UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação
Departamento de Engenharia de Computação e
Automação Industrial

Criptossistemas baseados em curvas elípticas: estudo de casos e implementação em processador de sinais digitais

Autor:

Arnaldo Jorge de Almeida Júnior

Orientador:

Prof. Dr. Marco Aurélio Amaral Henriques

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos necessários para a obtenção do título de Mestre em Engenharia Elétrica.

Banca Examinadora:

Prof. Dr. José Raimundo de Oliveira (FEEC / UNICAMP)

Prof. Dr. Reginaldo Palazzo Jr. (FEEC / UNICAMP)

Prof. Dr. Ricardo Dahab (IC / UNICAMP)

Campinas, São Paulo, Brasil

Maio, 2002

Resumo

Recentemente a criptografia baseada em curvas elípticas (ECC) tem recebido considerável atenção, evidenciada pela sua inclusão em padrões ANSI, IEEE, ISO e NIST, sua especificação para uso nas camadas de segurança de protocolos como ATM e WAP, bem como sua implementação em serviços como SET e IPSec. Apesar de toda a padronização, existem alguns aspectos relativos à adequação entre a forma e plataforma de implementação que não estão esclarecidos. Além de discutir tais aspectos, este trabalho explora várias abordagens para a implementação de algoritmos criptográficos baseados em curvas elípticas em um processador de sinais digitais (DSP). É mostrado que, em ambientes restritos, é necessário fazer uma escolha bastante cuidadosa dos vários parâmetros e condições de operação das curvas elípticas, para que sua característica de prover segurança com economia de recursos seja bem aproveitada. Por exemplo, apesar de corpos finitos binários frequentemente serem adotados pelos projetistas quando a plataforma apresenta recursos limitados, dados obtidos neste trabalho mostram que para plataformas limitadas e baseadas em DSPs esta não é necessariamente a melhor opção.

Abstract

Recently, cryptography based on elliptic curves (ECC) has attracted some attention, due to its inclusion in some ANSI, IEEE, ISO and NIST standards, its application in some protocol security layers as ATM and WAP, and its implementations in services as SET and IPsec. Despite all standardization, there are some issues related to the matching between implementation and processing environment that are not clear enough. In addition to the dicussion of such issues, this work explores several techniques for implementing cryptographic algorithms based on ellipitc curves on digital signal processors (DSP). It was found that in such environments it is necessary to carefully select elliptic curve parameters and operating conditions, in order to explore its characteristic of providing security with less computational resources. For example, binary fields frequently are chosen by designers when the processing environment has constrained resources. However, some results of this work show that, for constrained environments based in DSPs, this choice is not necessarily the best one.

Agradecimentos

Agradeço o auxílio de algumas pessoas que foram fundamentais para que este trabalho se concretizasse:

- à minha mãe, pelo seu eterno apoio em tudo que decido fazer;
- à minha irmã Iara Bianca, pela sua disposição em cuidar de tantas coisas para mim;
- ao prof. Dr. Marco Aurélio Amaral Henriques, meu orientador;
- aos professores da banca:
  - prof. Dr. José Raimundo de Oliveira;
  - prof. Dr. Reginaldo Palazzo Jr.;
  - prof. Dr. Ricardo Dahab.
- aos professores Dr. Edmundo da Silva Braga e Dr. Marco Antônio Robert Alves, DEMIC/FEEC/UNICAMP, pelo incentivo para iniciar o mestrado;
- ao Dr. Julio César López Hernández, Universidad del Valle, Colombia, pelas frutíferas discussões sobre curvas elípticas;
- ao Robson Geremias Macedo, aluno de graduação em Engenharia Elétrica, FEEC/UNICAMP, pelo auxílio na preparação, testes e obtenção de dados dos programas utilizados no trabalho.

Sumário

Lista de Figuras

Notações

$a \ mod \ n$	resto da divisão inteira de a por n (lê-se a módulo n. Quando n é um número primo esta operação é também chamada de *redução*. O resultado desta operação é comumente chamado de *resíduo*);
$a \equiv b \ (mod \ n)$	a é *congruente* a b módulo n, significa que n divide $a - b$;
$a \mid b$	a é um divisor de b;
G	grupo;
E	curva elíptica;
F	corpo;
F_p	corpo finito contendo p elementos, onde p é um primo;
F_q	corpo finito contendo q elementos, podendo ser $q = p$ ou $q = p^m$, p sendo um número primo e m um inteiro positivo;
F_{2^m}	corpo finito contendo 2^m elementos, onde m é um inteiro positivo;
GF	Galois Field (lê-se Corpo de Galois);
$GF(p)$	o mesmo que F_p;
O	Ponto no Infinito. Este ponto é o elemento identidade do grupo aditivo formado pelos pontos de curvas elípticas sobre corpos finitos;
$E(F_q)$	conjunto de todos os pontos de uma curva elíptica E definida sobre F_q incluindo o Ponto no Infinito O;
$\#E(F_q)$	se E é definido sobre F_q, então $\#E(F_q)$ denota o número de pontos na curva elíptica (incluindo o Ponto no Infinito O) e é chamado de *ordem* da curva E;
$F \setminus \{0\}$	elementos de um corpo, exceto o zero. Notação normalmente usada para denotar *grupo multiplicativo de* F;

Z_p	inteiros módulo p;
Z_p^*	inteiros módulo p excluindo o zero, se p for primo;
$Z_p[x]$	conjunto de todos os polinômios cujos coeficientes estão em Z_p;
$\lceil N \rceil$	menor inteiro que seja maior ou igual a N.

Abreviaturas

ANSI	American National Standards Institute
ATM	Asynchronous Transfer Mode
DSA	Digital Signature Algorithm
DSP	Digital Signal Processor
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
IEEE	Institute of Electrical and Electronics Engineers
IPSec	Internet Protocol Security
ISO	International Organization for Standardization
MD5	Message Digest, Versão 5
NIST	National Institute of Standards and Technology
PDA	Personal Digital Assistant
PGP	Pretty Good Privacy
OEF	Optimal Extension Fields
SET	Secure Electronic Transaction
SHA-1	Secure Hash Algorithm - Revisão 1
SSH	Secure Shell
SSL	Secure Sockets Layer
RSA	Rivest-Shamir-Adleman
WAP	Wireless Application Protocol
WTSL	Wireless Transport Security Layer

Capítulo 1

Introdução

Tanto no dia a dia das pessoas quanto no mundo dos negócios, presencia-se de forma cada vez mais intensa a utilização de sistemas de computação interconectados a taxas de comunicação cada vez mais altas. Existe uma busca constante pela *mobilidade*, bem como uma contínua migração de várias tecnologias da forma analógica para a digital, trazendo inúmeros benefícios. No entanto, surgem novos inconvenientes em conseqüência destas novas facilidades como, por exemplo, riscos de fraudes até pouco tempo inexistentes. Neste sentido, os mecanismos que provêem segurança aos sistemas de computação e seus dados são uma das questões principais para o sucesso das novas formas eletrônicas de interação entre as pessoas.

A plataforma PC teve posição de destaque durante a década de 90, sendo uma das principais propulsoras da chamada *economia digital*. Recentemente, entretanto, muitas aplicações tendem a um outro tipo de plataforma: a de processamento embutido, na qual microprocessadores atuam de forma significativa (e às vezes transparente) nas funções de telefonia sem fio, computação móvel, cartões inteligentes, entre outras. Tais aplicações muitas vezes tratam dados sensíveis e, portanto, se sustentam em mecanismos de segurança, como a criptografia.

Por motivos de requisitos de mercado (baixo custo), bem como para proporcionar portabilidade (baixo consumo, baixa dissipação de potência e uso de componentes integrados), estas plataformas de processamento embutido normalmente apresentam limi-

tações de recursos computacionais, o que dificulta a implementação dos algoritmos de criptografia tradicionais para se prover os serviços de segurança. Por este motivo, procuram utilizar as curvas elípticas como recurso matemático para prover criptografia, pelo fato destas necessitarem de menos recursos computacionais para sua implementação.

As tecnologias de comunicação emergentes, as tendências da tecnologia da informação, bem como as novas necessidades criadas pelo marketing tecnológico, apresentam uma demanda comum: processamento de dados em tempo real. Os Processadores de Sinais Digitais (DSPs) têm-se mostrado uma boa solução para tal processamento. Os DSPs foram desenvolvidos e otimizados para executarem certas operações em alta velocidade e, por esse motivo, formam o núcleo de aparelhos de comunicação que já são parte do cotidiano, como os telefones celulares, faxes, pagers, modems, bem como sistemas de telefonia sobre a Internet (VoIP) , entre outros. Assim como outros sistemas de comunicação, estes também demandam serviços de segurança e a criptografia baseada em curvas elípticas surge como uma boa solução para atender esta demanda.

1.1 Criptografia

Criptografia, do grego kryptós (escondido, oculto) + grápho (grafia, escrita), é a arte ou ciência de se escrever em cifra ou em código. A criptografia é uma ciência que antecede seu uso em aplicações computacionais, tendo sido usada por exemplo, para cifrar mensagens por Júlio César já na época do Império Romano.

Na criptografia moderna, o processo de conversão da mensagem original para mensagem cifrada é controlado por uma chave e. Esta chave é uma sequência de bits que determina o efeito da função de cifragem. O processo reverso de conversão da mensagem cifrada para a mensagem original é chamado de decifragem e é controlado por uma chave d.

Existem basicamente duas classes de criptografia, conhecidas como criptografia de chave simétrica (ou chave privada) e de chave assimétrica (ou chave pública). A relação entre e e d diferencia estas duas classes.

Na criptografia de chave simétrica $e = d$, ou seja, a mesma chave é usada para a

cifragem e decifragem. Sua principal vantagem é a velocidade para cifragem/decifragem e sua principal desvantagem é a dificuldade na distribuição e manutenção de chaves secretas. Dentre os muitos algoritmos criptográficos de chave simétrica estão o DES, RC6, FEAL, IDEA e o Rijndael.

Na criptografia de chave assimétrica $e \neq d$, ou seja, utiliza-se uma chave para cifrar e outra para decifrar, sua principal vantagem é a não necessidade de se manter um segredo em comum entre as partes e sua principal desvantagem é o baixo desempenho, visto que esta técnica lida com sofisticadas funções matemáticas. Dentre os vários algoritmos criptográficos de chave assimétrica estão o RSA, ElGamal e os baseados em curvas elípticas.

Os principais serviços de segurança oferecidos pela criptografia são apresentados a seguir:

Privacidade (sigilo): impede que pessoas não autorizadas tenham acesso ao conteúdo da mensagem, garantindo que apenas a origem e o destino tenham conhecimento do mesmo.

Integridade: garante que o conteúdo da mensagem não é alterado.

Autenticidade da origem: garante a identidade de quem envia a mensagem.

Não-repúdio: previne que alguém possa negar o envio ou recebimento de uma mensagem efetivamente enviada ou recebida.

1.2 Contribuições da Dissertação

As principais contribuições desta dissertação são:

- Apresentação de uma introdução às curvas elípticas e alguns de seus criptossistemas (ECC).
- Análise comparativa dos principais trabalhos acadêmicos que implementaram criptossistemas baseados em curvas elípticas sobre diversas plataformas de processamento.

- Proposta de implementação modular de criptossistemas baseados em curvas elípticas.
- Implementação de uma biblioteca escrita em C (com abordagem de modularização) para criptossistemas baseados em curvas elípticas.
- Implementação de criptossistema baseado em curvas elípticas para DSP típico do segmento de dispositivos portáteis.
- Implementação de várias opções de criptossistemas baseados em curvas elípticas, tais como corpos finitos binários F_{2^m}, representados por coodernadas afins, e corpos finitos F_p, representados por coordenadas afins e por coordenadas projetivas, a fim de avaliar qual corpo finito melhor se adequa à arquitetura de um processador de sinais digitais.
- Proposta de uma métrica para avaliação de criptossistemas baseados em curvas elípticas, a fim de permitir comparações entre diversas implementações feitas sob condições diversas.

Dois artigos foram derivados desta dissertação e apresentados em congressos de segurança de dados. Maiores detalhes a respeito destes artigos podem ser encontrados no Apêndice E.

1.3 Estrutura da Dissertação

O capítulo 2 apresenta uma breve introdução aos criptossitemas baseados em curvas elípticas, com ênfase na apresentação das várias alternativas de implementação. Vários trabalhos acadêmicos que implementaram criptossitemas baseados em curvas elípticas bem como suas alternativas de implementação também são apresentados neste capítulo.

O capítulo 3 descreve uma proposta de modularização de criptossitemas baseados em curvas elípticas em 5 camadas e sua implementação em um processador de sinais digitais.

O capítulo 4 apresenta uma proposta de classificação bem como uma avaliação de resultados de vários trabalhos que implementaram ECC.

O capítulo 5 contém conclusões e sugestões de trabalhos futuros.

Os Apêndices trazem detalhes adicionais, tais como a *Introdução às Curvas Elípticas* (Apêndice A), *Coordenadas Projetivas* (Apêndice B), *Esquemas Baseados em Curvas Elípticas* (Apêndice C), *Exemplos de Curvas Elípticas* (Apêndice D) e *Publicações Derivadas deste Trabalho* (Apêndice E).

Capítulo 2

Uma Visão Geral dos Criptossistemas Baseados em Curvas Elípticas

2.1 Curvas Elípticas

As curvas elípticas têm sido estudadas intensamente nos últimos 150 anos e desses estudos emergiu uma rica e profunda teoria [35]. A denominação *curvas elípticas* está associada ao fato de que no passado estas foram utilizadas para medir o perímetro de elipses e os comprimentos das órbitas dos planetas [7]. As curvas elípticas têm atraído muito interesse pelo fato de suas muitas aplicações em criptografia: fatoração de inteiros, teste de primalidade (verificação se um número qualquer é primo) e para construção de criptossistemas. Recentemente as curvas elípticas foram utilizadas no encaminhamento da demosntração do *último teorema de Fermat* [53].

Criptossistemas baseados em curvas elípticas proporcionam segurança equivalente a outros esquemas de criptografia, como o RSA por exemplo, mas com a vantagem de precisar de chaves menores, o que implica em menores requisitos de velocidade do processador, de memória e de largura de banda, facilitando sua implementação tanto do ponto de vista de software como de hardware. Uma introdução mais aprofundada sobre

curvas elípticas pode ser encontrada em [35], [7] e [22].

As principais propriedades das curvas elípticas podem ser encontradas no apêndice A onde são apresentados os conceitos necessários para o uso de corpos finitos na formação de curvas elípticas.

2.1.1 Criptossistemas Baseados em Curvas Elípticas

Vários serviços de segurança necessários em sistemas de informação, como garantia de integridade, autenticação e não-repúdio, necessitam de mecanismos de criptografia de chave pública para serem implementados. Uma das formas de se construir estes mecanismos é com a aplicação das curvas elípticas.

Criptografia de Chave Pública: Um Breve Histórico

Até meados dos anos 70 os principais serviços que a criptografia oferecia eram obtidos com a aplicação de algoritmos de chave simétrica, também chamados de algoritmos de chave secreta.

Em 1976, durante a *National Computer Conference* [13], Whitfield Diffie e Martin Hellman apresentaram um novo conceito de criptografia: a criptografia de chave pública. Eles propuseram um esquema baseado em duas chaves, uma pública e outra privada, onde uma mensagem, cifrada com uma chave, só poderia ser decifrada com a outra. No entanto, não chegaram a apresentar uma implementação prática deste novo conceito. Ainda em 1976, Diffie e Hellman apresentaram um esquema de troca de chaves secretas, baseado na dificuldade de se calcular logaritmos discretos [14].

Este novo conceito de criptografia se sustentaria na teoria dos números, um ramo da matemática pura, mais especificamente nas funções unidirecionais. Estas funções são relativamente fáceis de serem computadas, mas suas inversas são extremamente difíceis ou inviáveis. Uma boa introdução às funções unidirecionais pode ser vista em [35] e vários exemplos de funções unidirecionais são apresentados no capítulo 3 de [36].

Em 1978, Ron Rivest, Adi Shamir e Len Adleman apresentaram o primeiro algoritmo que efetivamente viabilizou as idéias de Diffie e Hellman: o RSA [43]. Este se baseou

na dificudade de se fatorar números inteiros e se tornou um dos principais esquemas de criptografia desde então.

Em meados da década de 80, Victor Miller [37] e Neal Koblitz [25] propuseram independependente o uso das curvas elípticas para aplicações em criptografia de chave pública, que apresenta como principal vantagem possuir segurança equivalente aos esquemas existentes, mas com chaves de menor tamanho.

Recentemente, o grupo de estudos de padronização IEEE P1363, concluiu a primeira versão do STANDARD SPECIFICATION FOR PUBLIC KEY CRYPTOGRAPHY, sendo que os métodos matemáticos padronizados foram: fatoração de números inteiros, logaritmos discretos e as curvas elípticas [22].

Logaritmos Discretos

Vários esquemas de criptografia de chave pública são baseados nos logaritmos discretos, como por exemplo o DSA (Digital Signature Algorithm) [2], os vários esquemas de ElGamal [16], o esquema de assinatura digital de Schnorr [46], o esquema de assinatura digital de Nyberg-Rueppel [39], entre outros.

Para todos os esquemas baseados nos logaritmos discretos, existe um análogo utilizando curvas elípticas. Exemplos de esquemas criptográficos baseados nos logaritmos discretos podem ser encontrados no apêndice C.

O *problema dos logaritmos discretos* consiste na dificuldade de se encontrar algum método computacionalmente viável de se calcular logaritmos num dado grupo G (Apêndice A). Antes de se apresentar a definição de logaritmos discretos, se faz necessário apresentar o conceito de *elemento gerador* de um grupo.

Elemento gerador: um elemento α, é dito gerador ou primitivo de um grupo G se todos os elementos β não nulos deste grupo puderem ser escritos sob a forma $\beta = \alpha^k$, onde k é um número inteiro.

Logaritmo discreto: Considere G um grupo multiplicativo finito de ordem n e α o gerador de G. O *logaritmo discreto* do elemento β de G na base α, denotado por $log_{\alpha}\beta \ mod \ n$, é o único inteiro x, $0 \leq x < n$, tal que $\beta = \alpha^x \ mod \ n$.

Um algoritmo óbvio para se calcular o logaritmo discreto consiste em computar potências sucessivas $\alpha^k mod \ n$ até que β seja encontrado (método da força bruta). Este método é computacionalmente inviável para n muito grande. Existem alguns métodos mais eficientes para se calcular logaritmos discretos como por exemplo Pollard's rho, Pohlig-Hellman e Index-calculus (uma apresentação de todos estes métodos pode ser encontrada em [36]).

2.1.2 Segurança de Criptografia Baseada em Curvas Elípticas

O problema dos logaritmos discretos sobre curvas elípticas, é considerado mais intratável que o dos logaritmos discretos sobre corpos finitos Z_p^* [27] e [7]. Entretanto, nem todas as curvas elípticas podem ser consideradas seguras, assim como nem todas as curvas elípticas são eficientes (do ponto de vista da aritmética elíptica).

Esta seção discute a segurança dos criptossistemas baseados em curvas elípticas e apresenta as classes de curvas elípticas que podem ser aplicadas para os propósitos de criptografia, bem como aquelas que devem ser evitadas.

Cálculo dos Logaritmos Discretos sobre Curvas Elípticas

Desde a invenção de cripto-sistemas de chave pública em 1976 por Diffie e Hellman, numerosos métodos para se implementar criptossistemas de chave pública foram propostos, todos baseados na dificuldade de se resolver algum problema matemático. Ao longo dos anos, muitos destes métodos foram criptoanalisados e considerados fracos e muitos outros demonstraram ser impraticáveis. Os três métodos de maior aceitação e padronizados pela IEEE P1363 [22] são aqueles baseados no problema dos logaritmos discretos, no problema da fatoração de inteiros e no problema dos logaritmos discretos sobre curvas elípticas.

Entre 1978 e 1984, vários algoritmos eficientes para resolver o problema dos logaritmos discretos foram propostos [31]. Os logaritmos discretos (LD) e a fatoração de inteiros (FI) são problemas tais que muitos algoritmos desenvolvidos para resolver um problema podem ser modificados para serem aplicados ao outro. Os melhores métodos conhecidos para se resolver os logaritmos discretos (LD) e a fatoração de inteiros possuem complexidade algorítmica com tempo sub-exponencial. Isto elevou o problema dos logaritmos discretos sobre curvas elípticas (DLCE) a um status especial, pois os melhores métodos conhecidos para calculá-lo possuem complexidade algorítmica com tempo exponencial [7]. É por esse motivo que a criptografia baseada em curvas elípticas requer chaves menores para manter o mesmo nível de segurança de outros esquemas de criptografia de chave pública.

Segurança vs. Tamanhos de Chaves

Várias referências apresentam comparações entre a segurança dos criptossistemas baseados em curvas elípticas e os algoritmos tradicionais de criptografia de chave pública [50],[7]. No entanto, existem algumas divergências entre as estimativas de tempo de processamento para se *descobrir* [1] uma chave, devido à dificuldade de se estimar o tempo de *utilização do processador* [2].

Na Tabela 2.1 é apresentada uma estimativa presente no anexo D da IEEE P1363 [22] em MIPS-ano, onde MIPS-ano é a quantidade aproximada de computação que uma máquina (capaz de executar um milhão de instruções por segundo), executaria em um ano (aproximadamente 3×10^{13} instruções).

[1] Tempo de processamento necessário para se calcular o problema em questão, por exemplo, no caso de ECC, o tempo necessário para calcular o *logaritmo discreto sobre curvas elípticas*.

[2] Estimativa de quantas instruções aritméticas um processador executa por segundo quando trabalha em um trecho de programa. Esta estimativa não depende somente do clock, mas também da arquitetura do processador, da quantidade e velocidade das memórias cache e RAM e do trecho particular do programa em execução.

L.D. e F.I.	C.E.	Volume de processamento (MIPS-ano)
512	128	$4,0 \times 10^5$
1024	172	$3,0 \times 10^{12}$
2048	234	$3,0 \times 10^{21}$
4096	314	$2,0 \times 10^{33}$

Tabela 2.1: Tamanho de chaves dos esquemas de criptografia e volume de processamento segundo norma P1363, anexo D [22].

Classes de Curvas Elípticas

Qualquer curva elíptica E pode ser representada como uma curva cúbica plana ou seja, em P^2, pela Eq. 2.1.

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2.1}$$

Esta equação é chamada de *equação de Weierstrass* [35]. Dependendo da característica do corpo onde a curva é definida, a Eq. (2.1) pode ser transformada e simplificada, como mostrado em [35] e [7].

Para corpos finitos primos F_p com $p > 3$, a equação de Weierstrass apresenta a forma da Eq. (2.2).

$$y^2 = x^3 + a_4x + a_6 \tag{2.2}$$

Estas curvas elípticas são consideradas seguras, desde que não sejam anômalas. Uma curva elíptica sobre F_p é dita anômala se $\#E(F_p) = p$, ou seja, se possui ordem (número de pontos) igual a p. Para esta classe de curvas existem vários ataques polinomiais [48], de forma que atualmente são consideradas inseguras para sistemas de criptografia.

Para corpos finitos de característica dois, F_{2^m}, a equação de Weierstrass pode representar curvas supersingulares, caso $a_1 = a_2 = 0$ e $a_3 = 1$, tomando a forma da Eq. (2.3)

$$y^2 + y = x^3 + a_4x + a_6 \tag{2.3}$$

As curvas supersingulares foram consideradas inicialmente como muito promissoras para criptografia em função de poderem ser calculadas rapidamente. Entretanto, no início dos anos 90, Menezes, Okamoto e Vanstone [34], mostraram que o problema dos logaritmos discretos sobre as curvas supersingulares, poderia ser eficientemente reduzido ao problema dos logaritmos discretos sobre F_p, de forma que estas curvas deixaram de ser aplicadas em criptografia.

Existem também as curvas não-supersingulares, que assumem a forma da Eq. (2.4)

$$y^2 + xy = x^3 + a_2x^2 + a_6 \tag{2.4}$$

Estas curvas são consideradas seguras e existem combinações destas curvas (coeficientes específicos e corpos finitos específicos), que proporcionam mais eficiência na aritmética elíptica, como é o caso, por exemplo, das Curvas de Koblitz [3] [26].

2.2 Alternativas de Implementação de Criptossistemas Baseados em Curvas Elípticas

São muitas as escolhas que devem ser feitas para se desenvolver um criptossistema baseado em curvas elípticas (ECC). Elas vão desde o tipo de curva elíptica, passando por sua representação, e chegando na escolha de algoritmos a serem aplicados. A seguir são apresentadas as alternativas típicas para a implementação de criptossistemas baseados em curvas elípticas.

2.2.1 Corpo Finito

Há pelo menos três escolhas possíveis :

[3] Curvas definidas em F_{2^m} nas formas $y^2 + xy = x^3 + 1$ e $y^2 + xy = x^3 + x^2 + 1$.

- *Corpo Primo (F_p)*: os p elementos deste corpo são inteiros menores que um primo p e as operações são implementadas em termos de aritmética de inteiros módulo p.

- *Corpo Binário (F_{2^m})*: contém 2^m elementos para algum inteiro m (chamado de grau do corpo). Os elementos deste corpo são uma cadeia de bits de tamanho m e a aritmética neste corpo é implementada em termos de operações sobre bits.

- *Corpo de Extensão Ótima (F_{p^m}) (Optimal Extension Field - OEF)*: o símbolo p é um número primo de Mersenne $2^n \pm c$, para n, c inteiros positivos e arbitrários e o símbolo m é um inteiro positivo maior que zero.

2.2.2 Representação da Base em F_{2^m}

Para descrever a aritmética de corpos finitos binários, primeiro é necessário especificar como a cadeia de bits será interpretada. Isto é normalmente referido como escolha da base. Uma base de um espaço vetorial pode ser definida como um conjunto linearmente independente de vetores que geram este espaço vetorial. Para corpos finitos binários (F_{2^m}), dependendo da base aplicada, existem técnicas eficientes para a execução das operações aritméticas. Há dois tipos comuns de base:

- *Base Polinomial*: Numa representação em base polinomial, cada elemento de F_{2^m} é representado por um polinomio de grau menor que m. Mais explicitamente, a sequência de bits $(a_{m-1} \cdots a_2 a_1 a_0)$ representa o polinômio:

$$a_{m-1}t^{m-1} + \cdots + a_2 t^2 + a_1 t + a_0$$

- *Base Normal*: É a base na forma:

$$\{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\},$$

onde $\beta \in F_{2^m}$, ou seja, um elemento α qualquer de F_{2^m} pode ser escrito na forma:

$$\alpha = \sum_{i=0}^{m-1} a_i \beta^{2^i},$$

onde $a_i \in \{0, 1\}$. A principal vantagem da representação normal esta no fato de que o cálculo do quadrado de um elemento passa a ser uma operação simples de deslocamento de um bit.

2.2.3 Representação das Coordenadas

Como discutido anteriormente, a equação das curvas elípticas é um caso especial da equação de Weierstrass e os pontos pertencentes a esta curva podem ser representados por vários sistemas de coordenadas, como por exemplo coordenadas afins, projetivas homogêneas, projetivas jacobianas ou outros sistemas de coordenadas projetivas [35]. Em casos onde o cálculo do inverso multiplicativo for significativamente mais complexo que o cálculo da multiplicação, pode ser mais eficiente a implementação de sistemas de coordenadas projetivas. Mais detalhes a respeito das coordenadas projetivas podem ser encontrados no apêndice B.

- *Sistema de coordenadas afim:* para F_p, $p > 3$, a equação da curva elíptica é apresentada sob a forma $E : y^2 = x^3 + ax + b$ com $a, b \in F_p$ e com a condição $4a^3 + 27b^2 \neq 0$. Assumindo que $P_1 = (x_1, y_1)$ e $P_2 = (x_2, y_2)$ sejam pontos pertencentes a $E(F_p)$, a soma $P_3 = (x_3, y_3) = P_1 + P_2$ pode ser calculada conforme as Eqs. (2.5) a (2.7).

$$x_3 = \lambda^2 - x_1 - x_2 \quad , \tag{2.5}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \; onde \tag{2.6}$$

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \; se \; P_1 \neq P_2, \\ \frac{3{x_1}^2 + a}{2y_1} \; se \; P_1 = P_2. \end{cases} \tag{2.7}$$

Para F_{2^m}, a equação da curva elíptica é apresentada sob a forma
$E : y^2 + xy = x^3 + ax^2 + b$ com $a, b \in F_{2^m}$ e com a condição $b \neq 0$. Assumindo

que $P_1 = (x_1, y_1)$ e $P_2 = (x_2, y_2)$ sejam pontos pertencentes a $E(F_{2^m})$, a soma $P_3 = (x_3, y_3) = P_1 + P_2$ pode ser calculada conforme as Eqs. (2.8) a (2.10).

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \quad , \tag{2.8}$$

$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \; onde \tag{2.9}$$

$$\lambda = \begin{cases} \frac{y_2+y_1}{x_2+x_1} \; se \; P_1 \neq P_2, \\ x_1 + \frac{y_1}{x_1} \; se \; P_1 = P_2. \end{cases} \tag{2.10}$$

- *Sistema de coordenadas projetivas homogêneas:* um ponto projetivo (X, Y, Z) na curva, satisfaz a equação de Weierstrass $Y^2Z = X^3 + aXZ^2 + bZ^3$ para F_p e a equação $Y^2Z + XYZ = X^3 + aX^2Z + bZ^3$ para F_{2^m} . Quando $Z \neq 0$, este ponto corresponde ao ponto afim $(X/Z, Y/Z)$. As equações para o cálculo de soma e duplicação de pontos projetivos podem ser encontradas em [3] para o caso de F_p e em [35] para o caso de F_{2^m}.

- *Sistema de coordenadas projetivas jacobianas:* um ponto projetivo jacobiano (X, Y, Z), satisfaz a equação de Weierstrass $Y^2 = X^3 + aXZ^4 + bZ^6$ para F_p e $Y^2 + XYZ = X^3 + aX^2Z^2 + bZ^6$ para F_{2^m} . Quando $Z \neq 0$, este ponto corresponde ao ponto afim $(X/Z^2, Y/Z^3)$. Este sistema de coordenadas é o recomendado pela norma IEEE P1363 e as equações para o cálculo de soma e duplicação de pontos projetivos jacobianos para F_{2^m} e F_p podem ser encontradas no apêndice B.

Recentemente, outros sistemas de coordenadas projetivas estão sendo propostos como o sistema de *Chudnovsky* e os sistemas combinados Jacobianas-afim, Jacobianas-Chudnovsky e Chudnovsky-afim apresentado em [8]. Em [31] e [32] é apresentado um novo sistema de coordenadas afim sobre F_{2^m}.

2.2.4 Algoritmos de Aritmética Modular

São os algoritmos que aplicam as operações modulares tais como adição, subtração, multiplicação e sua operação inversa, o inverso multiplicativo, que é normalmente a operação mais custosa.

2.2.5 Algoritmos de Aritmética Elíptica

Estes algoritmos implementam a adição e a duplicação de pontos, bem como a multiplicação escalar de pontos, ou seja, dado um ponto P e um escalar k, realizam o cálculo de kP. A eficiência desta operação pode definir o desempenho de um sistema com ECC.

2.2.6 Tipo de Curva Elíptica

- *Curvas Pseudo-aleatórias* são curvas geradas aleatoriamente. Deve ser verificado se estas definem uma ordem $\#E(F_q)$ (número de pontos) apropriada para aplicações criptográficas. Também deve ser verificado se a curva gerada é segura contra os principais ataques conhecidos, como descrito no capítulo 6 de [7].

- *Curvas especiais* são aquelas cujos coeficientes e o corpo finito foram escolhidos para otimizar a eficiência das operações da curva elíptica, como por exemplo as curvas de Koblitz sobre F_{2^m}.

2.2.7 Criptossistema

Geralmente, qualquer esquema de criptografia baseado nos logaritmos discretos, terá um análogo sobre as curvas elípticas, como por exemplo os esquemas de Diffie e Hellman [14], ElGamal [16], dentre outros. Os mais comuns são aqueles padronizados pela IEEE P1363 [22]:

- ECDH : esquema de troca de chaves Diffie-Hellman baseado em curvas elípticas. Este esquema é aplicado em conjunto com algum algoritmo de criptografia simétrica

como por exemplo o triplo-DES ou Rijndael (AES), para se obter um criptossistema completo.

- ECDSA : esquema de assinatura digital baseado em curvas elípticas (análogo ao DSA). É composto por duas operações: a geração e a verificação de assinatura. Este esquema é aplicado em conjunto com os algoritmos de hash SHA-1 ou MD5.

Conforme visto, são muitas as alternativas e as técnicas de implementação que fazem parte do projeto de um criptossistema baseado em ECC. O critério de escolha destes parâmetros e algoritmos normalmente estão relacionados ao desempenho e nível de segurança desejados, disponibilidade de recursos computacionais e arquitetura de processamento.

2.3 Trabalhos Anteriores de Implementação de Criptossistemas Baseados em Curvas Elípticas

Várias publicações apresentam algoritmos e métodos para computação eficiente de criptossistemas baseados em curvas elípticas, mas normalmente tratam o assunto de forma parcial e isolada e somente algumas publicações chegam a apresentar implementações completas de criptossistemas. Os sistemas apresentados neste capítulo, são implementações completas sobre alguma plataforma de processamento e que, consequentemente, tiveram um desempenho determinado pelos parâmetros e pelas técnicas de implementação escolhidas pelos seus autores. Estes trabalhos apresentam uma boa variedade de alternativas de implementação e de plataformas (microcontroladores, processadores digitais de sinais, processadores RISC e arquitetura Intel x86). Existem ainda iniciativas que implementam ECC em hardware específicos como, por exemplo, as FPGAs, mas tais iniciativas não foram abordadas neste trabalho.

2.3.1 De Win, Mister, Preneel e Wiener - 1998

Este artigo [11] é um dos mais referenciados. Foi um dos primeiros a tratar de forma completa um ECC, da aritmética em corpo finito e corpo elíptico ao criptossistema, para ambos os corpos F_p e F_{2^m}. A plataforma de processamento é um PC Pentium Pro200. A implementação em F_p foi feita em C/C++ e Assembly, utilizou as coordenadas projetivas e teve o melhor desempenho. A implementação em F_{2^m} foi feita em C++, utilizou um polinômio redutor (trinômio) melhorado e obteve resultados mais eficientes na operação de redução (módulo $p(x)$). Ela também aplicou algoritmos para otimizar o inverso multiplicativo e a multiplicação escalar de pontos. Na Tabela 2.2 é apresentado o perfil da implementação deste trabalho.

2.3.2 Bogdan Antonescu - 1999

A dissertação de mestrado de Bogdan Antonescu [3], implementa aritmética sobre F_p e F_{2^m} bem como aritmética de corpos elípticos em uma plataforma restrita (microcontrolador MC68302 - Motorola). O trabalho também apresenta uma implementação em um Pentium 200 MHZ e compara seus resultados com o trabalho de DeWin et al. [11]. Como não houve melhorias significativas nesta plataforma, optou-se por analisar aqui somente a implementação sobre a plataforma restrita MC68302. Neste trabalho, fica evidente o melhor desempenho da aritmética F_{2^m} em relação a F_p sobre a plataforma restrita. Certamente F_p leva desvantagem nesta plataforma visto que a instrução de multiplicação no processador da Motorola utilizado leva aproximadamente 70 ciclos de máquina para ser executada. Neste trabalho, Bogdan Antonescu não chega a implementar um criptossistema completo mas somente a aritmética acima citada. Na Tabela 2.3 é apresentado o perfil da implementação.

Software		
Corpo finito	F_p	F_{2^m}
Tamanho do Corpo (bits)	191	191
Representação de Coordenadas	Projetivas	Afim
Alg. Multiplicação	Método clássico [24]	Método de Schroeppel [47]
Alg. Inverso Multiplicativo	Não se aplica	Almost Inverse Algorithm [47]
Alg. multiplicação escalar (grupo elíptico)	Double and Add [22]	Double and Add e Sliding Window ambos com Signed digit [7]
Tipos de curvas	ANSI X9.62	ANSI X9.62
Linguagem	ASM/C/C++	C++
Hardware		
Processador	PentiumPro 200	
Arquitetura	CISC - 32 bits	
Clock	200 MHz	
Criptossistemas		
DSA, RSA e ECDSA		

Tabela 2.2: Detalhes da implementação de De Win, Mister, Preneel e Wiener - 1998 [11]

Software		
Corpo finito	F_p	F_{2^m}
Tamanho do Corpo (bits)	192	191 e 167
Representação de Coordenadas	Projetivas	Afim
Alg. Multiplicação	Biblioteca aritmética GNUMP [17]	Serial Multiplier : linear feedback shift register [3]
Alg. Inverso Multiplicativo	Não se aplica	Almost Inverse Algorithm [47]
Alg. multiplicação escalar (grupo elíptico)	Double and Add [22]	Double and Add e Sliding Window ambos com Signed digit [7]
Tipos de curvas	ANSI X9.62	ANSI X9.62
Linguagem	C	C e ASM
Hardware		
Processador	Motorola - MC68302	
Arquitetura	CISC - 32 bits	
Clock	25 MHz	
criptossistema		
Não implementou mas sugeriu ECDH e ECDSA		

Tabela 2.3: Detalhes da implementação de Bogdan Antonescu - 1999 - [3]

2.3.3 Hasegawa, Nakajima e Matsui - 1998

O artigo de Hasegawa, Nakajima e Matsui [21], um grupo da Mitsubishi, trata de uma implementação completa de um ECC sobre F_p, em uma plataforma restrita (microcontrolador M16C, 10 MHz, CISC 16 bits) da própria Mitsubishi. É dado um enfoque especial ao compromisso *(tamanho de código + dados) vs. velocidade* , pois uma das premissas era desenvolver um sistema com ECC sobre esta plataforma com no máximo 4KBytes de código/dados. Em relação ao apresentado na norma P1363 [22], foram feitas melhorias no método de somar e duplicar pontos elípticos, representados por coordenadas projetivas, diminuindo-se o número de variáveis temporárias necessárias. Para multiplicação escalar de pontos, foram implementados os algoritmos para ponto aleatório e ponto fixo, sendo que com o segundo obteve-se os melhores resultados. É importante salientar que o criptossistema foi escrito totalmente em Assembly. Na Tabela 2.4 é apresentado o perfil desta implementação.

2.3.4 Itoh, Takenaka, Torh, Temma e Kurihara - 1999

O artigo de Itoh, Takenaka, Torh, Temma e Kurihara [23], um grupo da Fujitsu, trata de uma implementação completa de um ECC sobre F_p, em um processador digital de sinais topo de linha da Texas Instruments: TMS320C6201, 200MHz, 16 bits, 1600 MIPS e Pipeline (8 unidades funcionais em paralelo + 2 unidades de multiplicação). As contribuições deste trabalho foram melhorias ao método de multiplicação modular de Montgomery [38], onde foi aproveitada a arquitetura de pipeline do DSP. Para aritmética em corpo elíptico, a contribuição foi diminuir o número de instruções de adição e multiplicação, representados por coordenadas projetivas, quando comparado ao método apresentado na norma P1363 [22]. Deve-se salientar que este trabalho faz uma abordagem especial na tentativa em se diminuir o número de instruções de adição, considerando que em um DSP estas instruções não têm tempos de execução desprezíveis, quando comparados às instruções de multiplicação. As rotinas básicas foram escritas

Software	
Corpo finito	F_p
Tamanho do Corpo (bits)	160
Representação de Coordenadas	Projetivas
Alg. Multiplicação	Não mencionado no artigo
Alg. Inverso Multiplicativo	Não se aplica
Alg. multiplicação escalar (grupo elíptico)	Tabela pré calculada para um ponto P fixo
Tipos de curvas	Curva aleatória
Linguagem	ASM
Hardware	
Processador	Mitsubishi- MC16C
Arquitetura	CISC - 16 bits
Clock	10 MHz
criptossistema	
ECDSA	

Tabela 2.4: Detalhes da implementação de Hasegawa, Nakajima e Matsui - 1998 - [21]

SOFTWARE	
Corpo finito	F_p
Tamanho do Corpo (bits)	160, 192 e 239
Representação de Coordenadas	Projetivas
Alg. Multiplicação	Montgomery melhorado e adaptado ao pipeline [38]
Alg. Inverso Multiplicativo	Não se aplica
Alg. multiplicação escalar (grupo elíptico)	Double and Add otimizado ao pipeline
Tipos de curvas	Curva aleatória
Linguagem	C e ASM
HARDWARE	
Processador	Texas instruments - TMS320C6201
Arquitetura	DSP - 32 bits
Clock	200 MHz
CRIPTOSSISTEMA	
DSA (c/RSA) e ECDSA	

Tabela 2.5: Detalhes da implementação de Itoh, Takenaka, Torh, Temma e Kurihara - 1998 - [23]

em Assembly e as demais em C. Na Tabela 2.5 é apresentado o perfil da implementação do trabalho.

2.3.5 Julio López - 2000

Julio Lópes apresenta em sua tese de doutorado [31] métodos eficientes para a aritmética sobre corpo finito F_{2^m} (multiplicação rápida [33]) , bem como para o grupo elíptico (métodos eficientes para duplicações de pontos elípticos e um novo sistema de coor-

denadas projetivas [32]). Os melhores resultados foram obtidos com o uso de curvas de Koblitz. Em sua tese também são apresentadas implementações de ECC sobre a plataforma PentiumII 400 MHz, RIM pager (Intel 386, 10MHz, CISC 32 bits) e Palm Pilot (Mototola 68000, 16 MHz, CISC 16 bits). O código foi escrito em C e na Tabela 2.6 é apresentado o perfil da implementação.

2.3.6 Woodbury, Bailey e Paar - 2000

O artigo de Woodbury, Bailey e Paar [55] apresentado na *CARDIS 2000 - Smart Card Research and Advanced Applications Conference* - é uma implementação de ECC sobre o microcontrolador 8051, comumente usado como processador nos smart cards mais populares como o Siemens 44C200 e Philips 82C852. Este trabalho apresenta uma implementação sobre Corpos de Extensão Ótima (*Optimal Extension Fields*). O corpo OEF utilizado neste trabalho foi o $F_{(2^8-17)^{17}}$ que é particulamente adequado para processadores de 8 bits. Segundo os autores este corpo OEF tem segurança equivalente a um corpo binário $F_{2^{134}}$. Na Tabela 2.7 é apresentado o perfil da implementação.

2.3.7 Aydos, Yanik e Koç - 2000

Trabalho que trata de uma implementação completa de um sistema com ECC sobre F_p em um processador ARM7TDMI (80MHz, RISC 32 bits) [5]. Nele foi utilizado o algoritmo de Montgomery [38] para multiplicação eficiente e foram usadas as coordenadas projetivas. O criptossistema implementado (ECDSA) é aplicado a um protocolo de autenticação - *Wireless Authentication Protocol* [4] - desenvolvido para telefonia móvel, handhelds e smartcards. Na Tabela 2.8 é apresentado o perfil da implementação feita.

Software			
Corpo finito	F_{2^m}		
Tamanho do Corpo (bits)	163, 233 e 283		
Representação de Coordenadas	Sistema próprio de coordenadas projetivas [32]		
Alg. Multiplicação	Método próprio [33], sendo uma extensão ao método de Lim/Lee's [30]		
Alg. Inverso Multiplicativo	Algoritmo estendido de Euclides [24]		
Alg. multiplicação escalar (grupo elíptico)	Conjunto de métodos próprios [32] e métodos de Solinas propostos em [49]		
Tipos de curvas	Curvas aleatórias e curvas de Koblitz (recomendadas pelo NIST [40])		
Linguagem	C		
Hardware			
Processador	RIM Pager (Intel - 386)	Palm Pilot (Motorola 68000)	Pentium II
Arquitetura	CISC 32 bits	CISC 16 bits	CISC 32 bits
Clock	10 MHz	16 MHz	400 MHz
criptossistema			
PGP com ECAES [1] e ECDSA			
Utilizou-se porções do OpenSSL [41] e do OpenPGP [42] para esta implementação			

Tabela 2.6: Detalhes da implementação de Julio César López Hernández - 2000 - [31]

Software	
Corpo finito	F_{p^m}, onde p é um pseudo primo de Mersenne ($2^n \pm c$)
Tamanho do Corpo (bits)	$F_{(2^8-17)^{17}} \simeq F_{2^{134}}$
Representação de Coordenadas	Afim
Alg. Multiplicação	específico a OEF [55]
Alg. Inverso Multiplicativo	específico a OEF [55]
Alg. multiplicação escalar (grupo elíptico)	método de precomputação e cadeia de adição de vetores (método *de Rooij* [10])
Tipos de curvas	Não mencionado no artigo
Linguagem	C e ASM
Hardware	
Processador	microcontrolador 8051
Arquitetura	CISC - 8 bits
Clock	12 MHz
criptossistema	
Não implementou, mas sugeriu ECDSA	

Tabela 2.7: Detalhes da implementação de Woodbury, Bailey e Paar - 2000 - [55]

SOFTWARE	
Corpo finito	F_p
Tamanho do Corpo (bits)	160, 176, 192, 208 e 256
Representação de Coordenadas	Coordenadas projetivas [9]
Alg. Multiplicação	Montgomery [38]
Alg. Inverso Multiplicativo	Não se aplica
Alg. multiplicação escalar (grupo elíptico)	Método combinado com o uso de coordenadas jacobianas
Tipos de curvas	Curvas aleatórias
Linguagem	Não consta
HARDWARE	
Processador	ARM7TDMI
Arquitetura	RISC - 32 bits
Clock	80 MHz
CRIPTOSSISTEMA	
ECDSA + *Wireless Authentication Protocol* apresentado em [4]	

Tabela 2.8: Detalhes da implementação de Aydos, Yanik e Koç - 2000 - [5]

Software	
Corpo finito	F_{2^m}
Tamanho do Corpo (bits)	163
Representação de Coordenadas	Sistema de coordenadas projetivas proposto por Julio López [32]
Alg. Multiplicação	Método proposto em [33]
Alg. Inverso Multiplicativo	Não se aplica
Alg. multiplicação escalar (grupo elíptico)	Sliding windows, Montgomery e método de Lim/Lee's [30] para ponto fixo
Tipos de curvas	Curvas aleatórias e curvas de Koblitz (recomendadas pelo NIST [40])
Linguagem	C
Hardware	
Processador	Motorala - Dragonball
Arquitetura	CISC - 32 bits
Clock	16 MHz
criptossistema	
Não implementou.	

Tabela 2.9: Detalhes da implementação de Weimerskich, Paar e Shantz - 2001 - [52]

2.3.8 Weimerskich, Paar e Shantz - 2001

O artigo de Weimerskich, Paar e Shantz [52] apresentado na *ACISP 2001 - Australasian Conference on Information Security and Privacy* - é uma implementação de ECC sobre Palm OS utilizando o PDA *Handspring Visor* com 2MB de memória, dispositivo que utiliza a CPU DragonBall da Motorola (16MHz, CISC 32 bits). O programa foi totalmente escrito em C utilizando-se o ambiente de compilação *Code Warrior*. Não foi implementado um criptossistema completo mas somente a aritmética necessária para a implementação de multiplicação em corpo elíptico (operação $Q = kP$, k um escalar e P um ponto pertencente à curva elíptica). Algumas técnicas utilizadas por Weimerskich, Paar e Shantz neste trabalho foram praticamente as mesmas apresentadas no trabalho de Júlio López [32],[33]. Duas abordagens de implementação foram apresentadas: uma para ponto fixo e outra para ponto aleatório. Os melhores resultados foram conseguidos com a abordagem para ponto fixo, devido ao uso da técnica de tabelas pré-calculadas. Estas tabelas chegam a necessitar de cerca de 11000 bytes, o que para o PDA usado não foi problema. Na Tabela 2.9 é apresentado o perfil da implementação feita.

2.4 Conclusões

O *problema dos logaritmos discretos sobre curvas elípticas* é o mecanismo algébrico que faz das curvas elípticas interessantes para aplicação em criptografia de chave assimétrica.

No entanto, para se implementar os criptossitemas baseados em curvas elípticas são muitas as escolhas que devem ser feitas, desde o tipo de curva elíptica, passando por sua representação e chegando na escolha de algoritmos a serem aplicados. Neste capítulo as principais alternativas de implementação foram apresentadas.

Buscou-se analisar também as opções adotadas pelos principais trabalhos que implementaram criptossitemas baseados em curvas elípticas, a fim de identificar algum direcionamento para uma boa escolha de opções e parâmetros. No entanto, esta análise é dificultada pelas diversidades nos cenários de implementação (plataformas de processamento, níveis de segurança, etc.) bem como pela falta de uma padronização na

apresentação de resultados.

Capítulo 3

Implementação de Curvas Elípticas em Processadores de Sinais Digitais

3.1 Arquitetura de um Processador de Sinais Digitais

O Processador de Sinais Digitais (DSP) foi projetado para implementação eficiente de algoritmos de processamento de sinais tais como filtragem e codificação, por exemplo. Os recursos comuns aos DSPs disponíveis no mercado são apresentados a seguir.

Multiplicação rápida: uma das tarefas mais comuns dos DSPs é a implementação de filtros digitais, que normalmente são expressos na forma de somatória de produtos (convolução). Esta operação é acelerada com a instrução MAC (multiply and accumulate) que leva um ciclo de máquina para ser executada.

Unidades múltiplas de execução: como as aplicações baseadas em DSPs manipulam dados amostrados em taxas elevadas e com requisitos de tempo-real, é comum haver uma ou mais unidades de execução em paralelo para acelerar este processamento. Por exemplo, unidades lógicas e aritméticas (ULA), registradores de deslocamento em paralelo à unidade de MAC e uso de pipeline são características comumente encontradas em um DSP para aumentar seu desempenho.

Acesso eficiente à memória: para que uma instrução MAC seja executada em um ciclo de máquina, é necessário que ocorra busca da instrução MAC, do dado e do coeficiente neste único ciclo. Por isso, é comum uma arquitetura baseada em múltiplos bancos de memória. Além disso, considerando que o acesso à memória dos algoritmos de DSPs tende a ter um padrão de repetição e endereçamento circular, existem modos de endereçamento, como ponteiros auto-incrementados para acelerar e economizar instruções nestas situações.

Custo computacional de laço igual à zero: tipicamente os algoritmos de DSP gastam boa parte de seu tempo de processamento em seções relativamente pequenas de software que são executadas repetidamente, ou seja, em laços (loops), de forma que a maioria dos DSPs fornecem um suporte à execução eficiente de laço, com um custo de controle do mesmo bastante reduzido.

3.2 Criptografia de Chave Pública em DSPs

O campo de implementação de algoritmos criptográficos em plataformas específicas é muito ativo. Na criptografia de chave simétrica, houve recentemente muitas implementações de algoritmos candidatos para substituir o DES (antigo algoritmo padrão americano [40]), como as descritas em [54] e [56]. Na criptografia de chave assimétrica, vários trabalhos trataram o desafio de se implementar os sofisticados mecanismos matemáticos, necessários a esse tipo de criptografia, em ambientes de processamento com recursos computacionais limitados, como o caso do trabalho de J. F. Dhem [12] que implementa RSA em smart-cards sem o uso de coprocessador criptográfico. Entretanto, a pesquisa feita sobre implementação de criptografia de chave pública em Processadores de Sinais Digitais (DSP) é limitada. Um resumo de 3 trabalhos importantes é apresentado a seguir.

O primeiro é a referência mais antiga (1986) de implementação de algoritmos criptográficos em DSP [6]. Neste trabalho o autor propõe um novo algoritmo para executar multiplicação modular que é a operação básica usada para implementar esquemas

baseados no RSA. O autor argumenta que a plataforma DSP é uma boa escolha para sua implementação, pois considera que a operação básica do RSA é a exponenciação x^e mod m, onde x, e e m são em geral inteiros em precisão múltipla, envolvendo muitas multiplicações inteiras e por isso ela é particularmente adequada para hardware com arquitetura multiplicador/acumulador (MAC).

No segundo trabalho [15] (1990), os autores descrevem uma biblioteca criptográfica desenvolvida para o DSP 56000 da Motorola que proporcionou velocidades comparáveis às implementações em hardware do mesmo algoritmo. A biblioteca inclui aritmética modular, uma implementação do DES, um algoritmo de função hash e outras funções. O artigo apresenta as tendências de uso e necessidades de ferramentas criptográficas e enfatiza a importância de se desenvolver soluções e aplicações criptográficas baseadas em processadores de propósito geral, tais como os DSPs. Ele descreve ainda, em detalhes, a implementação feita do algoritmo RSA. Em particular, foi focada a integração da operação de redução modular e multiplicação em precisão múltipla de acordo com o método de Montgomery [38]. Tal integração resultou em uma significativa melhoria de desempenho em DSP, mas não causou efeitos em uma implementação para o processador Intel 80386.

No terceiro [23] (1999), os autores propuseram dois novos métodos de implementação de algoritmos de chave pública no DSP TMS320C6201 da Texas Instruments (TI). Os autores sugerem o uso de DSPs como aceleradores criptográficos para sistemas servidores, tais como aqueles encontrados em aplicações de comércio eletrônico. Além disso, eles salientam que os DSPs apresentam duas grandes vantagens: primeiro, os DSPs são desenvolvidos com multiplicadores eficientes em hardware e podem executar em alta velocidade as multiplicações modulares, que são as operações básicas na maioria dos criptossistemas. A segunda vantagem dos DSPs é que eles podem ser usados como aceleradores criptográficos para vários algoritmos já que são programáveis. O primeiro método proposto em [23] é uma implementação modificada do algoritmo de multiplicação modular de Montgomery [38]. Como a arquitetura do TMS320C6201 permite paralelismo de instruções, o algoritmo foi modificado e adequado para tirar proveito do *pipelining*. A segunda abordagem relata métodos eficientes de implentação de crip-

tossistemas baseados em curvas elípticas. Os autores sugerem um método para reduzir o número de multiplicações e adições necessários para se calcular kP, onde k é um escalar e P um ponto da curva elíptica. É interessante observar que a plataforma TMS320C6201 utilizada em [23] é de uma família de alto desempenho e poder de processamento, mas que não é atrativa para implementações de dispositivos portáteis visto seu alto consumo de energia e custo.

3.3 ECC em DSPs para Aplicações Portáteis

Para o conhecimento do autor, até o momento não há na literatura nenhuma implementação de ECC em uma família de DSP que seja viável para aplicações em dispositivos portáteis. O único trabalho de ECC em DSP é sobre a família da Texas Instruments (TI) TMS320C6000 [23]. Entretanto, esta família apresenta sérias desvantagens de custo e consumo de energia, o que inviabiliza seu uso ao segmento de dispositivos portáteis e *wireless*.

Atualmente a TI mantém 3 famílias diferentes de DSPs baseadas em sua linha TMS320, sendo cada uma orientada a diferentes tipos de necessidades [51].

TMS320C2000 - Controle Digital : é a mais indicada para implementação de sistemas de controle digital, tais como controle PID, algoritmos para controle *sensorless*, geração de PWM, correção de fator de potência, etc. Ela é adequada para aplicações como eletrodomésticos, equipamentos médicos, impressoras e máquinas de venda automática, por exemplo.

TMS320C5000 - Comunicação Digital : tem como principal característica ser de baixo consumo de energia e de baixo custo. É a mais indicada para implementação de sistemas de comunicação digital, filtros FIR (Finit Impulse Response) e transformadas de Fourier, o que a torna recomendada para aplicações em telefonia celular, decodificadores DVD, MP3 players, decodificadores MPEG, interfaces de voz sobre IP, entre outras.

TMS320C6000 - DSP de alta performance : possui arquitetura de 32 bits, alcança de 1200 a 2400 MIPS, e suporta aritmética em ponto flutuante. É indicada para estações rádio base, modems, cable modems e PBX (Private Branch Exchange).

Considerando este cenário de poucas implementações e poucas informações sobre curvas elípticas em DSPs, optou-se por desenvolver um ECC sobre a família TMS320C5000 por ser a mais adequada para implementações em equipamentos de comunicação portáteis, onde a necessidade de criptografia está cada vez mais presente. Esta plataforma viabilizou um estudo de caso de implementação de ECC em um ambiente restrito, visto ter a mesma um desempenho modesto e consideráveis limitações de espaço de memória de programa e de dados. Especificamente, o DSP utilizado foi o TMS320VC5402, cujas principais características são: arquitetura 16 bits, 100 MHz (100 MIPS), 32KB RAM e 8KB ROM.

3.4 Arquitetura Estruturada de Implementação de ECC

Considerando a necessidade de implementar, integrar e testar facilmente as unidades funcionais de um ECC, utilizou-se a metodologia estruturada de programação, o que simplificou o trabalho devido a seus conceitos de hierarquia e modularização. Um ECC completo pode ser dividido em 5 camadas, como mostra a Fig. 3.1. Desta forma, os algoritmos que constituem a camada 1 formam a base do criptossistema e são utilizados pelos algoritmos da camada 2 que, por sua vez, são utilizados pelos algoritmos da camada 3 e assim sucessivamente. A fim de testar novos tipos de algoritmos, cada camada pode ser trocada independentemente das demais. A seguir é apresentada uma descrição de cada camada, bem como algumas referências para seus principais algoritmos. Sugere-se, como leitura complementar, o anexo A da norma IEEE P1363 [22], que apresenta os algoritmos básicos e as referências [20] e [8] para um aprofundamento teórico nos algoritmos para F_{2^m} e F_p respectivamente.

CAMADA 5
Aplicação
PGP, SSH, WAP ...
CAMADA 4
Protocolo de PKC
ECDSA, ECDH, ECElGamal, ...
CAMADA 3
Aritmética de pontos sobre Curvas Elípticas
$R \leftarrow P + Q,\ R \leftarrow P + P,\ Q = kP$
CAMADA 2
Aritmética modular sobre F_p ou F_{2^m}
4 operações básicas : A+B mod p, A-B mod p, A*B mod p e Inverso Multiplicativo
CAMADA 1
Aritmética de inteiros de precisão arbitrária
$A \leftarrow 0, A \leftarrow B, C \leftarrow A + B, \leftarrow A - B...$

Figura 3.1: Estrutura hierárquica de um criptossistema baseado em Curvas Elípticas

Camada 1 : este módulo consiste de uma coleção de rotinas de propósito geral para manipulação de inteiros de tamanho arbitrário. Funções de atribuição, adição, multiplicação, divisão, bem como rotinas de suporte (como inversão de sinal e reset de variável) também fazem parte desta coleção. Dependendo do corpo finito escolhido para o ECC - F_p ou F_{2^m} - a estrutura de dados que representa um elemento numérico nesta camada é diferente. Para F_p será necessária uma representação para inteiros grandes (na faixa de 0 a $p-1$) que é feita normalmente em precisão múltipla. Assim, tira-se proveito do tamanho da palavra do processador e a aritmética é implementada em termos de operações de palavras, utilizando as instruções primitivas do processador. Já em F_{2^m} os elementos são representados como cadeias de bits de tamanho m e a aritmética é implementada em termos de operações de bits. Em [24] e [36] são apresentados algoritmos para aritmética em precisão múltipla. Os algoritmos para F_{2^m} podem ser encontrados em [44] e [31]. Bibliotecas públicas podem ser utilizadas nesta camada, como GNUMP [17], FREELIP [29] e outras.

Camada 2 : para o corpo F_p, as operações desta camada são a adição modular $(a + b \ mod \ p)$ e sua operação inversa, a subtração modular $(a - b \ mod \ p)$, bem como a multiplicação modular $(a * b \ mod \ p)$ e a sua operação inversa, o inverso multiplicativo. O inverso multiplicativo de um elemento é denotado por a^{-1} e tem a propriedade de $a \cdot a^{-1} \equiv 1 \ (mod \ p)$. Para o corpo F_{2^m} as operações são as mesmas, mas deve-se considerar que o cálculo do módulo é feito por um polinômio irredutível. Normalmente o cálculo do inverso multiplicativo é feito com o algoritmo estendido de Euclides ou outros com função similar como, por exemplo, o algoritmo binário de J. Stein apresentado em [24]. O inverso multiplicativo é uma operação muito cara e por isso existe a alternativa de se utilizar as coordenadas projetivas. Para F_{2^m} o algoritmo AIA (Almost Inverse Algorithm) proposto por Schroppel et al [47] tem se mostrado um eficiente método de se calcular o inverso multiplicativo.

Camada 3 : de um ponto de vista prático, o desempenho de um criptossistema baseado

em curvas elípticas (ECC), depende principalmente da eficiência no cômputo da multiplicação escalar em corpo elíptico [31]. Nesta camada fica uma importante operação de um ECC, a multiplicação elíptica $Q = kP$, onde Q e P são pontos da curva elíptica e k é um escalar. Também nesta camada ficam as operações de adição de 2 pontos diferentes e de duplicação de pontos, que podem ser representados por coordenadas afins ou coordenadas projetivas. Maiores detalhes a respeito das coordenadas projetivas, bem como as equações de soma e duplicação de pontos projetivos apresentados na norma IEEE P1363 podem ser encontrados no apêndice B.

Camada 4 : esta camada implementa o serviço de criptografia propriamente dito. Geralmente, qualquer esquema de criptografia baseado nos logaritmos discretos terá um análogo sobre as curvas elípticas como, por exemplo, os esquemas de Diffie e Hellman [14], ElGamal [16], dentre outros. Os mais comuns são aqueles padronizados pela IEEE P1363 [22], e o apêndice C fornece detalhes adicionais a respeito de alguns esquemas de criptografia que podem ser implementados nesta camada.

Camada 5 : é nesta camada que estará a aplicação que faz uso do serviço de criptografia como por exemplo o PGP, SSL, WTSL (camada de segurança do protocolo WAP) ou alguma outra camada de segurança de um protocolo de comunicação. Bibliotecas públicas podem ser utilizadas nesta camada, como o OpenPGP [42], OpenSSL [41] e outras.

3.5 Implementação de ECC em DSP

O desempenho de um ECC depende principalmente da eficiência das computações e algoritmos para se calcular $Q = kP$, operação esta que é utilizada na implementação de todos os esquemas e protocolos de segurança baseados em curvas elípticas, ou seja, caso uma implementação de um ECC seja conforme a modularização apresentada na seção anterior, as camadas 1, 2 e 3 serão as principais responsáveis pelo desempenho deste

ECC. Portanto, neste trabalho para avalição de ECC em DSP foram implementadas estas 3 camadas.

O desempenho do ECC pode ser acelerado também pela escolha adequada do corpo finito no qual a curva elíptica está contida, que poderá ser primo F_p ou binário F_{2^m}.

Corpos finitos binários F_{2^m} têm se mostrado bastante eficientes e normalmente são recomendados para ambientes computacionais onde exista restrição de recursos. Dentre os vários motivos dessa eficiência está sua aritmética *carry-free* (não é necessário o "vai-um" na soma de dois números), bem como a disponibilidade de diferentes representações para este corpo [7].

Outra característica da aritmética em F_{2^m} é que a multiplicação dos elementos neste corpo dispensa a instrução de multiplicação primitiva do processador, pois pode ser uma seqüência de operações de deslocamentos e somas. Isto é interessante, já que a multiplicação costuma ser uma das mais caras instruções dos processadores *CISC*.

No entanto, a multiplicação nos DSPs é uma instrução rápida, normalmente executada em somente um ciclo de máquina, visto ser usada intensamente nos cálculos de filtros digitais.

Fica então caracterizada a seguinte questão: *corpos finitos binários F_{2^m} se comportam mais eficientemente em DSPs que corpos finitos primos F_p, considerando que nos DSPs a instrução de multiplicação é rápida ?*

Para esclarecer esta questão, implementou-se um ECC sobre F_p e outro sobre F_{2^m} usando a plataforma de DSP apresentada. Duas abordagens diferentes foram usadas para corpos finitos primos F_p. Se forem representados por coordenadas afins, para cada soma ou duplicação de pontos da curva elíptica, será necessária uma operação de inverso multiplicativo. Esta operação é demasiadamente cara pois é baseada no algoritmo estendido de Euclides, que não é um algoritmo eficiente. A representação por coordenadas projetivas resolve este problema pois dispensa o cálculo do inverso multiplicativo. Entretanto, demanda um maior número de multiplicações, como pode ser visto na Tabela 3.1. Apesar da operação inverso multiplicativo ser cara, é necessário conhecer bem a relação entre seu custo e o de uma mulplicação. Embora a literatura [11] cite normalmente valores como por exemplo 23:1 para F_p e 3:1 para F_{2^m} (utilizando

Operação	coordenadas afins	coordenadas projetivas
Adição geral (P+Q)	1 I + 3 M	16 M
Duplicação (2P)	1 I + 4 M	10 M

Tabela 3.1: Custo de adição e duplicação de pontos, em F_p, para $p > 3$, onde I = Inversão e M = Multiplicações

o algoritmo AIA para cálculo da operação inverso multiplicativo sobre corpos binários [47]) deve ser ressaltado que estes valores são significativos apenas dentro do contexto de uma dada plataforma. Se esta muda, estas relações precisam ser reavaliadas e isto pode influenciar na escolha da representação dos corpos finitos.

Para corpos finitos binários F_{2^m}, utilizou-se a representação polinomial na representação da base.

O programa (camadas 1, 2 e 3) foi totalmente escrito em C utilizando o *Code Composer Studio*, ambiente de desenvolvimento da Texas Instruments, voltado para aplicações em DSPs. Apesar deste ambiente de desenvolvimento ser de alto nível e moderno, houve dificuldades na sua integração com o kit de desenvolvimento do DSP utilizado (TMS320VC5402).

3.6 Resultados

A Tabela 3.2 apresenta os resultados das implementações para os corpos F_p (p = 112,128,160,192,224) e F_{2^m} (m = 113,131,163,193,233). As curvas elípticas utilizadas nos testes (detalhadamente descritas no apêndice D) estão de acordo com a norma P1363 [22] e são recomendadas pelo *Standards for Efficient Criptography - SEC1* [18]. O padrão SEC1 foi criado por um consórcio de empresas do segmento de comunicações móveis tais como 3Com, Fujitsu, Motorola entre outras e propõe algumas curvas elípticas sobre corpos finitos com cardinalidades menores que as curvas elípticas apresentadas em outros padroẽs como por exemplo o NIST. Por este motivo o padrão SEC1 é mais recomendado para ambientes computacionais restritos. Na Tabela 3.2 são apresentados os tamanhos

dos corpos (que pode ser interpretado como o tamanho da chave ou o nível de segurança) e os respectivos tempos (em segundos) requeridos para o cálculo de $Q = kP$. É importante salientar que o nível de segurança de ECC de 172 bits é equivalente ao RSA de 1024 bits ([22] - anexo D). Foram testados diversos valores de k buscando combinações binárias que certificassem casos extremos da multiplicação escalar elíptica (algoritmo double and add), como por exemplo todos bits de k em 1, bits 1 e 0 intercalados, etc. e os resultados são as médias aritméticas dos vários testes. Em todos eles foi observado praticamente o mesmo comportamento para o desempenho dos algoritmos.

Da análise da tabela, verifica-se que a implementação em F_p, utilizando coordenadas projetivas apresenta melhor desempenho em relação às outras. Também deve ser observado o comportamento da operação $Q = kP$ para F_p representado por coordenadas afins e F_{2^m}. A partir de 193 bits, a implementação de F_{2^m} apresenta desempenho inferior ao desempenho de F_p (192 bits) representado por coordenadas afins, o que demonstra que em DSPs os corpos primos são mais adequados para corpos com cardinalidade grande. Deve-se ressaltar que para o cálculo de $Q = kP$ foram utilizados métodos e algoritmos básicos apresentados na IEEE P1363 que, apesar de já otimizados de certa forma, não representam as abordagens mais eficientes da atualidade. Certamente pode-se procurar otimizar o código utilizando métodos e algoritmos mais eficientes.

Além da análise de desempenho para as 3 abordagens, fez-se um levantamento da área de programa requerida para suas implementações, visto que em ambientes computacionais restritos, deve-se procurar otimizar ao máximo o uso de memória. A Tabela 3.3 apresenta o espaço de memória alocado ao programa.

Nota-se que a implementação em coordenadas projetivas apresenta o melhor desempenho mas necessita de mais área de programa. Já a simplicidade na implementação de ECC para F_{2^m} reflete-se na pequena quantidade de memória requerida para o programa. Nas áreas de memória apresentadas não estão incluídas as áreas de dados e pilha.

Implementação	112	113	128	131	160	163	192	193	224	233
F_p coord. proj.	0,891	-	1,159	-	1,794	-	2,609	-	3,727	-
F_p coord. afins	1,128	-	1,547	-	2,561	-	3,961	-	6,017	-
F_{2^m}coord. afins	-	1,046	-	1,492	-	2,558	-	4,121	-	7,424

Tabela 3.2: Tempo (em segundos) da operação $Q = kP$ para corpos primos (coordenadas afins e projetivas) e corpo binário.

Camadas	F_p PROJ.	F_p AFINS	F_{2^m} AFINS
Camada 3	3370	1458	1376
Camada 2	922	922	542
Camada 1	2124	2124	1806
Total	6416	4504	3724

Tabela 3.3: Espaço (em bytes) alocado em memória para área de programa das camadas.

3.7 Conclusões

Foi apresentada uma proposta de modularização para implementação de ECC em 5 camadas e realizou-se a implementação de 3 camadas em um processador de sinais digitais consideravelmente limitado (recursos restritos). Esta implementação é provavelmente uma das primeiras publicadas na literatura em uma família de DSP voltada para aplicações em dispositivos portáteis.

Foram usadas as seguintes opções de ECC a fim de avaliar qual corpo finito (primo ou binário) melhor se adequava a arquitetura de um DSP: corpos finitos binários F_{2^m}, representados por coodernadas afins e corpos finitos primos F_p, representados por coordenadas afins e por coordenadas projetivas. De acordo com os testes realizados ficou comprovado que em tais processadores a implementação de curvas elípticas sobre corpos finitos primos pode ter um desempenho superior ao de uma implementação de curvas elípticas sobre corpos finitos binários, fato expressivo visto que corpos finitos binários frequentemente são adotados pelos projetistas quando a plataforma apresenta recur-

sos limitados, o que demonstra que é necessário fazer uma escolha bastante cuidadosa dos vários parâmetros e condições de operação das curvas elípticas, para que a característica de ECC de prover segurança com economia de recursos seja bem aproveitada em DSPs que apresentem recursos limitados. Além dos processadores de sinais digitais, tais considerações também devem ser levadas em conta para outras arquiteturas de processamento que ofereçam multiplicação rápida.

Capítulo 4

Proposta de Classificação de ECC e Análise dos Resultados

Praticamente todos os trabalhos apresentados no capítulo 2 (seções 2.3.1 a 2.3.8) reportaram os tempos do cômputo da operação de multiplicação elíptica, ou seja $Q = kP$, dados k um número escalar e P um ponto da curva elíptica, visto esta ser sempre utilizada na implementação dos esquemas e protocolos de segurança baseados em ECC. Como esta operação tem profunda influência no desempenho de um ECC, é possível de ser realizada uma análise comparativa dos resultados das implementações, para que se encontre as melhores escolhas de abordagens algorítmicas, versus plataformas de processamento.

Considerando que os trabalhos anteriores normalmente apresentam vários resultados (várias alternativas de implementação), para se fazer a análise escolheu-se aquelas que apresentam o melhor desempenho. A Tabela 4.1 ilustra os tempos dos melhores resultados conseguidos pelos autores e suas respectivas plataformas, ordenados por desempenho, ou seja, ordem crescente de tempo da operação $Q = kP$. Normalmente, o desempenho dos criptossistemas é avaliado pelo tempo que leva para efetuar uma determinada operação. Na verdade esta medida de tempo é uma medida relativa, pois depende fortemente do clock do processador e do tamanho do corpo, tornando mais difícil detectar se a escolha dos parâmetros e das técnicas de implementação levaram

Seção	Plataforma	Clock (em MHz)	Corpo finito	Tam. Corpo (em bits)	Tempo $Q = kP$ (em ms)
2.3.4	TMS320C6201	200	primo	160/192	2,88/4,15
2.3.1	PentiumPro	200	primo	191	21,1
2.3.7	ARM7TDMI	80	primo	160/192	44,8/69,2
2.3.2	PC	200	binário	192	96,4
2.3.3	M16C	10	primo	160	130
2.3.8	PalmPilot	16	binário	163	870
2.3.6	8051	12	OEF	$F_{(2^8-17)^{17}}$	1830
2.3.2	MC68302	25	binário	191	16000

Tabela 4.1: Melhores resultados conseguidos pelos autores e suas respectivas plataformas, ordenados pelo tempo da operação $Q = kP$.

realmente a um bom resultado. Sendo assim, se faz necessário buscar outra métrica para uma melhor avaliação de desempenho.

4.1 Métrica para Avaliação de Implementações de ECC

Esta métrica pode ser definida como uma medida ou estimativa da qualidade do projeto de um ECC. Ela permite ao projetista avaliar antecipadamente o projeto comparando estimativas e encontrando as melhores combinações de parâmetros para sua plataforma de processamento, evitando assim, o desenvolvimento de várias abordagens para posterior levantamento de desempenho.

Segundo *Gajski et al.* [19], existem dois tipos de métricas para software: métricas de custo e métricas de desempenho. Quando se dispõe das duas, o melhor critério de avaliação de uma determinada implementação é o da melhor relação custo/benefício, ou seja, (métrica de custo)/(métrica desempenho).

Métricas de custo

Sistemas implementados em software são compilados em um conjunto de instruções de um determinado processador. As duas métricas de custo associadas com o custo de implementação são a quantidade de memória de programa e a quantidade de memória de dados (incluindo pilha), requeridas para a execução do software em um processador específico. Cabe ao projetista de software escolher as melhores abordagens algorítmicas considerando a plataforma de processamento, pois algumas opções podem ser mais caras em determinadas plataformas que em outras.

Métricas de desempenho

As métricas de desempenho podem ser divididas em desempenho de computação e desempenho de comunicação. A métrica de desempenho de comunicação está relacionada com o tempo necessário para a interação e transferência de dados entre sistemas. A métrica de desempenho de computação está relacionada com o tempo necessário para a execução das tarefas do sistema. Ao projetista de software cabe escolher as melhores abordagens algorítmicas considerando a plataforma de processamento, pois algumas opções podem ser mais lentas em determinadas plataformas que em outras.

São raros os artigos que apresentam ECCs completos e com informações sobre o uso de memória de dados e de programa em suas implementações, inviabilizando assim uma avaliação baseada no custo da implementação. Conforme já apresentado, normalmente o desempenho dos criptossistemas é avaliado pelo tempo que se levou para efetuar uma determinada operação, mas na verdade esta medida de tempo é uma medida relativa.

A métrica para se avaliar um ECC deve ser independente da velocidade do processador utilizado e do nível de segurança escolhido e deve permitir perceber as melhores combinações de software e hardware que levam a relações custo/benefício (recursos/segurança) mais interessantes.

Considerando que quanto maior o tamanho de uma chave criptográfica, maior é a segurança de um criptossistema e que o tamanho da chave está diretamente relacionado com o tamanho (em bits) do corpo finito, propõe-se uma normalização da velocidade de

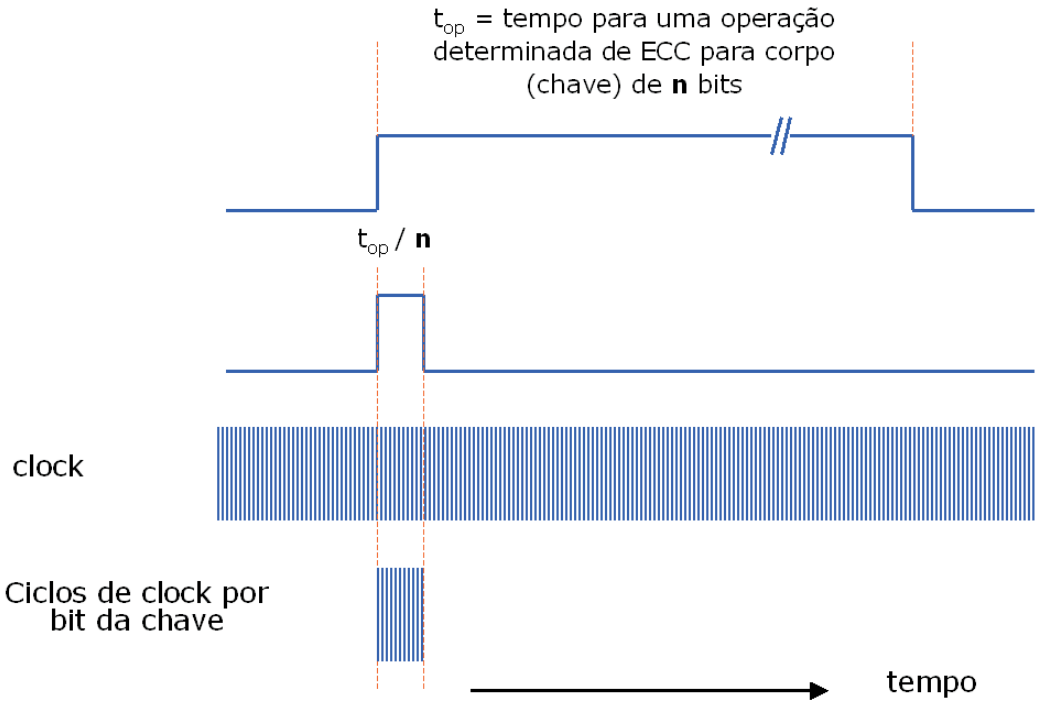

Figura 4.1: Normalização do tempo de uma operação pelo número de bits do corpo

processamento pelo tamanho do corpo, mais especificamente a normalização do número de ciclos de clock para cada bit do corpo (chave) necessários para efetuar uma determinada operação de um criptossistema, como ilustrado na Fig. 4.1.

Ex1: a melhor implementação de multiplicação elíptica ($Q = kP$) de Bogdan Antonescu (1999) [3], sobre PC - 200Mhz, para um corpo finito $F_{2^m}, m = 192$, levou 96,4 ms. A métrica deste ECC é 100416,6 ciclos de clock por bit da chave.

Ciclos de clock por bit da chave = (0,0964/192) $\times$ 200.000.000 = 100416,6

Ex2: a melhor implementação de multiplicação elíptica ($Q = kP$) de Hasegawa, Nakajima e Matsui (1998) [21], sobre o microcontrolador da Mitsubshi M16C - 10MHz, para um corpo finito F_p, $\lceil log_2 p \rceil = 160$ bits, levou 130 ms.

Ciclos de clock por bit da chave = (0,130/160) $\times$ 10.000.000 = 8125,0

Como pode ser observado nos exemplos acima, apesar do resultado da operação do

Seção	Plataforma	Clock (em MHz)	Corpo finito	Tam. Corpo	Ciclos de clock/bit da chave
2.3.4	TMS320C6201	200	primo	160	3600
2.3.4	TMS320C6201	200	primo	192	4322
2.3.3	M16C	10	primo	160	8125
2.3.1	PentiumPro	200	primo	191	22094
2.3.7	ARM7TDMI	80	primo	160	22400
2.3.7	ARM7TDMI	80	primo	192	28833
2.3.8	PalmPilot	16	binário	163	85398
2.3.2	PC	200	binário	192	100416
2.3.6	8051	12	OEF	$F_{(2^8-17)^{17}}$	163880
2.3.2	MC68302	25	binário	191	2094240

Tabela 4.2: Melhores resultados conseguidos pelos autores, ordenados segundo a métrica proposta: ciclos de clock por bit da chave

segundo exemplo, em termos absolutos, ser pior que o do primeiro exemplo, na verdade a implementação do segundo exemplo dispendeu um esforço computacional quase 12 vezes menor para efetuar os cálculos para cada bit do corpo (chave), o que demonstra que esta teve uma melhor escolha de abordagens algorítimicas e uma codificação mais eficiente para a sua plataforma de hardware.

A Tabela 4.2 relaciona as propostas apresentadas no capítulo 2 ordenadas segundo a métrica proposta [1] : *ciclos de clock por bit da chave.*

4.2 Análise de Resultados

Nota-se que nem sempre as implementações de maior frequência de clock foram as mais eficientes como, por exemplo, a de Hasegawa, Nakajima e Matsui [21] sobre o microcon-

[1] Para o trabalho sobre corpo OEF $F_{(2^8-17)^{17}}$ (2.3.6), utilizou-se 134 bits como tamanho da chave, apesar de ser representado por 17 bytes, visto ter segurança equivalente a corpos finitos binários $F_{2^{134}}$.

trolador MC16C. Este microcontrolador possui uma instrução de multiplicação relativamente rápida com somente três ciclos de máquina por instrução de multiplicação. O melhor desempenho desta implementação certamente vem do fato da aritmética sobre corpos finitos primos fazer uso intensivo da instrução de multiplicação. Outra implementação que apresentou um melhor posicionamento foi a de Weimerskich, Paar e Shantz [52] sobre o PalmPilot. Ambas tiveram suas posições melhoradas em função da métrica representar de maneira apropriada as propostas de criptossistemas que tiveram uma relação custo/benefício superior.

Na Tabela 4.3 é apresentada uma comparação do melhor resultado da implemetação do ECC sobre DSP, apresentada no capítulo 3 (F_p, utilizando coordenadas projetivas), com os demais trabalhos apresentados no capítulo 2, segundo a métrica ciclos de clock por bit da chave. Observa-se que a nova implementação em camadas não obteve um bom desempenho quando comparada com as melhores implementações. A justificativa para tal deve-se ao fato de que foram utilizados métodos e algoritmos básicos, não otimizados, e que não tornaram a aplicação de ECC tão eficiente quanto possível. Certamente, pode-se procurar otimizar o código utilizando métodos e algoritmos mais eficientes, como feito nos demais trabalhos. Além disso, o fato da implementação apresentada no capítulo 3 ser bastante modularizada, trazendo vantagens como a facilidade de integração e teste das unidades funcionais de um ECC, trouxe perdas na velocidade, comum quando se usa muitas funções aninhadas, visto o uso intenso da pilha para a passagem de parâmetros.

4.3 Conclusões

A métrica *ciclos de clock por bit da chave* é uma métrica comum, onde se busca revelar se os parâmetros de ECC, algoritmos e linguagem de programação estão bem adaptados às suas plataformas de hardware. Apesar das imprecisões da métrica, ainda assim ela permite perceber as melhores combinações de software e hardware que levaram a relações custo/benefício (recursos/segurança) mais vantajosas.

Seção	Plataforma	Clock (em MHz)	Corpo finito	Tam. Corpo	Ciclos de clock/bit da chave
2.3.4	TMS320C6201	200	primo	160	3600
2.3.4	TMS320C6201	200	primo	192	4322
2.3.3	M16C	10	primo	160	8125
2.3.1	PentiumPro	200	primo	191	22094
2.3.7	ARM7TDMI	80	primo	160	22400
2.3.7	ARM7TDMI	80	primo	192	28833
2.3.8	Palmtop	16	binário	163	85398
2.3.2	PC	200	binário	192	100416
2.3.6	8051	12	OEF	$F_{(2^8-17)^{17}}$	163880
3.5	TSM320VC5402	100	primo	112	795535
3.5	TSM320VC5402	100	primo	128	905438
3.5	TSM320VC5402	100	primo	160	1121250
3.5	TSM320VC5402	100	primo	192	1358894
3.5	TSM320VC5402	100	primo	224	1663893
2.3.2	MC68302	25	binário	191	2094240

Tabela 4.3: Comparação com a implentação apresentada no capítulo 3, ordenados segundo a métrica: ciclos de clock por bit da chave

Capítulo 5

Conclusões e Trabalhos Futuros

Nesta dissertação vimos que para se implementar criptossitemas baseados em curvas elípticas são muitas as escolhas que devem ser feitas, desde o tipo de curva elíptica, passando por sua representação, e chegando na definição de algoritmos a serem aplicados. Foram apresentados trabalhos que implementaram sistemas completos, apresentando uma boa variedade de alternativas de implementação sobre várias plataformas de hardware (microcontroladores, processadores digitais de sinais, processadores RISC e arquitetura Intel x86) e que, consequentemente, tiveram um desempenho determinado pelos parâmetros e pelas técnicas de implementação escolhidas pelos seus autores. Buscou-se analisar as opções para tentar identificar algum direcionamento das melhores escolhas para a implementação de ECC. No entanto, vimos que esta análise é dificultada pela diversidade dos cenários de implementação (plataformas de processamento, níveis de segurança, etc).

Apresentamos uma métrica, *Ciclos de clock por bit da chave*, para se avaliar ECC buscando uma independência da velocidade do processador utilizado e do nível de segurança escolhido, permitindo assim perceber as melhores combinações de software e hardware que levam a relações custo/benefício (recursos/segurança) mais vantajosos.

Foi apresentada uma proposta de modularização de implementação de ECC em 5 camadas e realizou-se a implementação de 3 camadas em um processador de sinais digitais consideravelmente limitado (recursos restritos). Acreditamos que esta implementação é

uma das primeiras publicada na literatura em uma família de DSP voltada para aplicações em dispositivos portáteis. De acordo com essa implementação, curvas elípticas sobre corpos finitos primos representados por coordenadas projetivas têm um desempenho superior ao de uma implementação de curvas elípticas sobre corpos finitos binários em DSP, apesar de requerer mais memória de programa para a sua implementação. Tal comportamento se deve ao fato que em DSP a instrução multiplicação é consideravelmente rápida. Além dos DSPs, este comportamento provavelmente se repetirá em outras arquiteturas de processamento que ofereçam multiplicação rápida.

Fizemos uma análise comparativa do desempenho de nossa implementação com as melhores implementações dos outros autores apresentados nesta dissertação e constatamos que nossa implemetação não obteve um bom desempenho . A justificativa para tal deve-se ao fato de que foram utilizados métodos e algoritmos básicos, não otimizados, e que não tornaram a aplicação de ECC eficiente na forma como foi implementada.

Para trabalhos futuros são várias as frentes de pesquisa que podem ser seguidas:

- Otimização das camadas inferiores, utilizando algoritmos mais eficientes.
- Implementação de rotinas básicas em assembly para se tirar proveito da arquitetura de DSP.
- Análise do comportamento de aritmética sobre base normais em DSP .
- Análise do comportamento de aritmética OEF (Optimal Extension Field) em DSP.
- Melhoria da métrica de avaliação de ECC levando em conta o custo do espaço de alocação de programa e de dados.
- Melhoria da métrica de avaliação de ECC levando em conta o crescimento não linear da complexidade algorítmica das operações inerentes a um ECC.

Apêndice A

Introdução às curvas elípticas

Muitos criptossistemas de chave pública requerem o uso de grupos algébricos. Um grupo é um conjunto de elementos que podem ser combinados através de uma operação, tal como a adição ou a multiplicação, e que satisfazem certas condições. A seguir são apresentadas definições dos grupos aditivos e multiplicativos. Definições mais detalhadas podem ser encontradas em [36].

A.1 Grupos Algébricos

Grupo Aditivo : um grupo aditivo G consiste de um conjunto de números com uma operação binária (operação entre 2 elementos) que satisfaz os seguintes axiomas:

(i) a operação é associativa, ou seja, $a + (b + c) = (a + b) + c$;

(ii) existe um elemento identidade $0 \in G$, ou seja, $a + 0 = a$;

(iii) existe uma operação inversa de forma que, para cada $a \in G$, existe um elemento inverso $(-a)$ tal que $a + (-a) = 0$.

A operação é chamada de abeliana se for comutativa. Ex: $a + b = b + a$.

Grupo Multiplicativo : um grupo multiplicativo G consiste de um conjunto de números com uma operação binária (operação entre 2 elementos) que satisfaz os seguintes

axiomas:

(i) a operação é associativa, ou seja, $a \times (b \times c) = (a \times b) \times c$;

(ii) existe um elemento identidade $1 \in G$, ou seja, $a \times 1 = a$;

(iii) existe uma operação inversa de forma que, para cada $a \in G$, existe um elemento inverso (a^{-1}) tal que $a \times a^{-1} = 1$.

A operação é chamada de abeliana se for comutativa. Ex: $a \times b = b \times a$.

A.2 Curvas Elípticas sobre os Números Reais

Uma curva elíptica sobre números reais pode ser definida como um conjunto de pontos *(x,y)* que satisfazem uma equação da forma:

$$y^2 = x^3 + ax + b, \tag{A.1}$$

onde x, y, a e b são números reais.

Cada valor de a e b define uma curva elíptica diferente. Por exemplo, a equação $y^2 = x^3 - 4x + 0.67$ gera a curva apresentada na Fig. A.1. Um ponto imaginário chamado de *ponto no infinito* [1] e representado por $\boldsymbol{O}$, deve ser adicionado ao conjunto de elementos (pontos) da curva elíptica para satisfazer os axiomas da formação de grupo; ele fará o papel do elemento identidade do grupo aditivo (elemento zero).

Para que o terceiro axioma relativo a formação de grupo aditivo seja respeitado, isto é, deve existir somente um elemento inverso $(-a)$ tal que $a + (-a) = 0$, uma curva elíptica deve satisfazer a Eq. A.2. Esta é a mesma equação do cálculo do discriminante de equações do terceiro grau (método Tartaglia/Cárdano). Se o discriminante for igual a zero, existem duas raízes reais e uma raíz imaginária que satisfazem a equação. Logo, evita-se a utilização de curvas com discriminante nulo, pois isto implica na existência

[1] Termo original em inglês: *point at infinity*

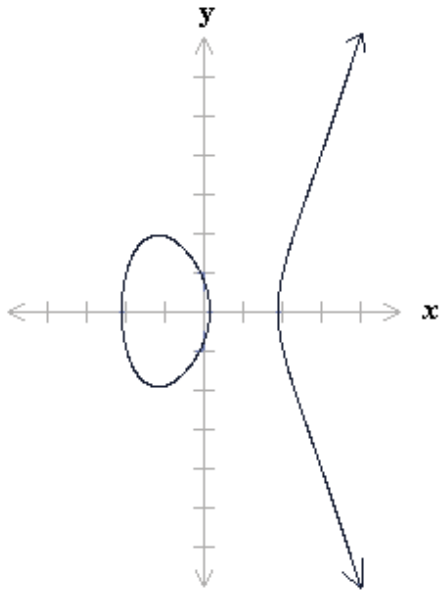

Figura A.1: Curva elíptica $y^2 = x^3 - 4x + 0.67$

de dois pontos repetidos (as duas raízes reais). Este teste é chamado de *teste de não singularidade* [35].

$$4a^3 + 27b^2 \neq 0 \tag{A.2}$$

Os grupos das curvas elípticas são grupos aditivos, ou seja, suas operações básicas são a adição e sua operação inversa, e estas podem ser solucionadas geometricamente. A seguir são apresentadas as soluções geométricas para as operações sobre curvas elípticas.

Adição de dois pontos distintos : a adição de um ponto P a um ponto Q pode ser calculada geometricamente traçando-se uma linha entre os dois pontos P e Q e projetando-a de tal forma a interceptar um terceiro e único ponto na curva elíptica. Este ponto é o negativo do resultado $(-R)$, de forma que o resultado da adição R é a sua reflexão no eixo x, como pode ser verificado na soma $P(-2.35, -1.86) + Q(-0.1, 0.836)$ sobre a curva elíptica $y^2 = x^3 - 7x$, apresentada na Fig. A.2.

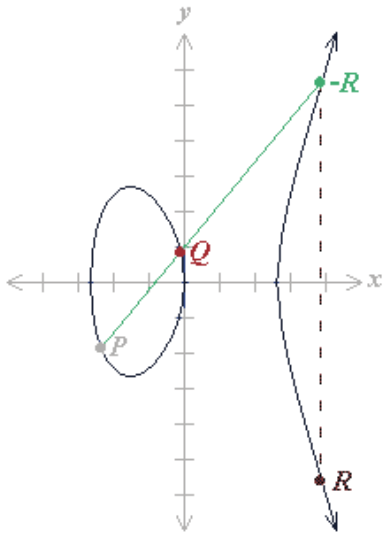

Figura A.2: P+Q = R(3.89.-5,62)

Adição de dois pontos P e $-P$: traçando uma linha vertical por P e $-P$, não ocorre intesecção na curva elíptica num terceiro ponto. Por definição $P + (-P) = \boldsymbol{O}$ (ponto no infinito), assim como o resultado de $P + \boldsymbol{O}$ é P, como pode ser verificado na curva elíptica $y^2 = x^3 - 6x + 6$, apresentada na Fig. A.3.

Duplicação de pontos (cálculo de 2P) : uma linha tangenciando P deve ser traçada até que esta intercepte a curva elíptica. Este ponto é $-R$ e sua reflexão no eixo x é o resultado desejado R, como pode ser visto na Fig. A.4. Caso $y_p = 0$, não haverá a intesecção pois se trata do ponto no infinito, ou seja, se $y_p = 0$, então $2P = \boldsymbol{O}$.

A abordagem geométrica é uma excelente forma de ilustrar a aritmética das curvas elípticas, apesar de não ser uma forma prática de implementação algorítmica, o que é feito com o método algébrico, conforme apresentado a seguir.

Método algébrico de adição de dois pontos distintos : considerando $P = (x_p, y_p), Q = (x_q, y_q)$ *e* $R = (x_r, y_r)$, a adição $R = P + Q$ é dada pelas Eqs. A.3 a A.5.

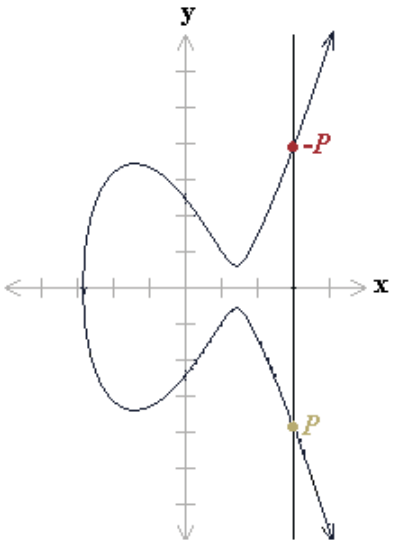

Figura A.3: P+(-P) = ***O***

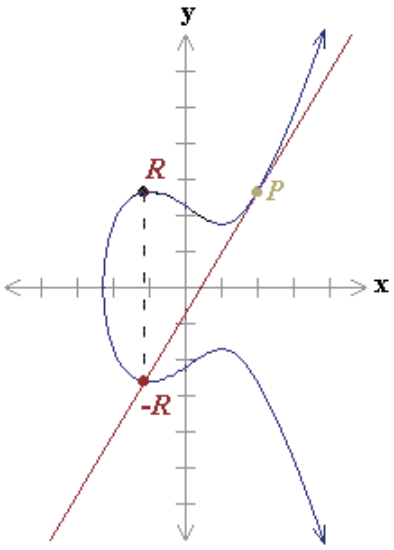

Figura A.4: P+P=2P=R

$$s = \frac{y_p - y_q}{x_p - x_q} \tag{A.3}$$

$$x_r = s^2 - x_p - x_q \tag{A.4}$$

$$y_r = s(x_p - x_r) - y_p \tag{A.5}$$

Método algébrico de duplicação de P (cálculo de $2P$): quando y_p não é igual a 0 as coordenadas de R=2P podem ser calculadas pelas Eqs. A.6 a A.8.

$$s = \frac{3x_p^2 + a}{2y_p} \tag{A.6}$$

$$x_r = s^2 - 2x_p \tag{A.7}$$

$$y_r = s(x_p - x_r) - y_p \tag{A.8}$$

A.3 Curvas Elípticas sobre Corpos Finitos

A.3.1 Corpos Finitos

Sistema algébrico consistindo de um conjunto finito F e de duas operações binárias, *adição* e *multiplicação*, definidas em F e satisfazendo os seguintes axiomas:

- F é um grupo abeliano [2] com respeito a $(+)$;
- $F \setminus \{0\}$ é um grupo abeliano com respeito a $(\times)$;
- Para todos x, y e z em F tem-se que (propriedade distributiva):

[2] Abeliano: que possui propriedades comutativas.

$$x \times (y + z) = (x \times y) + (x \times z) \tag{A.9}$$

$$(x + y) \times z = (x \times z) + (y \times z). \tag{A.10}$$

Um corpo finito, denotado por F_q ou $GF(q)$, é um corpo com q elementos. A *ordem* de um corpo finito é o número de elementos pertencentes a este corpo. Tais corpos finitos existem se $q = p$, ou $q = p^m$, onde p é um número primo e m um inteiro positivo. Uma revisão da teoria dos números necessária ao estudo da aplicação de corpos finitos em criptografia, bem como definições mais detalhadas dos conceitos aqui apresentados podem ser encontradas em [36].

A.3.2 Operações de Curvas Elípticas sobre Corpos Finitos

As operações de adição e duplicação de pontos de curvas sobre um corpo finito F_p são descritas a seguir.

Adição de pontos distintos : considerando $P = (x_p, y_p), Q = (x_q, y_q)$ *e* $R = (x_r, y_r)$, tem-se que a adição $R = P + Q$ é dada pelas Eqs. A.11 a A.13.

$$s = \frac{y_p - y_q}{x_p - x_q} \ \ mod\ p \tag{A.11}$$

$$x_r = s^2 - x_p - x_q \ \ mod\ p \tag{A.12}$$

$$y_r = -y_p + s(x_p - x_r) \ \ mod\ p. \tag{A.13}$$

Duplicação de pontos : quando y_p não é igual a zero, as coordenadas de $R(x_r, y_r) = 2P$ são obtidas das Eqs. A.14 a A.16.

$$s = \frac{3x_p^2 + a}{2y_p} \ \ mod\ p, \tag{A.14}$$

$$x_r = s^2 - 2x_p \mod p \tag{A.15}$$

$$y_r = s(x_p - x_r) - y_p \mod p \tag{A.16}$$

A.3.3 Corpos Finitos na Forma Polinomial F_{2^m}

Corpos finitos sobre F_{2^m}, também chamados de *corpos de característica dois ou corpos binários* são muito vantajosos do ponto de vista computacional, pois existem técnicas eficientes para execução das operações aritméticas. Além de simplificar a implementação tanto em hardware quando em software, eles possibilitam melhorias significativas em termos de desempenho. Ao serem implementados em hardware, por exemplo, os chips podem ser otimizados e ter sua área reduzida, proporcionando vantagens econômicas.

Um corpo finito binário F_{2^m} pode ser visualizado como um vetor de espaço de dimensão m sobre F_2, o que significa que existe um conjunto de m elementos $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$ em F_{2^m} , tais que cada $\alpha \in F_{2^m}$ pode ser escrito na forma polinomial :

$$\alpha = \alpha_{m-1}x^{m-1} + \ldots + \alpha_1 x + \alpha_0 \tag{A.17}$$

ou simplesmente como

$$\alpha = \sum_{i=0}^{m-1} \alpha_i x^i,$$

onde $\alpha_i \in \{0, 1\}$.

Além da representação polinomial, existem outras formas de se descrever a aritmética em F_{2^m}, como por exemplo as representações vetorial e de base normal entre outras.
A redução do polinômio é feita com a operação módulo *polinômio irredutível*.

> **Polinômios irredutíveis :** um polinômio irredutível f(x) pertencente ao corpo finito é aquele que não é divisível por qualquer outro polinômio de grau menor que o seu [28].

Os polinômios irredutíveis têm na aritmética polinomial o mesmo papel que os números primos tem na aritmética inteira, de forma que considerando $f(x) \in Z_p[x]$ [3] um polinômio irredutível de grau m, então $Z_p[x]/f(x)$ será um corpo finito de ordem p^m.

A aritmética de corpos finitos pode ser implementada de forma mais eficiente se os polinômios irredutíveis possuirem poucos termos não nulos, como por exemplo trinômios e pentanômios, conforme sugerido no capítulo 4 de [36] e recomendado em [22].

A.3.4 Operações sobre Corpos Finitos F_{2^m}

Em F_{2^m} a equação das curvas elípticas pode ser dada por

$$y^2 + xy = x^3 + ax^2 + b, \tag{A.18}$$

com a única condição de que b seja diferente de 0.

Adição de dois pontos distintos sobre F_{2^m}: considerando $P = (x_p, y_p)$, $Q = (x_q, y_q)$ e $R = (x_r, y_r)$, tem-se que a adição é dada pelas Eqs. A.19 a A.21.

$$s = \frac{y_q + y_p}{x_q - x_p}, \tag{A.19}$$

$$x_r = s^2 + s + x_p + x_q + a \tag{A.20}$$

$$y_r = s(x_p + x_r) + x_r + y_p \tag{A.21}$$

[3] $Z_p[x]$: conjunto dos polinômios com coeficientes em Z_p.

Duplicação de pontos (cálculo de 2P) sobre F_{2^m}: se $x_p = 0$, então $2P = \boldsymbol{O}$; se $x_p \neq 0$, $2P = R(x_r, y_r)$, tem-se que duplicação é dada pelas Eqs. A.22 a A.24.

$$s = x_p + \frac{y_p}{x_p} \tag{A.22}$$

$$x_r = s^2 + s + a \tag{A.23}$$

$$y_r = (x_p + x_p)s + x_r + y_p \tag{A.24}$$

Apêndice B

Coordenadas Projetivas

O sistema de coordenadas projetivas é um mecanismo que facilita os cálculos com pontos de curvas elípticas, pois dispensa a operação de inverso multiplicativo (a mais custosa) em suas equações de adição e duplicação. A desvantagem das coordenadas projetivas fica por conta do aumento considerável das operações de multiplicação. Para F_{2^m} existem métodos eficientes para se calcular a operação de inversão, diminuindo a razão inversão/multiplicação, de forma que raramente se implementam as coordenadas projetivas sobre F_{2^m}.

Nas Tabelas B.1 e B.2 são apresentadas as quantidades necessárias de operações de inversão (I), multiplicação (M) e quadrado (Q) para coordenadas afins e projetivas, para F_p e F_{2^m}, respectivamente.

Para F_p, a operação de quadrado tem praticamente o mesmo custo da multiplicação, de forma que é contabilizada como tal. Já em F_{2^m}, como a operação de quadrado tem custo muito mais baixo que a multiplicação, esta é apresentada separadamente. O termo a apresentado na Tabela B.2 se refere ao termo da Eq. $y^2 + xy = x^3 + ax^2 + b$ que é a equação da curva elíptica para corpos finitos binários.

Na dissertação de mestrado de Bogdan Antonescu [3], foram implementadas as coordenadas projetivas somente para F_p e preferiu-se utilizar para F_{2^m} o algoritmo AIA (Almost Inverse Algorithm) proposto por Schroppel et al. [47] como método eficiente de se calcular o inveso multiplicativo [3] para este corpo.

Operação	coordenadas afim	coordenadas projetivas
Adição geral (P+Q)	1 I + 3 M	16 M
Duplicação 2P	1 I + 4 M	10 M

Tabela B.1: Custo de adição de ponto em F_p, para $p > 3$

Operação	coordenadas afim	coordenadas projetivas
Adição ($a \neq 0$)	1 I + 2 M + 1 Q	15 M + 5 Q
Adição ($a = 0$)	1 I + 2 M + 1 Q	14 M + 4 Q
Duplicação 2P	1 I + 2 M + 1 Q	5 M + 5 Q

Tabela B.2: Custo de adição de pontos em F_{2^m}.

A norma P1363 [22] comenta que as coordenadas projetivas são bem ajustadas para computações internas mas não para comunicação externa, considerando que estas precisam de muito mais bits para serem representadas. Comenta também que estas coordenadas são mais comuns em F_p, considerando que a inversão tende a ser mais custosa nestes corpos.

Menezes [35] afirma que o uso de coordenadas projetivas proporciona ganhos em velocidade, visto dispensar a custosa operação de inverso multiplicativo, entretanto perde-se em espaço de memória de programa e de dados pois necessita-se de mais operações e mais variáveis temporárias para se implementar a soma e duplicação de pontos.

A seguir são apresentadas as fórmulas necessárias para a implementação de um sistema de coordenadas projetivas do tipo jacobiano (o mesmo apresentado na norma IEEE P1363 [22]).

Conversão de coordenadas do tipo afim (x, y) para o tipo projetivo (X, Y, Z):

$$X \leftarrow x, Y \leftarrow y, Z \leftarrow 1.$$

Conversão de coordenadas do tipo projetivo (X, Y, Z) para o tipo afim (x, y) :

$$x \leftarrow \frac{X}{Z^2}, y = \frac{Y}{Z^3}.$$

Duplicação de pontos sobre $\mathbf{F}_p$, $(p > 3)$: sobre a curva $y^2 = x^3 + ax + b \ mod \ p$

$$(X_2, Y_2, Z_2) = 2(X_1, Y_1, Z_1),$$

onde,

$$M = 3{X_1}^2 + a{Z_1}^4,$$
$$Z_2 = 2Y_1Z_1,$$
$$S = 4X_1{Y_1}^2,$$
$$X_2 = M^2 - 2S,$$
$$T = 8{Y_1}^4,$$
$$Y_2 = M(S - X_2) - T.$$

Adição de pontos sobre $\mathbf{F}_p$, $(p > 3)$: sobre a curva $y^2 = x^3 + ax + b \ mod \ p$

$$(X_2, Y_2, Z_2) = (X_1, Y_1, Z_1) + (X_0, Y_0, Z_0),$$

onde,

$$U_0 = X_0{Z_1}^2,$$
$$S_0 = Y_0{Z_1}^3,$$
$$U_1 = X_1{Z_0}^2,$$
$$S_1 = Y_1{Z_0}^3,$$
$$W = U_0 - U_1,$$
$$R = S_0 - S_1,$$
$$T = U_0 + U_1,$$
$$M = S_0 + S_1,$$
$$Z_2 = Z_0Z_1W,$$
$$X_2 = R^2 - TW^2,$$
$$V = TW^2 - 2X_2,$$
$$2Y_2 = VR - MW^3.$$

Duplicação de pontos sobre $\mathbf{F}_{2^m}$: sobre a curva $y^2 + xy = x^3 + ax^2 + b$

$$(X_2, Y_2, Z_2) = 2(X_1, Y_1, Z_1),$$

onde,

$$\begin{aligned} c &= b^{2^{m-2}}, \\ Z_2 &= X_1 {Z_1}^2, \\ X_2 &= (X_1 + c{Z_1}^2)^4, \\ U &= Z_2 + {X_1}^2 + Y_1 Z_1, \\ Y_2 &= {X_1}^4 Z_2 + U X_2. \end{aligned}$$

Adição de pontos sobre $\mathbf{F}_{2^m}$: sobre a curva $y^2 + xy = x^3 + ax^2 + b$

$$(X_2, Y_2, Z_2) = (X_1, Y_1, Z_1) + (X_0, Y_0, Z_0),$$

onde,

$$\begin{aligned} U_0 &= X_0 Z_1^2, \\ S_0 &= Y_0 {Z_1}^3, \\ U_1 &= X_1 {Z_0}^2, \\ W &= U_0 + U_1, \\ S_1 &= W_1 {Z_0}^3, \\ R &= S_0 + S_1, \\ L &= Z_0 W, \\ V &= RX_1 + LY_1, \\ Z_2 &= LZ_1, \\ T &= R + Z_2, \\ X_2 &= a{Z_2}^2 + TR + W^3, \\ Y_2 &= TX_2 + VL^2. \end{aligned}$$

Apêndice C

Esquemas Criptográficos Baseados em Curvas Elípticas

C.1 Diffie-Hellman

C.1.1 O Esquema Original

O método Diffie-Hellman se baseia em logaritmos discretos, onde em $\beta = \alpha^x \ mod \ n$, $0 \leq x \leq (n-1)$, é relativamente fácil calcular β dados α, x e n, mas é relativamente difícil calcular x dados β, α e n.

O esquema de negociação de chaves de sessão Diffie-Hellman consiste nos seguintes passos: dois usuários (referidos aqui como Ana e Beto) publicamente escolhem um corpo finito F_q (ver apêndice A) e um elemento fixo α deste corpo que seja um elemento primitivo (elemento gerador) do corpo. Ana seleciona então uma chave privada (um inteiro) k_A e publica a chave pública $P_A = \alpha^{k_A}$. Beto faz o mesmo com uma chave k_B. Ana pode então calcular $(\alpha^{k_B})^{k_A} = \alpha^{k_B k_A}$ e Beto pode calcular $(\alpha^{k_A})^{k_B} = \alpha^{k_A k_B}$. Este elemento comum para os usuários pode ser usado como uma chave de sessão para ser aplicada em algum algoritmo de criptografia de chave simétrica.

C.1.2 Diffie-Hellman sobre Curvas Elípticas

Nas curvas elípticas (Apêndice A) tem-se um problema equivalente ao dos logaritmos discretos quando se usa a equação $Q = kP$, onde Q, P $\in E(F_q)$ e $k < p$. É relativamente fácil calcular Q dado k e P, mas é demasiadamente difícil determinar k, dados Q e P.

Supondo que Ana e Beto queiram trocar de forma segura uma chave para ser usada posteriormente em algum sistema de criptografia simétrica, são os seguintes os passos a serem seguidos.

1. Ana e Beto escolhem publicamente um corpo finito F_q, uma Curva Elíptica E definida sobre F_q e um ponto fixo S na Curva Elíptica E.

2. Ana gera randomicamente um inteiro k_A, e o mantém em segredo, pois ele será sua chave privada.

   Beto faz o mesmo, ou seja, gera e mantém em segredo um inteiro k_B.

3. Ana calcula e publica $P_A = k_A S \in E(F_q)$.

   Beto calcula e publica $P_B = k_B S \in E(F_q)$.

4. Como Ana e Beto têm acesso a P_B e P_A, ambos podem calcular o mesmo ponto $P = k_A k_B S$.

   Para Ana, $P = k_A P_B = k_A(k_B S)$.

   Para Beto, $P = k_B P_A = k_B(k_A S)$.

5. Qualquer informação derivada do ponto P (sua coordenada x, por exemplo) pode ser usada pelas partes como uma chave de sessão para ser aplicada em algum algoritmo de chave simétrica.

C.2 ElGamal

C.2.1 O Esquema Original

Os passos para que Ana e Beto, usando o esquema de ElGamal, possam trocar uma mensagem ou chave de sessão são descritos a seguir.

Escolhe-se publicamente um corpo finito F_q e um elemento fixo α deste corpo que seja um elemento primitivo (elemento gerador) do corpo. O usuário Ana seleciona então uma chave privada k_A e publica a chave pública $P_A = \alpha^{k_A}$. Para o usuário Beto enviar uma mensagem M (que pode ser uma chave de sessão) para Ana, ele escolhe um r randômico e envia para ela o par $(\alpha^r, MP_A{}^r)$. Agora Ana, que conhece k_A, pode recuperar M calculando:

$$\frac{MP_A{}^r}{(\alpha^r)^{k_A}} = \frac{M\alpha^{rk_A}}{\alpha^{rk_A}} = M \tag{C.1}$$

C.2.2 ElGamal sobre Curvas Elípticas

1. Ana e Beto escolhem publicamente um corpo finito F_q , uma curva elíptica E definida sobre F_q e um ponto fixo $Z \in E(F_q)$

2. Ana gera randomicamente um inteiro k_A e o mantém em segredo pois ele será sua chave privada.

3. Ana calcula e publica $P_A = k_A Z \in E(F_q)$.

4. Para Beto enviar uma mensagem para Ana, por exemplo uma chave de sessão embutida no ponto $P_m \in E(F_q)$, ele gera randomicamente um escalar $r \in F_q$, calcula

$$P_r = rZ \tag{C.2}$$

$$P_h = P_m + rP_A \tag{C.3}$$

e envia a ela o par (P_r, P_h).

5. Para extrair a mensagem, Ana calcula

$$P_s = k_A P_r \tag{C.4}$$

e

$$P_m = P_h - P_s \tag{C.5}$$

O ponto P_m é recuperado corretamente como se mostra na Eq. C.6

$$P_m = P_h - k_A P_r = P_m + rP_A - k_A P_r = P_m + rk_A Z - k_A rZ = P_m \tag{C.6}$$

Para que Ana possa enviar uma mensagem para Beto, basta que Beto gere também uma chave privada k_B, publique $P_B = k_B Z \in E(F_q)$ e Ana calcule e envie (P_r, P_h).

C.3 DSA : Digital Signature Algorithm

Nos esquemas de assinatura digital, uma parte gera uma assinatura para uma mensagem com sua própria chave privada e a outra parte verifica a assinatura com a chave pública da parte que gerou a assinatura. O esquema de assinatura digital proporciona garantia de origem da mensagem.

Em agosto de 1991, o National Institute of Standards and Technology (NIST) propôs o Digital Signatute Algorithm (DSA) para uso em seu padrão de assinatura digital Digital Signature Standard (DSS) [45]. DSA é baseado em esquemas originalmente propostos por ElGamal e Schnorr [50], ou seja, na dificuldade de se computar logaritmos discretos.

C.3.1 O Algoritmo Original

- Geração de chave
  - Selecionar:
    - um número primo q tal que $2^{159} < q < 2^{160}$,
    - um número primo p tal que que $q|p-1$,

    * um elemento gerador $g \in Z_p^*$,
    * uma chave privada k_A tal que $1 < k_A < q - 1$.
  - Calcular a chave pública $P_A = g^{k_A} \ mod \ p$
  - Publicar p, q, g, P_A.

- Assinatura
  - Selecionar randomicamente um escalar k tal que $1 < k < q - 1$
  - Calcular:
    * $r = (g^k \ mod \ p) \ mod \ q$,
    * $e = hash(m)$ (obs: $hash(m) \Rightarrow$ função hash SHA-1 da mensagem m) e
    * $s = k^{-1}(e + K_A r)$
  - Enviar (r, s) anexados à mensagem m.

- Verificação de assinatura
  - $h = s^{-1} \ mod \ q$
  - $e' = hash(m')$ (m' é a mensagem recebida)
  - $h_1 = e'h \ mod \ q$
  - $h_2 = rh \ mod \ q$
  - $v = g^{h_1} {P_A}^{h_2}$
  - Teste : Se v = r, então a assinatura é aceita como correta.

C.3.2 A Prova do Teste

Para efeito de simplificação não serão apresentadas as operações $mod \ q$. A equação $h = s^{-1}$ pode ser reescrita da seguinte forma:

$$h = k(e + k_A r)^{-1}$$

Substituindo-a nas equaçõs de h_1 e h_2:

$$h_1 = e'[k(e + k_A r)^{-1}]$$

$$h_2 = r[k(e + k_A r)^{-1}]$$

Expandindo a equação de v tem-se

$$v = g^{h_1}(g^{k_A})^{h_2} = g^{h_1} g^{k_A h_2}$$

$$= g^{e'k(e+k_A r)^{-1}} g^{k_A rk(e+k_A r)^{-1}}$$

$$= g^{e'k(e+k_A r)^{-1} + k_A rk(e+k_A r)^{-1}}$$

$$= g^{k(e'+k_A r)(e+k_A r)^{-1}}$$

Caso $e = h(m)$ seja igual a $e' = h(m')$ então a expressão acima torna-se apenas

$$v = g^k.$$

Como $r = g^k$, então $v = r$. A assinatura está correta e pode-se afirmar que $m' = m$ e que somente a parte que conhece k_A poderia ter enviado m.

C.4 Esquema de Assinatura Digital Baseado nas Curvas Elípticas

Supõe-se que Ana queira assinar uma mensagem para Beto usando o ECDSA (Elliptic Curve Digital Signature Algorithm), ou seja, o algoritmo DSA sobre curvas elípticas.

C.4.1 O Algoritmo ECDSA

1. Ana e Beto escolhem publicamente um corpo finito F_q, uma curva elíptica E definida sobre F_q e um ponto fixo Z na curva elíptica $E(F_q)$.

2. Ana gera randomicamente um inteiro k_A, o qual manterá em segredo pois este será sua chave privada.

3. Ana calcula e publica $P_A = k_A F \in E(F_q)$.

4. Para Ana assinar uma mensagem para Beto, ela gera randomicamente um número escalar $k \in F_q$ e calcula C, r e s como mostram as Eqs. C.7, C.8 e C.10.

$$C = (C_x, C_y) = kZ \tag{C.7}$$

$$r = C_x \tag{C.8}$$

$$e = hash(m) \tag{C.9}$$

$$s = k^{-1}(e + k_A r) \tag{C.10}$$

5. Anexando à mensagem m, Ana envia o par (r, s) para Beto.

6. Para Beto verificar a mensagem enviada por Ana, ele calcula v pela Eq. C.15 e compara com r. Se $v_x = r$, então a assinatura está correta.

$$h = s^{-1} \tag{C.11}$$

$$e' = h(m') \tag{C.12}$$

$$h_1 = e'h \tag{C.13}$$

$$h_2 = rh \tag{C.14}$$

$$V = (v_x, v_y) = h_1 Z + h_2 P_A \tag{C.15}$$

C.4.2 A Prova do Teste

A equação $h = s^{-1}$ pode ser reescrita da seguinte forma:

$$h = k(e + k_A r)^{-1}$$

Substituindo h nas equações C.13 e C.14 e expandindo a Eq. C.15 obtém-se os resultados abaixo.

$$h_1 = e'[k(e + k_A r)^{-1}]$$

$$h_2 = r[k(e + k_A r)^{-1}]$$

$$V = e'[k(e + k_A r)^{-1}]Z + r[k(e + k_A r)^{-1}]P_A$$

Como $P_A = k_A Z$, então

$$V = e'[k(e + k_A r)^{-1}]Z + r[k(e + k_A r)^{-1}]k_A Z$$

Esta equação pode ser reescrita na forma:

$$V = k(e' + k_A r)(e + k_A r)^{-1} Z$$

Caso o hash da mensagem recebida por Beto (e'), for igual ao hash calculado pela Ana (e), os termos internos da equação acima são cancelados, resultando na equação,

$$V = kZ$$

Como $C = (C_x, C_y) = kZ$ e $r = C_x$, tem-se que, se $v_x = r$, a assinatura está correta.

Apêndice D

Curvas Elípticas Utilizadas

D.1 Sobre Corpos Finitos Primos

Os parâmetros que definem as curvas elípticas apresentadas sobre F_p consistem de um inteiro p que especifica o corpo finito F_p, dois elementos $a, b \in F_p$ que especificam a curva $E(F_p)$ definida pela eq: $E : y^2 \equiv x^3 + ax + b \pmod{p}$. É apresentado também, um ponto fixo G em $E(F_p)$.

F_{112}

---

p = DB7C2ABF 62E35E66 8076BEAD 208B

a = DB7C2ABF 62E35E66 8076BEAD 2088

b = 659EF8BA 043916EE DE891170 2B22

Ponto fixo $G(x, y)$

x = 09487239 995A5EE7 6B55F9C2 F098

y = A89CE5AF 8724C0A2 3E0E0FF7 7500

F_{128}

p = FFFFFFFD FFFFFFFF FFFFFFFF FFFFFFFF

a = FFFFFFFD FFFFFFFF FFFFFFFF FFFFFFFC

b = E87579C1 1079F43D D824993C 2CEE5ED3

Ponto fixo $G(x, y)$

x = 161FF752 8B899B2D 0C28607C A52C5B86

y = CF5AC839 5BAFEB13 C02DA292 DDED7A83

F_{160}

p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 7FFFFFFF

a = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 7FFFFFFC

b = 1C97BEFC 54BD7A8B 65ACF89F 81D4D4AD C565FA45

Ponto fixo $G(x, y)$

x = 4A96B568 8EF57328 46646989 68C38BB9 13CBFC82

y = 23A62855 3168947D 59DCC912 04235137 7AC5FB32

F_{192}

p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFF
a = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFC
b = 64210519 E59C80E7 0FA7E9AB 72243049 FEB8DEEC C146B9B1

Ponto fixo $G(x, y)$

x = 188DA80E B03090F6 7CBF20EB 43A18800 F4FF0AFD 82FF1012
y = 07192B95 FFC8DA78 631011ED 6B24CDD5 73F977A1 1E794811

F_{224}

p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFE56D
a = 00000000 00000000 00000000 00000000 00000000 00000000 00000000
b = 00000000 00000000 00000000 00000000 00000000 00000000 00000005

Ponto fixo $G(x, y)$

x = A1455B33 4DF099DF 30FC28A1 69A467E9 E47075A9 0F7E650E B6B7A45C
y = 7E089FED 7FBA3442 82CAFBD6 F7E319F7 C0B0BD59 E2CA4BDB 556D61A5

D.2 Sobre Corpos Finitos Binários

Os parâmetros que definem as curvas elípticas apresentadas sobre F_{2^m}, consistem de um inteiro m que especifica o corpo finito F_{2^m}, um polinômio irredutível $f(x)$ de grau m, especificando a representação da base polinomial de F_{2^m}, dois elementos $a, b \in F_{2^m}$ que especificam a curva $E(F_{2^m})$ definida pela eq: $E : y^2 + xy = x^3 + ax^2 + b \quad \in F_{2^m}$. É apresentado também, um ponto fixo G em $E(F_{2^m})$.

F_2^{113}

$f(x)=$ $x^{113} + x^9 + 1$

a = 006899 18DBEC7E 5A0DD6DF C0AA55C7

b = 0095E9 A9EC9B29 7BD4BF36 E059184F

Ponto fixo $G(x, y)$

x = 01A57A 6A7B26CA 5EF52FCD B8164797

y = 00B3AD C94ED1FE 674C06E6 95BABA1D

F_2^{131}

$f(x)=$ $x^{131} + x^8 + x^3 + x^2 + 1$

a = 03E5A889 19D7CAFC BF415F07 C2176573 B2

b = 04B8266A 46C55657 AC734CE3 8F018F21 92

Ponto fixo $G(x, y)$

x = 0356DCD8 F2F95031 AD652D23 951BB366 A8

y = 0648F06D 867940A5 366D9E26 5DE9EB24 0F

F_2^{163}

$f(x)=$ $x^{163} + x^7 + x^6 + x^3 + 1$

a = 07B6882C AAEFA84F 9554FF84 28BD88E2 46D2782A E2

b = 0713612D CDDCB40A AB946BDA 29CA91F7 3AF958AF D9

Ponto fixo $G(x, y)$

x = 03699796 97AB4389 77895667 89567F78 7A7876A6 54

y = 00435EDB 42EFAFB2 989D51FE FCE3C809 88F41FF8 83

F_2^{193}

$f(x)=$ $x^{193} + x^{15} + 1$

a = 0017858F EB7A9897 5169E171 F77B4087 DE098AC8 A911DF7B 01

b = 00FDFB49 BFE6C3A8 9FACADAA 7A1E5BBC 7CC1C2E5 D8314788 14

Ponto fixo $G(x, y)$

x = 01F481BC 5F0FF84A 74AD6CDF 6FDEF4BF 61796253 72D8C0C5 E1

y = 0025E399 F2903712 CCF3EA9E 3A1AD17F BOB3201B 6AF7CE1B 05

F_2^{233}

$f(x)=$ $x^{233} + x^{74} + 1$

a = 0000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

b = 0000 00000000 00000000 00000000 00000000 00000000 00000000 00000001

Ponto fixo $G(x, y)$

x = 017232BA 853A7E73 1AF129F2 2FF41495 63A419C2 6BF50A4C 9D6EEFAD 6126

y = 01DB537D ECE819B7 F70F555A 67C427A8 CD9BF18A EB9B56E0 C11056FA E6A3

Apêndice E

Publicações derivadas deste trabalho

- ***Uma proposta para avaliação de criptossistemas implementados em software baseados em curvas elípticas*, IX Simpósio de Computação Tolerante a Falhas, Workshop em Segurança de Sistemas Computacionais - Wseg'2001, Florianópolis-SC, 5 e 6 de Março de 2001**

  Resumo : recentemente a criptografia baseada em curvas elípticas (ECC) tem recebido considerável aceitação comercial, evidenciada pela sua inclusão nos padrões ANSI, IEEE, ISO e NIST, sua especificação para uso nas camadas de segurança de protocolos como ATM e WAP, bem como sua implementação em serviços como SET e IPSec. Criptossistemas baseados em curvas elípticas apresentam como principal vantagem o uso de chaves menores que aquelas empregadas em outros sistemas, como RSA por exemplo, mantendo o mesmo nível de segurança. Isto os torna interessantes para serem implementados em ambientes onde existe restrição de recursos (tempo de processamento, espaço de memória, largura de banda), como por exemplo em PDAs, telefones celulares, pagers e smart-cards. No entanto, são muitas as alternativas de implementação de ECC, desde a escolha do corpo finito aos algoritmos de aritmética modular e elíptica. Neste artigo fazemos uma análise comparativa de ECCs implementados em várias plataformas de hardware e classificamos as mesmas usando uma métrica comum, o que permite avaliar a eficiência de cada implementação independentemente de alguns fatores como frequência de

clock ou nível de segurança utilizados.

- ***Implementação em processador de sinais digitais de criptossistemas baseados em curvas elípticas*, III Simpósio de Segurança em Informática - SSI'2001, Instituto Tecnológico de Aeronáutica, São José dos Campos-SP, 24 a 26 de outubro de 2001**

  Resumo : este trabalho discute as questões envolvidas na implementação de sistemas de criptografia baseados em curvas elípticas sobre ambientes computacionais restritos, em particular sobre processadores de sinais digitais. A partir de implementações e testes de desempenho com corpos finitos primos e corpos binários, conclui-se que os corpos finitos primos podem oferecer um melhor desempenho neste tipo de ambiente, contrariando o que se acreditava anteriormente.

Referências Bibliográficas

[1] M. Abdalla, M. Bellare, e P. Rogaway. "DHAES : An Encryption Scheme on the Diffie-Hellman Problem" , 1999. http://www-cse.ucsd.edu/users/mihir/.

[2] American National Standards Institute. "Public Key Cryptography for Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", 1999.

[3] Bogdan Antonescu. "Elliptic Curve Cryptosystems on Embedded Microprocessors". Tese de mestrado, Worcester Polytechnic Institute, maio 1999.

[4] M. Aydos, B. Sunar, e Ç. K. Koç. "An High-Speed ECC-Based Wireless Authentication Protocol on an ARM microprocessor". 2nd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications Symposium on Information Theory - Dallas, Texas, outubro 1998.

[5] M. Aydos, T. Yanic, e Ç. K. Koç. "An Elliptic Curve Cryptography Based Authentication and Key Agreement Protocol for Wireless Communication". 16th Computer Security Application Conference- CSAC'00, 2000.

[6] P. Barret. "Implementing the Rivest, Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor". Springer-Verlag, LNCS 263, pp. 311-323, CRYPTO'86, 1986.

[7] Ian Blake, Gadiel Seroussi, e Nigel Smart. "Elliptic Curves in Cryptography". Cambridge University Press, 1999.

[8] Michael Brown, Darrel Hankerson, Julio L. Hernandez, e Alfred Menezes. "Software Implementation of Elliptic Curve Cryptography over Prime Fields". Technical report CORR2000-56, Department of Combinatorics and Optimization, University of Waterloo, 2000. http://www.cacr.math.uwaterloo.ca/.

[9] H. Cohen, A. Miyaji, e T. Ono. "Efficient Elliptic Curve Exponentiation Using Mixed Coordinates". Springer-Verlag, LNCS 1514, pp. 51-65, ASIACRYPT'98, 1998.

[10] Peter de Rooij. "Efficient Exponentiation Using Precomputation and Vector Addition Chains". Springer-Verlag, LNCS, pp.389-399, EUROCRYPT'98, 1998.

[11] Erik De Win, Bart Preneel, e Michael Wiener. "On the Performance of Signature Schemes Based on Elliptic Curves". Algorithmic Number Theory: Third International Symposium, pp. 252–266, 1998.

[12] Jean-Francois Dhem. "Design of an Efficient Public-Key Cryptographic Library for RISC-Based Smart Cards". Tese de Doutorado, Université Catholique de Louvain - Belgica, maio 1998.

[13] Whitfield Diffie e Martin E. Hellman. "Multiuser Cryptology Techniques". AFIPS National Computer Conference, pp. 109–112, novembro 1976.

[14] Whitfield Diffie e Martin E. Hellman. "New Directions in Cryptography". IEEE Transactions on Information Theory, IT-22(6):664–654, novembro 1976.

[15] Stephen R. Dussé e Burton S. Kaliski Jr. "A Cryptographic Library for the Motorola DSP56000". Springer-Verlag, "LNCS 473, pp. 230-244", EUROCRYPT'90, 1990.

[16] Taher ElGamal. "A Public-Key Cryptosystem and Signature Scheme Based on Discrete Logarithms". IEEE Transactions on Information Theory,31 "pp. 469-472, 1985.

[17] Library for Arbitrary Precision Arithmetic. www.gnu.org/software/gmp/gmp.html, ftp://ftp.gnu.org/gnu/gmp/.

[18] Standards for Efficient cryptography, 2000. www.secg.org.

[19] Daniel Gajski, Frank Vahid, Sanjiv Narayan, e Jie Gong. "Specification and Design of Embedded Systems". Prentice Hall, Englewood Cliffs, New Jersey 07632, 1994.

[20] Darrel Hankerson, Julio L. Hernandez, e Alfred Menezes. "Software Implementation of Elliptic Curve Cryptography over Binary Fields". Spring-Verlag, CHES 2000. julho 2000.

[21] Toshio Hasegawa, Junko Nakajima, e Mitsuru Matsui. "A Practical Implemetation of Elliptic Curve Cryptosystems over Prime Fields on a 16-bit Microcomputer". PKC'98, pp. 182–194, 1998.

[22] Institute of Electrical and Electronics Engineers, Inc. P1363 - Standard Specification for public Key Cryptography, 2000.

[23] K. Itoh, M. Takenaka, N. Torh, S. Temma, e Y. Kurihara. "Fast Implementation of Public-Key Cryptography on a DSP TMS320C6201". CHES'99, pp. 61–72, 1999.

[24] Donald Ervin Knuth. "The Art of Computer Programming", volume 2. Addison-Wesley Series in Computer Science and Information Processing, segunda edição, 1981.

[25] Neal Koblitz. "Elliptic Curve Cryptosystems". Mathematics of Computation, 48:203–209, 1987.

[26] Neal Koblitz. "Curves With Good Cryptographic Properties". CRYPTO'91, pp. 279–287, 1992.

[27] Neal Koblitz. "A Course in Number Theory and Cryptography". Springer, segunda edição, 1994.

[28] Serge Lang. "Estruturas Algébricas". Ao Livro Técnico, Rio de Janeiro, 1972.

[29] Free Large Integer Package. University of Oxford. ftp://ftp.ox.ac.uk/pub/math/freelip/freelip_1.0.tar.gz.

[30] C. H. Lim e P. J. Lee. "More Flexible Exponentiation with Precomputation". CRYPTO'94, pp. 95–107, 1994.

[31] Julio César López Hernández. "Implementação Eficiente em Software de Criptossistema de Curvas Elípticas". Tese de Doutorado, UNICAMP, abril 2000.

[32] Julio César López Hernández e Ricardo Dahab. "Improved Algorithms for Elliptic Curve Arithmetic in GF(2^n)". SAC'98, pp. 201–212, 1998.

[33] Julio César López Hernández e Ricardo Dahab. "High-Speed Software Multiplication in F_{2^m}". http://www.dcc.unicamp.br/ic-main/publications-e.html, 2000.

[34] A. Menezes, T. Okamoto, e S. Vanstone. "Reducing Elliptic curve Logarithms to Logarithms in a Finite Field". IEEE Transation on Information Theory, 39:1639–1646, 1993.

[35] Alfred J. Menezes. "Elliptic Curve Public Key Cryptosystems". Kluwer Academic Publishers, 1993.

[36] Alfred J. Menezes, Paul C. van Oorschot, e Scott A. Vanstone. "Handbook of Applied Cryptography". CRC Press, 1997.

[37] Victor Miller. "Uses of Elliptic Curves in Cryptography". CRYPTO'85, 1986.

[38] Peter L. Montgomery. "Modular Multiplication without Trial Division". Mathematics of Computation, 44(6):519–521, abril 1985.

[39] K. Nyberg e R. Rueppel. "A New Signature Scheme Based on the DSA Giving Message Recovery". First ACM Conference on Computer and Communication Security, pp. 58–61, 1993.

[40] National Institute of Stardards and Technology. FIPS publication 186-2, Feb 2000. http://csrc.nist.gov/fips.

[41] OpenSSL. http://www.openssl.org.

[42] The International PGP Home Page. http://www.pgpi.org.

[43] Ron Rivest, Adi Shamir, e Len Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM, 21:120–126, 1978.

[44] Michael Rosing. "Implementing Elliptic Curve Cryptography". Maning Publications Co., 1999.

[45] Bruce Schneier. "Applied Cryptography: Protocols, Algorithms and Code in C". John Wiley & Sons, Inc., segunda edição, 1996.

[46] Claus P. Schnorr. "Efficient Signature Generation for Smart Cards". CRYPTO'89, pp. 239–252, 1990.

[47] R. Schroeppel, H. Orman, S. O'Malley, e O. Spatscheck. "Fast Key Exchange with Elliptic Curve Systems". CRYPTO'95, 963:43–56, 1995.

[48] Nigel Smart. "The Discrete Logarithm Problem on Elliptic Curves of Trace One". Journal of Cryptology, 12:193–196, 1999.

[49] Jerome A. Solinas. "Improved Algorithms for Arithmetic on Anomalous Binary Curves". CRYPTO'97, 1997.

[50] William Stallings. "Cryptography and Network Security". Prentice Hall, segunda edição, 1998.

[51] Texas Instruments, TMS320C54x DSP CPU and Peripherals, 1997.

[52] André Weimerskirch, Christof Paar, e Sheueling Chang Shantz. "Elliptic Curve Cryptography on a Palm OS Device". Australasian Conference on Information Security and Privacy, julho 2001.

[53] Andrew Wiles. "Modular Elliptic Curves and Fermat's Last Theorem". Ann. Math., 142:443–551, 1995.

[54] Thomas J. Wollinger, Min Wang, Jorge Guajardo, e Christof Paar. "How Well Are High-End DSPs Suited for the AES Algorithms ?". The Third Advance Encryption Standard (AES3) Candidate Conference - New York - USA, abril 2000.

[55] Adam D. Woodbury, Daniel V. Bailey, e Christof Paar. "Elliptic Curve Cryptography on Smart Cards without Coprocessors". The Fourth Smart Card Research and Advanced Applications - CARDIS 2000, setembro 2000.

[56] John Worley, Bill Worley, Tom Christian, e Christopher Worley. "AES Finalists on PA-RISC and IA-64 Implementations and Performance". Hewlett Packard Labs - Fort Collins, CO, janeiro 1999.