Criptografia baseada em curvas elípticas: otimização de algoritmos básicos em plataforma DSP

Mariana Coelho de Oliveira, Danilo Prates de Oliveira e Marco Aurélio Amaral Henriques

***Resumo*— Este trabalho objetiva otimizar operações aritméticas básicas da implementação de um criptossistema baseado em curvas elípticas com corpos de extensão ótima. Foram obtidas melhorias no tempo de processamento de até 54,6% e no espaço em memória de até 43,0% com a otimização de uma rotina de multiplicação que usa intensivamente operações de redução modular.**

***Palavras-Chave*— Criptografia com Curvas Elípticas, Corpos de Extensão Ótima, Processador Digital de Sinais**

***Abstract*— This work aims at the optimization of basic arithmetic operations in the implementation of an elliptic curve cryptosystem based on optimal extension fields. Improvements of up to 54.6% on processing time and up to 43.0% on space were obtained with the optimization of a multiplication routine that intensively uses modular reduction operations.**

***Keywords*— Elliptic Curve Cryptography, Optimal Extension Fields, Digital Signal Processor**

I. Introdução

Em sistemas embarcados de computação, como telefones celulares e smart-cards, é cada vez mais necessária a autenticação de usuários e a proteção das informações armazenadas e/ou trocadas por eles. Técnicas criptográficas são ferramentas que atendem estas necessidades, mas seu custo pode ser alto. A criptografia baseada em curvas elípticas atinge níveis de seguraça equivalentes aos métodos tradicionais, como RSA, porém leva vantagem no tamanho das chaves, que podem ser bem menores [1]. Esta diminuição resulta em uma menor necessidade de processamento, de memória e de largura de banda, simplificando a implementação, especialmente em plataformas restritas. As curvas elípticas são conhecidas há mais de 100 anos na matemática, mas a sua utilização em sistemas de criptografia só foi proposta, por Koblitz e Miller, em 1985. A criptografia baseada em curvas elípticas tem se popularizado nos últimos anos e foi recentemente escolhida pelo governo norte-americano como esquema padrão para a proteção de documentos sensíveis.

As curvas elípticas ulilizadas em criptografia são definidas sobre corpos finitos, que são tipos de grupos algébricos, ou seja, conjuntos de elementos que podem ser combinados usando-se algumas operações, como adição, subtração ou multiplicação, e que satisfazem certas condições. Corpos finitos são normalmente identificados pela notação $GF(p^m)$, onde p é um primo e m é um inteiro positivo. A implementação de um criptossistema baseado em curvas elípticas (*Elliptic Curve Cryptosystem – ECC*) pode ser realizada em camadas , como ilustrado a seguir.

Mariana C. de Oliveira, Danilo P. de Oliveira e Marco A. A. Henriques, FEEC/UNICAMP, CP 6101, Campinas, SP, CEP 13083-970, E-mails: mariana@fee.unicamp.br, dprates@dca.fee.unicamp.br, marco@dca.fee.unicamp.br. Trabalho parcialmente financiado com recursos do PIBIC/CNPq e CAPES.

Interface com o Usuário
Protocolo Criptográfico
Aritmética na Curva Elíptica
Aritmética no Corpo Finito
Redução Modular (mod p)

Cada uma dessas camadas faz chamadas a rotinas da camada imediatamente inferior. Por isso otimizações nas camadas mais baixas podem causar um grande impacto no desempenho final do criptossistema. A camada mais baixa contém a rotina que implementa a redução modular (resto da divisão inteira ou mod p). Esta rotina é a mais recorrente no criptossistema e, por isso, os esforços deste trabalho se concentraram na mesma.

II. Corpos de Extensão Ótima

A implementação de um ECC requer a escolha de diversos parâmetros tais como a curva a ser utilizada, o sistema de coordenadas, a base para representação do corpo finito e o próprio corpo finito [1]. A escolha do corpo finito é muito importante pois define os algoritmos que realizam sua aritmética. Os corpos de extensão ótima (*Optimal Extension Fields – OEF*) são corpos finitos do tipo $GF(p^m)$, $p > 2$ que têm a vantagem de permitir a escolha de um p que mais se adeqüe às características específicas da plataforma de hardware: o valor de p pode ser selecionado para caber em uma única palavra do processador, simplificando significativamente as operações matemáticas [2]. Os elementos de um corpo de extensão ótima podem ser representados por polinômios de grau até $m-1$ com coeficientes pertencentes ao conjunto $\{0, 1, ..., p-1\}$. Qualquer operação aritmética efetuada com dois elementos do OEF deve resultar também num elemento do OEF. Para isso, utiliza-se um polinômio primo $f(x) = x^m - \omega$ (polinômio que não pode ser fatorado em dois polinômios de grau menor) e faz-se redução módulo $f(x)$ em todas as operações realizadas. Este trabalho utiliza o mesmo OEF $GF((2^{13}-1)^{13})$ (que proporciona segurança de 169 bits) e o mesmo polinômio irredutível $f(x) = x^{13} - 2$ adotados em um trabalho anterior usado como referência [3]. Os programas foram implementados sobre o DSP de 16 bits TMS320VC5402 da Texas Instruments, que tem clock de 40MHz, possui 16K palavras de memória e é indicado para dispositivos portáteis. O Code Composer Studio 2.21, também da Texas, foi utilizado como ambiente de desenvolvimento em linguagem C.

III. Otimizações e Resultados

A rotina de redução modular convencional usada no trabalho de referência [3] possui uma versão simplificada proposta na

literatura [1]. Esta redução modular simplificada (*Simplified Modular Reduction – SMR*) obtém $r = y \bmod p$ na forma

1. $r = y - ((y >> 13) << 13) + (y >> 13)$,
2. se $r \geq p$ faça $r \leftarrow r - p$,

onde ($>> 13$) e ($<< 13$) indicam deslocamentos de 13 bits à direita e à esquerda, respectivamente. SMR faz uma divisão do valor y em duas partes: uma com os $m = 13$ bits menos significativos e outra com os bits mais significativos restantes. Estas duas partes são somadas e o valor resultante representa o valor de y reduzido mod p, podendo ainda ser necessário subtrair o valor de p. No entanto, SMR apresenta duas restrições: ela só pode ser aplicada a uma subclasse dos corpos de extensão ótima chamada OEF tipo I [2] e ela só funciona se o valor a ser reduzido tiver um tamanho máximo de $(p-1)^2$, onde $p = 2^{13} - 1$ é o número primo que caracteriza o OEF utilizado. A primeira restrição é satisfeita, pois $GF((2^{13}-1)^{13})$ é um OEF do tipo I, mas a segunda nem sempre. A rotina que implementa a multiplicação de elementos do OEF não atende a segunda restrição ao chamar a rotina de redução modular. O algoritmo de multiplicação utilizado, tanto no trabalho de referência [3] quanto no atual, é o Column Major [4], que só se aplica a OEFs e é considerado o mais eficiente para este tipo de corpo finito. Após uma avaliação mais detalhada deste algoritmo, constatou-se que, ao chamar a redução modular, a entrada fornecida por ele pode atingir o valor de $(\omega(m-1)+1)(p-1)^2$, o que representa $25(p-1)^2$ no OEF utilizado (25 vezes maior que o limite). Portanto a entrada pode ter até $\lceil log_2 25 \rceil = 5$ bits a mais que os 26 esperados de uma multiplicação de dois elementos do conjunto $\{0, 1, ..., p-1\}$, resultando em um valor de até 31 bits. Aplicando-se SMR ao caso em questão, a parte com os bits mais significativos poderá ter até 31-13=18 bits e o resultado de sua soma à parte menos significativa poderá ter até 19 bits, mesmo após a subtração de p. Este resultado de até 19 bits pode ser significativamente superior ao valor de p (13 bits), o que mostra que SMR não traria o resultado esperado na maioria dos casos. Das $2^6 = 64$ combinações possíveis dos seis bits excedentes, apenas uma (os seis bits zerados) implicaria em um resultado correto da redução, isto é, com até 13 bits significativos. Isto significa que há apenas 1/64=1,56% de chance de SMR fornecer um resultado correto. Portanto, para tirar proveito de SMR, foi necessário realizar uma adaptação no mesmo. Observamos que, se aplicarmos novamente SMR a este resultado de até 19 bits, estaremos somando uma parte de 13 bits com outra de até 19-13=6 bits, podendo resultar em valores de até 14 bits. Este resultado ainda pode ser maior que p, mas uma rápida análise dos valores máximos que podem resultar desta soma mostra que temos até 63 valores superiores a p (de $p+1$ até $p+63$), que representam apenas $63/(p+63)$=0,76% de todos os valores possíveis. Logo, após uma segunda aplicação de SMR, há mais de 99% de chance de já termos o valor de y reduzido mod p corretamente. Para estes 0,76% de casos em que o resultado ainda pode estar acima de p, é suficiente subtrair p uma vez ou aplicar SMR uma terceira vez. A segunda opção custa mais caro que a primeira e, por isso, não foi adotada. O algoritmo proposto para substituir a redução modular convencional chamada pela multiplicação é descrito a seguir. Note que a parcela $y - ((y >> 13) << 13)$ pode ser calculada através de uma operação lógica AND bit a bit com $(1FFF)_{16}$, a qual é mais econômica no DSP utilizado.

Entradas: $p = 2^{13} - 1$, $y \leq 25(p-1)^2$
Saída: $r = y$ **mod** p

1. $r \leftarrow (y \ \& \ \ (00001FFF)_{16}) + (y >> 13)$
2. $r \leftarrow (r \ \& \ \ (00001FFF)_{16}) + (r >> 13)$
3. Se $r \geq p$ **faça** $r \leftarrow r - p$

Os resultados da otimização proposta foram bastante satisfatórios. A rotina de redução modular convencional da referência [3] gasta 109 ciclos do processador e ocupa 97 posições de memória. Já a rotina proposta apresentou um tempo de 18 ciclos e ocupou apenas 19 palavaras da memória, uma redução de 83,5% no tempo e 80,4% no espaço. Uma das rotinas do ECC que mais vezes chamam a rotina de redução modular é a de multiplicação de elementos de OEF. Portanto, para avaliar o impacto da otimização proposta, medimos o tempo de execução e o espaço da rotina de multiplicação quando a rotina de redução da referência é substituída pela rotina proposta. Na implementação da multiplicação são possíveis duas abordagens distintas: *rolled* (com laços) e *unrolled* (sem laços). A Tabela I compara os resultados de referência com os obtidos na nova implementação. Na abordagem *unrolled* houve a maior diminuição (-54,6%) no tempo de processamento com o uso do algoritmo proposto. No entanto, o espaço ocupado em memória diminuiu apenas 7,9%. Já na abordagem *rolled* houve uma melhora no tempo menos acentuada (-35,0%), mas o novo algoritmo conseguiu economizar mais espaço em memória (-37,6%) que na implementação *unrolled* da multiplicação.

TABELA I
COMPARAÇÃO ENTRE AS ROTINAS DE MULTIPLICAÇÃO.

Implementação	Ciclos	Tempo (μs)	Memória (palavras)
Referência (*rolled*)	3350	83,8	165
Proposta (*rolled*)	2176	54,4	94
$\Delta(\%)$	-35,0	-35,0	-43,0
Referência (*unrolled*)	3284	82,1	988
Proposta (*unrolled*)	1491	37,3	910
$\Delta(\%)$	-54,6	-54,6	-7,9

IV. CONCLUSÕES

Foi mostrado como é possível diminuir significativamente o tempo e espaço gastos por rotinas básicas de um criptossistema baseado em curvas elípticas que utilizam o corpo de extensão ótima $GF((2^{13}-1)^{13})$ em plataforma DSP de 16 bits.

REFERÊNCIAS

[1] Alfred Menezes, Darrel Hankerson and Scott Vanstone. *Guide to Elliptic Curve Cryptography.* Springer Professional Computing, 2004.
[2] D. V. Bailey and C. Paar. *Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms. CRYPTO'98*, 1998.
[3] David Reis Jr., Arnaldo J. de Almeida Jr. and Marco A. A. Henriques. *Evaluation of Elliptic Curves Operations over Optimal Extension Field on TMS320VC5402 Digital Signal Processor.* CIBSI2003 - II Congreso Iberoamericano de Seguridad Informática, México, Out. 2003.
[4] Jae Wook Chung, Sang Gyoo Sim, and Pil Joong Lee. *Fast implementation of elliptic curve defined over $GF(p^m)$ on CalmRISC with MAC2424 coprocessor. CHES '00*, 2000.