da idéia de cripto-sistemas assimétricos até meados da década de 80 apenas dois problemas se mostraram efetivos na construção de cripto-sistemas deste tipo: fatoração de números inteiros e o problema do logaritmo discreto sobre corpos finitos (PLD).

Na década de 80, Neil Koblitz [K87] e Victor Miller [M86] chegaram, de forma independente, à conclusão que a rica estrutura matemática das curvas elípticas é uma fonte de problemas matemáticos de difícil solução e, portanto, poderia ser utilizada como base para a construção de cripto-sistemas assimétricos.

Cripto-sistemas baseados nestas estruturas algébricas foram propostos e os mesmos são análogos aos cripto-sistemas assimétricos já existentes. No caso dos análogos ao RSA [D94, KMOV92] o interesse é mais acadêmico do que prático uma vez que não mostram vantagens sobre o mesmo.

Já os análogos àqueles baseados no PLD sobre corpos finitos se mostram mais vantajosos. Neste caso, tratando o PLDCE observa-se uma vantagem com relação ao nível de segurança uma vez que os algoritmos conhecidos atualmente para solucionar o PLD possuem complexidade subexponencial enquanto que para solucionar o mesmo problema sobre curvas elípticas os algoritmos possuem complexidade exponencial [ECC97]. Assim, tais cripto-sistemas tem se mostrado atrativos uma vez que mostram ter um mesmo nível de segurança com comprimentos de chave menores, sendo bastante indicados para aplicações onde há limitação de banda e de armazenamento como no caso dos *smart cards* [ECC98], os quais não possuem processadores, possuindo apenas 8K de RAM tornando inviável uma implementação usando o RSA com 1024 bits, mas tornando possível o uso de um cripto-sistema de curvas elípticas (CCE) com 160 bits.

Nesta dissertação só serão tratados os cripto-sistemas baseados no PLDCE.

4.2 – O Problema do Logaritmo Discreto sobre Curvas Elípticas

Considere uma curva elíptica E sobre F_q, onde $q=p^m$, p primo, $m \geq 1$. O PLDCE consiste em dados dois pontos $G \in E(F_q)$[15] e $P \in E(F_q)$, tal que, G tem ordem n e $P = lG$, $0 \leq l \leq n-1$, achar o inteiro l.

Devido a operação de adição dos pontos sobre curvas elípticas, definida no capítulo 2, encontrar Q dados l e P é uma tarefa relativamente simples, porém dados P e Q, o melhor algoritmo conhecido atualmente para encontrar l possui complexidade exponencial.

4.2.1 – Alguns ataques conhecidos

Apesar das estruturas algébricas conhecidas como curvas elípticas serem estudadas a mais de 150 anos, as mesmas só vieram a ser usadas em criptografia a partir de 1985, quando foram propostos os primeiros cripto-sistemas baseados no PLDCE.

Muitas vezes alguns críticos (defensores do RSA) se baseiam no pouco tempo de introdução deste problema como base de cripto-sistemas assimétricos, para questionar a segurança dos mesmos [RY97]. Por outro lado, é um no fato que as técnicas para fatoração de inteiros sofreram grandes avanços no final da década de 70, motivados pela invenção do RSA. Além disso, muitos dos resultados obtidos para o PLD desde a década de 70 também são aplicáveis ao PLDCE [M00], portanto ambos os problemas foram estudados aproximadamente a mesma quantidade de tempo.

Outro fator que faz com que o RSA seja mais estudado é a simplicidade do algoritmo, uma vez que os algoritmos que utilizam o PLDCE como base exigem um maior conhecimento matemático por serem baseados em estruturas algébricas mais complexas. Além disso, o *RSA laboratories* tem estimulado a criptoanalise do RSA através de desafios de fatoração [RSAFC] . A empresa Certicom também vem procurando estimular o estudo dos cripto-sistemas baseados em curvas elípticas através de um desafio criptoanalitico [CECCC].

[15] $E(F_q)$ denota os pontos sobre a curva elíptica E com ambas as coordenadas em F_q incluindo o ponto $\mathcal{O}$ (ponto no infinito).

Com relação ao problema de fatoração de inteiros, na última década dois algoritmos se mostraram muito efetivos, possuindo complexidade subexponencial: O crivo quadrático [S87] e o crivo numérico [BLP94, BLZ94]. Ambos tem análogos para a resolução do PLD. Dos dois o que tem mostrado uma melhor resposta é o crivo numérico. A tabela 4.1 mostra a potência computacional estimada requerida para fatorar inteiros utilizando o crivo numérico [O95].

Comprimento de *n* (em bits)	MIPS ano
512	3×10^{4}
768	2×10^{8}
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

Tabela 4. 3 - Potência computacional requerida para fatorar o inteiro *n* usando o crivo numérico.

Já no caso do PLDCE não surgiram desenvolvimentos significantes para a sua resolução até o momento. Os novos algoritmos descobertos são aplicáveis apenas a tipos especiais de curvas e corpos os quais podem facilmente ser evitados na prática. Também tem se tentado definir a noção de "smoothness" em curvas elípticas, o que faria que os algoritmos sub-exponenciais existentes para a resolução do PLD fossem aplicados também ao PLDCE, tornando os cripto-sistemas baseados neste problema não tão atrativos já que o tamanho da chave se aproximaria ao tamanho da chave dos cripto-sistemas baseados no problema da fatoração de inteiros, os quais são cripto-sistemas mais simples e de mais fácil entendimento em relação aos cripto-sistemas sobre curvas elípticas.

O melhor algoritmo conhecido para a resolução do PLDCE é o método Pollard ρ [P78], modificado por Gallant, Lambert e Vanstone [GLV00], e por Wiener e Zuccherato [WZ99], consumindo aproximadamente $(\sqrt{\pi n})/2$ passos, onde os passos aqui considerados são as adições em curvas elípticas. Alguns resultados obtidos com este método são mostrados na tabela 4.2 [KMV00]. Van Oorschot e Wiener [OW94,OW99] mostraram como o método Pollard ρ pode ser paralelizado utilizando *r* processadores, obtendo-se uma complexidade de $(\sqrt{\pi n})/2r$ passos para a resolução do PLDCE. Para curvas elípticas *E* definidas sobre um subcorpo F_{2^l} de F_{2^m} , o método Pollard ρ

paralelizado para a resolução do PLDCE sobre $E(F_{2^m})$ pode ser acelerado obtendo-se um tempo de execução de $(\sqrt{\pi ml/m})/(2r)$ [GLV00,WZ99].

Tamanho do Corpo (em bits)	Tamanho de *n* (em bits)	$(\sqrt{\pi n})/2$	MIPS anos[16]
163	160	2^{80}	8.5×10^{11}
191	186	2^{93}	7.0×10^{15}
239	234	2^{117}	1.2×10^{23}
359	354	2^{177}	1.3×10^{41}
431	426	2^{213}	9.2×10^{51}

Tabela 4. 4 – Potência computacional necessária para computar logarítmos em curvas elípticas usando o método de Pollard ρ.

Considere que, por exemplo 1000 computadores cada um com uma taxa de 1000 MIPS estão disponíveis e $n \approx 2^{160}$, assim a partir dos dados fornecidos pela tabela 4.2 observa-se que são necessários 96.000 anos para computar o logaritmo discreto sobre curvas elípticas neste caso.

Menezes, Okamoto e Vanstone [MOV93] usaram o *Weil Pairing*[17] sobre uma curva elíptica *E* para reduzir o PLDCE ao PLD, porém essa redução só é eficiente se a curva é supersingular[18] [M93], sendo possível neste caso resolver o PLDCE com tempo de execução subexponencial.

Porém, escolhendo uma curva elíptica aleatoriamente a probabilidade desta ser supersingular é exponencialmente pequena [K91]. Portanto, o ganho aqui obtido não é significativo, não constituindo assim uma ameaça real.

Em 1997, Smart [S99] e Satoh e Araki [SA98] descobriram, independentemente, que o PLDCE utilizando as curvas elípticas anômalas[19] pode ser calculado em tempo polinomial. Assim, tais curvas são consideradas totalmente inadequadas para aplicações em criptografia.

Observa-se, deste modo, que até o momento só existem ataques efetivos com complexidade exponencial, o que significa um mesmo nível de segurança proporcionado

[16] MIPS anos representa o tempo de computação em um ano de uma máquina capaz de executar um milhão de instruções por segundo. Nesta tabela foi considerado que uma máquina com esta capacidade de execução faz $4x10^4$ adições de pontos em curvas elípticas por segundo.
[17] Para uma definição de *Weil Paring* consultar seção 5.1.1 do capítulo 5 de [M93].
[18] Uma curva elíptica *E* sobre um corpo finito F_q de característica *p* é chamada *supersingular* se, e somente se, $p|t$ onde $\#E(F_q) = q+1-t$, onde $\#E(F_q)$ denota o números de pontos da curva elíptica *E*.
[19] Uma curva elíptica *E* sobre um corpo finito F_q é chamada *anômala* se, e somente se, $\#E(F_q) = q$.

por cripto-sistemas baseados no problema de fatoração de inteiros ou no PLD, com menores comprimentos de chave, possibilitando assim maior velocidade de processamento, menor consumo de energia e redução no tamanho dos códigos fonte.

Uma análise mais aprofundada sobre o aspecto de eficiência de tais cripto-sistemas será feito de modo comparativo na seção 4.4 deste capítulo.

4.3 – Alguns cripto-sistemas baseados no Problema do Logaritmo Discreto sobre Curvas Elípticas

Como foi dito, os cripto-sistemas aqui estudados são aqueles equivalentes a cripto-sistemas do tipo DSA e ElGamal, ou seja, baseados no problema do logaritmo discreto, porém neste momento utilizando como base o PLDCE.

No caso do PLD, os parâmetros escolhidos inicialmente eram o corpo finito e a representação dos elementos neste corpo. No caso dos cripto-sistemas baseados no PLDCE, uma curva elíptica apropriada é escolhida juntamente com um ponto gerador.

Nos cripto-sistemas baseados no PLD são evitados corpos de característica 2 apesar da facilidade de implementação e melhor performance nos mesmos, pois o PLD torna-se mais fácil de resolver sobre tais corpos. Isto não ocorre nos equivalentes que utilizam curvas elípticas e portanto esse corpos são tipicamente os utilizados na implementação de cripto-sistemas baseados no PLDCE.

Muitas técnicas para geração de curvas elípticas são utilizadas e todas tendem a ser matematicamente complicadas e além disso algumas possuem certas limitações, sendo portanto este o estágio mais complicado da implementação de um cripto-sistema desse tipo. Métodos para construção de curvas elípticas são descritos em [P1363/D13].Porém, uma vez gerada a curva elíptica a mesma poderá ser usada por múltiplos usuários num sistema, já que a curva juntamente com o seu ponto gerador G serão informações públicas do cripto-sistema.

Cada usuário possui um par chave pública/secreta formado por um número k escolhido aleatoriamente, sendo este a chave secreta e um múltiplo de G, kG, sendo a chave pública.

4.3.1 – Protocolo para Troca de Chave de Diffie-Hellman utilizando Curvas Elípticas

O protocolo para troca de chave de Diffie-Hellman original pode ser adaptado para usar o grupo formado pelos pontos de uma curva elíptica.

Suponha uma curva elíptica $E(F_q)$, e um gerador G da curva $E(F_q)$ combinado e público e que dois usuários, Alice e Bob, desejam compartilhar uma chave.

Inicialmente Alice gera aleatoriamente o inteiro k_A (mantido secreto), computa o ponto k_AG e o envia para Bob, que por sua vez, similarmente, gera aleatoriamente o inteiro k_B (mantido secreto), computa o ponto k_BG e o envia para Alice. A chave compartilhada $K_{AB} = k_Ak_BG$, é obtida por Alice a partir da multiplicação do seu inteiro secreto k_A pelo ponto k_BG enviado por Bob, e de forma similar é obtida por Bob pela multiplicação do seu inteiro k_B pelo ponto k_AG enviado por Alice.

Um criptoanalista que queira ler as mensagens cifradas com a chave $K_{AB} = k_Ak_BG$, deve obtê-la sabendo G, k_AG e k_BG, mas não k_A ou k_B. Assim, a obtenção da chave por um usuário não autorizado implica na resolução do PLDCE.

Exemplo 4.1 – Considere a curva elíptica $y^2 = x^3 + x + 1$ sobre F_{23} e os pontos da mesma listados abaixo.

O	(0,1)	(0,22)	(1,7)	(1,16)	(3,10)	(3,13)
(4,0)	(5,4)	(5,19)	(6,4)	(6,19)	(7,11)	(7,12)
(9,7)	(9,16)	(11,3)	(11,20)	(12,4)	(12,19)	(13,7)
(13,16)	(17,3)	(17,20)	(18,3)	(18,20)	(19,05)	(19,18)

E sabendo também que G=(0,1) é um dos geradores.

Suponha que Alice obtém aleatoriamente a chave secreta k_A=4, e consequentemente a chave pública $y_A = 4 \cdot G = 4 \cdot (0,1) \Rightarrow y_A = (0,4)$, de modo similar Bob obtém como chave secreta k_B=9 e chave pública $y_B = 9 \cdot G = 9 \cdot (0,1) \Rightarrow y_B = (0,9)$.

Assim, a fim de compartilhar uma chave Alice envia para Bob y_A que faz $K = k_B y_A = 9 \cdot (0,4) = (0,36)$, Bob por sua vez envia y_B para Alice que faz $K' = k_A y_B = 4 \cdot (0,9) = (0,36)$.

Nota-se assim que K=K' e portanto a chave foi compartilhada como desejado.

□

4.3.2 – Cripto-sistema de Taher ElGamal para Curvas Elípticas

O cripto-sistema original de ElGamal também é facilmente modificado a fim de utilizar o PLDCE ao invés do PLD como base.

Suponha agora que Bob deseja mandar uma mensagem $M \in E$ para Alice. Usando o protocolo de troca de chave de Diffie-Hellman Alice e Bob já trocaram $k_A G$ e $k_B G$. Bob escolhe um outro inteiro aleatório secreto l, e manda para Alice o texto cifrado C formado pelo par de pontos (lG, $M+l(k_A G)$).

Para decifrar a mensagem Alice multiplica lG por sua chave secreta k_A e então subtrai o resultando do segundo ponto do par recebido de Bob.

4.3.3 – ECDSA (Eliptic Curve Digital Signature Algorithm)

O ECDSA é um algoritmo criptográfico aprovado pelo FIPS para geração e verificação de assinaturas digitais e é o análogo ao DSA usando neste caso curvas elípticas.

A descrição do ECDSA é encontrada em [JM99, ANSI X9.62] e as curvas elípticas recomendadas se encontram no apêndice 6 de [FIPS 186-2].

Agora será dada uma breve descrição do algoritmo. A mesma consistirá três partes : *geração da chave*, *geração da assinatura* e *verificação da assinatura*.

Geração da Chave: Considere os parâmetros do sistema: a curva elíptica E definida sobre F_q, e P um ponto de ordem prima n (os mesmos são de conhecimento público). Supõe-se por simplicidade que q é primo, embora possa ser adaptado facilmente para potências de primos. Cada usuário A :

1. Seleciona um inteiro aleatório d no intervalo [1, n-1];
2. Computa $Q = dP$;

3. A chave pública de *A* é *Q*; e a chave privada de *A* é *d*.

Geração da Assinatura: Para assinar a mensagem *m*, *A*:

1. Seleciona um inteiro aleatório *k* no intervalo [1, *n*-1];
2. Computa $kP = (x_1,y_1)$ e $r = x_1 \bmod n$ (onde x_1 é considerado um inteiro entre 0 e $q-1$). Se $r = 0$ volta para o passo 1;
3. Computa $k^{-1} \bmod n$;
4. Computa $s = k^{-1}\{h(m) + dr\} \bmod n$, onde *h* é o *Secure Hash Algorithm* (SHA-1 [NIST95]). Se $s = 0$, então volta para o passo 1;
5. A assinatura para a mensagem *m* é o par de inteiros (*r*,*s*).

Verificação da Assinatura: Para verificar a assinatura (*r*,*s*) de *A*, o usuário *B* deve:

1. Obter uma cópia autenticada da chave pública *Q* de *A*;
2. Verificar se *r* e *s* são inteiros no intervalo [1, *n*-1];
3. Computar $w = s^{-1} \bmod n$ e $h(m)$;
4. Computar $u_1 = h(m)w \bmod n$ e $u_2 = rw \bmod n$;
5. Computar $u_1P + u_2Q = (x_0, y_0)$ e $v = x_0 \bmod n$;
6. A assinatura é considerada autêntica e portanto aceita, se e só se $v = r$.

A única diferença significante entre o DSA e o ECDSA é a geração de *r* que, no DSA consiste em tomar um elemento aleatório $(\alpha^k \bmod p)$ e reduzir modulo *q* (*q* é um primo divisor de *p*-1 com comprimento 160 bits, e α é um elemento de ordem *q* em F_p^*). Por sua vez, no ECDSA o inteiro $r \in [1, n-1]$ é gerado tomando a coordenada *x* do ponto aleatório *kP* e reduzindo modulo *n*.

Para que se tenha o mesmo nível de segurança do DSA, o parâmetro *n* no ECDSA deve possuir 160 bits de comprimento, assim tanto no DSA quanto no ECDSA a assinatura possui 320 bits de comprimento.

4.3.4 – Outros Algoritmos e Protocolos

Como já foi colocado anteriormente, todos os cripto-sistemas baseados no PLD tem um análogo elíptico, como por exemplo vários esquemas de assinaturas, protocolos para trocas de chave, e.g., KEA [NSA98] e MQV [MQV95], sugeridos para padronização.

Destaca-se o MQV que proporciona negociação autenticada de chaves (como a que se obtém através do protocolo de Diffie-Hellman usado com mensagens assinadas), este protocolo possui como vantagem a utilização de um número reduzido de operações elípticas e a resistência a uma grande quantidade de ataques [MB98]. Um fato curioso é que ele foi concebido desde o início considerando o PLDCE como base, ao contrário da maioria dos cripto-sitemas deste tipo.

4.4 – Eficiência dos Cripto-sistemas baseados no PLDCE (Análise Comparativa)

A análise de cripto-sistemas de chave pública deve levar em conta basicamente três fatores:

- **Sobrecarga Computacional** : Quantidade de computação requerida para fazer as operações de geração de chaves, cifragem , decifragem, assinatura e verificação de assinatura.
- **Tamanho da chave e dos parâmetros do cripto-sistema** – Quantidade de bits das chaves e quaisquer outros parâmetros requeridos pelo cripto-sistema, que devem ser armazenados pelo sistema.
- **Largura de Banda** – Quantidade de bits necessária no transporte do texto cifrado ou na assinatura.

As comparações serão feitas considerando cripto-sistemas que proporcionem o mesmo nível de segurança, como por exemplo o cripto-sistema de curva elíptica (CCE) com 161 bits, o RSA com 1024 bits e o DSA também com 1024 bits.

Assumindo-se que um cripto-sistema simétrico leva o mesmo tempo que uma multiplicação escalar[20] , i. e., a chave de um CCE deve ter o dobro do comprimento da chave de cripto-sistema simétrico a fim de ter o mesmo nível de segurança [J99] Por exemplo, o AES com chave de 128 bits terá como equivalente em nível de segurança um CCE com chave de comprimento 256 bits.

[20] Defini-se multiplicação escalar de um ponto P por um inteiro k, como sendo $kP=P+P+...+P$ (k termos).

Como especificado no X9.30 DSA, para o AES de 128 bits é apropriado, a fim de obter o mesmo nível de segurança, usar um *p* de comprimento 3072 bits no DSA. Um ataque ao DSA sobre um corpo de característica *p* é considerado ligeiramente mais difícil que um ataque ao RSA com módulo *n*, mas por simplicidade, assumi-se a mesma dificuldade.

A seguir é observada uma tabela com os comprimentos de chaves (bits) de cripto-sistema simétrico (ex: AES), CCE, DSA e RSA com mesmo nível de segurança.

Cripto-sistema simétrico	56	80	112	128	192	256
RSA *n*	512	1024	2048	3072	7680	15360
DSA *p*[21]	512	1024	2048	3072	7680	15360
DSA *q*[22]	112	160	224	256	384	512
CCE *n*	112	160	224	256	384	512

Tabela 4. 5 - Equivalência aproximada de chaves em bits para os melhores ataques gerais conhecidos.

O mesmo nível de segurança com comprimentos de chaves menores em cripto-sistemas baseados no PLDCE é possível neste caso devido ao uso da técnica conhecida como *compressão de pontos*.

A compressão de pontos é uma técnica que tira proveito da simetria da curva em relação a abscissa a fim de representar um ponto da curva de maneira mais compacta. Deste modo, para representar um ponto da curva basta utilizar a abscissa *x* e um único bit para distinguir entre os valores da ordenada *y* que torna (*x*, *y*) uma solução da equação da curva.

Sobrecarga Computacional

Em todos os cripto-sistemas procedimentos podem ser feitos a fim de economizar recursos computacionais. Por exemplo, no RSA pode-se empregar um pequeno expoente público *e* a fim de acelerar os procedimentos de cifragem e verificação de assinatura. Porém, observa-se que tal procedimento pode comprometer a segurança do cripto-sistema [BV98].

Tanto no CCE quanto no DSA, grande parte dos procedimentos de geração de assinatura e transformações de cifragem podem ser pré-computados.

[21] O parâmetro *p* primo do DSA é a característica do corpo finito sobre o qual as operações ocorrem.
[22] O parâmetro *q* primo do DSA é a ordem do subgrupo gerado pelo gerador *g*.

Ao se utilizar corpos do tipo F_{2^m} a aritmética modular passa a ser feita mais rapidamente. Com as implementações atuais os CCE são aproximadamente 10 vezes mais rápidos que o RSA e DSA.

Tamanho da Chave

A figura 4.1 compara o tempo requerido para quebrar um CCE com o tempo requerido para quebrar o RSA ou o DSA com relação a vários e distintos comprimentos de chave e usando o melhor algoritmo geral conhecido (crivo numérico).

Um tempo da ordem de 10^{12} representa um nível de segurança razoável atualmente por necessitar de uma quantidade razoável de computadores por todo o planeta trabalhando uma quantidade razoável de tempo, como foi visto baseado na tabela 4.2, são necessários 1000 computadores com uma taxa de 1000 MIPS ano processando por 96000 anos a fim de quebrar um CCE com chave de comprimento 160 bits. Assim, na figura 4.1 observa-se que para conseguir tal nível de segurança é necessário usar o RSA ou o DSA com uma chave de comprimento 1024 bits ou um CCE com uma chave de comprimento 160 bits.

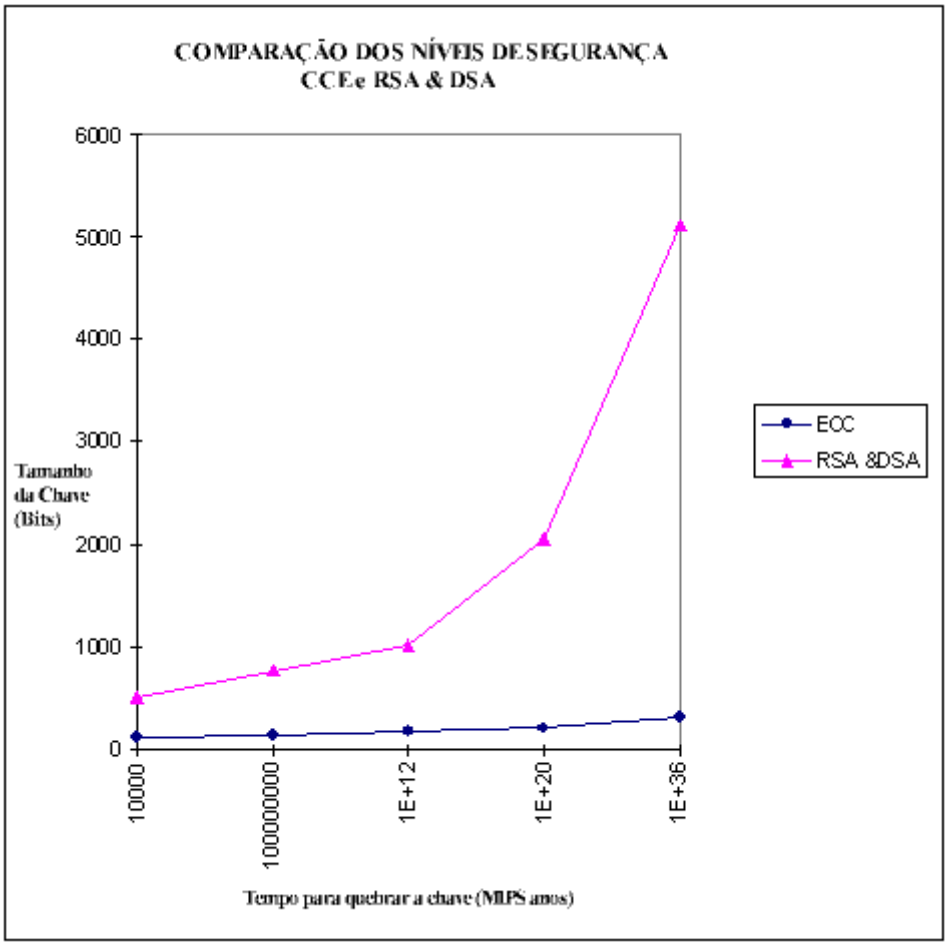


Figura 4. 2 - Comparação dos níveis de segurança.

Largura de Banda

Todos os três cripto-sistemas tem largura de banda similar quando se trata da cifragem ou assinatura de mensagens longas. Porém, o caso onde se trabalha sobre mensagens curtas é de grande importância já que os cripto-sistemas assimétricos são usados com muita frequência no transporte de chaves de cripto-sistemas simétricos.

Afim de comparar, considere que cada um dos cripto-sistemas é usado para assinar uma mensagem de 2000 bits (Tabela 4.4) e para cifrar uma mensagem de 100 bits (Tabela 4.5).

	Comprimento da Assinatura (bits)
RSA	1024
DSA	320
CCE	320

Tabela 4. 6 - Comprimento de assinatura para uma mensagem de 2000 bits

	Comprimento da Mensagem Cifrada (bits)
RSA	1024
ElGamal	2048
CCE	321

Tabela 4. 7 - Comprimentos de texto cifrado considerando um texto claro de 100 bits.

Observa-se a partir das tabelas 4.4 e 4.5 que os CCE proporcionam grande economia quanto ao uso da largura de banda.

Conclui-se portanto que com os atuais parâmetros os CCE se mostram mais eficientes em todos os aspectos considerados (sobrecarga computacional, tamanho das chaves e parâmetros e largura de banda utilizada) que os outros tipos de cripto-sistemas assimétricos. O que significa na prática, maior velocidade, menor energia consumida e redução no código fonte.

Recomendações

O NIST (US National Institute of Standards and Technology) recomenda chaves de comprimento 256, 384 e 512 bits para CCE obtendo mesmo nível de segurança que o AES[23] com 128 , 192 e 256 bits, respectivamente. O que para o RSA significaria comprimentos de chave de 3072, 7680 e 15630 bits, como foi observado anteriormente na tabela 4.3.

4.5 – Aplicações

Como foi visto até o momento, todos os cripto-sistemas baseados no PLD podem ser implementados utilizando como base o PLDCE. Na seção anterior observou-se a eficiência destes cripto-sistemas comparados com os outros do mesmo tipo.

Assim, tal cripto-sistema pode ser utilizado em qualquer uma das muitas aplicações dos cripto-sistemas assimétricos, mostrando porém maior eficiência.

Um bom exemplo de aplicação do uso de CCE é em mecanismos sem fio, já que os mesmos possuem potência computacional, memória RAM, potência, espaço para armazenamento de chave, de código fonte e de certificado limitados, e os CCE oferecem benefícios em todas essas áreas. Chaves menores reduzem o espaço de armazenamento

para chaves e certificados, e acelera a execução do protocolo. A redução do consumo de potência ocorre devido ao processamento eficiente assim como por conta da redução da quantidade de dados a ser transmitida.

Mecanismos como *tokens* and *smart cards* usados em criptografia devem ser capazes de proteger operações criptográficas e armazenar chaves privadas. Desse modo, a fim de serem mecanismos práticos devem ser pequenos, leves e baratos. A solução ideal parece ser os smart cards que além de facilmente transportáveis podem atingir custo de menos de US$ 10,00. Tais mecanismos são extremamente limitados ao que se trata de potência de processamento, armazenamento de parâmetros e espaço para o código fonte. Possuem também saída/entrada serial lenta, encarecendo a largura de banda em alguns casos.

No próximo capítulo será visto mais uma aplicação. Neste caso a CCE faz parte da construção de um padrão para comunicação móvel.

4.6 – Padronização

O ECDSA foi adotado em Janeiro 1999 como padrão oficial pelo ANSI (American National Standards Institute). O grupo de trabalho ANSI X9 (Serviços Financeiros) está construindo a padronização para os protocolos de troca e transporte de chave usando curvas elípticas.

O uso de curvas elípticas em criptografia assimétrica também faz parte do padrão P1363 (Especificações Padrão para Criptografia de Chave Pública) que está sendo construído. Neste documento são considerados também cripto-sistemas assimétricos baseados no problema da fatoração de inteiros e no PLD. Versões atualizadas são encontradas em http://grouper.ieee.org/groups/1363/.

O Protocolo para determinação de chave OAKLEY do IETF (*Internet Engineering Task Force*) descreve um protocolo para troca de chave que é uma variante de Diffie-Hellman. Um rascunho do documento pode ser encontrado em http://www.ietf.cnri.reston.va.us/.

O padrão em construção ISO/IEC 15946 especifica varias técnicas criptográficas baseadas em curvas elípticas, incluindo protocolos para assinatura digital, criptografia de chave pública e troca de chave.

Esses e outros padrões tem sido construídos levando em conta os cripto-sistemas baseados no PLDCE. Ao serem aprovados pelos respectivos órgãos espera-se que os cripto-sistemas baseados em curvas elípticas passem a ser amplamente usados pelos provedores de segurança de informação.

4.7 – Considerações Finais

[23] Nova cifra de bloco que susbstitui o DES como padrão nos EUA.

Foi introduzido neste capítulo um novo tipo de cripto-sistema assimétrico baseado em um outro problema matemático. Observou-se que o mesmo se mostra bastante vantajoso sobre os outros cripto-sistemas assimétricos e possui muitas aplicações.

Observou-se também que a complexa matemática envolvida dificulta um pouco o seu desenvolvimento, porém isto tende a mudar por todas as considerações feitas aqui e pelo interesse mostrado na padronização de tais cripto-sistemas.

Referências:

[ANSI X9.62] – *Public Key Cryptography for the Finacial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, 1998.

http://webstore.ansi.org/ansidocstore/dept.asp

[BLP94] – J. P. Buhler, H. W. Lenstra e C. Pomerance, *The Development of Number Field Sieve*, Lectures Notes in Computer Science, Vol. 1554, Springer-Verlag, 1994.

[BLZ94] – J. Buchmana, J. Loho e J. Zayer, *An Implementation of the General Number Field Sieve*, Advances in Cryptology Crypto '93, pp. 159-166, Springer-Verlag, 1994.

[BV98] – D. Boneh e R. Venkatesan, *Breaking RSA may not be Equivalent to Factoring*, Proceedings of Eurocrypt '98, pp. 59-71, 1998.

[D94] N. Demytko, *A New Elliptic Curve Based Analogue of RSA*, Advances in Cryptology – Eurocrypt'93, pp. 40-49, Springer-Verlag, 1994.

[ECC97] – A Certicom White Paper, *The Elliptic Curve Cryptosystem*, Abril 1997.
www.certicom.com

[ECC98] – A Certicom White Paper, *The Elliptic Curve Cryptosystem for Smart Cards – The Seventh in a Series of ECC White Papers*, Maio 1998.
www.certicom.com

[FIPS 186-2] – FIPS PUB 186-2 – Federal Publication Processing Standards Publication – *Digital Signature Standard (DSS)*, 27 Janeiro 2000.

http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf

[GLV00] – R. Gallant, R. Lambert e S. Vanstone, *Improving the Parallelized Pollard Lambda Search on Binary Anomalous Curves*, Mathematics of Computation, 69, pp. 1685-1697, 2000.

[J99] – D. B. Johnson, *ECC, Future Resiliency and High Security Systems*, Junho 1999.

[JM99] – D. Johnson e A. Menezes, *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, Technical Report CORR 99-31, Depto. of C&O, University of Waterloo, Canada, 23 de Agosto de 1999.
http://www.cacr.math.uwaterloo.ca

[K87] – N. Koblitz,, *Elliptic Curve Cryptosystems*, Mathematics of Computation, Vol. 48, No. 177, pp. 203-209, Janeiro 1987.

[K91] – N. Koblitz, *Elliptic Curve Implementatio of Zero-knowledge Blobs*, Journal of Cryptology , 4, pp. 207-213, 1991.

[KMOV92] – K. Koyama , U. M. Maurer, T. Okamoto e S. A. Vanstone, *New Public-Key Schemes Based on Elliptic Curves over the Ring* Z_n, Advances in Cryptology – Crypto'92, pp. 40-49, Springer-Verlag, 1992.

[KMV00] – N. Koblitz, A. Menezes e S. Vanstone, *The State of Elliptic Curve Cryptography*, Designs, Codes and Cryptography, volume 19, Number 2/3, pp. 173-193, Março 2000.

[M86] – V. S. Miller,, *Use of Elliptic Curves in Criptography*, Advances in Cryptology, Proceedings of CRYPTO'85, Springer Verlag Lecture Notes in Computer Science 218, pp. 417-426, 1986.

[M93] – A. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.

[MB98] – A. J. Menezes e S. Blake-Wilson, *Authenticated Diffie-Hellman Key Agreement Protocols*, Workshop on Selected Areas in Cryptography (SAC' 98), Lectures Notes in Computer Science, 1556, pp. 339-361, 1998.

[MOV93] – A. Menezes, T. Okamoto e S. Vanstone , *Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field*, IEEE Transactions in Information Theory, 39, pp. 1639-1646, 1993.

[MQV95] – Menezes, A. J., Qu, M. e Vanstone, S., *Some New Key Agreement Protocols Providing Mutual Implicit Authentication*, Workshop on Selected Areas in Cryptography (SAC' 95), pp. 22-32, 1995.

[NSA] – National Security Agency (NSA), *SKIPJACK and KEA algorithm especification*, version 2.0, 1998.

http://crsc.nist.gov/encryption/skipjack-kea.htm

[NIST95] - National Institute of Standards and Technology (NIST), *FIPS Publication 180-1: Secure Hash Standard (SHS)*, 17 de Abril de 1995.

http://www.itl.nist.gov/fipspubs/fip180-1.htm

[O95] – A. Odlyzko, *The Future of Integer Factorization*, CryptoBytes – The Technical Newsletter of RSA Laboratories, volume 1, number 2, pp. 5-12, 1995.

www.rsa.com

[OW94] – P. van Oorschot e M. Wiener, *Parallel Colision Search with Application to Hash Functions and Discrete Logarithms*, Proceedings of the 2nd ACM Conference on Computer and Communications Security, Fairfax, Virginia, pp. 210-218, 2-4 de Novembro de 1994.

[OW99] – P. van Oorschot e M. Wiener, *Parallel Colision Search with Cryptanalytic Applications*, Journal of Cryptology, 12, pp. 1-28, 1999.

[P78] – J. Pollard, *Monte Carlo Methods for Index Computation mod p*, Mathematics of Computation, 32, pp. 918-924, 1978.

[P1363/D13] – IEEE P1363/D13 (Draft Version 13) – *Standard Specifications for Public Key Cryptography*, 12 Novembro de 1999.

[RY97] – M. J. B. Robshaw e Y. L. Yin, *Elliptic Curves Cryptosystems*, An RSA Laboratories Technical Note, Revisada 27 Junho de 1997.
http://www.rsa.com/rsalabs/ecc/html/elliptic_curve.html

[S87] – R. D. Silverman, *The Multiple Polynomial Quadratic Sieve*, Mathematics of Computation, 48, pp. 329-339, 1987.

[S99] – N. Smart , *The Discrete Logarithm Problem on Elliptic Curve of Trace One, Journal of Cryptology*, 12, pp. 193-196, 1999.

[SA98] – T. Satoh e K. Araki, *Fermat Quotients and the Polynomial Time Discrete Log Algorithm for Anomalous Elliptic Curves*, Commentarii Mathematici Universitatis Sancti Pauli, 47, pp. 81-92, 1998.

[WZ99] – M. Wiener e R. Zuccherato, *Fast Attacks on Elliptic Curves Cryptosystems*, Fifth Annual Workshop on Selected Areas in Cryptography – SAC ’98, Lectures Notes in Computer Science, Springer-Verlag (1999).

Páginas na WEB

[RSAFC] – RSA Factoring Challenge
http://www.rsasecurity.com/rsalabs/challenges/factoring/index.html

[CECCC] – Certicom ECC Challenge
http://www.certicom.com/research/ecc_challenge.html

5

IP Móvel

Neste capítulo o protocolo utilizado em computação móvel é introduzido e aspectos ligados a segurança do mesmo são analisados.

5.1 – Introdução

Com a tendência atual de se ter equipamentos como *laptops*, *palmtops* e PDAs (*Portable Data Assistants*) menores e mais baratos, os usuários passaram a querer se comunicar através da Internet e com a sua *rede local*, mesmo estando em movimento, isto é, antes o acesso que só era possível através de um ponto fixo (casa, trabalho, universidade,...) seria feito de maneira móvel.

A fim de entender as diferenças da já conhecida conexão via IP e as futuras possibilidades, considere o desenvolvimento na área da telefonia nos últimos 20 anos. Uma transição análoga na área de redes, da dependência de pontos fixos à flexibilidade proporcionada pela mobilidade só está começando.

Computação e *networking* móveis não devem ser confundidos com computação e rede portátil existentes hoje. Em rede móvel, as atividades não são interrompidas quando o computador muda seu ponto de ligação com a Internet. Em vez disto, toda a reconexão necessária ocorre automaticamente e sem interatividade.

A computação verdadeiramente móvel oferece muitas vantagens, como por exemplo, possuir um acesso confiável à Internet a qualquer momento e em qualquer lugar, possibilitando liberdade de movimento sem ser necessário estar preso a um *desktop* a fim de se conectar à mesma. Tal característica pode ser comparada a liberdade proporcionada pelos telefones celulares.

A evolução na mobilidade em rede difere da ocorrida na área de telefonia em alguns aspectos bastante importantes. Os pontos extremos em telefonia são tipicamente humanos; no caso de aplicações computacionais as interações são feitas entre máquinas sem a intervenção humana. Exemplos óbvios de tais mecanismos são mecanismos computacionais móveis utilizados em aviões, navios e automóveis [P98].

Outra diferença está relacionada à adoção de tais sistemas. Muitos anos foram necessários para que os telefones celulares se tornassem mais leves e mais baratos, para que se mostrassem práticos. Por outro lado, mecanismos como *Palmtops* e PDAs já foram aceitos pelos usuários, assim a computação móvel deve se tornar popular mais rapidamente que a telefonia móvel.

Todavia, ainda existem alguns obstáculos técnicos a serem superados antes que a rede móvel possa se tornar amplamente utilizada. O mais fundamental é com relação ao *Internet Protocol* (IP) [P81], o protocolo utilizado atualmente para conectar as redes, rotear pacotes para os seus destinos de acordo com o endereço IP.

5.2 – Funcionamento do IP Móvel

Nesta seção será dada uma noção do funcionamento do IP Móvel. A terminologia utilizada se encontra no apêndice C.

O protocolo IP faz o roteamento de pacotes na rede da fonte ao destino, permitindo que os roteadores passem adiante os pacotes seguindo tabelas de roteamento existentes em cada um dos roteadores pelo qual o pacote passa até chegar ao seu destino. Tipicamente, as tabelas de roteamento mantém o *next-hop* de cada endereço IP, de acordo com o número da rede ao qual o endereço IP está conectado. O número da rede é derivado do número IP retirando-se alguns dos bits de menor ordem. Desta forma o endereço IP normalmente trás consigo informações que especificam a localização do nó. Este protocolo foi projetado de forma a não suportar mobilidade uma vez que a localização do nó na rede mantém-se inalterada sempre, e além disso, como já foi dito, um número IP identifica uma rede particular. Para que a mobilidade do servidor se torne possível utilizando o protocolo IP seria necessário que o nó mudasse o endereço IP a cada momento que mudasse a sua localização. Tal alternativa não é aceitável pois é impossível ao nó manter-se conectado uma vez que para manter as conexões existentes da camada de transporte é necessário que o nó ao se movimentar mantenha o mesmo endereço IP. Por exemplo, no protocolo TCP as conexões são indexadas por um conjunto de quatro números consistindo dos endereços IP e os números de porta da fonte e do destino. A mudança de um desses quatro números causa a perda ou interrupção da conexão. Por outro lado, o correto recebimento dos pacotes na atual localização do nó móvel depende do número de rede contido no endereço IP do mesmo, o qual sofreria mudanças dependendo do ponto de localização no momento.

Com o intuito de resolver esse problema surgiu a idéia de IP móvel. No projeto deste protocolo cinco características devem ser consideradas:

1) Um nó móvel deve ser capaz de se comunicar com outros nós depois de mudar sua localização na Internet, sem modificar seu número IP;

2) Um nó móvel deve ser capaz de se comunicar com outros nós que não implementam IP móvel;

3) Todas as mensagens usadas para transmitir informação para outro nó sobre a localização de um nó móvel devem ser autenticadas a fim de protegê-lo contra ataques de *redirecionamento remoto*;

4) O enlace ao qual o nó móvel está diretamente conectado à Internet pode freqüentemente ser um enlace sem fio. Desta forma tal enlace apresentará uma largura de banda substancialmente menor e será mais suscetível a erros que uma conexão via cabo. Além disso, nós móveis utilizam freqüentemente baterias como fonte de energia, e portanto a minimização no consumo de energia é muito importante. Assim, o número de mensagens administrativas mandadas pelo enlace ao qual um nó móvel está conectado à Internet deve ser minimizado, e as mensagens devem ser mantidas tão curtas quanto possíveis;

5) O IP móvel não deve colocar restrições adicionais na designação do endereço IP. Isto é, um nó móvel pode ter designado um endereço IP pela organização proprietária da máquina, assim como é feito com qualquer outro mecanismo de protocolo administrado pela organização. Em particular o endereço não tem que pertencer a qualquer conjunto globalmente restrito de endereços.

O IP móvel foi criado principalmente com o intuito de resolver o problema citado anteriormente, além de procurar atender as cinco características citadas. Sobre o problema exposto o mesmo podem ser resolvidos através da utilização de dois endereços IP pelo nó móvel: o fixo e o móvel. O *endereço fixo* é o endereço estático, usado, por exemplo, para identificar conexões TCP. Já o móvel muda a cada nova localização do nó, indicando o número da rede e portanto a localização do nó móvel com respeito à topologia da rede. O fixo faz com que o nó móvel pareça sempre receber dados na sua rede local. Quando o nó móvel não esta localizado na sua rede local diz-se que o mesmo se encontra na *rede externa*, assim é necessário a presença de um *agente local* que recebe todos os pacotes destinados ao nó móvel e os organiza para enviar à sua atual localização.

Sempre que o nó móvel troca de lugar registra o seu novo *endereço móvel* com o seu roteador local. Para enviar um pacote para um nó móvel a partir da sua rede local, o agente local envia o pacote da rede local para o endereço móvel. A nova entrega requer que o pacote seja modificado de modo que o endereço móvel apareça como o endereço IP do destino. Esta modificação pode ser entendida como uma transformação do pacote ou, mais especificamente, um redirecionamento. Quando o pacote chega ao endereço móvel, a transformação reversa é aplicada de modo que o pacote mais uma vez pareça ter o endereço fixo como endereço IP do destino. Quando o pacote chega ao nó móvel, endereçado ao seu endereço fixo, é processado corretamente pelo TCP ou qualquer outro protocolo de alto nível que o recebe a partir da camada de processamento IP do nó móvel.

No IP móvel o agente local redireciona pacotes da rede local para o endereço móvel construindo um novo cabeçalho para o IP o qual contém o endereço móvel do nó móvel como endereço IP do destino. Este novo cabeçalho encapsula o pacote original, fazendo com que o *endereço fixo* do nó móvel tenha nenhum efeito no roteamento do pacote encapsulado até que o mesmo chegue ao *endereço móvel*. Tal processo de encapsulamento é chamado *tunelamento*, o que sugere que o pacote "se esconde" através da Internet, evitando os efeitos usuais do roteamento IP.

Na Figura 5.1 podem ser observados duas *redes externas*, *B* e *C*, com *agentes externos*; duas *redes locais A* e *D*, com agentes locais; e nós móveis que estão conectados a

várias redes externas através de conexão por radio e infravermelho. Os túneis vão dos agentes locais, através da Internet, chegando finalmente aos agentes externos para o destino final.

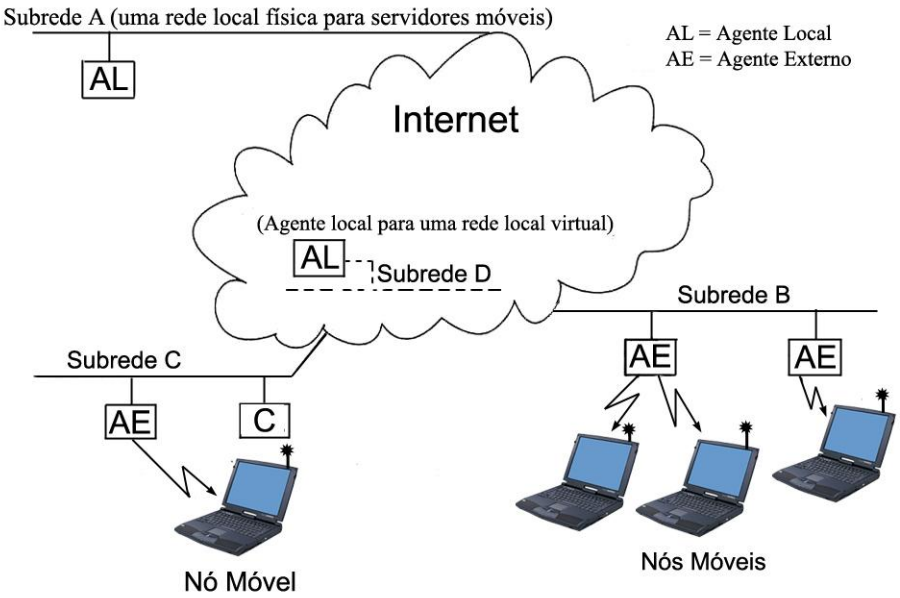


Figura 5. 15 - IP móvel.

O IP móvel portanto é melhor entendido como a cooperação entre três processos:

- Descobrindo o endereço móvel
- Registrando o endereço móvel
- Fazendo tunelamento até o endereço móvel

Descobrindo o Endereço móvel

O IP móvel proporciona duas maneiras de adquirir um endereço móvel:

1) Um endereço móvel do agente externo é um endereço móvel provido por um *agente externo* através das suas mensagens de anúncio. Neste caso o endereço móvel é um endereço IP do roteador externo. Neste modo, o roteador externo é o ponto final do túnel, e ao receber os datagramas tunelados, descapsula os mesmos e envia os datagramas resultantes para o nó móvel.

2) Um *endereço móvel colocado* é um endereço móvel adquirido pelo nó móvel como um endereço IP local através de alguns meios externos, que o nó móvel então associa com uma das suas próprias interfaces de rede. O endereço pode ser obtido dinamicamente como um endereço temporário pelo nó móvel, como no DHCP (*Dynamic Host Configuration Protocol*) [D97, AD97], ou pode pertencer

ao nó móvel como um endereço de longo termo para utilizar apenas enquanto visitando alguma *rede externa*.

O processo 1) não modifica os campos originais de anúncio dos roteadores existentes, mas simplesmente estende os mesmos a fim de associar funções de mobilidade. Deste modo, o roteador passa a ter não só as informações usuais como também informações adicionais sobre um ou mais endereços móveis. Quando os anúncios do roteador são estendidos contendo também os endereços móveis necessários são chamados *anúncio de agente*. Agentes locais e *agentes externos* tipicamente fazem *broadcast* de anúncios de agente em intervalos regulares (eg. uma vez a cada segundo ou a cada poucos segundos). Se um nó móvel necessita de um endereço móvel e não quer esperar pelo anúncio periódico, o nó móvel pode fazer um *broadcast*[24] ou um *multicast*[25] solicitando e sendo respondido a seguir por um *agente externo* ou *agente local* que receber a solicitação.

Assim as tarefas do *anúncio de agente* são

- Permitir a detecção dos roteadores de mobilidade (roteador local ou roteador externo);
- listar um ou mais endereços móveis disponíveis;
- informar o nó móvel sobre características especiais providas pelos roteadores externos, por exemplo, outras técnicas de encapsulamento.
- permitir que os nós móveis determinarem o número da rede e o status do seu enlace à Internet; e
- permitir que o nó móvel saiba se um operador é um agente local ou externo, e portanto se ele está numa rede local ou externa.

Os nós móveis usam solicitações como definidas em [D91] para detectar qualquer mudança no conjunto de agentes de mobilidade disponíveis na atual localização do nó móvel (Em IP móvel isto é denominado solicitação de operador). Se os anúncios não são mais detectados pelo agente externo que ofereceu previamente um endereço móvel ao nó móvel, o mesmo deve presumir que o operador não está mais ao alcance da interface de rede do nó móvel em questão. Nesta situação, o nó móvel deve começar a procurar um novo endereço móvel, ou possivelmente usar um endereço móvel conhecido dos anúncios que ainda esta recebendo. O nó móvel pode também preferir esperar o próximo anúncio se o mesmo não tiver recebido qualquer endereço móvel anunciado recentemente, ou ainda pode mandar um solicitação de operador.

Registrando o Endereço móvel

Uma vez que um nó móvel obteve um endereço móvel, seu agente local passa a ter conhecimento do mesmo, adicionando o mesmo a sua lista após ser registrado.

[24] Um *broadcast* é um processo pelo qual pacotes são endereçados a todos os destinos utilizando um código especial no campo de endereço. [T96]

[25] Alguns sistemas de broadcast também permitem transmissão para um subconjunto de máquinas, esse processo é conhecido como *multicasting*. [T96]

A figura 5.2, a seguir, mostra o processo de registro definido pelo IP móvel. O processo é iniciado quando o nó móvel, possivelmente com o auxílio de um agente externo, manda um pedido de registro com a informação do endereço móvel. Quando o agente local recebe o pedido, ele geralmente adiciona a informação necessária à sua tabela de roteamento, aprova o pedido, e manda a resposta sobre o pedido de registro de volta ao nó móvel.

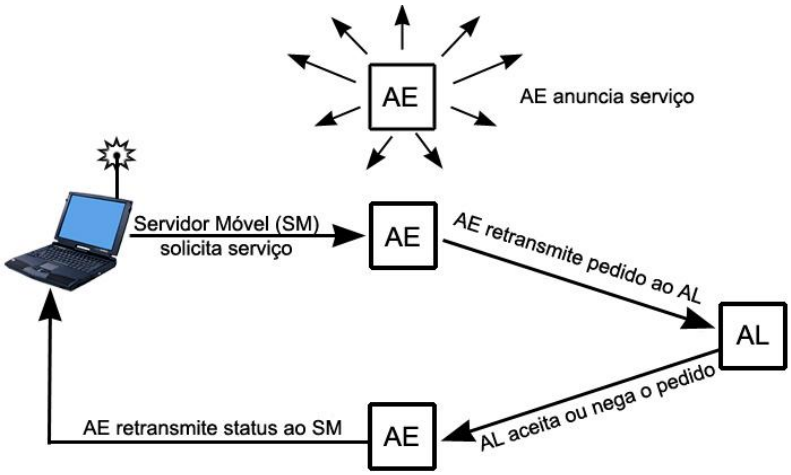


Figura 5. 16 - Processo de Registro de Endereço móvel em IP Móvel (AE=Agente externo, AL= Agente local, SM = Servidor Móvel).

Pedidos de registro contém parâmetros e sinalizadores que caracterizam o túnel pelo qual o agente local irá enviar os pacotes para o endereço móvel. Os túneis podem ser construídos de várias maneiras [P96a, P96b]. Quando o agente local aceita o pedido, ele começa a associar o endereço fixo do nó móvel ao endereço móvel, e mantém esta associação até que o *tempo de vida do registro* expire. A tripla que contém o endereço fixo, endereço móvel e o tempo de vida do registro é chamado *vínculo* do nó móvel. Um pedido de registro pode ser considerado uma *atualização de vínculo* mandado através do nó móvel. Tal atualização é um exemplo de um *redirecionamento remoto*, porque é mandado remotamente ao agente local a fim de alterar a tabela de roteamento do mesmo.

A necessidade de autenticar a informação do registro tem desempenhado um papel importante na determinação do parâmetros aceitáveis no projeto do IP móvel. Aspectos ligados a como registrar o endereço móvel de forma segura serão mostrados na seção 5.3.1.

Tunelamento para o Endereço móvel

A figura 5.3 mostra o processo de tunelamento no IP móvel. O mecanismo de encapsulamento padrão que deve ser permitido por todos os agentes de mobilidade usando IP móvel é o *IP-within–IP* [P96a]. Usando, *IP-within–IP*, o agente local, início do túnel, insere um novo cabeçalho IP (cabeçalho do túnel) na frente do cabeçalho de qualquer datagrama endereçado ao endereço fixo do nó móvel. O novo cabeçalho usa o *endereço móvel* do nó móvel como endereço IP destino, fim do túnel. O cabeçalho usa 4 como o número do mais alto nível do protocolo, indicando que o próximo cabeçalho do protocolo é novamente cabeçalho IP. No *IP-within-IP* o cabeçalho original é completamente preservado como a primeira parte do cabeçalho do túnel. Portanto, para recuperar o pacote original, o agente externo tem simplesmente que eliminar o cabeçalho do túnel e enviar o restante para o nó móvel.

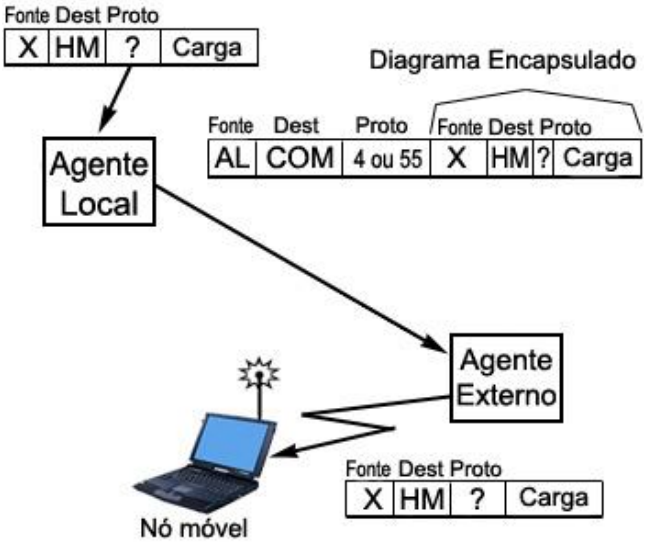


Figura 5. 17 - Tunelamento em IP móvel.

Logo, o funcionamento do IP móvel pode ser resumido da seguinte forma:

1) Os roteadores móveis (local ou externo) avisam sobre a sua presença através de mensagens de anúncio. Um nó móvel pode opcionalmente solicitar uma tal mensagem de qualquer operador móvel usando uma *mensagem de solicitação de operador*.

2) Um nó móvel recebe um *anúncio de agente* e determina se está na sua rede local ou em uma rede externa.

3) Quando o nó móvel detecta que está em sua rede local, ele opera sem serviços móveis. Ao voltar a sua rede local depois de ter sido registrado em um outro local, o nó móvel é desregistrado pelo seu agente local através de uma variação do processo de registro.

4) Quando um nó móvel detecta que está em uma rede externa, ele obtém um endereço móvel. O endereço móvel pode ser tanto um endereço móvel externo como um endereço móvel colocado.

5) O nó móvel, operando fora da *rede local*, então registra o seu *endereço móvel* com seu *agente local* através da troca do pedido de registro e da mensagem de resposta de registro, provavelmente através de um *agente externo*.

6) Datagramas mandados para o endereço fixo do nó móvel são interceptados e tunelados pelo seu agente local para o endereço móvel do nó móvel, recebidos no

fim do túnel (tanto pelo agente externo quanto pelo próprio nó móvel) e finalmente entregue ao nó móvel.

5.3 – Segurança em IP Móvel

Apesar de toda a flexibilidade e facilidade oferecidas, a mobilidade expõe os nós móveis e consequentemente toda a rede a ameaças à segurança. O ambiente da computação móvel é potencialmente muito diferente do ambiente de computação ordinário. Em muitos casos os computadores móveis estão conectados a rede através de enlaces sem fio. Tais enlaces são particularmente vulneráveis a espionagem passiva, *ataques por repetição*[26], e outros ataques ativos. Assim, técnicas de segurança tem grande importância para conexões de redes móveis sem fio.

Nesta seção alguns aspectos da segurança em IP móvel são mostrados. Inicialmente, aspectos de segurança no processo de registro do endereço móvel são analisados e então as ameaças mais comuns à segurança em redes móveis são mostradas, assim como as tecnologias, serviços e protocolos que devem ser usados para combater tais ameaças.

5.3.1 – Registrando o Endereço Móvel de Forma Segura

Nesta seção alguns técnicas de segurança utilizadas no processo de registro são mostradas.

Message Authentication Codes (MACs)

Cada nó móvel e roteador local deve compartilhar uma associação segura e estar apto a utilizar o MD5 [R92b] com chave de comprimento 128 bits a fim de criar assinaturas digitais não passíveis de falsificação para pedidos de registro. Maiores detalhes sobre o uso do MD5 no IP móvel são encontrados no capítulo 3, seção 3.4.1. Outros algoritmos de autenticação, modos de algoritmo, métodos de distribuição de chaves e comprimentos de chave podem também ser permitidos pelo protocolo.

Áreas de Segurança Relativas a este Protocolo

O protocolo de registro do IP móvel faz com que o tráfego do nó móvel até o seu endereço móvel seja tunelado e tal tunelamento pode ser altamente vulnerável se o registro não foi autenticado. O redirecionamento remoto feito no processo de registro é amplamente conhecido como um problema de segurança se não houver autenticação [B89]. Devido a natureza remota deste processo nota-se claramente a necessidade da utilização de um processo de autenticação [VK83]. O agente local precisa ter certeza que o pedido de registro de endereço móvel foi feito pelo nó móvel e não por outro nó qualquer querendo se passar pelo nó móvel em questão. Um nó malicioso pode, por exemplo, fazer com que o agente local altere sua tabela de roteamento com informações errôneas sobre endereço móvel, fazendo com que o nó móvel se torne inacessível a todas as comunicações via Internet.

Gerenciamento de Chave

[26] Ataque por Repetição : Uma seqüência de eventos ou comandos é observada e reproduzida posteriormente para que se possa efetivar alguma ação não autorizada.

O IP móvel necessita de um mecanismo de autenticação muito forte (por exemplo, MD5 chaveado) a fim de prevenir potenciais ataques baseados no processo de autenticação. Distribuição de chaves para centenas ou até mesmo milhares de usuários seria muito complicado para os administradores de rede. Por exemplo, em ambientes comerciais é importante que todas as mensagens entre o agente externo e o local sejam autenticadas de modo que a cobrança do serviço seja possível e que os provedores de serviço não permitam clientes ilegítimos de utilizar seus serviços.

Escolhendo Bons Números Aleatórios

A força de um mecanismo de autenticação depende de muitos fatores, incluindo a robustez do próprio algoritmo de autenticação, o segredo da chave utilizada, a força da chave usada, e a qualidade da implementação. Para fazer com que a autenticação através do MD5 chaveado seja útil, a chave de 128 bits deve ser pseudo-aleatória e secreta . Se números aleatórios (*nonces*) são utilizados na conexão com proteção contra ataques por repetição, esta chave deve ser selecionada de forma bastante cuidadosa.

Privacidade

Usuários que possuem dados sensíveis devem usar mecanismos que protejam os mesmos, mas que não causem conflitos com o IP móvel (eg. Criptografia). Usuários preocupados com analise de tráfego devem considerar o uso de cifragem sobre o enlace. Se privacidade local absoluta é requerida, o nó móvel pode criar um túnel para seu agente local [M97]. Então, datagramas destinados para o nó parecerão vir da rede local, podendo ser mais difícil encontrar a localização do nó móvel.

Proteção contra Ataques por Repetição para Pedidos de Registro

O campo de identificação é usado a fim de permitir que o agente local verifique que a mensagem de registro foi gerada recentemente pelo nó móvel, e desta forma não repetida de um registro prévio enviado por alguém inescrupuloso. Dois métodos são utilizados para evitar ataques por repetição.

O nó móvel e seu agente local devem concordar no método escolhido para proteção contra ataques por repetição que será usado por eles, por que um método permite alguma liberdade de ação enquanto que o outro não, e além disso a interpretação do campo de identificação depende do método de proteção contra ataques por repetição escolhido.

Qualquer método usado terá os 32 bits menos significativos copiados do pedido de registro para a sua resposta. O agente externo usa tais bits (e o endereço fixo do nó móvel) para “casar” os pedidos de registro com suas respostas correspondentes. O nó móvel deve verificar se os 32 bits menos significativos de qualquer registro são idênticos aos bits mandados no pedido de registro.

A identificação num novo pedido de registro não pode ser a mesma de outros pedidos, e não deve ser repetida enquanto o mesmo contexto de segurança está sendo usado entre o nó móvel e o agente local. Quando a proteção baseada em *timestamps* é utilizada, uma nova identificação de registro é escolhida para cada retransmissão, sendo assim considerada como um novo registro. Já quando o método baseado em números aleatórios é utilizado o pedido não respondido é retransmitido sem modificação; assim a retransmissão não é considerada um novo registro.

Dois métodos utilizados para evitar ataques por repetição são descritos a seguir.

Proteção** contra Ataques por Repetição **usando Timestamps

Todos os nós móveis obrigatoriamente devem implementar a proteção contra ataques por repetição baseada em *timestamp*. A idéia básica deste método é que o nó que gera a mensagem insere na mesma a atual hora do dia (*timestamp*), e o nó que recebe a mensagem verifica se o *timestamp* é suficientemente próximo a sua hora do dia. É obvio que os dois nós devem estar adequadamente sincronizados. Assim como outras mensagens, mensagens de sincronização devem detectar tentativas de violação através de mecanismo de autenticação determinado pelo contexto de segurança entre os dois nós.

Ao receber um pedido de registro com uma extensão da autenticação móvel-local, o agente local verifica se o campo identificação é válido. Para ser válido, o *timestamp* contido no campo identificação deve estar suficientemente próximo da hora indicada no relógio do agente local, e o *timestamp* deve ser maior que os *timestamps* aceitos anteriormente pelo nó móvel que fez os pedidos de registro. Tolerâncias e resincronização são consideradas parte de uma estratégia de segurança particular.

Se o *timestamp* é válido, o agente local copia completamente o campo identificação na resposta do registro quando a envia para o nó móvel. Se o *timestamp* não é válido, o agente local copia apenas os 32 bits menos significativos na resposta de registro e fornece os 32 bits mais significativos de sua hora do dia. Neste último caso, o agente local deve rejeitar o pedido de registro retornando o código 133 na resposta de registro.

Antes de utilizar os 32 bits mais significativos para a resincronização, o nó móvel deve verificar que os 32 bits menos significativos da identificação da resposta de registro são idênticos àqueles na tentativa de registro rejeitada.

Proteção** contra Ataques por Repetição **usando Números Aleatórios

A idéia básica deste método consiste em o nó *A* incluir um novo número aleatório a cada mensagem enviada para o nó *B*, e verificar se o nó *B* retorna o mesmo número na próxima mensagem para o nó *A*.

Ambas as mensagens usam um código de autenticação a fim de protegê-las de alguma alteração por um atacante. Ao mesmo tempo o nó *B* pode mandar seus próprios números aleatórios em todas mensagens para *A* (para ser repetidas pelo nó *A*), de modo que *A* também possa verificar as mensagens enviadas por *B*.

O agente local deve ter recursos para gerar números aleatórios úteis no caso do uso de proteção contra ataques por repetição usando números aleatórios assim como para outros propósitos. O número aleatório é inserido pelo agente local nos 32 bits mais significativos do campo de identificação de cada resposta de registro. O agente local copia os 32 bits menos significativos da identificação do pedido de registro nos 32 bits menos significativos da identificação da resposta de registro. Quando o nó móvel recebe uma resposta de registro autenticada do agente local, ele guarda os 32 bits de maior ordem da identificação para usar como os 32 bits do seu próximo pedido de registro.

Se um registro é rejeitado por causa de um número aleatório inválido, a mensagem de resposta de registro sempre dá ao nó móvel um novo número aleatório para ser usado no próximo registro. Assim, o método baseado em número aleatório é auto-sincronizável.

5.3.2 – *IP Security*

O IPSec [KA98a] é uma arquitetura criada pela IETF (*Internet Engineering Task Force*) formada por protocolos e serviços que descrevem mecanismos de segurança para o IPv4, IPv6 [DH98] e camadas superiores (TCP, UDP, etc), sendo usado no IP móvel.

Na figura 5.4 observa-se como o IPSec está estruturado. Três protocolos principais são definidos: o CA (Cabeçalho de Autenticação) [KA98b], o ESP (*Encapsulating Security Payload*) [A95] e o IKE (*Internet Key Exchange*) [HC98].

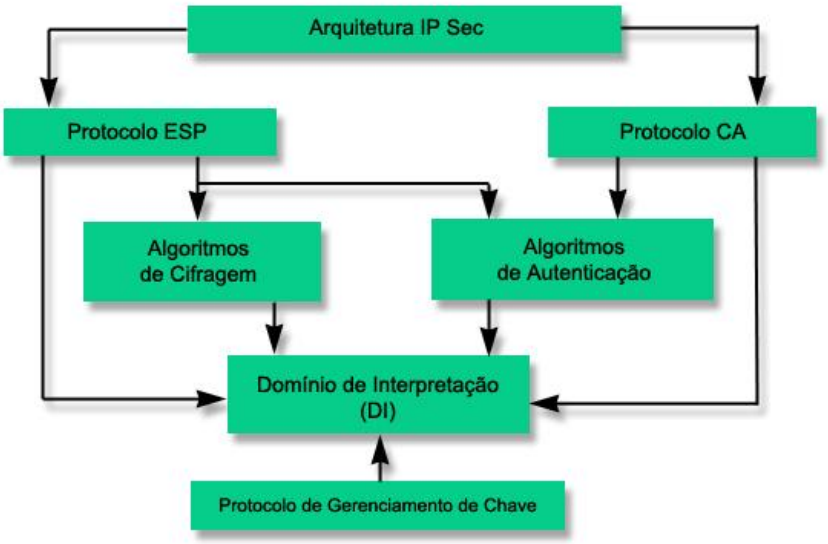


Figura 5. 18 - IPsec.

O protocolo CA é usado para proporcionar integridade e autenticidade para datagramas IP e proteção contra ataques por repetição. *Integridade* assegura que o datagrama não foi alterado de maneira inesperada ou maliciosa; a*utenticação* verifica a identidade da fonte solicitante e a *proteção anti-repetição* previne que os usuários recebam pacotes retardados intencionalmente por nós com intenções maliciosas.

O CA foi projetado para trabalhar com diferentes algoritmos para autenticação, porém o mais usado é o MD5 [R92b]. O MD5 é uma função matemática unidirecional que produz uma representação (*sumário de mensagem*) dos dados a serem autenticados cujo comprimento é 128 bits. A cada datagrama IP a ser enviado é associado um sumário de mensagem que é gerado a partir do datagrama mais uma chave apropriada. Ao receber o datagrama, o sumário de mensagem é calculado novamente a partir do datagrama mais a chave conhecida pelo receptor; se os dois sumários calculados forem iguais não houve alteração durante o transito e apenas remetentes com conhecimento da chave seriam capazes de enviá-lo. Maiores detalhes sobre o MD5 podem ser encontrados na seção 3..4 do capítulo 3.

O formato do IP CA é mostrado na figura 5.5 . Nela o *Next Header* é um campo de 8 bits no qual se define a próxima carga depois do CA; o Comprimento de Carga é um campo de 8 bits que define o comprimento do CA como sendo uma palavra de 32 bits de comprimento; a área reservada de 16 bits de comprimento está reservada para uso futuro e deve conter apenas zeros; o IPS (Índice de Parâmetros de Segurança) é um valor arbitrário de 32 bits que combinado com o IP do destino e o protocolo de segurança (CA) identifica unicamente o AS[27] (Associação de Segurança) para este datagrama; o número de seqüência é um campo de 32 bits contendo um contador monotonicamente crescente para propósito de proteção anti-repetição e o campo de Dados Autenticados é um campo cujo

[27] Associação de Segurança (AS) é um acordo unidirecional entre duas partes numa comunicação que especifica um conjunto de políticas e chaves para proteger futuras comunicações entre as partes.

comprimento é variável (deve ser um múltiplo de 32 bits) e que contém o VVI (Valor de Verificação de Integridade) do pacote.

Próximo Cabeçalho	**Comprimento da Carga**	**RESERVADO**
(IPS) Índice de Parâmetros de Segurança		
Campo do Número de Seqüência		
Dados de Autenticação (variável)		

Figura 5. 19 - Formato do cabeçalho IP de Autenticação.

A localização do CA no datagrama IP depende da versão do IP utilizada e se o modo usado é o modo de transporte[28] ou com tunelamento. Na figura 5.6 está ilustrado como um pacote autenticado e com tunelamento aparece no IPv6.

Cabeçalho IP (novo)	**Cabeçalho de Autenticação**	**Cabeçalho IP (original)**	**Cabeçalho TCP/UDP**	**Dados do usuário**

autenticado

Figura 5. 20 - Pacote Tunelado Autenticado IPv4.

Note que o protocolo CA não permite qualquer forma de cifragem, e desta forma não proporciona confidencialidade dos dados. Assim, com este propósito o ESP deve ser utilizado. Além de confidencialidade o ESP também proporciona todos os serviços proporcionados pelo CA. De fato o ESP deve ser usado conjuntamente com o CA.

Assim como o CA, o ESP foi construído para trabalhar com diferentes tipos de algoritmos para autenticação e cifragem. Indicava-se o DES em modo CBC, porém com a substituição do DES pelo AES, o processo de padronização para o uso do AES em modo CBC no IPSec já foi iniciado. Informações sobre a documentação atual com as definições para a utilização deste novo padrão de cifra dos EUA no IPSec assim como uma descrição do mesmo são proporcionados na seção 3.5 do capítulo 3.

Assim como algoritmos para autenticação, algoritmos para cifragem também dependem do uso de chaves a fim de se obter confidencialidade. Uma vez que o datagrama IP é cifrado, apenas usuários autorizados podem decifrá-lo.

O cabeçalho do ESP está ilustrado na figura 5.7. Os campos IPS e o número de seqüência tem a mesma funcionalidade descrita anteriormente com relação ao CA; carga é um campo de comprimento variável contendo dados descritos pelo campo Próximo Cabeçalho; o campo *Padding* é opcional, definido a fim de assegurar que o campo Dados de Autenticação (se presente) está alinhado a um limite de 4 bytes. O campo comprimento *Pad* indica o número de bytes de Pad que o precedem imediatamente. O campo Próximo Cabeçalho de comprimento 8 bits identifica o tipo de dados contidos no campo *Payload Data* (e.g. cabeçalho de extensão em IPv6 ou um identificador de protocolo em camadas superiores). O campo Autenticação de Dados é opcional, possui comprimento variável e contém um VVI (Valor de Verificação de Integridade) computado sobre o pacote ESP menos os dados autenticados.

(IPS) Índice de Parâmetros de Integridade

Autenticação

[28] Os protocolos CA e ESP suportam dois modos de uso: *transporte* e *com tunelamento*. No modo transporte os protocolos proporcionam segurança primariamente para protocolos nas camadas mais elevadas; no modo túnel, os protocolos são aplicados aos pacotes IP com tunelamento.

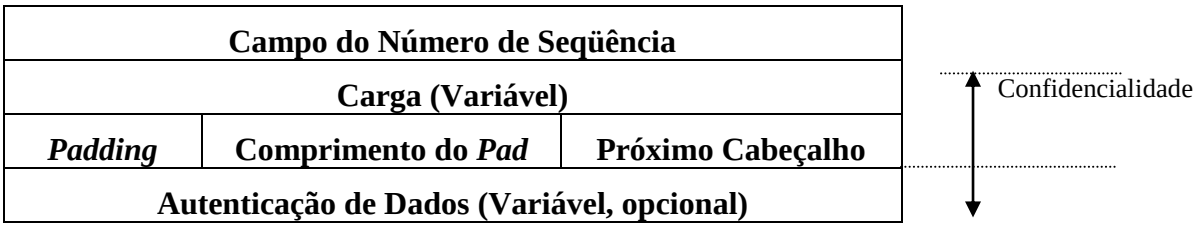


Figura 5. 21 - Cabeçalho de IP Encapsulating Security Payload (ESP).

O cabeçalho ESP é inserido depois do cabeçalho IP e antes do cabeçalho de protocolo de camada superior (modo transporte) ou antes do cabeçalho do IP encapsulado (modo com tunelamento). Para ilustração do caso do IP móvel, no qual é utilizado o modo túnel, a figura 5.8 mostra um pacote IPv6 cifrado com ESP e com tunelamento.

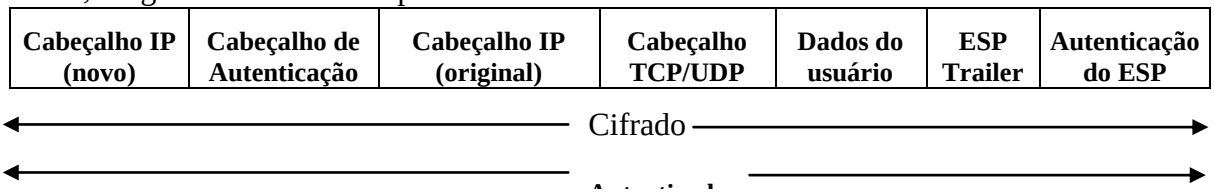


Figura 5. 22 - Pacote Tunelado Cifrado IPv4.

O compartilhamento das chaves de forma segura entre as partes envolvidas na comunicação dos dados é fundamental para um correto funcionamento dos protocolos CA e ESP, desta forma o protocolo IKE (*Internet Key Exchange*) foi definido tal que as partes envolvidas podem negociar parâmetros de segurança e estabelecer chaves de seção permitindo assim a definição da AS. A tabela 5.1 ilustra a tabela dos ASs identificados pelos seus SPI.

Índice de Parâmetro de Segurança	Algoritmo de Autenticação	Chave de Autenticação	Proteção Proteção contra Ataques por Repetição	Algoritmo de Cifragem	Chave de Cifragem
01234567	e. g., MD# chaveado	(uma chave secreta)	*Timestamp*		
89^{A}BCDEF				e.g. RSA	(chave pública/privada)

Tabela 5. 1 - Associações de Segurança.

As ASs podem ser estabelecidas entre usuários finais, *gateways* e usuários finais e *gateways*. Alguns *gateways* funcionam como parede corta-fogo e são conhecidos com *firewalls*[29] e são de fundamental importância na segurança das comunicações feitas via Internet. Diferentes tipos de *firewall* existem; no caso do IP móvel o mais sofisticado e importante é o chamado tunelador seguro (figura 5.9), o qual é implementado implementado utilizando os protocolos CA e ESP, e funciona da seguinte maneira com relação a pacotes recebidos de uma rede pública:

[29] Um *firewall* é um mecanismo ou conjunto de mecanismos localizados na fronteira entre dois domínios administrativos e configurado de maneira a somente permitir a entrada de pacotes no domínio privado que possuírem certas características (e.g. endereço IP fonte; portas TCP).

1) Se um pacote é transportado com tunelamento para o *firewall* e tem autenticação válida e/ou cifragem então o tunelamento é desfeito e ele passa a ser roteado de forma transparente para o nó destino dentro da rede privada.

2) De outra forma, o pacote é submetido ao servidor da camada de aplicação, o qual é configurado para fazer a filtragem de pacote baseada na aplicação.

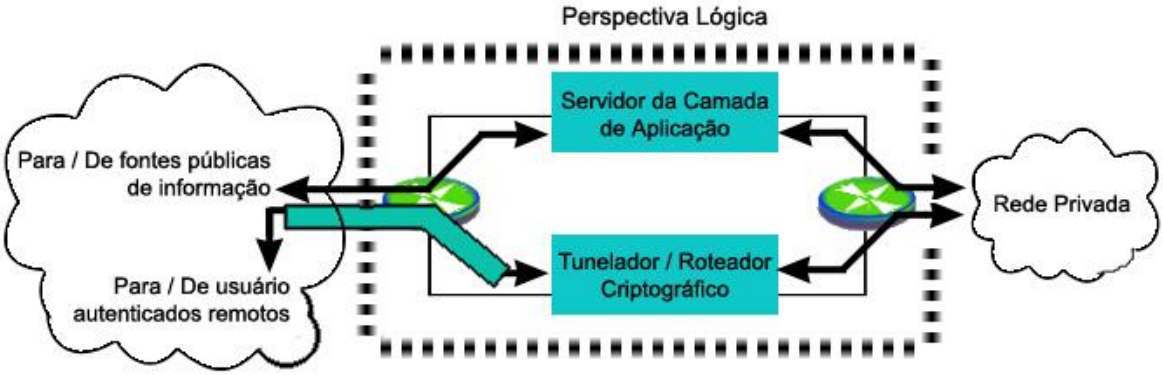


Figura 5. 23 - Tunelamento Seguro (Secure Tunneler).

Após a noção básica do funcionamento do IPSec passa-se a analise dos principais problemas de segurança ao se utilizar o IP móvel, iniciando-se com a análise de uma Intranet (seção 5.3.2) e passando a seguir a Rede Mundial (seção 5.3.3).

5.3.3 – IP Móvel em Intranets

Inicialmente será feita a analise da utilização do IP Móvel numa Intranet sem conexões com a Internet, sem *firewalls*, e com acesso físico seguro. Além disso, considera-se que os aplicativos para nós móveis estão instalados em todos os nós móveis da rede, e os aplicativos para agentes externos e locais estão instalados nos roteadores e que as chaves secretas de cifragem compartilhadas estão instaladas.

Em qualquer ambiente de rede, precauções contra ataques internos devem ser tomadas. Tais ataques são praticados freqüentemente por pessoas internas a rede que deveriam ser confiáveis, como por exemplo, empregados da própria empresa. Geralmente esses ataques envolvem acesso a informações sensíveis para propósitos maliciosos.

A seguir serão descritos alguns tipos de ataques e o que é feito no IP móvel para combatê-los.

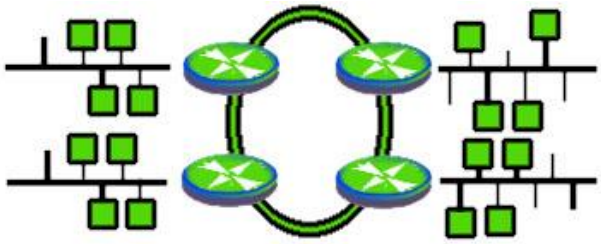


Figura 5. 24 - Modelo de rede para IP móvel em Intranets.

a) Negação de Serviço (*Denial of Service – DoS*)

Um ataque DoS tem como função interromper ou impedir o acesso a um sistema/aplicação. O sistema ou aplicação deixa de estar disponível, ou ainda, uma aplicação que possui tempo de execução crítico é atrasada ou abortada [BBSS97].

De um modo geral um ataque deste tipo pode ter duas formas: inundando-se um servidor com pacotes, fazendo com que o mesmo não possa processar pacotes úteis ou interferindo de alguma forma no fluxo dos pacotes entre os nós. No caso de uma rede utilizando IP móvel, este tipo de ataque ocorre quando um registro falso de um novo endereço móvel para um nó particular é feito. Esse registro falso causa dois problemas:

1) O nó com registro verdadeiro é desconectado;

2) O responsável pelo ataque se torna capaz de ver todo o tráfego direcionado ao nó em questão.

Este ataque é ilustrado na figura 5.11.

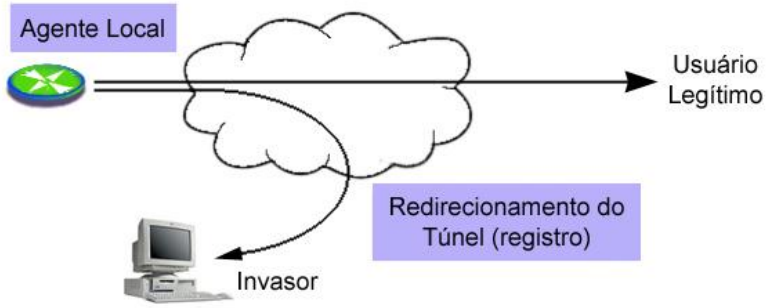


Figura 5. 25 - Ataque de negação de serviço numa rede IP móvel.

O IP Móvel previne falsos registros e consequentemente este tipo de ataque pedindo forte autenticação de todas as mensagens de registro que são trocadas a partir do processo de registro. Considerando que a chave privada compartilhada não é exposta, este ataque é impossível.

Relacionado ao ataque por negação de serviço está o ataque por repetição. Este ataque ocorre quando as mensagens de registro (cifradas) que são mandadas pelo nó móvel para o agente externo ao visitar uma rede, são gravadas e repetidas noutro momento. Este poderia ser um ataque bem sucedido por registro falso, porém o IP móvel possui duas maneiras de evitá-lo, preenchendo o campo ID com um *timestamp* ou um valor aleatório. Utilizando *timestamps* o receptor pode determinar o seu *timeliness* (o momento em que a mensagem foi gerada) e assim detectar e descartar a mensagem repetida. Já utilizando-se valores aleatórios as partes envolvidas combinam um valor especial para usar em cada uma das mensagens; mesmo que uma pessoa não autorizada tenha acesso ao valor que será usado na próxima mensagem de registro, inseri-lo numa mensagem autenticada é impossível.

b) Espionagem Passiva (*Passive Eavesdropping*)

Neste ataque informações são roubadas de forma passiva. O mesmo ocorre quando é possível acessar o tráfego entre o nó móvel e o agente local, desta forma todas as informações trocadas entre eles são escutadas.

Este tipo de ataque pode ocorrer, por exemplo, se a pessoa responsável pelo ataque conseguir acesso físico a rede e conectar um servidor a mesma. No caso de uma Ethernet compartilhada, todo o tráfego se encontra vulnerável a espionagem. Um espião perto o suficiente de uma rede sem fio pode receber pacotes que são transmitidos via radio; a recepção de tal tipo de sinal é praticamente impossível de ser evitada.

A fim de evitar tal ataque em redes sem fio é preciso que as informações transmitidas passem a ser transmitidas não de forma clara, mas sim cifradas. Considerando que a chave utilizada no processo de cifragem se encontra segura, tal tipo de ataque também se torna inviável.

c) Seqüestro de Sessão (*Session Stealing*)

O Seqüestro de Sessão, diferentemente da espionagem passiva, é um tipo de ataque no qual informações são roubadas de forma ativa. Nele o responsável pelo ataque segue os seguintes passos:

1) Espera o nó móvel se registrar com seu agente local;
2) Fica espionando até que algum informação interessante apareça;
3) Então começa a inundar o nó móvel com pacotes falsos, colocando-o assim fora de ação;
4) Rouba a sessão enviando pacotes que parecem vir do nó móvel, e ao mesmo tempo interceptando os pacotes enviados para o nó móvel em questão.

Tal ataque pode ocorrer tanto num enlace externo quanto em qualquer outro ponto entre o nó móvel e o agente local.

Novamente a proteção usada neste caso é a Criptografia, cifrando todo tráfego, de preferência em todos os lugares da conexão. Deste modo mesmo que a sessão seja seqüestrada não será possível acessar os dados roubados.

d) Outros ataques ativos

Ataques ativos não requerem, para ocorrer, que uma seção de IP móvel esteja em andamento. Este tipo de ataque consiste em se conectar e invadir os servidores na rede. Uma vez que o invasor conseguiu ganhar acesso físico à rede (via um acesso desprotegido da rede ou via uma interface aérea), o procedimento para este tipo de ataque consiste:

1) O invasor arranja um prefixo de rede para usar, o que pode ser feito ouvindo os anúncios de agente, examinando endereços IP em pacotes trafegando no segmento da rede, ou ainda apenas fazendo um pedido de configuração DHCP.
2) A seguir o invasor seleciona um número de servidor disponível para usar, o que pode ser feito “ouvindo-se” a rede por algum tempo e escolhendo aquele que

parece não estar sendo usado; isto é feito através um pedido ARP para o endereço IP resultante e vendo se não há resposta, ou ainda através de um pedido DHCP.

3) Uma vez que os passos anteriores tiveram êxito, o invasor pode começar ganhando acesso aos servidores.

5.3.4 – IP Móvel na Internet

O IP móvel pode permitir o movimento do usuário conectado a Internet para qualquer lugar sem expor sua Intranet a outras ameaças a segurança comuns a redes conectadas a Internet. A Figura 5.12 ilustra este cenário. Nela visualiza-se parte da Intranet com dados confidenciais conectada a Internet através de um firewall para assegurar conectividade a nós móveis autorizados. Também é possível observar nesta figura uma área pública desta Intranet que proporciona acesso para nós móveis.

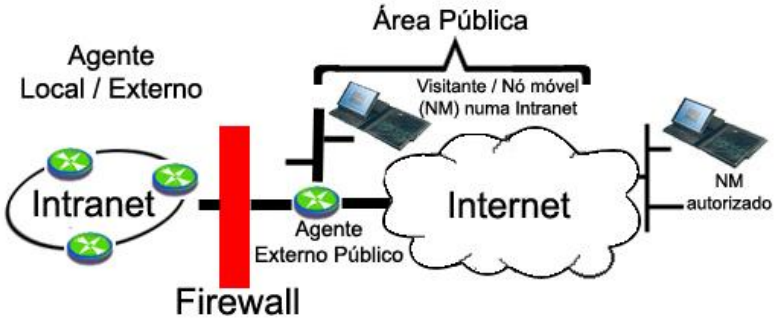


Figura 5. 26 - Cenário de IP móvel.

A mobilidade no cenário da Internet se caracteriza pela posição dos agentes locais e externos com respeito ao firewall e aos nós móveis. Embora os agentes locais são protegidos pelo *firewall*, todos os agentes externos não pode estar sob um *firewall*. O agente externo publico que não é protegido tem que proporcionar serviço aos nós móveis da área publica. Portanto este agente permite espionagem passiva ou ativa. Este cenário é considerado por toda seção mostrando como o nó móvel pode alcançar seu agente local de modo seguro.

Proteção do nó móvel

Embora não haja adição de problemas de segurança em relação a qualquer outra rede conectada à Internet, a vulnerabilidade da intranet com acesso a Internet introduz ameaças a segurança sobre os nós móveis que estão na Internet sem *firewall* e que devem ser protegidos contra esses problemas já conhecidos de segurança. A proteção contra esses ataques pode ser um método baseado na tecnologia VPNs (*Virtual Private Networks*).

A figura 5.13 mostra VPNs para proteção de Intranets. Uma VPN consiste em duas ou mais redes privadas físicas que são separadas por uma rede pública e se comporta como uma única rede. As VPNs são construídas para túneis autenticados e cifrados entre *firewalls* tunelados na fronteira de cada rede física. O *firewall* protege a rede permitindo o acesso apenas àqueles pacotes que foram autenticados e cifrados por um dos outros *firewalls*. A aplicação da tecnologia VPN para um nó móvel também é mostrada nesta

figura. Nela observa-se que o nó móvel é representado como uma rede privada com apenas um nó protegida por um firewall integrado.

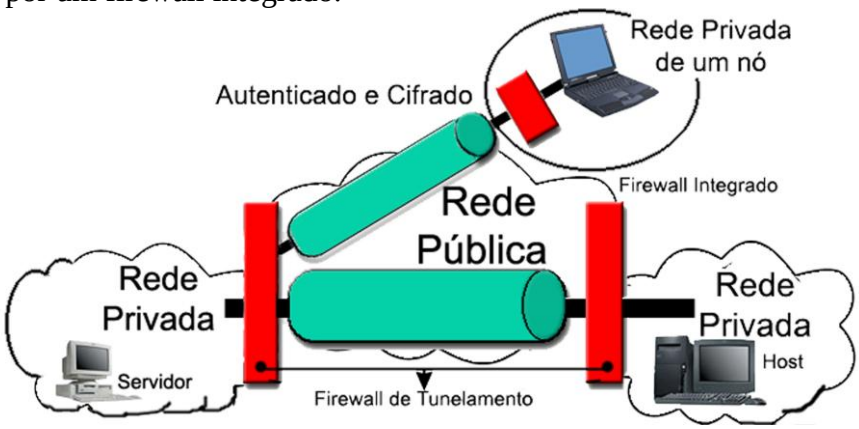


Figura 5. 27 - Rede privada virtual assegurando travessia segura de firewall.

Esta solução protege um nó móvel através de um tunelador seguro. Portanto o tunelador seguro é um *firewall* (Fig. 5.12) que proporciona um caminho criptograficamente protegido para usuários autorizados a acessar a rede privada através da rede pública. A solução deve proporcionar uma maneira de fazer com que o nó móvel possa se comunicar com todos os servidores e roteadores em todo o resto da VPN (qualquer uma das redes físicas, privadas) sem comprometer a segurança daquelas redes. O protocolo SKIP (*Simple Key-Management Protocol*) é uma maneira de atravessar o *firewall* de modo seguro. Na próxima seção é mostrado como este protocolo pode ser usado para proteger o nó móvel.

Travessia segura de firewall para IP móvel

O travessia de um *firewall* baseado no SKIP requer um pedido de registro tunelado de forma segura para o *firewall*. A implementação de um pedido de registro consiste de um IP Cabeçalho de Autenticação (CA), um *Encapsulation Security Payload* (ESP), e um protocolo para gerenciamento de chave a fim de satisfazer as funções de cifragem e autenticação para tunelamento. Portanto a implementação da travessia segura para IP móvel necessita da configuração do nó móvel e do *firewall* para se obter chaves públicas da configuração do nó móvel e agente local com uma faixa de endereços IP da intranet, e fornece ao nó móvel e ao *firewall* pelo menos um algoritmo de autenticação comum com chave secreta e um algoritmo de cifragem com chave secreta comum.

Depois das funções de autenticação e cifragem ter sido completadas, o nó móvel conectado ao enlace externo deve executar suas funções de IP móvel normais: determinar sua localização atual, adquirir endereço móvel, e registrando este endereço móvel com seu agente local usando seu *firewall* integrado. O que significa que o registro do nó móvel ao seu agente local é feito do mesmo modo que os pacotes são transportado na VPN e que o registro do endereço móvel com seu agente local deve tunelar de forma segura ao *firewall*.

O cabeçalho SKIP como um componente do pedido de registro informa ao *firewall* a identidade do nó móvel e permite o *firewall* determinar os algoritmos de cifragem e autenticação que foram usados pelo nó móvel. Junto ao cabeçalho SKIP outros componentes da mensagem de registro são um cabeçalho do pacote IP externo (acoplando o túnel ao *firewall*), um CA, cabeçalho de *Encapsulatin Security* e a mensagem de pedido de registro cifrada. A função de segurança desses componentes foi introduzida na seção 5.3.2. Detalhes adicionais do processo de registro podem ser encontrado em [P96] e [D91].

Uma vez que este processo de registro foi completado, o agente local começa atraindo pacotes destinados ao nó móvel e os tunela ao endereço móvel como sempre.

Referências:

[A95] – R. Atkinson, *IP Encapsulating Security Payload*, RFC 1827, Augusto 1995.
http://www.faqs.org/rfcs/rfc1827.html

[AD97] – S. Alexander e R. Droms, DHCP Optionse BOOTP Vendor Extensions, RFC 2132, Março 1997.

[B89] – S. M. Bellovin, *Security Problems in TCP/IP Protocol Suite*, ACM Computers Communications Review 19, Março 1989.

[BBSS97] – T. Bernstein, A. B. Bhimani, E. Schultz, C. A. Siegel, *Segurança na Internet*, Editora Campus, 1997.

[C95] – Y. Chen, *A Survey Paper on Mobile IP*, Agosto 1995.

[D91] – S. Deering, *ICMP Router Discovery Messages*, RFC 1256, Network Working Group Request for Comments 1321, Setembro 1991.

[D97] – R. Droms, *Dynamic Host Configuration Protocol*, RFC 2131, Março 1997.

[DH98] – S. Deering e R. Hiden, Internet Protocol, Version 6 (IPv6) Specification, RFC 2460, Dezembro 1998.
http://www.faqs.org/rfcs/rfc2460.html

[HC98] – D. Harkins e D. Carrel, *The Internet Key Exchange*, RFC 2409, Novembro 1998.
http://www.faqs.org/rfcs/rfc2409.html

[KA98a] – S. Kent e R. Atkinson, *Security Architeture for the Internet Protocol*, RFC 2401, Novembro 1998.
http://www.faqs.org/rfcs/rfc2401.html

[KA98b] – S. Kent e R. Atkinson, *IP Authentication Header*, RFC 2402, Novembro 1998.
http://www.faqs.org/rfcs/rfc2402.html

[M97] – G. Montenegro, *Reverse Tunneling for Mobile IP*, Network Working Group Request for Comments 2344, Maio 1998.

http://www.faqs.org/rfcs/rfc2344.html

[P81] – J. Postel, , *Internet Protocol*, RFC 791, Setembro 1981.

http://www.faqs.org/rfcs/rfc791.html

[P96a] - C. Perkins, *IP Encapsulation Within IP*, Network Working Group Request for Comments 2003, Maio 1996.

http://www.faqs.org/rfcs/rfc2003.html

[P96b] - C. Perkins, *Minimal Encapsulation Within IP*, Network Working Group Request for Comments 2004, Maio 1996.

http://www.faqs.org/rfcs/rfc2004.html

[P96c] – C. Perkins, *IP Mobility Support*, Network Working Group Request for Comments 2002, Outubro 1996.

http://www.faqs.org/rfcs/rfc2002.html

[P98] – C. E. Perkins, *Mobile Networking Through Mobile IP*, IEEE Internet Computing, January- February 1998.

[R92b] – R. L. Rivest, *The MD5 Message Digest Algorithm*, Network Working Group Request for Comments 1321, Abril 1992.

http://www.faqs.org/rfcs/rfc1321.html

[T96] – A. S. Tanenbaum, *Computer Networks*, 3ª Edição, Prentice Hall PTR, 1996.

[TSS99] – G. Tuquerres, M. R. Salvador e R. Sprenkles, *Mobile IP: Security and Application*, December 1999.

[V99] – M. S. Vann,, *An overview of Mobile IP with Attention to Wireless Networks*, Dezembro, 1999.

http://cs.lamar.edu/csfaculty/dosborne/project/p5.html.

[VK83] - V.L. Voydock e S.T. Kent, *Security Mechanisms in High-Level Networks*, *ACM Computer Surveys*, Vol. 15, No. 2, pp. 135-171, Junho 1983.

Páginas na Web

[P01] – C. E. Perkins - *Mobile Networking Terminology*.

http://computer.org/internet/v2n1/terms.htm

[P02] – C. E. Perkins – *Mobile IP Tutorial*.
http://computer.org/internet/v2n1/perkins.htm

[P03] – C. E. Perkins – *Nomadicity – How Mobility will Affect the Protocol Stack*.
http://computer.org/internet/v2n1/nomad.htm

6

Conclusões e Sugestões para Futuros Trabalhos

A criptografia é conhecida classicamente como uma arte muito antiga capaz de, através de certos métodos, processar informações tornando-as não legíveis, no caso de interceptação das mesmas por pessoas não autorizadas. Assim, a criptografia é uma das ferramentas utilizadas para proteger dados, proporcionando integridade e privacidade, evitando assim que tais dados sejam revelados, alterados, destruídos ou substituídos por pessoas não autorizadas. Além disso, diferentemente de outras ferramentas utilizadas na proteção de dados, os cripto-sistemas são os que se mostram mais completos, proporcionando alto nível de segurança com maior flexibilidade.

Esta dissertação apresenta um estudo sobre a Criptografia assimétrica baseada em curvas elípticas, com algumas aplicações focalizadas nas modernas técnicas de IP móvel. Inicialmente, uma revisão de alguns conceitos matemáticos pertinentes foi apresentada, após o que alguns aspectos básicos de Criptografia foram descritos. Diferentes tipos de problemas utilizados na construção de cripto-sistemas assimétricos foram abordados, notando-se claramente a vantagem do PLDCE sobre os demais. Os cripto-sistemas baseados neste problema, ponto central desta dissertação, foram então analisados e comparados com aqueles baseados em aritmética modular. A fim de ilustrar a adoção de ferramentas criptográficas utilizando curvas elípticas, um protocolo para redes móveis (IP móvel) que utiliza o IPSec na sua parte de segurança foi mostrado.

6.1 – Cripto-sistemas Simétricos x Cripto-sistemas Assimétricos

No decorrer deste trabalho foi possível observar algumas diferenças entre os cripto-sistemas simétricos e assimétricos, e como estes, apesar de ainda um pouco mais lentos que aqueles, possuem vantagens indiscutíveis, como por exemplo, a não necessidade de transmissão da chave secreta por um canal seguro, e um gerenciamento de chaves mais simples uma vez que, não precisando compartilhar a chave secreta, reduz-se de modo significante a quantidade de chaves a ser gerenciada. Além disso, promovem autenticação sem a possibilidade de repudiação, através do equivalente eletrônico da assinatura escrita, a assinatura digital.

Foram analisados os diferentes tipos de problemas utilizados atualmente como base na construção de cripto-sistemas assimétricos (fatoração de inteiros, PLD e PLDCE). Todos são capazes de proporcionar privacidade, autenticação, integridade e não repudiação, porém os baseados no PLDCE têm se mostrado mais vantajosos sobre os demais.

6.2 – Curvas Elípticas e Aritmética Modular

Dentre as vantagens observadas com o uso de cripto-sistemas assimétricos baseados no PLDCE e surgidos inicialmente na década de 80, encontra-se o fato de que cripto-sistemas baseados no PLD podem ser adaptados para ter como base o PLDCE, possuindo neste caso maior eficiência, pois podem utilizar grupos menores e corpos de característica 2 sem comprometimento da segurança. Isto implica uma maior velocidade na execução e a capacidade de possuir um mesmo nível de segurança com chaves notadamente menores, o que resulta num desempenho bastante superior. Além disso, por serem capazes de utilizar chaves com comprimentos significantemente menores para um mesmo nível de segurança, mecanismos que possuem tamanho, potência e banda reduzidas podem agora ter um alto nível de segurança sem comprometimento do desempenho.

Apesar das críticas por parte de alguns pesquisadores baseadas no fato deste tipo de cripto-sistemas ser estudado há relativamente pouco tempo e da dificuldade inerente do PLDCE e a maior facilidade do estudo dos outros problemas (Fatoração de Inteiros, PLD), uma prova bastante palpável da aceitação e reconhecimento do desempenho superior em relação aos outros é a adoção por parte de vários padrões de cripto-sistemas simétricos baseados no PLDCE. Um exemplo disto é o IPSec, protocolo de segurança utilizado pelos protocolos IP e IP móvel, no qual para se garantir a autenticidade tem-se como parte integrante o Protocolo para Troca de Chaves de Diffie-Hellman.

Nota-se desta forma o crescente interesse despertado com relação a estes cripto-sistemas, fazendo com que mais criptólogos procurem cada vez mais aprofundar o conhecimento sobre os mesmos.

6.3 – Sugestões para Futuros Trabalhos

Esta dissertação pode ser vista como um passo inicial para um maior entendimento das entidades matemáticas conhecidas como curvas elípticas e do seu uso na área de criptografia, com o intuito de poder futuramente contribuir de modo mais efetivo na área e auxiliar outros no conhecimento da mesma, procurando de alguma forma despertar maior interesse.

Como uma das sugestões para futuros trabalhos, tem-se o fato que ainda há a necessidade de se conceber maneiras ainda mais eficientes para se encontrar curvas elípticas, uma vez que a principal dificuldade de se implementar esquemas

criptográficos baseados em curvas elípticas definidas sobre corpos finitos é a necessidade de computação da cardinalidade dessas curvas.

Além disso, contribuições na formalização de padrões que ainda se encontram em andamento utilizando tais cripto-sistemas devem continuar a ser feitas através de, por exemplo, investigações sobre a segurança e desempenho de tais cripto-sistemas, assim como sugestões para novos padrões.

Contribuições na área de implementação são sempre importantes, a fim de tornar os cripto-sistemas assimétricos baseados em curvas elípticas cada vez mais vantajosos.

Por fim, é sugerida a busca de estruturas matemáticas que permitam a construção de novos cripto-sistemas assimétricos.

Apêndice A

GEOMETRIA PROJETIVA

O propósito deste apêndice é apresentar alguns conceitos básicos de geometria projetiva a fim de facilitar o entendimento da construção de um grupo aditivo a partir dos pontos de uma curva elíptica juntamente com o ponto no infinito.

A.1 – Coordenadas Homogêneas

Todas as transformações em geometria projetiva são não-lineares se as equações são expressas em coordenadas não-homogêneas, enquanto que, em coordenadas homogêneas, as transformações são lineares [w01]. Isto proporciona uma das principais motivações para o uso de coordenadas homogêneas, uma vez que os sistemas lineares são simbolicamente e numericamente mais fáceis de lidar do que os não-lineares. Outra vantagem da utilização de coordenadas homogêneas é tornar possível a representação numérica de pontos que se localizam no infinito, o que não é possível em coordenadas não homogêneas.

Define-se o *polinômio homogêneo* de $F(x,y)$ como sendo o polinômio $\tilde{F}(X,Y,Z)$ obtido de $F(x,y)$, pelas substituições $x=\frac{X}{Z}$ e $y=\frac{Y}{Z}$ (denominadores podem ser eliminados multiplicando-se pela potência de Z adequada) [K94]. Supondo que os polinômios considerados possuem coeficientes sobre o corpo K, e que se deseja encontrar triplas $X, Y, Z \in K$, tais que $\tilde{F}(X,Y,Z)=0$. Note que:

1) Para qualquer $\lambda \in K$, $\tilde{F}(\lambda X,\lambda Y,\lambda Z)=\lambda^n \tilde{F}(X,Y,Z)$ (n = grau total[30] de $F(x,y)$);

2) Para qualquer $\lambda \in K$ não nulo, $\tilde{F}(\lambda X,\lambda Y,\lambda Z)=0$ se e só se $\tilde{F}(X,Y,Z)=0$. Em particular, para $Z \neq 0$, tem-se $\tilde{F}(X,Y,Z)=0$ se e só se $F\left(\frac{X}{Z},\frac{Y}{Z}\right)=0$.

A.2 – Classes de Equivalência e Geometria Projetiva

[30] Grau total de um monômio $x^i y^j$ é dado por $i+j$. O grau total de um polinômio é dado pelo máximo grau total dos monômios que constituem tal polinômio.

Diz-se que duas triplas (*X*,*Y*,*Z*) e (*X*',*Y*',*Z*') são equivalentes se, para todo $\lambda \in K$, $\lambda \neq 0$, $(X',Y',Z') = \lambda(X,Y,Z)$. O conjunto das triplas equivalentes formam uma *classe de equivalência*. Define-se o *plano projetivo* P_K^2 como sendo o conjunto formado por todas as classes de equivalência de triplas não triviais [K84]. Uma maneira de visualizar o plano projetivo é considerar *X*, *Y*, *Z* reais, não todos nulos, ou seja, as triplas (*X*, *Y*, *Z*) formando a classe de equivalência dos pontos no espaço tridimensional ordinário[31] pertencentes às retas que passam pela origem.

Outra maneira de visualizar o plano projetivo sobre os reais, P_R^2, é considerar um plano do espaço tridimensional Euclidiano paralelo ao plano *XY*, por exemplo, o plano *Z*=1. Desta forma, todas as retas que passam pela origem com exceção das pertencentes ao plano *XY* interceptam o plano Z=1 em apenas um ponto. Desta forma para $Z \neq 0$, existe uma única classe de equivalência da forma (*x*,*y*,1) ($x = \frac{X}{Z}; y = \frac{Y}{Z}$). Assim, o plano projetivo pode ser identificado com os pontos (*x*,*y*) do plano ordinário mais os pontos onde *Z*=0, i. e., aqueles da forma (*x*,*y*,0), os quais formam a reta no infinito.

Assim, a *reta no infinito* consiste da classe de equivalência (*x*,*y*,0) com $y \neq 0$, contendo uma única tripla da forma (*x*,1,0), juntamente com um único ponto no infinito (1,0,0). Desta forma, nota-se que a reta no infinito também pode ser visualizada como uma reta ordinária.

Resumindo, uma *reta projetiva* P_K^1 sobre o corpo *K* consiste do conjunto de classes de equivalência formadas pelos pares (*x*,*y*). Desta forma P_K^2 pode ser visto como um plano ordinário (*x*,*y*,1) juntamente com uma reta ordinária (*x*,1,0) e com seu ponto no infinito (1,0,0). Generalizando, um espaço projetivo *n*-dimensional P_K^n é definido pela classe de equivalência de (*n*+1) – uplas que podem ser visualizadas como um espaço Euclideano de dimensão *n* formado pelas *n*-uplas $(x_1, x_2, \ldots, x_n)$ e um espaço P_K^{n-1} no infinito. A fim de explicar o ponto no infinito são necessários apenas P_K^1 e P_K^2.

A.3 – O Ponto no Infinito

Na prática, o *ponto no infinito* (também chamado *ponto ideal*) aparece como a intercessão de duas retas paralelas. Note que ao se obter uma representação para um ponto no infinito não há necessidade de tratá-lo diferentemente de qualquer outro ponto. As transformações e manipulações em geometria projetiva são aplicáveis tanto a pontos ideais quanto não-ideais.

Pelo que foi dito até o momento, define-se o plano projetivo como o conjunto formado pelos pontos ideais e não-ideais. Assim, dado um polinômio homogêneo $\tilde{F}(X,Y,Z)$ com *X*, *Y*, $X \in K$, as soluções da equação $\tilde{F}(X,Y,Z) = 0$ são dadas pelos pontos $(X,Y,Z) \in \mathrm{P}_K^2$, ou seja, as classes de equivalência (X,Y,Z) tais que $\tilde{F}(X,Y,Z) = 0$.

Os pontos em que $Z \neq 0$ são os pontos (*x*,*y*,1) para os quais $\tilde{F}(X,Y,1) = F(x,y) = 0$. Os pontos restantes , cujo *Z*=0, constituem os pontos pertencentes à reta no infinito.

[31] Espaço ordinário pode ser também denominado espaço de Euclides ou ainda espaço afim.

Considere novamente a curva elíptica $F(x,y) = y^2 - x^3 + n^2 x$. Viu-se que sua forma homogênea é $\tilde{F}(X,Y,Z) = Y^2Z - X^3 + n^2XZ^2$.

Os pontos no infinito nesta curva elíptica são as classes de equivalência (*x*,*y*,0), tal que $\tilde{F}(X,Y,0) = -X^3$, i. e., *X* = 0 com multiplicidade 3.

Assim, conclui-se que existe apenas uma classe de equivalência da forma (0,1,0).

A fim de facilitar a visualização, seja, *K* o corpo dos reais. Neste caso, os pontos na reta do infinito correspondem as retas que passam pela origem do plano *xy*. Assim, se $X \to 0$, $Y/X \to \infty$, correspondendo a um único ponto no infinito (0,1,0). Este é o ponto O mostrado no capítulo 2 e como pode se observar o mesmo se localiza na intercessão do eixo *y* com a reta no infinito. Note que qualquer curva elíptica contém similarmente exatamente um único ponto (0,1,0) no infinito.

Todos os conceitos utilizados em curvas do tipo $F(x,y) = 0$ no plano *xy* também são válidos para a curva equivalente $\tilde{F}(X,Y,Z) = 0$. Assim, noções como reta tangente em um ponto da curva, pontos de inflexão, pontos singulares e suavidade dependem apenas do que está acontecendo na vizinhança do ponto considerado. Considerando, por exemplo, o plano projetivo P_R^2 sobre *R*, para analisar os pontos com coordenadas Z≠0, basta trabalhar sobre o plano *xy*, onde a curva tem equação $F(x,y) = \tilde{F}(X,Y,1) = 0$. Já no caso dos pontos que possuem *Z*=0, coloca-se a tripla na forma $(x,1,0)$ ou na forma $(1,y,0)$, sendo desta forma considerados , respectivamente, pontos sobre a curva $F(x,1,z) = 0$ no plano *xz* e pontos sobre a curva $F(1,y,z) = 0$ sobre o plano *yz*.

A.4 – Exemplos

A fim de ilustrar o que foi colocado até o momento dois exemplos serão dados a seguir.

A.4.1 – Equação de Fermat

Considere a famosa equação de Fermat

$$x^N + y^N = 1 \qquad \textbf{(A.1)}$$

onde *x*, $y \in \boldsymbol{Q}$, N≥3. Supondo que $x = \frac{a}{c}$ e $y = \frac{b}{d}$ são soluções irredutíveis de (A.1), tem-se

$$\left(\frac{a}{c}\right)^N + \left(\frac{b}{d}\right)^N = 1 \qquad \textbf{(A.2)}$$

e

$$a^N d^N + b^N c^N = c^N d^N \qquad \textbf{(A.3)}$$

A partir de (A.3) observa-se que $c^N \mid a^N d^N$, porém, por hipótese mdc(*a*,*c*)=1, logo, $c^N \mid d^N \Rightarrow c \mid d$. Similarmente, considerando que $d^N \mid b^N c^N$, chega-se a $d \mid c$. Conclui-se, portanto que $c = \pm d$; assumindo-se *c* e *d* positivos, tem-se que *c*=*d*. Assim, qualquer solução para a equação (A.1) terá a forma $\left(a/c, b/c\right)$, dando a solução (a,b,c) com $a,b,c \in Z$ para a equação homogênea.

$$X^N + Y^N = Z^N \quad \textbf{(A.4)}$$

Inversamente, qualquer solução (a,b,c) de (A.4) onde $c\neq 0$, dará uma solução racional $\left(a/c, b/c\right)$ para (A.1). Contudo, diferentes soluções (a,b,c) podem implicar na mesma solução racional $\left(a/c, b/c\right)$. Por exemplo, as soluções (a,b,c) e (ta,tb,tc) de (A.4) implicam numa única solução racional para (A.1). Assim, ao se resolver a equação (A.1) deve-se tratar as triplas (a,b,c) e (ta,tb,tc) como uma mesma solução, para $t \neq 0$.

Observa-se porém, que existem certas soluções inteiras para a equação homogênea (A.4) que não possuem correspondentes racionais para (A.1).

O primeiro caso a ser considerado, ocorre quando N é impar. Para a equação (A.4), as triplas $(1,-1,0)$ e $(-1,1,0)$ são soluções porém não constituem soluções racionais para (A.1). A fim de visualizar o que ocorre considere uma seqüência de soluções (a_i,b_i,c_i), $i = 1,2,3,\ldots$, tal que $(a_i,b_i,c_i) \to (1,-1,0)$, quando $i \to \infty$, por exemplo. A fim de que tal procedimento possa ser considerado, considere neste momento a_i,b_i,c_i números reais. Observa-se assim que as soluções racionais $\left(a_i/c_i, b_i/c_i\right)$ para a equação (A.1) tendem para (∞,∞) quando $(a_i,b_i,c_i) \to (1,-1,0)$. Ou seja, as soluções $(1,-1,0)$ e $(-1,1,0)$ para (A.4) correspondem a soluções no infinito para (A.1).

Neste momento será dada a primeira definição de um plano projetivo, que é essencialmente uma definição algébrica.

A.4.2 – Interseções de Retas Paralelas

Defini-se uma reta em P_K^2 como o conjunto de pontos $[a,b,c] \in \mathrm{P}_K^2$ cujas coordenadas satisfazem uma equação da forma

$$\alpha X + \beta Y + \gamma Z = 0 \quad \textbf{(A.5)}$$

para algumas constantes α, β e γ não todas nulas. Note que se $[a,b,c]$ satisfaz (A.5) então $[ta,tb,tc]$ também satisfaz para qualquer t ≠ 0 [ST92].

Assim, para se verificar se um ponto em P_K^2 está em tal reta, pode-se utilizar qualquer coordenada homogênea para o ponto.

Sabe-se que num plano ordinário dois pontos determinam uma única reta. De modo similar, duas retas no plano determinam um único ponto na qual as duas retas se interceptam, a não ser que tais retas sejam paralelas, neste caso não há ponto em comum. A partir de um ponto de vista prático, será proporcionado a essas retas paralelas um ponto de interseção. Como esses pontos não existem no plano, os mesmos serão adicionados forçosamente.

A fim de definir a quantidade de pontos que devem ser adicionados considere as quatros retas $L_1, L_1^{'}$, L_2 e $L_2^{'}$, com L_1 e L_2 retas paralelas, e P o ponto extra onde as mesmas se interceptam, e, $L_1^{'}$ e $L_2^{'}$ também paralelas, com ponto comum P' (Fig. A.1). Suponha também que L_1 e $L_1^{'}$ se interceptam num ponto ordinário Q, i.e, $L_1 \cap L_1^{'} = \{Q\}$. Porém, como já foi dito, duas retas devem ter apenas um ponto em comum; segue-se portanto que $P \in L_1$ e $P' \in L_1^{'}$ devem ser distintos. Assim, conclui-se que deve-se inserir

um ponto extra para cada direção distinta no plano ordinário, e decretar que a reta L consiste dos seus pontos usuais juntamente com o ponto extra determinado pela direção da reta.

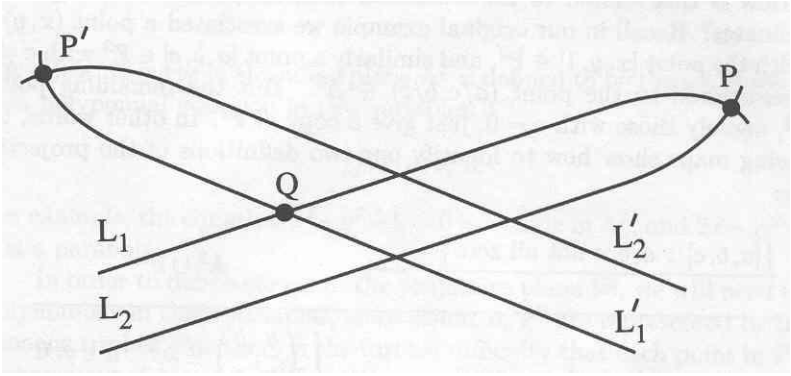


Figura A. 1 - Retas paralelas com ponto de interseção "no infinito".

Do que foi exposto chega-se a uma nova definição para plano projetivo desta vez em termos puramente geométricos. Assim, denotando o plano Euclidiano por

$$A_K^2 = \{(x, y): x \text{ e } y \text{ quaisquer números}\}, \qquad \textbf{(A.6)}$$

então, define-se, geometricamente o plano projetivo por

$$P_K^2 = A_K^2 \cup \{\text{conjunto de direções em } A^2\}. \qquad \textbf{(A.7)}$$

Duas retas tem a mesma direção se e só se são paralelas. Assim, pode-se definir direção através da noção de classe de equivalência formada pelas retas paralelas, i. e., uma *direção* é uma coleção de todas as retas paralelas a uma certa reta dada. Os pontos extras em P_K^2 associados às direções, i.e., os pontos que pertencem a P_K^2, mas que não pertencem a A_K^2 , são chamados *pontos no infinito*.

Como foi definido, uma reta em P_K^2 é definida como uma reta juntamente com seu ponto no infinito, que é determinado por sua direção. O conjunto de todos os pontos no infinito é considerado a reta no infinito denotada por L_∞. A interseção desta reta L_∞ com uma outra reta L qualquer em P_K^2 resulta no ponto no infinito correspondente à direção de L. Logo, após tudo que foi posto até o momento, nota-se que qualquer duas retas distintas em P_K^2 possuem um ponto em comum, concluindo-se portanto que não há retas paralelas em P_K^2.

Referências:

[K84] – N. Koblitz, *Introduction to Elliptic Curves and Modular Forms* , Springer-Verlag New York, Inc., 1984.

[K94] – N. Koblitz, *A Course in Number Theory and Cryptography*, 2ª Edição, Springer-Verlag New York, Inc., 1994.

[ST92] – J. H. Silverman e J. Tate, *Rational Points on Elliptic Curves*, Springer-Verlag New York Inc., 1992.

Página na Web

[w01]-http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/BEARDSLEY/beardsley.html

Apêndice B

TEORIA DA COMPLEXIDADE

No decorrer dessa dissertação expressões referentes à complexidade dos algoritmos criptográficos são utilizadas. A fim de promover um maior entendimento das mesmas, alguns conceitos básicos de teoria da complexidade são apresentados neste apêndice.

B.1 – Introdução

A principal meta da *teoria da complexidade* é proporcionar mecanismos para classificar problemas computacionais de acordo com os recursos necessários para resolvê-los. A classificação não depende necessariamente de um modelo computacional particular, mas da dificuldade intrínseca do problema. Os recursos usados para medição podem incluir tempo de execução, espaço para armazenamento, número de processadores, etc., mas tipicamente o foco principal é o tempo (*T*) e o espaço (*S*). Mais especificamente, em criptografia a *complexidade computacional* proporciona uma metodologia para analisar os recursos necessários nas técnicas e algoritmos criptográficos e também no estudo da dificuldade inerente de se "quebrar" uma cifra determinando desta forma o nível de segurança da mesma.

A Teoria da Informação [S49] afirma que algoritmos criptográficos (com exceção dos *one-time pads*) podem ser "quebrados". A Teoria da Complexidade diz se isso pode ocorrer antes do fim do universo [S96].

B.2 – Complexidade dos algoritmos

Um *algoritmo* é um procedimento computacional bem definido usado para resolver um problema e que recebe uma ou mais variáveis de entrada e retorna uma ou mais variáveis de saída.

Como foi visto na seção anterior a complexidade de uma algoritmo pode ser classificada usando diferentes fatores. Normalmente, os fatores mais usados para classificação de um algoritmo com relação a complexidade computacional são o tempo (*T*) e o espaço (*S*) As variáveis *T* e *S* são funções do comprimento da entrada.

O *comprimento da entrada* é o número total de bits necessários para representar a entrada em notação binária, usando um esquema próprio de codificação. Ocasionalmente, o comprimento de entrada será o número de itens de entrada [MOV96].

Geralmente a complexidade computacional de um algoritmo é expressa geralmente por sua ordem de magnitude, da forma $O(g(n))$ (chamada notação "*big O*"), onde $f(n) = O(g(n))$ significa que existem constantes c e n_0 tais que

$$f(n) \leq c \mid g(n) \mid \qquad \text{para} \qquad n \geq n_0. \tag{B.1}$$

Na prática, esta notação será usada apenas quando $g(n)$ é uma função mais simples que $f(n)$ e não cresce muito mais rapidamente que $f(n)$, em outras palavras, quando $g(n)$ fornece uma cota superior bastante próxima, sendo desta forma muito útil ao proporcionar uma boa idéia de quanto o crescimento da entrada, n, afetará o comportamento de $f(n)$. Exemplos que estão matematicamente corretos, mas que não são úteis na prática são:

(1) $n^2 = O(n^3 + n^2 \ln n + 6683)$;

(2) $n^2 = O(e^{(n^2)})$;

(3) $e^{-n} = O(n^2)$.

Se $f(n)$ e $g(n)$ são duas funções positivas para $n \geq n_0$, e se

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = cte \tag{B.2}$$

Então não é complicado mostrar que $f(n) = O(g(n))$. Se o limite é zero ainda é correto escrever $f(n) = O(g(n))$; porém neste caso também usa-se a denominação "little *o*", ou seja, diz-se que $f(n)=o(g(n))$. Isto significa que $f(n)$ é muito menor que $g(n)$ quando n é grande.

Exemplo B.1 – Suponha $f(n) = 22n^2 + 10n + 30$. Então $f(n) = O(n^2)$ porque $f(n) = 22n^2 + 10n + 30 \leq 23n^2$ para $n \geq 13$. Para se obter a ordem de magnitude basta observar o termo que domina o comportamento assintótico de $f(n)$ quando $n \to \infty$, deste modo $f(n)$ será expressa pela ordem de magnitude do maior termo, ignorando-se todas as constantes e termos de menor ordem.

□

Ao se medir o tempo e/ou espaço requerido por um algoritmo utilizando ordem de magnitude, tem-se como vantagem uma medida independente do sistema, sendo assim é desnecessário saber os tempos exatos das várias instruções ou o números de bits usados para representar as variáveis ou até mesmo a velocidade do processador. Mesmo considerando que, por exemplo, um computador é mais rápido do que outro, a ordem de magnitude não se altera. E esta consideração pode ser feita ao se lidar com algoritmos complexos como os utilizados em criptografia.

Outra vantagem da utilização desta notação é que a mesma permite ver como o requerimento de tempo e/ou espaço cresce quando a entrada cresce. Por exemplo, se $T = O(n)$, então dobrando a entrada o tempo de execução também é dobrado e se $T = O(2^n)$ adicionando-se um bit a entrada n o tempo de execução será dobrado.

Levando em conta o fator tempo, por exemplo, a fim de classificar um algoritmo, temos que o algoritmo é dito *polinomial* se seu tempo de execução é dado por $T(n) = O(n^t)$ para alguma constante *t*; se *t*=0 então é dito *constante*, *linear* se *t*=1 e *quadrático* se *t*=2, e assim por diante.

O algoritmo é dito *exponencial* se $T(n) = O(t^{cn})$ para *t* e *c* constantes e um *n* a entrada. Por exemplo, o algoritmo para fatoração de um inteiro *k* por divisões sucessivas tem complexidade $O(k^{1/2+\varepsilon})$ (onde ε > 0 e pode ser arbitrariamente pequeno). Como $n \approx \log_2^k$, obtém-se $O(e^{\left((\frac{1}{2}+\varepsilon)\ln 2\right)n}$.

Existem algoritmos que possuem complexidade entre polinomial e exponencial e são chamados *subexponenciais*.

Pode-se definir tal complexidade de pelo menos duas formas: Uma faz uso de uma função levando em conta um parâmetro γ, onde $0 \le \gamma \le 1$. Os extremos de γ indicam a complexidade polinomial $(\gamma = 0)$ e a complexidade exponencial $(\gamma = 1)$, os valores de γ entre 0 e 1 são classificados como subexponenciais e além disso indicam o quão próximo a complexidade está de uma complexidade polinomial ou exponencial [K98].

Outra forma de definir de modo mais simples é utilizando a noção de "little *o*". Desta forma um algoritmo cuja a complexidade é subexponencial tem seu tempo de execução limitado por uma função da forma $e^{f(n)}$, onde é o comprimento da entrada e $f(n) = o(n)$. Por exemplo, um algoritmo utilizando $e^{k/\ln\ln k}$ operações seria considerado uma algoritmo subexponencial.

Para um *n* grande, a complexidade temporal pode fazer uma enorme diferença com relação à praticidade do algoritmo. A tabela B.1 a seguir mostra o tempo de execução para diferentes classes de algoritmos. Foram considerados o comprimento de entrada $n=10^6$ e uma máquina capaz de fazer uma instrução por microsegundo (μseg), i. e., 10^6 instruções em 1 segundo, ou ainda $8{,}64 \times 10^{10}$ instruções por dia.

Além disso são ignoradas todas as constantes. Na tabela observa-se, por exemplo, que para um algoritmo com complexidade cúbica é impraticável para uma máquina seqüencial, porém utilizando-se 10^6 processadores em paralelo a mesma tarefa pode ser feita em aproximadamente 12 dias. Já no caso de um algoritmo com complexidade exponencial, executar tal algoritmo também não é prático mesmo utilizando-se todo o poder de computação disponível atualmente e a técnica de paralelização de processamento.

Classe	Complexidade	Número de Operações para $n=10^6$	Tempo de Execução
Polinomial			
Constante	$O(1)$	1	1μSeg
Linear	$O(n)$	10^6	1 Seg
Quadrática	$O(n^2)$	10^{12}	11,6 dias
Cúbica	$O(n^3)$	10^{18}	32000 anos
Exponencial	$O(2^n)$	$10^{301.030}$	$10^{301.006}$ vezes a idade do universo.

Tabela B. 1 - Tempo de execução para diferentes classes de algoritmos.

Considerando um ataque por busca exaustiva da chave, se $K = 2^n$ é o número de possíveis chaves, então o tempo de execução de um ataque deste tipo é dado por $T(n) = O(K) = O(2^n)$. Assim, o tempo de execução é linear com relação a quantidade de possíveis chaves, mas é exponencial com relação ao comprimento da chave, isto explica porque mesmo aumentando um pouco o comprimento da chave pode se causar um impacto considerável na quantidade de tempo necessário para se obter um texto claro utilizando-se esse ataque.

B.3 – Complexidade dos problemas

A Teoria da Complexidade classifica um problema de acordo com o tempo e espaço mínimo necessário para resolver o caso geral utilizando uma máquina de Turing (MT)[32].

Um problema resolvido em tempo polinomial é dito tratável porque pode normalmente ser resolvido em uma quantidade "razoável" de tempo para um tamanho "razoável" de entrada. Problemas que não são resolvidos em tempo polinomial são chamados intratáveis ou apenas "difíceis", porque com o crescimento da entrada *n* , a solução se torna impraticável mesmo nos computadores mais rápidos. Turing provou que há problemas que são indecidíveis, pois é impossível escrever um algoritmo para resolver tais problemas.

A figura B.1 mostra as principais classes de complexidade de problemas e suas relações presumidas (pois sobre tais relações pouco foi provado matematicamente). A *Classe P* (polinomial) consiste de todos os problemas que podem ser resolvidos em tempo polinomial. A *Classe NP* (Polinomial não determinística) é formada por todos os problemas que são resolvidos em tempo polinomial utilizando uma máquina de Turing não determinística, ou seja, capaz de fazer suposições. A classe NP inclui a classe P pois qualquer problema resolvido utilizando uma MT determinística pode ser resolvido utilizando uma MT não determinística.

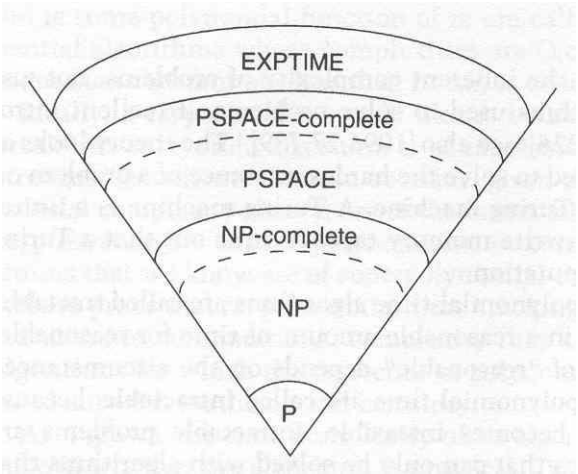


Figura B. 1 - Classes de Complexidade

Se todos os problemas NP fossem resolvidos em tempo polinomial com uma MT determinística então P=NP. Embora seja claro que os problemas NP sejam mais difíceis de resolver que os P, também não foi provado que P≠NP.

[32] Uma máquina de Turing é uma máquina de estado finito com uma fita de memória infinita para escrita e leitura . Essas características tornam a TM um modelo de computação realístico.

Steven Cook [C71]provou que o problema de, dada uma proposição Booleana, saber se existe uma maneira de designar valores verdadeiros as variáveis de maneira a fazer a fórmula verdadeira é um problema *NP-completo*. O que significa que se o mesmo é resolvido em tempo polinomial então P=NP (o chamado *satisfiability problem*).

A próxima classe desta hierarquia é a *classe PSPACE* (Espaço Polinomial). Problemas nesta classe podem ser resolvidos num espaço polinomial, mas não necessariamente num tempo polinomial. PSPACE inclui NP, porém alguns problemas dessa classe são considerados mais difíceis que os problemas da classe NP (o que não foi provado). A classe de problemas *PSPACE-completo* possui a propriedade que se qualquer um desses problemas está em NP, então PSPACE=NP e se estiver em P, então PSPACE=P.

Finalmente, a classe dos problemas *EXPTIME* é a classe dos problemas que são resolvidos em tempo exponencial.

Referências:

[C71] – S. A. Cook, *The Complexity of Theorem Proving Procedures*, Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing, 1971, pp. 151-158.

[K98] – N. Koblitz, *Algebraic Aspects of Cryptography*, Algorithms and Computations in Mathematics, Volume 3, Springer-Verlag Berlin Heidelberg, 1998.

[MOV96] – A. J. Menezes, P. C. Van Oorschot e S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press Series, 1996.

[S49] – C. E. Shannon, *Communication Theory of Secrecy Systems*, Bell System Thecnical Journal, v. 28, n.4, pp. 656-715, 1949.

[S96] – B. Scheneier, *Applied Cryptography Second Edition: Protocols, Algorithms and Source Code in C*, John Wiley & Sons, Inc., 1996.

Apêndice C

Terminologia de Redes Móveis

Este apêndice contém um glossário referente à terminologia utilizada no capítulo 5.

C.1 – Glossário

Agent Advertisement (Anúncio de Agente): O procedimento pelo qual um *nó móvel* toma conhecimento de um roteador de mobilidade (local ou externo).

Agent Discovery: Processo pelo qual o nó móvel detecta a presença de qualquer agente de mobilidade. Ocorre quando um nó móvel recebe um *anúncio de agente*, tanto como resultado de um *broadcast* ou em resposta a uma solicitação.

Agent Solicitation Message (Mensagem de Solicitação de Agente/Operador): Uma mensagem mandada com esperança de receber um anúncio de agente.

Authentication header (Cabeçalho de Autenticação): Um elemento do protocolo de um pacote IP (ou IPv6) que contém a informação necessária permitindo ao receptor ter certeza que o remetente tem acesso a um segredo compartilhado (e , desta forma, que a identidade do remetente é sabida).

Automatic Home agent Discovery: O processo pelo qual um nó móvel pode obter o endereço IP de um agente local em sua rede local, através da transmissão de um pedido de registro para o endereço de *broadcast* da subrede de sua rede local.

Biding (Vínculo): Vincula uma tripla de números que contém o endereço local e endereço móvel do nó móvel, e o *tempo de vida do registro* (quanto tempo os agentes de mobilidade podem usar o vínculo).

Biding Update (Atualização de Vínculo): A mensagem que fornece um novo *vínculo* para uma entidade que precisa saber o novo *endereço móvel* para um nó móvel. A atualização do vínculo contém o endereço local do nó móvel, novo endereço móvel e um novo tempo de vida de registro.

Care-of-Address (Endereço móvel): Um endereço que identifica a atual localização do nó móvel à Internet quando o nó móvel não está ligado à rede local. Ele pode ser designado de forma dinâmica relacionado as interfaces da rede do nó móvel, endereço móvel colocado ou associado ao seu agente externo.

Encapsulation: Processo de incorporação de um pacote IP original dentro de outro pacote IP, fazendo com que os campos do cabeçalho do IP original percam temporariamente seu efeito.

Foreign Agent (Agente externo): É um *agente móvel* de uma rede externa que pode auxiliar o *nó móvel* na recepção de datagramas (pacotes) mandados para *endereço móvel*.

Home Address (Endereço Fixo): Um endereço IP permanente designado a um nó móvel. Este endereço mantém-se inalterado independente da localização do nó móvel na Internet.

Home Agent (Agente local): Um roteador que mantém uma lista de nós móveis registrados numa *lista de visitantes*. É usado para remeter os pacotes endereçados ao nó móvel à rede local apropriada quando o nó móvel está fora da rede local.

IPv4: IP versão 4.

IPv6: IP versão 6, uma substituição ao protocolo IP, atualmente em desenvolvimento pela IETF.

Mobility Agent (Agente Móvel): Um nó (tipicamente um roteador) que oferecer mecanismos de suporte aos nós móveis. Um agente móvel pode ser tanto um agente local ou um agente externo.

Mobilidade: Habilidade de se manter conectado à Internet ao se mover de um ponto a outro.

Nó: um servidor ou um roteador.

Nó móvel : Um servidor ou roteador que pode mudar sua localização de uma rede ou subrede à outra através da Internet. Esta entidade tem pré-determinado um endereço local associado a sua rede local, o qual é usado por outros roteadores a fim de endereçar pacotes ao mesmo, independente da sua localização atual.

Nonce (número aleatório): Um número escolhido aleatoriamente, diferente das escolhas anteriores, inserido na mensagem como defesa para *ataques de repetição*.

Port : Um número relacionado a um pacote de dados para permitir demultiplexação por um protocolo de nível mais alto através do tratamento pelo correto processo de aplicação.

Foreign Network (Rede externa): A rede a qual o nó móvel está conectado quando não está conectado a rede local, e onde o *endereço móvel* é obtido pelo restante da Internet.

Home Network (Rede local): A rede na qual o nó móvel parece acessível, para o resto da Internet, por conta do seu endereço IP associado (endereço fixo).

Redirection: Uma mensagem que tem como intenção causar uma mudança no comportamento de roteamento do nó que está recebendo a mensagem.

Registro: Processo que ocorre quando o nó móvel está fora da sua *rede local* e registra seu endereço móvel com seu agente local. Dependendo do método usado, o nó móvel se registrará diretamente com seu agente local ou através de um agente externo, que passa adiante o registro para o agente local.

Registration Key (Chave de Registro): Uma chave secreta compartilhada entre o nó móvel e um agente externo que pode opcionalmente ser estabelecida durante o registro com o agente externo. Quando mais tarde ao se movimentar e registrar um novo endereço móvel na nova localização, o nó móvel usa a chave de registro compartilhada com o agente externo anterior para enviar a atualização do vínculo autenticada para o novo agente externo.

Registration Lifetime (Tempo de Vida do Registro): Tempo de duração no qual o vínculo é válido.

Remote Redirection (Redirecionamento Remoto): Um redirecionamento mandado de uma fonte que não está na rede local. A fonte pode estar localizada em qualquer lugar na Internet e pode ter intenção maliciosa e ser imprevisível.

Replay Attacks (Ataques de Repetição): Um ataque no qual um entidade maliciosa tenta forjar uma transação utilizando uma transação válida que foi gravada anteriormente quando a mesma estava ocorrendo entre duas entidades do protocolo.

Timestamp: Informação incluída a fim de indicar a hora na qual o elemento do protocolo foi criado ou inicialmente transmitido.

Túnel : Caminho usado por pacotes encapsulados. O modelo conceptual diz que enquanto encapsulado o pacote ficará protegido da rota normal fornecida pelo número IP até que o

mesmo encontre um agente de o retire do encapsulamento. É o caminho que leva os pacotes do agente local para o agente externo.

Tunelamento: Evitar o roteamento normal via Internet cercando (encapsulando) o pacote com um novo cabeçalho IP contendo um endereço de destino IP alternativo.

Visitor list (Lista de Visitantes): Lista de nós móveis visitando um agente externo.

Bibliografia:

[P98] – C. Perkins, *Mobile IP: Design Principles and Practices*, Addisn Wesley Longman, 1998.
[C95] – Yi-na Chen, *A Survey Paper on Mobile IP*, Agosto 1995.
http://www.cis.ohio-state.edu/~jain/cis788-95/mobile_ip/index.html
[P] – C. Perkins, *Mobile Networking Terminlogy*.
http://computer.org/internet/v2n1/terms.htm

Apêndice D

Implementação de Cripto-sistemas baseados no PLDCE

Este capítulo é dedicado à analise de aspectos relacionados à implementação de Cripto-sistemas baseados no PLDCE, como por exemplo: escolha da curva e representação do corpo F_q no qual são executadas as operações sobre a curva elíptica, $E(F_q)$, levando em conta importantes aspectos de segurança.

D.1 – Introdução

Como foi visto anteriormente, o PLDCE tem se mostrado mais difícil de resolver que o PLD sobre corpos finitos ou até mesmo o problema da fatoração de inteiros. Por exemplo, um cripto-sistema baseado numa curva elíptica $E(F_q)$ com um ponto $P \in E(F_q)$ cuja ordem é um primo de 160 bits oferece aproximadamente o mesmo nível de segurança de um DSA com modulo de 1024 bits e RSA com módulo de mesmo comprimento (Seção 4.4, capítulo 4).

A fim de ter uma idéia da eficiência computacional dos sistemas baseados em curvas elípticas serão comparados os tempos para computação de:

(i) kP onde $P \in E(F_{2^m})$, $E(F_{2^m})$ uma curva não-singular, $m \approx 160$, e k inteiro aleatório de 160 bits de comprimento; e

(ii) $\alpha^k \bmod p$, onde p é um primo com 1024 bits de comprimento e k é um inteiro aleatório com 160 bits.

Assumindo que uma multiplicação sobre o corpo F_q, onde $\log_2 q = l$, leva l^2 operações sobre bits, a multiplicação modular em (ii) requer $(1024/160)^2 \approx 41$ operações a mais do que uma multiplicação em (i).

A computação de kP utilizando o processo de duplicar e adicionar os pontos requer neste caso, 160 duplicações e 80 adições em curvas elípticas [R98]. Observando as fórmulas para adição e duplicação de pontos em curvas não-singulares (sub-seção 2.2.3 , capítulo 2) nota-se que são necessárias 1 inversão e 2 multiplicações sobre o corpo.

Assuma que o tempo para se executar uma inversão é equivalente ao de se executar 3 multiplicações [SOOS95, WBVGV96]. Logo, calcular kP requer em média 1200 multiplicações sobre o corpo, i. e. , $1200/41 \approx 29$ multiplicações com módulo de 1024 bits.

Por outro lado, calcular $\alpha^k \bmod p$ utilizando multiplicações e quadrados sucessivos requer em média 240 multiplicações com módulo de 1024 bits. Assim, espera-se que a operação (i) seja 8 vezes mais rápida que a operação (ii). Como multiplicações sobre F_{2^m} são substancialmente mais rápidas e não há comprometimento na segurança em cripto-sistemas que utilizam (i) maiores velocidades na execução podem ser obtidas na prática.

Outra importante conseqüência do uso de grupos menores é o baixo custo e menor consumo de energia, tornando possível implementações em mecanismos pequenos como *smart cards*, *pagers*, *palmtops*, e telefones celulares (Seção 4.5, Capítulo 4). Por exemplo, uma construção ASIC[33] para fazer operações de curvas elípticas sobre $F_{2^{155}}$ [AMV93] requer 12000 portas ocupando apenas 5% da área tipicamente projetada para um processador de *smart card*. Em comparação, um *chip* para fazer multiplicação modular de números de 512 bits tem 50000 portas e chips para multiplicação sobre $F_{2^{593}}$ necessitam de aproximadamente 90000 portas.

Além do baixo custo e menor consumo de energia proporcionado, possibilitando o uso de cripto-sistemas em mecanismos de tamanho reduzido, cripto-sistemas baseados em curvas elípticas tem como outra vantagem o fato que o corpo F_q utilizado na implementação dos mesmos pode ter a representação de seus elementos escolhidas de modo que a aritmética do corpo (adição, multiplicação e inversão) possa ser otimizada. Não sendo este o caso dos cripto-sistemas baseados no PLD pois como já foi visto dependendo da representação o cripto-sistema pode ser enfraquecido.

D.2 – Construção de Cripto-sistemas baseados em Curvas Elípticas

A construção de um cripto-sistema baseado em Curvas Elípticas requer os seguintes passos básicos:

1. Seleção do corpo F_q;
2. Seleção da representação dos elementos do corpo F_q;

[33] ASIC (*Application Specific Integrated Circuit*): É um tipo de circuito integrado cuja função não vem definida pelo fabricante e sim passa a ser definida pelo projetista levando em conta as suas necessidades. [ASIC]

3. Implementação da aritmética em F_q;
4. Seleção de uma curva elíptica E apropriada sobre F_q;
5. Implementação das operações da curva elíptica.

Nesta seção serão explicadas algumas maneiras de representar os elementos do corpo F_q e como isso pode afetar a implementação de cripto-sistemas baseados no PLDCE (D.2.1). Também serão mostradas algumas técnicas para a seleção de curvas elípticas (D.2.2).

D.2.1 – Representação dos elementos de F_q

A representação usada para os elementos de F_q pode ter impacto significante na praticidade, custo e velocidade de um cripto-sistema baseado em curvas elípticas (CCE), porém tal escolha até o momento não mostrou afetar a segurança do mesmo.

1. **Curvas Elípticas sobre F_p,**: A fim de minimizar o tempo de execução de uma multiplicação modular, o primo p deve ser um primo de Mersenne, i. e., um primo da forma $p = 2^k$-1 [C92]. Para um implementação em software do ECDSA sobre F_p consulte [WMPW98], e [C92] para uma implementação de aritmética em curvas elípticas sobre F_{p^m}, onde p é um primo da forma $2^k \pm c$ para algum c pequeno.
2. **Curvas Elípticas sobre** F_{2^m}: O corpo F_{2^m} pode ser visto como um espaço vetorial de dimensão ***m*** sobre F_2. Isto é, existe um conjunto com m elementos $\{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$ em F_{2^m} tal que cada $\alpha \in F_{2^m}$ pode ser unicamente escrito na forma

$$\alpha = \sum_{i=2}^{m-1} a_i \alpha_i \text{ , onde } a_i \in \{0,1\} \qquad \textbf{(D.1)}$$

Pode–se então representar α como o vetor binário $(a_0, a_1, \ldots, a_{m-1})$. A adição dos elementos do corpo é feita através do ou-exclusivo entre as representações vetoriais dos mesmos. Existem muitas bases diferentes de F_{2^m} sobre F_2.

a) <u>Bases Trinomiais</u>

Se $f(x)$ é um polinômio irredutível de grau m sobre F_{2^m}, então o corpo F_{2^m} pode ser representado como o conjunto de polinômios de grau menor que m sobre F_2, onde a multiplicação dos polinômios é feita modulo $f(x)$, i.e., a representação (D.1), onde $\alpha_i = x^i$, $0 < i < m$-1. Tal representação é chamada *representação em base polinomial*.

Uma *representação em base trinomial* é uma representação em base polinomial onde o polinômio $f(x)$ tem a forma $f(x) = x^m + x^k + 1$. Tal representação tem como vantagem o fato de que a redução modulo $f(x)$ pode ser feita de modo eficiente, tanto em software quanto em hardware. Em [SOOS95] uma descrição detalhada da aritmética sobre o corpo $F_{2^{155}}$ usando a representação em base trinomial é dada.

b) Base normal ótima

Quando os cripto-sistemas baseados em curvas elípticas foram propostos inicialmente, a representação em base normal ótima (BNO) foi considerada a mais eficiente, com respeito tanto a velocidade quanto a complexidade da arquitetura do *hardware*. Nos últimos anos a representação trinomial tem se mostrado mais rápida nas implementações em *software*. Foi mostrado recentemente [D97] que a combinação destas duas representações (trinomial e BNO) utilizando as vantagens de cada uma dessas representações procura alcançar máxima eficiência.

Apesar da representação matemática em base normal parecer complicada sua implementação em hardware ou software é bem simples uma vez que só utiliza as operações AND, XOR e rotações, as quais são as mais rápidas operações executadas por qualquer microprocessador, tornando assim a representação BNO muito atraente.

Como visto no capítulo 2 os elementos de um corpo podem ser representados usando polinômios sobre uma variável arbitrária (x na maioria das vezes). Um elemento β sobre um corpo F_{p^m} tem representação polinomial,

$$\beta = a_n x^n + \ldots + a_1 x + a_0 \qquad \textbf{(D.2)}$$

onde $n < m$, $a_i \in F_p$.

Uma base normal pode ser formada usando o conjunto:

$$\{\beta, \beta^p, \beta^{p^2}, \ldots, \beta^{p^{m-1}}\} \qquad \textbf{(D.3)}$$

ou seja, considerando *p* = 2, uma *base normal* de F_{2^m} sobre F_2 será da forma

$$\{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\} \quad \textbf{(D.4)}$$

onde $\beta \in F_{2^m}$.

Qualquer elemento num corpo pode ser representado utilizando o formato base normal. Como todas as representações são isomórficas, pode se obter o mesmo resultado matemático com diferentes operações. Um elemento α pode ser representado em base normal como

$$\alpha = \alpha_{m-1}\beta^{2^{m-1}} + \ldots + \alpha_2\beta^{2^2} + \alpha_1\beta^2 + \alpha_0\beta \quad \textbf{(D.5)}$$

isto é,

$$\alpha = \sum_{i=0}^{m-1} \alpha_i \beta^{2^i} \quad \textbf{(D.6)}$$

como a operação de elevar ao quadrado é linear em F_{2^m}, tem-se

$$\alpha^2 = \sum_{i=1}^{m-1} a_i \beta^{2^{i+1}} = \sum_{i=0}^{m-1} a_{i-1} \beta^{2^i} = (a_{m-1}, a_0, \ldots, a_{m-2}) \quad \textbf{(D.7)}$$

assim, a representação em base normal de F_{2^m} tem como vantagem o fato de que elevar um elemento do corpo ao quadrado é o mesmo que rotacionar sua representação vetorial, uma operação facilmente implementada em hardware. Isso deve-se a

$$\left(\beta^{2^i}\right)^2 = \beta^{2^{i+1}} \quad \textbf{(D.8)}$$

$$\beta^{2^m} = \beta \quad \textbf{(D.9)}$$

manipulações matemáticas simples explicam facilmente (D.8), já (D.9) provém das regras dos corpos finitos e é similar ao Teorema de Fermat.

Com relação a adição a mesma é feita da mesma forma na representação polinomial uma vez que os coeficientes são 0 ou 1; portanto a adição consiste apenas de um simples ou-exclusivo.

A multiplicação utilizando a representação em base normal é mais complicada. A base é a mesma de qualquer sistema matemático, os coeficientes são somados àqueles

que tem mesma potência de *x*. O fato de ter a maioria dos termos nulos faz com que maior atenção seja requerida na multiplicação sobre esta base.

A fim de mostrar como é feita a multiplicação, considere dois elementos sobre F_{2^m} :

$$A = \sum a_i \beta^{2^i} \qquad \textbf{(D.10)}$$

e

$$B = \sum b_j \beta^{2^j} \ . \qquad \textbf{(D.11)}$$

A multiplicação formal é dada por

$$C = A \bullet B = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \beta^{2^i} \beta^{2^j} \ . \qquad \textbf{(D.12)}$$

Escrevendo

$$C = \sum_{k=0}^{m-1} c_k \beta^{2^k} \ , \qquad \textbf{(D.13)}$$

pela definição de um elemento na base normal, a soma dupla de (D.12) deve ser mapeada na soma simples de (D.13). De fato, deve-se ter cada termo de multiplicação cruzada mapeado numa soma em termos de base, i.e.:

$$\beta^{2^i} \beta^{2^j} = \sum_{k=0}^{m-1} \lambda_{ijk} \beta^{2^k} \ . \qquad \textbf{(D.14)}$$

O coeficiente λ_{ijk} é chamado "matriz lambda" ou "tabela de multiplicação". Substituindo (D.14) em (D.12) pode se achar uma solução para cada coeficiente c_k de β^{2^k} na equação (D.13), dado pela soma dupla

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \lambda_{ijk} \ . \qquad \textbf{(D.15)}$$

Em [MOVW88] é provado matematicamente que (D.15) pode ser transformada a fim de requerer apenas λ_{ij0} para o calculo de todos os c_k, reduzindo assim a quantidade de trabalho necessário para a construção da matriz λ (tabela de multiplicação) . Assim,

$$c_k = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} a_{i+k} b_{j+k} \lambda_{ij0} \,. \qquad \textbf{(D.16)}$$

Nota-se desta forma que é apenas necessário o deslocamento das entradas tornando a implementação fácil, além de se reduzir a quantidade de memória requerida.

A chamada *representação em base normal ótima* [MOVW88], como já foi dito, fornece uma implementação mais eficiente de aritmética sobre corpos.

A fim de definir o que vem a ser uma base normal ótima considere $(a_0, a_1, \ldots, a_{m-1})$ e $(b_0, b_1, \ldots, b_{m-1})$ os vetores coordenadas dos elementos *A* e *B* do corpo F_{2^m}, respectivamente (seção D.2). Se *C* = *AB*, então *C* tem vetor coordenada $(C_0, C_1, \ldots, C_{m-1})$, onde C_k é uma forma bilinear em a_i e b_j, $0 \le i, j \le m-1$. Verifica-se facilmente que C_k é obtido através do deslocamento cíclico de *k* posições dos elementos de C_0.

Seja *C*(*N*) o número de termos não nulos em C_0, e portanto C_i, $0 \le i \le m-1$. Assim, *C*(*N*) satisfaz a inequação $C(N) \ge 2m-1$. Se *N* satisfizer essa inequação com igualdade, então a base em questão é dita uma base normal ótima. Logo, uma base normal ótima é aquela que possui quantidade mínima de elementos não nulos na matriz lambda; tal quantidade é denominada *Complexidade da Tabela de Multiplicação*. Assim num corpo de característica 2 a Complexidade da Tabela de Multiplicação é 2*m*-1.

Existem dois tipos de representação em base normal ótima sobre F_{2^m}, denominados Tipo I e Tipo II. A diferença básica entre esses dois tipos se baseia em como se encontra quais bits estão colocados na matriz lambda. No Tipo I apenas é necessário armazenar um vetor; para o Tipo II é necessário armazenar dois vetores. A seguir esses dois tipos serão descritos:

Tipo I

As regras para este tipo são:

1) *m*+1 deve ser um primo;

2) 2 deve ser elemento primitivo em Z_{m+1}.

É necessário encontrar o produto $\beta^{2^i} \cdot \beta^{2^j}$ em (D.14) uma vez que como foi visto em (D.16) pode-se utilizar a matriz lambda com *k*=0 para calcular todos os termos cruzados, tornando-se necessário apenas a resolução de

$$\beta^{2^i} \beta^{2^j} = \beta \qquad \textbf{(D.17)}$$

Também existe o caso especial em que

$$\beta^{2^i} \beta^{2^j} = 1 \qquad \textbf{(D.18)}$$

quando $2^i + 2^j \equiv 0 (\mathrm{mod}\, m+1)$. Para que o que foi colocado acima funcione de modo apropriado β deve ser um elemento de ordem *m* + 1 em F_{2^m}. Como 2 é um elemento primitivo de Z_{m+1}, $2^i \bmod m+1$ gera todos os elementos entre 1 e *m* com *i* variando de 0 à *m* – 1. Assim, β^2 é apenas outra maneira de obter todas as potências de β que geram a base em questão.

O modo mais fácil de resolver as equações (D.17) e (D.18) é rescrever as mesmas módulo *m* + 1 e resolver:

$$2^i + 2^j = 1 \bmod m+1 \qquad \textbf{(D.19)}$$

e

$$2^i + 2^j = 0 \bmod m+1 \qquad \textbf{(D.20)}$$

começando com *i* = 0: $2^0 = 1$ e $2^m = 1$ devido ao Teorema de Fermat uma vez que $m+1$ é primo. Para *i* = 0 a equação (D.19) não pode ter solução, apenas a equação (D.20) pode. Extraindo a raiz quadrada de $2^m = 1$, acha-se

$$2^{m/2} = \pm 1 \bmod m+1. \qquad \textbf{(D.21)}$$

Sabe-se que $2^0 = 1 \mathrm{mod}(m+1)$ e como 2 gera todos os números $\mathrm{mod}(m+1)$, a equação (D.21) tem apenas uma escolha

$$2^{m/2} = -1 \mathrm{mod}(m+1) \qquad \textbf{(D.22)}$$

Assim, para *i*=0, a equação (D.20) tem a seguinte solução

$$2^0 + 2^{m/2} = 0 \bmod (m+1) \qquad \textbf{(D.23)}$$

Foi dito no início desta seção que o tipo I necessita de um único vetor para armazenar informações sobre os produtos cruzados dos termos. Não é necessário armazenar nenhuma outra solução da equação (D.20), pois a equação (D.23) pode ser multiplicada por 2 e mantendo-se nula do lado direito. Para cada *i*, o elemento λ_{ij} não nulo da equação (D.20) será sempre:

$$j = m/2 + i \bmod m \qquad \textbf{(D.24)}$$

Os valores da equação (D.19) devem ser tabelados uma vez e então usados para verificar as quantidades corretas de deslocamento. O primeiro valor para *i* = 1 é encontrado facilmente, uma vez que

$$\begin{aligned} 2 + 2^j &= 1 \\ 2^j &= -1, \end{aligned} \qquad \textbf{(D.25)}$$

então, da equação (D.22), $j = m/2$.

Depois desses procedimentos é necessário construir as tabelas log e antilog a fim de encontrar $2^i \bmod(m+1)$ e *j* a partir de $1 - 2^i \bmod(m+1)$.

Exemplo D.1: Considere *m*=4. A construção da tabela antilog consiste apenas em multiplicar *i* por 2 mod 5 a cada entrada.

i	0	1	2	3	4
2^i	1	2	4	3	1

Tabela D. 1 - Tabela antilog para *m* = 4.

Já a construção da tabela log usa os valores 2^i como índices e os coloca na entrada *i*.

2^i	1	2	3	4
i	0	1	3	2

Tabela D. 2 - Tabela log para *m*=4.

□

Tipo II

Neste caso tem-se o mesmo número de termos do tipo I, porém ambos os conjuntos estão misturados, necessitando assim de dois conjuntos de vetores.

O tipo II pode ser subdividido em dois outros tipos que serão denominados aqui tipo IIa e tipo IIb.

A representação base normal ótima do tipo II sobre F_{2^m} pode ser criada se:

1) $2m + 1$ é um primo e

2a) 2 é elemento primitivo em Z_{2m+1}

ou

2b) $2m+1 \equiv 3 \bmod 4$ e 2 gera resíduos quadráticos em Z_{2m+1}.

A fim de gerar uma representação em base normal ótima de tipo II deve-se utilizar dois elementos, cada um de um corpo diferente.

Inicialmente, considere um elemento γ de ordem em $F_{2^{2^m}}$; este elemento será usado para obter o elemento β pertencente a F_{2^m}. Na prática não é necessário encontrar o elemento γ, apenas sua representação será utilizada a fim de construir-se a matriz λ. A soma $\gamma + \gamma^{-1}$ fornece o primeiro elemento β da base normal ótima a ser construída [MOVW88].

Os termos de produto cruzado de $\beta^{2^i} \cdot \beta^{2^j}$ são:

$$\beta^{2^i} \cdot \beta^{2^j} = (\gamma^{2^i} + \gamma^{-2^i})(\gamma^{2^j} + \gamma^{-2^j}) = (\gamma^{2^i+2^j} + \gamma^{-(2^i+2^j)}) + (\gamma^{2^i-2^j} + \gamma^{-(2^i-2^j)}) \quad \textbf{(D.26)}$$

Usando o fato que $\gamma^{2^k} + \gamma^{-2^k} = (\gamma + \gamma^{-1})^{2^k}$, tem-se:

$$\beta^{2^i} \cdot \beta^{2^j} = \beta^{2^k} + \beta^{2^{k'}} \qquad \text{se } 2^i \neq 2^j \bmod(2m+1) \qquad \textbf{(D.27)}$$

$$= \beta^{2^k} \qquad \text{se } 2^i \equiv 2^j \bmod(2m+1); \qquad \textbf{(D.28)}$$

Nas equações (D.27) e (D.28), k e k' são duas possíveis soluções para a multiplicação de quaisquer dois elementos da base. Esta base tem o menor número possível de elementos e isto é o que a faz ser ótima.

No caso em que $2^i \equiv 2^j \bmod(2m+1)$, os termos $\gamma^0 + \gamma^0$ se cancelam na expressão (D.26). Já no caso em que $2^i \neq 2^j \bmod(2m+1)$, pelo menos uma dessas equações:

$$\begin{aligned} 2^i + 2^j &= 2^k \bmod(2m+1) \\ 2^i + 2^j &= -2^k \bmod(2m+1) \end{aligned} \quad \textbf{(D.29)}$$

terá uma solução, e pelo menos uma das seguintes equações também terá solução:

$$\begin{aligned} 2^i - 2^j &= 2^{k'} \bmod(2m+1) \\ 2^i - 2^j &= -2^{k'} \bmod(2m+1) \end{aligned} \quad \textbf{(D.30)}$$

No caso de $2^i = \pm 2^j \bmod(2m+1)$, pelo menos uma das quatro equações a seguir terá solução:

$$2^i + 2^j = 2^k \bmod(2m+1) \quad \textbf{(D.31)}$$

$$2^i - 2^j = 2^k \bmod(2m+1) \quad \textbf{(D.32)}$$

$$2^i + 2^j = -2^k \bmod(2m+1) \quad \textbf{(D.33)}$$

$$2^i - 2^j = -2^k \bmod(2m+1) \quad \textbf{(D.34)}$$

Nas equações (D.29) e (D.30) existem duas possíveis soluções e nas equações de (D.31) a (D.34) existe apenas uma solução. Nota-se que todas essas equações são similares, assim ao invés de se trabalhar com diferentes conjuntos de equações pode-se combinar todas as equações e trabalhar com apenas um grupo de quatro equações.

A fim de construir a matriz λ, faz-se k=0 e acha-se as soluções para:

$$2^i + 2^j = 1 \quad \textbf{(D.35)}$$

$$2^i + 2^j = -1 \quad \textbf{(D.36)}$$

$$2^i - 2^j = 1 \quad \textbf{(D.37)}$$

$$2^i - 2^j = -1 \quad \textbf{(D.38)}$$

Exemplo D.2: Como exemplo para o tipo IIa considere $2m+1=19$, i.e., $m=9$ (ordem do corpo) que será o comprimento da matriz λ. Inicialmente são calculadas as

tabelas log e antilog. A tabela antilog (tabela D.3) toma o índice i e retorna $l = 2^i \bmod(2m+1)$ e a tabela log toma o índice *l* e retorna *i* (tabela D.4).

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2^i	1	2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1

Tabela D. 3 - Potências de 2^i mod 19 (Antilog)

I	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\text{Log}_2(i)$	0	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

Tabela D. 4 - Log na base 2 de *i* modulo 19 (log)

Fazendo $i = 1$ nas equações de (D.35) a (D.38), as escrevendo modulo 19 e subtraindo 2 de ambos os lados, tem-se:

$2^j = -1 = 18 \Rightarrow j = 9$

$2^j = -3 = 16 \Rightarrow j = 4$

$-2^j = -1 \Rightarrow 2^j = 1 \Rightarrow j = 0$

$-2^j = -3 \Rightarrow 2^j = 3 \Rightarrow j = 13$

Como a matriz λ possui apenas 9 elementos por coluna, as soluções para j devem estar no intervalo de 0 a 8, assim observa-se que apenas dois valores de *j* encontrados estão neste intervalo. Desta forma tem-se os primeiros elementos na matriz λ: $\lambda_{1,0} = 1$ e $\lambda_{1,4} = 1$. Todos os outros elementos , $\lambda_{1,j}$ devem ser 0. Continuando desta forma podem ser encontrados dois valores de *j* para cada valor de *i*, sendo estes os valores não nulos da matriz λ. Inicialmente são calculadas as tabelas log e antilog.

Uma descrição de implementação em hardware para aritmética sobre $F_{2^{155}}$ é dada em [AMV93].

Utilizando a representação em base normal tem-se como vantagem o fato que a raiz quadrada dos elementos em F_{2^m} pode ser calculada eficientemente. Isto é útil na recuperação de pontos utilizando a seguinte técnica de compressão:

Seja (x_1, y_1) um ponto sobre a curva elíptica $y^2 + xy = x^3 + ax^2 + b$ definida sobre F_{2^m}. Define-se $\tilde{y}_1 = 0$ se $x_1 = 0$ e $\tilde{y}_1$ é o bit mais a direita do elemento $y_1 x_1^{-1}$ se $x_1 \neq 0$. Assim, *P* pode ser representado agora por $(x_1, \tilde{y}_1)$. Dados x_1 e $\tilde{y}_1$, y_1 é facilmente recuperado a partir da técnica de Menezes e Vanstone [MV93].

c) Usando subcorpos

Suponha que $m = lr$, onde l é pequeno (e.g., l=8 ou l=16). Então o corpo F_{2^m} pode ser visto como uma extensão do corpo de grau r sobre F_{2^l}. Se $\{\alpha_0, \alpha_1, \ldots, \alpha_{r-1}\}$ é uma base para F_{2^m} sobre F_{2^l}, então cada elemento $\alpha \in F_{2^m}$ pode ser unicamente escrito na forma

$$\alpha = \sum_{i=0}^{r-1} a_i \alpha_i \qquad \textbf{(D.39)}$$

onde $a_i \in F_{2^l}$.

A multiplicação no corpo em F_{2^m} neste caso envolve várias operações sobre o corpo F_{2^l}. Como l é pequeno, a aritmética em F_{2^l} pode ser significantemente acelerada, por exemplo precomputando as tabelas de "log" e "antilog".

D.2.2 – Seleção da curva apropriada

Uma curva elíptica apropriada é aquela que obedece às seguintes condições:

(i) Resistir ao ataque Pollard ρ, i.e., $\#E(F_q)$ deve ter como fator um primo n suficientemente grande (por exemplo, $n > 2^{160}$);

(ii) Resistir ao ataque Semaev-Smart-Satoh-Araki [SA98, S98, S99], i.e., $\#E(F_q)$ não deve ser igual a q;

(iii) Resistir ao ataque de redução MOV [MOV93], i.e., n não deve dividir $q^k - 1$ para todo $1 \le k \le C$, onde C é grande o suficiente para tornar computacionalmente inviável achar o logaritmo discreto em $F^*_{q^C}$ (C = 20 é suficiente na prática).

A seguir serão comentadas quatro técnicas utilizadas na seleção de uma curva elíptica apropriada.

a) Usando o Teorema de Hasse: Esta técnica pode ser usada para selecionar curvas sobre F_{2^m} onde m é divisível por um inteiro pequeno $l > 1$.

Se E é uma curva elíptica definida sobre F_q, então E pode ser vista como uma curva elíptica sobre qualquer extensão F_{q^k} de F_q; $E(F_q)$ é um subgrupo de $E(F_{q^k})$. O

Teorema de Hasse [B99] permite calcular $\#E(F_{q^k})$, a partir de $\#E(F_q)$, da seguinte forma:

Seja $t = q + 1 - \#E(F_q)$, então $\#E(F_{q^k}) = q^k + 1 - \alpha^k - \beta^k$, onde α e β são números complexos determinados a partir da fatoração de $1 - tT + qT^2 = (1 - \alpha T)(1 - \beta T)$.

A fim de selecionar uma curva apropriada sobre F_{2^m} , inicialmente seleciona-se uma curva elíptica sobre um corpo F_{2^l} menor, onde $l \mid m$, computa-se $\#E(F_{2^l})$ exaustivamente, e então usa-se o Teorema de Hasse para determinar $\#E(F_{2^m})$. Se as condições (i), (ii) e (iii) (com $q = 2^m$) não forem satisfeitas outra curva é selecionada e o processo é repetido.

b) Método Global

É um método de seleção que consiste em escolher uma curva elíptica definida sobre um corpo numérico e então reduzi-la módulo um ideal primo [BM80], tal que a curva resultante sobre um corpo finito satisfaça as condições (i), (ii) e (iii). Por exemplo, inicia-se com a equação $y^2 = x^3 + ax + b$ com $a, b \in Q$ e a partir daí considera-se a mesma equação modulo p para p um primo grande, onde se deseja que N_p, o número de pontos na curva sobre F_p, seja um primo ou um primo multiplicado por um fator pequeno.

Aqui N_p sempre é divisível por $\#E_{tors}$, número de pontos de ordem finita da curva original sobre Q. Mas a razão $N_p / (\#E_{tors})$ será freqüentemente um primo. Nota-se que $\#E_{tors} \leq 16$ pelo Teorema de B. Mazur [M77], e $\#E_{tors} = 1$ para a maioria de curvas aleatórias. Para mais discussões sobre a primalidade de N_p consultar [K88].

Exemplo D.3 : Considere a curva

$$y^2 = x^3 - m^2 x \qquad \textbf{(D.40)},$$

onde m é um parâmetro inteiro. Considere agora esta curva módulo p tal que p não divide m e $p \equiv 1 (\mathrm{mod}\, 4)$ (note que se $p \equiv 3 (\mathrm{mod}\, 4)$ a curva é supersingular) .

Gauss encontrou uma fórmula simples para N_p. Inicialmente ele escreveu p como a soma de dois quadrados, $p = a^2 + b^2$, onde sem perda de generalização a é impar. O sinal de a é encontrado requerendo que $a + b \equiv \left(\frac{m}{p}\right) (\mathrm{mod}\, 4)$. Assim, $N_p = p + 1 - a$. Como a curva (D.40) possui quatro pontos ((0,0), $(\pm m, 0)$, ∞), segue-se que 4 divide N_p. Porém $N_p / 4$ não é um primo, freqüentemente.

□

c) Método da Multiplicação Complexa (MC)

Este método permite a escolha da ordem da curva antes que a mesma seja explicitamente construída. Desta forma as ordens podem ser geradas e testadas a fim de satisfazer as condições (i), (ii) e (iii), sendo a mesma construída apenas quando as três condições são obedecidas. Para curvas elípticas sobre F_p, o método MC também é chamado método Atkin-Morain [M91]; para curvas sobre F_{2^m} é chamado Método Lay-Zimmer [LZ94].

d) Escolhendo uma curva aleatoriamente

Outra maneira de escolher uma curva apropriada E sobre F_q é escolher os parâmetros a, $b \in F_q$ aleatoriamente (levando em conta que $4a^3 + 27b^2 \neq 0$ se q é impar e $b \neq 0$ se q é uma potência de dois). Selecionada a curva calcula-se $u = \#E(F_q)$ e fatora-se o mesmo. Tal processo é repetido até que as condições (i), (ii) e (iii) são satisfeitas. Mostra-se pelo Teorema de Lenstra [L87] que se os coeficientes a e b são escolhidos uniformemente de forma aleatória, então as ordens das curvas resultantes são aproximadamente distribuídas de forma uniforme.

A fim de calcular a ordem da curva, em 1985, Schoof [S85] apresentou um algoritmo com complexidade polinomial que calculava o número de pontos de uma curva definida sobre F_q para q impar; mais tarde tal algoritmo foi estendido por Koblitz [K90] para o caso em que q uma potência de 2. O algoritmo de Schoof não é eficiente para valores de q usados na prática (i.e., $q > 2^{160}$) tendo em pior caso tempo de execução de $O((\log q)^8)$ operações em bits. Nos últimos anos muitos trabalhos foram feitos para melhorar e refinar o algoritmo de Schoof. Um dos trabalhos [LM95] baseado em idéias de Atkin, Elkies e Couveignes mostraram tempos de execução de 4 e

3 minutos num DecAlpha 3000/500, no calculo da ordem de curvas sobre $F_{2^{155}}$ e sobre corpos de ordem prima de 155 bits, respectivamente.

Em Junho 1998, A. Joux e R. Lercier computaram a ordem de uma curva sobre $F_{2^{1663}}$ em 330 dias numa DEC Alpha; foi utilizado para isto o algoritmo de Schoof-Elkies-Atkin incorporando novas idéias de Lecier [L96]. Curvas elípticas apropriadas para uso criptográfico sobre corpos tão grandes quanto $F_{2^{196}}$ podem ser geradas aleatoriamente em algumas horas numa estação de trabalho [L97].

Referências:

[AMV93] – G. Agnew, R. Mullin e S. Vanstone, *An Implementation of Elliptic Curve Criptosystems over* $F_{2^{155}}$, IEEE Journal of Cryptology, 3, pp. 804-813, 1993.

[B99] – P. S. L. M. Barreto, *Curvas Elípticas e Criptografia: Conceitos e Algoritmos*, Junho 1999.

[BM80] – G. Birkhoff e S. MacLane, Álgebra Moderna Básica, 4ª Edição, Guanabara Dois, 1980.

[BP98] – D. Bailey e C. Paar, *Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms*, Advances in Cryptology – CRYPTO '97, Lecture Notes in Computer Science, 1462, pp. 472- 485, Springer-Verlag, 1998.

[C92] - R. Crandall, *Method and Apparatus for Public Key Exchange in a Cryptographic System*, U. S. Patent Number 5.159.632, Outubro 1992.

[D97] – D. Dahm, personal communication and `sci.crypt` postings, Setembro 1997.

[K88] – N. Koblitz, *Primality of the Number of Points on an Elliptic Curve over a Finite Field*, Pacific Journal of Mathematics, 131, pp. 157-165, 1988.

[K90] – N. Koblitz, *Constructing Elliptic Curve Cryptosystems in Characteristic 2*, Advances in Cryptology –CRYPTO '90, Lectures Notes in Computer Science, 537,pp.156-167, Springer-Verlag, 1990.

[LZ94] – G. Lay e H. Zimmer, *Constructing Elliptic Curves with Given Group Order over Large Finite Fields*, Algorithmic Number Theory, Lectures Notes in Computer Science, 877, Springer-Verlag, pp. 250-263, 1994.

[L87] – H. W. Lenstra, *Factoring Integers with Elliptic Curves*, Annals of Mathematic, 126, pp. 649-673, 1987.

[L96] – R. Lecier, *Computing Isogenies in* F_{2^n}, Algorithmic Number Theory, Proceedings Second Intern. Symp., ANTS-II, Lectures Notes in Computer Science, 1122, pp. 197-212, Springer-Verlag, 1996.

[L97] - R. Lecier, *Finding Good Random Elliptic Curves for Cryptosystems Defined* F_{2^n}, Advances in Cryptology – EUROCRYPT '97, Lectures Notes in Computer Science, 1233, pp. 379-392, Springer-Verlag, 1997.

[LM9544] – R. Lercier e F. Morain, *Counting the Points on Elliptic Curves over Finite Fields: Strategies and Performances*, Advances in Cryptology – EUROCRYPT '95, Lectures Notes in Computer Science, 921, pp. 79-94, Springer-Verlag, 1995.

[M77] – B. Mazur, *Modular Curves and Eisenstein Ideal*, Inst. Hautes Études Sci. Publ. Math., 47, pp. 33-186, 1977.

[M91] – F. Morain, *Bulding Cyclic Elliptic Curves in Cryptography*, Advances in Cryptology – EUROCRYPTO '91, Lectures Notes in Computer Science, 547,pp.328-336, Springer-Verlag, 1991.

[MOV93] – A. Menezes, T. Okamoto e S. Vanstone , *Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field*, IEEE Transactions in Information Theory, 39, pp. 1639-1646, 1993.

[MOVW88] - R. Mullin, I. Onyszchuk, S. Vanstone e R. Wilson, *Optimal Normal Bases in* $GF(p^n)$, Discrete Applied Mathematics, 22, pp. 149-161, 1988.

[MV93] - A. Menezes e S. Vanstone, *Elliptic Curve Cryptosystems and their Implementation*, Journal of Cryptology, 6, pp. 209-224, 1993.

[R98] – M. Rosing, *Implementing Elliptic Curve Cryptography*, Manning Publications Co., 1998.

[S87] – R. Schoof, *Nonsingular Plane Cubic Curves, Journal of Combinatorial Theory*, Series A, 46, pp. 183-211, 1987.

[S85] – R. Schoof, *Elliptic Curves over Finite Fields and the Computation of Square Roots mod p*, Mathematics of Computation, 44, pp. 483-494, 1985.

[S98] – I. Semaev, *Evaluation of Discrete Logarithms in a Group of p-torsion Points of na Elliptic Curve of Characteristic p*, Mathematics of Computation, 87, pp. 353-356, 1998.

[S99] – N. Smart, *The Discrete Logarithm Problem on Elliptic Curves of Trace One*, Journal of Cryptology, 12, pp. 193-196, Springer-Verlag New York Inc., 1999.

[SA98] – T. Satoh e K. Araki, *Fermat Quotients and the Polynomial Time Discrete Log Algorithm for Anomalous Elliptic Curves*, Commetarii Mathematici Universitatis Sancti Pauli, 47, pp. 81-92, 1998.

[SOOS95] – R. Schroeppel, H. Orman, S. O'Malley e O. Spatscheck, *Fast Key Exchange with Elliptic Curve Systems*, Advances in Cryptology – CRYPTO '95, Lectures Notes in Computer Science, 963, pp. 43-56, Springer-Verlag, 1995.

[W69] – W. Waterhouse, *Abelian Varieties over Finite Fields*, Ann. Sci. École Norm. Sup., 4 ª Série, 2, pp. 521-560, 1969.

[WBVGV96] – E. De Win, A. Bosselaers, S. Vandenberghe, P. De Gersem e J. Vandewalle, *A Fast Software Implementation for Arithmetic Operations in GF(2^n)*, Advances in Cryptology – ASIACRYPT '96, Lectures Notes in Computer Science, 1163, pp. 65-76, Springer-Verlag, 1996.

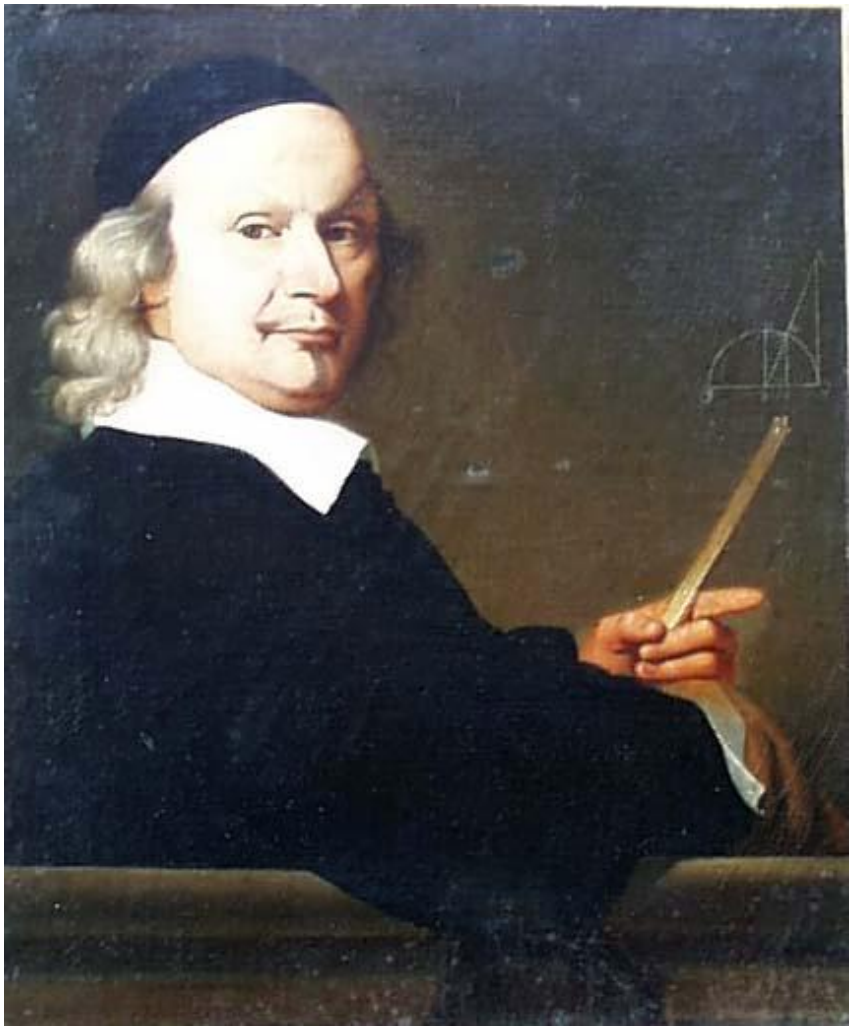
[WMPW98] – E. De Win, S. Mister, B. Preneel e M. Wiener, *On the Performance of Signature Schemes based on Elliptic Curves*, Algorithm Number Theory, Proceedings Third Intern. Symp., ANTS-III, Lectures Notes in Computer Science, 1423, pp. 252-266, Springer-Verlag, 1998.

Página na WEB

http://asic.korea.ac.kr/A-IntroAbout.html

Apêndice E

JOHN WALLIS



23 de Novembro de 1616 – Ashford, Kent, Inglaterra
28 de Outubro de 1703 – Oxford, Inglaterra.

John Wallis, contemporâneo de Newton e um dos primeiros grandes matemáticos ingleses, nasceu em Ashford, Inglaterra em 22 de Novembro de 1616. Em 1630 foi para a Escola de Felstead onde foi educado e aprendeu latin, grego e hebraico. Aos 15 anos, durante uma de suas férias, se deparou com um dos livros de aritmética de seu irmão e ficou curioso sobre todos aqueles símbolos [web01]. Com o auxílio do mesmo passou a dominar a matéria após duas semanas [web02].

Posteriormente foi para o Emmanuel College, Cambridge, onde deveria cursar medicina. Lá se tornou um dos primeiros pupilos de Francis Glissom a proclamar a descoberta de Harvey sobre a circulação do sangue. Porém, devido a ter seu interesse centralizado em matemática e teologia, deixou o estudo da medicina. Como nesta época, em Cambridge, não havia ninguém que pudesse orientá-lo em matemática, ele passou a estudar teologia. Graduou-se e obteve o grau de mestre em Teologia, sendo ordenado em 1640. Em 1660 tornou-se capelão de Charles II.

Apesar de não ter acesso a uma educação formal em matemática, Wallis não abandonou a sua grande paixão, sendo um autodidata nesta área.

Por ser perito em criptografia Wallis prestou grande serviço aos parlamentaristas do Partido Puritano decodificando os despachos reais durante a guerra civil. Acredita-se que devido a este fato ele foi indicado para a cátedra de geometria em Oxford em 1649, posição certa para Peter Turner o qual foi demitido pelo Parlamento. Wallis ficou nesta cátedra por mais de 50 anos até a sua morte.

Wallis fez parte de um grupo interessado em ciência natural e experimental que começou a ser formado em Londres. Tal grupo deu origem a *Royal Society*, sendo Wallis um dos seus fundadores[web03].

Na meia idade Wallis desenvolveu a habilidade em resolver problemas aritméticos mentalmente e em uma ocasião, enquanto estava dormindo, achou a parte inteira da raiz quadrada de 3×10^{40}. Muitas horas depois ele escreveu o resultado de memória. Em outra ocasião ele extraiu a raiz de um número com 53 dígitos mentalmente.

Wallis contribuiu substancialmente para a origem do cálculo, sendo um dos matemáticos ingleses mais influentes antes de Newton. Ele estudou trabalhos de Kepler, Cavalieri, Roberval, Torricelli e Descartes. Tomando como base os trabalhos desses matemáticos contribuiu enormemente introduzindo novas idéias na área do Cálculo. Ainda na área matemática, além das suas importantes contribuições na área de Cálculo, Wallis trabalhou com seções cônicas, sistematizou os métodos de análise de Descartes, trabalhou em equações de curvas, efetuou soluções para o problema da cicloide e deu o primeiro uso sistemático de fórmulas em álgebra.

Em 1655, Wallis publicou um tratado sobre seções cônicas onde as mesmas foram definidas analiticamente. A difícil e obscura *Géométrie* de Descartes, tornou-se mais compreensível para todos os matemáticos neste trabalho de Wallis. Foi aí que pela primeira vez essas curvas foram consideradas e que as curvas de segundo grau foram definidas. O seu trabalho mais importante foi o *Arithmetica Infinitorum* (1656). Neste tratado os métodos de analise de Descartes e Cavaliere foram sistematizados e estendidos.

Wallis determinou o valor da $\int_0^1 (1-x^2)^n dx$ para valores inteiros de *n*, tendo como base o método dos indivisíveis de Cavalieri. Ele idealizou um método de interpolação para tentar computar $\int_0^1 (1-x^2)^{1/2} dx$. Usando o conceito de continuidade de Kepler ele descobriu métodos para determinar o valor das integrais que mais tarde seriam usadas por Newton no seu trabalho do Teorema Binomial. Newton disse:

> *Sobre o início dos meus estudos matemáticos, assim que os trabalhos do nosso celebrado conterrâneo, Dr. Wallis, caíram nas minhas mãos, por considerar as Séries, pela intercalação dos quais, ele mostrou a área do círculo e da hipérbole....*

Em 1659, Wallis publicou um tratado, contendo a solução de problemas sobre cicloides propostos por Pascal. Neste trabalho ele explicou incidentalmente como os princípios estabelecidos no seu *Arithmetica Infinitorum* poderiam ser usados a fim de retificar curvas algébricas e deu uma solução do problema da retificação da parábola semi-cúbica $x^3 = ay^2$, a qual tinha sido descoberta pelo seu pupilo William Neil em 1657. Também foi neste trabalho que Wallis estabeleceu a seguinte expressão:

$$\pi/2 = (2.2.4.4.6.6.8.8.10...)/(1.3.3.5.5.7.7.9.9...)$$

A teoria da colisão dos corpos foi proposta pela *Royal Society* em 1668 para consideração de matemáticos. Wallis, Wren e Huygens mandaram soluções corretas e similares, todas dependendo do que agora é chamado "conservação da quantidade de movimento"; mas, enquanto Wren e Huygens consideravam apenas corpos perfeitamente elásticos, Wallis considerou também corpos não perfeitamente elásticos. Este trabalho foi seguido por outros, um em 1669 sobre estática (centros de gravidade), e outro em 1670 sobre dinâmica. Esses trabalhos resumiam essencialmente o que era conhecido até então sobre o assunto.

Em 1685 publicou um tratado sobre álgebra, o qual possuía uma descrição do desenvolvimento histórico da área, com muitas informações valiosas. Este foi o início de um estudo sério sobre a história dos matemáticos na Inglaterra. Estudantes de trigonometria reconhecerão a palavra "mantissa" a qual também foi introduzida por Wallis neste trabalho.

Além dos seus trabalhos em matemática ele escreveu também sobre teologia, lógica, e filosofia. Wallis foi o primeiro a inventar um sistema para ensinar a surdos-mudos. Tal sistema foi obtido por Wallis relacionando música e matemática.

Referências:

[web01]- http://www.maths.tcd.ie/pub/HistMath/People/Wallis/RouseBall/RB_Wallis.html

[web02]- http://lhs.lindy.k12.ny.us/hs/Math%20Web%20Site/Famous%20Mathematicians/John%20Wallis/Wallis.htm?llis.htm

[web03] - http://www-groups.dcs.st-andrews.ac.uk/~history/Mathematicians/Wallis.html