Compressão e Otimização de Chaves Públicas usando Algoritmo Genético em Criptografia Completamente Homomórfica

Joffre Gavinho Filho[1], Gabriel Pereira da Silva[1], Claudio Miceli[1]

[1]Instituto Técio Pacitti – NCE/IM – Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 2324 – 20.010-974 – Rio de Janeiro – RJ – Brasil

{joffreufrj, gabriel.silva, cmicelifarias}@gmail.com

Abstract. *The Homomorphic Encryption is an encryption technique for processing encrypted data without the need to decrypt them. This method is suitable for use in untrusted environments, such as cloud computing platforms. Various methods have been proposed to implement this technique. However, the greatest problem of these methods is that for there operation, there is the need to generate public keys with large sizes, whose immediate consequence is to cause such encryption schemes not reach the desired runtime performance. This article aims to optimize techniques for reducing public keys using of Genetic Algorithms (GA) for calibration of the parameters of the primitive of the Coron test variants and Bilar.*

Resumo. *A Criptografia Homomórfica é uma técnica de criptografia para o processamento de dados criptografados sem a necessidade de decifra-los. Tal método é indicado para uso em ambientes não confiáveis, como por exemplo as plataformas de computação em nuvem. Vários métodos têm sido propostos para a implementação dessa técnica. Porém, o grande problema destes métodos é que, para a sua operacionalização, há a necessidade da geração de chaves públicas com tamanhos excessivanente grandes. Este artigo tem como objetivo implementar e otimizar as técnicas de redução de chaves públicas usando Algoritmos Genéticos (AG) para a calibração dos parâmetros das primitivas do regime das variantes de testes de Coron e de Bilar.*

1. Introdução

Os avanços tecnológicos têm impulsionado o intercâmbio cada vez maior de informações e um grande desenvolvimento em todas as áreas de atuação. Entretanto, dependendendo do tipo de atividade exercida, bem como a estrutura necessária para a realização de tais atividades, o custo financeiro para sua realização pode mostrar-se bastante proibitivo. Uma das soluções encontradas para a redução dos custos e da mitigação na montagem de determinadas infraestruturas especializadas para atender as demandas informatizadas é a plataforma de Computação em Nuvem (*Cloud Computing*) [Sousa 2009]. Essa plataforma tem como premissa a utilização de uma estrutura com capacidades de armazenamento e processamento compartilhados e interligados por meio da Internet [Sousa 2009]. Uma das áreas extremamente sensíveis e que também faz parte das prioridades nos projetos de computação em nuvem, é a provisão de segurança da informação, sendo a Criptografia um dos métodos mais utilizados nesse tipo de

planejamento e execução. As técnicas criptográficas [Buchmann 2002] não são novas e, mesmo em sua versão computacional tem sido utilizadas a bastante tempo. No entanto, os avanços tecnológicos, tais como o aumento das capacidades de armazenamento e processamento dos computadores, ameaçam a segurança dos conhecidos algoritmos criptográficos, determinando com isso, constantes estudos sobre o tema, a fim de não tornar obsoletos os meios de proteção criptográficos da informação [NIST 2013].

Existem dois modelos básicos de criptografia computacional [Stalling 2007]: os simétricos e os assimétricos. A diferença fundamental entre estes é que os modelos simétricos têm apenas uma chave, utilizada tanto para codificação quanto para a decodificação da informação. Enquanto o modelo assimétrico usa duas chaves, uma pública, usada para criptografar os dados, e uma chave secreta, usada para descriptografar os blocos criptografados. Os dois modelos são usualmente utilizados no fornecimento de segurança no processamento e no armazenamento de dados usando a plataforma de computação em nuvem. No entanto, ao utilizar-se os métodos convencionais de criptografia em tais plataformas, os dados tornam-se sensivelmentes vulneráveis, pois, para: o processamento, o armazenamento e a manipulação das informações, há a necessidade de se descriptografar os dados, pois os servidores que compoem a nuvem, não tem como validar ou mesmo comparar os dados que estão codificados, tendo-se então a necessidade de acesso direto aos dados originais. Em suma, o principal problema com ambiente compartilhado e a criptografia, reside no processamento dos dados, porque na criptografia tradicional, os dados não podem ser alterados enquanto criptografados. Este conceito é chamado de "não-maleabilidade" [Coron 2011] e requer que todos os dados devam ser desencriptados antes de serem processados, mesmo em ambiente compartilhado.

Um dos cenários onde este conceito de “não-maleabilidade” [Coron 2011], pode tornar-se extremamente sensível frente a provisão de segurança é o ambiente de computação em nuvem. Pois, tal modelo de serviço apresenta diferentes níveis de riscos se comparado ao ambiente tradicional de tecnologia de informação [CSA 2009]. O provimento de recursos sob demanda para o processamento e armazenamento massivo de dados está sujeito a: falhas de segurança, abusos com relação à privacidade, violação de direitos autorais, etc. Tais problemas são observados, principalmente, nos servidores que compõem a nuvem, pois necessariamente os dados, que são acessados por eles, têm de ser desencriptados para serem manipulados.

Este tipo de vulnerabilidade não é observado em sistemas homomórficos, porque os dados são manipulados em sua forma criptografada, sem a necessidade de acessar a informação em texto claro, ou seja, o texto não encriptado. O esquema de encriptação homomórfico (EH) é baseado em processamento aditivo e multiplicativo de funções, isto é, além dos convencionais algoritmos de encriptação e desencriptação [Stalling 2007], há também um algoritmo de avaliação, que toma como entrada uma mensagem encriptada $f(m)$ e retorna uma criptografia de $f(m)$ [Gentry 2009]. Esquemas EH podem ser basicamente classificados em dois tipos: o primeiro são os esquemas EH, chamados de parcialmente homomórficos, que, além de operações de criptografia e decriptografia também executam operações de soma ou multiplicação, que tomam como entrada mensagens criptografadas m_1 e m_2 e regressam a criptografia de $m_1 + m_2$ ou $m_1 * m_2$, respectivamente. Se um esquema EH suporta tanto adição quanto multiplicação, ele também pode avaliar qualquer circuito aritmético em dados criptografados [Gentry

2009] e, portanto, podemos dizer que é um esquema de Criptografia Completamente Homomórfica (CCH), ou seja, se o $E(m)$ é a codificação de uma mensagem m, um modelo de criptografia é totalmente homomórfica se: $f(m) \rightarrow E(m_1 + m_2) = E(m_1) + E(m_2)$; e, $f(m) \rightarrow E(m_1 * m_2) = E(m_1) * E(m_2)$ [Coron 2011]. Usando esse tipo de regime, qualquer circuito pode receber uma avaliação homomórfica, permitindo a construção de programas que podem ser executados com as criptografias de suas entradas para produzir uma saída criptografada. Como esses programas não decodificam as informações, eles podem ser utilizados por terceiros não confiáveis sem revelar a sua entrada e estado interno. Por exemplo, pode-se somar dois números criptografados e, a menos que se possa descriptografar o resultado, não há como se descobrir o valor dos números originais individualmente [Gentry 2009].

O grande problema dos métodos propostos para os sistemas completamente homomórficos é que os seus tempos de execução, bem como o tamanho dos parâmetros utilizados, especialmente os das chaves públicas, crescem ciclicamente a cada iteração em complexidade na ordem de $O(\lambda^{10})$ [Coron 2011]. Onde λ é o parâmetro de segurança de todo o regime, e que define o tamanho, em bits de comprimento, das chaves geradas. Melhorias dos processos estão sendo propostas. Como é o caso [Coron 2011] que propôs um esquema de chaves públicas DGHV [DHGV 2010] totalmente homomórfico, e que reduz o tamanho dessas chaves geradas para aproximadamente $O(\lambda^{7})$ [Coron 2011]. Bem como em [Bilar 2014], que otimiza os testes de Coron de modo a reduzir os tempos de execução do esquema.

Este trabalho visa aplicar a heurísticas de Algoritmo Genético (AG) para reduzir os parâmetros utilizados no método de [Coron 2011], em conjunto com as implementações e testes feitos por [Bilar 2014] em tal método [Coron 2011]. Tendo como meta otimizar o desempenho em relação aos tempos de execução das técnicas de compressão. Para tal utilizamos o conceito de Algoritmo Genético (AG) [Lacerda 1999], que consiste em um método heurístico, que tem como objetivo encontrar a solução que corresponda ao ponto de máximo e/ou de mínimo de uma determinada função, sendo extremamente adequada ao propósito deste estudo.

Este artigo está organizado da seguinte forma: na Seção 2 são descritos os mecanismos de Criptografia Completamente Homomórfica e os métodos de compactação de chaves públicas, bem como os trabalhos relacionados; na Seção 3, a proposta de Compactação e Otimização é apresentada, bem como os experimentos realizados e as análises dos resultados obtidos; e finalmente, na Seção 4 são tecidas as conclusões finais e as propostas de trabalhos futuros.

2. Trabalhos Relacionados

Nesta seção, descrevemos um breve histórico da criação e desenvolvimento das técnicas homomórficas, bem como os métodos e trabalhos propostos para a redução de chaves no processo de Criptografia Completamente Homomórfica.

2.1. Criptografia Completamente Homomórfica (CCH)

A necessidade de um algoritmo de criptografia que permitisse a computação arbitrária em dados criptografados foi reconhecido muito cedo por criptógrafos. Em 1978, Rivest, Adleman e Dertouzos [RDA 1978], sugeriram a construção de homomorfismos secretos

- *privacy homomorphisms* [RDA 1978] - como uma forma de proporcionar um mecanismo de proteção para a computação em dados sensíveis. No entanto, o esquema, além de ser parcialmente homomófico, i.e., usar apenas a propriedade multiplicativa do método, ele não oferece proteção contra ataques de texto simples escolhido (*Chosen Plaintext Attack - CPA*) [Morris 1993], isto é, não possue segurança semântica. Sendo que, em criptografia, um esquema é considerado semanticamente seguro se: dado um texto cifrado de qualquer mensagem, e o tamanho dessa mensagem, não há nenhum algoritmo probabilístico em tempo polinomial (*Probabilistic Polynomial-Time Algorithm - PPTA*), [Michael 1980], que possa determinar qualquer informação da mensagem com significado maior do que a probabilidade de uma escolha aleatória. Em outras palavras, o conhecimento da mensagem cifrada, e o tamanho da mensagem original, não revela nenhuma informação sobre esta mensagem, bem como, nenhum dado pode ser extraído facilmente a partir do texto cifrado.

Após a construção de homomorfismos secretos - *privacy homomorphisms* [RDA 1978], a comunidade científica começou a procurar implementações práticas desta teoria, isto é, algoritmos capazes de executar a chamada criptografia homomórfica. Apesar de muitas tentativas, o problema permaneceu sem solução até recentemente, quando, em 2009, Craig Gentry [Gentry 2009] o solucionou sugerindo a utilização de reticulados ideais na construção de um sistema de encriptação totalmente homomórfico. Infelizmente, devido à complexidade das avaliações de multiplicações e o tamanho excessivamente grande das chaves públicas geradas no processo, a proposta de Craig Gentry não foi suficientemente eficaz para ser usado na prática.

Também em 2009, NP Smart e F. Vercauteren [Smart 2009] lançaram um sistema de encriptação totalmente homomórfica, sua influência principal foi o esquema de Gentry. O sistema proposto gerava blocos criptografados com pequenos erros, que se propagavam à medida que as operações homomórficas eram executadas. O problema com este sistema, é que, depois de um certo número de operações, os blocos de erros tornavam-se muito grandes, o que, em determinado momento, inviábilizava o processo de decriptografia da mensagem.

O primeiro desafio em criptografia homomórfica é torna-la eficientemente pratica. Enquanto a construção original de [Gentry 2009] é vista como impraticável, as últimas construções e os esforços de implementação melhoraram dramaticamente a eficiência da criptografia completamente homomórfica. Os esforços de implementação iniciais, com foco na proposta original de Gentry e suas variações [Gentry 2010] melhoraram os gargalos de eficiência. Implementações posteriores fizeram uso de avanços recentes nos algoritmos [DHGV 2010] e das técnicas algébricas [Brakerski 2012] procurando melhorar a eficiência concreta dos resultados em tais esquemas O tamanho de chaves públicas criadas por esses métodos, bem como a necessidade de uma estrutura de armazenamento é um dos maiores desafios na criptografia homomórfica. Em Dijk, Gentry, Halevi e Vaikuntanathan (DGHV) [DHGV 2010], por exemplo, o tamanho da chave pública se encontra no intervalo de λ^{10} bits e o tamanho da chave privada cerca de λ^2, onde λ é o parâmetro segurança dos esquemas. O tamanho da chave privada é necessário para garantir a segurança do regime. Coron, Mandal e Naccache [Coron 2012] propuseram uma modificação na geração de chaves, a fim de reduzir o tamanho da chave pública para λ^7. A modificação consiste na utilização de formas quadráticas dos elementos da chave pública, em vez de formas lineares, como é feito no

esquema DGHV. A idéia de Coron [Coron 2011] é armazenar apenas um pequeno subconjunto da chave pública e, quando necessário, gerar a chave pública completa multiplicativamente combinanda aos elementos do subconjunto das chaves secretas. Esta proposta mantém a segurança semântica, e se baseia no problema do máximo divisor comum aproximado parcial (*Partial Approximate Greatest Common Divisor*) [Coron 2011], que consiste em retirar o erro dos primeiros conjuntos de chaves públicas criadas durante o processo. Este esquema tem o mesmo problema básico de segurança do máximo divisor comum aproximado (*Approximate Greatest Common Divisor*) [DGHV 2010], isto é, o problema da Segurança Circular [Boneh 2008], que, dado um esquema de criptografia *E*, dizemos que *E* tem segurança circular se for seguro criptografar a chave privada com sua própria chave pública, esquema este que é a base do regime de segurança do DGHV, descrito a seguir.

2.1.1. O Esquema DGHV

Em 2010, Dijk, Gentry, Halevi e Vaikuntanathan propuseram um esquema totalmente homomórfico (DGHV) [DHGV 2010] usando apenas álgebra modular ao longo de um anel de inteiros, o que provou ter uma menor complexidade matemática quando comparada com os regimes baseados em reticulados ideias de Gentry [Gentry 2009]. Este mesmo padrão foi analisado por Coron, que propôs duas variantes [Coron 2012] e [Coron 2011] da mesma proposta, conseguindo a otimização do custo computacional de certas primitivas, e a diminuição do tamanho do esquema de chaves públicas originais utilizando várias técnicas de redução e de compressão de chaves. O esquema [DGHV 2010], *E*(*KeyGen*, *Encrypt*, *Decript, Recrypt, Evaluate*), consiste em cinco algoritmos, também chamados de primitivas. A primitiva *Keygen* é o esquema responsável por gerar o par de chaves (públicas e privadas); *Encrypt* é responsável por gerar o texto cifrado; *Decript* é responsável por decifrar o texto encriptado; *Recrypt* é utilizada para a criptografia da criptografia, isto é, a criptografia em níveis; e, *Evaluatel* que realiza o processamento em tuplas de bits codificados em um circuito lógico, e que retorna o equivalente criptografado a este circuito como se fosse aplicado aos dados originais, isto é, faz a manipulação homomórfica nos bits codificados, sem a necessidade de descriptografa-los.

2.1.2. O Método de Coron.

A primeira variante de Coron [Coron 2011], chamado DGHV com chave reduzida, inclui a adição de novos parâmetros quadráticos ao esquema de primitivas, armazenando assim apenas um pequeno conjunto de valores relacionados com a chave pública e, em seguida, gerando uma chave pública completa em tempo de execução. Usando esta técnica, Coron demonstrou a redução do tamanho da chave pública da ordem de $O(\lambda^{10})$ para a de $O(\lambda^{7})$, [Coron 2011].

No trabalho intitulado "*Fully homomorphic encryption over the integers with shorter public key*s" [Coron 2012], o comprimento da chave pública foi ainda mais reduzida, indo de $O(\lambda^{7})$ para $O(\lambda^{5})$. A principal inovação proposta por Coron deste esquema é que, em vez de armazenar todos os elementos-chave da criptografia, ele só armazena o valor de correção em relação a um gerador de números aleatórios. Assim, os dados a serem armazenados são menores, e, havendo necessidade dos dados, estes são recuperados "*on-the-fly*" pelas primitivas *Encryipt*, *Recrypt*, *Descript* e *Expand*. Além disso, é descrita uma técnica de troca de módulos, que permite este regime, sem usar a

estrutura proposta de *boostraping* descrito por Brakerski, Gentry e Vaikuntanathan, (BGV) [Brakerski 2012].

O regime inicial de inteiros usados por Coron como base para o seu trabalho, bem como para a criação de sua segunda variante, é fundamentada no trabalho de Gentry [Gentry 2009]. Este definiu o DGHV com base em um conjunto de inteiros, $x_i = p.q_i + r_i$, p $0 \leq i \leq \tau$, sendo λ o parâmetro de segurança. Os seguintes parâmetros devem ser utilizados para compor o esquema de Criptografia Homomórfica Reduzida (CHR) [Brakerski 2012], que, em seguida, deve ser reforçada para gerar o CCH de inteiros [Bilar 2014]:

• γ é o comprimento em bits de x_i.
• η é o comprimento em bits da chave secreta p.
• ρ é o comprimento em bits do ruído r_i.
• τ é o número da chave pública de x_i.
• ρ'é um parâmetro ruído secundário usado para criptografar, [Coron 2011].

O Esquema é definido com as seguintes restrições:

• $\rho = \omega\ (log\lambda)$, necessário para a proteção contra ataques de força bruta.
• $\eta \geq \rho.\Theta\ (\lambda log 2\lambda)$, necessário para a execução de operações homomórficos de avaliação.
• $\gamma = \omega\ (\eta 2.log\lambda)$, necessário para a proteção de ataques baseados no problema do MDC.
• $\tau \geq \gamma + \omega\ (log\lambda)$, necessário para reduzir a abordagem do MDC Aproximado.
• $\rho' = \rho + \omega\ (log\lambda)$, necessário para o parâmetro de ruído secundário.

2.1.3. A Implementação e Otimização de Bilar.

Em sua proposta, [Bilar 2014], implementou o esquema totalmente homomórfico de chave reduzida (DGHV sobre inteiros) proposto em [Coron 2011] que pode ser comparado com o esquema totalmente homomórfico de [Gentry 2009], e que se trata de um esquema totalmente homomórfico de construção mais simples. Contudo, essa simplicidade vem ao custo de que sua chave pública possui um tamanho estimado em $O(\lambda^{10})$, o que de acordo com [Coron 2011], torna inviável a aplicação em sistemas práticos. O esquema totalmente homomórfico DGHV com chave pública reduzida diminui o tamanho da chave pública gerada para $O(\lambda^7)$, criptografando de maneira quadrática os elementos da chave pública, ao invés de criptografá-los de maneira linear. Bilar utilizou a linguagem de programação Python [Python 2014], contando com a biblioteca de matemática e teoria numérica GMPY2 [Bilar 2014]. Bilar demonstrou uma pequena, porém significativa, otimização em termos de tempo de execução das primitivas de Coron [Bilar 2014].

2.1.4. O Algoritmo Genético.

Segundo [Lacerda 1999], Algoritmos Genéticos (AG) são métodos de otimização e busca, inspirados nos mecanismos de evolução de populações de organismos. São implementados como uma simulação em computador em que uma população de representações abstratas é selecionada na busca das melhores soluções. A evolução geralmente começa a partir de um conjunto de soluções criadas aleatoriamente e é realizado através de gerações. Em cada geração, a adaptação de cada solução na população é avaliada, alguns indivíduos são selecionados para a próxima geração e

recombinados ou mutados de modo a formar uma nova população. A nova população é então utilizada como entrada para a próxima iteração do algoritmo.

Entre os vários usos de tais algoritmos, podemos aplica-lo para atribuir pesos a conjuntos matemáticos, cuja otimização é encontrar a solução que corresponda ao ponto de máximo ou de mínimo para uma determinada função. Considerando, por exemplo, uma função $f(x)$ que consiste em k elementos que devem ser maximizados. A cada um dos elementos x_k é atribuído um peso j através da criação de vetores de cromossomos definidos como na Equação 1:

$$f(x) = j(x_1) + ... + j(x_k) \quad (1)$$

Cada cromossomo possui j posições, um para cada conjunto de k elementos. Cada posição [j, k] contém um número real no intervalo [0,1] escolhido aleatoriamente, que representa o número de elementos chamados de genes. Os cromossomos j são estabelecidos aleatoriamente no início do processo de atribuição de pesos. Em seguida, é criada a primeira geração da população que vai ser usado nesta fase. Cada cromossomo será processado um por um para avaliar o seu desempenho. O processamento é realizado da seguinte forma: O cromossomo j ($gene_1$, $gene_2$, ..., $gene_k$) é analisado em sua evolução pela fórmula do algoritmo genético do cromossomo (*fitness*), Equação 2 [Lacerda 1999].

$$fitness = \frac{Total - FP - 2 * FN}{Total} \quad (2)$$

Onde: *Total* = Total de dados analisados, *FP* = falsos positivos e *FN* = negativos falsos.

Depois de calcular a aptidão dos cromossomos j da primeira geração da população, tem-se início o processo evolutivo do algoritmo genético. A evolução da população é realizada através de seleção, cruzamento e mutação de cromossomos. O mais amplamente usado no método na fase de seleção é o método de roleta [Lacerda 1999]. No método de roleta, cada cromossomo é representado proporcionalmente à sua aptidão em comparação com a soma das competências, Equação 2, de todos os cromossomos da população. Um valor aleatório é gerado e o cromossomo correspondente na roleta é selecionado para gerar "prole". O número de cromossomos selecionados é igual ao tamanho original população. O método é formalizado como se segue: (*i*) a aptidão de todos os cromossomos é avaliada e somada (T*f*); (*ii*) há a geração de um número aleatório n: 0 <= n <= T*f*; (*iii*) é selecionado o cromossomo cuja aptidão adicionada às aptidões dos cromossomos anteriores é igual ou maior que n. Após a distribuição, é calculada a soma das aptidões (T*f*), um número aleatório é criado e há a escolha do cromossomo. O método da roleta [Lacerda 1999] é usado para selecionar dois cromossomos pai. Após essa fase se inicia a fase de mutação. A mutação consiste basicamente na mistura do material genético de dois indivíduos (pais) da população, produzindo dois novos indivíduos (filhos) que herdam características de seus pais. É utilizado o cruzamento em dois pontos (*two-point crossover*) [Lacerda 1999]. Ou seja, são definidos aleatoriamente dois pontos de corte nos cromossomos selecionados na fase de seleção. Um dos descendentes obtém a parte central de um dos pais e as partes

extremas do outro progenitor. Os filhos, então, substituem as posições ocupadas pelos pais. A operação de mutação impede a convergência prematura do algoritmo através da introdução de novas regiões em busca de espaço de solução. Este consiste de valores aleatórios para substituir alguns genes dos cromossomos. Usa-se a margem de Y% da população para efetuar a mutação de um dos cromossomos filhos. Um número aleatório entre 1 e Y é calculado, se o número estiver na região entre 1 e Y/10, o cromossomo filho sofre mutação, isto é, um número aleatório entre 1 e n é escolhido. Este número representa a posição do gene a ser substituído. Em seguida, um outro número real aleatório entre 0 e 1 é calculado e o gene selecionado está substituído por um novo número. Uma observação importante neste ponto: se a nova aptidão do cromossomo criado por mutação é menor do que a aptidão do cromossomo que está sofrendo o processo, a mutação não ocorre.

O processo evolutivo do algoritmo é composto por um total de *k* gerações, onde as fases mencionadas acima (seleção, cruzamento e mutação) são realizadas repetidamente. No final do processo o cromossomo (vetor de genes) com a maior aptidão, ou seja, o que está mais adaptado é escolhido como os valores ponderados para cada valor de *x* na função de análise [Lacerda 1999].

3. Proposta de Compactação e Redução de Chaves.

O processo de otimização pela combinação dos mecanismos propostos por [Coron 2011] e [Bilar 2014] para a compactação e redução de chaves, consiste basicamente na calibragem dos valores dos parâmetros usados nas primitivas criptográficas através do algoritmo genético. Essa seção apresenta os detalhes da proposta.

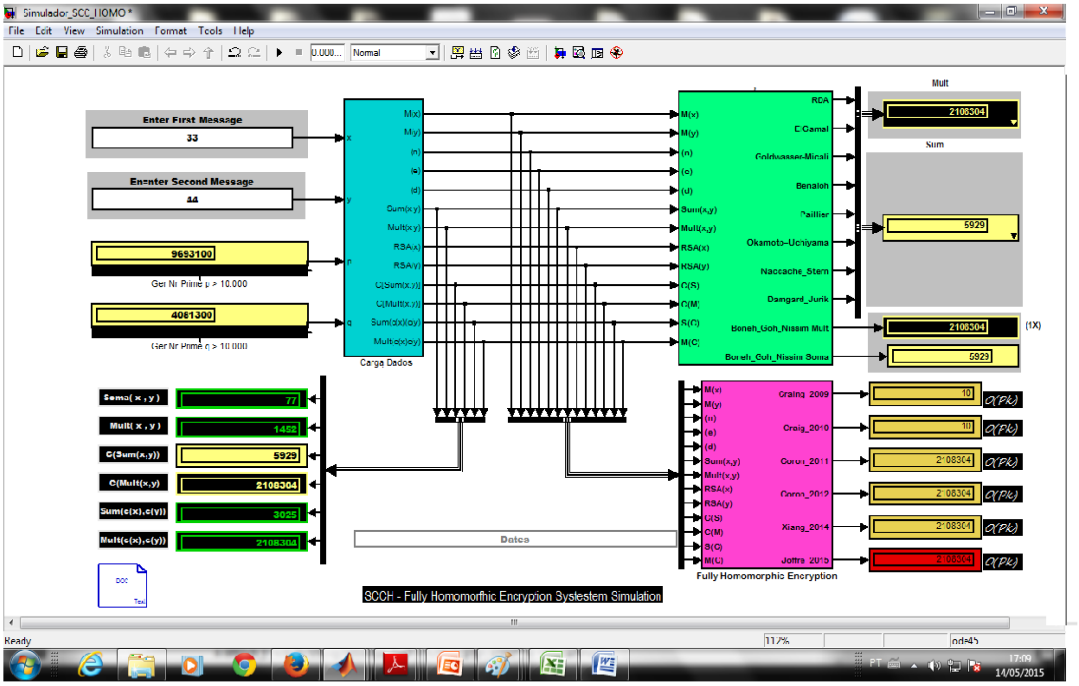

Figura 1. Tela Principal do Simulator de Criptografia Homomórfica.

O esquema proposto, tanto para a Criptografia Parcialmente, como para a Completamente Homomórfica, é implementado no software matemático Matlab/Simulink©, bem como todas as primitivas criptográficas descritas em [Coron 2011] e [Bilar 2014]

Podemos observar na Figura 1, a tela principal do simulador de criptografia homomórfica implementado no Simulink. Neste módulo é implementado o estado da arte em termos de metódos criptográficos homomórficos. Inclusive os esquemas de [Coron 2011] e o de [Bilar 2014].

Na Figura 2, observa-se os módulos necessários à base de todos os cálculos, entre eles: os cálculos de primalidade, geração de números aleatórios, cálculos modulares, bem como todos os cálculos numéricos básicos. O módulo de criptografia completamente homomórfico, mostrado na Figura 1, e que será expandido na Figura 3, é o principal módulo do simulador SCC_Homo (Simulador de Criptografia Completamente Homomórfico). Ele é utilizado para a implementação dos métodos abordados nessa proposta, bem como a execução das primitivas do Agoritmo Genético, isto é, onde todas às análises, mutações, seleções e desenvolvimentos são executados.

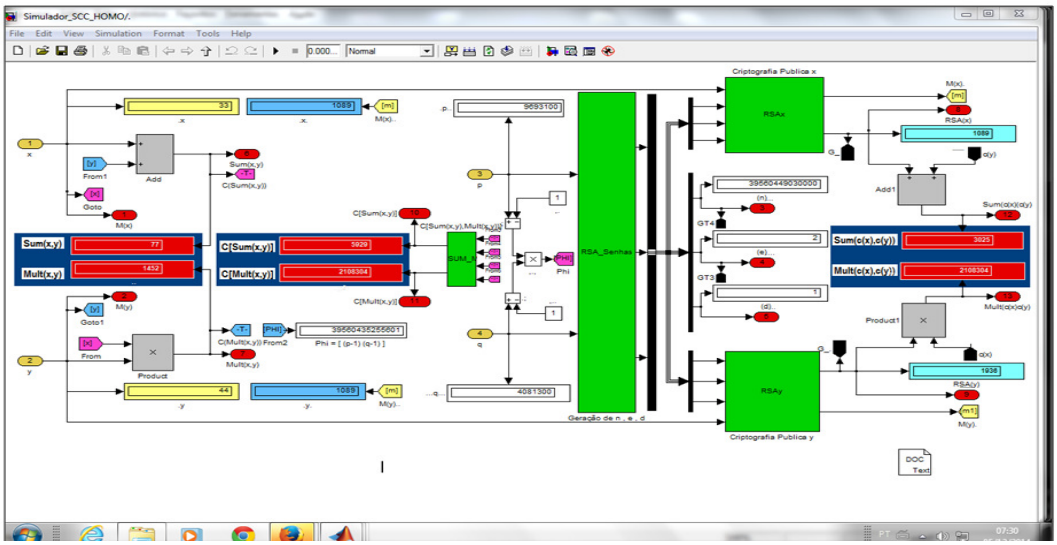

Figura 2. Módulo Básico de Cálculos.

Neste módulo (Figura 3) foram implementados alguns métodos de criptografia relevantes e necessários a demonstração da nossa proposta. Entre eles, o trabalho pioneiro de [Gentry 2009], juntamente com otimizações feitas por ele em [DGHV 2010].

Inclusive os métodos que são a base para esse trabalho [Coron 2011] e [Bilar 2014], onde os tamanhos das chaves públicas são otimizados e reduzidos. Em particular, devem ser observadas as primtivas criptograficas: *KeyGen*, *Encrypt*, *Decrypt*, *Evaluate*, *Recrypt* e *Expand*, que são codificadas e executadas através de simulações dos algoritmos propostos por [Coron 2011] e que são aplicados as análises, variações e validações dos valores dos parâmetros calculados usando Algoritmos Genéticos.

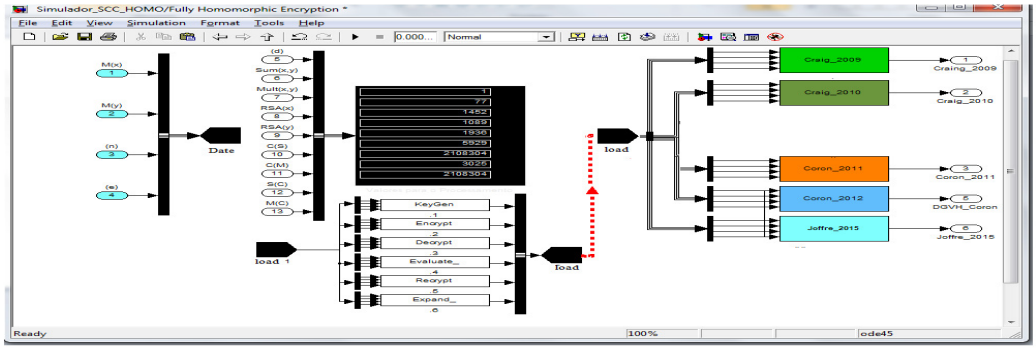

Figura 3. Módulo de Criptografia Completamente Homomórfica.

Coron implementou toda a sua proposta para o esquema DGHV utilizando o software matemático SAGE (Sistema para Experimentação de Álgebra e Geometria) [William 2012]. Como base comparativa, todas as métricas e primitivas de [Coron 2011], Tabela 1, e de [Bilar 2014], Tabela 3, implementadas originalmente em PYTHON, foram reimplementadas e simuladas em nossa proposta no simulador em Matlab/Simulink©. Os testes foram realizados, e os resultados analisados e comparados com aos resultados obtidos pelos autores em seus trabalhos, tendo como resposta a igualdade de soluções que corroboraram com os resultados apresentados pelos artigos originais.

Os valores dos parâmetros definidos por [Coron 2011], Tabela 1, são restritos, sendo: *Toy* possuindo um lambda (λ) equivalente a 42 bits de segurança, *Small* equivalente a 52 bits de segurança, *Medium* equivalente a 62 bits de segurança e *Large* equivalente a 72 bits de segurança. Tal parâmetro define o nível de segurança atribuído diretamente ao par de chaves gerados pela primitiva de geração de chaves *KeyGen*.

Tabela 1. Tempos de Execução Obtidos por Coron

Parâmetros	λ	ρ	η	μ	$\gamma \times 10^6$	Θ
Toy	42	16	336	56	0.061	195
Small	52	20	390	65	0.270	735
Medium	62	26	438	73	1.020	2925
Large	72	34	492	82	2.200	5700

A Tabela 2 ilustra os tempos de execução obtidos por [Coron 2011].

Tabela 2. Tempos de Execução Obtidos por Coron

Parâmetros de Segurança	*KeyGen*	*Encrypt*	*Decrypt*	*Expand*	*Recrypt*
Toy	0,06 s	0,05 s	0,00 s	0,01 s	0,41 s
Small	1,00 s	1,00 s	0,00 s	0,15 s	4,50 s
Medium	28.00 s	21,00 s	0,01 s	2,70 s	51,00 s
Large	10 min 00 s	7 min 15s	0,05 s	51,00 s	11min34 s

A literatura, normalmente, utiliza a medida do tempo de execução de cada uma das primitivas criptográficas (os algoritmos) a fim de quantificar e avaliar o desempenho de cada uma delas [Bilar 2014]. As primitivas são executadas de maneira repetitiva – todos os testes seguem uma distribuição normal e possuem intervalo de confiança de 95% – e tem o seu tempo de execução contabilizado por software. Em posse do tempo contabilizado e do número de vezes de execução das primitivas aplica-se uma média

aritmética simples sobre os mesmos, obtendo assim o tempo médio de execução das respectivas primitivas que pode ser utilizado de maneira comparativa entre variadas implementações e variados esquemas homomórficos [Bilar 2014]. A Tabela 3 ilustra os tempos de execução das primitivas criptograficas obtidos por [Bilar 2014], onde se observa que não houve resultados para os cálculos do parâmetro *Large* em virtude do "estouro" de memória em seus testes.

As avaliações, calibrações e análises, propostas nesse trabalho, inicializam-se com o processo de geração de valores ponderados para o parametro λ, por meio do Algoritmo Genético, e inicializados com a produção de uma massa de dados de 500 MB sem formatação. Essa massa de dados é gerada seguindo as métricas das primitivas de [Coron 2011] e reproduzem o estado original de testes de suas variantes [Coron 2011].

Tabela 3. Tempos de Execução Obtidos por Bilar (* "*memory overflow*")

Parâmetros de Segurança	*KeyGen*	*Encrypt*	*Decrypt*	*Evaluate*
Toy	0.6 s	0.02 s	0.0 s	0.2 s
Small	3.6 s	0.6 s	0.0 s	1.9 s
Medium	1 min 48 s	55 s	0.0 s	14.7 s
Large	*	*	*	*

Esta massa é inicialmente usada para duas finalidades: *i*) a calibração dos módulos de forma análoga as primitivas de [Coron 2011], principalmente a do algoritmo de avaliação do tempo de execução das primitivas; e, *ii*) ser usado como dados de treinamento para o algoritmo genético (A.G.).

Tabela 4. Testes do Parâmetro λ *Toy* Utilizando Algoritmo Genético

# λ	40	41	42	43	44	45
KeyGen	0,05´´	0,05´´	0,06´´	0,06´´	0,17´´	0,21´´
Encrypt	0,04´´	0,05´´	0,05´´	0,05´´	0,06´´	0,16´´
Decrypt	0,00´´	0,00´´	0,00´´	0,00´´	0,00´´	0,00´´
Expand	0,01´´	0,01´´	0,01´´	0,01´´	0,03´´	0,05´´
Recrypt	0,29´´	0,39´´	0,41´´	0,41´´	1,00´´	1.30´´
Evaluate	0.17´´	0.19´´	0.20´´	0.20´´	0.35´´	0.43´´

Mil (1.000) rodadas do processo são executadas para cada um dos tamanhos originais dos parâmetros de segurança λ: *Toy* (42 bits), *Small* (52 bits), *Medium* (62 bits) e *Large* (72 bits), seguido a proposta de [Coron 2011].

Tabela 5. Testes do Parâmetro λ *Small* Utilizando Algoritmo Genético

# λ	***50***	***51***	***52***	***53***	***54***	***55***
KeyGen	**0,49´´**	**0,59´´**	**1,00´´**	**3,00´´**	**7,00´´**	**8,00´´**
Encrypt	**0,53´´**	**0,59´´**	**1,00´´**	**3,50´´**	**7,10´´**	**8,00´**
Decrypt	**0,00´´**	**0,00´´**	**0,00´´**	**0,00´´**	**0,00´´**	**0,00´´**
Expand	**0,14´´**	**0,14´´**	**0,15´´**	**0,19´´**	**0,22´´**	**0,30´´**
Recrypt	**4,15´´**	**4,40´´**	**4,50´´**	**5,51´´**	**6,55´´**	**13,10´´**
Evaluate	**1.50´´**	**1.79´´**	**1.90 s**	**2´10´´**	**3´15´´**	**4.30´´**

Após esta fase inicial dos módulos de calibração e formação dos Algoritmos Genéticos, a fase evolucionária do processo é iniciada. Sendo processada em um total de 100 gerações, onde as fases de seleção, cruzamento e mutação são executadas repetidamente, com a finalidade da convergência do algoritmo para um valor central do tamanho do parâmetro de segurança λ. Nossos testes e simulações foram realizados em uma plataforma Intel (R) Núcleo E4500 CPU (TM) Duo com freqüência 2,20 GHz, 3 GB de RAM e um sistema operacional de 64-bits.

Tabela 6. Testes do Parâmetro *λ Medium* Utilizando Algoritmo Genético

#λ	*60*	*61*	*62*	*63*	*64*	*65*
KeyGen	**27,00´´**	**27,50´´**	**28,00´´**	**31,00´´**	**58,0´´**	**2´00´´**
Encrypt	**20,00´´**	**20,45´´**	**21,00s**	**22,12´´**	**24,10´´**	**28,00´´**
Decrypt	**0,01´´**	**0,01´´**	**0,01´´**	**0,01´´**	**0,02´´**	**0,03´´**
Expand	**2,30´´**	**2,60´´**	**2,70´´**	**4,80´´**	**7,30´´**	**10,70´´**
Recrypt	**48,00´´**	**50,00´´**	**51,00s**	**1´00´´**	**2´10´´**	**3´23´´**
Evaluate	**11.50s**	**13.20s**	**14.70s**	**30.15´´**	**1´00´´**	**3´12´´**

Em cada uma das gerações do algoritmo, 1000 rodadas foram realizadas para cada valor de λ. Variando-se na ordem de números inteiros [λ - 2, λ + 3]. Totalizando para cada parâmetro: *Toy*, *Small*, *Medium* e *Large*, 6.000 rodadas. Um total de 24.000 rodadas para cada geração. Por fim todo o processo evolutivo do algoritmo genético necessitou de 24.000.000 rodadas. Podemos observar nas Tabelas 4, 5, 6 e 7, quando da execução dos parâmetros de segurança λ, os tamanhos obtidos para cada primitiva, bem como os seus tempos de execução.

Tabela 7. Testes do Parâmetro *λ Large* Utilizando Algoritmo Genético

# λ	*70*	*71*	*72*	*73*	*74*	*75*
KeyGen	**9´10´´**	**9´55´´**	**10´00´´**	**12´00´´**	**17´00´´**	**21´00´´**
Encrypt	**3´35´´**	**5´12´´**	**7´15´´**	**11´10´´**	**15´00´´**	**21´00´´**
Decrypt	**0,04 ´´**	**0,05 ´´**	**0,05´´**	**0,05´´**	**0,06´**	**0,07 ´´**
Expand	**50,00´´**	**50,00´´**	**51,00´´**	**51,00´´**	**55,00´´**	**58,00´´**
Recrypt	**10´10´´**	**11´00´´**	**11´34´´**	**12´20´´**	**13´00´´**	**15´05´´**
Evaluate	**10´30´´**	**11´45´´**	**12´00´´**	**123´30´´**	**14´05´´**	**15´00´´**

Depois de todas as rodadas de simulações, testes identicos ao de [Coron 2011] foram realizados com valores calculados para parâmetros λ na verificação da Segurança Semântica do método. Foi verificado que os valores dos parâmetros λ convergiram para os observados na Tabela 8. Valores que, além de serem uma unidade de magnitude menores do que os parâmetros definidos por [Coron 2011], possuem uma redução substancial nos tempos de execução em cada umas das primitivas criptográficas do mecanismo. Mantendo mesmo assim a segurança semântica em todo o processo. Apesar de terem sido calculados valores menores, bem como menores tempos de execução das primitivas criptográficas, observado nas tabelas: 4, 5, 6 e 7, tais valores quando utilizado no método não provêem segurança semantica, não sendo apropriados para o uso no mecanismo.

Observamos na Figura 4 os gráficos comparativos dos tempos de execução das primitivas criptográficas: *Toy, Small, Medium e Large,* quando executadas pelo três métodos em análise nesse trabalho.

Além disso, introduzimos os testes de re-criptografia para analisar os tempos de execução ciclicos, isto é, a criptografia executada recursivamente nível a nivel pelo método. Testes esses não executados por [Bilar 2014] e observados nas tabela 7. Obtivemos também, os resultados para o parâmetro λ *Large*, não alcançado por [Bilar 2014] em virtude do "estouro" de mémoria de seus testes, observado nas Tabelas: 3 e 7. Tais testes foram realizados e demonstrados com a finalidade da maior abrangência e para maior visibilidade nas análises.

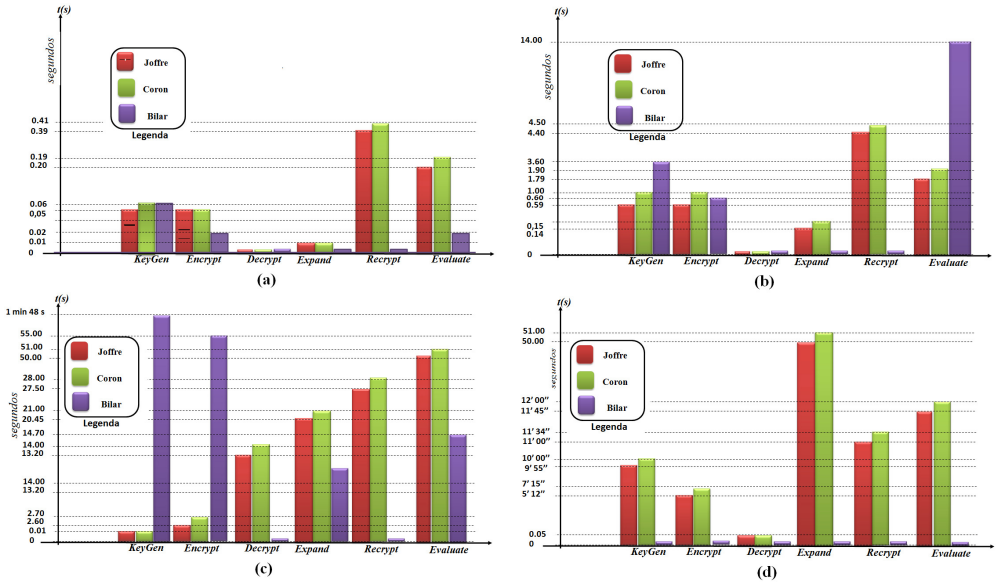

Figura 4. Comparativo dos tempos de execução das primitivas criptográficas dos métodos de Joffre, Coron e de Bilar. (a)*Toy.* (b)*Small.* (c)*Medium.* (d)*Large.*

Tabela 8. Tamanho Reduzido do Parâmetro λ com Segurança Semântica

Parametro	*Toy*	*Samll*	*Medium*	*Large*
λ	41	51	61	71

4. Conclusão e Trabalhos Futuros.

Neste trabalho foram utiluizados Algoritimos Genéticos para a redução dos tamanhos das chaves públicas e do tempo de execução das variantes das primitivas de Coron. Demonstramos com este trabalho que, ao utilizarmos esta técnica para a Calibração do Mecanismo de Coron, combinado à Técnica de Bilar, conseguimos reduzir o tamanho dos parâmetros de segurança do algoritmo criptográfico. Mantendo, mesmo assim, a segurança semântica do mecanismo e, alcançando, em consequência disso, a redução dos tempos de execução de todo o processo. Além disso, introduzimos os testes de re-criptografia não realizados por Bilar, e, conseguimos também, resultados para o teste do parâmetro λ *Large*, não alcançado por ele em seus experimentos. Tendo como consequencia o aumento da visibilidade das análises dos métodos. Como trabalho futuro, procuraremos relalizar os experimentos e calibrações por meio de outras

heurísticas com base em inteligência de enxames, como por exemplo: as colônias de formigas, o vôo dos pássaros, ou mesmo, o algoritmo de busca de alimentos dos sapos**.**

Referências

Bilar, G. R. "Implementação do esquema totalmente homomórfico sobre números inteiros utilizando python com compressão de chave pública". – Trabalho de Graduação - UNIVEM. 2014.

Boneh, D., Halevi, S., Hamburg, M., et al. "Circular-secure encryption from decision diffie-hellman". In: Advances in Cryptology–CRYPTO 2008,Springer, pp. 2008.

Buchmann, Johannes A. "Introdução a Criptografia". Ed. Berkeley, São Paulo 2002.

Brakerski, Z., gentry, C., Vaikuntanathan, V. "Fully homomórfica encryption without bootstrapping", ITCS 2012, 2012.

Coron, J., Naccache, D., Tibouchi, M. Optimization of Fully Homomórfica Encryption. Cryptology ePrint Archive, Report 2011/440, 2012.

Coron, J., Mandal, A., Naccache, D., et al. "Fully homomorphic encryption over the integers with shorter public keys", Advances in Cryptology–, pp. 487–504, 2011.

CSA Security Guidance for Critical Areas of Focus in Cloud Computing –v2.1. Cloud Security Alliance.2009.

DHGV - Dijk., M. Van, Gentry, C., Halevi, S. e Vaikuntanathan, V., Fully homomorphic encryption over the integers. *In H. Gilbert* (Ed.), EUROCRYPT 2010, LNCS, vol. 6110, Springer, p. 24-43, 2010.

Gentry, C. "Fully homomórfica encryption using ideal lattices". In: Proceedings of the 41st annual ACM symposium on Theory of computing, pp. 169–178. ACM, 2009.

Lacerda, E.G.M e Carvalho, A.C.P.L. "Introdução aos algoritmos genéticos", In: Sistemas inteligentes: aplicações a recursos hídricos e ciências ambientais. Editado por Galvão, C.O., Valença, M.J.S. Ed. Universidade/UFRGS: ABRH. 1999.

Michael O. Rabin. Probabilistic algorithm for testing primality. Journal of Number Theory , 12(1):128 – 138, 1980.

Morris, Christopher , "Navy Ultra's Poor Relations", in Hinsley, F.H.; Stripp, Alan, *Codebreakers: The inside story of Bletchley Park*, Oxford: Oxford University Press, p. 235, ISBN 978-0-19-280132-6- 1993

NIST- National institute of standards and technology. Cyber security Framework Development Overview.NIST's Role in Implementing Executive Order 7213636, Improving Critical Infrastructure Cybersecurity, Presentation to ISPAB, 2013.

RDA - R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms, in r. a. demillo et al. In Eds.), Foundations of Secure Computation, pages 169–179. Academic Press, 1978.

Smart, N.; Vercauteren, F. Fully homomórfica encryption with relatively small key and ciphertext sizes. Cryptology ePrint Archive, Report 2009/571, 2009.

Stalling, Willian, Criptografia e Segurança de Redes: Princípios E Práticas 4. Ed. Prentice Hall Brasil, pag 17-36. 2007.

Sousa, F. R. C.; Moreira, L. O.; Machado, J. C. Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios. Fortaleza. 2009.

William Stein. SAGE: A Computer System for Algebra and Geometry Experimentation. 2012.

