UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
INSTITUTO TÉRCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ANTÔNIO LACERDA JÚNIOR

GARANTIA DE ATUALIZAÇÃO MASSIVA DE FIRMWARE DE MEDIDORES INTELIGENTES ATRAVÉS DE UM RECIBO AGREGADOR

Rio de Janeiro
2017

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
INSTITUTO TÉRCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ANTÔNIO LACERDA JÚNIOR

GARANTIA DE ATUALIZAÇÃO MASSIVA DE FIRMWARE DE MEDIDORES INTELIGENTES ATRAVÉS DE UM RECIBO AGREGADOR

Dissertação de Mestrado submetida ao Corpo Docente do Departamento de Ciência da Computação do Instituto de Matemática, e Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção do título de Mestre em Informática.

Orientador: Luiz Fernando Rust da Costa Carmo
Co-orientador: Raphael Carlos Santos Machado

Rio de Janeiro
2017

. Lacerda Júnior, Antônio

Garantia de atualização massiva de firmware de medidores inteligentes através de um recibo agregador / Antônio Lacerda Júnior. 2017.
122 f.: il.

Dissertação (Mestrado em Informática) Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-Graduação em Informática, Rio de Janeiro, 2017.

Orientador: Luiz Fernando Rust da Costa Carmo.
Co-orientador: Raphael Carlos Santos Machado.

Teses. I. Carmo, Luiz Fernando Rust da Costa (Orient.). II. Machado, Raphael Carlos Santos (Co-orient.). III. Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-Graduação em Informática. IV. Título

CDD

ANTÔNIO LACERDA JÚNIOR

Garantia de atualização massiva de FIrmware de medidores inteligentes através de um recibo agregador

Dissertação de Mestrado submetida ao Corpo Docente do Departamento de Ciência da Computação do Instituto de Matemática, e Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção do título de Mestre em Informática.

Aprovado em: Rio de Janeiro, 21 de março de 2017

Prof. Dr. ~~Luiz~~ Fernando Rust da Costa Carmo (Orientador)

Dr. Raphael Carlos Santos Machado (Co-orientador)

Profa. Dra. Silvana Rossetto

Prof. Dr. José Ferreira de Rezende

Prof. Dr. Davidson Rodrigo Boccardo

Dedico este trabalho aos meus três filhos e à minha esposa.

AGRADECIMENTOS

Agradeço:

Aos meus orientadores, Luiz Fernando Rust e Raphael Machado, pelo tempo que eu lhes tomei, pela paciência que eu lhes exercitei e pela confiança que eu recebi.

Aos professores da banca examinadora, Davidson Rodrigo Boccardo, José Ferreira de Rezende e Silvana Rossetto.

Ao professor Jayme Szwarcfiter, homem sábio que eu tive o prazer de conhecer.

A todos os meus colegas da Dimel, em especial ao meu amigo Roberto Lima do Amaral e ao meu chefe, Marcos Trevisan Vasconcellos, pelo profissionalismo, pela confiança, pela amizade.

Ao Carlos Eduardo Galhardo, por sua ajuda na revisão do texto.

A todos os meus referenciais: Pitágoras de Samos, Tales de Mileto, Arquimedes de Siracusa, Platão, Euclides de Alexandria, Imperador Marco Aurélio, Imperador Juliano, Hipácia de Alexandria, Galileu Galilei, Pierre de Fermat, Isaac Newton, Jean-François Champollion, Leonhard Euler, William Blake, Carl Friedrich Gauss, Georg Friedrich Rieman, Hippolyte Rivail, Charles Babbage, Sophie Germain, Charles Sanders Peirce, José de Alencar, Srinivasa Ramanujan, Albert Einstein, Henri Poincaré, William Butler Yeats, Bertrand Russell, Fernando Pessoa, Pietro Ubaldi, George Dumézil, Carl Gustav Jung, Mircea Eliade, Joseph Campbell, Alan Turing, Claude Shannon, Francisco Cândido Xavier.

Aos meus pais, Antônio Lacerda e Maria Socorro.

À minha esposa, Ana Paula, e aos meus filhos, Pedro Arnaldo, Heitor e Clarissa.

A SUPERIORIDADE

"No mundo que nossa humanidade inventou, quanto mais o indivíduo ascende na hierarquia, mais espera que os outros lhe prestem serviço, pois se acomoda no privilégio de ser uma entidade distinta. Essa é uma imagem distorcida de como funciona a hierarquia no Universo; nele, quanto mais uma entidade ascende na escala hierárquica, mais serviço presta aos que existem em dimensões inferiores. O Sol, por exemplo, não está no centro porque pensa apenas em si, é a estrela central porque não guarda nada para si, oferece tudo que produz para que reinos inteiros da natureza possam evoluir e progredir. A superioridade, então, não se mede por quantos possam te servir, mas pela quantidade de seres humanos aos quais tu prestes serviço e pela utilidade prática que tua existência tiver para facilitar a vida dos outros."

Oscar Quiroga, 09/06/2016, site: www.estadao.com.br

Com este trabalho, espero facilitar a vida de outras pessoas.

RESUMO

Lacerda Júnior, Antônio. **Garantia de atualização massiva de firrmware de medidores inteligentes através de um recibo agregador**. 2017. 122 f. Dissertação (Mestrado em Informática) - PPGI, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2017.

Devido a deficiências na legislação penal, que está fora do âmbito da Metrologia Legal, o Inmetro vem, nos últimos anos, aumentando o nível de exigência dos requisitos de segurança de software e hardware a serem atendidos pelos instrumentos de medição, de forma que o Brasil está se destacando no cenário mundial nesse aspecto. Assim, uma nova geração de instrumentos de medição está surgindo, portando diversas funcionalidades criptográficas. Vários problemas ou dificuldades já foram resolvidos. Alguns ainda estão em processo de estudo. Este trabalho trata especificamente de estudar um problema relatado por fabricantes de instrumentos de medição. Supondo-se que foi detectada uma falha grave numa versão de software e vários instrumentos já foram vendidos e estão em utilização em diversos clientes. Não é possível fazer um *recall* de todos os instrumentos para a fábrica e, a partir daí, atualizar os instrumentos. No caso real, o fabricante determina a uma equipe de técnicos para realizar a atualização *in loco*. No final desse processo, a garantia de que todos os instrumentos foram atualizados repousa na alegação dos técnicos. O problema consiste em garantir que uma quantidade qualquer de instrumentos de medição teve seu software atualizado. Várias soluções foram pensadas, mas, para muitas, foram detectadas fragilidades. No final, sobraram três soluções sendo que uma delas se destacou por ser mais robusta, mas também matematicamente mais complexa, a ponto de se extrair duas conjecturas matemáticas interessantes.

Palavras-chave: Metrologia Legal, medidores inteligentes, atualização de firmware, recibos eletrônicos, encriptação homomórfica.

ABSTRACT

Lacerda Júnior, Antônio. **Garantia de atualização massiva de firrmware de medidores inteligentes através de um recibo agregador**. 2017. 122 f. Dissertação (Mestrado em Informática) - PPGI, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2017.

Due to deficiencies in Brazilian criminal law, which is outside the scope of Legal Metrology, Inmetro in recent years has increased the degree of exigency for security requirements in software and hardware to be accomplished by measuring instruments, in such manner that Brazil is standing out on the world stage in this subject. Thus, a new generation of measuring instruments is emerging, carrying many cryptographic functionalities. Several problems or obstacles have already been solved. For some ones, the study is still in progress. This paper deals specifically with the study of a problem reported by manufacturers of measuring instruments. Assuming a serious fault has been detected in a software version and many instruments have already been sold and are in use by several customers. It is not possible to do a recall of all instruments to the factory and then update the instruments. In the real case, the manufacturer delegates a team of technicians to perform the update *in loco*. At the end of this process, the assurance that all instruments have been updated rests on technicians' claim. The problem is to ensure that any number of measuring instruments have had their software updated. Several solutions were proposed, but weaknesses were detected for many of them. In the end three solutions remained and one of them stood out for being more robust, but also mathematically more complex. It was possible to extract two interesting mathematical conjectures from it.

Keywords: Legal Metrology, smart meters, firmware update, electronic receipts, homomorphic encryption.

LISTA DE FIGURAS

LISTA DE TABELAS

ABREVIATURAS E SIGLAS

ANEEL	Agência Nacional de Energia Elétrica
CMD	Comando
Conmetro	Conselho Nacional de Metrologia, Normalização e Qualidade Industrial
DH	Diffie-Hellman
IoT	*Internet of Things* (em português: Internet das Coisas)
Inmetro	Instituto Nacional de Metrologia, Qualidade e Tecnologia
MA	Média Aritmética
MG	Média Geométrica
MIT	*Massassuchets Institute of Technology*
NIST	*National Institute of Standards and Technology*
NSA	*National Security Agency*
PGP	*Pritty Good Privacy*
PLD	Problema do Logaritmo Discreto
RA	Recibo Agregador
RSA	Rivest-Shamir-Adleman
SDN	Software-Defined Networking
SI	Sistema Internacional de Unidades
VIM	Vocabulário Internacional de Metrologia
VIML	Vocabulário Internacional de Metrologia Legal
WELMEC	*Western European Legal Metrology Cooperation*

The word abbreviation sure is long for what it means.

Zach Galifianakis, comediante

LISTA DE SÍMBOLOS

$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$	Conjuntos dos números naturais, inteiros e racionais, respectivamente.
$\mathbb{R}$, $\mathbb{C}$	Conjuntos dos números reais e complexos, respectivamente.
$\mathbb{Z}_{\geq 0}$	Conjunto dos números inteiros não negativos.
$\mathbb{Z}/n\mathbb{Z}$	O conjunto de todas as classes de equivalência módulo n, também pode ser denotado por $\mathbb{Z}/n$.
$\mathcal{G}$	Grupo algébrico.
$\oplus$	Operação Xor ("ou exclusivo").
$x \mod n$	Operação modular ou resto da divisão: $x - \lfloor \frac{x}{n} \rfloor$.
$\binom{n}{k}$	Combinação de n elementos tomados de k em k, ou seja, $\frac{n!}{k!(n-k)!}$.
$\{0,1\}^n$	Uma cadeia binária qualquer de tamanho n.
$Div(n)$	Conjunto dos divisores do número n.
$Enc(m)$	Encriptação da mensagem m.
$Dec(c)$	Decriptação da mensagem cifrada c.
$H(m)$	Função de hash atuando em cima de uma mensagem m.
$\tau(n)$	Determina o número total de divisores positivos do inteiro n.
$\sigma(n)$	Determina a soma de todos os divisores positivos do inteiro n.
$\phi(n)$	Determina a quantidade de números inteiros menores que n que são coprimos de n.
$o(n)$	Notação para definir complexidade de algoritmos; descreve limites superiores assintóticos.
$H(p)$	Função de Entropia Binária

Símbolos são poderosos, pois são sinais visíveis de realidades invisíveis.

Santo Agostinho (354-430)

SUMÁRIO

1 INTRODUÇÃO

Tudo é loucura ou sonho no começo. Nada do que o homem fez no mundo teve início de outra maneira. Mas já tantos sonhos se realizaram, que não temos o direito de duvidar de nenhum.

Monteiro Lobato (1882-1948),
Miscelâneas, Editora Brasiliense,
1956, 7ª Edição, p. 178

O Inmetro, Instituto Nacional de Metrologia, Qualidade e Tecnologia, é o órgão executivo central do Conmetro, Conselho Nacional de Metrologia, Normalização e Qualidade Industrial (Lei 5.966/1973), e tem, como uma das suas atribuições, de elaborar e expedir regulamentos técnicos que disponham sobre o controle metrológico legal, abrangendo instrumentos de medição (inciso II do artigo 3º da Lei 9.933/1999). No inciso III, do mesmo artigo, a lei estabelece também que o Inmetro deve exercer, com exclusividade, o poder de polícia administrativa na área de Metrologia Legal.

De acordo com as resoluções do Conmetro, os regulamentos técnicos deverão dispor sobre características técnicas de insumos, produtos finais e serviços que não constituam objeto da competência de outros órgãos e de outras entidades da Administração Pública Federal, no que se refere a aspectos relacionados com segurança, prevenção de **práticas enganosas de comércio**, proteção da vida e saúde humana, animal e vegetal, e com o meio ambiente (parágrafo 1º do artigo 2º da Lei 9.933/1999).

As práticas enganosas de comércio, comumente denominadas fraudes, constituem um grande desafio para o Inmetro desde a sua fundação. No entanto, a complexidade das fraudes nos instrumentos de medição veio aumentando nas duas últimas décadas, passando a ocorrer principalmente no hardware e no software embarcado (firmware) do instrumento. Desta forma, a fraude tornou-se muito mais difícil de ser detectada por uma equipe de fiscalização. Devido a esse cenário, o Inmetro procurou aprofundar os seus conhecimentos de eletrônica e computação através da contratação de especialistas nessas áreas. Consequentemente, os regulamentos passaram a ser publicados contendo requisitos de segurança tanto para hardware quanto para software dos instrumentos.

Os atuais regulamentos metrológicos publicados pelo Inmetro passaram a exigir uma nova geração de instrumentos de medição. Uma geração de instrumentos imersa em aplicações criptográficas. Não somente problemas antigos passaram a ser resolvidos, como novas funcionalidades de segurança foram propostas e estão em fase de estudo. Este trabalho propõe uma nova funcionalidade, algo que ainda não está sendo exigido nos regulamentos, mas que poderá contribuir com a segurança da informação em sistemas de medição utilizados no cenário atual, no qual os instrumentos estão espalhados por todo o país e sem conectividade com outros dispositivos. A proposta consiste de um recibo agregador que possa garantir que uma quantidade qualquer de dispositivos teve sua versão de firmware atualizada.

1.1 Motivação

Em um mundo no qual dispositivos computacionais estão presentes nos mais diversos ambientes e campos de aplicação, o controle sobre o software embarcado em tais dispositivos torna-se cada vez mais relevante. De fato, a possibilidade de modificar inadvertidamente o software embarcado em um dispositivo pode ter como

consequência uma completa modificação do comportamento do dispositivo em questão, levando a riscos evidentes sobre a aplicação.

No presente trabalho, considerou-se a questão de controle de software sob a óptica de um agente que precisa garantir a atualização de um grande contingente de dispositivos inteligentes instalados remotamente (denomina-se tal cenário por meio da expressão *atualização massiva de software*). Trata-se de um problema relevante quando os dispositivos em questão exercem funções sensíveis, por exemplo, atuando em uma infraestrutura crítica, e uma vulnerabilidade é identificada na versão de software atualmente instalada neles, ensejando uma rápida atualização de todos os instrumentos. O cenário considerado no presente trabalho possui dois aspectos que dificultam o desenvolvimento de protocolos "padrão" baseados em recibos individuais. Por um lado, o processo de atualização dos dispositivos computacionais em questão deve ser realizado por intermédio de uma terceira parte e é esta parte que deve comprovar a atualização. Por outro lado, os dispositivos computacionais podem não estar conectados à Internet, dificultando uma verificação on-line da atualização do software.

O cenário em questão é a realidade observada, por exemplo, nas redes brasileiras de distribuição de energia elétrica. Tomando como exemplo os medidores de energia elétrica, é possível observar que muitos deles já se constituem de *smart meters*, ou seja, dispositivos com software embarcado e interfaces de comunicação locais — mas que, em geral, não possuem funcionalidade de comunicação remota por meio da Internet (de acordo com a resolução da ANEEL, é facultado às distribuidoras prover sistema de comunicação remoto, ver [1] ANEEL, Art. 7º). Ainda assim, caso se identifique uma vulnerabilidade crítica no software de tais medidores — por exemplo, que venha a permitir uma interrupção não-autorizada do fornecimento de energia elétrica — é necessário garantir a atualização de tal software. Em casos extremos, o agente regulador poderá exigir que o fabricante do medidor demonstre

que, de fato, realizou a atualização de todos os medidores instalados em campo.

No presente trabalho, desenvolve-se e analisa-se um protocolo que permite garantir a atualização do software de um conjunto de instrumentos de medição instalados em campo.

1.2 Descrição do Problema

> *Aquele que não consegue descrever um problema, nunca achará uma solução para o mesmo.*
>
> Confucius (551 a 479 a.C.)

No cenário apresentado, as autoridades responsáveis pela regulamentação de medidores inteligentes devem passar a exigir um controle mais rigoroso sobre os processos de atualização de versão de software destes dispositivos. No caso específico estudado (Inmetro), esta exigência se materializa, junto aos fabricantes de instrumentos de medição, com um controle mais estrito sobre as versões de software embarcado (firmware) nos instrumentos de medição vendidos a clientes (designados como "dispositivos em campo", ou seja, fora do ambiente da fábrica e já colocados em uso). A palavra "controle", neste contexto, exige, entre outras coisas, em garantir que, ao sair uma nova versão, todos os dispositivos sejam atualizados. Não será discutido neste documento os motivos que exigiriam essa atualização, pois podem variar desde uma falha grave descoberta no software até uma atualização customizada para um grande cliente, por exemplo, uma concessionária de energia elétrica.

Finalmente, o problema pode ser descrito da seguinte forma: caso um fabricante desenvolva uma nova versão do software do instrumento e esta nova versão

deva ser instalada em N instrumentos em campo, como garantir que todos os N instrumentos tiveram suas versões atualizadas? Essa pergunta, se analisada com mais critério, não parece ser fácil de ser respondida, pois a atualização do software normalmente é efetivada por uma equipe de técnicos em campo e eles podem alegar que realizaram a atualização em todos os N instrumentos. Mas como garantir que a tarefa foi efetivamente realizada? O **responsável pela atualização** (este termo será utilizado ao longo deste documento), que pode ser um chefe da equipe de técnicos, ou mesmo um gerente que coordena várias equipes de técnicos, terá que confiar na alegação dos mesmos ou será que lhe é possível uma evidência mais confiável que toda a tarefa foi completamente realizada? Através de conversas com fabricantes, evidenciou-se que esse problema de fato existe e restavam-lhes apenas a confiança na alegação dos técnicos.

A solução mais simples e imediata que pode ser proposta é a atualização de cada instrumento individualmente e a posterior verificação da versão do novo firmware por uma equipe diferente de técnicos. Além de continuar tendo de confiar em alegações de funcionários, esta abordagem tem custo alto para uma segunda equipe sair em campo a fim de realizar as verificações necessárias. A solução proposta neste documento visa se tornar uma alternativa mais efetiva e barata que essa primeira.

Uma outra pergunta também é de interesse: é possível obter essa garantia através de um único comprovante? Por exemplo, o responsável pela atualização poderia calcular previamente um número e estabelecer à(s) sua(s) equipe(s) de técnicos que realize(m) procedimentos nos medidores atualizados de forma que, ao final do processo de atualização dos N instrumentos, um número possa ser obtido e, ao ser comparado ao número previamente calculado, caso sejam iguais, ter-se-ia uma garantia inequívoca que a tarefa foi plenamente realizada. O cálculo prévio efetuado pelo responsável deve ter a característica de que aos técnicos não seja possível

descobrir como o comprovante foi calculado. Não permitindo assim uma burla à tarefa estabelecida. Na prática, há uma grande vantagem em conferir apenas um número em vez de N números. Esse número funcionaria como um **recibo agregador** (RA) para garantir que todos os instrumentos tiveram suas versões atualizadas. Será mostrado que duas das três soluções propostas permitem essa abordagem.

2 PESQUISAS RELACIONADAS

É importante esclarecer que o problema tratado nesta dissertação é novo. A fim de que se entenda os motivos desse ineditismo, é necessário uma explicação de como está a Metrologia Legal no país atualmente. Hoje o Brasil é pioneiro no alto nível de exigência dos requisitos de segurança de software e de hardware a serem atendidos pelos instrumentos de medição. Esse pioneirismo ocorreu devido à falta de uma legislação penal mais rigorosa na hora de punir quem frauda um instrumento de medição (legislação, portanto, fora do âmbito da Metrologia Legal). Em países mais desenvolvidos, o rigor está na fiscalização e na punição em quem frauda. No Brasil, a legislação penal é branda na hora de punir os fraudadores, ou, como é comumente escutado nas ruas, "a fraude compensa". Esse fato impulsionou o Inmetro a elevar os requisitos de seus regulamentos, com o objetivo de impedir ou dificultar ao máximo as tentativas de fraudes. Desta forma, no momento, o Brasil está sendo pioneiro em exigir requisitos em níveis mais altos do que no resto do mundo. Um bom exemplo disso é a exigência de apresentar código-fonte do software do instrumento. Tal requisito é exigido pela WELMEC (ver ref. [21] WELMEC) somente para classe de risco E (nível de risco mais alto; nessa classe entra, por exemplo, as máquinas caça-níqueis). O Inmetro, ao contrário, decidiu tornar a exigência de código-fonte como um requisito padrão para todos os setores. Devido a esse pioneirismo nos requisitos de segurança para software e hardware, não foram encontrados artigos que tratassem exatamente do mesmo problema analisado nesta dissertação.

Em [4] CHOI, o problema de atualização de versões de firmware é tratada em cenário de IoT em redes domésticas. Além disso, o método também serve para verificar a integridade do firmware através de sua validação. Em [9] KATZIR, os autores propõem uma solução para tornar mais segura a atualização de software

num cenário específico de dispositivos em redes de Corrente Alternada. Em [11] LIU, os autores tratam da atualização de software num cenário específico de IoT com redes de sensores sem fio. Em [12] MATTOS, os autores propõem um esquema de atualização reversa para SDN (Software-Defined Networking), no qual a garantia se dá por pacotes. Em [6] HONG, os autores propõem um framework de software e uma arquitetura de hardware no caso específico de IoT em Smartphones. Em todos esses artigos, o procedimento proposto visa garantir segurança no momento da atualização. Neste trabalho, o cenário ocorre à posteriori.

Mais detalhes sobre como o Inmetro lida com a segurança da informação em instrumentos de medição podem ser encontrados em [3] BOCCARDO e [15] PRADO.

3 CONCEITOS BÁSICOS

3.1 Metrologia Legal

If you can't measure something, you can't understand it. If you can't understand it, you can't control it. If you can't control it, you can't improve it.

H. James Harrington (1929), CIO Enterprise, Section 2, sep/1999

Segundo o VIM[1](Vocabulário Internacional de Metrologia), ver ref. [7] INMETRO, a metrologia é definida como "ciência da medição e suas aplicações", e o mesmo documento complementa a definição afirmando que "a metrologia engloba todos os aspectos teóricos e práticos da medição, qualquer que seja a incerteza de medição e o campo de aplicação". É uma ciência rica de assuntos, com um campo de aplicação muito vasto. Costuma-se dividi-la em três partes:

1) **Metrologia Fundamental (ou Científica)**: cujo interesse está nas grandezas fundamentais e suas respectivas unidades fundamentais de medidas, nos sistemas de unidades, na rastreabilidade dos padrões e no desenvolvimento de novas formas de medição, entre outros tópicos.

[1] O VIM é mantido pelo JCGM (*Joint Committee for Guides in Metrology*), um comitê formado por oito instituições BIPM (Birô Internacional de Pesos e Medidas), IEC (*International Electrotechnical Commission*), IFCC (*International Federation of Clinical Chemistry and Laboratory Medicine*), ISO (*International Organization for Standardization*), IUPAC (*International Union of Pure and Applied Chemistry*), IUPAP (*International Union of Pure and Applied Physics*), OIML (*International Organization of Legal Metrology*) e o ILAC (*International Laboratory Accreditation Cooperation*).

2) **Metrologia Aplicada (ou Técnica ou Industrial)**: cujo interesse está na aplicação da ciência de medição às indústrias em geral (fabricação de coisas e outros processos nos quais envolvam medições de grandezas físicas) e a sua utilização na sociedade, assegurando a adequação dos instrumentos de medição, a sua correta calibração e o controle de qualidade das medições.

3) **Metrologia Legal**: o VIML (Vocabulário Internacional de Metrologia Legal), ver ref. [8] INMETRO, define a Metrologia Legal como "a prática e o processo de aplicar à metrologia uma estrutura legal e regulamentadora e implementar sua execução". Por isso, o VIML também afirma que "o escopo da Metrologia Legal pode diferir de um país para outro". Dentre as várias atividades da Metrologia Legal estão:

- Estabelecer requisitos legais;
- Avaliação de modelos (mais detalhes no item 3.1.2);
- Supervisionar os produtos e as atividades regulamentados;
- Prover infraestrutura necessária para a rastreabilidade das medições e dos instrumentos de medição regulamentados ao SI (Sistema Internacional de Unidades) ou aos padrões nacionais.

O contexto deste trabalho se insere no âmbito da Metrologia Legal.

3.1.1 Atuais Requisitos Metrológicos para o Software

Até a publicação desta dissertação, o Inmetro já publicou os seguintes regulamentos contendo requisitos de segurança de software e hardware:

Tabela 3.1: Regulamentos em vigor contendo requisitos de segurança para software e hardware

Tipo de Instrumento	**Portaria**	**Observação**
Instrumentos de Pesagem Automáticos	Portaria nº 375, de 24 de julho de 2013 e alterada pela Portaria n.º 47, de 22 de janeiro de 2016	Basicamente constituídos pelas balanças rodoviárias.
Medidor de Velocidade	Portaria nº 544, de 12 de dezembro de 2014.	Trata dos radares eletrônicos em ruas, avenidas e rodovias.
Medidores de Energia Elétrica	Portaria nº 587, de 05 de novembro de 2012	A primeira família de medidores a ter regulamentos de segurança.
Medidor de Umidade de Grãos	Portaria nº 402, de 15 de agosto de 2013 e alterada pela Portaria n.º 617, de 20 de dezembro de 2013.	A pedido do Ministério da Agricultura para regulamentar as transações comerciais de grãos.
Mototaxímetro	Portaria n.º 393, de 26 de Julho de 2012.	Somente para os mototaxímetros. Os taxímetros são regulamentados por outra portaria que ainda não possui requisitos de software e hardware.
Computador de Vazão	Portaria n.º 499, de 02 de outubro de 2015.	Instrumentos para medir volumes de óleo em tubulações de refinarias.
BMC	Portaria n.º 559, de 15 de dezembro de 2016.	Bombas Medidoras de Combustíveis líquidos.

Outros regulamentos já estão em elaboração. Aquele que está mais próximo de ser publicado é o de instrumentos de pesagem não automática (balanças tradicionais). A seguir vem os regulamentos para medidores de volume de gás combustível, hidrômetros para água fria (de vazão nominal até quinze metros cúbicos por hora) e os etilômetros portáteis e não portáteis (utilizados pela fiscalização de trânsito na determinação da concentração de etanol no ar expirado).

3.1.2 Avaliação de Modelo

O VIML, [8] INMETRO, define avaliação de modelo como um procedimento de avaliação da conformidade realizado em um ou vários exemplares de um modelo identificado de um instrumento de medição, e cujo resultado está contido num relatório e/ou certificado de avaliação de modelo. Para tornar mais claro o conceito, o mesmo documento complementa dizendo que "Modelo" é utilizado em Metrologia Legal com o mesmo significado de "tipo".

A avaliação de modelos é uma das principais atividades da Metrologia Legal. Quando uma classe de instrumentos de medição está sob a regulamentação do Inmetro, qualquer fabricante que deseja vender instrumentos dessa classe no Brasil, deverá submetê-los a uma bateria de testes realizada por funcionários do Inmetro devidamente capacitados para a tarefa. O fabricante deverá ceder um ou mais exemplares para os testes. Caso esses exemplares sejam aprovados em todos os testes, eles serão tomados como modelos. A partir daí, o fabricante deverá construir cópias desses exemplares, gerando portanto uma família de instrumentos exatamente iguais. O instrumento que não seja cópia de um modelo aprovado, ou seja, não pertença a alguma família de instrumentos aprovada, não pode ser comercializado.

3.2 Conceitos Matemáticos Necessários

Em Matemática, somos servos mais que senhores.

Charles Hermite (1822-1901),
"Jacques Hadamard: A Universal
Mathematician". American
Mathematical Soc. pp. 33 34.

A matemática necessária para o completo entendimento desta dissertação consiste de alguns conceitos de teoria dos números (aritmética modular, funções aritméticas, Teorema de Euler, algoritmo de Euclides estendido) e álgebra abstrata (grupos, grupos cíclicos, geradores). Nesta dissertação, utilizam-se dois algoritmos criptográficos: o RSA e o Elgamal. Eles serão explicados em detalhes na próxima seção. Nesta seção, serão explicados os conhecimentos matemáticos mínimos necessários para cada um desses algoritmos.

Para começar, o conhecimento mais básico para se entender o conteúdo desta dissertação é o seguinte teorema:

Teorema 3.2.1. ***Teorema Fundamental da Aritmética:*** *Cada inteiro $n \geqslant 2$ tem uma fatorização como um produto de potências de primos: $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ onde os p_i são primos distintos, e os e_i são inteiros positivos, além disso, a fatorização é única.*

Nota: Todos os teoremas contidos nesta dissertação serão provados, menos este, pois a prova é longa. Caso o leitor queira vê-la, ela pode ser lida em [13] POLCINO, página 77.

Agora será apresentada a matemática necessária para o entendimento do Elgamal.

3.2.1 Grupos

Definição 3.2.2. ***Operação binária****: Uma operação binária sobre um conjunto $\mathcal{S}$ é uma relação que associa a cada par ordenado (a, b) de elementos de $\mathcal{S}$, um único elemento de $\mathcal{S}$.*

Alguns exemplos, a operação adição $+$ e a operação multiplicação $\times$ são exemplos clássicos de operações binárias. Mas é possível definir outras operações, por exemplo, a média aritmética MA entre dois números, dada por $MA(a, b) = \frac{a+b}{2}$, ou a média geométrica MG entre dois números, $MG(a, b) = \sqrt{ab}$, ambas definidas sobre o conjunto dos números complexos $\mathbb{C}$.

Definição 3.2.3. ***Grupo****: Um grupo é um conjunto não vazio $\mathcal{G}$ que possui uma operação binária $\star$, tal que os seguintes axiomas são atendidos.*

(1) Fechamento: $a \star b \in \mathcal{G}, \forall a, b \in \mathcal{G}$.

(2) Associatividade: $(a \star b) \star c = a \star (b \star c), \forall a, b, c \in \mathcal{G}$.

(3) Existência de identidade: Há um único elemento $e \in \mathcal{G}$, chamado de identidade, tal que $e \star a = a \star e = a, \forall a \in \mathcal{G}$.

(4) Existência de inverso: Para cada elemento $a \in \mathcal{G}$, existe um único elemento b tal que $a \star b = b \star a = e$. Este elemento b é denotado por a^{-1} e chamado de inverso de a.

Observação: Existe uma polêmica nessa definição de grupo. Alguns matemáticos consideram que o axioma fechamento deve ser atendido pela operação binária e não pelo grupo, ou seja, ele deveria ser um axioma implícito e não explícito ao grupo. O livro estudado pelo autor desta dissertação considera o axioma como explícito [22] YAN.

Um outro item polêmico na própria matemática é se o elemento 0 faz parte ou não dos números naturais $\mathbb{N}$. Tanto em [22] YAN quanto em [10] LIMA (este último é considerado como um livro de referência aqui no Brasil), os autores consideram o 0 como fora dos naturais. O autor desta dissertação também concorda pois a humanidade começou a contar **naturalmente** com 1, 2, 3 etc.

Definição 3.2.4. ***Grupo Comutativo:*** *Um grupo é comutativo se ele atender a mais este axioma.*

(5) Comutatividade: $a \star b = b \star a, \forall a, b \in \mathcal{G}$.

Os grupos comutativos são também chamados de grupos abelianos, em homenagem ao matemático norueguês Niels Henrik Abel (1802-1829).

Um grupo pode ser representado por $\langle \mathcal{G}, \star \rangle$, por $(\mathcal{G}, \star)$ ou, simplesmente por $\mathcal{G}$.

Quando a operação é a adição, costuma-se designar o grupo como "grupo aditivo"; quando a operação é a multiplicação, o grupo é chamado de "grupo multiplicativo".

Definição 3.2.5. ***Subgrupo:*** *Um subgrupo $\mathcal{H}$ é um subconjunto não vazio de um grupo $\mathcal{G}$ que também forma um grupo com a mesma operação. Costuma-se denotar*

esse fato assim $\mathcal{H} < \mathcal{G}$.

Exemplos de grupos:

1) Os conjuntos $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ e $\mathbb{C}$ formam grupos (abelianos e aditivos) com a operação adição. O conjunto $\mathbb{Z}$, com a mesma operação adição, é um subgrupo de $\mathbb{Q}$, de $\mathbb{R}$ e de $\mathbb{C}$. E $\mathbb{Q}$ um subgrupo de $\mathbb{R}$ e de $\mathbb{C}$. E $\mathbb{R}$ um subgrupo de $\mathbb{C}$.

2) O conjunto $\mathbb{N}$ não forma um grupo com a operação adição, pois não atende ao axioma (4), existência do inverso.

3) Os conjuntos $\mathbb{Q} \setminus \{0\}$, $\mathbb{R} \setminus \{0\}$ e $\mathbb{C} \setminus \{0\}$ formam grupos (abelianos e multiplicativos) com a operação multiplicação. O símbolo $\setminus\{0\}$ indica que o conjunto exclui o elemento 0 do conjunto original.

4) O conjunto $\mathbb{Z} \setminus \{0\}$ não forma um grupo com a operação multiplicação, pois não atende ao axioma (4), existência do inverso.

5) Os conjuntos $\mathbb{R}$ e $\mathbb{C}$ não formam um grupo com as operações MA e MG, pois não atendem ao axioma (2), associatividade.

Existem muitos resultados interessantes que podem ser extraídos do conceito de subgrupos. A primeira atitude a tomar quando se tem em mãos um subconjunto não vazio de um grupo é verificar se ele é um subgrupo ou não, para isso, existe um "atalho". De acordo com a definição de grupo, existem quatro axiomas que devem ser atendidos. No entanto, o segundo (da associatividade) e o terceiro (existência do elemento neutro) são sempre atendidos por qualquer subconjunto contendo ao menos um elemento neutro. Explica-se. Seja $\mathcal{G}$ um grupo qualquer e $\mathcal{H}$ um subconjunto de

$\mathcal{G}$. Do axioma da associatividade, o conjunto $\mathcal{H}$ deve satisfazer a seguinte condição: $h_1 \star (h_2 \star h_3) = (h_1 \star h_2) \star h_3, \forall h_1, h_2, h_3 \in \mathcal{H}$, mas como $h_1, h_2, h_3 \in \mathcal{G}$, então isso é sempre verdade. Quanto ao axioma da existência do elemento neutro, supondo-se que o elemento neutro de $\mathcal{H}$, e_H, seja diferente do elemento neutro de $\mathcal{G}$, e, então $h \star e_H = h$, multiplicando ambos os lados por h^{-1}, obtem-se $e_H = e$.

Então, na prática, para concluir que um subconjunto $\mathcal{H}$ de um grupo $\mathcal{G}$, contendo o elemento neutro e, forma um subgrupo, basta provar os axiomas do fechamento ($h_1 \star h_2 \in \mathcal{H}, \forall h_1, h_2 \in \mathcal{H}$) e da existência do inverso ($\exists h^{-1} \in \mathcal{H}, \forall h \in \mathcal{H}$).

O primeiro resultado interessante a cerca dos subgrupos é o seguinte: dado um elemento a qualquer de $\mathcal{G}$, defina-se o seguinte conjunto $\langle a \rangle = \{a^t | t \in \mathbb{Z}\}$. Onde a^t representa a aplicação da operação $\star$ repetidamente sobre a. Se $t < 0$, então $a^t = (a^{-1})^{|t|}$. Ora, pelo axioma do fechamento, está claro que $\langle a \rangle$ é um subconjunto de $\mathcal{G}$. Daí é possível extrair o seguinte resultado:

Teorema 3.2.6. *O conjunto $\langle a \rangle$ é um subgrupo de $\mathcal{G}$, ou seja, $\langle a \rangle < \mathcal{G}$.*

Prova: Utilizando o "atalho", deve-se provar primeiro que, dado $x, y \in \langle a \rangle$, tem-se $x \star y \in \langle a \rangle$. Como $x \in \langle a \rangle$, então $x = a^r$. Como $y \in \langle a \rangle$, então $y = a^s$. Assim, $x \star y = a^r \star a^s = a^{r+s} \in \langle a \rangle$. Deve-se provar agora que $\forall x \in \langle a \rangle$, tem-se $x^{-1} \in \langle a \rangle$. Mas é claro que $x^{-1} = (a^r)^{-1} = a^{-r} \in \langle a \rangle$.

c.q.d.

Definição 3.2.7. ***Subgrupo gerado por** a: O subconjunto $\langle a \rangle$ será denominado de subgrupo gerado por a.*

Definição 3.2.8. ***Ordem de um grupo**: A ordem de um grupo $\mathcal{G}$ é a quantidade*

de elementos no conjunto, ou seja, é a cardinalidade de $\mathcal{G}$, representada por $|\mathcal{G}|$ ou por $\#(\mathcal{G})$.

Definição 3.2.9. ***Grupo Finito**: Um grupo $\mathcal{G}$ é finito quando a quantidade de elementos do conjunto é finita.*

Definição 3.2.10. ***Ordem de um elemento**: A ordem de um elemento $a \in \mathcal{G}$ é a quantidade de elementos no subconjunto gerado por a.*

Teorema 3.2.11. *Seja a um elemento de um grupo qualquer $\mathcal{G}$ e $\langle a \rangle$ o subgrupo gerado por a. Então as seguintes condições são equivalentes: (i) A ordem de $|\langle a \rangle|$ é finita; (ii) $\exists t \geqslant 1$ tal que $a^t = e$.*

Prova: Primeiro, será provado $(i) \rightarrow (ii)$. Pela definição $\langle a \rangle = \{a^m | m \in \mathbb{Z}\}$. Por hipótese, o subgrupo $\langle a \rangle$ é finito. Então, existem $p, q \in \mathbb{Z}, p \neq q$ tais que $a^p = a^q$. Sem perda de generalidade, é possível supor que $p > q$. Então $a^p = a^q \leftrightarrow a^{p-q} = e$, então, tomando $t = p - q$, obtem-se $a^t = e$ com $t > 0$. Para provar a volta, ou seja, $(ii) \rightarrow (i)$, considera-se o inteiro $r = min\{t \leqslant 1; a^t = e\}$, se for provado que $r = n$, onde n é a ordem do elemento a então a prova termina. Ora, para isto, basta provar que:

1) Os elementos $e, a, a^2, \ldots, a^{r-1}$ são todos distintos.

Isso é verdade, pois, supondo que $a^p = a^q$ com $0 \leqslant p, q \leqslant r - 1, p \neq q$, e supondo $p > q$, tem-se $a^{p-q} = e$, com $0 < (p-q) < r$ e isso contradiz a minimalidade de r.

2) O conjunto $\langle a \rangle = \{e, a, a^2, \ldots, a^{r-1}\}$.

Isso também é verdade, pois $\forall m \in \mathbb{Z}, a^m = a^l$, para algum l tal que $0 \leqslant l < r$. Pelo algoritmo de Euclides, existem $q, l \in \mathbb{Z}$ tais que $m = qr + l$, com $0 \leqslant l < r$, e

portanto, $a^m = a^{qr+l} = (a^r)^q \times a^l = e^q \times a^l = a^l$.

Agora será definido um conceito que é o fulcro do algoritmo Elgamal: os geradores.

Definição 3.2.12. ***Geradores e grupo cíclico****: Se a ordem do elemento a for finita e coincidir com a ordem do grupo $\mathcal{G}$, então o elemento será chamado de* gerador do grupo *e será representado pela letra g e o grupo $\mathcal{G}$ será chamado de* grupo cíclico.

Até agora, todos os exemplos de grupos foram de ordem infinita. Será apresentado agora um exemplo de grupo com ordem finita. Antes disso é preciso apresentar a seguinte definição:

Definição 3.2.13. ***Classes de equivalência****: Na aritmética modular, fixando um valor inteiro n não nulo, então todos os inteiros ficam divididos em n subconjuntos denominados* classes de equivalência módulo n.

Exemplos:

1) Para $n = 2$, existem duas classes de equivalência, ou seja, dois subconjuntos dos inteiros. O primeiro subconjunto, representado por $\bar{0}$, é formado por todos os números a, tais que $(a \equiv 0) \mod 2$, assim, $\bar{0} = \{\cdots -4, -2, 0, 2, 4, \dots\}$, comumente $\bar{0}$ é chamado de conjunto dos pares. O segundo subconjunto, denominado $\bar{1}$, é formado por todos os números b, tais que $(a \equiv 1) \mod 2$, assim, $\bar{1} = \{\cdots -3, -1, 1, 3, 5, \dots\}$, comumente $\bar{1}$ é chamado de conjunto dos ímpares.

2) Para $n = 3$, existem três classes de equivalência:
$\bar{0} = \{\cdots -9, -6, -3, 0, 3, 6, 9\}$, formada pelos números a, tais que $(a \equiv 0) \mod 3$.

$\bar{1} = \{\cdots -10, -7, -4, 1, 4, 7, 10\}$, formada pelos números b, tais que $(b \equiv 1) \mod 3$.
$\bar{2} = \{\cdots -11, -8, -5, 2, 5, 8, 11\}$, formada pelos números c, tais que $(c \equiv 2) \mod 3$.

Definição 3.2.14. ***Conjunto de todas as classes de equivalência:*** *É o conjunto formado por todos as classes de equivalência módulo n, ou seja,* $\{\bar{0}, \bar{1}, \bar{2}, \ldots \overline{n-1}\}$.

O conjunto de todas as classes de equivalência módulo n é denotado por $\mathbb{Z}/n\mathbb{Z}$ ou $\mathbb{Z}/n$.

Observação: Atualmente, a notação alternativa $\mathbb{Z}_n$ não é recomendada por causa da possível confusão com o conjunto dos inteiros p-ádicos. No entanto, ainda é comum vê-la nos livros e artigos de criptologia. Neste trabalho, essa notação não será utilizada.

Cada classe de equivalência pode ser representada por qualquer um de seus elementos, entretanto é muito mais comum escolher o menor inteiro não negativo que pertence à classe. Ao se fazer isso, o conjunto de todas as classes de equivalência fica representada como $\{0, 1, 2, \ldots n-1\}$, que é o conjunto de todos os possíveis restos (ou resíduos) módulo n.

As operações adição e multiplicação nesses conjuntos de resíduos serão indicadas por $\oplus$ e $\otimes$ respectivamente. O conjunto de resíduos módulo n, $\mathbb{Z}/n\mathbb{Z}$, com a operação adição mostrada acima, forma um grupo abeliano finito. Enquanto o conjunto de resíduos módulo n sem o elemento zero, ou seja, $(\mathbb{Z}/n\mathbb{Z}) \setminus \{0\}$, forma um grupo abeliano finito com a operação multiplicação.

Para exemplificar tudo o que foi discutido até agora, as tabelas 3.2, 3.3 e 3.4 mostram como são essas operações para $n = 11$, quais são os subgrupos gerados por elementos, quais são os geradores do grupo e a ordem de cada elemento.

Tabela 3.2: Operação adição modular $\oplus$ para resíduos módulo 11

$\oplus$	0	1	2	3	4	5	6	7	8	9	10
0	0	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10	0
2	2	3	4	5	6	7	8	9	10	0	1
3	3	4	5	6	7	8	9	10	0	1	2
4	4	5	6	7	8	9	10	0	1	2	3
5	5	6	7	8	9	10	0	1	2	3	4
6	6	7	8	9	10	0	1	2	3	4	5
7	7	8	9	10	0	1	2	3	4	5	6
8	8	9	10	0	1	2	3	4	5	6	7
9	9	10	0	1	2	3	4	5	6	7	8
10	10	0	1	2	3	4	5	6	7	8	9

Para entender a tabela 3.2, toma-se como exemplo os números 8 e 7, efetuando a adição modular, $(8 \oplus 7) \mod 11 = 15 \mod 11 = 4$. Então, o elemento na linha 8 e coluna 7 é igual a 4. A tabela é simétrica, pois a operação adição modular é comutativa.

Tabela 3.3: Operação multiplicação modular $\otimes$ para resíduos módulo 11

$\otimes$	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10
2	0	2	4	6	8	10	1	3	5	7	9
3	0	3	6	9	1	4	7	10	2	5	8
4	0	4	8	1	5	9	2	6	10	3	7
5	0	5	10	4	9	3	8	2	7	1	6
6	0	6	1	7	2	8	3	9	4	10	5
7	0	7	3	10	6	2	9	5	1	8	4
8	0	8	5	2	10	7	4	1	9	6	3
9	0	9	7	5	3	1	10	8	6	4	2
10	0	10	9	8	7	6	5	4	3	2	1

Para entender a tabela 3.3, toma-se como exemplo novamente os números 8

e 7, efetuando a multiplicação modular, $(8 \otimes 7) \mod 11 = 56 \mod 11 = 1$. Então, o elemento na linha 8 e coluna 7 é igual a 1. A tabela é simétrica, pois a operação multiplicação modular é comutativa.

Tabela 3.4: Exponenciação modular para resíduos módulo 11

a^b	1	2	3	4	5	6	7	8	9	10
1	(1)	1	1	1	1	1	1	1	1	1
2	2	4	8	5	10	9	7	3	6	(1)
3	3	9	5	4	(1)	3	9	5	4	1
4	4	5	9	3	(1)	4	5	9	3	1
5	5	3	4	9	(1)	5	3	4	9	1
6	6	3	7	9	10	5	8	4	2	(1)
7	7	5	2	3	10	4	6	9	8	(1)
8	8	9	6	4	10	3	2	5	7	(1)
9	9	4	3	5	(1)	9	4	3	5	1
10	10	(1)	10	1	10	1	10	1	10	1

A tabela 3.4 necessita de uma explicação. A tabela representa as exponenciações modulares $a^b \mod 11$. A base a está representada pelos números na primeira coluna, enquanto os expoentes estão na primeira linha. Assim, considerando $a = 7$ e $b = 5$, tem-se $7^5 \mod 11 = 16807 \mod 11 = 10$, justamente o valor encontrado na linha 7 e na coluna 5. Os números 1 entre parênteses representam o último elemento distinto encontrado, a partir daí, os números se repetem numa mesma linha (consequência do teorema 3.2.11). Os números distintos numa mesma linha formam um subgrupo. Para exemplificar, as linhas das bases 3, 4, 5 e 9 produzem produzem o mesmo subconjunto: $\{1, 3, 4, 5, 9\}$. Para visualizar porque esse subconjunto forma um subgrupo, ver a tabela 3.5.

Tabela 3.5: Subgrupo com a multiplicação módulo 11

$\otimes$	1	3	4	5	9
1	1	3	4	5	9
3	3	9	1	4	5
4	4	1	5	9	3
5	5	4	9	3	1
9	9	5	3	1	4

Da tabela 3.4, forma-se a tabela 3.6, onde estão todos geradores e seus respectivos subgrupos e a ordem do subgrupo. Em especial, é possível ver os geradores $\{2, 6, 7, 8\}$, os quais formam o grupo inteiro.

Tabela 3.6: Subgrupos e geradores para resíduos módulo 11

Subgrupo	Geradores	Ordem
$\{1\}$	1	1
$\{1, 10\}$	10	2
$\{1, 3, 4, 5, 9\}$	$3, 4, 5, 9$	5
$(\mathbb{Z}/11\mathbb{Z}) \setminus \{0\}$	$2, 6, 7, 8$	10

Acredita-se que, com esses conceitos iniciais sobre grupos e geradores, é possível entender completamente o algoritmo Elgamal. Agora, será apresentada a matemática necessária para o entendimento do RSA.

3.2.2 Funções Aritméticas

The elementary theory of numbers should be one of the very best subjects for early mathematical instruction. It demands very little previous knowledge, its subject matter is tangible and familiar; the processes of reasoning which it employs are simple, general and few; and it is unique among the mathematical sciences in its appeal to natural human curiosity

G. H. Hardy (1877-1947) in "An Introduction to the Theory of Numbers"

Definição 3.2.15. ***Função aritmética****: Uma função f é chamada de aritmética se ela atribui a cada inteiro n não negativo, um único valor real ou complexo $f(n)$.*

Pela definição, as funções aritméticas são quaisquer funções do tipo $f : \mathbb{Z} \to \mathbb{R}$ (ou $\mathbb{C}$), mas, na prática, elas são utilizadas para apresentar propriedades dos números inteiros. Por exemplo, a função $f(n) = n^5 + \sqrt[5]{n+5} + 5$, pela definição é uma função aritmética, mas, na prática, ela não descreve uma propriedade dos números inteiros, portanto, não tem muita utilidade para a Teoria dos Números (a não ser que alguém descubra alguma propriedade interessante dos inteiros dada por essa fórmula).

Para uma abordagem mais didática, serão apresentadas as duas funções aritméticas mais conhecidas, a $\tau(n)$ e a $\sigma(n)$. Elas serão explicadas, apesar de não serem utilizadas nesta dissertação, mas a compreensão delas auxiliará no estudo da *função totiente*, a função aritmética que realmente será utilizada nesta dissertação.

Definição 3.2.16. ***Função*** $\tau(n)$*: A função $\tau(n)$ representa o número total de divisores positivos do inteiro n. Definida pela equação abaixo:*

$$\tau(n) = \sum_{d|n} 1 \tag{3.1}$$

A função τ recebeu esse símbolo da língua alemã: *Teiler* significa divisor.

Exemplo: O inteiro 12 possui 6 divisores, ou seja, $Div(12) = \{1, 2, 3, 4, 6, 12\}$, portanto, $\tau(12) = 6$.

Definição 3.2.17. ***Função*** $\sigma(n)$*: A função* $\sigma(n)$ *representa a soma de todos os divisores positivos do inteiro* n*. Definida pela equação abaixo:*

$$\sigma(n) = \sum_{d|n} d \tag{3.2}$$

Exemplo: A soma dos divisores de 12 é $1+2+3+4+6+12 = 28$, portanto, $\sigma(12) = 28$.

Definição 3.2.18. ***Função aritmética multiplicativa****: Uma função aritmética* f *é multiplicativa se, para dois inteiros positivos quaisquer* n *e* m*, com* $mdc(n, m) = 1$*,* $f(nm) = f(n) \times f(m)$*.*

Teorema 3.2.19. *As funções* $\tau(n)$ *e* $\sigma(n)$ *são ambas multiplicativas.*

Prova: Primeiro, é necessário tomar dois inteiros positivos n_1 e n_2 coprimos entre si, ou seja, $mdc(n_1, n_2) = 1$. Suponha-se que n_1 possui r divisores dados por $Div(n_1) = \{d_{1,1}, d_{1,2}, \ldots d_{1,r}\}$ e que n_2 possui s divisores dados por $Div(n_2) = \{d_{2,1}, d_{2,2}, \ldots d_{2,s}\}$. Assim, $\tau(n_1) = r$ e $\tau(n_2) = s$. Para se ter um divisor de $n_1 n_2$, basta formar um divisor da forma $d_1 d_2$, onde $d_1 \in Div(n_1)$ e $d_2 \in Div(n_2)$, então pelo Princípio Multiplicativo da Análise Combinatória, a quantidade de divisores

que se pode formar para $n_1 n_2$ é rs, portanto, $\tau(n_1 n_2) = \tau(n_1)\tau(n_2)$. No caso da função $\sigma(n_1 n_2)$, a soma será:

$$\begin{aligned} \sigma(n_1 n_2) = \\ d_{1,1}d_{2,1} + d_{1,1}d_{2,2} + \cdots + d_{1,1}d_{2,s} + \\ d_{1,2}d_{2,1} + d_{1,2}d_{2,2} + \cdots + d_{1,2}d_{2,s} + \\ d_{1,r}d_{2,1} + d_{1,r}d_{2,2} + \cdots + d_{1,r}d_{2,s} \end{aligned} \tag{3.3}$$

Colocando em evidência os fatores em comum:

$$\sigma(n_1 n_2) = (d_{1,1} + d_{1,2} + \cdots + d_{1,r})(d_{2,1} + d_{2,2} + \cdots + d_{2,s}) \tag{3.4}$$

Mas isso é justamente:

$$\sigma(n_1 n_2) = \sigma(n_1)\sigma(n_2) \tag{3.5}$$

c.q.d.

Exemplo, para $n = 8$ e $m = 15$, tem-se $mdc(8, 15) = 1$, $\tau(8) = 4$, $\sigma(8) = 15$, $\tau(15) = 4$, $\sigma(15) = 24$. Daí é possível aplicar o teorema 3.2.19.

$$\tau(120) = \tau(8 \times 15) = \tau(8) \times \tau(15) = 4 \times 4 = 16 \tag{3.6}$$

$$\sigma(120) = \sigma(8 \times 15) = \sigma(8) \times \sigma(15) = 15 \times 24 = 360 \tag{3.7}$$

De fato, os divisores de 120 são:

$$Div(120) = \{1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60, 120\}$$

Ou seja, 16 divisores, cuja soma é 360.

Definição 3.2.20. ***Função aritmética completamente multiplicativa****: Uma função aritmética f é completamente multiplicativa se, para dois inteiros positivos quaisquer n e m, $f(nm) = f(n) \times f(m)$.*

As funções aritméticas completamente multiplicativas diferem das multiplicativas pelo simples fato de que as primeiras não exigem a condição $mdc(n, m) = 1$.

Teorema 3.2.21. *As funções $\tau(n)$ e $\sigma(n)$ não são completamente multiplicativas.*

Prova: Para provar que não são, basta mostrar um contraexemplo para cada uma delas. Sejam $n = 12$ e $m = 15$, notar que o $mdc(12, 15) = 3$. Tem-se $\tau(12) = 6$ e $\tau(15) = 4$, e $\tau(12 \times 15) = \tau(180) = 18$, o que não corresponde ao produto $\tau(12) \times \tau(15) = 6 \times 4 = 24$. Da mesma forma para a função σ. Tem-se $\sigma(12) = 28$ e $\sigma(15) = 24$, e $\sigma(12 \times 15) = \sigma(180) = 546$, o que não corresponde ao produto $\sigma(12) \times \sigma(15) = 28 \times 24 = 672$.

c.q.d.

Agora será definida a função aritmética que será utilizada nesta dissertação: a função totiente[2], a qual é representada pela letra grega ϕ.

Definição 3.2.22. ***Função totiente:*** *A função totiente $\phi(n)$ é a função aritmética que mostra a quantidade de números inteiros menores que n que são coprimos de n, ou seja, relativamente primos a n.*

Para exemplificar, toma-se o inteiro 12, então $\phi(12) = 4$, pois os inteiros positivos que são coprimos com 12 e abaixo dele são $\{1, 5, 7, 11\}$.

Interessante ressaltar que o conjunto dos números coprimos a um dado inteiro n é tão utilizado pelos matemáticos que ganhou uma representação própria: $(\mathbb{Z}/n\mathbb{Z})^*$. Daí que a função $\phi(n)$ também pode ser definida como $\phi(n) = |(\mathbb{Z}/n\mathbb{Z})^*|$. Dito em palavras, a função totiente de um inteiro positivo n pode ser definida como a cardinalidade do conjunto dos inteiros coprimos com n e menores que ele.

O próximo teorema mostra como calcular o valor de $\phi(n)$ quando n for uma potência de primo, ou seja, $n = p^k$.

Teorema 3.2.23. *A função $\phi(p^k)$ é dada por $p^k - p^{k-1}$.*

Prova: A prova é bem simples. Como $mdc(a, p^k) = 1$ se, e somente se, a não é múltiplo de p e há p^{k-1} múltiplos de p no intervalo $1 \leqslant a \leqslant p^k$, então basta subtrair p^{k-1} do total p^k

c.q.d

[2] A função totiente recebeu esse nome através do matemático inglês James Joseph Sylvester em seu artigo "*On certain ternary cubic-form equations*", publicado no *American Journal of Mathematics*, de 1879.

No caso particular de n ser primo (representado como p), tem-se $\phi(p) = p-1$.

Teorema 3.2.24. *A função $\phi(n)$ é multiplicativa.*

Prova: É necessário provar que, se $mdc(n,m) = 1$ então $\phi(nm) = \phi(n)\phi(m)$. Para isso, considera-se os números $1, 2, 3, \ldots, nm$, arrumados em forma matricial assim:

$$\begin{array}{ccccc} 1 & 2 & 3 & \cdots & n \\ n+1 & n+2 & n+3 & \cdots & 2n \\ 2n+1 & 2n+2 & 2n+3 & \cdots & 3n \\ 3n+1 & 3n+2 & 3n+3 & \cdots & 4n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (m-1)n+1 & (m-1)n+2 & (m-1)n+3 & \cdots & mn \end{array} \tag{3.8}$$

Observa-se nessa disposição dos números que, como $mdc(ni+j, n) = mdc(j, n)$, então, caso um número nessa disposição seja primo relativo com n, então todos os números na mesma coluna são primos relativos com n. Logo existem $\phi(n)$ colunas nas quais todos os números são primos relativos com n. Por outro lado, toda coluna possui um conjunto completo de restos módulo m: se duas entradas são tais que $(ni_1 + j \equiv ni_2 + j) \mod m$, então $(i_1 \equiv i_2) \mod m$ pois n é invertível módulo m já que $mdc(m,n) = 1$, logo como $0 \leqslant i_1, i_2 < m$ devemos ter $i_1 = i_2$. Desta forma, em cada coluna existem exatamente $\phi(m)$ números que são primos relativos com m e portanto o total de números nesta tabela que são simultaneamente primos relativos com m e n (ou seja, primos com nm) é $\phi(nm) = \phi(n)\phi(m)$.

c.q.d

Para exemplificar, sejam $n = 12$ e $m = 5$, tem-se $mdc(5, 12) = 1$, $\phi(12) = 4$

e $\phi(5) = 4$. Calculando $\phi(12) \times \phi(5) = 4 \times 4 = 16$. Obtém-se o mesmo resultado fazendo $\phi(12 \times 5) = \phi(60) = 16$, pois os inteiros coprimos com 60 são:

$$(\mathbb{Z}/60\mathbb{Z})^* = \{1, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 49, 53, 59\}$$

Dos dois teoremas acima, resulta o seguinte resultado:

Se o inteiro positivo n, na sua forma fatorada, for escrito desta maneira: $n = p_1^{k_1} \times p_2^{k_2} \times p_3^{k_3} \times \ldots p_k^{k_r}$, então:

$$\phi(n) = \phi(p_1^{k_1} \times p_2^{k_2} \times p_3^{k_3} \times \ldots p_n^{\alpha_n}) \tag{3.9}$$

$$\phi(n) = \phi(p_1^{k_1}) \times \phi(p_2^{k_2}) \times \phi(p_3^{k_3}) \times \ldots \phi(p_k^{k_r}) \tag{3.10}$$

$$\phi(n) = (p_1^{k_1} - p_1^{k_1-1}) \times (p_2^{k_2} - p_2^{k_2-2}) \times (p_3^{k_3} - p_3^{k_3-3}) \times \ldots (p_r^{k_r} - p_r^{k_r-r}) \tag{3.11}$$

$$\phi(n) = p_1^{k_1}(1 - \frac{1}{p_1}) \times p_2^{k_2}(1 - \frac{1}{p_2}) \times p_3(1 - \frac{1}{p_3}) \times \ldots p_r^{k_r}(1 - \frac{1}{p_r}) \tag{3.12}$$

Finalmente, chega-se à fórmula geral para $\phi(n)$:

$$\phi(n) = n \prod_{p|n} (1 - \frac{1}{p}) \tag{3.13}$$

Outra conclusão que é possível saber acerca da função totiente é que:

Teorema 3.2.25. *A função $\phi(n)$ não é completamente multiplicativa.*

Prova: Basta encontrar contraexemplos. Usando valores já conhecidos, sejam $n = 12$ e $m = 15$, onde $mdc(12, 15) = 3$. O valor de $\phi(12) = 4$, já visto antes. Falta calcular o valor de $\phi(15)$. Simples, basta $\phi(15) = \phi(3 \times 5) = \phi(3) \times \phi(5) = (3-1) \times (5-1) = 8$. Portanto, $\phi(12) \times \phi(15) = 4 \times 8 = 32$, o que não corresponde ao valor $\phi(12 \times 15) = \phi(180) = \phi(2^2 \times 3^2 \times 5) = \phi(2^2) \times \phi(3^2) \times \phi(5) = (2^2-2)(3^2-3)(5-1) = 48$

c.q.d

Definição 3.2.26. ***Semiprimo:*** *Um número inteiro n é chamado de semiprimo se ele for o produto de dois números primos p e q, ou seja, $n = pq$.*

No caso particular de n ser semiprimo (representado como pq), tem-se:

$$\phi(n) = \phi(pq) = (p-1)(q-1) \tag{3.14}$$

Nota: O resultado anterior é muito importante. Ele mostra que, de posse de p e q, o cálculo de $\phi(n)$ é imediato. Contudo, se a fatorização de n não for conhecida, também não se conhece o valor de $\phi(n)$. Este interessante fato é utilizado como base para o algoritmo RSA.

A seguir, serão apresentados dois teoremas que caminham juntos: o teorema de Fermat e o teorema de Euler.

Teorema 3.2.27. ***Teorema de Fermat:*** *Sejam p um primo e a um inteiro tal que p não divide a. Então* $(a^{p-1} \equiv 1) \mod p$.

Prova: Dado o conjunto de inteiros $A = \{a, 2a, 3a, \ldots, (p-1)a\}$ e escolhendo dois elementos quaisquer desse conjunto, eles não serão congruentes entre si, módulo p. É fácil de ver isso porque, se $(xa \equiv ya) \mod p$, onde $1 \leqslant x, y \leqslant (p-1)$, seria possível cancelar a em ambos os lados, pois $mdc(a, p) = 1$. Ter-se-ia então $(x \equiv y) \mod p$, o que não pode acontecer, pois os elementos do conjunto $B = \{1, 2, 3, \ldots, (p-1)\}$ não são congruentes entre si, módulo p. Pode-se concluir que os elementos de A são congruentes aos elementos de B em alguma ordem. O que resulta em $(p-1)$ congruências da forma:

$$\begin{aligned} (a \equiv x_1) \mod p \\ (2a \equiv x_2) \mod p \\ (3a \equiv x_3) \mod p \\ ((p-1)a \equiv x_{p-1}) \mod p \end{aligned} \tag{3.15}$$

Onde $x_1, x_2, x_3 \ldots x_{p-1}$ são os elementos de B em alguma ordem. Multiplicando membro a membro todas essas equações, obtem-se:

$$(a \times 2a \times 3a \times \cdots \times (p-1)a \equiv 1 \times 2 \times 3 \times \cdots \times (p-1)) \mod p \tag{3.16}$$

O que é equivalente a:

$$((p-1)!a^{p-1} \equiv (p-1)!) \mod p \tag{3.17}$$

Como $mdc(p,(p-1)!)=1$, é possível cancelar e assim:

$$(a^{p-1} \equiv 1) \mod p \tag{3.18}$$

c.q.d.

Em 1747, Euler conseguiu uma generalização do teorema de Fermat.

Teorema 3.2.28. ***Teorema de Euler****: Sejam a e n inteiros, com $n \geqslant 1$, tais que $mdc(a,n)=1$. Então $(a^{\phi(n)} \equiv 1) \mod n$.*

Prova: A prova deste teorema possui quase a mesma estrutura da prova do teorema de Fermat, mas agora, em vez de um primo p, tem-se um inteiro positivo n, tal que $mdc(n,a)=1$. Como $mdc(n,(n-1)!) \neq 1$, não se poderia cancelar o termo como foi feito em 3.17. É necessário fazer uma leve mudança na definição do conjunto A.

Dado o conjunto de inteiros $A=\{x_1,x_2,x_3,\ldots,x_t\}$, formado por todos os inteiros compreendidos no intervalo $1 \leqslant x_i \leqslant (n-1)$ e que são coprimos com n, em outras palavras, $A=(\mathbb{Z}/n\mathbb{Z})^*$. A definição do conjunto B também será alterada, $B=\{x_1a,x_2a,x_3a,\ldots,x_ta\}$. Como x_i é relativamente primo com n, então o resto da divisão de x_ia por n deve ser um outro elemento x_j de A. Assim, as congruências abaixo são válidas:

$$\begin{aligned}(x_1 a \equiv y_1) \mod n \\ (x_2 a \equiv y_2) \mod n \\ (x_3 a \equiv y_3) \mod n \\ (x_t a \equiv y_t) \mod n\end{aligned} \tag{3.19}$$

Onde $y_1, y_2, y_3 \ldots y_y$ são os elementos de A em alguma ordem. Multiplicando membro a membro todas essas equações, obtem-se:

$$(x_1 a \times x_2 a \times x_3 a \times \cdots \times x_t a \equiv x_1 \times x_2 \times x_3 \times \cdots \times x_t) \mod n \tag{3.20}$$

O que é equivalente a:

$$(x_1 \times x_2 \times x_3 \times \cdots \times x_t \times a^t \equiv x_1 \times x_2 \times x_3 \times \cdots \times x_t) \mod n \tag{3.21}$$

Como cada x_i é coprimo com n, é possível cancelar todos, isso resulta em:

$$(a^t \equiv 1) \mod n \tag{3.22}$$

Mas t é justamente a cardinalidade do conjunto A, logo, $t = |A| = |(\mathbb{Z}/n\mathbb{Z})^*|$, o que equivale à função totiente, portanto, $t = \phi(n)$.

$$(a^{\phi(n)} \equiv 1) \mod n \tag{3.23}$$

Quando n é um primo p, $\phi(p) = p - 1$, e assim $(a^{p-1} \equiv 1) \mod n$, o que resulta no teorema de Fermat, ou seja, este é um caso particular do teorema de Euler.

c.q.d.

Uma última observação acerca da exponenciação modular. Numa exponenciação normal sobre os inteiros, o resultado cresce rapidamente e torna-se difícil lidar com números tão compridos. Para se ter uma ideia, o maior número primo conhecido atualmente é $2^{74.207.281} - 1$, esse número foi descoberto em janeiro de 2016 e tem cerca de 22.338.618 de dígitos (equivalente a um arquivo texto com mais de 22MB de tamanho!). Enquanto isso, um expoente de apenas 74.207.281 é minúsculo para uma exponenciação modular. Nos algoritmos criptográficos, os expoentes modulares podem ter mais de 400 dígitos! Como isso é possível? Essa facilidade se deve ao fato de que existe um algoritmo em tempo polinomial para calcular exponenciações modulares, o que não ocorre para a exponenciação comum.

Acredita-se que, com esses conceitos iniciais sobre funções aritméticas e, em especial, sobre a função totiente e o teorema de Euler, é possível entender completamente o algoritmo RSA.

3.3 Criptografia

Today we live in the information age. Everyone has computers. All the trappings of the information age are part of our lives. Paper mail is being replaced by electronic mail. Digital communications is taking over. We need encryption. The common person needs encryption to function effectively in the information age. So it's time for cryptography to step out of the shadows of spies and military stuff, and step out into the sunshine and be embraced by the rest of us.

Phil Zimmermann

Numa entrevista para rádio, durante o programa *High Tech Today*, em 02 de fevereiro de 1996, Phil Zimmermann, o criador do PGP (*Pritty Good Privacy*), com propriedade, afirmou que "é hora de a criptografia sair das sombras dos espiões e militares, caminhar à luz do dia e ser adotada por todos nós". Realmente, nas últimas décadas, é possível ver como a criptografia entrou no dia-a-dia das pessoas. Atualmente, o cidadão médio já deve ter, ao menos, ouvido falar de criptografia. Ele não sabe como ela funciona direito, mas sabe que é uma coisa que traz confiança.

Na próxima seção, alguns conceitos básicos de criptografia serão esclarecidos para o melhor entendimento do assunto tratado nesta dissertação.

3.3.1 Vocabulário

A primeira coisa que deve ser esclarecida é que a criptografia é uma técnica, enquanto a criptologia é a ciência à qual a criptografia pertence. Como é sempre bom começar pelo começo, o primeiro conceito a ser definido é o de criptologia.

Definição 3.3.1. ***Criptologia****: Ciência que reúne os conhecimentos (matemáticos, computacionais, psicológicos, filológicos etc.) e técnicas necessários à criptografia (e à sua contraparte: a criptoanálise) e à esteganografia (e à sua contraparte: a esteganálise).*

Observação: Cada conceito secundário (criptografia, criptoanálise, esteganografia, esteganálise) que aparece nessa definição será explicado posteriormente.

Somente a partir do século XX é que a criptologia ganhou o aspecto de ciência como se conhece hoje. Antes ela estava mais perto de ser uma arte, o que não surpreende, pois, como já disseram vários cientistas e filósofos, "o homo-faber precede o homo-sapiens".

Antes de aprofundar os conceitos básicos da Criptologia, é necessário ter em mente o conceito de fluxo normal de uma comunicação.

Definição 3.3.2. ***Fluxo normal de uma comunicação****: Modelo teórico de como ocorre a comunicação entre duas entidades. Contém 3 partes fundamentais: o emissor, o receptor e o canal de comunicação.*

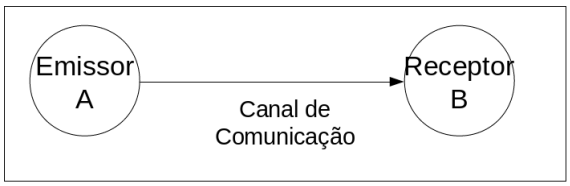

Figura 3.1: Fluxo normal de uma comunicação

Nesse modelo teórico, o emissor (representado pela letra A) tem como objetivo transmitir uma mensagem (representada pela letra m) ao receptor (representado pela letra B).

Deve-se notar que também ocorre a comunicação do receptor para o emissor, mas o modelo continua válido, pois os papéis se invertem. Em seu famoso artigo "*A Mathematical Theory of Communication*", ver ref. [16] SHANNON, Claude Shannon descreveu de forma precisa esse objetivo:

> The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have meaning; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem.

O próximo passo é estabelecer a diferença entre cifra e código.

Definição 3.3.3. ***Cifra****: Algoritmo que atua na representação de uma mensagem, modificando-a de forma que somente pessoas autorizadas (que conhecem um "segredo" ou comumente uma "chave") podem desfazer a modificação, recuperando a mensagem original.*

Definição 3.3.4. ***Código****: Alteração no significado de uma mensagem, modificando-o de forma que somente pessoas autorizadas (que conhecem como se dá a mudança de significados) podem desfazer a modificação, recuperando o significado original.*

Apesar de os leigos tratarem os dois conceitos como sinônimos, a diferença entre cifra e código é evidente. Enquanto a cifra funciona a partir de um algoritmo, o código funciona com a substituição direta no significado da mensagem. Por isso, para a utilização de um código, basta uma tabela. Assim, existem diversos códigos públicos: código Morse, código fonético internacional, código de barras, código QR e dezenas de outros.

Estabelecer a diferença entre cifra e código é fundamental para definir os verbos derivados: cifrar, decifrar, codificar e decodificar.

Definição 3.3.5. ***Mensagem original (ou mensagem em claro)****: O conceito é evidente, a mensagem na sua forma original, ou seja, antes de ser submetida a uma cifra ou a um código.*

Definição 3.3.6. ***Cifrar****: Consiste em submeter uma mensagem em claro ao processo direto de cifração. O resultado é uma mensagem cifrada. Costuma-se representar a cifração de uma mensagem m por uma função $E(m)$.*

Definição 3.3.7. ***Mensagem cifrada****: Mensagem resultante do processo direto de cifração. Costuma-se representá-la pela letra c. Assim, o processo de cifração pode ser representado como $E(m) = c$.*

Definição 3.3.8. ***Decifrar****: Consiste em submeter uma mensagem cifrada ao processo inverso de cifração. O resultado é uma mensagem em claro. Costuma-se representar a decifração de uma mensagem cifrada c por uma função $D(c)$.*

Definição 3.3.9. ***Mensagem decifrada****: Mensagem resultante do processo inverso de cifração, ou seja, deve ser idêntica à mensagem em claro. Assim, o processo de decifração pode ser representado como $D(c) = m$.*

As definições 3.3.5, 3.3.6, 3.3.7, 3.3.8 e 3.3.9 estão representadas na figura 3.2.

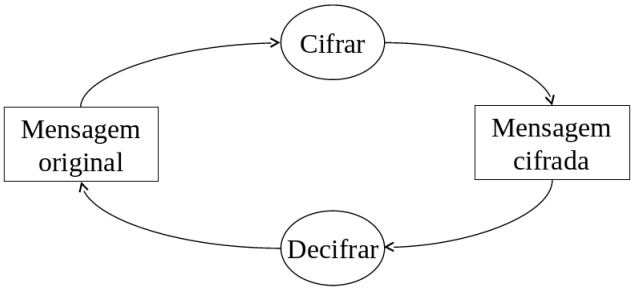

Figura 3.2: Processos direto e inverso de cifração

Definição 3.3.10. ***Codificar****: Consiste em submeter uma mensagem em claro ao processo direto de codificação. O resultado é uma mensagem codificada. O processo de codificação não possui uma forma padronizada ou comumente aceita de representação.*

Definição 3.3.11. ***Mensagem codificada****: Mensagem resultante do processo direto de codificação.*

Definição 3.3.12. ***Decodificar****: Consiste em submeter uma mensagem codificada ao processo inverso de codificação. O resultado é uma mensagem decodificada. O processo de decodificação também não possui uma forma padronizada ou comumente aceita de representação.*

Definição 3.3.13. ***Mensagem decodificada****: Mensagem resultante do processo inverso de codificação, ou seja, deve ser idêntica à mensagem em claro.*

As definições 3.3.5, 3.3.10, 3.3.11, 3.3.12 e 3.3.13 estão representadas na figura 3.3.

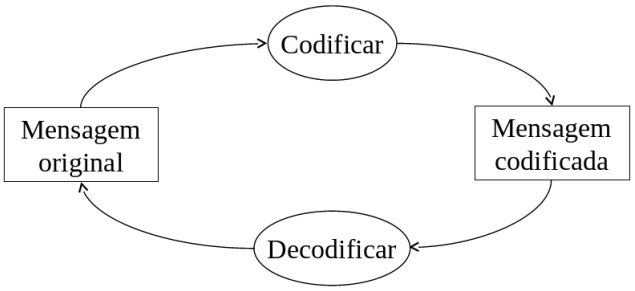

Figura 3.3: Processos direto e inverso de codificação

Definição 3.3.14. ***Quebrar uma cifra (ou um código)****: Afirma-se que uma cifra (ou um código) foi quebrada(o) quando é possível recuperar a mensagem original a partir da mensagem cifrada (ou codificada) sem conhecer o segredo (chave ou tabela de códigos). Isso pode ocorrer para um par específico* (m, c) *ou, em casos mais graves, para todos os pares. Nestes últimos casos, a cifra apresenta uma falha grave em seu algoritmo ou a tabela de códigos caiu em mãos erradas. De qualquer forma, deve-se abandonar o algoritmo de cifração e passar a utilizar outro, ou descartar a tabela de códigos e construir outra.*

Normalmente, quebrar uma cifra é um problema matemático, enquanto quebrar um código é um problema de linguística.

Atualmente os códigos estão em desuso no âmbito da segurança da informação em sistemas computadorizados, mas eles continuam sendo utilizados em várias situações. O crime organizado, por ter ciência que suas comunicações podem estar sendo monitoradas pela polícia, recorre frequentemente a códigos estabelecidos entre eles. Um fato histórico relevante acerca dos códigos é que, durante a Segunda Guerra Mundial, várias cifras e vários códigos foram quebrados, mas somente um código permaneceu seguro durante toda a guerra: o Código Navajo, código utilizado pelos índios norte-americanos Navajos nas comunicações militares. Portanto, não

é aconselhável subestimar o poder dos códigos. Embaixo estão alguns exemplos de palavras da tabela utilizada pelos Navajos:

- "peixe de ferro" = submarino
- "beija-flor" = avião de caça
- "terra vermelha" = batalhão
- "mexicano" = companhia
- "pequena baleia" = cruzador
- "tubarão" = destroyer

Finalmente é possível definir criptografia e criptoanálise.

Definição 3.3.15. ***Criptografia****: Proteger uma mensagem através de um processo de cifração ou codificação.*

Definição 3.3.16. ***Criptoanálise****: Conjunto de conhecimentos e técnicas para se tentar descobrir uma mensagem criptografada (cifrada ou codificada) sem o conhecimento do segredo.*

Portanto, a criptoanálise é a contraparte da criptografia. A criptoanálise acumula conhecimentos e técnicas eficazes para quebrar uma criptografia. Um bom criptólogo deve ter conhecimento das duas áreas.

Agora serão definidos os termos esteganografia e estaganálise.

Definição 3.3.17. ***Esteganografia****: Conjunto de conhecimentos e técnicas para se tentar esconder uma comunicação.*

A diferença entre criptografia e esteganografia deve ficar clara. As duas têm objetivos diferentes, mas complementares. Enquanto a criptografia protege a mensagem, a esteganografia protege a própria existência da comunicação. Um exemplo clássico para entender a diferença é a comunicação entre dois presidentes. Eles podem se comunicar e a existência dessa comunicação pode ser pública, mas o conteúdo pode ser secreto, ou seja, eles podem utilizar a criptografia na comunicação. Mas a comunicação entre um presidente e um criminoso não pode acontecer, ou pelo menos, não se espera que isso ocorra! Caso um presidente queira se comunicar com um criminoso, então, além da criptografia, a própria comunicação precisa ser secreta. Nesse caso, deve-se utilizar criptografia juntamente com a esteganografia.

Definição 3.3.18. ***Estaganálise****: Conjunto de conhecimentos e técnicas para se tentar descobrir a existência de uma comunicação secreta.*

Portanto, a estaganálise é a contraparte da esteganografia. A estaganálise acumula conhecimentos e técnicas eficazes para quebrar uma esteganografia. Um bom criptólogo também deve ter conhecimento dessas duas áreas.

Os objetivos da criptologia foram se ampliando, de forma que hoje, os 4 principais objetivos (também chamados de serviços) da criptologia são:

Definição 3.3.19. ***Confidencialidade****: permitir o acesso à informação apenas para pessoas autorizadas.*

A confidencialidade foi o primeiro objetivo da Criptologia. A primeira pessoa, com registros históricos, a utilizar técnicas para obter confidencialidade foi o imperador romano Júlio César, com uma cifra simples mas muito eficaz para a época, a qual ficou conhecida como Cifra de César.

Definição 3.3.20. ***Integridade****: assegurar a integridade da informação que trafega*

no canal de comunicação, ou seja, assegurar que não há alteração da informação por pessoas não autorizadas.

Definição 3.3.21. ***Autenticação****: assegurar a autenticação das entidades que se comunicam entre si, ou seja, assegurar que a pessoa é realmente quem ela diz que é.*

Definição 3.3.22. ***Irretratabilidade (ou não-repúdio)****: Garantir que o produtor da informação não pode negar a autoria da mensagem.*

É importante deixar claro que um único algoritmo criptográfico não é capaz de fornecer todos os serviços simultaneamente. Isso só é possível pela combinação de vários algoritmos em conjunto.

Definição 3.3.23. ***Princípio de Kerckhoff****: A força de um sistema criptográfico deve estar na chave, ou seja, não é obrigatório manter em segredo o algoritmo utilizado.*

Esse princípio possui várias formas de ser expresso. A definição 3.3.23 é só uma das mais conhecidas. Esse princípio foi mal interpretado ao se considerar obrigatório que o algoritmo seja público e somente a chave seja privada. O que o princípio diz, na verdade, é que, mesmo que o sistema venha a cair em mãos inimigas, a chave deve ser forte o suficiente para não causar desvantagem ao sistema. Claude Shannon, em [17] SHANNON, reescreveu esse princípio da seguinte forma: "*We shall assume that the enemy knows the system being used.*"

Definição 3.3.24. ***Ataque****: Qualquer tentativa de modificação no fluxo normal de uma comunicação é chamada de ataque.*

Definição 3.3.25. ***Interceptação****: O atacante tem acesso às mensagens trafegadas no canal de comunicação. É um ataque à confidencialidade. Ex: grampo em linha telefônica ou na rede de computadores.*

A definição 3.3.25 está representada na figura 3.4.

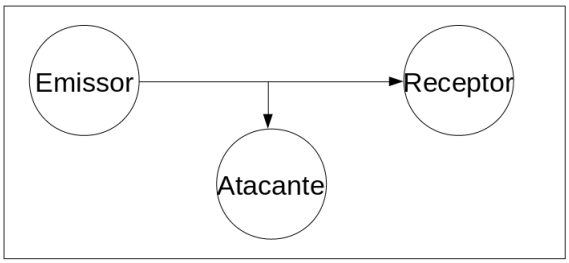

Figura 3.4: Ataque de interceptação

Definição 3.3.26. ***Modificação****: O atacante, além de ter acesso às mensagens, ainda é capaz de modificá-las. É um ataque à integridade. Ex: alterar a conta bancária em uma mensagem, a fim de que o depósito não seja realizado na conta bancária correta, mas na conta bancária indicada pelo atacante.*

A definição 3.3.26 está representada na figura 3.5.

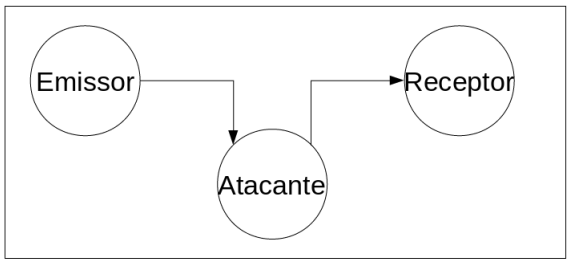

Figura 3.5: Ataque de modificação

Definição 3.3.27. ***Fabricação****: O atacante encaminha mensagens a terceiros se passando por uma outra pessoa. É um ataque à autenticidade. Ex: vírus que envia email da sua caixa postal se passando por você.*

A definição 3.3.27 está representada na figura 3.6.

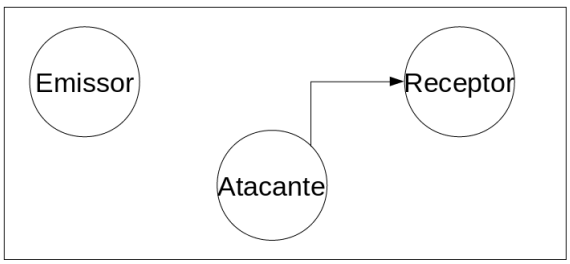

Figura 3.6: Ataque de fabricação

Definição 3.3.28. ***Interrupção****: O atacante interrompe a comunicação. A interrupção é um ataque contra a disponibilidade.*

A definição 3.3.28 está representada na figura 3.7.

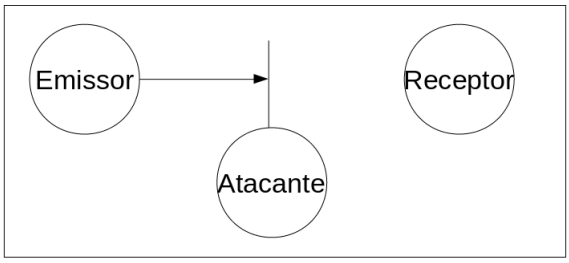

Figura 3.7: Ataque de interrupção

A criptologia é inútil contra um ataque de interrupção! Devido à existência de ameças e problemas fora do âmbito da Criptologia, hoje essa ciência é um sub-ramo de uma ciência maior denominada **Segurança da Informação**.

3.3.2 Criptografia Simétrica

As primeiras cifras criadas apresentavam um padrão que foi denominado "simétrico", ou seja, a mesma chave utilizada pelo emissor deve ser utilizada pelo receptor. Como a chave deveria ser a mesma em ambos os lados do canal de comunicação, fez-se essa referência ao conceito de simetria. Mas não se deve confundir, a simetria está no processo, não na chave em si, como se a chave fosse do tipo palíndromo.

Na prática, esse processo simétrico de criptografia (ou, simplesmente, criptografia simétrica) tem uma desvantagem considerável, pois, para garantir que os dois lados do canal de comunicação tenham a mesma chave, é necessário que a chave seja transmitida em um outro canal que também deve ser seguro. Caso tal canal não exista, deve-se existir um encontro entre o emissor e o receptor para a transmissão da chave.

Por motivos didáticos e tradicionais, costuma-se representar o emissor e o receptor como duas pessoas imaginárias, denominadas Alice e Bob (em vez de simplesmente letras A e B), além desses personagens, existem outros (Charles, David, Erin, Frank etc), a terceira personagem mais comumente utilizada é a atacante Eva (de *eavesdropper*). Portanto é possível montar um cenário imaginário no qual Alice e Bob tentam se comunicar de forma segura, defendendo-se contra os ataques da sagaz Eva. Alice será representada pela letra A, Bob pela letra B e Eva pela letra E. As chaves serão representadas pela letra K. A chave em posse de Alice será representada por K_A, e a chave em posse de Bob será representada por K_B. Eva não precisa ter sua própria chave.

O que de fato caracteriza a criptografia simétrica é que a comunicação só é possível quando $K_A = K_B$.

Esse processo simétrico de criptografia foi tido como único padrão possível durante mais de 20 séculos. A busca por um processo assimétrico foi considerada como uma busca pelo Santo Graal na Criptologia, ou seja, algo quimérico, praticamente impossível, atividade para sonhadores. Felizmente, alguns "sonhadores" conseguiram se manter firme no propósito e conseguiram, nos anos 70, alcançar o Santo Graal da Criptologia com a ajuda da matemática. Toda essa história é muito interessante e vale uma leitura mais aprofundada. Leituras já tradicionalmente recomendadas são O Livros dos Códigos, [18] SINGH, livro que aborda do ponto de vista histórico, e *Decrypted Secrets*, [2] BAUER, livro clássico do assunto.

3.3.3 Funções de Hash

Para um bom entendimento das funções de hash[3], é aconselhável começar com um conceito simples, mas bastante utilizado na matemática.

Definição 3.3.29. ***Princípio da casa dos pombos***[4]*: Este princípio estabelece que, caso se tenha n pombos para serem distribuídos em m casas e* $n > m$*, então ao menos uma casa terá dois pombos. Numa linguagem matemática, se um conjunto finito A tiver mais elementos que um outro conjunto B, então não existe uma função* $f : A \to B$ *tal que* f *seja injetora.*

Como é um princípio, então não é necessário provar. Os princípios estão para a Filosofia, assim como os axiomas estão para a Matemática e os dogmas estão para as Religiões.

[3] De acordo com Donald Knuth, o registro mais antigo de utilização do termo *hash* é um relatório de janeiro de 1953 do pesquisador da IBM Hans Peter Luhn.

[4] O princípio da casa dos pombos também é conhecido como princípio das gavetas de Dirichlet, pois o primeiro relato desse princípio foi feito por Dirichlet em 1834, nomeando-o como *Schubfachprinzip* ("princípio das gavetas").

Embora esse princípio esteja baseado numa evidência simples e imediata, ele pode ser útil para resolver problemas ou descrever fenômenos que, num primeiro contato, não sejam triviais. Para aplicar o princípio corretamente, é necessário estabelecer corretamente quem fará o papel dos pombos e quem fará o papel das casas.

Voltando ao conceito de função de hash, uma forma matemática de definir uma função de hash H é através da seguinte expressão:

$$H(\{0,1\}^n) = \{0,1\}^k, n, k \in \mathbb{N} \tag{3.24}$$

Onde $\{0,1\}^n$ é uma cadeia binária qualquer de tamanho n, e $\{0,1\}^k$ é uma cadeia binária de tamanho fixo k. Como n é variável e k assume um valor fixo, então é possível ter $n > k$. Assim, pelo princípio da casa dos pombos, duas cadeias binárias diferentes podem produzir o mesmo resultado (ou seja, o mesmo hash), em outras palavras, as funções de hash não são unívocas. Quando isso ocorre, dá-se o nome "colisão". Quanto mais raras forem as colisões, melhor é o algoritmo utilizado pela função. Além disso, o processo é unidirecional, o que impossibilita descobrir qual foi a cadeia binária original que produziu o resultado. Matematicamente falando, as funções de hash não são invertíveis (isso é bem evidente, pois a primeira condição para existir a função inversa é que a função seja injetora).

Em Computação, um algoritmo é considerado *determinístico* quando, para uma determinada entrada, ele produzirá sempre a mesma saída a qualquer momento que o algoritmo for executado, além disso, a máquina na qual o algoritmo é executado sempre passará pela mesma sequência de estados. Caso contrário, o algoritmo é considerado probabilístico. Portanto, pelo que foi exposto até agora, as funções de hash são determinísticas, mas não unívocas.

Mas agora surge a pergunta: Para que serve uma função de hash? Para transformar uma grande quantidade de dados numa pequena porção de informação. O que fazer com essa pequena porção de informação? Muitas coisas, na Criptologia, essa pequena porção de informação pode ser utilizada para garantir a integridade de uma mensagem. As funções de hash são muito utilizadas por SGBD's (Sistemas Gerenciadores de Base de Dados), e também em engenharia genética. Na Criptologia, as funções de hash devem atender a graus maiores de exigência ao ponto de, àquelas que conseguem atender a tais exigências, dá-se um nome especial "funções de hash criptográficos". São duas exigências:

1. Se ao menos um bit da cadeia de entrada for alterado, resulta em uma cadeia de saída diferente (esse é o conhecido Efeito Avalancha);
2. Não é viável computacionalmente encontrar duas cadeias de entrada que produzam a mesa cadeia de saída.

Não é fácil desenvolver uma função de hash criptográfico. Muitas já foram propostas e poucas passaram nos testes da comunidade internacional. O MD5 é uma função de hash que se tornou muito conhecida e que permaneceu muito tempo sendo utilizada, mas, com o avanço tecnológico, o tempo de vida útil dela acabou para atividades mais críticas. O MD5 foi criado em 1991, por Ronald Rivest, um dos criadores do algoritmo RSA, que será visto no próximo item, 3.3.4. Atualmente, as funções de hash mais utilizadas formam uma família chamada SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256), essa família foi criada por pesquisadores da NSA (*National Security Agency*) e foi tornada padrão pelo NIST em 2001. Em 2006, o NIST publicou uma competição para encontrar substitutos para o SHA-2. Em 2 de outubro de 2012, o algoritmo Keccak foi escolhido como vencedor da competição. Em 5 de agosto de 2015, o NIST tornou o Keccak o novo padrão e denominou-o de SHA-3.

3.3.4 Criptografia Assimétrica

Os pioneiros na criptografia assimétrica foram os matemáticos Whitfield Diffie e Martin Hellman, essa dupla é mundialmente conhecida como Diffie-Hellman. Eles, de fato, não criaram a criptografia assimétrica, mas descobriram uma forma de tornar segura a transferência de chave do emissor para o receptor num processo simétrico de criptografia. Essa forma segura é através de um algoritmo que tornou-se também mundialmente conhecido como "Algoritmo Diffie-Hellman" ou simplesmente "Algoritmo DH". Ao resolver tal problema, Diffie e Hellman indicaram a "porta de acesso" para a criptografia assimétrica, mas não conseguiram abri-la.

Os pioneiros, de fato, da criptografia assimétrica, aqueles que conseguiram "abrir a porta de acesso", foram três matemáticos do MIT (*Massassuchets Institute of Technology*): Ronald Rivest, Adi Shamir e Leonard Adleman. O algoritmo que eles criaram tornou-se mundialmente conhecido como "algoritmo RSA", a sigla RSA vem dos nomes do trio Rivest-Shamir-Adleman. A partir daí, muitos outros algoritmos surgiram e se estabeleceram.

A criptografia assimétrica tornou-se possível devido à formalização matemática pela qual passou a Criptologia no século XX. Para entender em detalhes como são os algoritmos DH, RSA e todos os outros subsequentes, o interessado deve ter conhecimentos avançados de álgebra: aritmética modular, grupos, anéis, corpos, corpos finitos, geradores de grupos, raízes primitivas, curvas elípticas, reticulados etc. Para esta dissertação, a matemática necessária para o entendimento dos dois algoritmos criptográficos assimétricos a serem usados está descrita no item 3.2.

3.3.4.1 RSA

O RSA será descrito aqui na sua forma pura, original. Atualmente, o RSA é implementado de forma ligeiramente diferente e em várias versões, mas esta descrição será básica para entender qualquer versão.

A função aritmética que é o fulcro do RSA é a *função totiente*. Por isso, é uma forte recomendação ler antes a seção 3.2 para o completo entendimento deste algoritmo.

O cenário é o seguinte. Bob, deseja enviar uma mensagem secreta para Alice. Para isso, Bob precisa conhecer a chave pública de Alice. Neste cenário, Bob não necessita ter um par de chaves.

Etapa 1: Como gerar as chaves

Esta etapa é feita pela Alice.

- Passo 1: Escolher aleatoriamente dois números primos p e q.
- Passo 2: Determinar o semiprimo $n = pq$.
- Passo 3: Aplicar a função totiente em n: $\phi(n) = (p-1)(q-1)$.
- Passo 4: Escolher um inteiro e, tal que $1 < e < \phi(n)$, e que $mdc(e, \phi(n)) = 1$.
- Passo 5: Calcular d de forma que $de \equiv 1 \mod \phi(n)$.
- Passo 6: Publicar a chave pública (n, e) e armazenar de forma segura a chave privada (n, d).

Observação 1: No passo 1, o tamanho dos primos deve ser grande o suficiente para produzir um n de aproximadamente 2048 bits.

Observação 2: No passo 5, utiliza-se o algoritmo de Euclides estendido.

Observação 3: Também no passo 5, devido ao próprio conceito da operação modular, a expressão $de \equiv 1 \mod \phi(n)$ *é equivalente a dizer que existe um inteiro* k *qualquer tal que:*

$$de = 1 + k\phi(n) \tag{3.25}$$

Etapa 2: Como cifrar

Esta etapa é feita pelo Bob.

- Passo 1: Representar a mensagem numa forma numérica. Seja m esse valor numérico.
- Passo 2: Realizar a potenciação modular: $c = m^e \mod n$.
- Passo 3: Enviar a mensagem cifrada c para Alice.

Observação: Existe um algoritmo eficiente para calcular exponenciação modular.

Etapa 3: Como decifrar

Esta etapa é feita pela Alice.

- Passo 1: Realizar a potenciação modular: $m = c^d \mod n$.
- Passo 2: Fazer o processo inverso de recuperar a mensagem original a partir do número m.

Será provado agora por que o algoritmo RSA funciona. Deve-se partir do processo de decifração.

$$c^d \mod n = (m^e)^d \mod n = m^{ed} \mod n$$

Substituindo pela equação 3.25 (ver observação 3 da etapa 1), fica então:

$$m^{1+k\phi(n)} \mod n = m^1 \times m^{k\phi(n)} \mod n = m \times (m^{\phi(n)})^k \mod n$$

Aplicando o Teorema de Euler, o termo $m^{\phi(n)}$ pode ser substituído pelo inteiro 1:

$$m^1 \times (1)^k \mod n = m^1 \times 1 \mod n = m \mod n = m$$

A segurança do RSA está fundamentada na dificuldade de se fatorar grandes números. Afirmar que um número é primo ou composto, é rápido. Existem algoritmos em tempo polinomial para determinar a primalidade de um inteiro. Agora, uma vez que se saiba que um número é composto, como encontrar seus fatores primos? Ainda não existem algoritmos em tempo polinomial para decompor um número inteiro. Quando os fatores são pequenos, a tarefa fica fácil, mas quando os fatores são enormes (ver passo 1 da etapa 1), o tempo de processamento, com a capacidade

computacional atual, é inviável. Enquanto não se descobrir um algoritmo polinomial e os computadores quânticos não forem fabricados, o RSA continuará sendo um forte algoritmo criptográfico.

3.3.4.2 Elgamal

O próximo algoritmo a ser utilizado nesta dissertação é o Elgamal, que recebeu o mesmo nome de seu criador, o criptólogo egípcio Taher Elgamal. Este algoritmo foi proposto em 1985.

Será utilizado o mesmo cenário já descrito anteriormente, com Bob desejando enviar uma mensagem para Alice.

Etapa 1: Como gerar as chaves

Esta etapa é feita pela Alice.

- Passo 1: Escolher um primo q grande (de 2048 bits) e um gerador g para o grupo multiplicativo $(\mathbb{Z}/q\mathbb{Z})^*$.
- Passo 2: Escolher um número aleatório x em $\{1, \ldots, q-1\}$. Esse número x será a chave privada.
- Passo 3: Calcular $h = g^x \mod q$.
- Passo 4: Publicar a chave pública (q, g, h) e armazenar de forma segura a chave privada x.

Etapa 2: Como cifrar

Esta etapa é feita pelo Bob.

- Passo 1: Representar a mensagem numa forma numérica. Seja m esse valor numérico.
- Passo 2: Escolher um número aleatório y em $\{1, \ldots, q-1\}$. Esse número y será a *chave da mensagem*. Essa chave também é conhecida como "chave efêmera", pois ela é usada uma única vez e descartada. Na próxima mensagem, deve-se escolher outra diferente.
- Passo 3: Calcular $c_1 = g^y \mod q$.
- Passo 4: Calcular $s = h^y \mod q$.
- Passo 5: Calcular $c_2 = (m \times s) \mod q$.
- Passo 6: A mensagem cifrada é constituída do par (c_1, c_2). Enviar a mensagem cifrada para Alice.

Etapa 3: Como decifrar

Esta etapa é feita pela Alice.

- Passo 1: Realizar a potenciação modular: $s = c_1^x \mod q$.
- Passo 2: Calcular $s^{-1} \mod q$.
- Passo 3: Calcular $m = (c_2 \times s^{-1}) \mod q$.
- Passo 4: Fazer o processo inverso de recuperar a mensagem original a partir do número m.

A segurança do Elgamal está baseada na dificuldade de se resolver o PLD (Problema do Logaritmo Discreto). Na Física ou na Engenharia, existem diversas formas de se resolver um logaritmo, ou seja, encontrar o valor de $f(x)$ dado por $f(x) = \log_b x, x \in \mathbb{R}$. Há várias séries infinitas que dão um resultado aproximado. Infelizmente, na álgebra discreta, a álgebra que trata apenas de números inteiros, ainda não foi encontrado um algoritmo polinomial para se resolver o problema quando a exponenciação é modular.

Pelo que já foi visto para o RSA e agora para o Elgamal, a segurança dos algoritmos criptográficos repousa na inexistência (pelo menos até agora) de algoritmos de complexidade polinomial para resolver determinados problemas matemáticos, ou seja, só se tem conhecimento de algoritmos de complexidade exponencial. Para o RSA, tem-se o problema da fatoração de grandes números. Para o Elgamal, tem-se o problema do logaritmo discreto para grandes números. Existem outros algoritmos criptográficos baseados também em outros problemas matemáticos ainda sem solução polinomial.

3.3.5 Assinatura Digital

A assinatura digital é o ápice da Criptologia. É o mais refinado serviço prestado por essa ciência. A assinatura digital nasceu com criptografia assimétrica. Serão respondidas algumas perguntas para introduzir o assunto.

1) Por que a criptografia simétrica não permite assinaturas digitais?

Porque, na criptografia simétrica, para duas ou mais pessoas se comunicarem, elas devem compartilhar a mesma chave. Mas, dada uma mensagem específica criptografa com essa mesma chave, não será possível afirmar quem a produziu, pois todos

conhecem a chave. Ou seja, a criptografia simétrica não permite a irretratabilidade (definição 3.3.22), propriedade também conhecida conhecida como "não-repúdio".

2) O que acontece quando, numa comunicação com criptografia assimétrica, o proprietário da chave privada criptografa a mensagem com essa mesma chave?

Já foi visto antes que, na criptografia assimétrica, *"uma chave desfaz o que a outra faz"*. Quando uma mensagem é criptografada com a chave privada X, quem poderá desfazer a operação é quem tem a chave pública Y, ou seja, qualquer pessoa portando a chave pública. Nesse caso, percebe-se que não se busca a confidencialidade (pois quem possui a chave pública poderá recuperar a mensagem), o que se tem aí é a garantia que a mensagem veio daquela pessoa específica, ou seja, dois serviços de uma só vez: a autenticidade e a irretratabilidade.

Autenticidade e irretratabilidade são a base da assinatura digital. Quando esse conceito surgiu inicialmente (no artigo [5] DIFFIE), o procedimento era exatamente o descrito na pergunta 2). Anos depois, o procedimento de assinatura digital sofreu uma leve modificação, sendo introduzida uma operação de hash da mensagem, a criptografia com a chave privada passa a ser realizada em cima do hash da mensagem e não na mensagem em si.

Já foi visto que uma função de hash é do tipo $H(\{0,1\}^n) = \{0,1\}^k$, onde $n, k \in \mathbb{N}$ e n assume qualquer valor e k é fixo. Como a saída da função de hash tem um tamanho já conhecido, fica melhor para definir um tamanho de chave criptográfica privada.

3.3.6 Criptografia Homomórfica

Por muitos anos, a Criptografia Assimétrica foi considerada, numa visão otimista, o Santo Graal da Criptologia, numa visão pessimista, uma impossibilidade definitiva. Mas eis que a busca chegou ao fim e o mundo conheceu a Criptografia Assimétrica a partir dos anos 70. No entanto, as ciências são dinâmicas, o conhecimento avança e novas metas devem ser estabelecidas. Neste momento, a Criptologia estabeleceu como nova meta a ser alcançada a Criptografia Completamente Homomórfica. O que vem a ser isso?

O homomorfismo, na verdade, é um atributo de uma criptografia, algumas têm, outras não. Consiste na propriedade de ser possível realizar operações algébricas com textos cifrados, sem o conhecimento do texto original, mas o resultado final reflete no texto original. Algumas criptografias são homomórficas para a adição, outras, para a multiplicação, algumas outras para a operação Xor. Algumas outras, para nenhuma operação, ou seja, não apresentam homomorfismo. E qual a utilidade disso? A aplicação mais evidente dessa propriedade e que costuma aparecer em qualquer texto introdutório é a contagem de votos criptografados.

Supondo que haja uma votação. As pessoas votam numa máquina que criptografa seu voto. Representando uma pessoa pela letra i e seu respectivo voto por v_i, então os votos criptografados serão dados por $c_i = E(v_i)$, onde $E(x)$ é a função de encriptação. Se a cifra tiver a propriedade homomórfica para a adição, então somando todos os votos cifrados $E(v_1) + E(v_2) + E(v_3) + \cdots + E(v_n)$, onde n é o total de votantes, resultará na cifra da soma de todos os votos $E(v_1+v_1+v_1+\ldots v_n)$. Com a decifração desse valor, obtém-se o valor final da eleição, sem que os votos individuais tenham sido publicados.

A propriedade do homomorfismo já é bem conhecida para diversas cifras, mas

não se conhecem ainda cifras que sejam homomórficas para duas ou mais operações. As primeiras, foram denominadas cifras parcialmente homomórficas, enquanto este outro grupo, ainda não encontrado, fora denominado de cifras completamente homomórficas. Logo, descobri-las se tornou atualmente o grande desafio da Criptologia.

3.3.6.1 RSA Original

O RSA como foi proposto originalmente possui a propriedade homomórfica para a multiplicação, como mostrado abaixo:

Dados de entrada: chave pública (e, N)

Encriptação da primeira mensagem:

$$RSA(m_1) = (m_1)^e \mod N$$

Encriptação da segunda mensagem:

$$RSA(m_2) = (m_2)^e \mod N$$

Multiplicando as duas mensagens encriptadas:

$$RSA(m_1) \times RSA(m_2) = (m_1)^e \times (m_2)^e \mod N = (m_1 \times m_2)^e \mod N = RSA(m_1 \times m_2)$$

3.3.6.2 Elgamal

O algoritmo Elgamal também possui a propriedade homomórfica para a multiplicação, como mostrado abaixo:

Dados de entrada: um grupo cíclico G de ordem q, um gerador g, e um número $h = g^x \mod q$, onde x é a chave privada. Em outras palavras, a entrada é uma tupla ternária (q, g, h).

Encriptação da primeira mensagem:

$EG(m_1) = (c_{1,1}, c_{1,2}) = (g^y, m_1 \times h^y)$, onde y é a chave efêmera da mensagem m_1.

Encriptação da segunda mensagem:

$EG(m_2) = (c_{2,1}, c_{2,2}) = (g^z, m_2 \times h^z)$, onde z é a chave efêmera da mensagem m_2.

Multiplicando as duas mensagens encriptadas:

$$\begin{aligned} EG(m_1) \times EG(m_2) &= (c_{1,1} \times c_{2,1}, c_{1,2} \times c_{2,2}) \\ &= (g^y \times g^z, m_1 \times h^y \times m_2 \times h^z) \\ &= (g^{y+z}, m_1 \times m_2 \times h^{y+z}) = EG(m_1 \times m_2) \end{aligned}$$

Todas as operações no Elgamal são *mod q*.

Como é possível observar nos dois algoritmos, a multiplicação das mensagens encriptadas resulta na encriptação da multiplicação das mensagens originais. Para este trabalho, serão exigidas criptografias homomórficas para a multiplicação.

4 SOLUÇÕES PROPOSTAS

Não me desencorajo, porque cada tentativa errada descartada é outro passo à frente.

Thomas Alva Edison (1847-1931)

Quando este problema apareceu no Inmetro, imaginou-se que haveria uma solução simples para o mesmo, mas as primeiras soluções que apareceram foram derrubadas porque foram detectadas fragilidades graves em cada uma delas. Por fim, restaram três soluções que serão apresentadas neste capítulo. Inicialmente, é necessário fazer uma observação. Em todas as soluções propostas, um diferente comando é proposto. Para quem não conhece as atividades do Inmetro, esse procedimento pode causar dúvidas ou mau entendimento. A fim de que isso não ocorra, deve-se esclarecer como um novo comando passa a fazer parte do conjunto de comandos que um instrumento é capaz de executar.

O Inmetro publica regulamentos aos quais os instrumentos devem atender. Quando surge a necessidade de um novo comando, este passa a ser exigido na próxima versão do regulamento. Os fabricantes devem implementar o novo comando nos novos modelos de instrumento. Depois disso, os fabricantes devem submeter o novo modelo de instrumento ao processo de Avaliação de Modelo realizado pela Diretoria de Metrologia Legal (já descrito no item 3.1.2). Fica evidente que, do momento em que um novo comando é proposto, até o momento que os primeiros instrumentos começam a ser vendidos, pode demorar mais de um ano.

Para todas as soluções descritas, a ordem de execução das atualizações não é

relevante, ou seja, a equipe responsável pode começar pela atualização de qualquer instrumento, e depois um outro qualquer, assim sucessiva e aleatoriamente. Também considera-se que os técnicos não têm conhecimento das chaves privadas ou secretas dos instrumentos.

De acordo com a solução, poderão existir cálculos prévios à execução ou cálculos posteriores à execução. Evitou-se ao máximo cálculos feitos pelo responsável pela atualização, mas não foi possível suprimi-los. A primeira solução é a que exige menos cálculo (nenhum prévio e poucos posteriores), mas é também a menos segura. A segunda solução também não exige cálculo prévio, mas exige bastante cálculo posterior. A terceira solução exige muito cálculo prévio, mas pouquíssimo cálculo posterior.

Nesta dissertação, será utilizada a mesma notação dos padrões NIST (ver ref. [14] NIST): X para chave privada, Y para chave pública. Outras representações utilizadas são: cifras assimétricas E_1 (para função de encriptação), e portanto D_1 (para função de decriptação); cifras parcialmente homomórficas E_2 e D_2 (para as funções de encriptação e decriptação, respectivamente).

Representando a quantidade de dispositivos em uso como N e identificando um dispositivo qualquer pela letra i, onde $1 \leqslant i \leqslant N$. A notação fica estabelecida na tabela 4.1.

All notation should be as simple as the nature of the operation to which it is applied.

Charles Babbage (1791-1871)

Tabela 4.1: Notação

Símbolo	**Significado**
Id_i	Identificador do dispositivo i.
X_i	Chave criptográfica privada do dispositivo i.
Y_i	Chave criptográfica pública do dispositivo i.
X_r	Chave criptográfica privada do responsável pela atualização.
Y_r	Chave criptográfica pública do responsável pela atualização.
H_i	Hash da versão de software instalada no dispositivo i.
H	Hash da nova versão de software.
P_i	Número primo aleatório gerado para o dispositivo i.
$E_1(M, Z_i) = C$	Encriptação da mensagem em claro M com qualquer das chaves Z_i do dispositivo i, produzindo a mensagem cifrada C.
$D_1(C, Z_i) = M$	Decriptação da mensagem cifrada C com qualquer das chaves Z_i do dispositivo i, produzindo a mensagem em claro M.
$E_2(M, Z_r) = C$	Encriptação homomórfica da mensagem em claro M com qualquer das chaves do responsável pela atualização, produzindo a mensagem cifrada C.
$D_2(C, Z_r) = M$	Decriptação homomórfica da mensagem cifrada C com qualquer das chaves do responsável pela atualização, produzindo a mensagem em claro M.

4.1 Solução com Criptografia Simétrica

Esta solução utiliza criptografia simétrica, portanto, caso haja necessidade de rever algum conceito, recomenda-se a leitura do tópico 3.3.2.

4.1.1 Requisitos Funcionais

Esta solução exige os seguintes requisitos funcionais.

Requisito 1: Os instrumentos deverão armazenar chaves criptográficas simétricas S_i.

Requisito 2: O responsável pela atualização deverá ter conhecimento da chave de cada instrumento. Isso, na prática, significa que o responsável deverá possuir uma tabela relacionando cada identificador de instrumento com a sua respectiva chave criptográfica.

Requisito 3: Os instrumentos deverão executar um novo comando. O instrumento i deve criptografar o valor H_i (hash da sua versão instalada) com a sua chave criptográfica simétrica S_i. O novo comando pode ser resumido na forma abaixo:

Identificador do comando: CMD01
Entrada do Comando: vazia
Saída do Comando: $C_i = Enc(H_i, S_i)$

4.1.2 Cálculos Prévios

Cálculos prévios, ou seja, cálculos realizados pelo responsável pela atualização, não são necessários.

4.1.3 Execução

A equipe de técnicos realiza a atualização do firmware de cada um dos N instrumentos. A cada atualização, a equipe deve executar o comando descrito no requisito 3 e deve também armazenar as respostas C_i obtidas dos instrumentos.

4.1.4 Cálculos Posteriores

Ao final da atualização da versão nos N instrumentos, a equipe terá posse de todos os $C_i's$, ou seja, $\{C_1, C_2, C_3, \ldots, C_N\}$. Todos esses valores devem ser informados ao responsável pela atualização e este deverá verificar cada C_i da seguinte forma: $H_i = Dec(C_i, S_i)$, ou seja, pela decriptação de cada C_i, obtem-se o respectivo H_i. Estes devem ser todos iguais a H. Se algum for diferente, é uma evidência que o instrumento não foi atualizado.

4.1.5 Possibilidade de Recibo Agregador

Esta solução permite a utilização de um recibo agregador. Como o responsável pela atualização conhece todas as chaves secretas S_i de cada instrumento i e também conhece o valor do hash H da nova versão do software, então é possível definir um recibo agregador RA como sendo o resultado da operação Xor entre todos

os $C_i's$, da seguinte forma:

$$RA = C_1 \oplus C_2 \oplus C_3 \oplus \cdots \oplus C_N \tag{4.1}$$

4.1.6 Análise da solução

A possibilidade de utilizar um RA é uma grande vantagem. É uma solução fácil de implementar. Além disso, esta solução carrega em si todas as vantagens e desvantagens da criptografia simétrica: ela é leve e rápida, mas a chave secreta precisa ser protegida com muito cuidado. Na prática, com a rotatividade de funcionários, o risco de toda a tabela de chaves cair em mãos erradas é muito alto. E, segundo o Princípio de Kerckhoff (definição 3.3.23), as chaves são os pontos mais sensíveis de um sistema de segurança. Outra característica da solução é ser *determinístíca*, pois cada recibo individual, como foi proposto $C_i = Enc(H_i, S_i)$ é determinístico e, ao se fazer a agregação de todos os recibos individuais através da operação Xor, o resultado final, o RA, também se torna determinístico.

O fato de ser determinística não significa que a solução seja unívoca, pois duas situações diferentes (dois conjuntos diferentes de instrumentos) podem resultar no mesmo RA. Isso, na prática, não compromete a segurança, desde que o tamanho em bits dos $C_i's$ seja relativamente grande.

4.2 Solução com Assinaturas Digitais

Esta solução utiliza assinatura digital, portanto, caso haja necessidade de rever algum conceito, recomenda-se a leitura do tópico 3.3.5.

Nesta segunda e mais segura solução para o problema, cada instrumento, identificado por Id_i, criptografa, com a respectiva chave privada X_i, o hash H_i da versão de software instalada nele. Para isso, deverá haver um comando para solicitar esse procedimento (esse comando não existe atualmente nos regulamentos). A saída do comando será então $C_i = Enc(H_i, X_i)$. A equipe responsável pela atualização dos medidores deve, após cada atualização realizada, executar esse comando e armazenar o resultado enviado pelo instrumento.

4.2.1 Requisitos Funcionais

Esta solução exige os seguintes requisitos funcionais.

Requisito 1: Os instrumentos deverão armazenar chave criptográfica assimétrica X_i (chave privada).

Requisito 2: O responsável pela atualização deverá ter conhecimento da chave pública Y_i de cada instrumento. Isso, na prática, significa que o responsável deverá possuir uma tabela relacionando cada identificador de instrumento com a sua respectiva chave criptográfica pública.

Requisito 3: Os instrumentos deverão executar um novo comando. O instrumento i deve criptografar o valor H_i (hash da sua versão instalada) com a sua chave criptográfica privada X_i. O novo comando pode ser resumido na forma abaixo:

Identificador do comando: CMD02

Entrada do Comando: vazia

Saída do Comando: $AD_i = Enc(H_i, X_i)$

Observação: Como visto no item 3.3.5, o que está sendo feito neste comando (criptografar o hash com chave privada) é, na verdade, a assinatura digital do firmware instalado no instrumento. Por isso, os $C_i s$ serão chamados de $AD_i' s$ (AD de Assinatura Digital).

4.2.2 Cálculos Prévios

Cálculos prévios, ou seja, cálculos realizados pelo responsável pela atualização, não são necessários.

4.2.3 Execução

A equipe de técnicos realiza a atualização do firmware de cada um dos N instrumentos. A cada atualização, a equipe deve executar o comando descrito no requisito 3 e deve também armazenar as respostas $AD_i' s$ obtidas dos instrumentos.

4.2.4 Cálculos Posteriores

Ao final da atualização da versão nos N instrumentos, a equipe terá posse de todas as $AD_i' s$, ou seja, $\{AD_1, AD_2, AD_3, \ldots, AD_N\}$. Todos esses valores devem ser informados ao responsável pela atualização e este deverá verificar cada AD_i da seguinte forma: $H_i = Dec(AD_i, Y_i)$, ou seja, pela decriptação de cada AD_i, obtém-se

o respectivo H_i. Estes devem ser todos iguais a H. Se algum for diferente, é uma evidência que o instrumento não foi atualizado.

4.2.5 Possibilidade de Recibo Agregador

Esta solução não permite a definição de um RA, devido à impossibilidade de prever como serão as assinaturas digitais.

4.2.6 Análise da solução

Vantagem: É uma solução simples de implementar. Os instrumentos de medição já estão saindo com chaves privadas armazenadas neles e com capacidade para realizar assinaturas digitais.

Desvantagem: Para o responsável pela atualização, fica o encargo de verificar cada assinatura digital recebida, ou seja, esta solução não permite a implementação de um RA. É também uma solução determinística.

4.3 Solução com Criptografia Homomórfica

Do what you can, with what you've got, where you are.

in "Theodore Roosevelt: An Autobiography", chapter IX, p.337

As duas soluções anteriores apresentam uma sutil desvantagem. Se o fabri-

cante decide criar um novo comando como descrito abaixo:

Identificador do comando: CMD03
Entrada do Comando: Uma informação qualquer M.
Saída do Comando: $C_i = Enc(M, K_i)$

Onde a chave K_i pode ser tanto a chave privada como a chave secreta do instrumento i. Esse simples comando, que pode ser útil em muitas situações, torna as duas soluções anteriores completamente inseguras. De fato, basta trocar o valor de M por H e assim produz-se a resposta esperada na solução, sem o firmware estar atualizado. Seria necessário proibir o fabricante de implementar tal comando. Essa vulnerabilidade das soluções se deve também ao fato de as duas serem determinísticas, ou seja, produzirão sempre a mesma resposta: o hash criptografado ou com a chave secreta (no caso da primeira solução) ou com a chave privada (no caso da segunda solução).

A solução descrita neste item resolve os problemas de segurança das soluções anteriores e ainda permite a implementação de um RA. A solução insere elementos de aleatoriedade previamente calculados, os quais são números primos aleatórios que serão associados a cada instrumento. Cada número primo é enviado ao seu respectivo instrumento criptografado com a chave pública deste. Assim, somente aquele determinado instrumento poderá desfazer a criptografia e extrair o seu número primo associado. Multiplicando todos os números primos, tem-se um RA unívoco, graças ao Teorema Fundamental da Aritmética (teorema 3.2.1).

É então interessante observar que as duas primeiras soluções são determinísticas, mas não unívocas, enquanto esta terceira solução é não-determinística (ou seja, probabilística), mas unívoca.

Será visto que, dependendo da quantidade de instrumentos, se esta for muito grande, a solução pode deixar de ser unívoca.

4.3.1 Requisitos Funcionais

Para esta solução, os seguintes requisitos funcionais devem ser atendidos.

Requisito 1: O dispositivo deverá armazenar e proteger uma chave privada e com ela ser capaz de criptografar.

Requisito 2: O responsável pela atualização deverá conhecer previamente a chave pública de todos os dispositivos em campo. Isso, na prática, significa que o responsável deverá possuir uma tabela relacionando cada identificador de instrumento com a sua respectiva chave criptográfica pública.

Requisito 3: O dispositivo deverá ter uma nova funcionalidade. Um comando que receba como parâmetro de entrada uma mensagem C_i cifrada com chave pública Y_i, e que retorne o seguinte valor:

Identificador do comando: CMD04
Entrada do Comando: Uma mensagem cifrada C_i.
Saída do Comando: $E_2(H_i \oplus D_1(C_i, X_i))$

Em palavras, o instrumento deve retornar o valor da operação Xor entre a mensagem decifrada e o valor do hash da versão do firmware instalado nele. Todo esse valor deve ser encriptado com uma criptografia homomórfica na multiplicação.

4.3.2 Cálculos Prévios

O responsável pela atualização deverá gerar números primos aleatórios distintos para cada dispositivo. Para facilitar, números primos aleatórios de mesmo tamanho b em bits. De acordo com a tabela 4.2.

Tabela 4.2: Primos aleatórios distintos para cada dispositivo

Identificador do Instrumento	**Chave Pública**	**Primo aleatório**
Id_1	Y_1	P_1
Id_2	Y_2	P_2
Id_3	Y_3	P_3
...	...	...
Id_N	Y_N	P_N

Um observação: Os primos aleatórios gerados são de tamanho b em bits, mas poderiam ter também tamanhos maiores ou menores. O mais importante é serem primos. Comumente, as funções que geram números aleatórios recebem como entrada o tamanho dos números em bits. Assim, se explica por que foi afirmado que, para facilitar, seriam escolhidos primos aleatórios de mesmo tamanho.

Para cada número aleatório gerado, deve-se realizar uma operação Xor com o hash H da nova versão de software. Conforme a tabela 4.3.

Tabela 4.3: Operação Xor com os números aleatórios

Identificador do Instrumento	**Chave Pública**	**Primo aleatório**	**Xor**
Id_1	Y_1	P_1	$H \oplus P_1$
Id_2	Y_2	P_2	$H \oplus P_2$
Id_3	Y_3	P_3	$H \oplus P_3$
...	...	...	...
Id_N	Y_N	P_N	$H \oplus P_N$

A última etapa da preparação é encriptar os valores da última coluna da tabela 4.3 com a chave pública dos dispositivos. Conforme tabela 4.4.

Tabela 4.4: Encriptação com a chave pública

Identificador do Instrumento	**Chave Pública**	**Xor**	**Encriptação**
Id_1	Y_1	$H \oplus P_1$	$C_1 = E_1(H \oplus P_1; Y_1)$
Id_2	Y_2	$H \oplus P_2$	$C_2 = E_1(H \oplus P_2; Y_2)$
Id_3	Y_3	$H \oplus P_3$	$C_3 = E_1(H \oplus P_3; Y_3)$
...	...	...	...
Id_N	Y_N	$H \oplus P_N$	$C_N = E_1(H \oplus P_N; Y_N)$

4.3.3 Execução

Cada valor C_i da última coluna da tabela 4.4 deverá ser enviado ao dispositivo i como entrada para o comando descrito no requisito funcional 3. A saída esperada está representada na tabela 4.5.

Tabela 4.5: Saída que se espera do dispositivo

Identificador do Instrumento	**Chave Pública**	**Entrada**	**Saída**
Id_1	Y_1	$C_1 = E_1(H \oplus P_1; Y_1)$	$E_2(H_1 \oplus D_1(C_1; X_1)) = E_2(P'_1)$
Id_2	Y_2	$C_2 = E_1(H \oplus P_2; Y_2)$	$E_2(H_2 \oplus D_1(C_2; X_2)) = E_2(P'_2)$
Id_3	Y_3	$C_3 = E_1(H \oplus P_3; Y_3)$	$E_2(H_3 \oplus D_1(C_3; X_3)) = E_2(P'_3)$
...	...	...	...
Id_N	Y_N	$C_N = E_1(H \oplus P_N; Y_N)$	$E_2(H_N \oplus D_1(C_N; X_N)) = E_2(P'_N)$

4.3.4 Prova Matemática

Na matemática, a meta é a prova absoluta, e, uma vez que se tenha demonstrado alguma coisa, ela está provada para sempre, sem espaço para mudanças.

Simon Singh, em "O Último Teorema de Fermat"

Falta agora a prova matemática de que as saídas da função serão os números primos aleatórios P_i se $H_i = H$, ou seja, se a versão do software no dispositivo i for igual à nova versão. As saídas do comando são dadas pela expressão 4.2.

$$H_i \oplus D_1(C_i; X_i) \tag{4.2}$$

Substituindo o valor de C_i pelos valores da coluna "Entrada" (tabela 4.5), obtém-se 4.3:

$$H_i \oplus D_1(E_1(H \oplus P_i; Y_i); X_i) \tag{4.3}$$

Como a função de decriptação é o inverso da função de encriptação, então a expressão 4.3 resulta em 4.4:

$$H_i \oplus H \oplus P_i \tag{4.4}$$

O trecho $H_i \oplus H$ será anulado se, e somente se, $H_i = H$, pois a função

Xor se anula sempre que as duas entradas são iguais (propriedade conhecida como *auto-inverso*: $x \star x = e, \forall x \in \mathcal{G}$). Resultando, portanto, no número primo aleatório P_i.

c.q.d.

4.3.5 Cálculos Posteriores

Ao final da atualização da versão de todos os N instrumentos, a equipe terá posse de todos os $P_i's$ criptografados, ou seja, $\{E_2(P_1'), E_2(P_2'), E_2(P_3'), \ldots, E_2(P_N')\}$. Não se pode afirmar ainda que nenhum dos $P_i's$ seja igual ao respectivo P_i precalculado. Tal igualdade só ocorrerá se realmente a versão instalada no instrumento for a nova versão de firmware. Portanto, a confirmação final ocorrerá somente se, para cada P_i' encontrado, ocorrer a igualdade $P_i' = P_i$.

4.3.6 Passo Final

Para comprovar que o conjunto numérico $\{P_1', P_2', P_3', \ldots, P_N'\}$ é igual ao produzido inicialmente $\{P_1, P_2, P_3, \ldots, P_N\}$ será utilizada criptografia homomórfica. Basta fazer a multiplicação das saídas $\{E_2(P_1'), E_2(P_2'), E_2(P_3'), \ldots, E_2(P_N')\}$. Assim:

$$E_2(P_1') \times E_2(P_2') \times E_2(P_3') \times \cdots \times E_2(P_N') = E_2(P_1' \times P_2' \times P_3' \times \cdots \times P_N') \quad (4.5)$$

Portanto, a garantia de que a atualização de todos os instrumentos foi realizada com sucesso será dada pela igualdade:

$$E_2(P'_1 \times P'_2 \times P'_3 \times \cdots \times P'_N) = E_2(P_1 \times P_2 \times P_3 \times \cdots \times P_N) \tag{4.6}$$

Pois a unicidade do resultado é garantida pelo Teorema da Fatoração Única.

O valor $E_2(P_1 \times P_2 \times P_3 \times \cdots \times P_N)$ passa a ser o RA (recibo agregador) procurado.

4.3.7 Análise da solução

A solução descrita possui um nível de complexidade maior que o nível das soluções anteriores, mas é a compensação por resolver as fragilidades destas. A solução é unívoca (Teorema Fundamental da Aritmética, teorema 3.2.1), mas tem uma grande desvantagem prática. O tamanho do produto $P = P_1 \times P_2 \times P_3 \times \cdots \times P_N$ depende da quantidade de instrumentos e do tamanho, em bits, dos primos escolhidos. Quanto maior a quantidade de instrumentos, maior fica o número P. Como os algoritmos criptográficos geralmente são baseados em exponenciação modular, caso o expoente seja muito grande, a solução deixará de ser unívoca. Será que é possível escolher tamanhos menores para os primos a ponto da solução continuar unívoca, mas sem afetar a segurança do sistema? A partir da implementação da solução, notou-se que sim.

Quando esta solução foi inicialmente pensada, não utilizava criptografia homomórfica. Os números primos saíam do instrumento completamente expostos.

Depois foi descoberta uma forma de burlar o sistema a partir dessa fragilidade. Surgiu então a necessidade de atender a dois requisitos simultaneamente: (1) ser possível multiplicar os números primos e (2) mantê-los em confidencialidade. Para atender aos dois requisitos simultaneamente, a única forma seria aplicar criptografia homomórfica para multiplicação sobre cada número primo. Assim, eles ficariam protegidos e a operação multiplicação continuaria viável.

4.3.8 Implementação da solução

Para implementar a solução, é fundamental saber a quantidade de números primos com um tamanho em bits predefinido. Para isso, será utilizada a sequência $A007053$ (ver ref. [19] SLOANE), a qual determina a quantidade de números primos menores ou iguais a uma potência de 2. As duas primeiras colunas da tabela 4.6 correspondem aos 13 primeiros valores dessa sequência (uma tabela estendida encontra-se no apêndice A.1). Os valores da terceira coluna são os números que interessam. Eles são calculados a partir dos valores da segunda coluna. Como exemplo, será analisada a linha 11. O valor na terceira coluna é 137, o qual é calculado subtraindo do valor na segunda coluna da mesma linha (valor 309) o valor na mesma coluna mas na linha anterior, linha 10, (valor 172). Dessa linha 11, é possível concluir que existem 137 números primos com 11 bits.

Tabela 4.6: Quantidade de números primos com b bits

b	Quantidade total de primos abaixo de 2^b	Quantidade de primos com b bits (q_b)
1	0	0
2	2	2
3	4	2
4	6	2
5	11	5
6	18	7
7	31	13
8	54	23
9	97	43
10	172	75
11	309	137
12	564	255
13	1028	464

Para simular um caso real, será considerado como objetivo atualizar o firmware de 50 medidores. Primeira coisa a se fazer é tomar o dobro desse número ($2 \times 50 = 100$), e procurar na tabela o primeiro valor acima disso que esteja na terceira coluna. Neste caso, será 137, valor na linha 11 da terceira coluna. Como o valor na primeira coluna corresponde também à quantidade de bits, então será necessário escolher 50 números primos de 11 bits. Fica claro então que o raciocínio consiste em escolher um subconjunto de 50 números primos de um conjunto contendo 137 números primos. A quantidade total de subconjuntos possíveis é determinada a partir do cálculo dos coeficientes binomiais, que costuma ser representado por $\binom{n}{k}$, onde n é o total de elementos disponíveis e k é a quantidade de elementos nos subconjuntos. Esse cálculo é bem conhecido e dado por $\frac{n!}{k!(n-k)!}$. Substituindo os valores do exemplo, obtém-se:

$$\frac{137!}{50!(137-50)!} = 781974501912974556006225962944131201 60 \tag{4.7}$$

A segurança da solução fica assim distribuída em todos os dispositivos. Encontrar o número primo escolhido para um único dispositivo, tem baixa probabilidade (1/137) de ser encontrado, mas, se tentar encontrar todos os 50 números primos, a probabilidade fica representada por 4.8.

$$\frac{1}{781974501912974556006225962944131201 60} \tag{4.8}$$

4.3.9 Análise da implementação da solução

> *If you are faced by a difficulty or a controversy in science, an ounce of algebra is worth a ton of verbal argument.*
>
> J. B. S. Haldane (1892-1964)

Por que se deve escolher valores na tabela com o dobro da quantidade de dispositivos a serem atualizados?

Porque a quantidade de dispositivos a serem atualizados determina a quantidade de números primos que será necessária escolher. Ou seja, determina o tamanho do subconjunto de números primos que será gerado. Quanto maior a quantidade de subconjuntos, maior será a força da solução. A quantidade de subconjuntos é dada pelo cálculo dos coeficientes binomiais, já explicado acima. Fixando um tamanho para o conjunto total de elementos, a quantidade de subconjuntos cresce à medida

que o tamanho dos subconjuntos se aproxima da metade do total. Esse fato será provado a seguir.

Será utilizada a Aproximação de Stirling (ver ref. [20] STANICA), descrita na equação 4.9:

$$x! = [1 + o(1)]\sqrt{2\pi x}\left(\frac{x}{e}\right)^x \tag{4.9}$$

Substituindo a equação 4.9 na fórmula do coeficiente binomial, obtém-se a equação 4.10.

$$\frac{n!}{k!(n-k)!} = [1 + o(1)]\frac{\sqrt{2\pi n}(\frac{n}{e})^n}{\sqrt{2\pi k}(\frac{k}{e})^k\sqrt{2\pi(n-k)}(\frac{n-k}{e})^{n-k}} \tag{4.10}$$

Simplificando a equação 4.10, resulta em 4.11.

$$\binom{n}{k} = [1 + o(1)]\sqrt{\frac{n}{2\pi k(n-k)}}\left(\frac{n}{k}\right)^k\left(\frac{n}{n-k}\right)^{n-k} \tag{4.11}$$

Aplicando o logaritmo nos dois lados da equação 4.11, obtém-se 4.12.

$$log\binom{n}{k} = log[1 + o(1)] + log(\sqrt{\frac{n}{2\pi k(n-k)}}) + log\left(\frac{n}{k}\right)^k + log\left(\frac{n}{n-k}\right)^{n-k} \tag{4.12}$$

Como $k = cn$, com $c \in [0, 1]$, então chega-se a 4.13.

$$log\binom{n}{k} = log[1+o(1)]+log\left(\sqrt{\frac{1}{2\pi c(1-c)}}\right)+log\left(\sqrt{\frac{1}{n}}\right)+log\left(\frac{n}{k}\right)^{k}+log\left(\frac{n}{n-k}\right)^{n-k} \tag{4.13}$$

Os dois últimos termos da equação 4.13 são mais significativos que os outros, portanto, tem-se 4.14:

$$log\binom{n}{k} = [1+o(1)]\left[log\left(\frac{n}{k}\right)^{k}+log\left(\frac{n}{n-k}\right)^{n-k}\right] \tag{4.14}$$

Reescrevendo a equação 4.14, obtém-se 4.15:

$$log\binom{n}{k} = [1+o(1)]\left[\frac{k}{n}.log\left(\frac{n}{k}\right)+\frac{n-k}{n}.log\left(\frac{n}{n-k}\right)\right]n \tag{4.15}$$

Finalmente chega-se a 4.16:

$$log\binom{n}{k} = [1+o(1)]h\left(\frac{k}{n}\right)n \tag{4.16}$$

Onde $h(p) = -p.log(p) - (1-p).log(1-p)$ é conhecida como Função de Entropia Binária, cuja derivada é $h'(p) = log\frac{(1-p)}{p}$. O valor máximo para $h(p)$ será quando $h'(p) = 0$. Isso ocorre quando $p = \frac{1}{2}$. No nosso caso, como $p = k/n$, então $k/n = 1/2$, daí tem-se: $n = 2k$.

Por que é possível escolher tamanhos pequenos para os números primos escolhidos?

Porque a quantidade de subconjuntos cresce exponencialmente quando $k \approx n/2$. Basta ver que:

$$\binom{n}{n/2} = \frac{n!}{(n/2)!.(n/2)!} \tag{4.17}$$

Substituindo pela Aproximação de Stirling, equação 4.9, obtém-se 4.18:

$$\binom{n}{n/2} = [1 + o(1)].\left(\frac{2}{\sqrt{2\pi n}}\right).2^n \in \Theta(2^n.n^{1/2}) \tag{4.18}$$

Devido a este fato, é que foi possível afirmar que, para cardinalidades na ordem de centenas de dispositivos, o problema pode ser resolvido ainda de forma unívoca.

4.3.10 Possíveis ataques

> *Only a cryptanalyst can judge the security of a crypto system.*
>
> Friedrich L. Bauer (1924-2016) in "Decrypted Secrets"

A parte mais crítica da segurança da solução está na criptografia homomórfica utilizada. No caso do RSA puro, como é uma criptografia determinística (ou seja, uma mesma mensagem em claro sempre vai produzir a mesma mensagem cifrada), e os números primos possuem tamanhos relativamente baixos, um ataque de força bruta rapidamente poderia encontrar os respectivos números primos.

Como o algoritmo Elgamal é probabilístico (ou seja, uma mesma mensagem em claro cifrada várias vezes não produz o mesmo texto cifrado), então utilizá-lo torna a solução mais segura, pois um ataque de força bruta é inviável.

4.3.11 Agregação de Recibos Agregadores

Será tratado agora o cenário no qual haja a necessidade de atualizar o software de milhares de medidores, por exemplo, medidores de energia elétrica numa cidade. A tarefa pode ser dividida por regiões da cidade, para cada região, calcula-se o RA correspondente. A forma como o RA foi criado, permite a agregação de vários recibos agregadores devido à propriedade associativa da multiplicação.

Supondo-se que a cidade foi dividida em R regiões. Para cada região, foi calculado o seu respectivo RA_i, com $1 \leqslant i \leqslant R$. Cada região contendo k_i medidores. Assim, tem-se:

$$\begin{aligned}
RA_1 &= (P_{1,1} \times P_{1,2} \times P_{1,3} \times \cdots \times P_{1,k_1}) \mod Q \\
RA_2 &= (P_{2,1} \times P_{2,2} \times P_{2,3} \times \cdots \times P_{2,k_2}) \mod Q \\
RA_3 &= (P_{3,1} \times P_{3,2} \times P_{3,3} \times \cdots \times P_{3,k_3}) \mod Q \\
&\vdots \\
RA_R &= (P_{R,1} \times P_{R,2} \times P_{R,3} \times \cdots \times P_{R,k_R}) \mod Q
\end{aligned}$$

Com $k_1 + k_2 + k_3 + \cdots + k_R = N$, onde N é o número total de medidores. O RA final para toda a cidade será dado por:

$$RA_{final} = RA_1 \times RA_2 \times RA_3 \times \cdots \times RA_R$$

Esse resultado só será correto se todos os $RA's$ estiverem sob o mesmo módulo Q. Outra observação a ser feita é que, nesta situação, a solução não será unívoca.

4.3.12 Proposta de maior segurança para uma implementação em caso real

Numa situação real, até mesmo o RA pode ser mantido sigiloso e a verificação pode ser feita apenas respondendo a um número limitado de consultas, por exemplo, 3 consultas. Isso permitiria aumentar em muito a segurança da solução e permitiria diminuir ainda mais o tamanho dos números primos utilizados. De fato, o tamanho pode ser reduzido em, no máximo, 2 bits (ver item 4.3.13). No exemplo analisado antes, da atualização dos 50 dispositivos, é possível reduzir o tamanho de 11 para 10 bits, mas não para 9, pois com 9 bits, só estão disponíveis 43 números primos, um número abaixo dos 50 necessários. Assim, em vez da linha 11, será utilizada a linha 10 da tabela 7. O que resultaria em:

$$\frac{75!}{50!(75-50)!} = 52588547141148893628 \tag{4.19}$$

O que resulta numa probabilidade:

$$\frac{1}{52588547141148893628} \tag{4.20}$$

O resultado bem menor que o anterior 4.8, mas o sistema continua seguro, pois um acerto em 3 tentativas continua sendo impraticável.

4.3.13 Conjecturas Matemáticas

A definição de um bom problema de matemática reside na matemática que ele produz, não no problema em si.

Andrew Wiles, que provou o
Último Teorema de Fermat

Na seção anterior, 4.3.12, afirmou-se que seria possível reduzir o tamanho dos números primos aleatórios no máximo em até 2 bits. Essa afirmação na verdade não tem prova ainda. É apenas uma conjectura que foi extraída da observação da tabela no *apêndice B*. O entendimento dessa tabela é muito importante. Recomenda-se a leitura do referido apêndice antes de continuar com a leitura deste tópico.

Escolhendo-se três valores consecutivos de q_b na tabela do *apêndice A*, será sempre possível escolher um valor para N tal que atenda à situação descrita na figura 4.1.

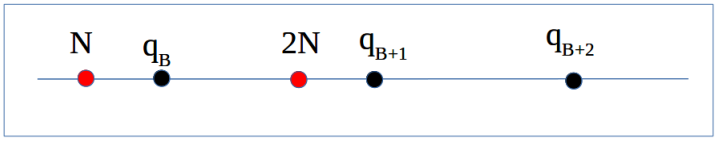

Figura 4.1: Situação comum de acontecer conforme tabela no *Apêndice B*

Os próximos dois casos não são triviais, por isso, são importantes para o estabelecimento de conjecturas.

Conjectura 1: Escolhendo-se três valores consecutivos de q_b na tabela do *apêndice A*, existirá ao menos um valor para N tal que atenda à situação descrita

na figura 4.2.

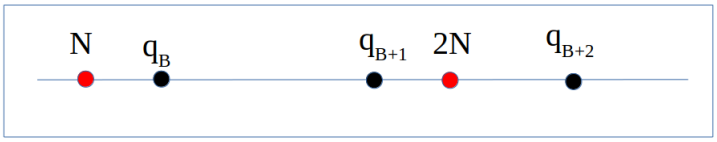

Figura 4.2: Situação que ocorre para alguns casos conforme tabela no *Apêndice B*

Conjectura 2: Escolhendo-se três valores consecutivos de q_b na tabela do *apêndice A*, é impossível encontrar um valor para N tal que atenda à situação descrita na figura 4.3.

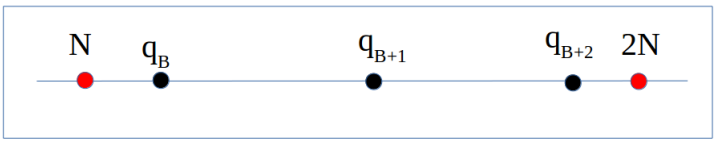

Figura 4.3: Situação que não ocorre conforme tabela no *Apêndice B*

As duas conjecturas provavelmente são desafiantes e podem vir a se tornar problemas abertos em matemática, em outras palavras, são dignas de atenção. Foi desenvolvido um programa para testar a veracidade das conjecturas até o valor $N = 10^{10}$ (dez bilhões) e elas suportaram esse teste inicial. Mesmo assim, as conjecturas devem passar por um teste de força bruta mais rigoroso para saber se elas continuam valendo. Se se encontrar algum valor que invalide alguma delas, a conjectura se mostrará falsa e deve ser descartada. No entanto, se após um teste de força bruta, as conjecturas se mantiverem valendo, então será necessária uma pesquisa mais aprofundada do assunto. Pode vir a se tornar o começo para um problema a ser tratado em um doutorado em matemática.

5 CONCLUSÃO

5.1 Resumo do Trabalho

Neste trabalho, procurou-se resolver um problema real relatado ao Inmetro por fabricantes de instrumentos de medição. O problema consiste em se obter uma forma de garantia de atualização massiva do software de uma quantidade qualquer de medidores inteligentes que estão sendo utilizados em campo. Este problema não trata da atualização de software de um ou poucos instrumentos. O cenário abarcado pelo problema ocorre quando há a necessidade de atualização de uma grande quantidade de instrumentos devido, por exemplo, a uma detecção de falha grave. Atualmente a atualização é feita por equipes de técnicos e a única garantia são relatórios dos técnicos, ou seja, a confiança está na alegação deles. Com este trabalho, foi possível obter uma garantia efetiva que o trabalho foi realizado. Além disso, procurou-se atender a um requisito a mais que é a possibilidade de se obter a garantia através de um único recibo, o chamado Recibo Agregador (RA).

Foram apresentadas três soluções. A primeira solução envolve criptografia simétrica. É uma solução leve de ser executada em instrumentos de medição menos robustos, mas tem um problema de segurança no controle das chaves criptográficas.

A segunda solução envolve assinatura digital. É a solução mais fácil de ser implementada, pois todos os medidores inteligentes já estão saindo de fábrica com par de chaves criptográficas pública e privada, mas esta solução não permite um RA.

A terceira solução envolve criptografia homomórfica. É uma solução matematicamente e computacionalmente mais complexa de ser implementada, mas também mais efetiva e segura. Nesta solução, são gerados números primos aleatórios (um primo para cada instrumento a ser atualizado). É necessário que, em cada instrumento, execute-se um comando para poder extrair o seu respectivo número primo. Cada número primo extraído será o comprovante individual do respectivo instrumento. O produto de todos os primos define um único número inteiro. Esse fato é uma consequência do Teorema Fundamental da Aritmética, teorema 3.2.1. Esse número resultante do produto de todos os primos será considerado o RA da solução. Após algumas análises da segurança da solução, verificou-se que o número primo não pode sair do instrumento sem uma proteção. É necessário haver a confidencialidade do número primo, ou seja, o número primo deve sair do instrumento criptografado. No entanto, para atender aos dois requisitos: os números devem sair criptografados do instrumento e os números devem ser multiplicados; a única forma é utilizar a criptografia homomórfica.

5.2 Contribuições

As principais contribuições deste trabalho são:

1) Resolver um problema real com uma solução que poderá fazer parte de futuros regulamentos do Inmetro.

2) Mostrar que a solução é factível e segura.

3) Utilizar criptografia homomórfica, um dos assuntos que estão em evidência atualmente na Criptologia. A solução também utiliza o Teorema Fundamental da Aritmética.

4) Extrair duas conjecturas interessantes a cerca dos números primos.

5.3 Trabalhos Futuros

O que caracteriza um estudo sério é a continuidade que se lhe dá.

Allan Kardec em "O Livro dos Espíritos"

Neste trabalho, tratou-se somente o contexto de instrumentos de medição regulados pelo Inmetro. Mais especificamente, da necessidade de atualizar o software de uma família inteira de instrumentos de medição de um mesmo modelo, ou seja, todos os instrumentos são idênticos. A quantidade de instrumentos vendidos de um mesmo modelo depende do tipo de instrumento. Alguns modelos de instrumentos, como de medidores de energia elétrica que atuam na fronteira entre duas distribuidoras, formam apenas algumas centenas de unidades vendidas no país inteiro. Outros modelos, como de medidores de energia elétrica de residências, podem chegar a muitos milhares. Mas, caso se deseje ampliar o contexto para IoT, alguns modelos podem chegar a milhões de réplicas vendidas. Então, como trabalho futuro, poderia ser estudado uma outra solução mais eficiente para tratar de quantidades tão grandes.

As duas conjecturas propostas são dignas de atenção e merecem também uma análise mais cuidadosa. É natural começar procurando alguma relação com o Teorema dos Números Primos, que trata da distribuição dos números primos para valores abaixo de n. Seja $\Pi(n)$ a função aritmética que retorna a quantidade de números primos abaixo de n. Esse famoso teorema estabelece que:

$$\Pi(n) \approx \frac{n}{\log n}$$

Portanto, essa informação já é um bom começo para um novo trabalho!

REFERÊNCIAS

[1] ANEEL. **Resolução Normativa**. Brasília, DF, Brasil: Agência Nacional de Energia Elétrica, 2012. parágrafo 7º. (502).

[2] BAUER, F. L. **Decrypted Secrets**. Alemanha: Springer, 2000.

[3] BOCCARDO, D. R. et al. Software Evaluation of Smart Meters within a Legal Metrology Perspective: a brazilian case. **Innovative Smart Grid Technologies Conference Europe (ISGT Europe)**, [S.l.], Out 2010.

[4] CHOI, B. C. et al. Secure Firmware Validation and Update for Consumer Devices in Home Networking. **IEEE Transactions on Consumer Electronics**, [S.l.], v.62, n.1, p.39 44, Fev 2016.

[5] DIFFIE, W.; HELLMAN, M. New Directions in Cryptography. **IEEE Transactions On Information Theory**, [S.l.], v.IT-22, n.6, November 1976.

[6] HONG, S. G.; KIM, N. S.; HEO, T. A Smartphone Connected Software Updating Framework for IoT Devices. **IEEE International Symposium on Consumer Electronics (ISCE)**, [S.l.], Jun 2015.

[7] INMETRO. **VIM - Vocabulário Internacional de Metrologia**. Duque de Caxias, RJ, Brasil: Instituto Nacional de Metrologia, Qualidade e Tecnologia, 2012. Portaria. (232).

[8] INMETRO. **VIML - Vocabulário Internacional de Metrologia Legal**. Duque de Caxias, RJ, Brasil: Instituto Nacional de Metrologia, Qualidade e Tecnologia, 2016. Portaria. (150).

[9] KATZIR, L.; SCHWARTZMAN, I. Secure Firmware Updates for Smart Grid Devices. **Innovative Smart Grid Technologies Conference Europe (ISGT Europe)**, [S.l.], Dez 2011.

[10] LIMA, E. L. **Análise Real**. Brasil: Impa, 2001. Coleção Matemática Universitária, Volume 1.

[11] LIU, J.; TONG, W. A Framework for Dynamic Updating of Service Pack in the Internet of Things. **IEEE International Conferences on Internet of Things, and Cyber, Physical and Social Computing**, [S.l.], p.33 42, Out 2011.

[12] MATTOS, D. M. F.; DUARTE, O. C. M. B.; PUJOLLE, G. Reverse Update: a consistent policy update scheme for software-defined networking. **IEEE Communications Letters**, [S.l.], v.20, n.5, p.886 889, Mai 2016.

[13] MILIES, C. P.; COELHO, S. P. **Números**: uma introdução à matemática. Brasil: Edusp, 2001.

[14] NIST. **Digital Signature Standard (DSS)**. Gaithersburg, MD, USA: National Institute of Standards and Technology, 2013. FIPS-Federal Information Processing Standards Publication. (FIPS PUB 186-4).

[15] PRADO, C. B. do et al. Software Analysis and Protection for Smart Metering. **NCSL International Measure**, [S.l.], v.9, n.3, 2014.

[16] SHANNON, C. E. A Mathematical Theory of Communication. **Bell System Technical Journal**, [S.l.], v.27, n.3, 1948.

[17] SHANNON, C. E. Communication Theory of Secrecy Systems. **Bell System Technical Journal**, [S.l.], v.28, n.4, 1949.

[18] SINGH, S. **O Livro dos Códigos**. Brasil: Record, 2001.

[19] SLOANE, N. J. A. **The On-Line Encyclopedia of Integer Sequences**. http://oeis.org/A007053.

[20] STANICA, P. Good Lower and Upper Bounds on Binomial Coefficients. **IEEE Communications Letters**, [S.l.], v.2, n.3, Mai 2001.

[21] WELMEC. **Measuring Instruments Directive 2004/22/EC**. Slovenia: National Institute of Standards and Technology, 2009. Software Guide. (WELMEC 7.2).

[22] YAN, S. Y. **Number Theory for Computing**. UK: Springer, 1998.

APÊNDICE A

Os matemáticos odeiam fazer uma declaração falsa.

Simon Singh, em "O Último Teorema de Fermat"

Tabela A.1: Sequência $A007053$ [19] ampliada até $b = 40$.

b	Quantidade total de primos abaixo de 2^b	Quantidade de primos com b bits (q_b)
1	0	0
2	2	2
3	4	2
4	6	2
5	11	5
6	18	7
7	31	13
8	54	23
9	97	43
10	172	75
11	309	137
12	564	255
13	1028	464
14	1900	872
15	3512	1612
16	6542	3030
17	12251	5709
18	23000	10749
19	43390	20390
20	82025	38635
21	155611	73586
22	295947	140336

Continua na próxima página.

Tabela A.1 continuação da página anterior.

b	Quantidade total de primos abaixo de 2^b	Quantidade de primos com b bits (q_b)
23	564163	268216
24	1077871	513708
25	2063689	985818
26	3957809	1894120
27	7603553	3645744
28	14630843	7027290
29	28192750	13561907
30	54400028	26207278
31	105097565	50697537
32	203280221	98182656
33	393615806	190335585
34	762939111	369323305
35	1480206279	717267168
36	2874398515	1394192236
37	5586502348	2712103833
38	10866266172	5279763824
39	21151907950	10285641778
40	41203088796	20051180846

APÊNDICE B

As with everything, so with a mathematical theory: beauty can be perceived, but not explained.

Arthur Cayley (1821-1895)

Será dada uma explicação para o entendimento da tabela B.1. A primeira coluna contém a quantidade de instrumentos a serem atualizados. Essa quantidade será representada por N. A tabela apresenta duas propostas para quem implementará a terceira solução, 4.3. Uma das propostas segue estritamente o que foi proposto na descrição da solução. A outra proposta sugere uma forma de diminuir o tamanho em bits dos primos gerados. A primeira proposta sempre vai funcionar, mas a segunda proposta nem sempre. Com a tabela, é possível comparar as duas e saber quando poderá utilizar uma ou a outra. As colunas $b1$ e $b2$ contêm os tamanhos em bits dos números primos aleatórios a serem gerados. As colunas q_{b1} e q_{b2} contêm as quantidades de números primos de tamanhos $b1$ e $b2$, respectivamente (conforme tabela no *apêndice A*). As colunas $SC1$ e $SC2$ representam as quantidades de subconjuntos gerados mantendo a mesma quantidade de elementos no subconjunto, ou seja, fixando N e variando a quantidade total de elementos disponíveis, ou seja, para q_{b1} e q_{b2}. Assim, tem-se:

$$SC1 = \binom{q_{b1}}{N} = \frac{q_{b1}!}{N!(q_{b1}-N)!}$$
$$SC2 = \binom{q_{b2}}{N} = \frac{q_{b2}!}{N!(q_{b2}-N)!}$$

Para exemplificar como a tabela deve ser analisada, será escolhida a linha para $N = 44$, a tabela oferece duas opções para os tamanhos dos primos aleatórios a serem

gerados $b_1 = 10$ ou $b_2 = 11$. Caso seja escolhida a primeira opção, tem-se $q_{b1} = 75$ primos possíveis. Caso seja escolhida a segunda opção, tem-se $q_{b2} = 137$ primos possíveis. O primeiro caso produzirá mais de 10^{21} subconjuntos. O segundo caso produzirá mais de 10^{36} subconjuntos. Este exemplo mostra que, mesmo escolhendo a primeira opção, o procedimento continuará seguro.

A mesma análise feita para $N = 43$ já é bem diferente. Aqui não resta dúvida que se deve utilizar os valores propostos pela solução padrão, ou seja, $b_2 = 11$ e $q_{b2} = 137$, o que resulta numa quantidade de subconjuntos maior que 10^{35}. O valor mais baixo produzirá apenas 1 subconjunto.

O objetivo desta tabela é mostrar que é possível reduzir, em alguns casos, o tamanho dos números primos. Essa redução é de, no máximo, 2 bits para cada primo. Nunca poderá ser três bits (ver 4.3.13). Observando a coluna $SC2$, os valores são sempre crescentes e de forma exponencial, conforme explicado no item *4.3.8 Análise da implementação da solução*. Esse é o resultado padrão seguindo estritamente a solução como ela foi proposta. Observando a coluna $SC1$, é possível observar que os valores são altos em alguns casos, mas baixos em outros. Por isso, existe a possibilidade de se diminuir o tamanho dos primos. Para decidir, basta comparar os valores de $SC1$ e $SC2$.

Tabela B.1: Quantidade de subconjuntos (SC) seguindo estritamente a solução proposta (coluna SC2) e com diminuição de até 2 bits (coluna SC1)

N	$b1$	q_{b1}	$SC1$	$b2$	q_{b2}	$SC2$
3	5	5	10	6	7	35
4	5	5	5	7	13	715
5	5	5	1	7	13	1287
6	6	7	7	7	13	1716
7	6	7	1	8	23	245157
8	7	13	1287	8	23	490314
9	7	13	715	8	23	817190
10	7	13	286	8	23	1144066
11	7	13	78	8	23	1352078
12	7	13	13	9	43	15338678264
13	7	13	1	9	43	36576848168
14	8	23	817190	9	43	78378960360
15	8	23	490314	9	43	151532656696
16	8	23	245157	9	43	265182149218
17	8	23	100947	9	43	421171648758
18	8	23	33649	9	43	608359048206
19	8	23	8855	9	43	800472431850
20	8	23	1771	9	43	960566918220
21	8	23	253	9	43	1052049481860
22	8	23	23	10	75	5,1632228474619E+18
23	8	23	1	10	75	1,18978613441513E+019
24	9	43	800472431850	10	75	2,57786995789946E+19
25	9	43	608359048206	10	75	5,25885471411489E+19
26	9	43	421171648758	10	75	1,01131821425286E+20
27	9	43	265182149218	10	75	1,83535527771816E+20
28	9	43	151532656696	10	75	3,14632333323113E+20
29	9	43	78378960360	10	75	5,09921367799528E+20
30	9	43	36576848168	10	75	7,81879430625943E+20
31	9	43	15338678264	10	75	1,13498627026347E+21
32	9	43	5752004349	10	75	1,56060612161227E+21
33	9	43	1917334783	10	75	2,03351706755538E+21
34	9	43	563921995	10	75	2,51199167168605E+21
35	9	43	145008513	10	75	2,94261881540366E+21

Continua na próxima página.

Tabela B.1 continuação da página anterior.

N	$b1$	q_{b1}	$SC1$	$b2$	q_{b2}	$SC2$
36	9	43	32224114	10	75	3,26957646155962E+21
37	9	43	6096454	10	75	3,44631032434663E+21
38	9	43	962598	11	137	1,02698462506701E+34
39	9	43	123410	11	137	2,60696097132396E+34
40	9	43	12341	11	137	6,38705437974369E+34
41	9	43	903	11	137	1,51108359715887E+35
42	9	43	43	11	137	3,45390536493457E+35
43	9	43	1	11	137	7,63072115508799E+35
44	10	75	1,13498627026347E+21	11	137	1,63019951949607E+36
45	10	75	7,81879430625943E+20	11	137	3,36907900695855E+36
46	10	75	5,09921367799528E+20	11	137	6,73815801391709E+36
47	10	75	3,14632333323113E+20	11	137	1,30462208354565E+37
48	10	75	1,83535527771816E+20	11	137	2,44616640664809E+37
49	10	75	1,01131821425286E+20	11	137	4,44303694268736E+37
50	10	75	5,25885471411489E+19	11	137	7,81974501912974E+37
51	10	75	2,57786995789946E+19	11	137	1,33395650326331E+38
52	10	75	1,18978613441513E+19	11	137	2,20615883232009E+38
53	10	75	5,1632228474619E+18	11	137	3,53817925938128E+38
54	10	75	2,10353523415114E+18	11	137	5,50383440348198E+38
55	10	75	8,03167998494073E+17	11	137	8,30578646343645E+38
56	10	75	2,86845713747883E+17	11	137	1,21620444643177E+39
57	10	75	9,56152379159611E+16	11	137	1,72829052913988E+39
58	10	75	2,96736945256431E+16	11	137	2,38384900571018E+39
59	10	75	8550047575185300	11	137	3,19193341442549E+39
60	10	75	2280012686716080	11	137	4,14951343875314E+39
61	10	75	560658857389200	11	137	5,23791040629494E+39
62	10	75	126600387152400	11	137	6,4206643690067E+39
63	10	75	26123889412400	11	137	7,64364805834132E+39
64	10	75	4898229264825	11	137	8,83796806745714E+39
65	10	75	828931106355	11	137	9,92571798345187E+39
66	10	75	125595622175	11	137	1,08280559819475E+40
67	10	75	16871053725	11	137	1,14745070853473E+40
68	10	75	1984829850	11	137	1,18119925878576E+40
69	10	75	201359550	12	255	2,55371899750898E+63
70	10	75	17259390	12	255	6,78559619338101E+63
71	10	75	1215450	12	255	1,76807788137393E+64

Continua na próxima página.

Tabela B.1 continuação da página anterior.

N	$b1$	q_{b1}	$SC1$	$b2$	q_{b2}	$SC2$
72	10	75	67525	12	255	4,51842125240004E+64
73	10	75	2775	12	255	1,132700122177E+65
74	10	75	75	12	255	2,7858300302191E+65
75	10	75	1	12	255	6,72313647292875E+65
76	11	137	5,23791040629494E+39	12	255	1,59232179621997E+66
77	11	137	4,14951343875314E+39	12	255	3,70163118861525E+66
78	11	137	3,19193341442549E+39	12	255	8,44731219966044E+66
79	11	137	2,38384900571018E+39	12	255	1,89262564473405E+67
80	11	137	1,72829052913988E+39	12	255	4,16377641841491E+67
81	11	137	1,21620444643177E+39	12	255	8,99581324966183E+67
82	11	137	8,30578646343645E+38	12	255	1,90886768956239E+68
83	11	137	5,50383440348198E+38	12	255	3,97872422041317E+68
84	11	137	3,53817925938128E+38	12	255	8,14691149894126E+68
85	11	137	2,20615883232009E+38	12	255	1,63896690155171E+69
86	11	137	1,33395650326331E+38	12	255	3,23981829376501E+69
87	11	137	7,81974501912974E+37	12	255	6,29344013386537E+69
88	11	137	4,44303694268736E+37	12	255	1,20147493464703E+70
89	11	137	2,44616640664809E+37	12	255	2,25445296725902E+70
90	11	137	1,30462208354565E+37	12	255	4,1582132507222E+70
91	11	137	6,73815801391709E+36	12	255	7,53961743262817E+70
92	11	137	3,36907900695855E+36	12	255	1,34401875972937E+71
93	11	137	1,63019951949607E+36	12	255	2,35564578318158E+71
94	11	137	7,63072115508799E+35	12	255	4,05972996675974E+71
95	11	137	3,45390536493457E+35	12	255	6,88017394366652E+71
96	11	137	1,51108359715887E+35	12	255	1,14669565727775E+72
97	11	137	6,38705437974369E+34	12	255	1,87963514955838E+72
98	11	137	2,60696097132396E+34	12	255	3,03043217990024E+72
99	11	137	1,02698462506701E+34	12	255	4,80583689135695E+72
100	11	137	3,90254157525465E+33	12	255	7,49710555051684E+72
101	11	137	1,42964394341012E+33	12	255	1,15054590131694E+73
102	11	137	5,04580215321218E+32	12	255	1,73709871375303E+73
103	11	137	1,71459296468375E+32	12	255	2,58035051654576E+73
104	11	137	5,60540007685073E+31	12	255	3,77128152418227E+73
105	11	137	1,76169716701023E+31	12	255	5,42346200144307E+73
106	11	137	5,31833107021956E+30	12	255	7,67471037940057E+73
107	11	137	1,54082488950286E+30	12	255	1,06872135189784E+74

Continua na próxima página.

Tabela B.1 continuação da página anterior.

N	$b1$	q_{b1}	$SC1$	$b2$	q_{b2}	$SC2$
108	11	137	4,28006913750795E+29	12	255	1,46454407482296E+74
109	11	137	1,13873399071312E+29	12	255	1,97511907338509E+74
110	11	137	2,89859561272432E+28	12	255	2,62152167922022E+74
111	11	137	7,05063797689699E+27	12	255	3,42451030168407E+74
112	11	137	1,6367552446368E+27	12	255	4,40294181645094E+74
113	11	137	3,62113992176283E+26	12	255	5,57186442258836E+74
114	11	137	7,62345246686912E+25	12	255	6,94039252638199E+74
115	11	137	1,52469049337382E+25	12	255	8,50952474973792E+74
116	11	137	2,89165438398484E+24	12	255	1,02701160772699E+75
117	11	137	5,19014889433176E+23	12	255	1,22012490148762E+75
118	11	137	8,7968625327657E+22	12	255	1,42692573224823E+75
119	11	137	1,40454107666007E+22	12	255	1,64276323796646E+75
120	11	137	2,1068116149901E+21	12	255	1,86179833636198E+75
121	11	137	2,9599832607299E+20	12	255	2,077213019908E+75
122	11	137	3,88194525997364E+19	12	255	2,28152905465305E+75
123	11	137	4,7340795853337E+18	12	255	2,46701922169801E+75
124	11	137	5,34492856408644E+17	12	255	2,62618175213014E+75
125	11	137	5,5587257066499E+16	12	255	2,75223847623238E+75
126	11	137	5294024482523712	12	255	2,83961112627151E+75
127	11	137	458537553604416	12	255	2,8843294117246E+75
128	11	137	35823246375345	13	464	2,03494012494435E+117
129	11	137	2499296258745	13	464	5,30030916264575E+117
130	11	137	153802846692	13	464	1,36584889960487E+118
131	11	137	8218472724	13	464	3,48239337761851E+118
132	11	137	373566942	13	464	8,78512874808306E+118
133	11	137	14043870	13	464	2,19297950704029E+119
134	11	137	419220	13	464	5,41698669276369E+119
135	11	137	9316	13	464	1,32415230267557E+120
136	11	137	137	13	464	3,20328020279604E+120
137	11	137	1	13	464	7,66916720085477E+120

APÊNDICE C

Tabela C.1: Lista de Definições e Teoremas no Item 3.2.

Item	Texto
Teorema 3.2.1	Teorema Fundamental da Aritmética
Definição 3.2.2	Operação binária
Definição 3.2.3	Grupo
Definição 3.2.4	Grupo Comutativo
Definição 3.2.5	Subgrupo
Teorema 3.2.6	$\langle a \rangle < \mathcal{G}$.
Definição 3.2.7	Subgrupo gerado por a
Definição 3.2.8	Ordem de um grupo
Definição 3.2.9	Grupo Finito
Definição 3.2.10	Ordem de um elemento
Teorema 3.2.11	Equivalentes: (i) $\lvert\langle a\rangle\rvert$ é finita; (ii) $\exists t \geqslant 1$ tal que $a^t = e$.
Definição 3.2.12	Geradores e grupo cíclico
Definição 3.2.13	Classes de equivalência
Definição 3.2.14	Conjunto de todas as classes de equivalência
Definição 3.2.15	Função aritmética
Definição 3.2.16	Função $\tau(n)$
Definição 3.2.17	Função $\sigma(n)$
Definição 3.2.18	Função aritmética multiplicativa
Teorema 3.2.19	As funções $\tau(n)$ e $\sigma(n)$ são multiplicativas.
Definição 3.2.20	Função aritmética completamente multiplicativa
Teorema 3.2.21	As funções tau e sigma não são compl. multiplicativas.
Definição 3.2.22	Função totiente
Teorema 3.2.23	A função $\phi(p^k)$ é dada por $p^k - p^{k-1}$.
Teorema 3.2.24	A função $\phi(n)$ é multiplicativa.
Teorema 3.2.25	A função $\phi(n)$ não é compl. multiplicativa.
Definição 3.2.26	Semiprimo
Teorema 3.2.27	Teorema de Fermat
Teorema 3.2.28	Teorema de Euler

APÊNDICE D

Tabela D.1: Lista de Definições no Item 3.3.

Item	Texto
Definição 3.3.1	Criptologia
Definição 3.3.2	Fluxo normal de uma comunicação
Definição 3.3.3	Cifra
Definição 3.3.4	Código
Definição 3.3.5	Mensagem original (ou mensagem em claro)
Definição 3.3.6	Cifrar
Definição 3.3.7	Mensagem cifrada
Definição 3.3.8	Decifrar
Definição 3.3.9	Mensagem decifrada
Definição 3.3.10	Codificar
Definição 3.3.11	Mensagem codificada
Definição 3.3.12	Decodificar
Definição 3.3.13	Mensagem decodificada
Definição 3.3.14	Quebrar uma cifra (ou um código)
Definição 3.3.15	Criptografia
Definição 3.3.16	Criptoanálise
Definição 3.3.17	Esteganografia
Definição 3.3.18	Estaganálise
Definição 3.3.19	Confidencialidade
Definição 3.3.20	Integridade
Definição 3.3.21	Autenticação
Definição 3.3.22	Irretratabilidade (ou não-repúdio)
Definição 3.3.23	Princípio de Kerckhoff
Definição 3.3.24	Ataque
Definição 3.3.25	Interceptação
Definição 3.3.26	Modificação
Definição 3.3.27	Fabricação
Definição 3.3.28	Interrupção
Definição 3.3.29	Princípio da casa dos pombos

APÊNDICE E

Código do programa escrito em linguagem Java para verificar as conjecturas até o valor $N = 10285641779 > 10^{10}$.

```
1  package mestrado;
2
3  import java.math.BigInteger;
4  import java.util.ArrayList;
5  import java.util.List;
6
7  public class VerificaConjecturas {
8      static private List<BigInteger> lista = new ArrayList<
           BigInteger>();
9
10     public static void main(String[] args) {
11         startList();
12         BigInteger n = new BigInteger("3");
13         boolean erro = false;
14         for(Integer bits = 5; bits <= 39 && !erro; bits++) {
15             boolean valida = false;
16             BigInteger q1 = lista.get(bits);
17             BigInteger q2 = lista.get(bits+1);
18             BigInteger q3 = lista.get(bits+2);
19             n = q2.divide(new BigInteger("2"));
20             while(n.compareTo(q1) <= 0) {
21                 BigInteger dn = n.multiply(new BigInteger("2")
                       );
22                 if(dn.compareTo(q3) > 0) {
23                     System.out.println("Segunda conjectura
                           falhou para N = " + n);
24                     erro = true;
25                     break;
26                 } else if(valida == false && dn.compareTo(q2)
                       > 0) {
27                     valida = true;
28                 }
29                 n = n.add(new BigInteger("1"));
```

```
30              }
31              System.out.print("Testado para n = " + n);
32              System.out.print(" (Primeira conjectura = " +
                    valida + ")");
33              System.out.println(" (Segunda conjectura = " + !
                    erro + ")");
34          }
35      }
36
37      public static void startList() {
38          lista.add(new BigInteger("0")); // 0 bits
39          lista.add(new BigInteger("0")); // 1 bit
40          lista.add(new BigInteger("2")); // 2 bits
41          lista.add(new BigInteger("2")); // 3 bits
42          lista.add(new BigInteger("2")); // 4 bits
43          lista.add(new BigInteger("5")); // 5 bits
44          lista.add(new BigInteger("7")); // 6 bits
45          lista.add(new BigInteger("13")); // 7 bits
46          lista.add(new BigInteger("23")); // 8 bits
47          lista.add(new BigInteger("43")); // 9 bits
48          lista.add(new BigInteger("75")); // 10 bits
49          lista.add(new BigInteger("137")); // 11 bits
50          lista.add(new BigInteger("255")); // 12 bits
51          lista.add(new BigInteger("464")); // 13 bits
52          lista.add(new BigInteger("872")); // 14 bits
53          lista.add(new BigInteger("1612")); // 15 bits
54          lista.add(new BigInteger("3030")); // 16 bits
55          lista.add(new BigInteger("5709")); // 17 bits
56          lista.add(new BigInteger("10749")); // 18 bits
57          lista.add(new BigInteger("20390")); // 19 bits
58          lista.add(new BigInteger("38635")); // 20 bits
59          lista.add(new BigInteger("73586")); // 21 bits
60          lista.add(new BigInteger("140336")); // 22 bits
61          lista.add(new BigInteger("268216")); // 23 bits
62          lista.add(new BigInteger("513708")); // 24 bits
63          lista.add(new BigInteger("985818")); // 25 bits
64          lista.add(new BigInteger("1894120")); // 26 bits
65          lista.add(new BigInteger("3645744")); // 27 bits
66          lista.add(new BigInteger("7027290")); // 28 bits
67          lista.add(new BigInteger("13561907")); // 29 bits
68          lista.add(new BigInteger("26207278")); // 30 bits
69          lista.add(new BigInteger("50697537")); // 31 bits
70          lista.add(new BigInteger("98182656")); // 32 bits
```

```
71          lista.add(new BigInteger("190335585")); // 33 bits
72          lista.add(new BigInteger("369323305")); // 34 bits
73          lista.add(new BigInteger("717267168")); // 35 bits
74          lista.add(new BigInteger("1394192236")); // 36 bits
75          lista.add(new BigInteger("2712103833")); // 37 bits
76          lista.add(new BigInteger("5279763824")); // 38 bits
77          lista.add(new BigInteger("10285641778")); // 39 bits
78          lista.add(new BigInteger("20051180846")); // 40 bits
79          lista.add(new BigInteger("39113482640")); // 41 bits
80      }
81  }
```