QC-MDPC McEliece: an Optimized Implementation of a New McEliece Variant

H. O. Martins and A. C. A. Nascimento

***Abstract*— This paper presents the implementation of an optimized version of a McEliece variant.The McEliece cryptosystem is an example of code-based cryptography which is an alternative to the most popular and commercial cryptosystems nowadays as it is believed to be immune to quantum computing. It has simple and fast algorithms, but its drawback is the size of the keys it has to deal with. By substituting the Goppa codes of the McEliece original proposalby LDPC and MDPC codes it's possible to achieve much smaller keys. And by applying programming technicssuch as parallelization of operations and also utilizing efficient decoders of LDPC codes it's possible to achieve really good results and optimal performances of the code-based cryptosystem showing that it really has to be considered as a strong substitute to RSA and DSA as quantum computers emerge to easily compute discrete logarithms and factor large integers.**

***Keywords*— Post-quantum cryptography, Code-based cryptography, Coding-theory, Efficient decoding.**

I. INTRODUÇÃO

A SEGURANÇA dos criptossistemas de chave pública utilizados na atualidade tais como o RSA e o DSA está fundamentada no enorme esforço computacional necessário para a fatoração de números inteiros grandes ou para o cálculo de logaritmos discretos. Porém, estes criptossistemas podem ser quebrados em tempo polinomial por intermédio de computadores quânticos executando o algoritmo de Shor ou de Grover [1].

Os computadores quânticos poderão vir a representar o fim do RSA e do DSA, mas não da criptografia assimétrica, pois existem criptossistemas como os baseados em códigos corretores de erro que continuam imunes tanto à computação quântica quanto à clássica ou tradicional.

Se correspondem a uma possível alternativa para a ameaça representada pelos computadores quânticos, o grande problema dos criptossistemas baseados em códigos é a sua eficiência, basicamente no que diz respeito ao tamanho das suas chaves. O criptossistema de chave pública de McEliece com códigos Goppa proposto a mais de 30 anos [2], configurado para padrões de segurança atuais de 128 bits, apresenta chaves com tamanho de mais de um milhão e meio de bits, enquanto o RSA, para o mesmo nível de segurança, trabalha com chaves de alguns poucos milhares de bits.

Pesquisas atuais estão sendo conduzidas na busca de códigos alternativos que substituam os códigos de Goppa e possibilitem aos criptossistemas baseados em códigos tamanhos de chaves significativamente menores, mas que preservem as propriedades de segurança destes criptossistemas. Recentemente, Misoczkiet al. [3] apresentaram um artigo no qual propõem a utilização de códigos quase-cíclicos com verificação de paridade de densidade moderada (códigos QC-MDPC) em substituição aos códigos de Goppa. Como resultado, alcançaram os 128 bits de nível de segurança com uma chave de tamanho inferior a 1000 bits.

II. CONTRIBUIÇÃO

Este trabalho consiste na implementação de uma versão otimizada de um criptossistema de McEliece baseado em códigos QC-MDPC. Esta implementação está fundamentada na proposta de Misoczkiet al [3].

Como a decodificação é em regra a etapa mais custosa em criptossistemas baseados em códigos, o trabalho procurou centralizar todos os seus esforços na construção de uma versão otimizada do decodificador.

Na construção do decodificador o presente trabalho agregou à implementação proposta por Misoczkiet al. a contribuição de von Maurich eGüneysu[4], que corresponde a uma variante eficiente do melhor decodificador para códigos QC-MDPC em dispositivos embarcados.

O trabalho também procurou fazer uso de algumas técnicas computacionais que economizam repetições na execução de tarefas exaustivas tais como multiplicações de matrizes, bem como recorreu a extensões do conjunto de instruções do processador, as quais possibilitam o acesso a registradores específicos do hardware de modo a obter ganho de performance no processamento de operações lógicas e matemáticas através da paralelização.

III. PRELIMINARES

O primeiro criptossistema baseado na teoria de códigos foi proposto em 1978 por R. J. McEliece [2]. Em sua descrição original, a chave secreta do criptossistema de McElice corresponde a um código de Goppa.

H. O. Martins, Universidade de Brasília (UnB), Brasília, Brasil, homer@unb.br

A. C. A. Nascimento, Universidade de Brasília (UnB), Brasília, Brasil, andclay@ene.unb.br

Invited Paper

Em [1] encontra-se bem detalhado o funcionamento do criptossistema de McEliece com códigos Goppa. Em [3] existe uma descrição bastante completa de códigos QC-MDPC, bem como da codificação/decodificação McEliece baseada em códigos QC-MDPC. A seguir segue um resumo das principais ideias acerca dos códigos QC-MDPC, assim como do criptossistemaMcEliece.

A. Códigos QC-MDPC

Um **código binário (*n*,*r*)-linear** C de comprimento *n*, dimensão $n - r$ e codimensão*r* corresponde a um subespaço vetorial $(n - r)$ dimensional de$\mathbb{F}_n^2$. Ele é gerado pelas linhas de uma matriz $G \in \mathbb{F}_2^{(n-r)\times n}$chamada matriz geradora de C. E também é o núcleo de uma matriz $H \in \mathbb{F}_2^{r\times n}$conhecida como matriz de verificação de paridade de C.

Um **código (*n*,*r*)-linear** é considerado **quase-cíclico** (**QC**) se existir um inteiro $n_0$tal que qualquer deslocamento circular de uma palavra-chave por n_0 posições dá como resultado uma outra palavra-chave.

Um **código (*n*,*r*,*w*)-MDPC** é um código linear de comprimento *n*, codimensão*r* e que admite uma matriz de verificação de paridade com peso de *Hamming* constante de suas linhas igual a *w*.

Para a construção de um **código (*n*,*r*,*w*)-QC-MDPC**escolhe-se uma palavra aleatória de comprimento$n = n_0.p$e peso *w* que corresponde à primeira linha da matriz H. As outras *r* - 1 linhas são obtidas por intermédio de *r* - 1 deslocamentos quase-cíclicos, de tal maneira que cada bloco $H_i$tem peso de linhaw_i tal que $w = \sum_{i=0}^{n_0-1} w_i$. A matriz geradora G pode ser calculada a partir dos blocos $H_i$como $G = [I \mid Q]$, onde I é a matriz identidade e $Q = \begin{bmatrix} (H_{n_0-1}^{-1}.H_0)^T \\ (H_{n_0-1}^{-1}.H_1)^T \\ \vdots \\ (H_{n_0-1}^{-1}.H_{n_0-2})^T \end{bmatrix}$.

B. Codificação/Decodificação QC-MDPC McEliece

As operações de geração de chaves, codificação e decodificação McEliece baseadas num código (*n*,*r*,*w*)-QC-MPDC com capacidade para correção de *t* erros são definidas como:

Geração das chaves: produza um vetor aleatório binário *h* de peso *w*. Ele corresponde à primeira linha da matriz H e as demais *r* - 1 linhas são obtidas através de deslocamentos quase-cíclicos de *h*. Obtenha a matriz G correspondente a H na forma reduzida escalonada. A chave pública é G e a chave privada é H.

Codificação: para codificar uma mensagem *m* produza um vetor aleatório binário *e* de peso menor ou igual a *t*. A mensagem codificada *x* é dada por $x \leftarrow mG + e$.

Decodificação: para decodificar *x* em *m*, calcule

$$mG \leftarrow \Psi_H(mG + e),$$

ondeΨ_Hé um decodificador para o código QC-MDPC com conhecimento da matriz H. A mensagem *m* é recuperada a partir das primeiras (*n* - *r*) posições de *m*G.

C. Decodificação Eficiente

A decodificação é a etapa que mais consome tempo na criptografia baseada em códigos. Portanto, a opção pelo algoritmo decodificador mais apropriado é crucial para a obtenção de uma performance desejável na tarefa de decodificação. Para decodificar códigos MDPC existem duas categorias de algoritmos: os mais simples, como aqueles que usam a técnica de inversão de bits (*bit-flipping*) e têm menor capacidade de correção de erros, e aqueles mais elaborados e que alcançam uma melhor capacidade corretora, como por exemplo o algoritmo soma-produto. Misoczkiet al [3] sugerem a utilização dos algoritmos de inversão de bits, tendo em vista que estes têm baixa complexidade, são iterativos e rápidos.

Em termos gerais, todos os algoritmos que usam a técnica de inversão de bits seguem a mesma ideia. Primeiramente, a cada iteração eles calculam a síndrome da mensagem a ser decodificada. A seguir são computados os números de equações de verificação de paridade não satisfeitas associadas a cada bit da mensagem. Cada bit associado a mais que *b* equações não satisfeitas é invertido. Este processamento é repetido até que a síndrome seja completamente zerada ou até que seja alcançado um número limite de iterações, quando considera-se que o processo de decodificação falhou. A grande diferença entre os diversos decodificadores é na maneira como o limite *b* é calculado, o que pode variar desde o maior número de equações não satisfeitas até o pré-cálculo de limites baseados em parâmetros do código utilizado.

O Algoritmo1 é apresentado em [3] e ilustra o parágrafo acima.

Algoritmo 1 variante do algoritmo de *bit-flipping* para decodificação de códigos QC-MDPC

Entrada: $y = mG + e$, com peso *w* e dimensão *n*
Saída: *c* tal que $Hc^T = 0$, ou FALHA.

```
while δ∈ N > 0
    s ← Hc^T
    for i = 1 to dim(s) do
     if s[i] = 1 then
        for j = 1 to w do
            contador[j] ← #upc para cada bit
  //#upc = número de equações de verificação de paridade
  //nãosatisfeitas
        end for
     end if
    end for
    Maxupc ← o maior #upc
    for i = 1 to n do
     if contador[i] ≥ (Maxupc − δ) then
        Flip c_i
     end if
    end for
    if Hc^T = 0 then
     return c
    endif
    δ ← δ – 1 // No caso de falha na decodificação
endwhile
return FALHA
```

Recentemente vários algoritmos de decodificadores foram analisados por Heyseet al.[5] quanto a sua adequação e performance ao trabalhar com os códigos QC-MDPC em dispositivos embarcados. O Algoritmo 2abaixo superou todos os outros tanto em tempo de processamento bem como na taxa de falha de decodificação.

Este decodificador pré-calcula seus limiares b_i baseado nos parâmetros do código como proposto por Gallager [6] e, ao contrário de outras soluções que recalculam a síndrome após cada iteração, apenas faz a atualização da mesma enquanto processa a decodificação da mensagem cifrada.

Algoritmo 2 varianteeficiente de algoritmo de *bit-flipping* para decodificação de códigos QC-MDPC

Entrada: $y = mG + e$, com peso w e dimensão n
Saída: c tal que $Hc^T = 0$, ou FALHA.
while$\delta \in N > 0$
  $s \leftarrow Hc^T$
  fori = 1 **to** dim(s) **do**
   if$s$[i] = 1 **then**
    for j = 1 **to**w**do**
     contador[j] ← #upc para cada bit
  //#upc = número de equações de verificação de paridade
  //nãosatisfeitas
    end for
   end if
  end for
  Maxupc ← o maior #upc
  fori = 1 **to**n**do**
   ifcontador[i] ≥ b_i**then**
    Flipc_i
    $s \leftarrow s$**XOR**h_j
   end if
  end for
  if$s$ = 0 **then**
   return c
  endif
  δ ← δ – 1 //No caso de falha na decodificação
endwhile
returnFALHA

IV. IMPLEMENTAÇÃO

A proposta do trabalho é a construção de uma implementação eficiente do criptossistemaMcEliece baseado em códigos QC-MDPC.

O desenvolvimento do criptossistema foi feito em linguagem C na plataforma Intel e desconsiderou qualquer limitação de memória ou capacidade de processamento e armazenamento do hardware.

Para sua realização, teve como ponto de partida a proposta [3] com a construção do codificador e do decodificador lá descritos, adotando os seguintes parâmetros apresentados no artigo (em bits), para um nível de segurança de 80 bits:
$n_0 = 2$, $n = 9600$, $r = 4800$, $w = 90$, $t = 84$, tamanho da chave = 4800.

Portanto, uma mensagem de 4800 bits (r) é codificada em uma mensagem cifrada de 9600 (n) bits à qual 84 bits de erro (t) são adicionados. A matriz de paridade H possui linhas com peso constante 90 (w) e consiste de 2 blocos H_0 e H_1 (n_0).

Após a construção inicial do criptossistema e a tomada de tempo de execução, o esforço todo passou a ser direcionado para a otimização e melhora do desempenho da rotina de decodificação pois, como mencionado, esta etapa é a maior consumidora de tempo de processamento.

É importante ressaltar que tanto na codificação quanto na decodificação a aleatoriedade para a geração dos vetores binários (vetor h e vetor$e$) foi simulada, uma vez que a utilização de um gerador de números aleatórios verdadeiro está fora do escopo deste trabalho.

A seguir encontra-se detalhado o trabalho de implementação:

Geração das chaves: como não havia restrição de projeto, os vetores e matrizes binários, mesmo sendo esparsos, ficaram todos armazenados integralmente em memória durante a execução do programa, o que possibilitou ganho em algumas operações tais como as atualizações de contadores na decodificação, por exemplo. Para a manipulação dessas estruturas de dados foi utilizada a biblioteca numérica de domínio público para representação de vetores e matrizes *Meschach* [7], a qual provê os tipos básicos de dados matriz e vetor bem como algumas operações primárias tais como a cópia de blocos de dados entre ambos. A biblioteca foi devidamente customizada para que os tipos de dados armazenados pelas matrizes e vetores binários fossem representados como números inteiros de 32 bits sem sinal, os quais foram substituídos por números inteiros de 64 bits sem sinal numa segunda versão.

Durante a tarefa de geração das chaves do criptossistema existe a necessidade de inversão de matrizes binárias. Para a codificação desta tarefa foi utilizado o trabalho de Jasinskiet al. [8], o qual traz um algoritmo eficiente que implementa uma versão melhorada do método de Gauss-Jordan para a inversão de matrizes binárias.

Codificador: para a construção do codificador, com relação a aspectos de desempenho, o maior cuidado tomado foi com a representação de conjuntos de bits por números inteiros sem sinal de 32 bits inicialmente e 64 bits numa segunda etapa. Desta maneira os bits puderam ser processados em blocos, o que garante melhora no desempenho de tarefas que necessitem ser aplicadas em linhas de matrizes ou vetores, tais como a multiplicação de matrizes por matrizes ou de vetores por matrizes, nas quais são executadas operações lógicas (XOR's e AND's) nas linhas binárias.

Decodificador: assim como na geração de chaves e na codificação, as matrizes e vetores foram definidos como elementos de tipos de dados inteiros sem sinal de 32 bits e 64 bits numa segunda etapa. Na decodificação procurou-se fazer uso ao máximo de recursos e técnicas que permitissem a economia no número de operações executadas pelo processador e a consequente melhora no desempenho do processamento.

Como mostrado, numa tarefa de decodificação existe o cálculo inicial da síndrome, que corresponde à multiplicação da matriz de paridade H pelo vetor transposto que representa a mensagem codificada c. Medindo-se o tempo de processamento de todos os procedimentos que compõem a rotina de decodificação, percebe-se que esta multiplicação corresponde à

etapa de maior custo no algoritmo. Um truque bastante simples e eficiente e que reduz em muito o tempo gasto para se decodificar uma mensagem é, na computação da síndrome, calcular o produto do vetor transposto pela matriz transposta. E na realização deste cálculo percorre-se cada elemento individual binário do vetor e as posições dos bits não nulos selecionam as linhas da matriz transposta às quais deve ser aplicado o XOR. O resultado desta operação é a síndrome. Como o vetor mensagem codificada tem cerca de apenas um terço de suas posições diferentes de zero, isto significa que somente aproximadamente um terço dos XOR's das linhas da matriz H precisa efetivamente ser computado para se chegar ao valor da síndrome. Este truque foi implementado e mostrou-se bastante útil devido à configuração da matriz e do vetor que deviam ser multiplicados.

No algoritmo da decodificação, após o cálculo da síndrome há um passo no qual é executado o cálculo dos contadores que registram a quantidade de equações de checagem de paridade não satisfeitas associadas a cada bit da mensagem. Esta contagem é repetida a cada atualização da síndrome e, portanto, foi testada alternativa na qual os contadores eram apenas atualizados em vez de zerados e recalculados. Nos testes armazenou-se a síndrome anterior e comparou-se a mesma com a síndrome atual e apenas os contadores relacionados às posições dos bits que mudaram entre ambas as versões da síndrome eram incrementados ou decrementados, dependendo da mudança do bit de zero para um ou de um para zero. A medição de tempo dos testes descartou esta solução alternativa ao mostrar que a recontagem de todos os contadores da maneira como está proposto no algoritmo é a estratégia mais rápida.

Uma outra otimização que representou impacto positivo no tempo de processamento foi a utilização de instruções vetoriais intrínsecas nas operações de XOR que existem tanto na multiplicação matriz por vetor quanto no processamento da síndrome durante a decodificação.

Instruções intrínsecas são aquelas que o compilador mapeia em uma sequência de uma ou mais instruções em linguagem de máquina com o grande benefício de possibilitar o acesso a novos registradores específicos do hardware assim como a novos tipos de dados que não estão disponíveis através dos métodos padrão das linguagens de programação.

As funções intrínsecas são inerentemente mais rápidas que as convencionais providas pela linguagem, O código para uma função intrínseca geralmente é inserido em linha, evitando a sobrecarga de uma chamada de função e permitindo instruções de máquina altamente eficientes, pois o compilador tem o conhecimento prévio e específico de como as instruções intrínsecas se comportam.

Mediante o uso de instruções vetoriais intrínsecas – instruções SIMD (*single instructionmultiple data*) - é possível a paralelização do código com a realização de operações lógicas ou matemáticas em múltiplos pares de operandos simultaneamente.

Para o processamento das matrizes e vetores foram utilizadas instruções vetoriais intrínsecas Intel do conjunto de extensões AVX e AVX2. Com o conjunto de instruções AVX é possível o acesso a registradores de 256 bits da família de processadores Intel com a microarquitetura *Sandy Bridge*. O conjunto de instruções AVX2 torna disponíveis registradores de 512 bits da família de processadores com microarquitetura *Haswell*. O ganho de tempo resultante do uso das instruções vetoriais nestes casos deve-se à paralelização: como afirmado, após serem carregados conjuntos de elementos de 32 ou 64 bits múltiplos inteiros do tamanho dos registradores, é executado um XOR simultâneo em vários pares de operandos.

Uma última otimização diz respeito à redução no número de operações de cálculo de síndromes, que correspondem a multiplicações de vetores mensagem por matrizes de paridade, o que, como já foi dito, é o maior gargalo de desempenho no algoritmo de decodificação. Comparando os Algoritmos 1 e 2 apresentados observa-se que em 2 a síndrome é calculada apenas uma vez e a partir de então ela passa a ser atualizada apenas através de XOR's. As duas propostas foram implementadas e observou-se ganho significativo no tempo de processamento da rotina de decodificação que faz economia na quantidade de multiplicações realizadas. Há que ser ressaltado que o tempo de execução é bastante melhorado com a redução do número de multiplicações necessariamente em conjunto com a utilização dos limiares $b_i$pré-calculados em cima dos parâmetros do código. Estes limiares diminuem significativamente a quantidade de iterações executadas pela rotina de decodificação.

V. RESULTADOS

Os resultados apresentados abaixo foram obtidos a partir de execuções das várias versões implementadas em linguagem C do criptossistema de McEliece baseado em códigos QC-MDPC. Para tais foi utilizada uma estação de trabalho com CPU Intel Core I7-4770 trabalhando na frequência de 3,40 GHz e rodando sistema operacional Linux Ubuntu12.04 LTS.

Em cada versão implementada do criptossistema, os tempos medidos foram calculados a partir da média dos tempos de execução de mil decodificações de uma mesma mensagem cifrada. Procedeu-se da mesma maneira para a obtenção do número médio de iterações para cada decodificação.

Inicialmente foi construída uma versão do criptossistema que corresponde literalmente ao que está proposto em [3]. Não foi utilizada nenhuma estratégia de otimização do código fonte, apenas alterou-se o tipo de dados dos elementos armazenados por vetores e matrizes para inteiros de 32 bits sem sinal e 64 bits sem sinal e mediu-se os tempos respectivos, apresentados na Tabela I. A ideia é estabelecer um marco inicial comparativo e indicativo da evolução dos resultados do trabalho. No artigo os autores mencionam que o tempo de decodificação por eles alcançado é de cerca de 3ms.

TABELA I. VERSÃO ORIGINAL DO CRIPTOSSISTEMA PROPOSTO EM [3].

Tipo de dados (vetores/matrizes)	Tempo de decodificação (ms)
Inteiros de 32 bits sem sinal	8.230
Inteiros de 64 bits sem sinal	6.740

A seguir o trabalho concentrou-se na construção e teste de versões alternativas otimizadas. O passo inicial da

decodificação corresponde ao cálculo da síndrome, que nada mais é que o resultado do produto da mensagem codificada pela matriz de paridade. Analisando-se a performance da rotina de decodificação, observa-se que esta multiplicação é uma etapa de alto custo de processamento. Portanto, uma primeira otimização significou a aplicação de um truque na rotina de multiplicação de matriz por vetor, a qual passou a selecionar para a operação de XOR apenas as linhas da matriz transposta associada cujas posições correspondem a valores não nulos do vetor. Como o vetor possui apenas um terço de seus elementos diferentes de zero, esta estratégia economizou dois terços de operações XOR desnecessários para a obtenção da síndrome.

Além da redução do número de XOR's para o cálculo da síndrome, o uso das instruções vetoriais intrínsecas na rotina de multiplicação possibilitou o paralelismo com a execução de XOR's simultâneos em múltiplos pares de operandos carregados nos registradores específicos de 256 bits (AVX) e 512 bits (AVX2).

Todas estas ações refletem-se nos tempos mostrados na Tabela II, onde a decodificação passou a acontecer em menos de 3ms.

TABELA II. VERSÃO OTIMIZADA 1 DO CRIPTOSSISTEMA QC-MPDC MCELIECE.

Instruções vetoriais	Tempo de decodificação (ms)
AVX	2.595
AVX2	2.362

Conforme afirmado, a multiplicação para o cálculo da síndrome é a etapa que mais demanda esforço computacional na tarefa de decodificação de uma mensagem cifrada. No Algoritmo 1 proposto em [3] esta multiplicação é executada em cada uma das iterações que fazem parte das tentativas de decifração da mensagem.

O Algoritmo 2 apresentado em [4] elimina esses cálculos repetidos da síndrome, assim como reduz a quantidade de iterações necessárias para a decodificação da mensagem. Uma versão alternativa do decodificador foi implementada contemplando o Algoritmo 2 e os tempos medidos, como esperado, espelharam a melhora significativa da decodificação como resultado da eliminação das multiplicações sucessivas do vetor mensagem cifrada pela matriz de paridade e a substituição desta tarefa pela simples atualização da síndrome calculada no passo inicial da decodificação, assim como a redução do número de iterações do algoritmo com a substituição dos seus limiares por valores pré-calculados baseados em parâmetros do código.

A Tabela III mostra que esta ação de substituição do algoritmo de decodificação original contemplou o objetivo do trabalho proposto com a produção de uma implementação eficiente do criptossistemaMcEliece baseado em códigos QC-MDPC.

TABELA III. COMPARAÇÃO ENTRE VERSÕES ALTERNATIVAS DO CRIPTOSSISTEMA QC-MPDC MCELIECE

Instruções vetoriais	Tempo de decodificação (ms)	Algoritmo	Iterações
AVX	2.595	1	5
AVX	0.945	2	2
AVX2	2.362	1	5
AVX2	1.173	2	2

VI. CONCLUSÃO

Neste trabalho foi construída uma versão otimizada eficiente de um criptossistema de McEliece baseado em códigos QC-MDPC. Este criptossistema acena como um possível substituto dos criptossistemas assimétricos populares imune ao computador quântico. Na sua concepção original o criptossistema de McEliece baseado em códigos de Goppa apresentava como inconveniente o tamanho grande das suas chaves. A proposta [3] apresenta uma versão na qual níveis de segurança atuais são alcançados com chaves de dimensões compatíveis com as dos criptossistemas de chave pública atualmente difundidos. A implementação de uma versão com tempos de processamento eficientes é mais uma evidência que reforça estes criptossistemas como candidatos a alternativas viáveis caso a computação quântica se estabeleça além dos modelos teóricos e dos laboratórios.

REFERÊNCIAS

[1] D. J. Bernstein, J. Buchmann, and E. Dahmen, editors. *Post-Quantum Cryptography*. Springer-Verlag, 2009.

[2] R. J. McEliece, *"A Public-Key Cryptosystem Based On Algebraic Coding Theory"*, Deep Space Network Progress Report, vol. 44, pp. 114–116, Jan. 1978.

[3] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. L. M. Barreto, *"MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes"*, IEEE International Symposium on Information Theory, vol. 2013, 2013.

[4] I. von Maurich, T. Güneysu,*"Lightweight Code-based Cryptography: QC-MDPC McEliece Encryption on Reconfigurable Devices"*, Procedings of the IEE Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014, pp. 1-6.

[5] S. Heyse, I. von Maurich, and T. Güneysu, *"Smaller Keys for Code-Based Cryptography: QC-MDPC McEliece Implementations on Embedded Devices"*, in CHES, ser. Lecture Notes in Computer Science, G. Bertoni and J.-S. Coron, Eds., vol. 8086. Springer, 2013, pp. 273– 292.

[6] R. Gallager, *"Low-density Parity-check Codes"*, Information Theory, IRE Transactions on, vol. 8, no. 1, pp. 21–28, 1962.

[7] Meschach: Matrix computations in C. Disponível em: <http://homepage.math.uiowa.edu/~dstewart/meschach/html_manual/manual.html> Acesso em 20 de maio de 2014.

[8] R. P. Jasinski, V. A. Pedroni, A. Gortan, W. Godoy Jr, *"An Improved GF(2) Matrix Inverter with Linear Time Complexity"*, Procedings of the IEE International Conference on Reconfigurable Computing and FPGAs (ReConFig), 2010, pp. 322-327.

[9] Intel IntrinsicsGuide. Disponível em:<https:t//software.intel.com/sites/landingpage/IntrinsicsGuide/> Acesso em 20 de maio de 2014.

[10] W. C. Huffman and V. Pless, *"Fundamentals of Error-Correcting Codes"*, 2010.

Homero de Oliveira Martins é graduado em Engenharia Elétrica pela Universidade de Brasília (UnB), Brasília, Brasil, em 1997. Recebeu também o título de Bacharel em Ciência da Computação pela Universidade de Brasília (UnB), Brasília, Brasil, em 1993. Atualmente é aluno de Mestrado em Engenharia Elétrica, na área de redes de comunicação, da Universidade de Brasília (UnB). Suas principais áreas de pesquisas são: Criptografia e Segurança da Informação. Trabalha no Centro de Informática da Câmara dos Deputados desde 1999 na Coordenação de Engenharia de Sistemas.

Anderson A. C. Nascimento é atualmente professor do departamento de Engenharia Elétrica da Universidade de Brasília (UnB), Brasília, Brasil. Recebeu o título de Doutor em *Information and Communication Engineering* da Universityof Tokyo (U.T.), Japão em 2004; Mestre em *Information and Communication Engineering* da Universityof Tokyo (U.T.), Japão em 2001; e graduado em Engenharia Elétrica pela Universidade de Brasília (UnB), Brasília, Brasil, em 1998. Suas principais áreas de pesquisas são: computação segura de duas partes e multipartes; teoria quântica da informação; segurança demonstrável e criptografia baseada em códigos.