Uma proposta para avaliação de criptossistemas implementados em software baseados em curvas elípticas

Arnaldo J. de Almeida Jr. Marco Aurélio A. Henriques

[ajaj|marco]@dca.fee.unicamp.br

Faculdade de Engenharia Elétrica e de Computação
Departamento de Engenharia de Computação e Automação Industrial
UNICAMP
C.P. 6101 - Campinas - SP - 13083-970

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.*

Palavras-chave : criptografia assimétrica, curvas elípticas, corpos finitos

1 Introdução

Em 1985, Victor Miller [15] e Neal Koblitz [12] propuseram independependente o uso das curvas elípticas para aplicações em criptografia de chave pública. A razão principal para o interesse por ECC (Elliptic Curve Cryptography) é o fato de não existir nenhum algoritmo sub-exponencial conhecido até o momento para se resolver o problema dos logaritmos discretos sobre curvas elípticas [13]. Isto significa que as chaves usadas em ECC podem ser consideravelmente menores comparadas às de outros criptossistemas como RSA e DSA, mantendo-se o mesmo nível de segurança.

Recentemente, o grupo de estudos de padronização IEEE P1363 concluiu a primeira versão do Standard Specification for Public Key Cryptography [9], sendo que os métodos matemáticos padronizados para a implementação de esquemas de chave pública foram: fatoração de números inteiros, logaritmos discretos e logaritmos discretos sobre curvas elípticas. Outros orgãos de padronização também já incluiram as curvas elípticas em seus padrões como ANSI [1], ISO [10] e NIST [17].

Nos últimos anos, muitos artigos foram escritos a respeito dos vários aspectos de implementação de ECC. No entanto a maioria destes artigos não considera todos os aspectos envolvidos numa implementação eficiente. São muitas as escolhas que devem ser feitas para se desenvolver uma ECC, desde o tipo de curva elíptica, sua representação, bem como os algoritmos a serem aplicados. Observa-se todavia uma certa dificuldade quando se tem uma determinada plataforma de processamento e deseja-se decidir quais são as melhoras escolhas para esta determinada plataforma. Por exemplo, as escolhas ótimas em uma plataforma PC podem ser muito diferentes das escolhas ótimas para uma plataforma restrita como os smart-cards [7].

Nesse sentido, analisamos o perfil de alguns sistemas ECCs implementados em diferentes plataformas de hardware e recentemente publicados e apresentamos uma classificação, que pode servir como referência para futuros trabalhos, pois mostra quais sistemas ficaram mais adaptados nos seus ambientes de processamento.

O restante deste artigo está organizado conforme a seguir. A Seção 2 apresenta as alternativas típicas de implementação em ECC; a Seção 3 apresenta o perfil de alguns trabalhos de implementação de ECC; a Seção 4 apresenta uma análise comparativa destes trabalhos e a Seção 5 apresenta as conclusões.

2 Alternativas típicas em ECC

Corpo finito: existem três escolhas:

- *Corpo Primo* é o corpo $GF(p)$ que contém um número primo p de elementos. Os elementos deste corpo são inteiros módulo p e as operações são implementadas em termos de aritmética de inteiros módulo p.
- *Corpo Binário* é o corpo $GF(2^m)$ que contém 2^m elementos para algum 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 de bits.
- *Corpo de Extensão Ótima (Optimal Extension Field - OEF)* é o corpo $GF(p^m)$ onde p é representado por um número primo de Mersenne $(2^n \pm c)$, para n, c inteiros positivos e arbitrários. Corpos OEF foram propostos para uso em criptografia recentemente e não existe ainda ataques contra ECC sobre OEFs. Entretanto, OEFs não estão incluídos em nenhuma padronização oficial.

Representação da base: para descrever a aritmética de corpos 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 independente de vetores que geram este espaço vetorial. Para corpos finitos binários $GF(2^m)$, dependendo da base aplicada, existem técnicas eficientes para execução das operações aritméticas. Há dois tipos comuns de base:

- *Base Polinomial* é especificada por um polinômio irredutível módulo 2, chamado de corpo polinomial. A sequência de bits $(a_{m-1} \cdots a_2 a_1 a_0)$ representa o polinômio $a_{m-1}t^{m-1} + \cdots + a_2 t^2 + a_1 t + a_0$ sobre $GF(2)$. A aritmética é implementada como aritmética módulo $p(t)$, onde $p(t)$ é o corpo polinomial.
- *Base Normal* é a base da forma: $\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}$ onde $\beta \in GF(2^m)$.

Representação das coordenadas: a equação das curvas elípticas é um caso especial da equação de Weierstrass [14] e os pontos pertencentes à esta curva podem ser representados por vários sistemas de coordenadas, como por exemplo coordenadas afim, projetivas homogêneas ou jacobianas entre outras. Em casos onde o cálculo do inverso multiplicativo for significativamente mais caro que o cálculo da multiplicação, pode ser mais eficiente a implementação de um sistema de coordenadas projetivas homogênea ou jacobianas. Por questões de espaço, neste artigo somente serão apresentados os detalhes para o sistema de coordenadas afim. Maiores detalhes sobre os outros sistemas de coordenadas podem ser obtidos em [9].

- *Sistema de coordenadas afim:* Para $GF(p)$, $p > 3$ a equação da curva elíptica é apresentada sob a forma $E : y^2 = x^3 + ax + b$ com $a, b \in GF(p)$ e com a condição de $4a^3 + 27b^2 \neq 0$. Assumindo que $P_1 = (x_1, y_1)$ e $P_2 = (x_2, y_2)$ sejam pontos pertencentes a $E(GF(p))$, a soma $P_3 = (x_3, y_3) = P_1 + P_2$ pode ser calculada conforme a equação (1).

$$x_3 = \lambda^2 - x_1 - x_2 \quad , \quad y_3 = \lambda(x_1 - x_3) - y_1 \ onde \qquad \lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \ se \ P_1 \neq P_2, \\ \frac{3{x_1}^2 + a}{2y_1} \ se \ P_1 = P_2. \end{cases} \tag{1}$$

Para $GF(2^m)$, a equação da curva elíptica é apresentada sob a forma $E : y^2 + xy = x^3 + ax^2 + b$ com $a, b \in GF(2^m)$ e com a condição de $b \neq 0$. Assumindo que $P_1 = (x_1, y_1)$ e $P_2 = (x_2, y_2)$ sejam pontos pertencentes a $E(GF(2^m))$, a soma $P_3 = (x_3, y_3) = P_1 + P_2$ pode ser calculada conforme a equação (2).

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \ , \ y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \ onde \qquad \lambda = \begin{cases} \frac{y_2 + y_1}{x_2 + x_1} \ se \ P_1 \neq P_2, \\ x_1 + \frac{y_1}{x_1} \ se \ P_1 = P_2. \end{cases} \tag{2}$$

Algoritmos da aritmética modular: são os algoritmos que aplicam as operações modulares tais como adição, subtração, multiplicação e sua operação inversa. Normalmente a operação do inverso multiplicativo é a mais custosa.

Algoritmos da aritmética elíptica: estes algoritmos consistem de adição e duplicação de pontos bem como de multiplicação escalar de pontos, ou seja, dado um ponto P e um escalar k, o cálculo de kP. A eficiência desta operação pode definir o desempenho de um sistema com ECC.

Tipo de curva elíptica :

- *Curvas pseudo-aleatórias* são curvas geradas aleatoriamente. Deve ser verificado se estas definem uma ordem $\#E(GF(q))$ (número de pontos) apropriada para aplicações criptográficas. Também deve ser verificado se a curva gerada é segura contra os principais ataques conhecidos (maiores detalhes no capítulo 6 de [3]).
- *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 (também chamadas de curvas binárias anômalas) sobre $GF(2^m)$.

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 [5], ElGamal [6], dentre outros. Os mais comuns são aqueles padronizados pela IEEE P1363 [9]:

- 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 (novo padrão americano, substituto do DES).
- 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 o algoritmo de hash SHA-1 ou MD5.

3 Perfil de implementação de trabalhos anteriores

Várias publicações apresentam algoritmos e métodos para computação eficiente dos 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 nesta seção são implementações reais sobre alguma plataforma de hardware e que, consequentemente, tiveram um desempenho determinado pelos parâmetros de ECC, bem como pelas técnicas de implementação escolhidas pelos seus autores. Estes trabalhos foram selecionados por apresentarem uma boa variedade de altenativas de implementação e de plataformas (microcontroladores, processadores digitais de sinais, processadores RISC e arquitetura Intel x86), bem como por terem implementado o ECDSA, apresentando seus respectivos tempos para geração e verificação de assinatura, dados estes que serão utilizados na análise comparativa.

3.1 De Win, Mister, Preneel e Wiener - 1998

Este artigo [4] é um dos mais referenciados pelos artigos mais recentes. Foi um dos primeiros a tratar de forma completa a ECC, da aritmética em corpo finito e corpo elíptico ao criptossistema, para ambos os corpos $GF(p)$ e $GF(2^m)$. A plataforma de processamento é um PC Pentium Pro200. Na implementação de De Win, Mister, Preneel e Wiener sobre a plataforma citata, foi melhor o desempenho do criptossistema sobre $GF(p)$. A implementação de $GF(p)$ foi feita em C/C++ e Assembly, e utilizou-se as coordenadas projetivas. A de $GF(2^m)$ foi feita em C++ onde utilizou-se polinômio redutor (trinômio) melhorado e obteve-se resultados mais eficientes na operação de redução (módulo $p(x)$). Também aplicou-se algoritmos para otimizar o inverso multiplicativo e a multiplicação escalar de pontos.

3.2 Hasegawa, Nakajima e Matsui - 1998

O artigo de Hasegawa, Nakajima e Matsui [8], um grupo da Mitsubishi, trata de uma implementação completa de uma ECC sobre $GF(p)$, 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. Foram feitas melhorias ao método de somar e duplicar pontos elípticos, representados por coordenadas projetivas, comparado ao método apresentado na P1363 [9], diminuindo-se o número de variáveis temporárias necessárias ao algoritmo. Para multiplicação escalar de pontos foram implementados os algoritmos para ponto aleatório e ponto fixo, sendo que com o segundo obteve-se melhores resultados. É importante salientar que o criptossistema foi escrito totalmente em Assembly.

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

O artigo de Itoh, Takenaka, Torh, Temma e Kurihara [11], um grupo da Fujitsu, trata de uma implementação completa de um ECC sobre $GF(p)$, em um processador digital de sinais da Texas Instruments TMS320C6201, 200MHz, 16 bits, um DSP topo de linha (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 [16] onde foi aproveitada a arquitetura de pipeline do DSP. Para aritmética em corpo elíptico, a contribuição deste trabalho foi em diminuir o número de instruções de adição e multiplicação, representados por coordenadas projetivas, comparado ao método apresentado na P1363 [9]. 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 são desprezíveis quando comparadas às instruções de multiplicação. As rotinas básicas foram escritas em Assembly e as demais em C.

3.4 Julio Lopés - 2000

Julio Lopés apresenta em sua tese de doutorado [13] contribuições (métodos eficientes) para a aritmética no corpo finito $GF(2^m)$ (multiplicação rápida) , bem como para o grupo elíptico (métodos eficientes para duplicações de pontos elípticos e um novo sistema de coordenadas projetivas). Resultados melhores 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.

3.5 Aydos, Yanik e Koç - 2000

É o trabalho mais recente (dezembro de 2000). Se trata de uma implementação completa de um sistema com ECC sobre $GF(p)$, em um processador ARM7TDMI (80MHz, RISC 32 bits) [2]. Neste trabalho foi utilizado o algoritmo de Montgomery [16] para multiplicação eficiente e foram usadas as coordenadas jacobianas para a representação de pontos, o que levou a uma diminuição do número instruções de adição e multiplicação na duplicação de pontos mas aumentou o número de instruções de quadrados.

4 Análise Comparativa

Normalmente a eficiência dos sistemas com ECC que implementam o ECDSA é avaliada pelo tempo que se levou para efetuar geração e verificação de assinatura. Na verdade esta medida (tempo) é uma medida relativa, pois depende fortemente do clock do processador e do tamanho do corpo, tornando menos visível se a escolha dos parâmetros para o ECC, bem como as técnicas de implementação, levaram realmente a um bom resultado.

Em nossa análise aplicamos uma normalização na velocidade e no tamanho do corpo. Ela se baseia em uma nova métrica que consiste no número de ciclos de clock por cada bit do corpo (chave) que o processador precisou para efetuar as operações de geração e verificação de assinatura. Os resultados estão apresentados nas tabelas 1 e 2.

Para os trabalhos que apresentaram vários resultados (várias alternativas de implementação), escolheu-se o de melhor desempenho para a análise. Quando houve várias implementações para corpos finitos diferentes, incluímos seus resultados (para os diferentes corpos) na análise.

Nota-se que nem sempre as implementações de maior frequência de clock foram as mais eficientes. Destaca-se o oitavo classificado (nas duas tabelas) que, mesmo trabalhando a 10 MHz, teve uma relação

Classif.	Seção	Plataforma	Clock (MHz)	Implementação	Chave	Geraç. (ms)	Ciclos de clock	Ciclos de clock / bits da chave
1	3.3	DSP	200	GF(p)/mult. Montg./coord. proj.	160	1,09	218000	1362
2	3.3	DSP	200	GF(p)/mult. Montg./coord. proj.	192	1,5	300000	1562
3	3.3	DSP	200	GF(p)/mult. Montg./coord. proj.	239	2,66	532000	2225
4	3.4	PentiumII	400	GF(2^m)/tec. própria/cur. Koblitz	163	2,11	844000	5177
5	3.1	PPro200	200	GF(p)/coord. proj.	191	6,3	1260000	6596
6	3.4	PentiumII	400	GF(2^m)/tec. própria/cur. Koblitz	233	4,03	1612000	6918
7	3.4	PentiumII	400	GF(2^m)/tec. própria/cur. Koblitz	283	5,64	2256000	7971
8	3.2	M16C	10	GF(p)/ kP:ponto fixo	160	150	1500000	9375
9	3.1	PPro200	200	GF(2^m)/trinômio melhorado	191	11,3	2260000	11832
10	3.5	ARM7	80	GF(p)/coord. jacobianas	160	46,4	3712000	23200
11	3.5	ARM7	80	GF(p)/coord. jacobianas	192	71,3	5704000	29708
12	3.5	ARM7	80	GF(p)/coord. jacobianas	176	65,4	5232000	29727
13	3.5	ARM7	80	GF(p)/coord. jacobianas	208	96,2	7696000	37000
14	3.5	ARM7	80	GF(p))/coord. jacobianas	256	153,5	12280000	47968
15	3.4	RIMPager	10	GF(2^m)/tec. própria/cur. Koblitz	163	1011	10110000	62024
16	3.4	RIMPager	10	GF(2^m)/tec. própria/cur. Koblitz	233	1910	19100000	81974
17	3.4	RIMPager	10	GF(2^m)/tec. própria/cur. Koblitz	283	2760	27600000	97526
18	3.4	PalmPilot	16	GF(2^m)/tec. própria/cur. Koblitz	163	1793	28688000	176000
19	3.4	PalmPilot	16	GF(2^m)/tec. própria/cur. Koblitz	233	3080	49280000	211502
20	3.4	PalmPilot	16	GF(2^m)/tec. própria/cur. Koblitz	283	5485	87760000	310106

Tabela 1: Classificação para geração de assinatura

custo/benefício superior ao de várias outras implementações com frequências maiores. A justificativa para tal eficiência deve-se provavelmente ao uso de Assembly na implementação do sistema, bem como ao uso de um ponto P fixo, para a multiplicação escalar de pontos. As tabelas também nos mostram que o uso de recursos oferecidos pela arquitetura DSP permitiram implementações bem mais eficientes que aquelas em uma plataforma Pentium com o dobro da frequência de clock, para chaves variando de 160 a 239 bits.

5 Conclusão

Apresentamos uma análise de eficiência usando uma nova métrica comum, na tentativa de revelar quais parâmetros na implementação de ECC ficaram mais adaptados às plataformas de hardware utilizadas, independentemente do clock e do tamanho da chave. Esta análise se baseia no número de ciclos de clock que cada processador precisou para efetuar, para cada bit da chave, as operações de geração e verificação de assinatura.

Muitas arquiteturas de processamento fazem divisão interna do tempo de clock, o que significa que o ciclo de máquina pode ser diferente (maior) que o clock do processador. Além disso sistemas que são programados em Assembly tiram mais proveito da arquitetura do processador.

Acreditamos que, apesar das aproximações inerentes aos critérios da análise, ainda assim ela permite perceber as melhores combinações de software e hardware que levaram a relações custo/benefício (recursos/segurança) mais interessantes e oferece uma referência para orientar futuras implementações.

Referências

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

[2] M. Aydos, T. Yanic, and Ç. K. Koç. An elliptic curve cryptography based authentication and key agreement protocol for wireless communication. *16th Computer Security Application Conference- CSAC'00*, 2000.

[3] Ian Blake, Gadiel Seroussi, and Nigel Smart. *Elliptic Curves in Cryptography*. Cambridge university press, 1999.

Classif.	Seção	Plataforma	Clock (MHz)	Implementação	Chave	Verif. (ms)	Ciclos de clock	Ciclos de clock / bits da chave
1	3.3	DSP	200	GF(p)/mult. Montg./coord. proj.	160	3,78	756000	4725
2	3.3	DSP	200	GF(p)/mult. Montg./coord. proj.	192	5,5	1100000	5729
3	3.3	DSP	200	GF(p)/mult. Montg./coord. proj.	239	9,78	1956000	8184
4	3.4	PentiumII	400	GF(2^m)/tec. própria/cur. Koblitz	163	4,09	1636000	10036
5	3.4	PentiumII	400	GF(2^m)/tec. própria/cur. Koblitz	233	7,87	3148000	13510
6	3.4	PentiumII	400	GF(2^m)/tec. própria/cur. Koblitz	283	11,46	4584000	16197
7	3.1	PPro200	200	GF(p)/coord. proj.	191	26,0	5200000	27225
8	3.2	M16C	10	GF(p)/ kP:ponto fixo	160	630	6300000	39375
9	3.5	ARM7	80	GF(p)/coord. jacobianas	160	92,4	7392000	46200
10	3.5	ARM7	80	GF(p)/coord. jacobianas	176	131,3	10504000	59681
11	3.5	ARM7	80	GF(p)/coord. jacobianas	192	148,3	11864000	61791
12	3.1	PPro200	200	GF(2^m)/trinômio melhorado	191	60,0	12000000	62827
13	3.5	ARM7	80	GF(p)/coord. jacobianas	208	194,3	15544000	74730
14	3.5	ARM7	80	GF(p))/coord. jacobianas	256	313,4	25072000	97937
15	3.4	RIMPager	10	GF(2^m)/tec. própria/cur. Koblitz	163	1826	18260000	112024
16	3.4	RIMPager	10	GF(2^m)/tec. própria/cur. Koblitz	233	3701	37010000	158841
17	3.4	RIMPager	10	GF(2^m)/tec. própria/cur. Koblitz	283	4716	47160000	166643
18	3.4	PalmPilot	16	GF(2^m)/tec. própria/cur. Koblitz	163	3263	52208000	320294
19	3.4	PalmPilot	16	GF(2^m)/tec. própria/cur. Koblitz	233	5878	94048000	403639
20	3.4	PalmPilot	16	GF(2^m)/tec. própria/cur. Koblitz	283	9059	144944000	512169

Tabela 2: Classificação para verificação de assinatura

[4] Erik De Win, Bart Preneel, and Michael Wiener. On the performance of signature schemes based on elliptic curves. *Algorithmic Number Theory: Third International Symposium*, pages 252–266, 1998.

[5] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):664–654, November 1976.

[6] Taher ElGamal. A public-key cryptosystem and signature scheme based on discrete logarithms. *CRYPTO'84*, 1985.

[7] Darrel Hankerson, Julio L. Hernandez, and Alfred Menezes. Software implementation of elliptic curve cryptography over binary fields. In *CHES 2000*. Spring-Verlag, July 2000.

[8] Toshio Hasegawa, Junko Nakajima, and Mitsuru Matsui. A practical implematation of elliptic curve cryptosystems over prime fields on a 16-bit microcomputer. *PKC'98*, pages 182–194, 1998.

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

[10] International Standards Organization. *Information Technology - Security Techniques - cryptographic Technics Based on Elliptic Curves*, 1999. Committee Draft.

[11] K. Itoh, M. Takenaka, N. Torh, S. Temma, and Y. Kuriara. Fast implementation of public-key cryptography on a dsp tms320c6201. *CHES'99*, pages 61–72, 1999.

[12] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[13] Julio César López Hernández. *Implementação Eficiente em Software de criptossistema de Curvas Elípticas*. PhD thesis, UNICAMP, April 2000.

[14] Alfred J. Menezes. *Elliptic Curve Public Key Criptosystems*. Kluwer Academic Publishes, 1993.

[15] Victor Miller. Uses of elliptic curves in cryptography. *CRYPTO'85*, 1986.

[16] P.L. Montegomery. Modular multiplication without trial division. Mathematics of computations - 44(170):519-521, 1985.

[17] National Institute of Standards and Technology. *Digital Signature Standard - FIPS 186-2*, February 2000.