Comparação de Eficiência Entre DGHV e GSW no contexto de Recuperação de Dados com Privacidade

Alex Luiz Domingues Cassinelli, 249884

[1] Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Campinas, SP – Brazil

a249884@dac.unicamp.br

Resumo. *A criptografia homomórfica possibilita a realização de operações sobre dados cifrados sem revelar o conteúdo das mensagens, característica que a torna especialmente atraente para aplicações que demandam altos níveis de privacidade. Entre essas aplicações está a Recuperação Privativa de Informações (PIR), na qual um cliente consulta um banco de dados sem expor qual o elemento desejado. Contudo, a eficiência prática desses sistemas varia significativamente entre diferentes esquemas homomórficos, sobretudo em relação ao tempo de execução e uso de memória. Diante disso, este relatório conduz uma análise comparativa entre dois esquemas homomórficos representativos: DGHV, baseado em teoria dos números, e GSW, fundamentado em problemas de reticulados. O estudo examina suas propriedades estruturais, comportamento de ruído e desempenho em cenários de PIR, avaliando como cada esquema se comporta sob diferentes tamanhos de bancos de dados e padrões de acesso. Por fim, busca-se identificar qual deles oferece o melhor equilíbrio entre segurança, viabilidade prática e custo computacional para aplicações de recuperação privativa.*

1. Introdução

A criptografia evoluiu de sistemas clássicos simples para construções modernas com propriedades avançadas, como o suporte a operações sobre dados cifrados. Esse avanço possibilita aplicações antes difíceis, incluindo computação distribuída segura e protocolos de recuperação privativa de informação. A criptografia homomórfica é um tipo de criptografia capaz de realizar operações, como adições e multiplicações, sobre os dados criptografados. Este desenvolvimento abriu portas para novas técnologias, como a computação distribuída segura, o treinamento de modelos de inteligência artificial usando dados secretos e, para o contexto deste relatório, a recuperação privativa de informações. Entretanto, a criptografia homomórfica não é um monolito singular, mas é baseada em diferentes tipos de problemas matemáticos, cada um com seus pontos positivos e negativos.

Dessa forma, este relatório tem como objetivo estudar dois esquemas criptográficos homomórficos baseados em fundamentos matemáticos distintos, analisando seus mecanismos internos, limitações e propriedades operacionais. Como componente prático, conduz-se também uma comparação experimental entre ambos, utilizando seu desempenho na tarefa de recuperação privativa de informações (PIR) como critério de avaliação.

O relatório está organizado da seguinte forma: inicia-se com uma fundamentação teórica sobre os problemas matemáticos que originam os esquemas estudados; em seguida, descrevem-se formalmente os algoritmos DGHV e GSW, bem como suas propriedades homomórficas e limites de ruído. Por fim, apresenta-se a metodologia adotada para os experimentos, seguida da análise dos resultados obtidos.

2. Fundamentação Teórica

2.1. Criptografia Homomórfica

A criptografia homomórfica é uma área da criptografia e seus esquemas podem ser definidos como:

- **Definição 2.1:** Considere
  - $\mathcal{M}$ como o conjunto de mensagens possíveis da cifra;
  - m_0 e $m_1 \in \mathcal{M}$ como duas mensagens;
  - $c_i = HE.Enc(m_i)$ como a função de encriptação da cifra homomórfica, com c_i sendo o texto cifrado resultante;
  - $m_i = HE.Dec(c_i)$ como a função de decifração da cifra homomórfica, com m_i sendo o texto claro correspondente;
- HE é uma cifra homomórfica para uma operação $\star$, se houver uma operação $\diamond$ tal que:

$$HE.Enc(m_1) \diamond HE.Enc(m_2) = HC.Enc(m_1 \star m_2)$$

, $\forall m_1, m_2 \in \mathcal{M}$ [Acar et al. 2018].

Em termos práticos, as cifras homomórficas permitem realizar operações sobre textos cifrados como se estivessem sendo aplicadas diretamente aos textos claros, e o resultado da operação, uma vez decifrado, corresponde ao resultado da operação feita em claro. Para garantir segurança, esses esquemas utilizam um conceito fundamental chamado ruído, que é introduzido em cada criptograma. O ruído funciona como um embaralhamento que impede ataques diretos ao sistema, mas também impõe um limite à quantidade de operações que podem ser realizadas: se o ruído crescer além de um certo limite, a mensagem se torna indecifrável.

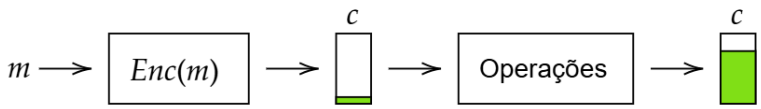

Figure 1. Representação do funcionamento do ruído

Toda cifra homomórfica é composta por três algoritmos principais: *KeyGen*, responsável pela geração de chaves; *Enc*, que realiza a cifragem do texto claro e *Dec*, que recupera o texto original a partir do criptograma.

Com base na quantidade de operações que suportam antes que o ruído inviabilize a decifragem, as cifras homomórficas são divididas em três categorias. As cifras homomórficas parciais, do inglês, *Partial Homomorphic Encryption* (PHE), suportam

apenas um tipo de operação (adição ou multiplicação). Um exemplo clássico é o *textbook* RSA [Doan et al. 2023], que é homomórfico apenas para multiplicação. Já as cifras do tipo *Somewhat Homomorphic Encryption* (SHE) permitem tanto somas quanto multiplicações, mas apenas até uma profundidade limitada de circuito, antes que o ruído impeça a decifragem correta, e podem ser chamadas, também, de cifras totalmente homomórficas por nível. Por fim, as *Fully Homomorphic Encryption* (FHE) não impõem limite teórico ao número de operações, pois são capazes de executar um procedimento de *bootstrapping*, que "renova" o criptograma reduzindo o ruído acumulado a um nível seguro. É importante observar que o *bootstrapping* não elimina o ruído completamente, já que adiciona um novo ruído próprio, mas garante que o texto cifrado permaneça utilizável para novas operações.

Por fim, esquemas homomórficos costumam ter fatores de expansão, ou seja, o quanto de armazenamento é necessário para armazenar o criptograma gerado pela mensagem.

2.2. Problema Máximo Divisor Aproximado Comum (ACD)

O problema do Máximo Divisor Aproximado Comum (ACD) é um problema vindo da teoria dos números e, antes de definí-lo, é necessário definir a distribuição que dá suporte ao problema [Pereira and Morais 2021].

Definição 2.1 (Distribuição $\mathscr{D}_{\gamma,\rho}(p)$): Considere os parâmetros inteiros γ, η, ρ tal que $\gamma > \eta > \rho > 0$. Seja p um primo que satisfaz $2^{\eta-1} \leq p \leq 2^{\eta}$. A distribuição $\mathscr{D}_{\gamma,\rho}(p)$ é definida como:

$$\mathscr{D}_{\gamma,\rho}(p) := \{q \leftarrow [\![0, 2^{\gamma}/p[\![, r \leftarrow [\![-2^{\rho}, 2^{\rho}[\![, \text{ devolva } x := pq + r\}$$

Definição 2.2 (Problema ACD)[Howgrave-Graham 2001]: O problema ACD aplicado sobre os parâmetros γ, η, ρ, também podendo ser chamado (γ, η, ρ)–ACD, consiste em recuperar o valor primo p. Há a versão decisional do problema, chamada de (γ, η, ρ)–ACD decisional, que consistem em diferenciar entre a distribuição $\mathscr{D}_{\gamma,\rho}(p)$ e a distribuição normal definida sobre $\mathscr{U}([\![0, 2^{\gamma}/p[\![)$.

A partir da criptanálise do problema ACD, baseada nos melhores ataques conhecidos, obtêm-se relações assintóticas entre os parâmetros que garantem segurança adequada. Em particular, requer-se que

$$\rho = \Omega(\lambda), \eta = \Omega(\lambda), \gamma = \Omega\big(\lambda(\eta - \rho)^2\big),$$

assegurando que recuperar o primo secreto p permaneça computacionalmente inviável.

2.3. Problemas LWE e RLWE

O problema do Aprendizado com Erros (LWE) é um problema vindo da área de reticulados e, antes de definí-lo, é necessário definir a distribuição que o suporta [Pereira and Morais 2021].

Definição 2.3 (Distribuição $\mathscr{A}_{s,\sigma}$: Considere $n, q \in \mathbb{N}^*$ e $\sigma \in \mathbb{R}, \sigma > 0$. Seja $s \in \mathbb{Z}_q^n$ um vetor fixo secreto. A distribuição $\mathscr{A}_{s,\sigma}$ é definida como:

- Amostre a uniformemente de $\mathbb{Z}_q^n$ e amostre o ruído e a partir da distribuição gaussiana $\mathcal{X}_{\sigma}$;

- Calcule $b := a \cdot s + e \mod q$;
- Devolva (a, b);

Definição 2.4 (Problema LWE): O problema LWE, com parâmetros n, q, σ, ou apenas (n, q, σ)–LWE consiste em calcular o vetor secreto s a partir de uma amostra arbitrariamente grande de $\mathscr{A}_{s,\sigma}$. A versão decisional do problema se resume a diferenciar as distribuições $\mathscr{A}_{s,\sigma}$ de $\mathscr{U}(\mathbb{Z}_q^n \times \mathbb{Z}_q)$.

A criptanálise do problema LWE é baseada em um fator $\alpha := q/\sigma$. Essa relação entre q e σ é crucial porque determina diretamente o tamanho do ruído e, portanto, o nível de segurança do esquema GSW, que será usado nos experimentos deste trabalho. Entretanto, por conta de σ definir a distribuição gaussiana discreta, é de interesse que ele se mantenha pequeno. Portanto, o comum é fixar σ e avaliar os outros parâmetros. Para n, o melhor ataque é uma busca exaustiva, portanto, para garantir a segurança do sistema, basta tomar $n = \Omega(\lambda)$. Por sua vez, q depende, também, de n, pois, conforme o valor de q aumenta, a eficiência dos ataques acompanha o crescimento, portanto, para manter o nível de segurança λ, tem-se a restrição $\log q \in O(n \log(\lambda)/\lambda)$.

O problema de Aprendizado com Erros em Aneis Polinomiais (RLWE), por sua vez, é uma extensão do LWE, porém, ao invés de usar vetores de inteiros, ele se utiliza de polinômios. O principal motivo dessa alteração é o fator de expansão dos criptogramas LWE. Ao passar para o RLWE, o fator de expansão passa de $(n + 1) \log q$ para $((n + 1) \log q/n = O(\log q)$.

Definição 2.5 (Distribuição $\mathscr{R}_{s,n,q,\sigma}$: Considere $N, q \in \mathbb{N}^*$ e $\sigma \in \mathbb{R}, \sigma > 0$. Sejam $n := \varphi(N), R := \mathbb{Z}[X]/\langle \Phi_N(X) \rangle, R_q := R/qR$ e seja $z \in R_q$ um polinômio fixo secreto. A distribuição $\mathscr{R}_{s,n,q\,sigma}$ é definida como:

- Amostre a uniformemente de R_q;
- Para $0 \leq i < n$, amostre $e_i \leftarrow \mathcal{X}_\sigma$ e defina $e = \sum_{i=0}^{n-1} e_i X^i \in R$;
- Calcule $b := a \cdot z + e$ em R_q;
- Devolva $(a, b) \in R_q^2$;

Definição 2.6 (Problema RLWE): O problema RLWE, com parâmetros N, q, σ, ou apenas (N, q, σ)–RLWE consiste em calcular o polinômio secreto z a partir de uma amostra arbitrariamente grande de $\mathscr{R}_{s,n,q,\sigma}$. A versão decisional do problema se resume a diferenciar as distribuições $\mathscr{R}_{s,n,q,\sigma}$ de $\mathscr{U}(R_q \times R_q)$.

Os parâmetros para o RLWE seguem dos parâmetros do problema LWE, entretanto, como n não é mais definido diretamente, deve-se definir $n = \varphi(N)$.

3. O Esquema DGHV

O esquema DGHV, como mencionado na seção 2 é um esquema que se baseia no problema ACD e, portanto, se categoriza como um esquema baseado em Teoria de Números. Deve-se notar que este relatório considerou a versão mais simples do esquema, que é uma versão simétrica.

O esquema DGHV é inicialmente uma SHE, permitindo avaliar circuitos de profundidade limitada antes que o ruído inviabilize a decifragem. Contudo, o esquema também possui um procedimento de *bootstrapping* que o eleva a FHE. No contexto deste relatório, analisa-se a versão SHE, uma vez que a operação de recuperação de dados apresenta profundidade relativamente baixa.

3.1. Definição do Esquema

Considerando o parâmetro de segurança λ, os parâmetros γ, η, ρ como os parâmetros que geram a distribuição $\mathscr{D}_{\gamma,\rho}(p)$, r como o ruído e m como uma mensagem $m \in \mathcal{M}$, o esquema DGHV pode ser definido como o seguinte conjunto de algoritmos [van Dijk et al. 2010]:

- **KeyGen(1^λ):** Ao receber o parâmetro de segurança λ, gera os parâmetros γ, η, ρ referentes ao nível de segurança e, com os parâmetros em mãos, ele escolhe a chave secreta p de η bits;
- **Enc(p, m):** O algoritmo amostra $c' \leftarrow \mathscr{D}_{\gamma,\rho}(p)$ até que $c' \equiv c \mod p$ seja par. Após amostrar c', o algoritmo realiza $c = c' + m$ e retorna o criptograma c;
- **Dec(p, c):** Realiza-se $m' \equiv c \mod p$ e então $m = m' \mod 2$;

O fator de expansão do DGHV é assintótico ao tamanho do maior componente de seu criptograma, no caso, $p \cdot q$, sendo o maior (como amostrado na distribuição ACD, é q, que possui γ bits, portanto, os criptogramas do esquema DGHV possuem fator de expansão da ordem de γ bits.

3.2. Análise do ruído

Para analisar o ruído do DGHV, é necessário que sejam analisados como os criptogramas se comportam durante as operações.

Para a soma dos criptogramas, c_1, c_2, tem-se:

$$\begin{aligned} c_1 + c_2 = c_{add} \rightarrow \\ c_{add} = (pq_1 + 2r_1 + m_1) + (pq_2 + 2r_2 + m_2) = \\ = p(q_1 + q_2) + 2(r_1 + r_2) + (m_1 + m_2) \end{aligned}$$

Ou seja, o ruído cresce de forma aditiva, tendo em vista que $c_1 + c_2 \rightarrow 2(r_1 + r_2)$.

A multiplicação entre criptogramas c_1, c_2 é analisada de forma análoga. Considere Q como todos os termos que possuem o termo p em comum:

$$\begin{aligned} c_1 \cdot c_2 = c_{mult} \\ c_{mult} = (pq_1 + 2r_1 + m_1) \cdot (pq_2 + 2r_2 + m_2) = \\ p \cdot Q + (4r_1r_2 + 2r_1m_2 + 2m_2r_1) + (m_1m_2) \end{aligned}$$

Portanto, o ruído apresenta crescimento multiplicativo, já que apresenta a multiplicação direta dos ruídos.

Esta análise revela que o ruído tende a crescer rapidamente durante as operações, principalmente se forem envolvidas multiplicações, o que implica que os circuitos que podem ser avaliados pelo esquema são de pouca profundidade [van Dijk et al. 2010].

O limite superior para o ruído decorre da operação de decifragem. Como a recuperação da mensagem realiza

$$m' \equiv c \pmod p = m + 2r \pmod p,$$

é necessário que o termo de ruído satisfaça $|r| < p/2$, garantindo que a redução módulo p não exceda a região centrada na mensagem m. Caso essa condição falhe, o valor obtido após o módulo é ambíguo, resultando em decifragem incorreta.

4. O Esquema GSW

O esquema GSW, é um esquema que se baseia no problema RLWE e, portanto, se categoriza como um esquema baseado em reticulados. O esquema é uma FHE, possuindo uma operação de *bootstrapping*, entretanto, neste relatório a operação não foi utilizada, tendo em vista a baixa profundidade do circuito da operação de recuperação de dados, não havendo necessidade para tal.

A proposta inicial deste esquema foi feita sobre o LWE [Gentry et al. 2013], mas, para este relatório, foi implementada uma versão diferente, proposta por [Ducas and Micciancio 2015], por conta de sua compressão de mensagens em formato de polinômio, o que diminui o, já alto, fator de expansão do esquema.

4.1. Definição do Esquema

Antes de definir o esquema, é necessária uma definição auxiliar, referente à decomposição de criptogramas. Ela é utilizada para representar elementos de R_q em uma base conveniente para operações homomórficas. Seja q o módulo do esquema e escolha-se uma base B. Defina-se $\ell = \lceil \log_B(q) \rceil$ e o vetor

$$g = (B^0, B^1, \ldots, B^{\ell-1}).$$

Para qualquer $a \in \mathbb{Z}_q$, representado em $(-q/2, q/2)$, define-se $g^{-1}(a)$ como o vetor de coeficientes da decomposição de a na base B, preservando o sinal. A decomposição de um criptograma c é então dada por

$$G^{-1}(c) = \left(g^{-1}(c_0), \ldots, g^{-1}(c_{d-1})\right) \in R_q^{d\ell}.$$

Além da operação de decomposição, é necessário, também, introduzir o conceito de *gadget matrix*, tendo em vista que o vetor resultante da decomposição possui tamanho $d\ell$ e g possui dimensão ℓ. Portanto, define-se $G := I_d \otimes g^T \in \mathbb{Z}^{d\ell \times \ell}$, obtendo uma matriz da forma:

$$G \;:=\; I_d \otimes g \;=\; \begin{pmatrix} g & 0 & \cdots & 0 \\ 0 & g & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & g \end{pmatrix}.$$

Finalmente, é possível definir o esquema GSW. Considere o parâmetro de segurança λ, a distribuição $\mathscr{R}_{s,N,q,\sigma}$, e como o ruído e m como uma mensagem $m \in \mathcal{M}$, o esquema GSW pode ser definido como o seguinte conjunto de algoritmos [Pereira and Morais 2021]:

- **ParamGen(1^λ):** Gera os parâmetros $\sigma > 0$, $N, B, \ell \in \mathbb{Z}$, com N sendo potência de 2. $q := B^\ell$. N, q, σ devem garantir λ bits de segurança para o problema RLWE. Devolva a tupla (N, q, σ, B, ℓ).
- **KeyGen(params):** Defina o polinômio secreto $z = \sum_{i=0}^{N-1} z_i X^i$, com cada $z_i \in \{-1, 0, 1\}$. Devolva z.

- **Enc**$(z, m,$ **params):** Amostre ℓ amostras RLWE, da forma $(a_i, b_i) \leftarrow \mathscr{R}_{s,N,q,\sigma}$ e defina $C' \in R_q^{2\ell \times 2}$, com cada linha sendo composta por uma das amostras. Perceba que $C' = [a, b]$. Devolva $C = C' + m \cdot G \mod q$.
- **Dec**$(z, C,$ **params):** Seja (a, b) a última linha de C. Calcule $u := b - a \cdot z \in R_q$. u pode ser interpretado como um polinômio em $\mathbb{Z}[X]$ Devolva $\lceil \frac{B \cdot u}{q} \rfloor \mod B$.

Além disso, é interessante que sejam definidas as operações de adição e multiplicação entre criptogramas:

- **Soma**(C_1, C_2)**:** Devolva $C_{add} = C_1 + C_2$;
- **Mult**(C_1, C_2)**:** Devolva $C_{mult} = G^{-1}(C_1) \cdot C_2$;

Por fim, note que um criptograma GSW possui informação redundante, tendo em vista que apenas a última linha da matriz de $2\ell \times 2$ é necessária, portanto, o fator de expansão do esquema é alto:

- Cada criptograma GSW possui 2ℓ amostras RLWE;
- Cada amostra RLWE é composta por um par (a_i, b_i), ambos polinômios contendo:
  - N elementos;
  - Cada elemento possui um coeficiente $[0, q-1]$, portanto, cada coeficiente possui $\log q$ bits.
- Portanto, cada criptograma GSW possui um fator de expansão de $4\ell \cdot N \cdot \log q$;

4.2. Análise do ruído

Para analisar o ruído do GSW, assim como na análise do DGHV, é necessário analisar os criptogramas do esquema, e como eles se comportam durante as operações homomórficas.

Para a soma dos criptogramas C_1, C_2, lembre que $C_i = [a_i, b_i]$ e $b_i = a_i \cdot z + e$ tem-se:

$$c_1 + c_2 = ([a_1, b_1] + m_1 G) + ([a_2, b_2] + m_2 G) =$$
$$[a_1 + a_2 + (a_1 + a_2) \cdot z + e_1 + e_2] + (m_1 + m_2)G$$

Como é possível notar, a adição apresenta crescimento aditivo do ruído.

Para a multiplicação dos criptogramas C_1, C_2, temos:

$$C_{mult} = [G^{-1}(C_1) \cdot a_2, G^{-1}(C_1) \cdot b_2] + G^{-1}(C_1) \cdot m_2 G \tag{1}$$

Para simplificação da notação, considere $a' := G^{-1}(C_1) \cdot a_2$, pode-se reescrever a equação 1 como:

$$C_{mult} = [a', a' \cdot z + G^{-1}(C_1) \cdot e_1] + m_2 G^{-1}(C_1) \cdot G \tag{2}$$

Por propriedade de $G^{-1}(C_1) \cdot G = C_1$, portanto, finaliza-se a análise da multiplicação com:

$$C_{mult} = [a' + m_2 a_1, (a' + m_2 a_1) \cdot z + G^{-1}(C_1) \cdot e_2 + m_2 e_1] + m_1 m_2 G$$

Ou seja, o crescimento do ruído para o GSW também é multiplicativo. Esse desdobramento mostra que o esquema é eficiente em avaliar circuitos profundos, tendo em vista o crescimento lento do ruído.

O limite para o ruído, no esquema GSW, é dado analisando a função de descriptografar, porém, como preliminares, considere $w := \lceil \frac{B \cdot u}{q} \rfloor$, além disso, a linha necessária para descriptografar um criptograma, a última, pode ser escrita como $(a, b := a \cdot z + e + B^{\ell-1}m$, ou seja, $u := b - a \cdot z = e + m \cdot q/B \mod q$. Então, pode-se dizer que, sobre os inteiros, existe um polinômio v tal que: $u = e + m \cdot (q/B) + vq \in \mathbb{Z}[x]$.

Finalmente, tem-se que o limite do ruído é dado por:

$$w := \lceil \frac{B \cdot u}{q} \rfloor = \lceil \frac{B \cdot e}{q} + m + vB \rfloor = \lceil \frac{B \cdot e}{q} \rfloor + m + vB$$

Portanto, ao realizar $w \mod B$, o termo vB naturalmente será removido, restando, apenas, o termo $\lceil \frac{B \cdot e}{q} \rfloor$. Para a decriptografação correta, é necessário que o termo possua a seguinte característica: $||\frac{B \cdot e}{q}||_\infty < \frac{1}{2}$, para que o termo m seja arredondado para baixo, tendo em vista que m é inteiro.

Então, para finalizar a análise do limite do ruído, para garantir que a operação Dec(C) seja correta, o ruído não deve ultrapassar:

$$||e||_\infty < \frac{q}{2B}$$

5. Recuperação Privativa de Informações (PIR)

A segurança e privacidade nas redes sempre foi uma grande preocupação, tanto para usuários, quanto para provedores, e, atualmente, é um tópico bastante pesquisado. Vários avanços foram concretizados para atingir um modelo de privacidade elevado para usuários, entre eles está o desenvolvimento da Recuperação Privativa de Informações, do inglês, *Private Information Retrieval* (PIR).

A PIR é uma forma de busca em banco de dados, onde é possível que um cliente recupere informações sem que o próprio banco saiba o que foi recuperado. As aplicações são claras, entretanto, levou algum tempo até que a PIR deixasse de ser apenas teórica e passasse para o campo prático.

Existem dois tipos principais de PIR: PIR da teoria da informação, do inglês, *Information Theoretic PIR* (IT-PIR); e PIR computacional, do inglês, *Computational PIR* (cPIR). A IT-PIR foi apresentada por [Chor et al. 1995], enquanto que a cPIR foi publicada por [Kushilevitz and Ostrovsky 1997]. Este relatório se concentra em protocolos pertencentes à classe cPIR.

5.1. cPIR

Esquemas do tipo cPIR se baseiam em criptografia homomórfica para garantir sigilo do usuário, em especial, os protocolos escoram-se na dificuldade da primitiva criptográfica usada. Isto vai de encontro com a base da IT-PIR, que se utilizam da teoria da computação para garantir sigilo. Essa troca de paradigma oferece uma troca de comportamento: enquanto há um relaxamento para certos requisitos da IT-PIR, especificamente na necessidade da replicação da base de dados, existe um custo computacional atrelado ao uso da criptografia para garantir a privacidade do usuário.

A cPIR foi proposta por [Kushilevitz and Ostrovsky 1997], onde os autores expunham um modelo que se utiliza de criptografia homomórfica para cifrar a posição do dado pedido, impedindo que o banco de dados descubra qual seria tal posição, mas, por propriedade da criptografia homomórfica, permite que operações aritméticas sejam feitas sobre as cifras enviadas. Em especial, o modelo proposto por [Kushilevitz and Ostrovsky 1997] se utiliza de uma cifra baseada no problema dos Resíduos Quadráticos, portanto, sendo baseada na Teoria dos Números.

Em 1998, [Stern 1998] publicou um novo modelo para a cPIR, que serviu de base para muitos dos modelos propostos desde então, incluindo o primeiro modelo cPIR prático: XPIR. O modelo é baseado na representação do banco de dados como um hipercubo de d dimensões, sendo necessário, apenas, $d\sqrt[d]{n}$, com $d = 2$ e $d = 3$, normalmente, ou seja, um custo relativamente baixo, por exemplo, considerando $n = 10^3$, $d = 3$, seriam necessários apenas 30 criptogramas, divididos em três vetores de dez elementos cada, ao invés de mil, como em uma representação do banco de dados como um vetor de uma única dimensão. Para atingir tal complexidade, o autor utilizou da cifra Naccache-Stern. A Figura 2 ilustra um hipercubo possível para uma base de dados de 16 elementos, utilizando $d = 2$ dimensões.

Figure 2. Ilustração de um hipercubo de 16 elementos, e dimensão $d = 2$

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}

Como exemplo de utilização da representação hipercúbica, considere que deseja-se extrair o elemento x_5 do banco de dados. O cliente, sabendo da posição do elemento, monta dois vetores de uma única dimensão, $v_1 = \begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}$, $v_2^T = \begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}$. Com esses dois vetores, basta realizar uma multiplicação vetorial e o elemento x_5 será devolvido. A Figura 3 traz uma representação gráfica do procedimento descrito. Note que são necessários apenas oito criptogramas para conseguir o criptograma desejado, enquanto que seriam necessários dezesseis, caso o banco de dados fosse representado como um vetor de uma única dimensão.

Figure 3. Exemplo de extração de elemento de um banco de dados em formato hipercúbico

$$\begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix} \times \begin{array}{|c|c|c|c|}\hline x_0 & x_1 & x_2 & x_3 \\ \hline x_4 & x_5 & x_6 & x_7 \\ \hline x_8 & x_9 & x_{10} & x_{11} \\ \hline x_{12} & x_{13} & x_{14} & x_{15} \\ \hline \end{array} \longrightarrow \begin{bmatrix}x_4 & x_5 & x_6 & x_7\end{bmatrix} \times \begin{bmatrix}0\\1\\0\\0\end{bmatrix} = x_5$$

A seguir, serão explicitados alguns modelos e protocolos importantes para esta pesquisa. Note que o protocolo OnionPIR foi publicado posteriormente ao protocolo SHECS-PIR. Esta organização foi feita para preservar a continuidade do modelo de organização de banco de dados usado pelos modelos XPIR, SealPIR e OnionPIR.

6. Metodologia

A realização deste trabalho segue o modelo proposto pelo fluxograma 4.

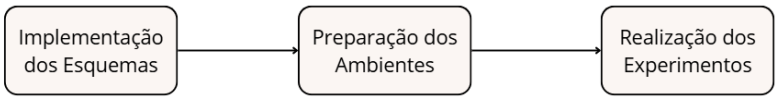

Figure 4. Fluxograma representando o fluxo de trabalho

A etapa de Implementação dos Algoritmos se refere a certos cuidados durante a escrita dos mesmos. A etapa de Preparação dos Ambientes diz respeito à escrita dos códigos dos experimentos, como cada banco de dados foi implementado, quais estruturas foram usadas para a representação dos mesmos e como os pedidos são realizados. Por fim, a etapa de Realização dos Experimentos diz respeito ao contexto da experimentação, quais métricas foram tomadas, detalhes da máquina usada, entre outros detalhes.

Como métricas foram definidos: Tempo total de execução, porque é importante saber quanto tempo leva para realizar a consulta dos elementos; Consumo de RAM, para saber o quão pesadas são as operações no sistema; Crescimento do Ruído, serve para analisar os parâmetros usados, sabendo se devem ser apertados, ou afrouxados.

6.1. Implementação dos Algoritmos

Os algoritmos foram implementados na linguagem SageMath, uma linguagem derivada do Python, sendo uma linguagem menos eficiente que outras mais otimizadas, como o C++. O SageMath foi escolhido, pois é uma linguagem que apresenta funções matemáticas implementadas, o que facilita na hora de escrever as funções dos algoritmos, em especial, do esquema GSW, que depende de funções em aneis polinomiais.

Como parte da implementação, foi necessário modificar o valor de criptografia do esquema DGHV, uma vez que o modelo original suporta apenas mensagens binárias. Para permitir mensagens inteiras, adotou-se um módulo auxiliar t[1] , modificando-se as funções Enc e Dec para:

$$c = p \cdot q + t \cdot r + m, \qquad m = (c \bmod p) \bmod t.$$

Essa alteração não reduz a segurança do esquema, pois preserva a relação essencial entre ruído e chave secreta: desde que $|t \cdot r| < p/2$, o processo de decifragem continua correto e os ataques ao ACD permanecem inalterados, afetando apenas o tamanho do ruído e não a distribuição estrutural dos amostras.

[1]O código-fonte completo pode ser encontrado em GitHub.

6.2. Preparação dos Ambientes

Para realização dos experimentos, foram preparados dois conjuntos de ambientes, representando os esquemas criptográficos. Para cada conjunto, foram construídos ambientes com bancos de dados, estes sendo implementados usando as duas representações: vetorial e hipercúbica. Foram construídos quatro bancos de dados por ambiente, ou seja, quatro bancos de dados vetoriais e quatro matriciais.

Os bancos de dados possuem 2^L elementos cada, sendo $L = 2, 8, 10, 20$. Esses valores foram escolhidos por representarem um aumento gradual na quantidade de operações necessárias para realizar a recuperação do elemento pedido pelo cliente, até um valor relativamente grande, como $L = 20$.

Os elementos nos bancos de dados são inteiros, considerando i, j como os valores referentes ao posicionamento dos elementos na estrutura do banco de dados, os elementos dos bancos de dados vetoriais são representados pela fórmula: $x_i = i$, enquanto que os elementos dos bancos de dados matriciais são representados como: $x_{i,j} = 2 \cdot i + j$. Os elementos foram construídos dessa forma para facilitar na realização da criptografia.

Para criptografar os bancos de dados com GSW, foi necessário polinomializar os elementos, ou seja, primeiramente, os elementos foram representados em binário e, em seguida, os bits foram inseridos em polinômio de grau N como coeficientes:

$$m = (\beta_0, \beta_1, \ldots, \beta_{L-1}) \rightarrow$$
$$\rightarrow m(X) = \sum_{i=0}^{N-1} \beta_i X^i$$

Em seguida, esse polinômio foi passado para a função de criptografia do esquema GSW. Como exemplo, considere o valor 13, cuja representação binária é $(1, 1, 0, 1)$. Ao polinomializá-lo em um anel de grau $N = 8$, obtém-se:

$$m(X) = 1 + 1X + 0X^2 + 1X^3.$$

Os demais coeficientes até X^7 são completados com zero.

Além da biblioteca padrão da linguagem SageMath, foram utilizadas algumas bibliotecas adicionais, para facilitar na criação dos bancos de dados e para realizar as medições. As bibliotecas são:

- Numpy: Escolhida para facilitar com a construção dos bancos de dados e algumas operações entre vetores;
- Random: Usada para escolher elementos aleatoriamente;
- Time: Escolhida para realizar as medições de tempo;
- Csv: Usada para criar os arquivos com os resultados das métricas
- Psutil: Utilizada para realizar a medição do consumo de RAM;
- Os: Usada para realizar a criação e edição dos arquivos locais;

6.3. Realização dos Experimentos

Inicialmente, o experimento instancia o algoritmo criptográfico com parâmetros relevantes, em seguida, ele constrói os vetores de recuperação, criptografa o banco de dados

e realiza as operações necessárias entre eles para recuperar o elementos. Por fim, o elemento é verificado para confirmar que é o elemento pedido e esse elemento é devolvido para o cliente. Todas as operações foram implementadas em um único método, portanto as etapas estão todas em sequência.

Os parâmetros escolhidos para instanciar o esquema DGHV partiram da criptanálise do problema ACD, descrita na seção 2.2, portanto os valores escolhidos foram: $\gamma = 128^2, \eta = (L+1) \cdot 128, \rho = 128$ para o banco de dados em formato vetorial e $\gamma = 128^2, \eta = (L+5) \cdot 128, \rho = 128$, para o banco de dados em formato matricial, sendo L o número de bits necessário para representar o maior elemento do banco de dados. Com esses valores de parâmetros, o nível de segurança do esquema DGHV é de $\lambda = 128$ bits. O motivo da diferença entre as versões vetorial e matricial é por conta da quantidade de operações na versão matricial ser maior, necessitando de um valor de η maior.

Para o esquema GSW, foi utilizado um estimador [Biasioli et al. 2024] para calcular o nível de segurança. Nos experimentos envolvendo o banco de dados em formato matricial, e envolvendo o banco de dados vetorial com $L = 2$ elementos, os parâmetros mantiveram $\lambda = 128$ bits de segurança, mas para os experimentos envolvendo o banco de dados vetorial com $L = 8$ e $L = 10$ elementos, o nível de segurança foi reduzido a $\lambda = 80$ bits, por conta do tempo de execução proibitivo dos parâmetros necessários para executar com 128 bits de segurança. Os parâmetros usados foram: $N = 512, q = 2^{15}, \sigma = 3.2, B = 2$, para o GSW matricial e vetorial com $L = 2$ elementos, e $N = 832, q = 2^{30}, \sigma = 3.2, B = 2^3$, para o GSW vetorial com $L = 8$ e $L = 10$ elementos.

Para realizar os experimentos, foi considerado que o usuário tem conhecimento da posição do elemento desejado no banco de dados. Sendo assim, foram criados vetores *one-hot*, com um bit 1 na posição do elemento pedido. No caso do banco de dados vetorial, foi construído um vetor com n elementos, com o bit 1 na posição i, enquanto para a versão matricial, foram criados dois vetores, cada um com $\sqrt{n}$ elementos, o primeiro vetor posicionou o bit 1 na posição i enquanto que o segundo posicionou o bit 1 na posição j, como descrito na subseção 5.1

Em seguida, são realizadas as operações de multiplicação vetorial e matricial, sendo que as operações de soma e multiplicação entre criptogramas foram usadas no lugar de soma e multiplicação comum, exceto pela função de soma do esquema GSW, tendo em vista que a função apenas soma os criptogramas, sendo idêntica a uma função de soma comum.

Por fim, o resultado é verificado, se estiver correto, os testes continuam, se não, param.

As medições são realizadas durante os testes, com a medição de tempo total começando antes da instanciação dos esquemas criptográficos e terminando após o retorno da função de testes. No decorrer da função de testes, outros tempos são medidos: o tempo apenas para instanciar o algoritmo criptográfico, o tempo para montar os vetores e criptografar o banco de dados e o tempo para realizar as operações. A RAM é medida após a instanciação do esquema criptográfico, após a montagem dos vetores e após as operações. Por fim, a medição do ruído mede os ruídos da criptografia dos vetores e do banco de dados, os ruídos dos criptogramas durante as operações e o ruído do criptograma

resultante.

Os experimentos foram executados em uma quantidade variável, pois foi notado que houve custo de tempo elevado para alguns dos testes, bem como a baixa variação dos resultados, indicando que os tempos eram relativamente constantes, tendo poucos *outliers*.

7. Resultados

Nesta seção, encontram-se expostos os resultados das medições dos testes executados, bem como uma breve discussão sobre o significado dos resultados. Os testes foram realizados em uma máquina virtual, utilizando Ubuntu 24.04, com processador Intel Core i5-11400H e 8 GB de RAM. Os comandos foram executados, ou diretamente no terminal do Linux, ou através de um arquivo bash, usado para facilitar nos testes com L maior.

Um comentário breve antes de iniciar a discussão dos resultados é que não foi possível realizar os testes nos ambientes com $L = 20$ elementos com o esquema GSW, isso se dá por conta do fator de expansão do esquema que, como citado na subseção 4.1, é de $4\ell \cdot N \cdot \log q$ bits. Portanto, um valor numérico que representa o valor de memória RAM necessária para armazenar um elemento desse banco de dados criptografado é de:

- $4 \cdot 15 \cdot 512 \cdot 15 = 460.800$ bits, ou, aproximadamente, 57 KB para um elemento do conjunto de parâmetros $N = 512, q = 2^{15}, sigma = 3.2, B = 2$;
- $4 \cdot 27 \cdot 832 \cdot 30 = 2.695.680$ bits, ou, aproximadamente 337 KB para um elemento do conjunto de parâmetros $N = 832, q = 2^{30}, sigma = 3.2, B = 2^3$;

7.1. Análise dos Tempos de Execução

As medições de tempo estão indicados nos gráficos 5 a 7, além das médias dos valores serem indicadas nas tabelas 1 a 4. O gráfico 5 está relacionado com a tabela 1, o gráfico 6 está relacionado com a tabela 2, o gráfico 7 está relacionado com a tabela 3.

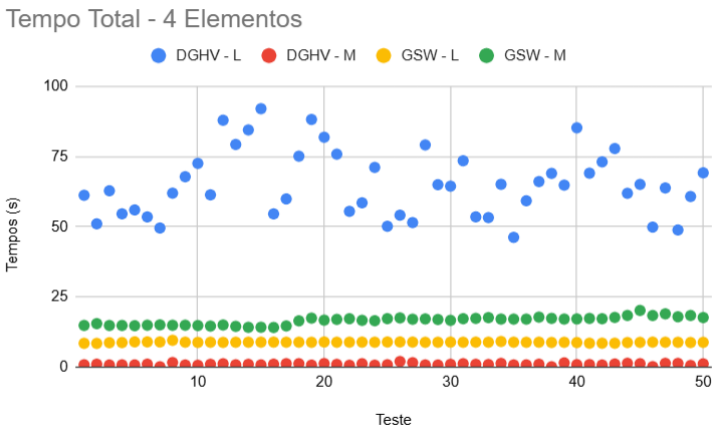

Figure 5. Gráfico representando os tempos medidos para $L = 2$

Tipo de Banco de Dados	Bancos de Dados	Média de Tempos (s)
DGHV	Linear	0,25
	Matricial	0,96
GSW	Linear	8,83
	Matricial	16,50

Table 1. Tempos médios de execução para $L = 2$

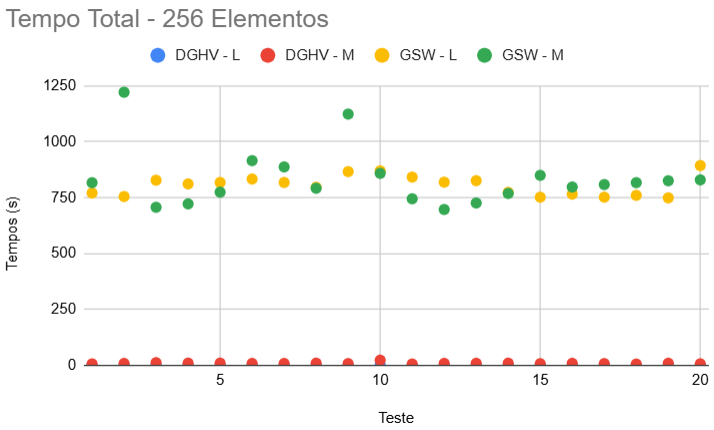

Figure 6. Gráfico representando os tempos medidos para $L = 8$

Ambiente	Banco de Dados	Média de Tempo (s)
DGHV	Linear	1,82
	Matricial	9,41
GSW	Linear	804,97
	Matricial	834,15

Table 2. Tempos médios de execução para $L = 8$

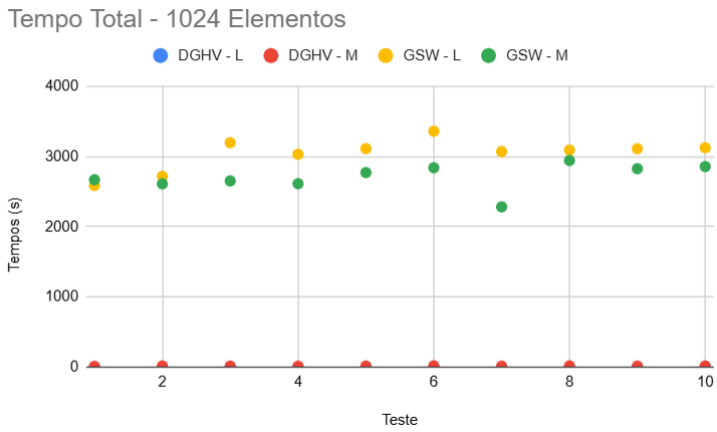

Figure 7. Gráfico representando os tempos medidos para $L = 10$

Tipo de Banco de Dados	Bancos de Dados	Média de Tempos (s)
DGHV	Linear	3,76
	Matricial	13,89
GSW	Linear	3039,50
	Matricial	2705,42

Table 3. Tempos médios de execução para $L = 10$

Ambiente	Banco de Dados	Média de Tempo (s)
DGHV	Linear	165,01
	Matricial	269,11
GSW	Linear	—
	Matricial	—

Table 4. Tempos médios de execução para $L = 20$

Como pode-se notar, os tempos de execução foram bastante díspares, especialmente a partir de $L = 8$ elementos. Embora, do ponto de vista teórico, as operações do GSW são mais eficientes do que as do DGHV, por conta de suas raízes nos reticulados, o GSW foi de 8 a 300 vezes mais lento que o DGHV. Conjectura-se dois principais motivos para esta diferença: Em primeiro lugar, o GSW é composto por várias amostras RLWE, ou seja, o número de operações realizado é maior que o número do DGHV, em segundo lugar, a falta de otimizações da linguagem podem favorecer operações mais simples, como o produto entre inteiros, em relação ao produto matricial.

7.2. Análise do Consumo de RAM

A tabela 5 expõe as médias das medições do consumo de RAM pelas operações realizadas para os ambientes do DGHV e 6 o fazem para os ambientes do GSW:

Banco de Dados	L	Média de Consumo (MB)
Linear	2	241
	8	264
	10	267
	20	4549
Matricial	2	261
	8	309
	10	284
	20	2515

Table 5. Médias de consumo de RAM durante as operações para $L = 2$

Banco de Dados	L	Média de Consumo (MB)
Linear	2	243
	8	381
	10	811
Matricial	2	247
	8	329
	10	549

Table 6. Médias de consumo de RAM durante as operações para $L = 8$

Como pode-se constatar, o consumo de RAM seguiu um padrão de aumento conforme o número de elementos aumentou, estabelecendo seu pico ao atingir $L = 20$ elementos. Um comportamento curioso foi a diminuição de consumo médio entre $L = 8$ e $L = 10$ nos ambientes de DGHV matricial. Descontando esse comportamento aberrante, o crescimento do consumo de RAM se comportou de maneira esperada, com o esquema GSW, em média, precisando de mais RAM para realizar suas operações, tendo em vista a quantidade de operações, embora não tenha sido uma quantia tão maior.

Note que esta medição não inclui o custo da RAM para armazenamento do banco de dados.

7.3. Análise do Crescimento do Ruído

As tabelas 7 e 8 mostram as médias de ruído para os testes de $L = 2$ e $L = 8$. Note que as medições de ruído são em $\log_2$, ou seja, a medida de ruído r é, na verdade, 2^r:

Tipo de Banco de Dados	Início	Final
DGHV - Linear	147	295
DGHV - Matricial	146	440
GSW - Linear	3,18	10,08
GSW - Matricial	3,71	11,15

Table 7. A tabela representa os valores de ruído como $\log_2 r$, para $L = 2$

Tipo de Banco de Dados	Início	Final
DGHV - Linear	147	297
DGHV - Matricial	147	445
GSW - Linear	3,18	16,73
GSW - Matricial	3,71	12,55

Table 8. A tabela representa os valores de ruído como $\log_2 r$, para $L = 8$

Como pode ser constatado, o crescimento do ruído não variou muito entre os testes, tendo em vista a maior quantidade de elementos nos bancos de dados. Tendo em vista a baixa mudança dos parâmetros, os valores de ruído iniciais não mudam, entretanto, era esperado um crescimento maior do ponto de vista do DGHV, já que, com mais produtos entre elementos, e com o crescimento multiplicativo do ruído, era esperado que ele

crescesse bem mais. Por outro lado, o bom comportamento do esquema GSW mostra que os valores de parâmetros para ele estão grandes demais, e que podem ser ajustados para diminuir o tempo de execução.

Não foram inclusos testes com L maiores, tendo em vista que não há aumento na profundidade das operações, portanto, não obtendo-se um crescimento expressivo do ruído.

8. Conclusão

A criptografia homomórfica é uma área da criptografia que surgiu a partir da Teoria dos Números, mas que se expandiu para a área dos Reticulados, e que apresenta muito potencial para todo tipo de aplicações, desde uso no treinamento de modelos de Inteligência Artificial, até a recuperação privativa de dados (PIR).

Neste relatório, foi realizada um estudo comparativo entre dois esquemas criptográficos homomórficos, o DGHV e o GSW, baseados em Teoria dos Números e Reticulados, respectivamente, com o intuito de analisar o comportamento de ambos esquemas sob a ótica da PIR, em especial seu custo de tempo.

O esquema DGHV apresentou melhor comportamento durante os testes, liderando quase todas as métricas estudadas, como tempo de execução e consumo de RAM, entretanto, o esquema GSW demonstrou um melhor crescimento de ruído. Em síntese, os experimentos mostraram que o DGHV apresentou desempenho significativamente superior em termos de tempo e uso de memória para a tarefa de PIR. O esquema GSW, embora possua propriedades homomórficas mais ricas e um comportamento de ruído mais controlado, sofre com o elevado fator de expansão e com o custo computacional associado às operações matriciais.

Como limitação deste estudo, destaca-se que todas as implementações foram intencionalmente ingênuas e executadas em SageMath, o que não reflete uma implementação otimizada em C/C++ com FFT, NTT ou técnicas modernas de compressão de criptogramas. Além disso, o fator de expansão do GSW poderia ser reduzido com variantes como FHEW/TFHE ou com técnicas de key switching.

Ainda assim, o GSW demonstra potencial como componente auxiliar em esquemas baseados em RLWE, especialmente em cenários nos quais o produto externo e operações estruturadas proporcionam vantagens substanciais. Assim, embora o DGHV se mostre mais adequado para PIR no ambiente experimental estudado, há espaço para soluções híbridas ou alternativas baseadas em reticulados que explorem as vantagens estruturais do GSW.

9. Referências

References

[Acar et al. 2018] Acar, A., Aksu, H., Uluagac, A. S., and Conti, M. (2018). A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4).

[Biasioli et al. 2024] Biasioli, B., Kirshanova, E., Marcolla, C., and Rovira, S. (2024). A tool for fast and secure lwe parameter selection: the fhe case.

[Chor et al. 1995] Chor, B., Goldreich, O., Kushilevitz, E., and Sudan, M. (1995). Private information retrieval. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 41–50.

[Doan et al. 2023] Doan, T. V. T., Messai, M.-L., Gavin, G., and Darmont, J. (2023). A Survey on Implementations of Homomorphic Encryption Schemes. *The Journal of Supercomputing*, 79:15098–15139.

[Ducas and Micciancio 2015] Ducas, L. and Micciancio, D. (2015). Fhew: Bootstrapping homomorphic encryption in less than a second. pages 617–640.

[Gentry et al. 2013] Gentry, C., Sahai, A., and Waters, B. (2013). Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Proceedings of the 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2013)*, volume 7881 of *Lecture Notes in Computer Science*, pages 75–92. Springer.

[Howgrave-Graham 2001] Howgrave-Graham, N. (2001). Approximate integer common divisors. In *Revised Papers from the International Conference on Cryptography and Lattices*, CaLC '01, page 51–66, Berlin, Heidelberg. Springer-Verlag.

[Kushilevitz and Ostrovsky 1997] Kushilevitz, E. and Ostrovsky, R. (1997). Replication is not needed: single database, computationally-private information retrieval. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 364–373.

[Pereira and Morais 2021] Pereira, H. and Morais, E. (2021). *Introdução à criptografia completamente homomórfica com implementação em Sage*, pages 1–50.

[Stern 1998] Stern, J. P. (1998). A new efficient all-or-nothing disclosure of secrets protocol. ASIACRYPT '98, page 357–371, Berlin, Heidelberg. Springer-Verlag.

[van Dijk et al. 2010] van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. In *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'10, page 24–43, Berlin, Heidelberg. Springer-Verlag.