INTERFACE TECNOLÓGICA

A IMPORTÂNCIA DO TEOREMA FUNDAMENTAL DA ARITMÉTICA NA CRIPTOGRAFIA DE CHAVE PÚBLICA

THE IMPORTANCE OF THE FUNDAMENTAL THEOREM OF ARITHMETIC ON PUBLIC-KEY CRYPTOGRAPHY

Bruna Pissará Gonçalves – pissara2@gmail.com
Faculdade de Tecnologia (Fatec) – Taquaritinga – SP – Brasil

Matheus Simões Minguini – minguinimatheus@gmail.com
Instituto Federal de Educação, Ciência e Tecnologia (IFSP) – São Carlos – SP – Brasil

DOI: 10.31510/infa.v17i2.1038
Data de publicação: 18/12/2020

RESUMO

O presente trabalho teve como objetivo apresentar a importância da aplicabilidade do teorema fundamental da aritmética na criptografia de chave pública, no qual seu papel é imprescindível na segurança da informação devido a dificuldade em decompor números compostos em fatores primos. Para isso, o estudo baseou-se em revisões bibliográficas com demonstrações matemáticas do postulado e da infinitude dos números primos, bem como um algoritmo desenvolvido pelos autores que realiza a fatoração de dois números distintos em suas quantidades de bits, evidenciando assim, que tal operação computacional é onerosa e não determinística. Nesse sentido, a pesquisa apresenta empiricamente através de experimentos e demonstrações, o quão laborioso seria a tentativa de obtenção inapropriada dos dados criptografados utilizando o algoritmo de chave assimétrica, salientando que, a computação une-se à matemática para garantir a impraticabilidade na decriptação de uma chave pública, garantindo dessa forma, a segurança e a integridade da informação no mundo digital.

Palavras-chave: Teorema, Algoritmo, Aritmética, Criptografia, Chave-pública, Fatoração.

ABSTRACT

This paper aims to present the empirical importance of the fundamental theorem of arithmetic on public-key cryptography, where its role is indispensable in information security due to the difficulty of factoring a compound number on its prime factors. To achieve its goal, the research relies on bibliographic reviews with mathematical proofs of the theorem, as well as the prime numbers being infinite. In order to have experiments around that, the authors drew up an algorithm to decompose two numbers with different amounts of bits, highlighting by doing so, that this is an extensive and non-deterministic operation. In this regard, the research presents on a practical and mathematical matter how laborious would be the attempt to obtain inaproprial access to classified and encrypted information when it is being kept secured by an asymmetrical algorithm, emphasizing that, computing joins mathematics to guarantee the impracticality in

decrypting a public key, thus guaranteeing the security and integrity of information in the digital world.

Keywords: Theorem, Algorithm, Arithmetic, Cryptography, Public-key, Factorization.

1 INTRODUÇÃO

A criptografia é o estudo de técnicas matemáticas e computacionais relacionadas com a segurança da informação que visam ocultar dados (SANT' ANA JÚNIOR, 2013).

Desde que o ser humano começou a comunicar-se percebeu que nem sempre era de sua vontade que terceiros acessem as informações transmitidas. Para esconder suas comunicações surgiram técnicas de ocultação para tornar inteligíveis as mensagens repassadas para outros de forma que somente o remetente e o destinatário seriam capazes de saber o conteúdo original. As técnicas de ocultação destas informações desenvolveram-se de forma a utilizar conceitos da matemática, inicialmente criados sem finalidade prática mas que encaixam-se perfeitamente à segurança da informação (SANT' ANA JÚNIOR, 2013).

A criptografia de chave pública baseia-se no teorema fundamental da aritmética, o mesmo postula que todo número composto pode ser representado em seus fatores primos. O maior desafio, contudo, está em fatorar números grandes, pois não existe um padrão na sequência dos números primos (SOUZA, 2018).

Baseado nisso, o objetivo deste trabalho é evidenciar a significância obtida pelo uso prático do teorema fundamental da aritmética na criptografia de chave pública, através de revisões bibliográficas, apresentação de deduções matemáticas, bem como um algoritmo relativamente performático elaborado pelos autores com intuito de simular a decomposição numérica de dois números diferentes em quantidades de bits, expondo assim, de maneira empírica, a complexidade e inviabilidade computacional da fatoração prima de um número composto. Espera-se, então, salientar a importância da matemática em fusão com a computação, de modo que garanta a segurança e integridade dos dados.

2 FUNDAMENTAÇÃO TEÓRICA

A presente seção transcorrerá, inicialmente, sobre os números primos e o teorema fundamental da aritmética. Logo na sequência, abordará sobre a criptografia de chave pública que tem como base os dois primeiros tópicos abordados.

2.1 Os números primos e o Teorema Fundamental da Aritmética

Alguns números podem ser decompostos em números inteiros menores. Por exemplo o 4 pode ser dividido em dois lotes de 2. Entretanto, alguns números não podem ser decompostos dessa maneira e possuem quatro divisores: *-1, 1, -p* e *p,* sendo *p* o próprio número a ser dividido. Por exemplo, o número 13 não pode ser dividido por nenhum outro exceto por 1 e 13. Tais números indivisíveis são chamados de números primos (GOLDSMITH, 2016).

Os gregos evitavam lidar com o conceito de infinito, pelas dificuldades que esse conceito sempre causava, assim na Proposição 20 do Livro 9 dos Elementos de Euclides, demonstrou-se que dado qualquer conjunto de primos, sempre existe um número primo fora deste conjunto, o que significa dizer que o conjunto de números primos é infinito (Demonstrado no item 2.1.2) (ÁVILA, 2010). Crilly (2017) salienta que não existe fórmulas para determinar se um número é primo, e aparentemente não existe um padrão no aparecimento deles entre os números inteiros.

Souza (2018) informa que os números primos formam os blocos básicos que constroem todos os números, sabendo que todo número composto pode ser transformado em números menores, como no caso do 4, que pode ser representado como sendo $2 \cdot 2$ $ou\ ainda,$ 2^2. É comum dizer que os primos são os átomos da Matemática. Como na Química, em que os átomos compõem toda a matéria, os primos compõem todos os números compostos e é isso que prova o Teorema Fundamental da Aritmética.

De acordo com Coutinho (2005), o teorema nos diz duas coisas importantes: primeiramente, todo inteiro pode ser escrito como potências de primos (Demonstrado no item 2.1.1). Além disso, só há uma escolha possível de primos para a decomposição de um inteiro informado.

O teorema foi primeiramente publicado no livro "Os Elementos" de Euclides (300 a.C), o matemático insistia que qualquer afirmação matemática devia receber uma prova lógica antes de ser considerada verdadeira. A obra possui uma inestimável importância para a ciência pois foi a precursora no uso sistemático de diagramas, junto com um uso limitado de símbolos e uma grande dose de lógica (STEWART, 2019). Contudo, de acordo com Souza (2018) sua primeira demonstração correta e completa foi feita por Gauss, e publicada em 1801, na obra *Disquisitiones Arithmeticae*.

2.1.1 Demonstração de que todo inteiro pode ser escrito como potências de primos

O Teorema Fundamental da Aritmética postula que todo inteiro pode ser escrito como produto de números primos.

Millies e Coelho (2006) demonstram tal conjectura utilizando o Princípio de Indução, dessa maneira:

Para $a = 2$, o enunciado é verdadeiro, já que 2 é, ele próprio, um número primo. Suponha-se agora que o resultado seja verdadeiro para todo inteiro b, tal que $2 \leq b < a$. Deve-se mostrar então que isso vale para a.

Se a é primo, a conjectura está demonstrada. Caso contrário, a admite um divisor positivo b, tal que $1 < b < a$. Isto é, $a = bc$, e também $1 < c < a$. Pela hipótese da indução, b e c podem ser escritos como produto de primos, na forma:

$$b = p_1 \cdots p_k ,$$

$$c = q_1 \cdots q_k$$

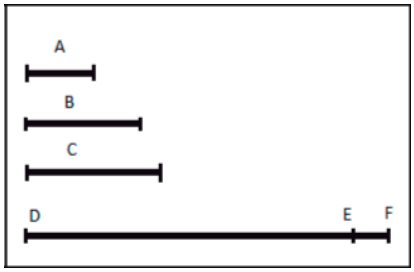
Substituindo, temos que $a = p_1 \cdots p_k \cdot q_1 \cdots q_k$, e o resultado também vale para a.

2.1.2 Demonstração da infinidade dos números primos

De acordo com Ribenboim (2012), o teorema entendido como "existem infinitos números primos" utiliza um raciocínio indutivo juntamente com o método da prova por absurdo. O método, segundo Juliani (2008), consiste em supor verdadeira a negação de uma tese e se por meio de um raciocínio dedutivo for encontrada uma contradição (absurdo), a tese se torna a negação da negação, ou seja, verdadeira.

Sejam, então, A, B e C os números primos propostos. Diga-se que não há mais números primos que A, B e C (EUCLIDES, 2009).

Figura 1 – Representações sobre primos



Fonte: Euclides, 2009 (adaptado)

Para tanto, tome-se o menor número medido por *A, B, C*, chamando-o de *DE*, e acrescentando-se a *DE* a unidade *EF*, como indica a figura 1. Então, *DF* é primo ou não. Supondo que seja primo, então terão sido encontrados os números primos *A, B, C* e *EF*, que são mais que *A, B,* e *C*.

Suponha-se, entretanto, que *EF* não seja primo; então, é medido por algum número primo: seja, assim, medido por um número primo *G*. Diga-se que *G* não é o mesmo que nenhum dos números *A, B, C*, pois, se fosse possível, seria. Mas *A, B, C*, medem o DE, então G medirá também o DE. E também mede o EF; e G, sendo um número, medirá também a unidade restante DF; o que é um absurdo. Logo, G não é o mesmo que nenhum dos números anteriores A, B, C. E foi suposto que era primo. Por conseguinte, hão sido achados mais números primos do que a quantidade proposta dos números A, B e C (EUCLIDES, 2009).

2.2 Criptografia de chave pública

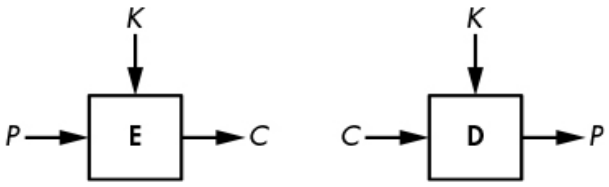
As pessoas vêm usando mensagens cifradas há mais de 4 mil anos. Algumas técnicas envolviam trocar letras do alfabeto ou usar símbolos, mas muitas se baseavam na Matemática. Essa ciência dos códigos secretos (e da tentativa de decifrá-los) é conhecida como criptografia (GOLDSMITH, 2016).

De acordo com Souza (2018) criptografar é elaborar um processo ou uma "chave" que é usada para "misturar" a mensagem e só quem tem a "chave" é capaz de tornar a mensagem compreendida por quem a recebe.

Como pontua Aumasson (2018), quando uma mensagem está sendo encriptada, *plaintext* é o termo usado para se referir à mensagem não encriptada e *ciphertext* a mensagem encriptada. Portanto, é correto dizer que um *cipher* é composto de duas funções: encriptação transforma um *plaintext* em um *ciphertext* e desencriptação transforma um *ciphertext* em um *plaintext.*

A figura 2 exemplifica essas funções:

Figura 2 - Dinâmica de funcionamento da encriptação e desencriptação

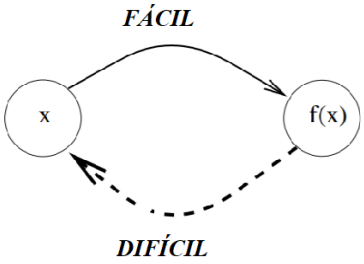


Fonte: Aumasson, 2018.

A função *E* que representa a encriptação recebe como entrada os valores *K* e *P*, produzindo como saída o *C* - o *ciphertext*. Existe também a função que representa a desencriptação, isto é, a função *D* recebe como entrada *K* e *C*, produzindo como saída *P* - p*laintext*. Nota-se que as duas funções fazem uso de *K*, que nesse caso representa a chave secreta (AUMASSON, 2018).

As funções *E* e *D* citadas acima são conhecidas como *one-way functions.* Como informa Goldreich (2004), em termos simples, essas funções são fáceis de calcular mas muito difíceis de desfazer. A ilustração abaixo mostra de forma abstrata uma *one-way function.*

Figura 3 - Dinâmica de funcionamento de uma *one-way function*



Fonte: Goldreich, 2004.

A criptografia está baseada na suposição de que tais funções existam. Afinal, usuários devem ser capazes de usar alguma informação de conhecimento unicamente deles e ainda sim, um atacante (que não possui essa informação privada) não deveria ser capaz de quebrar a função facilmente (GOLDREICH, 2004).

2.2.1 **Algoritmo de chave pública RSA**

Segundo Souza (2018), na década de 1970 dois cientistas da computação, Ronald Reivest e Adi Shami, com a ajuda do matemático Leonard Adleman, criaram um método de

criptografia RSA – o nome é um acrônimo para as iniciais dos autores. Com o RSA existem duas chaves: uma pública e uma privada. A pública pode ser usada para criptografar qualquer mensagem. Para "destrancar" a mensagem, o usuário usa a chave privada, sendo essa última mantida em segredo.

Abaixo, explica-se o algoritmo de chave pública RSA a fim de evidenciar como os números primos e a decomposição de números compostos são utilizadas no cotidiano da segurança digital.

O algoritmo, como mostra Paixão (2003), possui os seguintes passos:

Tabela 1 – Passos do algoritmo RSA

Passo	Descrição	Cálculo
1	Escolhe-se dois primos, chamados p e q	$p = 11, q = 13$
2	Calcula-se o produto de p e q, que chamará N	$N = p \cdot q = 143$
3	Calcula-se o $\varphi(N)$	$\varphi(N) = (p-1)(q-1) = 120$
4	Escolhe-se um e relativamente primo a $\varphi(N)$	$e = 7$
5	Calcula-se a inversa de e módulo $\varphi(N)$	$e^{-1}\ mod\ \varphi(N) = 103$

Fonte: Adaptado de Paixão (2003).

Após essa estapa, obtem-se as duas chaves:

1) Chave pública: *<N, e>,* ou seja, *<143, 7>*
2) Chave privada: <N, d>, ou seja, *<143, 103>*

Paixão (2005) ilustra o seguinte exemplo: em posse das duas chaves, suponha que um indivíduo chamado José queira receber uma mensagem do seu amigo Carlos. A mensagem é a letra 'T'. Para que isso aconteça, antes, as letras do alfabeto devem passar por uma etapa de pré-codificação, isto é, devem ser associadas a um número. Para o exemplo, as letras serão associadas aos seus respectivos números ordinais crescentes. Dessa forma, como o 'A' é a primeira letra do alfabeto, o seu número corresponderá ao 1, o 'B' será o 2 e assim sucessivamente. Porque o '*T'* corresponde a vigésima letra do alfabeto, então o transforma-se no seu número ordinal crescente correspondente: o *20*. Antes de iniciar a comunicação, é feito

todo o processo mencionado e o par de números que correspondem a chave pública é enviado a Carlos.

Então, Carlos com base na chave pública <143, 7>, encripta a mensagem *'T'*, gerando o *cyphertext C*, como segue:

$$C = T^e \, mod \, N = 20^7 \, mod \, 143 = 136$$

Envia-se então, a mensagem cifrada *C*. Ao receber a mensagem, José irá desencriptá-la aplicando a chave secreta, dessa forma:

$$C^d \, mod \, N = 136^{103} \, mod \, 143 = 20$$

Agora, basta encontrar a letra correspondente ordinalmente ao 20 no alfabeto, sendo essa, o *'T'*. A mensagem foi decriptada.

Como evidenciado, encontrar dois números primos é essencial para utilizar o sistema de criptografia de chave pública (LEMOS, 2010). Na prática, o código de criptografia de chave-pública só é quebrado se os dois primos que geraram o produto (chave pública) forem encontrados (COUTINHO, 2005).

3 PROCEDIMENTOS METODOLÓGICOS

Para a elaboração do artigo foi realizada pesquisa bibliográfica, partindo do levantamento de referências teóricas já analisadas e publicadas (GIL, 2007). O método pode ser entendido nas ciências como o conjunto de processos que o espírito humano deve empregar na investigação e demonstração da verdade (CERVO E BERVIAN, 1996).

O presente estudo trata sobre o Teorema Fundamental da Aritmética aplicado na criptografia de chave pública, e sua atuação na encriptação da informação contribuindo para a redução das chances de ataques e assim garantindo a integridade da informação.

Para isso a pesquisa apresenta a demonstração matemática do Teorema Fundamental da Aritmética e da infinidade dos números primos. Bem como a fatoração, através de um algoritmo desenvolvido pelos autores, de dois números compostos de tamanhos distintos expondo a mudança significativa no tempo de processamento de cada um, com o intuito de evidenciar a inviabilidade da fatoração prima.

3.1 Ambiente de performance do algoritmo

- 12Gb de memória RAM;
- Processador Intel(R) Core (TM) i5-6200U, quad-core;

- CPU de 2.8 GHz.

3.2 Fatoração

Primeiramente, é importante salientar que, como apontado anteriormente, os números primos são infinitos e além disso, não possuem um padrão em sua sequência. Desse modo, não existe outro modo de fatoração se não por tentativa com base nas permutações de possibilidades. Existem algoritmos que tornam tais comparações mais rápidas, contudo, mesmo eles a precisam fazer.

Desse modo, o tempo de fatorar um número é não determinístico, pois uma mudança na entrada inicial (os bits, por exemplo) aumentam significativamente a complexidade de fatoração e consequentemente, o tempo de execução.

Os quadros abaixo expõem singularmente os dois números que foram decompostos em seus fatores primos através do algoritmo desenvolvido pelos autores, a quantidade de dígitos, bits e bytes de cada um e, seus respectivos tempos de execução.

A primeira fatoração foi efetuada com número de 31 dígitos (100 bits e 12 bytes) e levou 15 segundos para fatorá-lo em seus fatores primos (Quadro 1).

Quadro 1 - Fatoração de um número com 100 bits.

Número usado	Dígitos	Bits	Bytes	Tempo
78945612302257826971009088070 60	31	100	12	15 segundos

Fonte: Os autores.

Já a segunda fatoração levou 31458 segundos (aproximadamente 8 horas, 43 minutos e 48 segundos) para fatorar um número de 33 dígitos, isto é 106 bits e 13 bytes (Quadro 2).

Quadro 2 - Fatoração de um número com 113 bits.

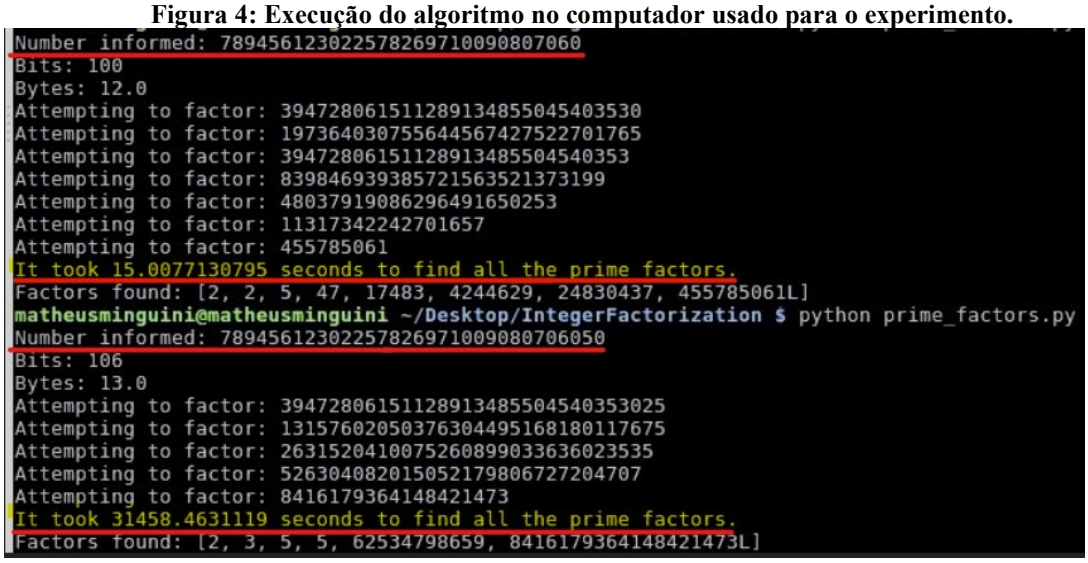
Número usado	Dígitos	Bits	Bytes	Tempo
78945612302257826971009088070 6050	33	106	13	31458.46 segundos

Fonte: Os autores.

Na figura 4 abaixo, pode-se visualizar o processamento do algoritmo. No terminal é impresso as seguintes informações durante o processamento: número a ser fatorado, os seus bits, bytes e o número que está sendo decomposto naquele momento. Após a operação, exibi-se então, o tempo de processamento e por fim, os fatores primos encontrados.

INTERFACE TECNOLÓGICA

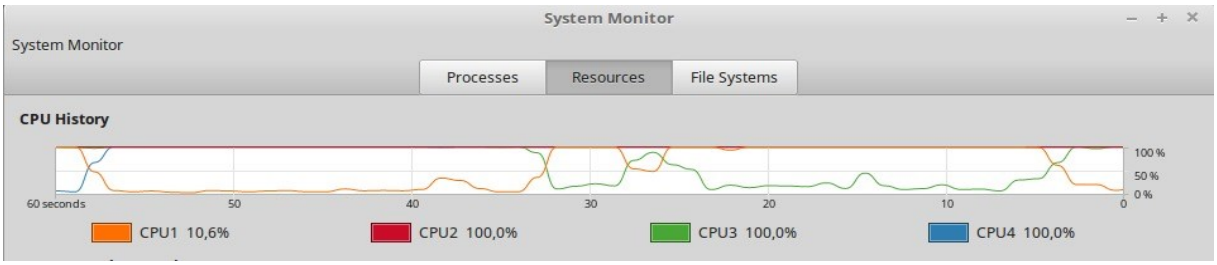
Figura 4: Execução do algoritmo no computador usado para o experimento.



Fonte: Os autores.

Os processadores do computador de verificação ficaram consideravelmente onustos durante a execução dos testes, chegando em sua maioria, aos 100% da capacidade de processamento, como pode ser verificado na figura 5.

Figura 5: Uso do processamento no computador de testes



Fonte: Os autores.

O algoritmo foi escrito na linguagem de scripts Python. O código-fonte do algoritmo está disponível no GitHub. O repositórios é *Open source* e disponível para a contribuição da comunidade.

Endereço do código fonte: https://github.com/MatheusMinguini/IntegerFactorization

4 RESULTADOS E DISCUSSÕES

As fatorações realizadas se deram a partir da execução de um algoritmo desenvolvido pelos autores, com o intuito de analisar a diferença que mudanças pequenas nas quantidades de bits mudam significativamente o tempo de processamento, fato esse que inviabiliza a tentativa de quebra criptográfica computacional a partir do método de fatoração de inteiros.

Para testar de forma empírica o tempo de execução das fatorações, dois números de diferentes bits foram escolhidos: o primeiro de 100 bits e o segundo de 106 bits.

A tabela abaixo aponta a diferença de tempo entre uma fatoração e outra:

Tabela 2 – Tempo entre uma fatoração e outra

	$\Delta\, Bits$	$\Delta\, Bytes$	$\Delta\, Tempo$ (%)
Fatoração I / Fatoração II	+6	+1	≈209.720

Da primeira para a segunda fatoração houve um aumento de 6 bits e de 209.720 % de tempo de execução.

Pode-se observar que o tempo de execução da decomposição numérica aumentou significativamente mesmo com uma pequena diferença de bits entre os números fatorados.

Além disso, como visto anteriormente, o procedimento de fatoração é algo oneroso para o equipamento que o está executando. No exemplo dos testes, a máquina utilizou aproximadamente 77.65% da sua capacidade processual.

Seguindo na prova empírica do quanto a execução de uma fatoração de inteiros é algo oneroso e inviável, pesquisadores realizaram experimentos parecidos: Aoki et. al (2010) publicou o recorde de fatoração de um número com 232 dígitos, isto é, 768 bits. Os pesquisadores relatam que a execução do algoritmo foi realizada usando centenas de máquinas e levou aproximadamente 2 anos. Conjecturavelmente, caso o mesmo conjunto de passos fora sido executado em um computador de um único processador AMD Opteron de 2,2 Ghz com 2 Gb de memória RAM, a decomposição prima teria demorado por volta de 15 mil anos. Os acadêmicos relatam que tais recordes não são efêmeros, pois o recorde anterior ao deles foi um número de 200 dígitos em 2005. É importante mencionar que a DigiCert (Entidade Certificadora americana) suporta RSA-2048 (porém chaves de 3072 e 4096 bits também estão disponíveis).

5 CONSIDERAÇÕES FINAIS

Os números primos, assim como apresentada a demonstração matemática durante este trabalho, são infinitos. Esse fato torna a tentativa de fatoração dos inteiros por força bruta inviável, afinal, não existe uma lista com todos os números primos para que se tente um a um.

Além disso, mesmo após mais de 2.000 anos de estudos e conjecturas, não encontrou-se um padrão no aparecimento desses números no conjunto dos números inteiros. Isso implica

que não existe uma progressão ou recorrência com um termo geral que forneça o n-ésimo primo. Em outras palavras, não é possível saber o próximo primo através de uma fórmula. O único modo de encontrá-lo é testando.

Dado que o acesso a uma lista dos números primos até um número bem grande é complexo e, extrapolar as permutações de possibilidades da fatoração é impossível, não resta outro modo de fatoração se não o computacional - com algoritmos que dado um número composto, devolvem os seus fatores primos.

Com isso, o presente trabalho teve como objetivo exibir conceitos fundamentais e mostrar empiricamente como um teorema postulado por Euclides e provado por Gauss tornou-se útil na segurança da informação através de algoritmos de chave pública. Para isso, um algoritmo foi escrito em Python pelos autores e a fatoração de dois números semelhantes em quantidade de bits foi realizada. Tal experimento objetivou a exposição da discrepância no tempo de processamento demandado, evidenciando, assim, de maneira prática o quão indeterminista essa operação é para a computação do começo do século XXI.

A criptografia de chave pública confia nessa indeterminação e postula que mesmo os computadores altamente performáticos demoram a executar essa fatoração. Por isso, nesse tipo de criptografia, dois primos são multiplicados e o produto dessa operação pode ser conhecido por todos (chave pública). Esse conhecimento pouco interfere na segurança criptográfica, afinal, para conseguir acesso à informação a partir do mesmo, precisa-se fatorá-lo a fim de descobrir quem são os dois primos que compõem a chave privada.

Em suma, para que ocorra uma quebra criptográfica é necessário fatorar o número (chave pública) em seus fatores primos (Teorema Fundamental da Aritmética) para encontrar as chaves privadas. Contudo, como não há uma maneira de antecipar o próximo primo candidato durante a quebra criptográfica e não existe uma lista completa dos números primos para tentar-se um a um, a informação trafega livremente e segura nos meios digitais.

REFERÊNCIAS

AOKI, K. et al. **FACTORIZATION IF A 768-BIT RSA MODULUS.** Version 1.4. Fevereiro de 2010.

AUMASSON, J. **Serious Cryptography.** São Francisco: No starch press, 2018.

ÁVILA, G. S. S. **Várias Faces da Matemática: Tópicos para licenciatura e leitura geral.** São Paulo: Blucher, 2010.

CERVO, A.L.; BERVIAN, P.A. **Metodologia científica**. 4.ed. São Paulo: Makron Books, 1996.

COUTINHO, S. C. **Números Inteiros e a Criptografia RSA.** 2 ed. Rio de Janeiro: IMPA, 2005.

CRILLY, T. **50 ideias matemáticas que você precisa conhecer**. São Paulo: Planeta, 2017.

EUCLIDES. **Os elementos. Tradução e introdução de Irineu Bicudo.** São Paulo: UNESP, 2009.

GIL, A.C. **Métodos e técnicas de pesquisa social.** São Paulo: Atlas, 2007.

GOLDREICH, O. **Foundations of Cryptography**. Cambridge: Cambridge University Press, 2004.

GOLDSMITH, M. **Do zero ao infinito (e além)**. São Paulo: Benvirá, 2016.

JULIANI, R. T. **O desejo do absurdo.** In: 1º Congresso de História das Ciências e das Técnicas e Epistemologia – UFRJ / HCTE, Rio de Janeiro, 2008.

LEMOS, M. **Criptografia, Números Primos e Algoritmos.** 4 ed. Rio de Janeiro: IMPA, 2010.

MILLIES, C, P; COELHO, S, P. **Números: uma introdução à Matemática.** 3 ed. São Paulo: Editora da Universidade de São paulo, 2006.

PAIXÃO, **Implementação e análise comparativa de variações do criptossistema RSA.** Dissertação (Mestrado em Ciência da Computação). Instituto de Matemática e Estatística, Universidade de São Paulo. São Paulo, 2003.

RIBENBOIM, P. **Números Primos: velhos Mistérios e novos recordes.** Rio de Janeiro: IMPA, 2012.

SANT'ANA JUNIOR, B. **Introdução a matemática aplicada à criptologia.** [s.n.], 2013.

SOUZA, H. M. **21 Teoremas matemáticos que revolucionaram o mundo**. São Paulo: Planeta, 2018.

STEWART, I. **Desbravadores da matemática: Da alavanca de Arquimedes aos fractais de Mandelbrot.** Rio de Janeiro: Zahar, 2019.