PROFESSOR DE MATEMÁTICA ONLINE
Revista eletrônica da Sociedade Brasileira de Matemática

PMO v.13, n.1, 2025
ISSN: 2319-023X
https://doi.org/10.21711/2319023x2025/pmo1309

Construções criptográficas clássicas com teoria dos números: além do RSA

Tiago Rossi Dias[1] Jerônimo Pellegrini

Resumo

Este trabalho pretende abordar construções criptográficas fundamentadas usando Teoria dos Números, mas diferentes do RSA, já bastante conhecido, precedendo estas construções com uma discussão de funções de mão única, que são parte fundamental da criptografia contemporânea, e terminando com uma proposta de aplicação em sala de aula. De início, aborda o conceito de criptografia de chave pública, dando como exemplo o criptossistema RSA. Apresenta os fundamentos da criptografia moderna – probabilidade desprezível e funções de mão única. Dá exemplos de funções de mão única: exponenciação modular, multiplicação de primos, quadrado modular e soma de subconjuntos. Apresenta o conceito de predicado *hard-core*. Expõe, de maneira detalhada, o uso de resíduos quadráticos em criptografia, incluindo a geração de números aleatórios com o gerador Blum-Blum-Shub, o criptossistema de Rabin e provas de conhecimento zero. Termina com a proposta de aplicação destes conceitos em sala de aula, na forma de uma atividade onde se desenvolve um leilão, usando quadrados modulares.

Palavras-chave: criptografia; teoria dos números; resíduos quadráticos; proposta didática

Abstract

This work aims to present cryptographic constructions based on Number Theory, but different from RSA, and this presentation is preceded by a discussion of one-way functions, which are a fundamental part of contemporary cryptography, leading to a proposal of a didactic application in classroom. Initially, the concept of public key cryptography is discussed, using as example the RSA cryptosystem. The fundamentals of modern cryptography are presented – negligible probability and one-way functions. Examples are given of one-way functions: modular exponentiation, multiplication of primes, modular squaring and subset sum. The concept of hard-core predicate is also presented. A detailed exposition of quadratic residues in cryptography follows, including the generation of random numbers with the Blum-Blum-Shub generator, the Rabin cryptosystem and zero-knowledge proofs. Finally a proposal of application for the classroom is presented, in the form of an activity where a secure auction is made possible using modular squaring.

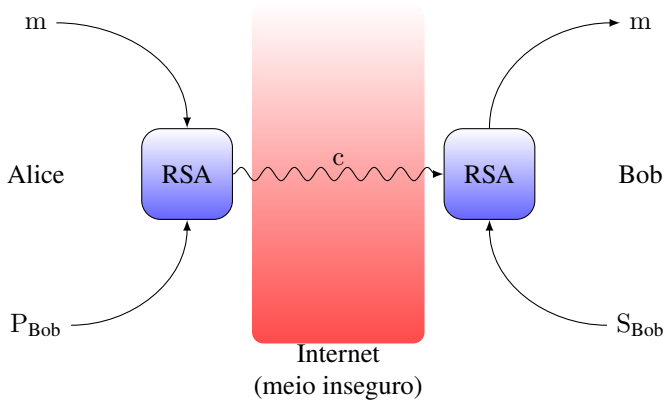
Keywords: cryptography; number theory; quadratic residues; didactic proposal.

[1] Parcialmente financiado pela Capes, como bolsista de mestrado do programa PROFMAT-UFABC.

1. Introdução

É comum na literatura de Teoria dos Números a presença do criptossistema RSA como exemplo de aplicação [7, 11, 6, 14, 15]. Não à toa, é uma aplicação surpreendentemente prática da teoria, e seu desenvolvimento é muito elegante.

O criptossistema RSA, cujo nome deriva das iniciais dos sobrenomes dos criadores (Rivest, Shamir e Adelman), é um algoritmo de *encriptação* que, a *grosso modo* falando, serve para que uma entidade A (usualmente chamada de "Alice" pelos criptólogos) possa enviar mensagens a uma entidade B (chamado de "Bob") sem que ninguém mais conheça seu conteúdo. Para isso, tanto Alice como Bob precisam ter *chaves criptográficas* – cada um tem uma *chave pública* e uma *chave privada*. Ambas as chaves são, no caso do RSA, números. As chaves públicas têm esse nome justamente porque são publicadas para o mundo todo. Já as chaves privadas são mantidas em segredo pelos seus detentores. Quando Alice quer mandar uma mensagem a Bob, ela usa um método para transformar essa mensagem em uma sequência de números. Em seguida usa a chave pública de Bob para transformar a mensagem em números aparentemente aleatórios e os envia para Bob. Quando Bob recebe a mensagem cifrada, ele consegue decifrá-la, porque sua chave privada permite fazê-lo (e somente com essa chave isso é possível). Na figura a seguir, m é a mensagem secreta; c é a mensagem encriptada; $\mathrm{P_{Bob}}$ e $\mathrm{S_{Bob}}$ são as chaves pública e secreta de Bob.



O funcionamento do RSA é detalhado na Seção 2. Este criptossistema, no entanto, não é a única aplicação de Teoria dos Números em Criptografia. E, ainda que não seja usada na criptografia pós-quântica [2] da maneira como se fazia na criptografia clássica, as aplicações na criptografia clássica ainda são interessantes por sua elegância e continuam tendo grande valor pedagógico, inclusive por permitir ilustrar de maneira simples um pouco do *modus operandi* da pesquisa em criptografia, com experimentos, probabilidades desprezíveis, e reduções entre problemas.

Neste artigo, expomos brevemente o conceito de função de mão única que fundamenta criptossistemas modernos, e, para isso, identificaremos funções que determinam a probabilidade de falha em criptossistemas (por exemplo, a probabilidade de decriptação por uma entidade não autorizada, sem o uso da chave). Como a probabilidade de falha depende de algum parâmetro de segurança, essa função tem como argumento um número (por exemplo, o número de bits usados para representar a chave). Nos interessará neste trabalho, portanto, a probabilidade assintótica de sucesso do adversário, que queremos que seja uma função "desprezível", ou seja, que tende a zero muito rapidamente à medida que o parâmetro de segurança cresce.

Apresentamos a seguir uma alternativa didática ao RSA usando resíduos quadráticos, o critério de Euler e o Teorema Chinês do Resto, que pode ser aplicada com alunos do 8º e 9º do Ensino Fundamental, mas

que também pode ser aplicada no Ensino Médio com poucas adaptações. Além disso, ilustraremos uma aplicação prática do conceito de homomorfismo de grupos, tendo portanto interessante valor pedagógico. Adicionalmente, apresentamos outras construções criptográficas fundamentadas no mesmo problema: o protocolo de Fiat-Feige-Shamir, para provas de conhecimento zero e uma aplicação simples de encriptação homomórfica.

2. O RSA

Entraremos agora em detalhes de funcionamento do criptossistema RSA, mencionado na Introdução. Relembramos o cenário: Alice quer enviar uma mensagem a Bob, mas não quer que outras pessoas possam ler a mensagem. Ela usará a *chave pública* de Bob para encriptar a mensagem e Bob usará sua *chave privada* para decifrar.

Antes da mensagem ser enviada, Bob criou suas duas chaves, pública e privada. Para isso:

- Escolheu aleatoriamente dois números primos grandes, p e q;
- Calculou $n = pq$;
- Escolheu uma número e, tal que $\mathrm{mdc}(e, \phi(n)) = 1$;
- A chave pública P_{Bob}, então, é o par (n, e);
- Determinou um número secreto $d \equiv e^{-1} \pmod{\phi(n)}$.
- A chave privada é o par $S_{Bob} = (n, d)$.

As duas transformações feitas (de m em c e depois novamente de c em m) pelo algoritmo RSA são: para encriptar, Alice calcula o menor c tal que

$$c \equiv m^e \pmod{n}.$$

Para decriptar, Bob usa sua chave secreta, que é o inverso da chave pública módulo $\phi(n)$:

$$\begin{aligned} m &\equiv c^d && \pmod{n} \\ &\equiv (m^e)^d && \pmod{n} \\ &\equiv m^{ed} && \pmod{n} \\ &\equiv m^1 && \pmod{n}. \end{aligned}$$

É claro que Bob escolherá, ao realizar este cálculo, o menor m natural que satisfaça a congruência, e não algum número qualquer da forma $kn + m$.

Quando o par de chaves é criado, a escolha de e é feita usando $\phi(n)$ porque queremos obter o inverso de e. É um fato conhecido que para todos $a, n \in \mathbb{N}$, $a^{\phi(n)} \equiv 1 \pmod{n}$. Assim,

$$\begin{aligned} m^{\phi(n)} &\equiv 1 \pmod{n} \\ m^{k\phi(n)} &\equiv 1 \pmod{n} \end{aligned}$$

Para obter m, multiplicamos os dois lados por m:

$$\begin{aligned} m^{k\phi(n)} m &\equiv m \pmod{n} \\ m^{k\phi(n)+1} &\equiv m \pmod{n} \end{aligned}$$

O objetivo, portanto, é que ed seja igual a $k\phi(n) + 1$.

$$\begin{aligned} ed &= k\phi(n) + 1 \\ ed - 1 &= k\phi(n) \\ ed &\equiv 1 \pmod{\phi(n)} \end{aligned}$$

Portanto, Bob escolherá e co-primo com $\phi(n)$, e d o inverso de e módulo $\phi(n)$ (o nero e é escolhido com $mdc(e, \phi(n)) = 1$ para que tenha inverso módulo $\phi(n)$, de outra forma não seria possível obter d).

3. Funções desprezíveis e funções de mão única

Esta seção apresenta dois conceitos fundamentais em criptografia [9]: o de função desprezível e o de função de mão única.

Um dos objetivos da criptografia é desenvolver métodos para manter informação em segredo, ou, mais realisticamente, minimizar a probabilidade de que segredos sejam descobertos por pessoas não autorizadas (que usualmente são denominadas simplesmente de "adversários"). Na qualidade de criptólogos, nosso objetivo (dentre outros) é tentar evitar que um adversário descubra um segredo.

Como não podemos garantir completamente que o adversário não poderá descobrir a chave, tentamos, então, impor que ele somente o consiga com probabilidade muito baixa. Assim ele terá que tentar por muito tempo (uma quantidade inviável de tempo) até acertar. Por exemplo, se uma chave secreta tem n bits, e se a única maneira de um adversário obtê-la é por tentativas ao acaso, sabemos que sua probabilidade de sucesso será $1/2^n$. Se esta probabilidade for muito baixa, aumentamos a quantidade de bits (de n para $n + k$), e a probabilidade passa a ser $1/2^{n+k}$. É claro que o adversário pode tentar uma a uma as chaves, mas quanto menor a probabilidade de acerto, maior é a expectativa de tempo necessário para quebrar a chave.

Neste exemplo, n é um *parâmetro de segurança*. Nem sempre a segurança de uma construção criptográfica será medida de maneira assim simples, a partir da quantidade de bits em uma chave. Essa probabilidade pode ser a de conseguir fatorar um número inteiro com k bits; a de determinar o logaritmo de um elemento em um grupo; a de determinar a menor base possível para um reticulado de dimensão finita[2]; ou a de sucesso em outras buscas em diferentes estruturas discretas. Mencionamos anteriormente a *probabilidade assintótica de sucesso do adversário*, que queremos que seja uma função "desprezível", tendendo a zero rapidamente à medida que cresce um parâmetro de segurança. O próximo passo é, portanto, definir claramente essas ideias (probabilidade como função desprezéivel, tarefas executadas "rapidamente", e a combinação desses conceitos em um só, o de "função de mão única").

Definição 1. Uma função f é desprezível se, para qualquer polinômio $p(n)$, existe $N \in \mathbb{N}$ tal que para todo natural $k > N$,

$$f(k) < \frac{1}{p(k)}.$$

Em outras palavras, uma função é desprezível se ela cresce mais lentamente que o recíproco de qualquer polinômio.

Como, para qualquer polinômio $p(n)$ de grau d existe um monômio x^c, $c > d$, que cresce mais rapidamente, para mostrar que uma função f é desprezível, basta provar que, para cada c, existe N tal que para todos os inteiros $k > N$ temos $f(k) < \frac{1}{k^c}$.

[2] O problema de encontrar a "menor" base para um reticulado é NP-completo [10], o que significa que, dado o que se sabe hoje sobre complexidade de algoritmos, este problema dificilmente será resolvido por algoritmo eficiente.

A função que demos inicialmente como exemplo, $f(n) = \frac{1}{2^n}$, é desprezível: seja $p(n) = n^c$ um monômio qualquer. Presumimos, para simplificar, que $n > 1$ e então, lembrando que tanto n como c são naturais, verificamos que

$$\begin{aligned} \frac{1}{2^n} &< \frac{1}{n^c} \\ n^c &< 2^n \\ n^c &< n^n && \text{(porque } n > 1\text{)} \\ c &< n \end{aligned}$$

E temos que para todo $n > c$, a desigualdade vale.

Isto é bastante relevante porque, ao trabalhar com um alfabeto binário, contendo apenas os símbolos $0, 1$, a quantidade de palavras possíveis com n caracteres é 2^n. Logo a probabilidade de se escolher uma em particular é $\frac{1}{2^n}$, sendo essa a probabilidade de descobrir uma chave secreta ao selecionar um valor aleatoriamente.

Para manter coerência com a literatura, utilizaremos a notação em inglês[3] para função desprezível, **negl**. Desta forma, **negl**(n) significará "alguma função desprezível de n".

A seguir, encontram-se duas proposições sobre as funções desprezíveis.

Proposição 1. *Sejam f e g duas funções desprezíveis. Então $f + g$ é uma função desprezível.*

Demonstração. Se f e g são desprezíveis, então existem $N_1, N_2 \in \mathbb{N}$ tais que $\forall n_1 > N_1$, $f(n_1) < \frac{1}{n^{c_1}}$ e $\forall n_2 > N_2$, $f(n_2) < \frac{1}{n^{c_2}}$. Seja $N = \max(N_1, N_2)$ e $c = \max(c_1, c_2)$. Então, para $n > N$, temos

$$f(n) + g(n) < \frac{1}{n^c} + \frac{1}{n^c} = \frac{2}{n^c} \leq \frac{n}{n^c} = \frac{1}{n^{c-1}},$$

e, portanto, $f(n) + g(n)$ é desprezível. □

Proposição 2. *Sejam $p(n)$ um polinômio positivo qualquer e $f(n)$ uma função desprezível. Então, o produto $p(n) \cdot f(n)$ é uma função desprezível.*

Demonstração. Seja x^c um monômio de grau c qualquer, logo $x^c \cdot p(x)$ é um polinômio. Como f é uma função desprezível, temos que existe N tal que $\forall k > N$,

$$f(k) < \frac{1}{k^c \cdot p(k)} \Rightarrow p(k) \cdot f(k) < \frac{1}{k^c},$$

portanto $p(n) \cdot f(n)$ é uma função desprezível. □

A seguir definiremos *função de mão única*, que é um conceito central na criptografia moderna. Informalmente, uma *função de mão única* é uma função f tal que (1) f pode ser computada rapidamente (ou seja, em tempo polinomial); e (2) qualquer algoritmo que obtenha x a partir de $f(x)$ (ou seja, que compute a inversa de f, ou mesmo que determine um elemento na pré-imagem de $f(x)$) deve obter sucesso com probabilidade muito baixa. Para definir rigorosamente estes conceitos, formalizaremos "rapidamente" e "obtenção de sucesso com baixa probabilidade".

[3] Em textos em Inglês, o termo é *"negligible"*.

Queremos que o adversário não consiga "rapidamente" encontrar uma chave secreta. Este conceito é formalizado com o nome de "complexidade de tempo".

Definição 2 (complexidade de tempo)**.** Se A é um algoritmo que recebe uma entrada e e produz uma saída s, então a *complexidade de tempo* de A é a quantidade de passos que o algoritmo dá, como função do tamanho da entrada e.

Por exemplo, um algoritmo receberá um vetor v com n números em ordem crescente, e precisa verificar se um determinado número específco x está no vetor. Se o algoritmo buscar sequencialmente x, fará no pior caso n comparações, porque precisará percorrer todos os n números do vetor. A complexidade do algoritmo é descrita como "linear em n", porque a função que descreve o tempo levado para terminar é uma função linear em n:

$$T(n) = n.$$

Um algoritmo melhor verificaria o elemento no meio do vetor; se não for x, ele poderá descartar metade do vetor (a da direita ou a da esquerda), porque sabe que o elemento do meio é maior, ou é menor, que x. O total de operações para o pior caso será bem menor que o do algoritmo anterior:

$$T(n) = \log_2(n),$$

e dizemos que este algoritmo tem complexidade de tempo logarítmica.

Se f e g são funções de $\mathbb{N}$ em $\mathbb{N}$, então dizemos que "g é $O(f)$" quando f cresce assintoticamente no mínimo tão rápido que g. Em outras palavras, existem N e k naturais tais que, para todo $n > N$,

$$kf(n) > g(n).$$

Por exemplo, dizemos que $\log_2(n)$ é $O(n)$, porque a função $f(n) = n$ cresce assintoticamente mais rápido que $g(n) = \log_2(n)$. Também é verdade que $f(n) = n$ é $O(n)$, que é $O(n^2)$, e que é também $O(n^{10})$.

Definição 3 (tempo polinomial)**.** Dizemos que um algoritmo tem *complexidade de tempo polinomial* quando a função que dá sua complexidade de tempo é menor ou igual que algum polinômio (como é o caso dos dois algoritmos descritos anteriormente) – ou seja, que a complexidade de tempo é $O(p(n))$ para algum polinômio p. Também dizemos que este algoritmo é *"rápido"*.

A seguir descrevemos o que significa "inverter rapidamente" uma função. Para isso, definimos um *experimento* (um procedimento hipotético, simulando uma tentativa de inverter a função) chamado `Invert`.

Experimento $\texttt{Invert}(A, f, n)$:

1. Escolha, com probabilidade uniforme, $x \in \{0,1\}^n$, e calcule $y = f(x)$.
2. O algoritmo A recebe uma cadeia de n uns ("111...1"), e também y. A saída do algoritmo x'.
3. O resultado do experimento é 1 se $f(x') = y$, e 0 caso contrário.

A sequência de uns é passada ao algoritmo para garantir que ele precise ler pelo menos n bits, de forma que o tempo mínimo usado por A seja $O(n)$.

Definição 4 (função de mão única)**.** Uma função f é dita *de mão única* se possui as seguintes características:

• Fácil de calcular: existe um algoritmo que calcula f em tempo polinomial;

- Difícil de inverter: se A é um algoritmo (possivelmente fazendo escolhas aleatórias) que executa em tempo polinomial, então a probabilidade de A encontrar x a partir de $f(x)$ deve ser uma função desprezível em n, que é um *parâmetro de segurança*, função de x. Ou seja, denotando por $\Pr[E]$ a probabilidade de um evento E, temos
$$\Pr\left[\texttt{Invert}(A, f, n) = 1\right] < \textbf{negl}(n).$$

Dado isto tudo, e reafirmando que funções de mão única estão no coração da criptografia contemporânea, prosseguimos com uma afirmação possivelmente desconcertante: não se sabe se existem, de fato, funções de mão única. Acredita-se que algumas funções sejam boas candidatas a função de mão única, mas esta crença se fundamenta em conjecturas, como por exemplo $P \neq NP$, ou mesmo no fato de, após décadas de tentativas, não ter havido sucesso na busca de algoritmos eficientes para invertê-las. Estas "boas candidatas" são as funções usadas como blocos básicos na construção de criptossistemas. A criptografia contemporânea se baseia, portanto, na *suposição* de que certas funções são de mão única – isso porque após muitas décadas de tentativas, matemáticos e cientistas da computação não conseguiram encontrar algoritmos que possam inverter rapidamente essas funções. Como não se conhece alternativa, as "candidatas a função de mão única" são usadas no desenvolvimento da criptografia. O criptólogo, portanto, sempre trabalha condicionando a validade de suas criações à veracidade da afirmação "existem funções de mão única"[4].

Para ilustrar o conceito de função de mão única, expomos nas quatro seções a seguir funções que são candidatas a funções de mão única: exponenciação modular, multiplicação de primos, quadrado modular e soma de subconjuntos.

3.1. Exponenciação modular

O primeiro exemplo de candidata a função de mão única que apresentamos é a exponenciação modular.

Seja p um número primo, e g um gerador de $\mathbb{Z}_p$. A função *exponenciação modular* é

$$\begin{aligned}\texttt{exp} : \mathbb{Z}_p &\to \mathbb{Z}_p\\ x &\mapsto g^x \pmod p.\end{aligned}$$

A exponenciação modular pode ser efetuada em tempo polinomial, mas não se conhece algoritmo eficiente para o *cálculo de índice* (ou seja, determinar x a partir de g^x) em um grupo, como $\mathbb{Z}_p$.

3.2. Multiplicação de primos

A multiplicação de primos também é um exemplo de candidata a função de mão única.

Seja $\mathbb{P}$ o conjuntos dos números primos. A função *multiplicação de primos* é

$$\begin{aligned}\cdot : \mathbb{P}\times\mathbb{P} &\to \mathbb{N}\\ (x, y) &\mapsto x \cdot y.\end{aligned}$$

A multiplicação de primos, como a de quaisquer inteiros, pode ser realizada rapidamente. Mas para a fatoração de um número em primos – ainda que se saiba que são exatamente dois fatores – não se conhece algoritmo eficiente.

[4]Neste aspecto, a criptografia lembra um pouco a Física, onde teorias são propostas (com base em observações da realidade), mas não são "provadas"; apenas pode eventualmente ocorrer de serem percebidas como inadequadas ou insuficientes para explicar fenômenos observados. A criptografia hoje propõe um "modelo" onde se pressupõe que certas funções são de mão única.

O uso desta função em criptossistemas pode se dar de muitas maneiras, mas em geral o par (x, y) seria um segredo, e o produto xy seria informação pública.

3.3. Quadrado modular

Outro exemplo de função de mão única é o quadrado modular. Este exemplo é mais longo, por abordar a dificuldade de extrair raízes quadradas módulo um número composto.

Seja $m = pq$, onde tanto p como q são primos da forma $4k + 3$. A função *quadrado módulo* m é

$$\begin{aligned} \mathtt{SQ}_m : \mathbb{Z}_m &\to \mathbb{Z}_m \\ x &\mapsto x^2 \pmod m. \end{aligned}$$

Essa função não é injetora e nem sobrejetora: sempre será verdade que $(-1)^2 = (+1)^2 \pmod m$ e, portanto, $(m-1)^2 = 1^2 \pmod m$. Além disso, como o domínio e o contradomínio têm o mesmo tamanho, ao não ser injetora, ela também não pode ser sobrejetora. Para um exemplo concreto, se $m = 3 \cdot 7$, temos em $\mathbb{Z}_{21}$,

$$\begin{aligned} 2^2 &= 4 \pmod{21} \\ 5^2 &= 4 \pmod{21}, \end{aligned}$$

mas não existe número inteiro x tal que $x^2 = 2 \pmod{21}$.

Calcular $x^2 \pmod n$ pode ser feito em tempo polinomial e existem algoritmos probabilísticos polinomiais que calculam a raiz quadrada módulo p, se p for um número primo. Se $n = pq$ e a fatoração de n é conhecida, é relativamente fácil calcular a raiz quadrada módulo n pelo Teorema Chinês do Resto. Na verdade, o problema de calcular a raiz quadrada módulo n é análogo ao de fatorar $n = pq$, se p e q forem primos.

Relembramos o critério de Euler.

Teorema 1. *Sejam p primo ímpar e a um inteiro coprimo com p. Então*

$$a^{(p-1)/2} \equiv \begin{cases} +1 \pmod p & \text{se, e somente se, } a \text{ é resíduo quadrático módulo } p, \\ -1 \pmod p & \text{se, e somente se, } a \textbf{ não } \text{é resíduo quadrático módulo } p. \end{cases}$$

Usaremos este teorema na demonstração do Lema 1, a seguir, que determina a razão pela qual exigimos que os números primos usados sejam da forma $4k + 3$.

Lema 1. *Sejam p, q dois primos da forma $4k+3$. Então para todo resíduo quadrático $y \in \mathbb{Z}$, com $y > 0$, há quatro raízes quadradas módulo pq, sendo pelo menos duas delas distintas. Ou seja, existem x_1, x_2, x_3, x_4 tais que*

$$\begin{aligned} (x_1)^2 &\equiv y \pmod{pq} \\ (x_2)^2 &\equiv y \pmod{pq} \\ (x_3)^2 &\equiv y \pmod{pq} \\ (x_4)^2 &\equiv y \pmod{pq}, \end{aligned}$$

com $x_1 \not\equiv x_2 \pmod{pq}$.

Demonstração. Sejam p, q, y como no enunciado. Pelo critério de Euler,

$$\begin{aligned} 1 &= y^{(p-1)/2} && \pmod p \\ y &= yy^{(p-1)/2} && \pmod p \\ &= y^{1+(p-1)/2} && \pmod p \\ &= y^{(p+1)/2} && \pmod p \\ &= y^{[(p+1)/4]^2} && \pmod p \end{aligned}$$

Isto caracteriza todas as raízes quadráticas módulo p (não deixamos de identificar nenhuma delas, porque o critério de Euler vale nos dois sentidos da implicação – é um "se, e somente se,").

Como $p = 3 \pmod 4$, sabemos que $(p+1)/4$ é inteiro, e

$$\pm\sqrt{y} = \pm y^{(p+1)/4} \pmod p$$

Repetindo o desenvolvimento para q, temos

$$\pm\sqrt{y} = \pm y^{(q+1)/4} \pmod q$$

Agora construímos um sistema modular com quatro equações,

$$\begin{aligned} n &= +y^{(p+1)/4} && \pmod p \\ n &= -y^{(p+1)/4} && \pmod p \\ n &= +y^{(q+1)/4} && \pmod q \\ n &= -y^{(q+1)/4} && \pmod q, \end{aligned}$$

que podemos resolver usando o Teorema Chinês do Resto. Obtivemos, portanto, exatamente quatro raízes quadradas de y módulo pq. Pelo menos duas são distintas, porque como p e q são primos ímpares, não é possível que

$$\begin{aligned} +y &\equiv -y \pmod p, \quad \text{ou que} \\ +y &\equiv -y \pmod q. \end{aligned}$$

□

Dissemos que o cálculo da função pode ser feito rapidamente, e o próximo Lema determina isso.

Lema 2. *O método usado para mostrar a existência das raízes quadradas módulo* pq *pode executado em tempo polinommial.*

Demonstração. O método consiste basicamente em enumerar os valores

$$\begin{aligned} &+ y^{(p+1)/4} && \pmod p \\ &- y^{(p+1)/4} && \pmod p \\ &+ y^{(q+1)/4} && \pmod q \\ &- y^{(q+1)/4} && \pmod q, \end{aligned}$$

o que implica em calcular quatro exponenciações modulares, e em seguida resolver o sistema modular. Há algoritmos para realizar esses cálculos me tempo polinomial (a descrição destes está fora do escopo deste trabalho). □

O Teorema 2 sugere que um algoritmo para determinar raizes quadradas módulo pq necessitaria da existência de um algoritmo para a fatoração desse módulo em p e q, reduzindo a segurança desta função de mão única à da função de multiplicação de inteiros, já abordada. Tendo somente o produto $m = pq$, a extração de raiz quadrada é difícil; tendo os dois primos separados, extrair raiz quadrada passa a ser fácil. Esta é uma demonstração típica de complexidade de problemas computacionais (em que a criptografia se apoia).

Teorema 2. *Seja* $m = pq$*, com* p, q *dois primos da forma* $4k + 3$*. Se, para todo* y*, resíduo quadrático módulo* m*, existir como computar rapidamente uma das raízes quadradas de* y *módulo* m *sem conhecer* p *e* q*, então existe um algoritmo para determinar rapidamente* p *e* q*.*

Demonstração. Suponha que exista um algoritmo R que calcule rapidamente raízes quadradas módulo m. Sejam $r \in \mathbb{Z}_m^*$, e $y \equiv r^2 \pmod{m}$, como no enunciado. Mostraremos que, se existe tal algoritmo (que possa calcular uma das raízes quadradas de y módulo pq), é possível obter p e q rapidamente usando-o.

Suponha, portanto, que o algoritmo R exista, e que $y = r^2$. Como são quatro raízes quadradas, e não sabemos se a raiz encontrada é igual a r, denotamos a raiz encontrada pelo algoritmo por s, ou seja:

$$\begin{aligned} r^2 &\equiv y \pmod{m} \\ R(y) &\equiv s \pmod{m} \\ s^2 &\equiv y \pmod{m} \end{aligned}$$

Então $s^2 \equiv r^2 \equiv y \pmod{m}$, mas isso não significa que necessariamente $s \equiv r \pmod{m}$.

Agora, observamos que

$$\begin{aligned} r^2 &\equiv s^2 \pmod{m} \\ r^2 - s^2 &\equiv 0 \pmod{m} \\ (r+s)(r-s) &\equiv 0 \pmod{m}, \end{aligned}$$

e $r - s$ deve ser divisível por p ou por q.

Como $r - s$ é divisível por p ou por q, sabemos que o MDC de $r - s$ e m não pode ser um. E, como $m = pq$, com p e q primos, só nos resta concluir que $\mathrm{mdc}(r - s, m)$ é igual a p ou a q.

Assim, usando o algoritmo R podemos construir outro algoritmo, F, que fatora m.

Algoritmo $F(m)$:

1. Escolha aleatoriamente $r \in \{2, \ldots, m-1\}$.
2. Seja $y = r^2$.
3. Usando R, obtenha *outra* raiz quadrada de y: $s = R(y)$.
4. Calcule $d = \mathrm{mdc}(r - s, m)$. O resultado será p ou q.
5. Para obter o outro primo, divida d pelo primo já calculado no passo (4).

Este método pode ser executado rapidamente (em tempo polinomial), porque depende apenas de computar as raízes quadradas, que por hipótese pode ser feito em tempo polinomial, e determinar o MDC de dois números menores que m, que pode também ser feito em tempo polinomial usando o algoritmo de Euclides. □

Damos um exemplo pra o Teorema 2 (com números pequenos para clareza): se $p = 5$ e $q = 13$, então $m = pq = 65$. Escolhemos $x = 44$, e obtemos $y = 44^2 \pmod{65} = 51$. Se houver método para obter rapidamente as raízes quadradas de 51 módulo 65, encontraremos

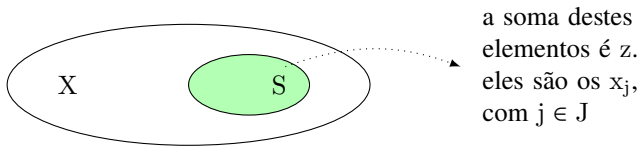
$$21, 31, 34, 44.$$

Inicialmente, não saberemos, entre as quatro, quais são $\pm r$ e quais são $\pm s$; mas isto é fácil de verificar: Tentamos $44 - 21 = 23$, mas $\mathrm{mdc}(23, 65) = 1$ e, portanto, estas devem ser $\pm r$, e calculamos $r - (-r) = 2r \pmod{65}$. Não nos servem. Tentamos agora $44 - 32 = 10$, e obtemos $\mathrm{mdc}(10, 65) = 5$, que é um dos dois primos. A obtenção do outro agora é trivial.

O uso do quadrado modular em criptografia exige que a fatoração do módulo ($m = pq$) seja secreta, mas que o módulo m seja público.

3.4. Soma de subconjuntos

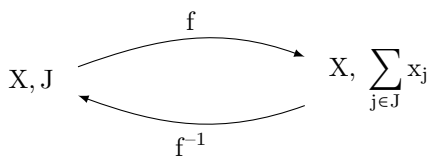
Como um complemento, apresentamos também a *soma de subconjuntos*, que é outra candidata a função de mão única. Seja um conjunto $X = \{x_1, x_2, \ldots, x_n\}$ de números naturais, e $S \subset X$. O problema consiste em, a partir de X e de z, a soma de S (ou seja, $z = \sum_{x \in S} x$, determinar S. Usualmente, formalizamos isto usando índices (o problema é determinar os índices j dos elementos de S).



Dado X e o conjunto J dos índices dos $x_j \in S$, a soma do subconjunto definido por J é

$$\texttt{somaS} : \mathbb{N}^n \times \mathbb{N}^n \to \mathbb{N}^n \times \mathbb{N}$$
$$\texttt{somaS}(x_1, x_2, \ldots, x_n, J) = (x_1, x_2, \ldots, x_n, \sum_{j \in J} x_j).$$

Observe que `somaS` dá como resultado não apenas a soma, mas o conjunto original e a soma (porque de outra forma o problema da inversão seria trivial). Na figura a seguir, f é fácil e f^{-1} é difícil.



Diferentemente das funções anteriores, a dificuldade de inversão desta não se baseia na dificuldade da fatoração de inteiros ou de cálculo de índice, e tal dificuldade se mantém na presença de computadores quânticos.

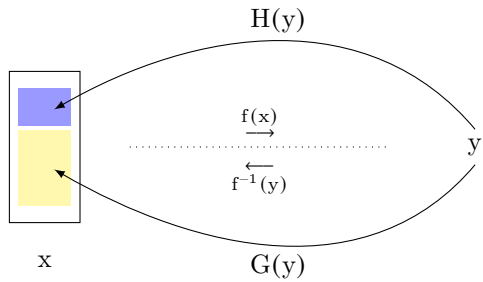
Ao usar esta função em criptossistemas, o conjunto J dos índices que definem S será privado; e o conjunto $\{x_1, x_2, \ldots, x_n\}$, assim como a soma, serão públicos.

3.5. Predicado hard-core

Candidatas a função de mão única são usadas para construir sistemas criptográficos: a construção do sistema é feita de forma que algo que não deva ser "permitido" exija o cálculo de inverso (ou determinação de elemento na pré-imagem) de uma função deste tipo. Por exemplo, decriptar uma mensagem ou produzir a assinatura digital em um documento sem a chave privada devem exigir a inversão de uma candidata a função de mão única.

No entanto, "não ser possível calcular elemento da pré-imagem" pode não ser suficiente.

Se f é função de mão única, sua definição determina que não seja fácil obter x a partir de $f(x)$. Nada impede, no entanto, que seja possível obter *alguma* informação sobre x. Por exemplo, pode ser difícil determinar completamente x, mas talvez seja possível determinar sua paridade. Desta forma, é útil definir que propriedade de x será difícil de obter a partir de $f(x)$.



Na figura, o valor x é representado como um retângulo. As duas partes são os bits de x, quando representado em binário; a parte superior (alguns dos bits) pode ser obtida usando uma função $H(y)$; a parte inferior (outros bits) podem ser obtidas usando $G(y)$. Se H é difícil de computar mas G é fácil, então H é predicado hard-core de f, mas G não é[5].

Definição 5. Uma função H é um predicado hard-core de uma função f se, para todo algoritmo probabilístico com tempo polinomial A, existe uma função desprezível **negl** tal que a probabilidade do algoritmo se comportar da mesma forma que $H(x)$ não se afasta de $1/2$ mais do que uma quantidade desprezível em n:

$$Pr[A(f(x)) = H(x)] \leq \frac{1}{2} + \mathbf{negl}(n),$$

onde n é o tamanho (em bits) de x, e onde $Pr[E]$ denota a probabilidade do evento E;.

Por exemplo, se presumirmos que o quadrado modular é de fato função de mão única, então obter a paridade de x a partir de x^2 é um predicado hard-core, mas não se pode dizer o mesmo de todos os bits de x. Isto é formalizado no Lema 3, cuja demonstração é dada no artigo original de Blum, Blum e Shub [3].

Lema 3. *Se $SQ_m(x)$ é função de mão única, então a função que determina a paridade de x é predicado hard-core de SQ_m.*

[5]Este exemplo simplifica a situação: H pode ser mais complexa do que "determinar uma parte dos bits de x". Por exemplo, poderia ser "x é múltiplo de 5?" ou "x é número perfeito?".

A paridade de x é determinada por seu bit menos significativo (quando este bit é zero, x é par; quando é um, x é ímpar). Assim, "determinar a paridade de x" é o mesmo que determinar este bit. O Lema determina que a função

$$H(x^2) = \begin{cases} 0 & \text{se } x \text{ é par} \\ 1 & \text{se } x \text{ é ímpar} \end{cases}$$

é um predicado hard-core da função quadrado modular $\mathtt{SQ}_m(x)$.

É interessante observar que, embora para $x \in \mathbb{N}$ a paridade de x^2 seja a mesma que a de x, isto não é verdade quando tomamos x e x^2 módulo m. Por exemplo, se $m = 253$ e $x = 105$, temos

$$\begin{aligned} 105^2 &= 11025 && \text{(ímpar)}, \\ 105^2 &\equiv 146 \pmod{253} && \text{(par)}, \end{aligned}$$

de outra forma o bit de paridade de x^2 seria sempre o mesmo do de x, e não poderia ser predicado hard-core.

Assim, usar o bit de paridade de x para codificar informação é uma prática segura. Para exemplos simples ou demonstrações em sala de aula, no entanto, usamos todos os bits de x, para simplificar a tarefa. E, de fato, nem sempre o predicado *hard-core* da função será usado (como, por exemplo, no caso do criptossistema de Rabin, descrito na Seção 4.2).

4. Criptografia com resíduos quadráticos

Uma das funções de mão única que apresentamos – o quadrado modular – é particularmente interessante quando aplicada no contexto da Criptografia.

Com o quadrado modular, é possível construir geradores de números aleatórios; sistemas de encriptação de mensagens; e provas de conhecimento zero, que são sistemas que permitem convencer um interlocutor de algo sem passar a ele qualquer informação.

Além de um gerador de números aleatórios (chamado de "Blum-Blum-Shub") e um sistema e encriptação (o "de Rabin"), esta seção expõe um protocolo para leilão seguro – todos usando a mesma função de mão única.

4.1. Geração de números aleatórios

É fundamental em criptografia a geração segura de números (ou bits). Por "segura", entendemos "dados $n-1$ bits gerados por um processo, o n-ésimo deve ser difícil de prever". Formalizamos esta noção na definição de *gerador pseudo-aleatório*, a seguir.

Definição 6. Seja ℓ um polinômio e G um algoritmo determinístico de tempo polinomial que, com uma entrada $s \in \{0,1\}^n$, retorne $G(s)$ de tamanho $\ell(n)$. G é um gerador pseudo-aleatório se:

1. $\forall n, \ell(n) > n$.
2. Para qualquer algoritmo probabilístico de tempo polinomial D usado para verificar se um número foi gerado por G, existe uma função desprezível **negl** tal que

$$|Pr(D(r) = 1) - Pr(D(G(s) = 1)| \leq \mathbf{negl}(n),$$

onde $\Pr[E]$ denota a probabilidade do evento E; r é um número de tamanho $\ell(n)$ escolhido aleatoriamente; e s é a *semente*, também escolhida aleatoriamente, e de tamanho n. A semente é um número inicial a partir do qual uma sequência de númreos aleatórios é gerada.

A definição basicamente diz que não deve ser possível que um observador consiga, em tempo polinomial, distinguir a sequência $G(s)$ de uma sequência aleatória, o que equivale a determinar o próximo número da sequência, dados os anteriores.

Apresentamos a seguir o gerador Blum-Blum-Shub, publicado em 1982 por Lenore Blum, Manuel Blum e Michael Shub [3].

Este gerador, também chamado de "Gerador $x^2 \pmod N$", usa a operação de quadrado modular, que conjecturamos ser uma função de mão única anteriormente. Para usá-lo, precisamos de dois números primos p e q, com $p \neq q$ e $p \equiv q \equiv 3 \pmod 4$. Calculamos então $N = pq$. É importante que p e q sejam de mesmo tamanho (que sejam representados com a mesma quantidade de bits), para dificultar a fatoração de N, e que a semente seja coprima com N.

O gerador calcula $G(s) = b_1, b_2, b_2, \ldots, b_n$, onde

$$\begin{aligned} s_0 &= s, \\ s_i &= s_{i-1}^2 \pmod N, \\ b_i &= s_i \pmod 2. \end{aligned}$$

Em cada operação, somente o bit de paridade ($s_i \pmod 2$) é usado, e não todos os bits de s_i, porque conforme o Lema 3, enunciado na Seção 3.5, este é um predicado hard-core da função quadrado modular (presumindo que seja função de mão única).

Como exemplo, escolhemos um módulo: $p = 11$ e $q = 47$ (ambos são da forma $4k+3$). Então $N = pq = 517$. Podemos escolher qualquer número entre 2 e 516 como semente – escolhemos 5. A sequência será:

$$\begin{array}{lrrcrcl} s_0 = & 5 & & & & & \\ s_1 = & 5^2 = & 25 & \equiv & 25 \pmod{517} & \to & b_1 = 1 \\ s_2 = & 25^2 = & 625 & \equiv & 108 \pmod{517} & \to & b_2 = 0 \\ s_3 = & 108^2 = & 11664 & \equiv & 290 \pmod{517} & \to & b_3 = 0 \\ s_4 = & 290^2 = & 84100 & \equiv & 346 \pmod{517} & \to & b_4 = 0 \\ s_5 = & 346^2 = & 119716 & \equiv & 289 \pmod{517} & \to & b_5 = 1 \\ s_6 = & 289^2 = & 83521 & \equiv & 284 \pmod{517} & \to & b_6 = 0 \end{array}$$

4.2. Criptossistema de Rabin

Um dos primeiros criptossistemas de chave pública foi criado em 1979 por Michael Rabin [12], logo após a publicação do RSA em 1978 [13].

Descrevemos a seguir o criptossistema de Rabin.

Seja $m = pq$, onde p e q são primos da forma $4k + 3$. Os dois primos devem ser suficientemente grandes para necessitarem de r dígitos para serem descritos na base 2.

- A chave secreta é o par (p, q).
- A chave pública é o inteiro m.

• A encriptação de uma mensagem n é $y = n^2 \pmod m$

A decriptação de uma mensagem y usando a chave privada (p, q) é feita obtendo as quatro raízes quadradas de y módulo pq, conforme descrito anteriormente. Observamos que y tem raiz quadrada módulo pq, porque foi obtido justamente elevando um elemento ao quadrado. Também percebemos que é possível obter as raízes rapidamente, sabendo a fatoração (p, q), que é necessária para construir o sistema modular e encontrar as quatro raízes.

Evidentemente, uma desvantagem desse criptossistema é que ao decifrar a mensagem podemos ter quatro soluções (teremos duas soluções apenas se $p \mid n$ ou $q \mid n$) e, a não ser que o contexto diga qual das soluções é a correta, alguma informação adicional deve ser transmitida de maneira a não deixar dúvidas quanto a mensagem original (por exemplo, um número indicando qual das soluções é a mensagem).

A segurança do criptossistema de Rabin se sustenta na segurança da função quadrado modular, uma vez que o criptossistema é, essencialmente, a aplicação direta daquela função.

Uma propriedade do criptossistema de Rabin é que a função de encriptação é um homomorfismo de mensagens em textos cifrados. Dizemos que se trata de um *criptossistema homomórfico*.

Definição 7. Um esquema de encriptação $E(k, m)$, onde k é uma chave e m a mensagem, é homomórfico se, para todas mensagens x e y, $E(k, x) \cdot E(k, y) = E(k, x \otimes y)$, onde $\cdot$ e $\otimes$ são operações quaisquer. A igualdade usada aqui significa "representam a mesma mensagem".

O criptossistema de Rabin é homomórfico, e neste caso há três observações importantes. Primeiro, no caso deste criptossistema, as duas operações são a mesma (tanto a operação $\cdot$ como a operação $\otimes$ na definição de encriptação homomórfica (Definição 7) são, neste caso, a multiplicação módulo m). Observamos também que para o criptossistema de Rabin duas mensagens são consideradas equivalentes se são congruentes módulo m, e portanto o criptossistema é homomórfico se $E(k, a)E(k, b) \equiv E(k, ab) \pmod m$. Finalmente, notamos que este criptossistema não usa o predicado *hard-core* da função quadrado modular – se usasse, não seria homomórfico (é, portanto, uma escolha de equilíbrio entre segurança e necessidade prática).

Teorema 3. *O criptossistema de Rabin é homomórfico – se duas mensagens encriptadas com a mesma chave pública forem multiplicadas, o resultado é a encriptação da multiplicação das duas mensagens:* $E(k, a)E(k, b) \equiv E(k, ab) \pmod m$.

Demonstração. Seja m a chave pública, como descrito anteriormente, e sejam a e b duas mensagens quaisquer. Então

$$\begin{aligned} E(k, a) &\equiv a^2 \pmod m \\ E(k, b) &\equiv b^2 \pmod m, \end{aligned}$$

e consequentemente,

$$\begin{aligned} E(k, a)E(k, b) &\equiv a^2b^2 \pmod m \\ &\equiv (ab)^2 \pmod m \\ &\equiv E(k, ab) \pmod m, \end{aligned}$$

o que conclui a demonstração. □

A propriedade de encriptação homomórfica é útil na construção de diferentes protocolos criptográficos, em que seja necessário realizar computações em geral sem que os participantes conheçam o conteúdo do que está sendo calculado (eleições, sistemas que usam informações médicas, e outros onde seja essencial preservar privacidade). O criptossistema de Rabin é homomórfico para apenas uma operação (e portanto denominado "parcialmente homomórfico"); há outros criptossistemas que são "completamente homomórficos", porque mantém a propriedade de homomorfismo para duas operações [1].

4.3. Provas de Conhecimento Zero

Informalmente, uma prova de conhecimento zero é uma interação em que uma entidade convence outra de um fato, mas sem dar elementos que permitam que essa segunda entidade faça o mesmo. Por exemplo, Alice prova para Bob que algo é verdade, mas sem que Bob obtenha qualquer informação que o ajude a convencer uma terceira pessoa do mesmo fato. O artigo de Manuel Blum, que inicialmente propôs a idéia de *provas de conhecimento zero*, trazia o provocativo título[6] "Como demonstrar um teorema sem que ninguém mais consiga fazê-lo"[4].

O protocolo de Fiat-Feige-Shamir foi desenvolvido [8] por Uriel Feige, Amos Fiat e Adi Shamir em 1988, e permite que uma entidade P ("Peggy") prove a um verificador V ("Victor") que conhece um certo segredo s, sem que V possa repetir esse procedimento para outros – ou seja, Peggy prova para Victor que conhece o segredo, mas sem que Victor possa reproduzir essa prova para outros. De fato, Victor não pode sequer provar que interagiu com Peggy (daí o nome "prova de conhecimento zero" – a quantidade de informação passada de Peggy para Victor é zero, e mesmo assim ela o convence de que conhece o segredo).

Novamente sorteamos p, q, dois primos, e determinamos $N = pq$.

Peggy tem como segredo os números $s_1, \ldots, s_k$, todos co-primos com N. Victor recebe os números $v_i \equiv s_i^{-2} \pmod N$, $1 \leq i \leq k$ (que são os inversos de $s^2 \pmod N$), e deve verificar se Peggy realmente possui o segredo. Para tal, são executados os seguintes passos:

- Peggy escolhe um inteiro aleatório r e envia a Victor $x \equiv r^2 \pmod N$;
- Victor manda para Peggy uma cadeia de números $b_1, \ldots, b_k$, onde cada b_i é escolhido equiprovavelmente em $\{0, 1\}$ (ou seja, é uma sequência aleatória de bits, escolhida equiprovavelmente dentre todas as sequências possíveis com comprimento k);
- Peggy devolve a Victor o número y tal que $y \equiv r \cdot s_1^{b_1} \cdot s_2^{b_2} \cdots s_k^{b_k} \pmod N$. Isso significa que cada bit b_i, escolhido aleatoriamente por Victor, determina se s_i será ou não computado no produto y, já que $s_i^0 = 1$ e $s_i^1 = s_i$;
- Victor verifica que $x \equiv y^2 \cdot v_1^{b_1} \cdot v_2^{b_2} \cdots v_k^{b_k} \pmod N$.

A sequência é repetida várias vezes, mudando o inteiro aleatório r a cada repetição.

Percebe-se que, para $k = 1$, Peggy deve informar r ou $r \cdot s_1$, dois números cujo quociente é uma raiz quadrada módulo N de v_1.

Teorema 4. *Se Victor escolhe k para o tamanho da cadeia de números e exige que o esquema de Fiat-Feige-Shamir seja repetido t vezes (ou seja, os parâmetros de segurança são t e k), então a probabilidade de um adversário convencer Victor sem, de fato, ter o segredo é desprezível em $t \cdot k$.*

[6] O título original é "How to Prove a Theorem So No One Else Can Claim It".

Demonstração. Para $k > 1$, é fácil para Peggy calcular y se ela possui o segredo e o único conhecimento que Victor consegue é uma raiz de um número da forma $x \equiv y_2 \cdot v_1^{b_1} \cdot v_2^{b_2} \cdots v_k^{b_k} \pmod N$, o que não é suficiente para descobrir o segredo $s_1, \ldots, s_k$. Se Eva, que não possui o segredo, quiser se passar por Peggy, precisará adivinhar qual a sequência de números $b_1, \ldots, b_k$. Victor pedirá antes de enviar x, pois como não sabe calcular o inverso módulo N das raízes quadradas de $v_1, \ldots, v_k$, ela precisará escolher $x \equiv y^2 \cdot v_1^{b_1} \cdot v_2^{b_2} \cdots v_k^{b_k} \pmod N$ e retornar a Victor o valor y, que passará no teste. Como há 2^k possibilidades para a sequencia $b_1, \ldots, b_k$, ao repetir esse esquema t vezes, a probabilidade que Eva engane Victor será de $1/(2^{kt})$, que é desprezível. □

5. Aplicação em Sala de Aula: um leilão com quadrados modulares

O objetivo desta aula é motivar os alunos e mostrar uma aplicação da álgebra abstrata discreta ao 8^o e 9^o anos do Ensino Fundamental.

O processo descrito abaixo funciona para leilões e licitações. A diferença é que, enquanto no leilão ganha quem der o maior lance, na licitação ganha quem ofertar o menor preço.

Antes do jogo ser feito, é necessário explicar aos alunos alguns conceitos de aritmética modular, mostrar como funciona o resto das divisões de números inteiros e que esse resto é importante em um grande número de situações, visto que a maioria dos alunos tem por hábito efetuar a divisão de números racionais (utilizando a vírgula) mesmo quando não convém ao problema. Nesse ponto pode ser interessante até mesmo o uso de calculadora, visto que eles ainda precisam pensar no algoritmo da divisão para conseguir o resto da divisão.

Ao trabalhar essa revisão das divisões de números naturais, podemos relacionar o conteúdo com a habilidade da BNCC [5] *(EF07MA04) Resolver e elaborar problemas que envolvam operações com números inteiros*. Outras habilidades estão relacionadas com a atividade, principalmente *(EF08MA06) Resolver e elaborar problemas que envolvam cálculo do valor numérico de expressões algébricas, utilizando as propriedades das operações*, quando apresentamos a expressão que deve ser calculada para apresentar o valor a ser publicado. E, por fim, *(EF09MA06) Compreender as funções como relações de dependência unívoca entre duas variáveis e suas representações numérica, algébrica e gráfica e utilizar esse conceito para analisar situações que envolvam relações funcionais entre duas variáveis*, para mostrar que a segurança depende do conceito criptográfico de função de mão única, apresentado anteriormente nesse artigo.

Os alunos são divididos em grupos e a cada grupo é dado um mesmo número arbitrário C de créditos. Esses créditos poderão ser usados para comprar pontos em um leilão aberto. Após um número R de rodadas, a equipe que tiver mais pontos ganhará o jogo.

A tática deve ser que cada grupo dê o maior lance possível, mas se gastar todos os seus créditos de início não terá como ganhar os lances posteriores. Cada grupo deverá apresentar seu lance de forma pública, mas se os outros grupos souberem o lance dos concorrentes antes de divulgar o seu, podem alterar seu lance. Para evitar isso, cada grupo irá criptografar seu lance usando o seguinte processo:

1. Cada grupo escolhe seu lance L em segredo.
2. Os grupos calculam $H_i = L_i^2 \pmod N$, sendo i o número de cada grupo, N é um número fornecido previamente pelo professor e $N = pq$, com p e q números primos desconhecidos pelos alunos.
3. Cada grupo apresenta H_i para os demais.
4. Após todos conhecerem H_i , todos revelam L_i e podem verificar que $H_i = L_i^2 \pmod N$.

Alguns cuidados devem ser tomados com a escolha dos números. Por exemplo, $N > 2C$, pois algum grupo poderia dar uma lance L e dizer que seu lance original era $N - L$ pois $L^2 = (N - L)^2 \pmod N$.

Também pode acontecer de um grupo testar vários valores para descobrir qual lance seu oponente mandou. Para evitar isso, considere que o lance máximo L tem M casas decimais. O grupo escolhe um número Y e calcula $H = (Y \cdot 10^M + L)^2 \pmod N$. Esse processo aumenta um pouco os cálculos, mas torna quase impossível descobrir por tentativa e erro o valor L escolhido. Ao apresentar o valor $Y \cdot 10^M + L$ para verificação, os M últimos dígitos corresponderão ao lance L.

Exemplificamos usando $N = 437$. A fatoração de N é $19 \cdot 23$, mas não será fornecida aos alunos.

Serão feitos quatro grupos e cada grupo receberá $C = 100$ créditos para gastar em $R = 4$ rodadas. Cada um dos grupos deverá criar uma estratégia para vencer. Após a primeira rodada, os lances pensados por cada grupo e os valores H calculados são:

	Grupo 1	Grupo 2	Grupo 3	Grupo 4
L	25	35	10	40
Y	0	0	3	0
$Y \cdot 10^3 + L$	25	35	3010	40
H	188	351	216	289

Apenas o grupo 3 viu necessidade de usar o método alternativo pois $10^2 = 100 \pmod{437}$ e seria fácil para os demais descobrirem seu lance. Quando todos revelarem seus lances, o grupo 4 ganhará essa rodada e a segunda rodada começará. Lembrando que, após essa rodada, todos os grupos ainda dispõem de 100 créditos para a compra, com exceção do grupo 4 que possui apenas 60 créditos.

Se houver a preocupação com a possibilidade de dois grupos efetuarem o mesmo lance (e isso seria imediatamente perceptível a todos), pode-se sortear números aleatórios W_i largamente espaçados, publicamente conhecidos, para cada grupo, e calcular $(W_i + L)^2$.

É interessante mencionar aos alunos que calcular as raízes módulo N, sendo N um número composto, é difícil se seus fatores forem grandes, mas que há métodos de calcular as raízes módulo p, sendo p um número primo.

Dependendo do andamento da atividade, pode ser interessante que o professor mostre, após o leilão, que ele (e só ele, que conhece os fatores p e q de N) consegue obter as raízes, sendo necessária uma autoridade confiável (no caso, o professor) para escolher p e q de forma a fornecer um N cuja fatoração seja desconhecida por todos.

Essa aula foi colocada em prática nos anos de 2018 e 2019 em uma escola da rede pública do Estado de São Paulo que participa do Programa Ensino Integral, com alunos de 8º e 9º do Ensino Fundamental, durante as aulas do componente Eletivas. Essas aulas tinham a característica de serem bem contextualizadas e a atividade foi planejada como maneira de gamificar uma disputa entre grupos diferentes. Cada grupo representava uma nação que utilizava os "créditos" do leilão como uma maneira de alocar recursos para obter novas tecnologias. A gamificação ajudava no engajamento, pois o grupo que não fizesse os cálculos corretamente a tempo seria desclassificado da rodada e perderia recursos que seriam utilizados em aulas futuras.

Para certificar-se do entendimento de todos, foi interessante fazer uma ou duas rodadas "abertas", na qual os grupos mostravam seus cálculos passo a passo, para que todos pudessem verificar se estavam realizando as operações corretamente. Nessas mesmas rodadas, aproveitei para mostrar que eu, como autoridade confiável, podia verificar os possíveis lances por conhecer a fatoração do número dado.

6. Conclusão

Este trabalho traz à luz uma antiga aplicação da Teoria dos Números em Criptografia, diferente do conhecido RSA. Abre-se espaço, assim, para discussões interessantes em aulas de Ensino Médio – algoritmos e protocolos criptográficos (que vão além do simples "encriptar uma mensagem"), comportamento assintótico de funções, existência de mais de uma raiz quadrada de um número, e a proximidade de construções criptográficas usando um mesmo fundamento: geradores de números aleatórios, sistemas de encriptação e provas de conhecimento zero são construídos a partir da mesma idéia, de quadrado modular. Todos estes sistemas são possíveis de implementar, especialmente em turmas onde alunos tenham aulas de introdução à programação, pois os algoritmos são pequenos e simples, adequados a tal cenário.

Referências

[1] ARMKNECHT, F. et al. A Guide to Fully Homomorphic Encryption. Disponível em: https://eprint.iacr.org/2015/1192.

[2] BERNSTEIN, D. J.; BUCHMANN, J.; DAHMEN, E. Post-Quantum Cryptography. [s.l.] Springer Science & Business Media, 2009.

[3] BLUM, L.; BLUM, M.; SHUB, M. Comparison of two pseudo-random number generators. Em: Advances in Cryptology. Boston, MA: Springer US, 1983. p. 61–78.

[4] BLUM, M. How to prove a theorem so no one else can claim it. Em: Proceedings of the International Congress of Mathematicians. Berkley, CA: USA, 1986. p. 1444-1451.

[5] BRASIL. Ministério da Educação. Base Nacional Comum Curricular. Brasília, 2018.

[6] BRESSOUD, D.; WAGON, S. A Course in Computational Number Theory. [s.l.] Key College Publishing, 2000.

[7] COUTINHO, S. C. Números inteiros e criptografia RSA. Rio De Janeiro: Impa, 2009.

[8] FEIGE, U.; FIAT, A.; SHAMIR, A. Zero-knowledge proofs of identity. Journal of Cryptology, v. 1, n. 2, p. 77–94, jun. 1988.

[9] KATZ, J.; LINDELL, Y. Introduction to modern cryptography. Boca Raton, Fl: CRC Press, 2021.

[10] MICCIANCIO, D.; GOLDWASSER, S. Complexity of Lattice Problems: A Cryptographic Perspective. Daniele Micciancio and Shafi Goldwasser. Boston, Massachusetts: Springer, 2012.

[11] NIVEN, I.; ZUCKERMAN, H. S.; MONTGOMERY, H. L. An Introduction to the Theory of Numbers. [s.l.] Wiley, 1991.

[12] RABIN, M. O. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.

[13] RIVEST, R. L.; SHAMIR, A.; ADELMAN, L. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM. v. 21, n. 2, p. 120-126, fev. 1978.

[14] SHOUP, V. A Computational Introduction to Number Theory and Algebra. [s.l.] Cambridge University Press, 2009.

[15] YAN, S. Y. Computational Number Theory and Modern Cryptography. Singapura: John Wiley and Sons, 2013.

Tiago Rossi Dias
Escola Estadual Doutor Antonio Ablas Filho - Santos/SP
<tiagorossidias@gmail.com>

Jerônimo Pellegrini
Universidade Federal do ABC
<jeronimo.pellegrini@ufabc.edu.br>

Recebido: 17/10/2023
Publicado: 31/03/2025