Implementação em Software de Criptografia Assimétrica para Redes de Sensores com o Microcontrolador MSP430

Conrado P. L. Gouvêa, Julio López[1]

[1]Instituto de Computação, Universidade Estadual de Campinas (Unicamp)

{conradoplg,jlopez}@ic.unicamp.br

Abstract. *We describe a software implementation of Elliptic Curve Cryptography (ECC) and Pairing-Based Cryptography for the MSP430 microcontroller family, which is used in sensors such as the Tmote Sky, TelosB and TinyNode. The ECDSA digital signature protocol for ECC and the NIKDP non-interactive key distribution protocol for PBC were implemented on the 80 and 128-bit levels of security, using binary and prime fields. We have obtained a pairing computation that is 21% to 28% faster due to a new optimization for modular reduction in prime fields and an efficient binary field implementation.*

Resumo. *Descreve-se uma implementação em software de Criptografia de Curvas Elípticas (CCE) e Criptografia Baseada em Emparelhamentos (CBE) para o microcontrolador MSP430, utilizado em sensores como Tmote Sky, TelosB e TinyNode. Para a CCE, foi implementado o protocolo ECDSA de assinatura digital e para a CBE, o protocolo NIKDP de acordo de chaves não-interativo, nos níveis de segurança de 80 e 128 bits, usando corpos primos e binários. Obteve-se um cálculo de emparelhamento de 21% a 28% mais rápido através de uma nova otimização para a redução modular em corpos primos e uma implementação eficiente de corpos binários.*

1. Introdução

Redes de sensores sem fio são compostas por inúmeros dispositivos posicionados em uma área de interesse, cuja finalidade é coletar dados sobre o ambiente e enviá-los para uma estação base próxima. Idealmente, tais sensores devem ter baixo custo (muitas vezes, são descartáveis) e por esta razão normalmente têm capacidade extremamente limitada. Assim, proteger a sua comunicação, que é facilmenter interceptável, consiste em um desafio que tem motivado muitos trabalhos.

Na literatura recente foram bem estabelecidas as vantagens da criptografia assimétrica sobre a simétrica neste cenário, devido ao problema de gerenciamento de chaves [Oliveira et al. 2008]. Na assimétrica, dois sistemas se sobressaem: a Criptografia de Curvas Elípticas (CCE) e a Criptografia Baseada em Emparelhamentos (CBE). O objetivo deste trabalho é descrever uma implementação em software do estado da arte desses dois tipos de sistemas, nos níveis de segurança de 80 e 128 bits, para a família de microcontroladores MSP430 de 16 bits. Tal família foi escolhida por estar presente em múltiplos sensores como o Tmote Sky, TelosB e TinyNode; e por possuir um baixo consumo de energia. O nível de segurança de 80 bits é o mais utilizado em pesquisas recentes, o que permite realizar comparações, e também é mais viável. Já o nível de 128 bits, embora

Tabela 1. Instruções principais da família MSP430

Instrução	Operação
`mov` *src*,*dst*	`dst = src`
`add` *src*,*dst*	`dst += src` (soma)
`addc` *src*,*dst*	`dst += src + C` (soma com *carry*)
`adc` *dst*	`dst += C` (adiciona *carry*)
`rrc` *op*	*Shift* de *op* para a direita com *carry*. *carry* vira bit mais significativo; bit menos significativo é escrito no *carry*
`rra` *op*	*Shift* aritmético de *op* para a direita. Bit mais significativo não é alterado; bit menos significativo é escrito no *carry*
`rla` *op*	*Shift* de *op* para a esquerda. Bit menos significativo é zerado
`rlc` *op*	*Shift* de *op* para a esquerda. *Carry* vira bit menos significativo; bit mais significativo é escrito no *carry*
`swpb` *op*	Troca os bytes de *op*

computacionalmente mais caro, começa a receber mais atenção por ser considerado o mínimo aceitável em certas aplicações. Em particular, destaca-se que a Infra-Estrutura de Chaves Públicas Brasileira (ICP-Brasil) aceita, no mínimo, o nível de 128 bits de segurança [ICP-Brasil 2009].

2. A Família MSP430

Os microcontroladores da família MSP430 possuem várias características em comum como serem de 16 bits e possuírem o mesmo conjunto de 27 instruções e 12 registradores de propósito geral. A frequência de clock e tamanhos de ROM e RAM variam para cada membro — o MSP430F1611, utilizado pelo Tmote Sky e TelosB, possui clock de 8 MHz, 48 KB de ROM e 10 KB de RAM; enquanto o MSP430F2417 utilizado pelo TinyNode 184 possui clock de 16 MHz, 8 KB de RAM e 92 KB de ROM.

O conjunto de instruções da família MSP430 inclui adições, subtrações e shifts de apenas um bit; as mais importantes estão listadas na Tabela 1. Multiplicações de números inteiros são realizadas através de um multiplicador em hardware, um periférico mapeado em memória presente nos dois modelos citados. Não há instrução de divisão. Operandos podem ser referenciados através de quatro modos de endereçamento. As instruções suportam o uso de constantes imediatas (por exemplo, somar 20 a um registrador) que são codificadas em palavras de *offset* adjacentes às instruções. As constantes -1, 0, 1, 2, 4 e 8 não requerem palavras de *offset* devido à uma codificação especial.

O número de ciclos tomado na execução de uma instrução é calculado de forma relativamente simples, salvo algumas exceções. Primeiramente, é necessário um ciclo para se ler a instrução propriamente dita. Adiciona-se um ciclo para cada origem em memória (leitura) e dois ciclos para cada destino em memória (escrita). Finalmente, adiciona-se um ciclo para cada palavra de *offset* utilizada.

3. Protocolos Criptográficos

3.1. Fundamentos

Uma curva elíptica E sobre um corpo K é definida pela equação $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$, com $a_i \in K$. Um ponto em uma curva elíptica é um par $(x, y) \in K \times K$ que satisfaz a equação da curva. Pode-se definir uma operação de soma de pontos de forma que o conjunto de todos os pontos da curva, mais um elemento neutro denominado ponto no infinito, forme um grupo. Tal grupo é denotado $E(K)$. Dado um ponto $P = (x_P, y_P)$ de $E(K)$ e um inteiro k, define-se a operação de multiplicação de ponto kP como sendo a soma de k parcelas do ponto P, utilizando sucessivamente a operação de soma de pontos em uma curva elíptica.

Dados três grupos $\mathbb{G}_1$, $\mathbb{G}_2$ e $\mathbb{G}_T$ de ordem prima r, com $\mathbb{G}_1$ e $\mathbb{G}_2$ aditivos e $\mathbb{G}_T$ multiplicativo, um emparelhamento bilinear é definido como um mapa

$$e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

com a propriedade de bilinearidade, isto é, para todo $a, b \in \mathbb{Z}$, $R \in \mathbb{G}_1$ e $S \in \mathbb{G}_2$ tem-se que $e(aR, bS) = e(R, S)^{ab}$. Na CBE, utiliza-se emparelhamentos onde $\mathbb{G}_1$ e $\mathbb{G}_2$ são subgrupos de grupos de pontos em curvas elípticas e $\mathbb{G}_T$ é o subgrupo multiplicativo de um corpo finito. Se $\mathbb{G}_1 = \mathbb{G}_2$, diz-se que o emparelhamento é simétrico. Neste trabalho, o emparelhamento sobre corpo binário é simétrico e sobre corpo primo é assimétrico.

3.2. ECDSA

Neste trabalho foi implementado o protocolo criptográfico *Elliptic Curve Digital Signature Algorithm* (ECDSA) [National Institute of Standards and Technology 2009] de assinatura digital, por ser extremamente popular e padronizado. Ele é composto de três operações: geração de chave, assinatura e verificação. A geração de chave consiste na multiplicação de um ponto fixo G, que é conhecido por todos os participantes, pela chave privada (um número inteiro) escolhida aleatoriamente. A assinatura também requer uma multiplição de G, enquanto a verificação requer o cálculo de $kG + \ell Q$ onde Q é um ponto correspondente à chave pública de quem assinou a mensagem. Para mais detalhes, refere-se ao padrão [National Institute of Standards and Technology 2009].

Diz-se que um esquema criptográfico oferece um nível de n bits de segurança se, para quebrá-lo, são necessários um número de passos da ordem de 2^n. Nos casos de interesse neste trabalho, uma curva sobre um corpo de n bits fornece $n/2$ bits de segurança. Assim, os níveis de segurança de 80 e 128 bits requerem curvas sobre corpos de no mínimo 160 e 256 bits, respectivamente. As curvas escolhidas para se implementar o ECDSA foram a secp160r1 [Certicom Research 2010] (sobre corpo primo de 160 bits), P-256 (sobre corpo primo de 256 bits), K-163 (sobre o corpo binário $\mathbb{F}_{2^{163}}$) e K-283 (sobre $\mathbb{F}_{2^{283}}$) [National Institute of Standards and Technology 2009]. Tais curvas permitem uma redução modular rápida (com somente adições, shifts e xors [Hankerson et al. 2004]) devido às formas especiais dos módulos primos e polinômios de redução.

3.3. NIKDP

Um dos problemas da CCE consiste na autenticação de chaves públicas, que requer uma infra-estrutura que pode ser cara no contexto de redes de sensores. A Criptografia Baseada em Identidades (CBI) oferece uma alternativa à criptografia assimétrica clássica onde

a chave pública consiste na identidade do participante (como seu e-mail ou número de série) e assim tais chaves são implicitamente autenticadas. A sua desvantagem é requerer um Centro de Geração de Chaves (CGC) responsável por gerar as chaves privadas dos participantes do sistema e que portanto pode se fazer passar por qualquer um deles. Porém, no cenário de redes de sensores sem fio, tal centro é aceitável porque os sensores confiam implicitamente no CGC, que consiste nos próprios administradores da rede. O método mais popular para instanciar a CBI é a CBE.

Neste trabalho, foi implementado um protocolo de CBI para acordo de chaves não interativo proposto por [Sakai et al. 2000] e generalizado por [Dupont and Enge 2006], onde ele é denominado *Non-Interactive Key Distribution Protocol* (NIKDP). Tal protocolo permite duas entidades combinarem uma chave mútua sem se comunicarem, somente possuindo a identidade um do outro. Por esse motivo, ele é bastante interessante para redes de sensores sem fio, pois nos sensores a comunicação geralmente consome muito mais energia do que a computação. A seguir, descreve-se tal protocolo.

Sejam $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ e $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_2$ duas funções de hash. O CGC gera sua chave mestra escolhendo aleatoriamente $s \in \{1, ..., r-1\}$. A distribuição das chaves privadas é feita fornecendo ao sensor de identidade *A*, antes de se implantá-lo, as chaves $S_{1\mathrm{A}} = sH_1(A)$ e $S_{2\mathrm{A}} = sH_2(A)$.

Suponha que os sensores com identidades *A* e *B* desejam computar uma chave compartilhada. Se o emparelhamento for simétrico, $\mathbb{G}_1$ e $\mathbb{G}_2$ são o mesmo grupo, assim como as duas funções de hash são iguais e as duas chaves privadas são iguais. Assim, *A* computa $e(S_{1\mathrm{A}}, H_1(B)$ e *B* computa $e(S_{1\mathrm{B}}, H_1(A))$. Devido à bilinearidade e à simetria, tem-se que

$$e(S_{1\mathrm{A}}, H_1(B)) = e(H_1(A), S_{1\mathrm{B}}) = e(S_{1\mathrm{B}}, H_1(A)) \ . \tag{1}$$

Com isso os dois sensores geram o mesmo valor, que pode ser utilizado para se derivar uma chave. Se o emparelhamento for assimétrico, a última igualdade da Equação 1 não é válida. Mas pode-se chegar a duas equações: $e(S_{1\mathrm{A}}, H_2(B)) = e(H_1(A), S_{2\mathrm{B}})$ e $e(H_1(B), S_{2\mathrm{A}}) = e(S_{1\mathrm{B}}, H_2(A))$; basta agora que os dois sensores escolham uma das equações e cada um compute um dos lados dela. Em [Szczechowiak et al. 2009] sugere-se que os sensores podem combinar qual das equações utilizar com uma pequena quantidade de comunicação, mas existe uma alternativa simples que mantém o aspecto não interativo. Pode-se definir que o sensor com identidade lexicograficamente menor use sua primeira chave privada no primeiro argumento do emparelhamento, e o sensor com identidade maior use sua segunda chave no segundo parâmetro. Assim, implicitamente, os dois sensores utilizam a mesma equação e computam o mesmo valor.

A segurança do NIKDP pode ser fundamentalmente reduzida à dificuldade de se resolver o problema do logaritmo discreto em $\mathbb{G}_1$, $\mathbb{G}_2$, e $\mathbb{G}_T$. Como $\mathbb{G}_1$ e $\mathbb{G}_2$ são curvas elípticas, utiliza-se o mesmo raciocínio utilizado no ECDSA — os corpos subjacentes devem ter no mínimo 160 e 256 bits para os níveis de segurança de 80 e 128 bits, respectivamente. O corpo finito do qual $\mathbb{G}_T$ é subgrupo, contudo, deve ter ordem maior que $\mathbb{G}_1$ e $\mathbb{G}_2$ devido à existência de ataques especializados de complexidade sub-exponencial para corpos finitos. O padrão [National Institute of Standards and Technology 2007] recomenda ordens de 1024 e 3072 bits para os níveis de 80 e 128 bits, respectivamente. Para implementar o NIKDP, foram escolhidas duas curvas BN [Barreto and Naehrig 2006] sobre corpos primos de 160

e 256 bits utilizando o emparelhamento Optimal Ate [Vercauteren 2010], além de curvas supersingulares sobre os corpos binários $\mathbb{F}_{2^{271}}$ (como em [Szczechowiak et al. 2009]) e $\mathbb{F}_{2^{353}}$ utilizando o emparelhamento η_T [Barreto et al. 2007]. Nota-se que a curva sobre $\mathbb{F}_{2^{271}}$ oferece segurança de apenas 70 bits, mas foi implementada para permitir comparações. A curva sobre $\mathbb{F}_{2^{353}}$ oferece 80 bits de segurança.

4. Implementação em Software

O código de nossa implementação foi escrito na linguagem C, com operações críticas de corpo finito escritas na linguagem *assembly*. Tal implementação foi integrada com o *toolkit* criptográfico RELIC[1]. Utilizou-se o compilador MSPGCC versão 3.2.3. Para executar os programas, foi utilizado o simulador MSPsim [Eriksson et al. 2007].

4.1. Aritmética de Corpos Primos

A aritmética de corpos primos consiste nas operações de adição, subtração, multiplicação, quadrado e inversão. Dessas, as mais importantes são a multiplicação e quadrado, pois protocolos tanto de CCE e CBE gastam cerca de 75% do seu tempo nelas. A multiplicação (analogamente, o quadrado) em um corpo primo consiste em dois passos: a multiplicação inteira dos dois operandos de n palavras cada, obtendo-se um resultado intermediário de $2n$ palavras, e a posterior redução módulo o primo para se obter o resultado final de n palavras.

A multiplicação de inteiros é realizada com o algoritmo Comba [Comba 1990], uma versão orientada a colunas do algoritmo usual de multiplicação. Também pode-se utilizar um algoritmo híbrido que combina as duas técnicas, como em [Szczechowiak et al. 2009]. Contudo, pode-se obter um melhor desempenho utilizando-se o Comba pois o passo fundamental de tal algoritmo consiste em multiplicar duas palavras obtendo-se um resultado de duas palavras, e então somá-las em um acumulador de três palavras (a terceira acumula os *carries* das somas); e tal operação de multiplicar e acumular é fornecida pelo multiplicador em hardware da MSP430. Conforme [Gouvêa and López 2009], o seu uso permite uma multiplicação mais eficiente. Adicionalmente, pode-se utilizar o algoritmo Karatsuba-Ofman [Karatsuba and Ofman 1963] para calcular o produto de dois operandos de n palavras dividindo-os em duas partes de $n/2$ palavras cada, e realizando-se três multiplicações entre operandos de $n/2$ palavras mais algumas adições. Verificou-se que o uso de Karatsuba-Ofman é vantajoso no corpo de 256 bits, e portanto o algoritmo foi utilizado nesse caso.

A redução é realizada com o algoritmo de Montgomery [Montgomery 1985], pois ele não requer divisões, que não estão disponíveis na MSP430. Fundamentalmente, ele tem a mesma estrutura que o algoritmo Comba, porém com o primeiro operando sendo o módulo primo e o segundo gerado durante o algoritmo, e o resultado sendo adicionado ao número sendo reduzido. Assim, pode-se utilizar a mesma otimização do uso da operação de multiplicar e acumular do multiplicador em hardware, como descrito em [Gouvêa and López 2009].

No contexto de CBE, encontrou-se uma otimização bastante eficaz para a redução modular. Em curvas BN, o módulo primo é dado pelo polinômio $p(x) = 36x^4 + 36x^3 +$

[1] `http://code.google.com/p/relic-toolkit/`

$24x^2 + 6x + 1$, onde x é o parâmetro que define a curva sendo utilizada. Considere as curvas geradas pelos valores $x = 2^{38} + 2^5 + 2^4 + 1$ (adequada para o nível de segurança de 80 bits) e $x = -2^{62} - 2^{55} - 1$ (sugerida em [Nogami et al. 2008], adequada para 128 bits). Os módulos primos, nesses casos, são

$$p_1 = \texttt{0x2400 0000 6ED0 0000 7FE9 C000 419F EC80 0CA0 35C7},$$
$$p_2 = \texttt{0x2523 6482 4000 0001 BA34 4D80 0000 0008}$$
$$\texttt{6121 0000 0000 0013 A700 0000 0000 0013},$$

respectivamente, na base 16. Pode-se observar que p_1 possui dois dígitos de 16 bits valendo zero enquanto p_2 possui cinco dígitos valendo zero e um dígito valendo um. Como a redução Montgomery é análoga a uma multiplicação, pode-se descartar os passos envolvendo multiplicações por dígitos zero e otimizar passos envolvendo multiplicações por dígitos um. Por exemplo, para p_2, antes eram necessários $16^2 = 256$ passos de multiplicar e acumular durante a redução Montgomery. Ignorando-se os passos envolvendo multiplicações por zero, são necessários $256 - 16 \cdot 5 = 176$ passos, uma redução de 31%.

4.2. Aritmética de Corpos Binários

A arimética de corpos binários consiste nas operações de adição, subtração, multiplicação, inversão e raiz quadrada. O quadrado aqui se torna mais importante, pois pode ser calculado mais eficientemente e de forma bastante diferente do que a multiplicação. Dessas, as mais críticas são a multiplicação, quadrado e raiz quadrada.

A multiplicação em um corpo binário também é composta de dois passos: multiplicação polinomial dos operandos seguida de redução polinomial. A multiplicação de polinômios é realizada através do algoritmo López-Dahab (LD) [López and Dahab 2000], que é apontada como a mais eficiente neste cenário por múltiplos trabalhos [Szczechowiak et al. 2009, Aranha et al. 2010]. Opcionalmente, também pode-se utilizar o algoritmo de Karatsuba-Ofman de maneira análoga ao corpo primo. Alternativamente, a estratégia de blocos para se implementar a multiplicação LD [Aranha et al. 2010] pode fornecer resultados superiores ao Karatsuba-Ofman.

O quadrado de um elemento em um corpo binário é composto pelo quadrado polinomial do elemento seguido da redução polinomial. Sendo $a(z) = a_{m-1}z^{m-1} + \ldots + a_1 z + a_0$ um elemento do corpo binário $\mathbb{F}_{2^m}$, observa-se que $a(z)^2 = \sum_{i=0}^{m-1} a_i z^{2i} = a_{m-1}z^{2m-2} + \ldots + a_2 z^4 + a_1 z^2 + a_0$. Isto é, o quadrado do polinômio $a(z)$ pode ser computado inserindo-se zeros entre pares de bits sucessivos da representação binária de $a(z)$. Tal processo pode ser acelerado com o uso de tabelas de *lookup* que fornecem a expansão de 16 bits de todos os valores de 8 bits possíveis. Portanto, a tabela possui $2^8 \cdot 2 = 512$ bytes, e pode ser armazenada e lida diretamente da ROM, sem ocupar espaço na RAM.

A redução realizada após uma multiplicação ou quadrado polinomial pode ser acelerada devido à forma especial dos polinômios de redução das curvas escolhidas. Assim, pode-se computá-la somente com shifts e xors, como descrito em [Hankerson et al. 2004], por exemplo. Para a curva sobre $\mathbb{F}_{2^{271}}$ utilizada no emparelhamento η_T, pode-se escolher como polinômio de redução o trinômio $z^{271} + z^{201} + 1$ ou o pentanômio $z^{271} + z^{207} + z^{175} + z^{111} + 1$. Escolheu-se o pentanômio pois a redução, neste caso, necessita somente de shifts de um bit, ao contrário do trinômio que requer shifts de um e seis bits. Analo-

gamente, escolheu-se o trinômio $z^{353} + z^{95} + 1$ para o corpo $\mathbb{F}_{2^{353}}$ pois a redução, neste caso, necessita de shifts de somente um e dois bits.

A raiz quadrada de um elemento de um corpo binário consiste em, dado um elemento $a(z)$, encontrar $c(z)$ tal que $c(z)^2 = a(z)$. Neste trabalho, ela é utilizada somente nas curvas sobre $\mathbb{F}_{2^{271}}$ e $\mathbb{F}_{2^{353}}$, durante o cálculo do emparelhamento η_T. Como detalhado em [Fong et al. 2004], pode-se observar que

$$\sqrt{a(z)} = \sum_{i \text{ par}} a_i z^{\frac{i}{2}} + \sqrt{z} \sum_{i \text{ ímpar}} a_i z^{\frac{i-1}{2}},$$

isto é, cada somatório nada mais é do que o agrupamento dos bits pares e ímpares da expansão binária de $a(z)$, e o segundo somatório deve ser multiplicado por $\sqrt{z}$. Tal cálculo pode ser acelerado com o uso de tabelas de *lookup* que fornecem a contração dos 8 bits pares e ímpares de uma palavra de 16 bits. Tais tabelas possuem $2^8 = 512$ bytes cada, e também podem ser armazenadas em ROM. A multiplicação do somatório dos bits ímpares com $\sqrt{z}$ é calculada eficientemente nas curvas citadas notando-se que $\sqrt{z} \equiv z^{136} + z^{104} + z^{88} + z^{56}$ em $\mathbb{F}_{2^{271}}$ e que $\sqrt{z} \equiv z^{177} + z^{48}$ em $\mathbb{F}_{2^{353}}$. Verifica-se que a multiplicação pelo primeiro polinômio pode ser feita somente com shifts de oito bits, que são computados com o auxílio da instrução `swpb`; e com somente shifts de um bit para o segundo polinômio.

A inversão no corpo binário pode ser calculada através do algoritmo de Euclides estendido. Para otimizá-la, foi utilizada uma abordagem análoga a [Aranha et al. 2010] com o uso de funções dedicadas para o shift de um a oito bits que são fundamentais durante a inversão.

4.3. Algoritmos para Multiplicação de Ponto

Como mencionado, as operações principais de CCE utilizadas no ECDSA são a multiplicação de um ponto fixo (kG) e a multiplicação simultânea de um ponto fixo e um aleatório ($kG + \ell Q$). Para a multiplicação de ponto fixo, foram selecionados o métodos wNAF [Solinas 2000] e Comb [Lim and Lee 1994] no caso primo e o wTNAF no caso binário [Solinas 2000]. Tais métodos utilizam tabelas de pré-computação de múltiplos do ponto fixo G para acelerar o cálculo. Para a multiplicação simultânea, foi selecionado o método de entrelaçamento de wNAFs e wTNAFs [Möller 2001] para o caso primo e binário, respectivamente. Tal método utiliza uma tabela pré-computada para o ponto fixo e uma tabela menor, calculada na hora da multiplicação, para o ponto aleatório.

4.4. Algoritmos para Cálculo de Emparelhamentos

O algoritmo padrão para o cálculo de emparelhamentos é o algoritmo de Miller [Miller 1986]. Foi utilizado o emparelhamento Optimal Ate [Vercauteren 2010]. Neste trabalho, somente são utilizadas curvas BN [Barreto and Naehrig 2006], e segue-se a abordagem de [Devegili et al. 2007] na sua implementação, em especial no cálculo em corpos de extensão. Para se obter o melhor desempenho possível, é necessária a utilização de inúmeras otimizações propostas na literatura, como o uso de coordenadas Jacobianas, uso de *twists*, eliminação de denominador, adição encapsulada de pontos [Chatterjee et al. 2005], novas fórmulas para a multiplicação pelo cofator em $\mathbb{G}_2$ [Scott et al. 2009] e cálculo aperfeiçoado de quadrados na exponenciação final [Granger and Scott 2010].

5. Resultados

Utilizando o simulador MSPsim, foi realizada a medição dos ciclos de cada operação implementada. Em alguns casos são fornecidos tempos em segundos; esses são derivados do número de ciclos presumindo um *clock* de 8 milhões de Hertz. O *clock* exato depende do modelo e da voltagem fornecida. Naturalmente, para modelos de 16 MHz, pode-se obter uma estimativa dos tempos finais dividindo por dois os tempos fornecidos.

5.1. Tempos de Execução para Corpos Primos

A Tabela 2 apresenta os tempos das múltiplas operações envolvidas na aritmética de corpos primos. Primeiramente, observa-se que foi obtido um pequeno ganho em comparação aos mesmos algoritmos de [Gouvêa and López 2009]. Particularmente, na redução, obteve-se um ganho de cerca de 8% ao se observar que não é necessário ler o módulo primo da memória — como ele é fixo, pode-se incluir seus dígitos diretamente nas instruções, utilizando constantes imediatas. Contudo, o ganho mais expressivo vem do uso de primos esparsos na redução — 14% para o corpo de 160 bits de 26% para o corpo de 256 bits. Ao se considerar a multiplicação no corpo, tais ganhos são reduzidos para 7% e 13%. O tempo do algoritmo HMMB listado na Tabela 2 corresponde à nossa implementação em software da técnica para hardware introduzida em [Fan et al. 2009]. Tal técnica explora o formato especial do módulo primo em curvas BN e utiliza uma representação polinomial dos operandos. Apesar de bastante interessante, é razoavelmente complexa. Como se pode observar, ela não fornece ganhos significativos nesta plataforma em relação à técnica proposta de redução utilizando primos esparsos.

5.2. Tempos de Execução para Corpos Binários

A Tabela 3 apresenta os tempos para as diversas operações da aritmética de corpos binários. Primeiramente, observa-se que foi possível obter um ganho de 5% em relação à multiplicação de [Szczechowiak et al. 2009]. Adicionalmente, utilizando a multiplicação LD com blocos ao invés de Karatsuba, foi possível obter um ganho de 10% em relação ao mesmo trabalho. Finalmente, o quadrado e raiz são 35% e 53% mais rápidos que os de [Szczechowiak et al. 2009] — presume-se que o motivo seja o uso de um pentanômio ao invés de um trinômio que, como indicado em [Scott 2007] pode permitir uma implementação mais eficiente, como se verificou neste caso.

5.3. Tempos de Execução para a CCE

A Tabela 4 apresenta os desempenhos dos algoritmos de CCE, e a Figura 1 ilustra os tempos obtidos no ECDSA. No caso primo, obteve-se resultados análogos a [Gouvêa and López 2009]. Observa-se que a curva binária fornece um desempenho maior do que curva prima. Utilizando-se a multiplicação de ponto Comb, a curva prima consegue se aproximar à curva binária na mutiplicação de ponto fixo (e consequentemente na assinatura ECDSA). Contudo, vale notar que isso acarreta num custo maior de memória, já que a tabela de pré-computação do Comb não pode ser reutilizada no entrelaçamento de wNAFs.

5.4. Tempos de Execução para a CBE

A Tabela 5 apresenta os desempenhos de algoritmos de CBE, e a Figura 1 ilustra os tempos obtidos no NIKDP. No nível de 80 bits, primeiramente observa-se que a curva BN

Tabela 2. Tempos em ciclos da aritmética em corpos primos

Algoritmo	Corpo de 160 bits	Corpo de 256 bits
Multiplicação Comba	1 565	3563
Quadrado Comba	1 350	2 946
Redução		
Montgomery[a]	1 785	3 989
Montgomery	1 659	3 600
Montgomery, primo esparso	1 413	2 670
secp160r1 / P-256	342	709
Multiplicação no corpo		
Comba[a]	3 389	7 604
Comba	3 263	7 198
Comba, primo esparso	2 998	6 249
HMMB		7 797
Quadrado no corpo		
Comba[a]	3 172	6 952
Comba	3 046	6 581
Comba, primo esparso	2 783	5 630
Inverso no corpo, Euclides estendido	194 434	403 435

[a] [Gouvêa and López 2009]

Tabela 3. Tempos em ciclos da aritmética em corpos binários

Algoritmo	$\mathbb{F}_{2^{163}}$	$\mathbb{F}_{2^{271}}$	$\mathbb{F}_{2^{283}}$	$\mathbb{F}_{2^{353}}$
Multiplicação no corpo				
LD, Karatsuba[a]		10 147		
LD, Karatsuba		9 647		14 362
LD, blocos	4 355	8 712	9 543	
Quadrado no corpo				
Tabela[a]		1 363		
Tabela	697	799	1 277	983
Raiz quadrada no corpo				
Tabela[a]		1 644		
Tabela		775		934
Redução				
Rápida	448	430	888	494
Inversão no corpo				
Euclides estendido	84 367	175 236	179 803	256 810

[a] [Szczechowiak et al. 2009]

Tabela 4. Tempos em segundos de algoritmos e protocolos de CCE

Algoritmo	secp160r1	K-163	P-256	K-283
Multiplicação de ponto fixo				
Comb, $w = 3$	0,274		0,852	
5NAF	0,443	0,224	1,422	0,683
Multiplicação simultânea				
Entrelaçamento ($w_k = 5$, $w_\ell = 4$)	0,589	0,432	1,832	1,432
ECDSA				
Geração de chave, Comb	0,277		0,860	
Geração de chave, 5NAF	0,460	0,229	1,443	0,722
Assinar, Comb	0,317		0,924	
Assinar, 5NAF	0,487	0,256	1,485	0,772
Verificar	0,625	0,481	1,882	1,474

Tabela 5. Tempos em segundos de algoritmos e protocolos de CBE

	Corpo Binário		Corpo Primo	
Algoritmo	70 bits	80 bits	80 bits	128 bits
Emparelhamento				
Tate[a] / Xate[b]			5,109	14,516
Optimal Ate			4,051	10,695
η_T, Karatsuba[c]	1,762			
η_T, Karatsuba	1,375	2,586		
η_T, Blocos	1,260			
NIKDP				
Geração de chave	1,020[d]	1,650	4,079	13,253
Acordo de chave ($\mathbb{G}_1$)	1,592[d]	2,668	4,139	10,987
Acordo de chave ($\mathbb{G}_2$)			5,439	13,578

[a] [Gouvêa and López 2009], em curva MNT com $k = 6$, nível de segurança de 70 bits
[b] [Gouvêa and López 2009]
[c] [Szczechowiak et al. 2009]
[d] Usando multiplicação Karatsuba

oferece um ganho de 20,7% em relação à curva MNT em [Gouvêa and López 2009]. Tal resultado pode parecer surpreendente visto que a curva BN utiliza um $\mathbb{G}_T$ de 1920 bits, muito além dos 1024 bits necessários para este nível de segurança. Contudo, ele é explicado pelas inúmeras otimizações existentes para curvas BN. No caso binário, observa-se um ganho de 22% em relação a [Szczechowiak et al. 2009] com a mesma multiplicação Karatsuba, e 28,5% com a multiplicação em blocos. Tal ganho aparenta vir da otimização da aritmética e do uso de pentanômio como polinômio de redução.

No nível de 128 bits, pode-se observar um ganho de 26,3% em relação a [Gouvêa and López 2009] no cálculo de emparelhamento. Tal ganho é explicado pelo uso da otimização de primo esparso proposta e pela nova fórmula para o cálculo da expo-

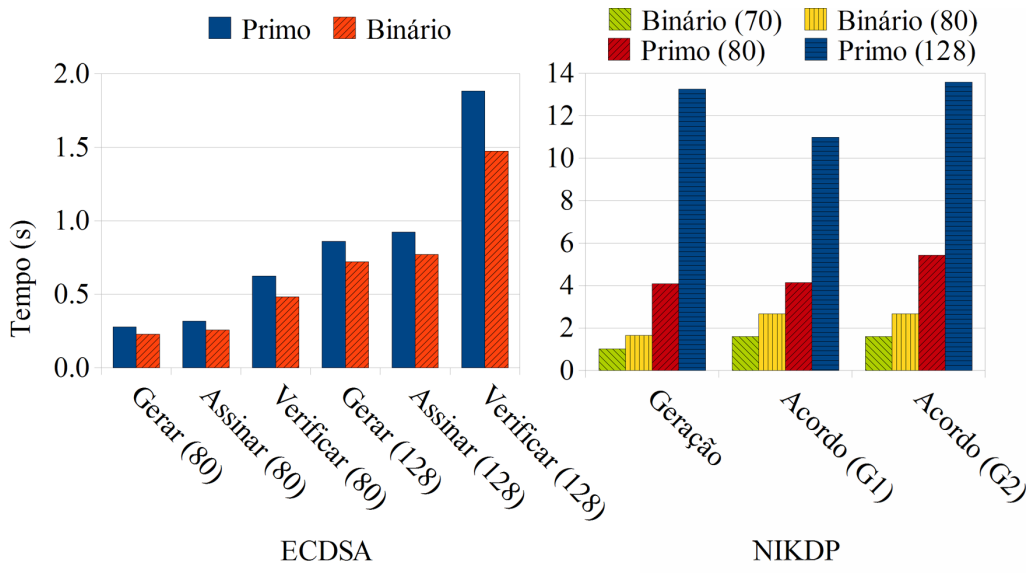

Figura 1. Tempos do ECDSA e NIKDP

nenciação final em [Granger and Scott 2010]. É interessante notar que o acordo de chave com hash em $\mathbb{G}_2$ é mais caro do que em $\mathbb{G}_1$ no caso assimétrico devido à necessidade de se multiplicar o ponto obtido após o hash pelo cofator da curva elíptica.

5.5. ROM, RAM e Energia

A Figura 2 apresenta o tamanho de ROM, o uso de RAM por variáveis globais e o uso máximo de RAM na pilha. Pode-se verificar que o uso de ROM é relativamente alto, considerando os 48 KB disponíveis nos sensores Tmote Sky e TelosB. O uso de RAM pode ser considerado aceitável ao se observar que a maior parte é alocada da pilha e liberada após o cálculo. Destaca-se que o uso de Karatsuba para o NIKDP sobre o corpo binário $\mathbb{F}_{2^{271}}$ fornece uma economia de 6,8 KB de ROM, aumentando o tempo do acordo de chaves em 115 ms. Também percebe-se que sobre corpos primos se obtém programas menores, provavelmente pelo fato da existência do multiplicador em hardware que permite realizar cálculos com um número menor de instruções. A Tabela 6 apresenta o consumo estimado de energia dos algoritmos implementados, assumindo uma corrente de 3,31 mA sob 3,6 V, necessários para se obter o *clock* desejado [Dudacek and Vavricka 2007]. Considerando-se que com uma bateria de 500 mAh pode-se calcular cerca de 40 000 vezes a operação mais cara implementada (acordo de chave em $\mathbb{G}_2$ no caso primo, 128 bits de segurança), conclui-se que o consumo de energia não é tão crítico neste cenário.

Tabela 6. Consumo em milijoules de algoritmos implementados

	Corpo Binário			Corpo Primo	
Algoritmo	70 bits	80 bits	128 bits	80 bits	128 bits
ECDSA: Assinar		3,1	9,2	3,8	11,0
ECDSA: Verificar		5,7	17,6	7,4	22,4
NIKDP: Acordo de chave	19,0	31,8		64,8	161,8

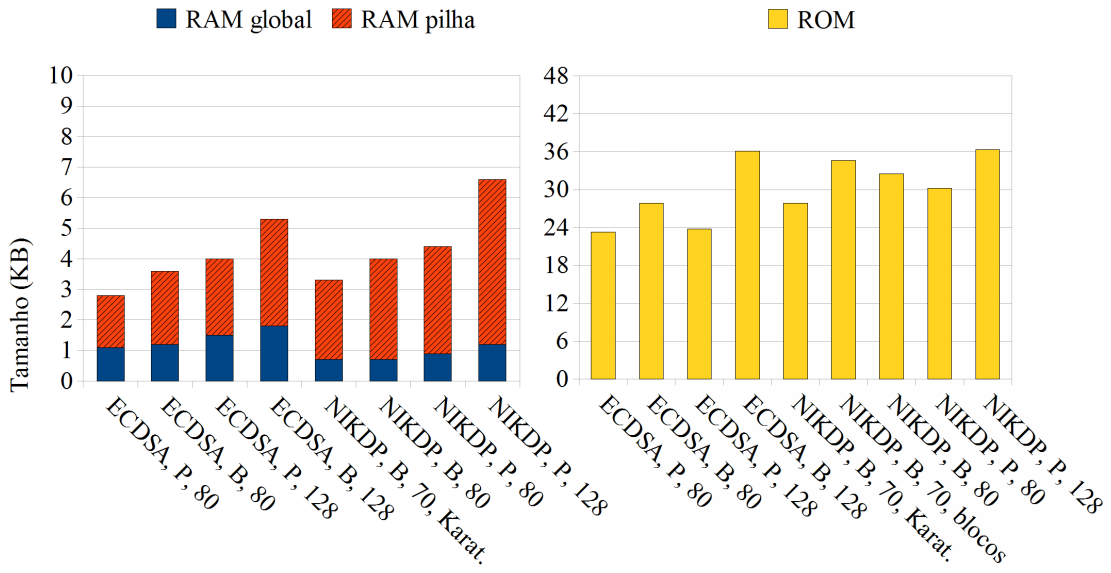

Figura 2. Tamanho de ROM e uso de RAM para protocolos de CCE e CBE

6. Conclusão

Apesar de desafiadora, a implementação de criptografia para redes de sensores sem fio é viável. Contudo, ainda é necessária pesquisa no sentido de melhorar a eficiência e o tamanho de código dos programas, principalmente no nível de segurança de 128 bits. Neste trabalho, foram apresentados os melhores tempos conhecidos para a CCE e CBE na família de microcontroladores MSP430. Em particular, obteve-se uma multiplicação de corpo primo 12% e 18% mais rápida para corpos primos de 160 e 256 bits, respectivamente, através da nova técnica proposta de redução com primo esparso (que nota a existência de multiplicações por zero, que podem ser descartadas), e uma multiplicação 14% mais rápida para corpo binário de 271 bits com a multiplicação em blocos. Em relação aos melhores tempos conhecidos, obteve-se um cálculo de emparelhamento 20,7% mais rápido sobre corpo primo de 160 bits, 26,3% mais rápido sobre corpo primo de 256 bits, e 28,5% mais rápido sobre corpo binário de 271 bits. Tais ganhos vêm, principalmente, do uso de redução com primo esparso no caso primo e do uso de pentanômio como polinômio de redução no caso binário (que reduz o número de *shifts* necessários), além de uma implementação cuidadosa em *assembly*. Destaca-se que a redução com primo esparso pode ser aplicada em outras plataformas de 8 e 16 bits, e também em maiores níveis de segurança.

Agradecimentos

Ao CNPq pela bolsa de produtividade em pesquisa ao segundo autor e a Diego Aranha pela ajuda com a implementação da operação de inversão em corpo binário.

Referências

Aranha, D. F., Oliveira, L. B., López, J., and Dahab, R. (2010). Efficient implementation of elliptic curve cryptography in wireless sensors. *Advances in Mathematics of Communications*, 4(2):169–187.

Barreto, P. S. L. M., Galbraith, S., Ó hÉigeartaigh, C., and Scott, M. (2007). Efficient pairing computation on supersingular abelian varieties. *Designs, Codes and Cryptography*, 42(3):239–271.

Barreto, P. S. L. M. and Naehrig, M. (2006). Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer Berlin / Heidelberg.

Certicom Research (2010). SEC 2: Recommended elliptic curve domain parameters. `http://www.secg.org/`.

Chatterjee, S., Sarkar, P., and Barua, R. (2005). Efficient computation of Tate pairing in projective coordinate over general characteristic fields. In *Information Security and Cryptology — ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 168–181. Springer Berlin / Heidelberg.

Comba, P. G. (1990). Exponentiation cryptosystems on the IBM PC. *IBM Systems Journal*, 29(4):526–538.

Devegili, A. J., Scott, M., and Dahab, R. (2007). Implementing cryptographic pairings over Barreto-Naehrig curves. In *Pairing-Based Cryptography — Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 197–207. Springer Berlin / Heidelberg.

Dudacek, K. and Vavricka, V. (2007). Experimental evaluation of the MSP430 microcontroller power requirements. In *The International Conference on "Computer as a Tool" — EUROCON, 2007*, pages 400–404.

Dupont, R. and Enge, A. (2006). Provably secure non-interactive key distribution based on pairings. *Discrete Applied Mathematics*, 154(2):270–276.

Eriksson, J., Dunkels, A., Finne, N., Österlind, F., and Voigt, T. (2007). MSPsim – an extensible simulator for MSP430-equipped sensor boards. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*.

Fan, J., Vercauteren, F., and Verbauwhede, I. (2009). Faster $\mathbb{F}_p$-arithmetic for cryptographic pairings on Barreto-Naehrig curves. In *Cryptographic Hardware and Embedded Systems — CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 240–253. Springer Berlin / Heidelberg.

Fong, K., Hankerson, D., López, J., and Menezes, A. (2004). Field inversion and point halving revisited. *IEEE Transactions on Computers*, 53(8):1047–1059.

Gouvêa, C. P. L. and López, J. (2009). Software implementation of pairing-based cryptography on sensor networks using the MSP430 microcontroller. In *Progress in Cryptology — INDOCRYPT 2009*, volume 5922 of *Lecture Notes in Computer Science*, pages 248–262. Springer Berlin / Heidelberg.

Granger, R. and Scott, M. (2010). Faster squaring in the cyclotomic subgroup of sixth degree extensions. In *Public Key Cryptography — PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 209–223. Springer Berlin / Heidelberg.

Hankerson, D., Menezes, A., and Vanstone, S. (2004). *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York.

ICP-Brasil (2009). Padrões e algoritmos criptográficos da ICP-Brasil. `http://www.iti.gov.br/`.

Karatsuba, A. and Ofman, Y. (1963). Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7:595.

Lim, C. H. and Lee, P. J. (1994). More flexible exponentiation with precomputation. In *Advances in Cryptology — CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 95–107. Springer Berlin / Heidelberg.

López, J. and Dahab, R. (2000). High-speed software multiplication in $\mathbb{F}_{2^m}$. In *Progress in Cryptology — INDOCRYPT 2000*, volume 1977 of *Lecture Notes in Computer Science*, pages 93–102. Springer Berlin / Heidelberg.

Miller, V. S. (1986). Short programs for functions on curves. *Unpublished manuscript*, 97:101–102.

Möller, B. (2001). Algorithms for multi-exponentiation. In *Selected Areas in Cryptography*, volume 2259 of *Lecture Notes in Computer Science*, pages 165–180. Springer Berlin / Heidelberg.

Montgomery, P. L. (1985). Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521.

National Institute of Standards and Technology (2007). Recommendation for key management. `http://www.itl.nist.gov`.

National Institute of Standards and Technology (2009). FIPS 186-3: Digital signature standard (DSS). `http://www.itl.nist.gov`.

Nogami, Y., Akane, M., Sakemi, Y., Kato, H., and Morikawa, Y. (2008). Integer variable χ-based Ate pairing. In *Pairing-Based Cryptography — Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 178–191. Springer Berlin / Heidelberg.

Oliveira, L., Scott, M., López, J., and Dahab, R. (2008). TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. In *Networked Sensing Systems, 2008. INSS 2008. 5th International Conference on*, pages 173–180.

Sakai, R., Ohgishi, K., and Kasahara, M. (2000). Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security, Okinawa, Japan*.

Scott, M. (2007). Optimal irreducible polynomials for GF(2^m) arithmetic. Cryptology ePrint Archive, Report 2007/192. `http://eprint.iacr.org/`.

Scott, M., Benger, N., Charlemagne, M., Perez, L. J. D., and Kachisa, E. J. (2009). Fast hashing to G_2 on pairing-friendly curves. In *Pairing-Based Cryptography — Pairing 2009*, volume 5671 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg.

Solinas, J. A. (2000). Efficient arithmetic on Koblitz curves. *Designs, Codes and Cryptography*, 19(2):195–249.

Szczechowiak, P., Kargl, A., Scott, M., and Collier, M. (2009). On the application of pairing based cryptography to wireless sensor networks. In *Proceedings of the second ACM conference on Wireless network security*, pages 1–12. ACM New York.

Vercauteren, F. (2010). Optimal pairings. *IEEE Transactions on Information Theory*, 56(1):455–461.