UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

ANTONIO EMERSON BARROS TOMAZ

RESGATE DE AUTORIA EM ESQUEMAS DE ASSINATURA EM ANEL

FORTALEZA
2014

ANTONIO EMERSON BARROS TOMAZ

RESGATE DE AUTORIA EM ESQUEMAS DE ASSINATURA EM ANEL

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará como parte dos requisitos para obtenção do Título de Mestre em Engenharia de Teleinformática.

Orientador: Prof. Dr. José Cláudio do Nascimento

FORTALEZA

2014

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca de Pós-Graduação em Engenharia - BPGE

T615r Tomaz, Antonio Emerson Barros
Resgate de autoria em esquemas de assinatura em anel / Antonio Emerson Barros Tomaz. – 2014.
67 f. : il. , enc.; 30 cm.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2014.
Área de concentração: Eletronagnetismo Aplicado.
Orientação: Prof. Dr. José Cláudio do Nascimento.

1. Teleinformática. 2. Criptografia. I. Título.

CDD 621.38

UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

CAMPUS DO PICI, CAIXA POSTAL 6007 CEP 60.738-640
FORTALEZA – CEARÁ - BRASIL
FONE (+55) 85 3366-9467 – FAX (+55) 85 3366-9468

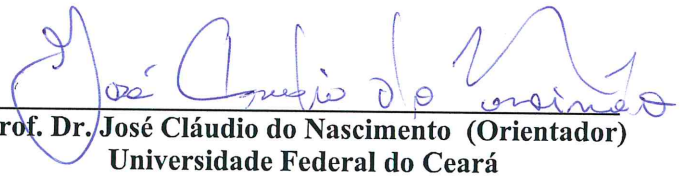
ANTONIO EMERSON BARROS TOMAZ

RESGATE DE AUTORIA EM ESQUEMAS DE ASSINATURA EM ANEL

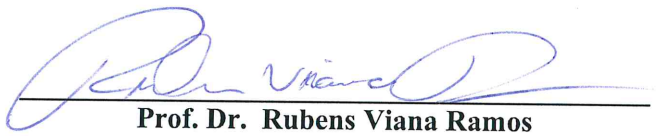
Dissertação submetida à Coordenação do Programa de Pós-Graduação em Engenharia de Teleinformática, da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Teleinformática.
Área de concentração: Eletromagnetismo Aplicado.

Aprovada em: 23/05/2014.

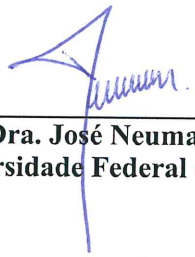
BANCA EXAMINADORA



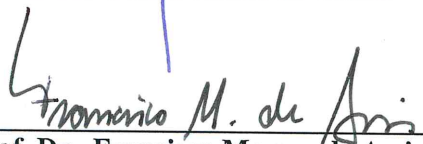
Prof. Dr. José Cláudio do Nascimento (Orientador)
Universidade Federal do Ceará



Prof. Dr. Rubens Viana Ramos
Universidade Federal do Ceará



Profa. Dra. José Neuman de Souza
Universidade Federal do Ceará



Prof. Dr. Francisco Marcos de Assis
Universidade Federal de Campina Grande

À minha filha Lara e à minha esposa Fernanda.

AGRADECIMENTOS

Agradeço a Deus, por prover a força que preciso.

À minha mãe, por ser um exemplo de força, pelas orações feitas enquanto eu viajava para estudar, mas sobretudo por ter sempre acreditado que a educação pode melhorar nossas vidas.

À minha esposa, por ter sido paciente, inacreditavelmente forte e sobretudo uma grande companheira nessa caminhada.

À minha filha, por ser a principal fonte de alegria na minha vida.

Ao meu orientador Prof. José Cláudio do Nascimento, pela atenção e incentivo.

Ao colega de estudos José Arteiro Frota Filho por ter contribuído para o desenvolvimento deste trabalho.

Ao meus irmãos por terem ajudado a formar uma família forte.

Ao meus colegas da Divisão de Tecnologia da Informação - UFC/*Campus* de Sobral pelo apoio incondicional que me deram.

Ao diretor, vice-diretor, técnico-administrativos do *Campus* de Sobral / UFC pelo apoio e aos professores Dr. Jarbas Joaci e Dr. Iális Cavalcante por gentilmente compartilharem suas experiências acadêmicas comigo.

Aos meus importantes amigos por terem proporcionado momentos de alegria e descontração quando eu estava vivendo momentos tensos.

“Não se pode olhar pra trás sem se aprender alguma coisa pro futuro.”

(Renato Russo)

RESUMO

A proposta apresentada nesta dissertação representa uma expansão do conceito original de assinatura em anel. Um esquema de assinatura em anel permite que um membro de um grupo divulgue uma mensagem anonimamente, de tal forma que cada um dos membros do grupo seja considerado o possível autor da mensagem. A ideia principal de uma assinatura em anel é garantir o anonimato do assinante e ainda garantir a autenticidade da informação, mostrando que a mensagem partiu de um dos membros do referido grupo. Esta dissertação apresenta um esquema de assinatura em anel baseado no esquema de Rivest *et al.* (2001), em que o assinante pode, mais tarde, revogar seu anonimato apresentando valores secretos que provam que somente ele seria capaz de gerar tal assinatura. Esta propriedade será chamada aqui de *resgate de autoria*. A principal diferença em relação ao trabalho de Rivest *et al.* (2001) é apresentada antes mesmo de começar a geração da assinatura. Os valores utilizados como entrada para a função trapdoor serão códigos de autenticação de mensagem - MACs gerados pelo algoritmo HMAC, um algoritmo de autenticação de mensagem baseado em função hash resistente à colisão. Essa modificação simples permitirá que, no futuro, o assinante revele-se como o verdadeiro autor da mensagem apresentando os valores secretos que geraram os MACs.

Palavras-chave: assinatura em anel, códigos de autenticação de mensagem, MAC, funções hash, funções trapdoor, RSA, criptografia.

ABSTRACT

The proposal presented in this thesis represents an expansion of the original concept of ring signature. A ring signature scheme allows a member of a group to publish a message anonymously, so that each member of the group can be considered the author of the message. The main idea of a ring signature is to guarantee the anonymity of the subscriber also ensure the authenticity of information, showing that the message came from one of the members of that group. This thesis presents a signature scheme based on (RIVEST *et al.*, 2001), where the subscriber can later revoke anonymity presenting secret values that prove that he would only be able to generate such a signature. This property will be referred to here as *rescue of authorship*. The main difference to the proposal of Rivest *et al.* (2001) is presented before we even begin signature generation. The values used as input to the trapdoor function are message authentication codes - MACs generated by the HMAC algorithm, an algorithm for message authentication based on hash function collision resistant. This simple modification will allow, in the future, the subscriber to reveal itself as the true author of the message by showing the secret values to generate those MACs.

Keywords: ring signature, message authentication code, MAC, hash functions, trapdoor functions, RSA, cryptography.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AKS	Algoritmo para testar a primalidade de inteiros criado por Agrawal,Kayal e Saxena
AC	Autoridade Certificadora
AR	Autoridade de Registro
DSS	*Digital Signature Standard*
GNFS	*General Number Field Sieve*
HMAC	Algoritmo de MAC baseado em função hash
ITI	Instituto Nacional de Tecnologia da Informação
$ITU-T$	*Telecommunication Standardization Sector*
NIST	*National Institute of Standards and Technology*
MAC	*Message Authentication Code*
*MD*5	*Message Digist 5*
RSA	Algoritmo de criptografia assimétrica criado por Rivest,Shamir e Adleman
SHA	*Secure Hash Algorithm*
X.509	Padrão para certificados digitais

SUMÁRIO

1 INTRODUÇÃO

A criptografia surgiu devido a necessidade de proteger informações importantes, sobretudo informações militares. O esquema de criptografia mais antigo que se conhece, remonta aos tempos do Império Romano. O imperador Júlio César utilizava uma técnica que consistia em substituir cada letra da mensagem pela letra que estava a três letras de distância no alfabeto. Essa técnica ficou conhecida como cifra de César. As técnicas de criptografia evoluíram e os motivos pelos quais se usa criptografia se diversificaram. Atualmente a criptografia não é mais usada apenas para fins militares, mas no passado, era um assunto exclusivo do Exército e do governo de um país. Atualmente a criptografia é acessível a qualquer pessoa e é amplamente utilizada em transações comerciais.

A criptografia é a base do desenvolvimento deste trabalho, porém o que será apresentado não é como usar a criptografia para proteger informações. Este trabalho apresenta como vazar informações, mas protegendo a identidade do delator.

Em ambientes corporativos ou governamentais, alguns indivíduos passam a ter acesso a certas informações importantes e consideram-se na obrigação de torná-las públicas. Porém, para proteger sua integridade física, preferem divulgar tais informações de forma anônima. Algumas dessas informações, quando tornam-se públicas, normalmente causam conflitos, principalmente quando a informação publicada revela escândalos de corrupção ou até mesmo crimes que violam os direitos humanos.

Inicialmente o autor da divulgação das informações deseja permanecer anônimo, pois não sabe o que pode acontecer com ele, caso as acusações não possam ser provadas. Esse indivíduo pode sofrer retaliações ou até mesmo sofrer um atentado. Mas, algumas vezes, essas informações são tão importantes para uma nação ou até mesmo para o mundo que o delator passa a ser considerado um herói. É nesse momento que o indivíduo delator gostaria de se apresentar como o autor da divulgação e assim poder receber homenagens e os devidos reconhecimentos pelo ato. Como pode ser observado, qualquer cidadão poderia apresentar-se como delator, porém não haveria como provar sua autoria.

Nesse caso, antes mesmo de pensar em ser reconhecido como autor de um ato importante

em seu país, o delator pensa em como garantir o anonimato incondicionalmente. Se essa situação se passar em um cenário em que as comunicações se dão quase que totalmente por meios computacionais, como nos dias atuais, esse indivíduo pode se beneficiar da criptografia para mantê-lo anônimo. Foi pensando nisso que Rivest *et al.* (2001) propuseram um método capaz de garantir o anonimato desse indivíduo. Trata-se de um esquema de assinatura em anel.

A noção de assinatura em anel foi introduzida em (RIVEST *et al.*, 2001). Um esquema desse tipo permite que um membro de um grupo divulgue uma mensagem anonimamente, de tal forma que cada um dos membros do grupo seja considerado o possível autor da mensagem. A ideia principal de uma assinatura em anel é garantir o anonimato e ainda garantir a autenticidade da informação, mostrando que a mensagem partiu de um dos membros do referido grupo. Mas o trabalho de Rivest *et al.* (2001) não prevê um problema importante nesse tipo de situação. O problema é o seguinte: o delator garante seu anonimato, mas não pode revelar-se como o autor da divulgação das informações, caso seja conveniente posteriormente. Este trabalho expande a ideia original de Rivest *et al.* (2001) e apresenta um método que garante o resgate da autoria da assinatura por parte do delator, caso ele deseje.

Em junho de 2013, um caso de vazamento de informação chamou a atenção de todo o mundo. O caso descrito a seguir representa bem uma situação em que a proposta apresentada neste trabalho poderia ser aplicada. Um membro da Agência Central de Inteligência (CIA) dos Estados Unidos, o então consultor técnico Edward Snowden resolveu revelar que o governo dos Estados Unidos monitorava e-mails e videoconferências de quem usava os serviços de empresas como Google, Skype e Facebook. Além disso Snowden revelou também que ligações telefônicas de cidadãos americanos e estrangeiros estavam sendo monitoradas. Por ter revelado detalhes do projeto de monitoramento global dos Estados Unidos, as autoridades federais apresentaram acusações formais contra esse ex-agente da CIA. Snowden poderia ter usado um esquema de assinatura em anel com resgate de autoria para preservar seu anonimato e garantir a revogação do seu anonimato quando fosse conveniente. Como Snowden não se preocupou em esconder sua identidade, precisou fugir do país e pedir asilo à Rússia.

Outro caso notório que também pode se enquadrar na proposta apresentada aqui é o caso de Julian Assange fundador do site WikiLeaks. Assange se tornou mundialmente conhecido quando começou a divulgar informações de delatores sobre temas críticos, como por exemplo, possíveis crimes de guerra cometido pelo Exército dos Estados Unidos. Assange, juntamente com a equipe do WikiLeaks, é responsável por divulgar documentos sigilosos de vários países relacionados à corrupção ou violação dos direitos humanos, mas sobretudo documentos confidenciais referente ao governo dos Estados Unidos. Assange preocupa-se em manter a fonte das informações que recebe anônima, mas sempre se expôs como membro do WikiLeaks. No caso de Assange, os esquemas de assinaturas em anel podem lhe ajudar a preservar suas fontes. Assange está desde 2012 asilado na embaixada do Equador no Reino Unido e teme ser

assassinado.

Assange contou com a colaboração importante de um membro do Exército dos Estados Unidos chamado Bradley Manning Edward. Em 2009 Manning trabalhou como analista de inteligência numa base do Exército no Iraque e tinha acesso a um banco de dados sigiloso. As informações sigilosas incluía alguns ataque aéreos feitos pelo Exército norte-americano ao Iraque e ao Afeganistão. Além disso, Manning tinha acesso a diversos telegramas diplomáticos e milhares de relatórios do Exército norte-americano. De posse dessas informações, em 2010, Manning vazou tais informações para o WikiLeaks que divulgou grande parte do material. Manning foi descoberto porque comentou o vazamento com um contato online. O contato de Manning informou ao Exército quem foi o responsável pelo vazamento das informações divulgadas pelo WikiLeaks. Em 2013, Manning foi condenado a 35 anos de prisão com base na Lei de Espionagem dos Estados Unidos. Esse é mais um caso em que o delator poderia ter feito uso de um esquema em assinatura em anel para preservar sua identidade.

Os casos de Snowden, Assange e Manning são apenas os casos mais conhecidos em que um esquema de assinatura em anel poderia ter evitado grandes problemas para quem deseja vazar uma informação sigilosa. As informações divulgadas por Snowden e Assange causaram um enorme desconforto aos governos de vários países, porém são informações importantes que ambos consideraram que o mundo precisava saber.

1.1 Objetivo

O objetivo deste trabalho é acrescentar uma propriedade a um método de denúncias devidamente assinadas que protege o delator, conhecido como assinaturas em anel (RIVEST *et al.*, 2001). Nota-se que a democratização da informação depende da ousadia dos delatores em suportar as consequências de suas denúncias. As fontes necessitam ser anônimas em alguns momentos, mas em outros precisam da exposição dos delatores, como no caso de Snowden, para que as informações divulgadas possam ter a devida credibilidade e repercussão, porém isso trás responsabilidades e consequências. Esta dissertação expande a ideia inicial de Rivest *et al.* (2001) e apresenta um protocolo que possibilita ao autor da assinatura em anel revogar seu anonimato. Tal propriedade será chamada de *resgate de autoria* e permite que os delatores vazem informações escolhendo permanecerem anônimos ou revelarem a autoria das denúncias em momentos que eles considerarem cruciais para sua segurança.

1.2 Organização do trabalho

Para apresentar um protocolo de resgate de autoria de uma assinatura em anel, este trabalho faz uso de ferramentas computacionais amplamente utilizadas em operações pela Internet. Tais ferramentas serão apresentadas e terão seus pontos fortes e fracos discutidos como descrito a seguir.

O capítulo 2 apresenta os princípios da segurança da informação e os fundamentos da criptografia. A criptografia representa a base de todo o trabalho, principalmente a criptografia de chave pública, introduzida por Diffie e Hellman (1976b) que a partir dela Rivest *et al.* (2001) puderam introduzir o conceito de assinatura em anel. Ainda neste capítulo há uma discussão sobre números primos e o problema da fatoração de inteiros, dois assuntos que serão fundamentais para discutir a segurança do algoritmo RSA no capítulo 4.

O capítulo 3 apresenta os Códigos de Autenticação de Mensagens - MACs e as funções hash, duas ferramentas utilizadas para resgatar a autoria da assinatura em anel. O principal objetivo do capítulo 3 é discutir o quão estas ferramentas são resistente a ataques, já que o delator de uma mensagem vai confiar em tais ferramentas para provar sua autoria.

No capítulo 4, apresenta-se o conceito de assinatura digital e como pode ser usada em transações comerciais e para produzir esquemas de assinaturas em anel. O algoritmo RSA é adotado como elemento da assinatura em anel, portanto a discussão da segurança desse algoritmo torna-se necessária.

Finalmente o capítulo 5, apresenta um protocolo para resgatar a autoria de uma assinatura em anel. Esse capítulo também apresenta um exemplo de cenário em que tal propriedade é útil. Além disso, mostra os trabalhos que tem relação com a presente dissertação. Isto é, relaciona trabalhos que já apresentaram tal propriedade de resgate de autoria, embora com outros nomes e outros métodos como em (LV; WANG, 2003; DONG *et al.*, 2012; FILHO; NASCIMENTO, 2011). Em Filho e Nascimento (2011) uma maneira de revogar o anonimato (inclusive com o nome resgate de autoria) foi proposta utilizando o conceito de dispositivos quânticos a prova de falsificação. Foi a partir desta proposta que surgiu a ideia de apresentar a propriedade de resgate de autoria de assinatura em anel utilizando apenas ferramentas computacionais populares e disponíveis para implementação.

1.3 Produção Bibliográfica

TOMAZ, A. E. B.; FILHO, J. A. F.; NASCIMENTO, J. C. Recuperando a autoria de uma assinatura em anel usando códigos mac. In: Simpósio Brasileiro de Telecomunicações, 30a., 2012, Brasília. *Anais...* Brasília: SBrT-Sociedade Brasileira de Telecomunicações, 2012.

2 PRINCÍPIOS DE SEGURANÇA E CRIPTOGRAFIA

O problema tratado neste trabalho se passa em um cenário conflituoso dentro de ambientes governamentais. O objeto desse conflito é a informação. É importante deixar claro, que este trabalho não apresenta técnicas para conseguir informações confidenciais. Pelo contrário, o que será tratado aqui é como divulgar certas informações supostamente importantes, mas mantidas em segredo por corporações ou governos. Embora o objetivo seja divulgar informações e não escondê-las, princípios da segurança da informação serão utilizados para esconder a identidade do delator.

Segurança da informação pode ser definida como a proteção de informação e sistemas de informação de acesso, modificação, destruição ou divulgação não autorizada (ANDRESS, 2011).

Este capítulo apresenta uma base teórica sobre os principais elementos que compõe a segurança da informação e a criptografia. Frequentemente serão apresentados cenários fictícios para exemplificar algum conceito. Para seguir a tradição em trabalhos sobre segurança da informação, Alice e Bob serão os personagens que aparecerão nos diversos cenários apresentados nesta dissertação. Possivelmente uma terceira personagem, chamada Eva, pode aparecer como uma intrusa na comunicação de Alice e Bob.

2.1 Propriedades da segurança da informação

As quatro principais propriedades da segurança da informação são confidencialidade, integridade, disponibilidade e autenticidade. Esses conceitos formam a base para as discussões sobre segurança que trata este trabalho.

2.1.1 Confidencialidade

Uma informação tem confidencialidade quando está protegida de divulgação ou exposição para indivíduos não autorizados. Confidencialidade garante que somente aqueles com direitos e privilégios sejam capazes de acessar a informação (WHITMAN; MATTORD, 2011). Quando indivíduos ou sistemas não autorizados podem ver a informação, então diz-se que a confidencialidade foi quebrada. Existem diversas formas de garantir a confidencialidade de uma informação, desde segurança física até técnicas matemáticas. Uma forma comum é a criptografia.

2.1.2 Integridade

Uma informação tem integridade quando é completa e não está corrompida. A integridade da informação é ameaçada quando essa informação é exposta à corrupção (WHITMAN; MATTORD, 2011). Uma informação está corrompida quando sofre manipulação não autorizada ou indesejada. Pode-se entender por manipulação, ações como inclusão, exclusão ou substituição de partes da informação. Quando uma informação sofre perda da integridade, não necessariamente foi resultado de um ataque por vírus ou por um indivíduo mal intencionado. Frequentemente, as informações sofrem corrupção durante a transmissão, devido a interferências naturais no canal de comunicação. A principal maneira de verificar se uma informação está íntegra é por meio das funções hash.

2.1.3 Disponibilidade

A disponibilidade é a propriedade da informação ser acessível e modificável no momento oportuno por aqueles que sejam autorizados a praticar tal ação (GOODRICH; TAMASSIA, 2013).

2.1.4 Autenticidade

Outro conceito bastante importante na segurança da informação é a autenticidade. A autenticidade da informação é um serviço relacionado à identificação das partes envolvidas na comunicação. Quando a comunicação entre duas pessoas acontece por meios que não permitem a identificação visual, é necessário estabelecer uma forma de garantir que a outra pessoa é realmente quem alega ser. Uma forma eficiente de prover autenticação é por meio dos MACs (Códigos de Autenticação de Mensagens).

Uma consequência da autenticidade de uma informação é o não repúdio. Quando uma informação é transmitida por um indivíduo cuja identidade não foi confirmada, esse indivíduo

pode, posteriormente, negar que emitiu tal informação. Assim, o **não repúdio** é a propriedade que afirmações autênticas emitidas por um indivíduo não podem ser negadas.

2.2 Criptografia

A informação sempre foi motivo de batalhas deste as civilizações mais antigas. Para proteger estratégias de guerra, exércitos utilizavam uma técnica conhecida como criptografia.

Tradicionalmente, a criptografia é definida como uma técnica capaz de tornar uma mensagem incompreensível, de forma que somente o destinatário seja capaz de decifrá-la e compreendê-la. Nesse cenário tradicional, o principal objetivo da criptografia é garantir a confidencialidade da comunicação entre duas pessoas quando estas utilizam um canal inseguro.

Atualmente, esse cenário básico não representa todos os objetivos da criptografia moderna. Segundo Goldreich (2004), desde a década de 1970, problemas como a construção de assinaturas digitais não falsificáveis e projetar protocolos tolerantes a falhas também foram considerados como domínio da criptografia. Desta forma, a definição seguinte representa melhor a criptografia moderna.

A criptografia é o estudo de técnicas matemáticas relacionadas a aspectos de segurança da informação, tais como a confidencialidade, integridade de dados, autenticação de entidade e autenticação da origem dos dados (MENEZES *et al.*, 1996).

2.2.1 Terminologia

Antes de apresentar outros conceitos relacionados à criptografia, é necessário conhecer o significado de alguns termos que serão utilizados ao longo deste trabalho.

- **Texto claro**: texto original, escrito em linguagem natural;
- **Texto cifrado**: texto ilegível, não compreensível. Exceto para o destinatário, que é capaz de decifrá-lo;
- **Cifrar ou Encriptar**: transformar texto claro em texto cifrado;
- **Decifrar ou Decriptar**: transformar texto cifrado em texto claro;
- **Chave**: conjunto de dados utilizados para cifrar e decifrar uma mensagem. No contexto computacional uma chave é um cadeia de bits usada por um algoritmo para produzir um texto cifrado ou obter um texto claro a partir de um texto cifrado.

2.2.2 O conceito de chave

A chave é simplesmente um valor numérico ou uma string que será utilizada durante a cifragem de uma mensagem. Tecnicamente, o que é usado no algoritmo de cifragem é apenas a representação binária destas strings. A mensagem que se deseja cifrar também será representada por uma cadeia de bits. O texto cifrado vai depender da função criptográfica e da chave utilizada. A função criptográfica [1] que transforma uma mensagem clara em uma mensagem incompreensível pode ser pública, apenas a chave utilizada nesta função é secreta. Assim, mesmo os algoritmos criptográficos sendo conhecidos, tais algoritmos são projetados cuidadosamente para não ser possível chegar ao texto claro a partir do texto cifrado.

Dependendo do tipo de chave utilizada nas operações de cifrar e decifrar informações, a criptografia é classificada em criptografia simétrica e criptografia assimétrica.

2.2.3 Métodos de cifragem

A maneira como o algoritmo de criptografia pode transformar o texto claro em texto cifrado pode ser de duas formas. A primeira é uma técnica conhecida como **cifra de substituição** e a segunda é conhecida como **cifra de transposição ou permutação**.

Cifra de substituição

A técnica de substituição é aquela em que as letras do texto claro são substituídas por outras letras, por números ou símbolos. Quando o texto claro é visto como uma sequência de bits, a substituição envolve substituir padrões de bits de texto claro por padrões de bits de texto cifrado (STALLINGS, 2008).

Cifra de transposição

Nas cifras de transposição, as letras em um bloco de tamanho m de um texto claro são reorganizados de acordo com uma permutação específica de tamanho m (GOODRICH; TAMASSIA, 2013). Cada permutação π, também tem uma permutação inversa π^{-1}, que desfaz todo o que foi feito por π. Em qualquer cifra de transposição, quem define a permutação a ser utilizada é a chave.

2.2.4 Criptografia simétrica

A criptografia simétrica é conhecida por criptografia de chave secreta. Esse método de criptografia utiliza a mesma chave para cifrar e decifrar uma mensagem. O emissor e o receptor concordam em utilizar uma determinada chave e então a compartilham entre eles. Esse

[1] Aqui o termo função criptográfica é utilizado com o mesmo sentido de algoritmos criptográfico.

compartilhamento da chave precisa ser seguro, do contrário, todo o processo criptográfico seria em vão.

Genericamente, um processo de criptografia simétrica pode ser representado pelo cenário a seguir: suponha que Alice deseja enviar uma mensagem confidencial a Bob. Alice sabe que o canal de comunicação não é seguro, portanto a mensagem a ser enviada será criptografada. Alice compartilha uma chave secreta com Bob, considere que a chave foi compartilhada em um momento anterior. Esse esquema de criptografia simétrica utilizado por Alice e Bob precisa de quatro elementos básicos:

- A mensagem a ser criptografada $M = \{M_1, M_2, ..., M_n\}$, em que M_i é cada elemento do texto claro. Esses elementos são cada letra de um alfabeto finito. Como atualmente a criptografia acontece em sistemas computacionais, cada um desses elementos será utilizado em sua forma binária, em que $M_i \in \{0, 1\}$.
- A chave criptográfica $K = \{K_1, K_2, ..., K_j\}$, previamente compartilhada entre Alice e Bob.
- Um algoritmo criptográfico E que realiza a transformação do texto claro em texto cifrado.
- Um algoritmo que representa a operação inversa D que realiza a transformação do texto cifrado em texto claro.

De posse dos elementos acima, o processo de cifragem pode ser realizado para produzir o texto cifrado $C = \{C_1, C_2, ..., C_t\}$. Isso pode ser escrito como

$$E(K, M) = C.$$

Ao receber a mensagem criptografada C, Bob realiza a operação inversa

$$D(K, C) = M.$$

Observe que é necessário um canal de comunicação para o compartilhamento da chave secreta diferente do canal utilizado para transmitir a mensagem criptografada. Assim nasce um dos grandes problemas da criptografia, conhecido como o problema da distribuição de chaves.

Claramente, o grande desafio desse método é como compartilhar a chave criptográfica de forma segura entre o emissor e o receptor da mensagem. A maneira mais natural de compartilhar essa chave seria estabelecer uma comunicação utilizando um outro canal considerado seguro ou um encontro pessoal privado. Outra maneira é fazer uso de um centro de distribuição de chaves, com o qual os usuários compartilham sua chave secreta.

Buchmann (2002) exemplifica a distribuição de chave usando uma terceira parte confiável. Suponha que Alice deseja enviar uma mensagem para Bob, então ela criptografa a mensagem

usando sua chave secreta e a envia ao centro das chaves. O centro, conhecendo todas as chaves secretas, decifra a mensagem usando a chave de Alice, e então a criptografa novamente usando a chave de Bob e envia a mensagem criptografada com a nova chave para Bob. Entretanto, o centro de chaves passa a ter conhecimento de todas as mensagens secretas. Conforme Diffie (1988), não há vantagem desenvolver sistemas criptográficos impenetráveis se os usuários foram obrigados a compartilhar suas chaves com um centro de distribuição de chaves cuja segurança pode ser comprometida por qualquer roubo ou suborno.

Para resolver o problema de distribuição de chave, Diffie e Hellman (1976a) apresentaram um método no qual duas pessoas podem produzir uma chave secreta compartilhada através da troca de informações públicas. Esta técnica ficou conhecida como **protocolo de troca de chave Diffie-Hellmam**.

O protocolo Diffie-Hellmam descreve uma maneira de trocar chaves secretas utilizando canais inseguros. Observe que esse protocolo não é em si um criptossistema, ele é utilizado para gerar e compartilhar a chave de modo seguro quando se pretende fazer uso de um sistema criptográfico simétrico. A relação matemática entre as informações públicas e a chave secreta é baseada na teoria dos números. O protocolo é descrito a seguir:

1. Alice e Bob concordam em usar um número primo p e como base g.
2. Alice escolhe um inteiro secreto $x \in \mathbb{Z}_p$, e usa para calcular $X = g^x \, mod \, p$. Então envia X para Bob.
3. Bob escolhe um inteiro secreto $y \in \mathbb{Z}_p$, e usa para calcular $Y = g^y \, mod \, p$. Então envia Y para Alice.
4. Alice calcula a chave secreta $K = Y^x \, mod \, p$.
5. Bob calcula a chave secreta $K = X^y \, mod \, p$.

De acordo Goodrich e Tamassia (2013), a segurança do protocolo é baseado na suposição de que é difícil para o atacante determinar a chave secreta K a partir dos parâmetros públicos p,g e dos valores X e Y, obtidos por espionagem. Mesmo sabendo que $X = g^x \, mod \, p$ e $Y = g^y \, mod \, p$, é computacionalmente difícil recuperar x de X ou y de Y quando p é um primo maior que 300 dígitos e x e y são maiores que 100 dígitos. Isso é equivalente ao problema do logaritmo discreto.

A computação de logaritmo discreto é um problema considerado computacionalmente difícil, isto é, não se conhece qualquer algoritmo que resolva esse problema em tempo polinomial. Caso o atacante descubra x e y poderá calcular a chave secreta $K = g^{x*y} \, mod \, p$. Porém sem o conhecimento de x e y não há métodos conhecidos para calcular K a partir de p e g.

2.2.5 Criptografia assimétrica

Até a década de 1970, a única forma de criptografar informações era por meio da criptografia simétrica. Diffie e Hellman (1976b) introduziram o conceito de criptografia de chave pública e deram exemplos de como esse esquema poderia ser realizado.

Criptografia de chave pública ou criptografia assimétrica baseia-se na utilização de um par de chaves conhecidas como chave privada e chave pública. Uma é usada para cifrar e a outra para decifrar. Existe uma relação entre a chave pública e a chave privada que garante que se uma mensagem for cifrada com a chave privada, apenas a chave pública poderá decifrá-la.

Os elementos essenciais para implementar um criptossistema assimétrico são os seguintes:

- A mensagem a ser criptografada $M = \{M_1, M_2, ..., M_n\}$, em que M_i é cada elemento do texto claro.
- Uma chave pública K_P, para cada participante, armazenada em algum repositório público.
- Uma chave privada K_S, para cada participante, armazenada em local seguro, de conhecimento apenas do proprietário.
- Um algoritmo criptográfico E que realiza a transformação do texto claro em texto cifrado.
- Um algoritmo que representa a operação inversa D que realiza a transformação do texto cifrado em texto claro.

Um processo genérico usando criptografia assimétrica pode ser representado pelo cenário a seguir: nesse esquema criptográfico, cada usuário (uma pessoa ou um computador) possui uma chave privada, que não é de conhecimento de mais ninguém e uma chave pública que é de conhecimento geral, inclusive de quem não faz parte da transmissão.

Suponha que Alice queira enviar uma mensagem confidencial M para Bob. Alice utiliza a chave pública de Bob K_{P_b} para cifrar a mensagem e então envia a mensagem cifrada para Bob

$$E(K_{P_b}, M) = C.$$

Ao receber a mensagem, Bob utiliza sua chave privada K_{S_b} para decifrar a mensagem

$$D(K_{S_b}, C) = M.$$

Assim, caso Eva consiga interceptar a mensagem, mesmo tendo conhecimento da chave pública de Bob, não conseguirá decifrá-la, pois isso só pode ser feito com a chave privada de Bob.

A criptografia assimétrica pode ser usada das seguintes formas:

1. O emissor cifra a mensagem com a chave pública do receptor. A mensagem é decifrada com a chave privada do receptor. Desta forma garante a **confidencialidade** da mensagem, mas não garante a autenticidade e o não-repúdio.

2. O emissor cifra a mensagem com sua própria chave privada. A mensagem é decifrada com a chave pública do emissor. Desta forma garante a **autenticidade** e o **não repúdio**, mas não garante a confidencialidade.

Requisitos para criptossistema assimétricos

Embora os criadores do conceito de criptografia de chave pública não tenham apresentado um algoritmo que de fato realizasse a cifragem da informação, eles estabeleceram condições a serem atendidas por tais criptossistemas (DIFFIE; HELLMAN, 1976b).

- É computacionalmente fácil calcular um par de chaves (K_P, K_S).
- É computacionalmente difícil calcular a chave privada K_S a partir do conhecimento da chave pública K_P.
- É computacionalmente fácil para um emissor, conhecendo a chave pública K_P do receptor, calcular $E(K_P, M) = C$.
- É computacionalmente difícil para um adversário, conhecendo apenas a chave pública K_P e um texto cifrado C, encontrar a mensagem original M.

Neste contexto, os termos "fácil" e "difícil" tem um significado especial. Diz-se que um problema é computacionalmente fácil (ou viável ou eficientemente computável) se existe um algoritmo que pode ser executado em tempo polinomial para resolvê-lo. Por outro lado, um problema é dito computacionalmente difícil (ou inviável) se não há algoritmo de tempo polinomial conhecido que possa solucioná-lo.

2.3 O problema da fatoração de números inteiros

Os principais criptossistemas de chave pública utilizados atualmente tem sua segurança diretamente relacionada com a dificuldade de fatorar números inteiros extremante grandes, como por exemplo números com mais de 100 dígitos.

O problema de fatoração de números inteiros consiste em encontrar fatores primos de um número inteiro n, que pode ser escrito $n = p_1^{e_1} p_2^{e_2} ... p_k^{e_k}$, em que p_i são primos distintos e cada $e_i \geq 1$ é a multiplicidade de p_i (MENEZES *et al.*, 1996).

Suponha que são conhecidos dois números inteiros p e q com mais de 100 dígitos cada e deseja-se calcular o produto $n = p \cdot q$. É fácil calcular esse resultado nos computadores atuais. Porém se dado apenas n suficientemente grande é computacionalmente inviável encontrar os fatores primos p e q de n.

Esta seção apresenta uma noção dos métodos de fatoração de inteiros para estabelecer uma relação com a segurança do RSA (RIVEST *et al.*, 1978). O RSA foi o primeiro criptossistema de chave pública desenvolvido baseado nas ideias de Diffie e Hellman (1976b) e é o principal criptossistema utilizado no processo de geração de assinaturas digitais. O RSA será discutido com mais detalhes no capítulo 4. O objetivo principal desta seção não é demonstrar cada um desses métodos, mas deixar claro que é extremamente difícil fatorar números inteiros grandes. Então, é com base nessa dificuldade que a segurança das chaves criptográficas do sistema RSA se apoiam.

Antes de aplicar qualquer método para fatorar um número inteiro n é necessário identificar se tal número é primo ou composto. Os números primos são fundamentais para a criptografia e principalmente na geração de chaves do RSA.

2.3.1 Os números primos

Os números primos despertam o interesse dos matemáticos deste a Grécia antiga. Foram os gregos que descobriram que os números primos são os elementos básicos de formação dos outros números. Embora a história mostre indícios que outros povos antigos já haviam percebido algumas características dos números primos. A obra clássica intitulada *Elementos* de Euclides de Alexandria, do século III a.C, é o registro mais antigo que se conhece sobre números primos.

Um numero primo é um número inteiro positivo maior que 1 divisível somente por 1 e por ele mesmo.

Os matemáticos classificam os números inteiros em números **primos** e **compostos**. Todo número inteiro positivo $n > 1$ se não é primo é chamado de número composto.

Os números primos são conceitos fundamentais para os criptossistemas de chave pública e assinaturas digitais. A criptografia assimétrica faz uso de números primos para gerar chaves criptográficas, isso significa que atividades comuns como transmitir mensagens com segurança e as transações comerciais pela Internet dependem fundamentalmente dos números primos.

Os números primos sempre atraíram a atenção de matemáticos do mundo todo, mas a partir da década de 1970 não só os matemáticos estavam interessados nos números primos. O artigo de (RIVEST *et al.*, 1978) sobre criptografia de chave pública, mostrou como os números primos poderiam ser usados para implementar sistemas criptográficos. Assim, fez

com que os primos se tornassem elementos importantes na segurança das comunicações por computadores e transações comerciais. Os números primos desempenham um papel fundamental na criptografia, a segurança da criptografia é baseada na dificuldade de fatorar números inteiros grandes em seus fatores primos.

Os números primos despertam o interesse e o respeito dos matemáticos porque são os elementos básicos na construção de todos os outros números. Os números compostos são formados pelo produto de números primos ou por números menores que são formados pelo produto de números primos.

Todo número inteiro maior que 1 pode ser representado de forma única como um produto de números primos, a menos da ordem dos primos. (Teorema Fundamental da Aritmética)

Os números primos tem uma outra característica importante que os torna tão fascinante: não exite padrão. A sequência de números primos parece totalmente aleatória. Apesar de sua aparente aleatoriedade, os matemáticos de todas as épocas acreditam que existe uma fórmula que descreva precisamente qual o próximo número primo de uma lista. A falta de padrão na formação dos números primos é considerado um dos grandes mistérios da matemática.

2.3.2 Distribuição dos primos e testes de primalidade

Na década de 1970, o trabalho de Rivest *et al.* (1978) tornou os números primos tão importantes para a criptografia como para a matemática. O RSA, por exemplo, usa dois números primos grandes para gerar o par de chaves. São números com 512 bits ou aproximadamente 150 algarismos. A geração de números primos grandes, pode acontecer escolhendo-se números inteiros aleatórios e em seguida aplica-se um teste de primalidade.

Como visto na seção anterior, não existe padrão na formação dos números primos. O que se sabe, a partir das observações do matemático Carl Friedrich Gauss é que a medida que os números vão aumentando, os números primos vão ficando cada vez mais raros. Então como encontrar números primos tão grandes para serem usados na criptografia? O Teorema dos Números Primos pode ajudar a responder esta questão. Esse teorema diz que: se $\pi(n)$ denota o número de primos $\leq n$, então

$$\lim_{n\to\infty} \frac{\pi(n)}{n/\ln n} = 1$$

Na prática, é necessário uma média de $\ln(n)$ testes para encontrar um primo da ordem de n. Como todos os inteiros pares podem ser imediatamente descartados a média correta é $\ln(n)/2$. Por exemplo, suponha que um primo com ordem de grandeza 2^{200} seja procurado, então cerca de $\ln(2^{200})/\ 2 = 70$ tentativas são necessárias para encontrar um primo (STALLINGS, 2008).

Conforme exemplo em (TERADA, 2008), a probabilidade de um inteiro n ser primo é

aproximadamente $1/ln\,n$, em que $ln\,n$ denota o logaritmo natural de n.

Tabela 1: Probabilidade de um inteiro n ser primo

n	100	1.000	10.000	100.000
$\frac{1}{ln\,n}$	1/4,6052 = 0.21715	1/6,9078 = 0,14476	1/9,2103 = 0,10857	1/11,513 = 0,08685

Fonte: (TERADA, 2008)

Testar se um número inteiro positivo n, quando n é extremamente grande, é uma tarefa custosa em termos computacionais. Porém há alguns algoritmos que provam a primalidade de forma eficiente.

Esta seção apresenta a descrição de alguns métodos para identificar se um determinado número é primo.

2.3.2.1 Divisão por tentativas / Crivo de Eratóstenes

A divisão por tentativas é o método mais intuitivo para descobrir se um número n é primo. Basta dividir n por todos os inteiros x , em que $2 \leq x < n$, se for encontrado um divisor, então n é composto. O número de tentativas pode ser reduzido para $\sqrt{n}$, conforme teorema a seguir.

Se n é um inteiro positivo composto, então n possui um divisor primo p que é menor ou igual a $\sqrt{n}$.

Desta forma basta que se tente dividir n por todo número primo p menor ou igual a $\sqrt{n}$. Assim, se for encontrado um divisor primo de n, então n é composto. Caso contrário, n é primo. Observe que se n for muito grande, esse método se torna impraticável. A demonstração desse teorema pode ser vista em (BUCHMANN, 2002).

Outro método simples é um método conhecido como crivo de Eratóstenes. É o mais antigo que se conhece para determinar se um número $n \in \mathbb{N}$ é primo. Foi criado por Eratóstenes, diretor de biblioteca da Alexandria no século III a.C. O método consiste em escrever uma determinada lista de números de 1 até n. A partir disso, basta eliminar os múltiplos de cada primo, fazendo com que, ao final do processo, sobrem apenas os primos menores que n. Observe que esse método é semelhante ao método de tentativas, pois estará com todos os números identificados como primos ou compostos ao chegar em $\sqrt{n}$.

Esses métodos são eficientes apenas para números inteiros pequenos, isto significa que não pode ser utilizado para identificar a primalidade dos números usados para gerar as chaves de criptossistemas assimétricos.

2.3.2.2 Teste de Miller-Rabin

O teste de Miller-Rabin (MILLER, 1976; RABIN, 1980) é um método probabilístico de tempo polinomial baseado no pequeno teorema de Fermat. É bastante utilizado porque tem uma probabilidade de erro menor em relação a outros testes de primalidade probabilísticos como, por exemplo, o teste de Solovay-Strassen (SOLOVAY; STRASSEN, 1977). O teste de Miller-Rabin é para dizer se um inteiro n é composto, o que significa que, se o teste afirma que n é composto, então ele realmente é composto. Já se o teste declarar que n é primo, existe apenas a possibilidade de n ser primo. Não há uma certeza.

Para qualquer inteiro impar composto n, a probabilidade de que o teste de Miller-Rabin declare n como "primo" é menor do que $\frac{1}{4}$, enquanto o teste de Solovay-Strassen tem probabilidade de erro de $\frac{1}{2}$. Além disso o teste Solovay-Strassen tem um custo computacionalmente maior (MENEZES *et al.*, 1996). A probabilidade de erro do teste de Miller-Rabin pode ser reduzido a um número arbitrariamente pequeno, aumentando o número de iterações realizada, assim a probabilidade pode ser $(\frac{1}{4})^t$, em que t representa o número de iterações ou de testemunhos de n. Os detalhes do teste de Miller-Rabin e exemplos de uso podem ser visto em (TERADA, 2008)

Em resumo, o teste declara que n é composto com certeza ou que n é primo com probabilidade $\geq 1 - (\frac{1}{4})^t$. O algoritmo tem complexidade temporal de $O(\log^3 n)$ (BACH; SHALLIT, 1996).

2.3.2.3 Teste de Agrawal-Kayal-Saxena

O algoritmo AKS (AGRAWAL *et al.*, 2004) é um método determinístico polinomial para testar a primalidade de inteiros. O principal avanço mostrado pelo trabalho de Agrawal *et al.* (2004) é que o teste de primalidade está na classe P (YAN, 2012). O algoritmo original é executado com um custo computacional $O(\log^{12} n)$. Em (AGRAWAL *et al.*, 2004), baseado nas conjecturas de Hendrik Lenstra Jr, os autores mostram que é possível produzir uma versão do algoritmo na qual a complexidade é $O(\log^6 n)$. Mesmo assim, o algoritmo ainda tem um desempenho abaixo do algoritmo de Miller-Rabin, que mesmo para números grandes consegue fornecer uma resposta com uma probabilidade de erro muito baixa em segundos. Assim, de acordo com Stallings (2008), o algoritmo AKS não parece ser tão eficiente como o algoritmo de Miller-Rabin. Até agora ele não superou essa técnica probabilística mais antiga.

2.3.3 Algoritmos de fatoração

Como visto no início desta seção, a dificuldade computacional do problema da fatoração de inteiros tem sido usada em alguns criptossistemas assimétricos bem conhecidos. Embora exista

diversos trabalhos publicados sobre algoritmos de fatoração, não se conhece um algoritmo computacionalmente eficiente para resolver esse problema em máquinas não quânticas. Caso fosse descoberto um algoritmo eficiente, vários sistemas criptográficos importantes seriam quebrados, inclusive o RSA. Os algoritmos apresentados aqui são algoritmos que fatoram números genéricos, sem características especiais.

Suponha que n seja um número inteiro composto já identificado por um dos algoritmos citados anteriormente, então pode-se aplicar um dos algoritmos de fatoração a seguir.

2.3.3.1 Divisão por tentativa

O método mais simples de encontrar os fatores de um número inteiro n é fazer a divisão por algum inteiro k tal que $2 \leq k < n$. Se algum desses números dividir n então k é um fator de n. Na verdade, se n for composto, algum fator deverá ser encontrado antes de chegar a $\sqrt{n}$ (COUTINHO, 2005).

Observe que, para números grandes, esse método é tão ineficiente quanto o método de divisão por tentativas para testar primalidade. O algoritmo precisa executar $\sqrt{n}$ operações. Para números de 100 dígitos ou mais, isto é $n \geq 10^{100}$, é necessário pelo menos 10^{50} divisões. Isso explica o motivo pelo qual os números que geram as chaves do RSA precisam ser números grandes. Uma análise desse método pode ser vista em (BRIGGS, 1998). Suponha que um número inteiro n de 60 dígitos a ser fatorado. Em seguida, deve-se verificar se n é divisível por qualquer um dos primos entre 2 e 10^{30}. Numa suposição otimista, que apenas $0,1\%$ são primos. Mesmo assim, seria necessário cerca de 10^{27} divisões. Numa outra suposição otimista, um computador seria capaz de executar 10^{15} dessas divisões por segundo, ainda levaria cerca de 10^{12} segundos, aproximadamente 31.000 anos para executar esse cálculo. Observe que mesmo com suposições bem otimistas esse não é um método prático para fatorar um inteiro grande.

2.3.3.2 Um método de fatoração mais eficiente

Os algoritmos para fatoração de inteiros, normalmente são divididos em duas categorias: os algoritmos para fins especiais e os algoritmos de propósito geral. Dessa primeira classe fazem parte os métodos que assumem que os números a serem fatorados satisfazem certas condições, como os métodos de Pollard (1975).

A segunda classe é formada por algoritmos de uso geral, isto é, algoritmos que tratam os inteiros de forma completamente geral. Nesta classe está o GNFS (General Number Field Sieve), um poderoso algoritmo de fatoração de propósito geral. Atualmente é o método mais rápido que se conhece para fatorar inteiros grandes [2]. Na verdade, o algoritmo mais

[2]Geralmente entende-se por inteiro "grande", um número com mais de 110 algarismos.

rápido já criado para esse proposito é o algoritmo de Shor (1994) que tem complexidade $O((\ln n)^2 \cdot \ln \ln n)$. Porém esse algoritmo foi projetado para computadores quânticos que ainda não estão disponíveis devido a falta de avanços tecnológicos.

O GNFS é um algoritmo complexo que utiliza resultados de vários campos da matemática. Uma análise detalhada da base matemática desse algoritmo pode ser vista em (BRIGGS, 1998). Como o GNFS é considerado o mais importante algoritmo de fatoração de inteiros atualmente, sua complexidade de tempo tem um impacto importante em ataques contra o RSA. Este algoritmo roda em tempo $exp\left((c+o(1))n^{\frac{1}{3}}log^{\frac{2}{3}}n\right)$ para qualquer $c < 2$ (BONEH, 1999). [3] De acordo com Briggs (1998), o GNFS detém o recorde de ter fatorado o maior inteiro geral, um número de 200 algarismos.

Desta forma, os outros métodos de fatoração que levam mais tempo que o GNFS para fatorar um número não são mais importantes, quando a questão é a segurança de sistemas criptográficos (BOUCINHA, 2011).

Embora tenha havido vários avanços nessa área, um algoritmo para máquinas não quânticas que fatora em tempo polinomial grandes inteiros ainda não é conhecido.

[3] A classe EXP de complexidade contém todos os problemas que rodam em tempo exponencial, isto é, $O(2^{n^k})$.

3 MAC E FUNÇÕES HASH

Este capítulo apresenta duas classes de funções utilizadas em sistemas criptográficos. A primeira classe é formada pelas funções hash utilizadas para verificação de integridade de dados. As funções hash podem ser aplicadas em diversas situações, porém neste trabalho, interessa apenas seu uso na criptografia. A segunda classe é formada pelas funções MAC (*Message Authentication Code*), um tipo de função utilizada para prover códigos de autenticação de mensagens. As funções MAC podem ser construídas com base em cifras de blocos ou podem ser construídas com base em funções hash. Quando a função MAC é construída com base em funções hash é chamada de *função hash chaveada*.

Este trabalho discute apenas as funções MAC construídas a partir de funções hash. Essas funções pertencem a um grupo conhecido como funções one-way ou funções unidirecionais. Uma função $f : X \to Y$ é chamada de sentido único, se dado $x \in X$ é fácil computar $f(x) = y$, mas dado apenas $y \in Y$ é computacionalmente difícil encontrar $x \in X$ a partir de y.

3.1 Funções hash

Uma função hash é uma função H que recebe como entrada uma mensagem m no formato de uma cadeia de bits de tamanho arbitrário e retorna uma cadeia de bits x de tamanho fixo, chamado de **código hash** ou **etiqueta hash**, em que $|m| > |x|$. Esta função pode ser denotada por

$$H(m) = x$$

Mais precisamente, uma função hash é um mapeamento unidirecional $H : \{0,1\}^* \to \{0,1\}^i$, para algum $i > 1 \in \mathbb{N}$. Neste mapeamento, $D = \{0,1\}^*$ é o domínio da função e representa as possíveis mensagens de tamanhos arbitrários e $I = \{0,1\}^i$ é a imagem da função e representa as possíveis etiquetas hash de tamanho i. Nas funções de hash utilizadas atualmente, o tamanho de i é 160, 256 ou 512 bits, assim a quantidade de etiquetas hash possíveis em cada caso é 2^i.

O tamanho da mensagem[1] $m \in D$ é muito maior que o tamanho da etiqueta hash $x \in I$, isto significa que uma função hash pode reduzir a mensagem original em uma string muito menor de tamanho fixo, independente do tamanho da entrada. Por causa dessa redução de tamanho, as funções hash também são conhecidas como função resumo.

As funções hash estão diretamente relacionadas com a verificação da integridade de mensagens. O objetivo central de uma função hash é gerar uma etiqueta que forneça uma identificação unívoca para cada mensagem. Como o conjunto D é muito maior que o conjunto I, é possível que duas mensagens m e m' sejam mapeadas para a mesma etiqueta hash x, isso é conhecido como colisão. Formalmente, uma **colisão** da função hash H é quando duas mensagens $m \neq m'$ produzem $H(m) = H(m')$.

A forma como uma função hash é construída garante que, se houver qualquer alteração na mensagem original, ainda que modificado um único bit, uma outra etiqueta hash será gerada. Dessa forma é possível identificar se uma mensagem foi violada durante a transmissão.

Exemplo comum da aplicação de uma função hash

Suponha que Alice queira proteger um arquivo m, armazenado em seu computador, de mudanças indevidas. Alice pode gerar uma etiqueta hash desse arquivo com a função $H(m) = x$, após sua última alteração. Posteriormente, ao desconfiar que o arquivo m sofreu alguma alteração e tornou-se m', Alice pode novamente fazer uso da função hash usando como entrada o arquivo supostamente modificado, fazendo $H(m') = x'$. Se $x = x'$, então há grande probabilidade do arquivo não ter sido alterado, se $x \neq x'$ então Alice constata que o arquivo foi alterado indevidamente. Em rigor, como aplicação criptográfica, as funções de hash não tem o objetivo de garantir confidencialidade de uma mensagem e sim a integridade.

3.1.1 Propriedades essenciais das funções hash criptográficas

Para serem usadas em sistemas criptográficos, as funções hash devem satisfazer algumas propriedades adicionais e passam a ser chamadas de funções hash criptográficas (STALLINGS, 2008).

1. **Resistência à pré-imagem**: dada uma mensagem m é fácil calcular uma etiqueta hash x tal que $H(m) = x$. Mas dado uma etiqueta hash x é computacionalmente difícil encontrar m a partir de x. Esta propriedade também é conhecida como propriedade **unidirecional**.

2. **Resistência à segunda pré-imagem**: dado uma determinada mensagem m, é computacionalmente difícil encontrar uma mensagem $m' \neq m$, tal que produza a mesma etiqueta hash $H(m') = H(m)$.

[1] Neste contexto, o termo **mensagem** é utilizado para representar qualquer conjunto de dados.

3. **Resistência à colisão**: é computacionalmente difícil encontrar um par de mensagens (m, y) qualquer que produza o mesmo hash, tal que $H(m) = H(y)$.

Além dessas propriedades necessárias às funções hash criptográficas, toda função hash deve ter duas propriedades básicas que se confundem com a própria definição:

- Compressão: a função deve mapear uma cadeia de tamanho bits arbitrária para uma cadeia de bits de tamanho fixo.
- Fácil de computar: dada uma mensagem m é fácil computar $H(m) = y$.

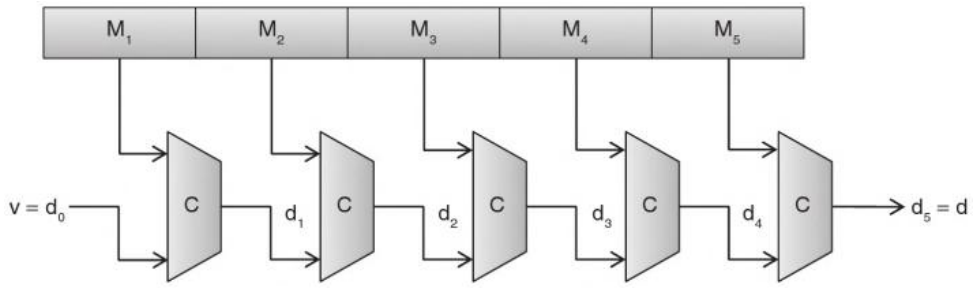
3.1.2 Construção de funções hash resistentes à colisão

Segundo Damgård (1989), uma função hash é chamada livre de colisão, se mapear mensagem de qualquer tamanho para string de tamanho fixo, mas de tal forma que encontrar mensagens x e y com $H(x) = H(y)$ seja um problema difícil.

As principais funções hash utilizadas atualmente são construídas com base em uma estrutura conhecida como construção de Merkle-Damgård (MERKLE, 1979). Esse método define como construir funções hash resistentes à colisão a partir de funções de compressão resistentes à colisão. A construção de Merkle-Damgård utiliza uma estrutura conhecida como função de hash iterada que faz uso de uma função de compressão repetidas vezes.

Uma função de compressão é uma função que a partir de uma cadeia de bits de tamanho m gera uma cadeia de bits de tamanho n, em que $m > n$. Formalmente, uma função de compressão é um mapeamento $C : \{0,1\}^m \rightarrow \{0,1\}^n$, com n, $m \in \mathbb{N}$.

Figura 1: Construção de função hash pelo método Merkle-Damgård



Fonte: (GOODRICH; TAMASSIA, 2013)

M_i representa cada bloco da mensagem M; C é a função de compressão; d_i representa a etiqueta hash gerada em cada iteração e v é o valor de inicialização.

Segundo Merkle (1989) e Damgård (1989), se a função de compressão é resistente à colisão, então a função de hash construída também é resistente à colisão. Portanto, o problema de

projetar uma função hash segura que tenha como entrada mensagens de qualquer tamanho, reduz-se a projetar uma função de compressão resistente à colisão que tenha como entrada alguma cadeia de bits de tamanho fixo. Um algoritmo para a construção de uma função hash utilizando o método de Merkle-Damgård é descrito no algoritmo 1.

Algoritmo 1 A construção Merkle-Damgård

Entrada: Função de compressão C resistente a colisão
Saída: Função h com a propriedade de resistência à colisão

1: Dada uma função de compressão $C : \{0,1\}^n \times \{0,1\}^b \rightarrow \{0,1\}^n$ que utiliza como entrada duas sequências $\{0,1\}^n$ e $\{0,1\}^b$ e produz um hash de tamanho n.
2: Dada uma mensagem M, divide-se M em t blocos $(M_1, M_2, ..., M_t)$ de b bits cada. O último bloco será completado, se necessário, com um bit igual a 1 seguido de bits 0 até o limite b;
3: Em seguida cria-se um bloco extra M_{t+1} que contém a representação binária do tamanho da mensagem, isto é, o tamanho de M;
4: Seja $d_0 = v$ o valor inicial utilizado como entrada para a função de compressão C na primeira iteração;
5: Para cada iteração $i = 1, 2...t+1$, definir $d_i = C(d_{i-1}, M_1)$;
6: O valor do hash d da mensagem original, será o valor hash d_t produzido na última iteração.

Na função de compressão C, $\{0,1\}^b$ representa os blocos de entrada de b bits e $\{0,1\}^n$ representa a etiqueta hash de n bits produzida pela própria função de compressão na iteração anterior. Como na primeira iteração não há valor de hash, será assumido um valor v chamado de *vetor de inicialização*. Observe que em cada iteração i uma etiqueta hash d_i será gerada e utilizada como uma das entradas da função de compressão $C(d_{i-1}, M_i)$ na iteração seguinte, a outra entrada é o bloco M_i da mensagem original. Na primeira iteração, assume-se o hash inicial sendo $d_0 = v$. Observe na figura 1 o funcionamento da construção Merkle-Damgård, dada uma função de compressão C.

3.1.3 Segurança das funções hash

A segurança das funções hash está diretamente relacionada com a dificuldade de gerar colisões, quanto maior a dificuldade de encontrar colisão, maior o nível de segurança da função. Há basicamente duas formas de atacar a resistência à colisão de uma função hash. A primeira é por criptoanálise e a segunda é por meio de um método conhecido como ataque do aniversário.

3.1.3.1 Ataque por criptoanálise

Na criptoanálise das funções hash o criptoanalista foca na estrutura interna da função de compressão. Como visto na seção 3.1.2, a construção da maioria das funções hash atuais são projetadas com base em uma estrutura iterada que faz uso repetidamente de uma função de compressão.

O objetivo é encontrar uma técnica eficiente que consiga gerar uma colisão apenas com uma única execução da função. O criptoanalista estuda a mudança no padrão dos bits em cada iteração para tentar identificar a estrutura interna da função de compressão (KUMAR, 2006).

Basicamente o que difere uma função hash de outra, quando ambas utilizam a construção Merkle-Damgårda (Seção 3.1.2), é a função de compressão utilizada.

As funções de compressão podem ser construídas utilizando cifras de bloco, aritmética modular ou até mesmo serem construídas especificamente para uma determinada função hash (MENEZES *et al.*, 1996).

3.1.3.2 Ataque do aniversário

O ataque do aniversário é uma técnica que busca diminuir o tempo necessário para encontrar uma colisão. Em 1979, G. Yuval, apresentou um ataque a um esquema de assinatura digital com base no paradoxo do aniversário (YUVAL, 1979). Antes de descrever como ocorre o ataque do aniversário nas funções hash, é necessário uma breve explicação sobre a matemática do paradoxo do aniversário.

O Paradoxo do aniversário

O paradoxo do aniversário é baseado na seguinte pergunta: quantas pessoas precisam estar reunidas para que a probabilidade de pelo menos duas delas compartilharem a mesma data de aniversario seja maior que 50%?

No paradoxo do aniversário, é necessário um grupo de 23 pessoas para que a probabilidade de duas dessas pessoas compartilharem a mesma data de aniversário seja maior que 50%. A probabilidade parece alta demais para um grupo tão pequeno, assim cria-se essa ideia de paradoxo. De forma resumida, o paradoxo do aniversário é descrito a seguir.

Deve-se calcular a probabilidade de pelo menos duas pessoas fazerem aniversário no mesmo dia, considerando um grupo de n pessoas e um ano de 365 dias. Como existem muitas possibilidades de satisfazer esse problema (encontrar mais de duas pessoas aniversariando na mesma data), então é mais fácil calcular a probabilidade de todas as n pessoas fazerem aniversários em datas diferentes. Desta forma, a primeira pessoa do grupo tem 365 chances em 365 de não compartilhar a mesma data de aniversário com outra pessoa, assim a probabilidade da primeira pessoa não fazer aniversário na mesma data de alguém é $365/365 = 1$. A segunda pessoa pode fazer aniversário em qualquer um dos 364 dias restantes, menos no dia do aniversário da primeira pessoa, assim $364/365 = 0,9973$. Observe que a quantidade de dias possíveis sem coincidir datas de aniversário vai diminuindo, portanto a probabilidade de haver colisões vai aumentando.

Generalizando esta observação, a probabilidade $\bar{p}(n)$ de n pessoas terem aniversários

diferentes é

$$\bar{p}(n) = \left(\frac{365}{365}\right) \cdot \left(\frac{364}{365}\right) \cdot \left(\frac{363}{365}\right) \cdots \left(\frac{365-(n-1)}{365}\right). \tag{3.1}$$

Uma equação fechada para a equação 3.1, pode ser escrita na forma

$$\bar{p}(n) = \frac{365!}{365^n(365-n)!}, \tag{3.2}$$

em que $\bar{p}(n)$ é a probabilidade de não haver colisão de datas, portanto a probabilidade de haver colisão é

$$p(n) = 1 - \bar{p}(n). \tag{3.3}$$

Assim, para $n = 23$, a probabilidade de pelo menos duas pessoas compartilharem a mesma data de aniversário é de 50.7%. A base matemática de como a equação 3.1 resultou na equação 3.2 podo ser vista no apêndice A.

O Ataque do aniversário nas funções hash

O raciocínio do paradoxo do aniversário pode ser estendido para encontrar colisões em uma determinada função hash. Esse ataque é independente do algoritmo, isto é, pode ser aplicado à qualquer função hash.

O problema do aniversário é apenas um problema específico. Aqui o número 365 (representando a quantidade de dias do ano) será substituído por $n = 2^b$ em que b é quantidade de bits da etiqueta hash e 2^b é quantidade de etiquetas hash possíveis.

Suponha dois cenários em que alguma função hash H com n saídas possíveis pode ser utilizada. No primeiro cenário, Bob conhece uma mensagem m e deseja encontrar uma outra mensagem y tal que $H(y) = H(m)$. Em quantas entradas k Bob teria que aplicar a função hash H para encontrar, com probabilidade 0.5, tal mensagem y cujo hash seja exatamente igual a $H(m)$. Neste caso aplica-se o mesmo raciocínio utilizado no paradoxo do aniversário. Para esse problema, se a função for aplicada a uma única mensagem y, a probabilidade de $H(y) = H(m)$ é $1/n$.

No segundo cenário, Bob deseja encontrar um par de mensagens qualquer (m, y) tal que $H(m) = H(y)$. Observe que Bob não está procurando uma mensagem m que gere um hash específico como no primeiro cenário, neste caso basta duas mensagens quaisquer produzirem o mesmo hash. Esse cenário lembra o questionamento do paradoxo do aniversário, em que os valores de hash possíveis correspondem as datas de aniversários possíveis. Assim, mais uma vez, tomando como base tal paradoxo, a equação

$$p(k) = 1 - \frac{n!}{n^k(n-k)!} \tag{3.4}$$

calcula aproximadamente quantas entradas k para a função hash H são necessárias para gerar

colisão com probabilidade de 0.5. A partir da equação 3.4, pode-se dizer que a quantidade de entradas k para gerar uma colisão, com probabilidade de 0.5 é

$$k = \sqrt{2(ln\,2)n} = 1.1774\sqrt{n} \tag{3.5}$$

ou simplesmente

$$k \approx \sqrt{n}. \tag{3.6}$$

A base matemática de como a equação 3.4 resultou na equação 3.6 podo ser vista no apêndice A.

A partir do ataque do aniversário, G. Yuval projetou um algoritmo chamado *Ataque pela raiz quadrada* (YUVAL, 1979). Dada uma função hash H com n bits de saída, o total de etiquetas hash possíveis é igual a 2^n. Produzir duas mensagens m e m' tal que $H(m) = H(m')$, é necessário aproximadamente $2^{n/2}$ tentativas.

O ataque explora eficientemente uma propriedade específica das funções hash, a propriedade de resistência à colisão (propriedade 3 da seção 3.1.1). Observe atentamente, que o ataque a essa propriedade, ainda que bem sucedido, não implica necessariamente na quebra geral de resistência à colisão da função atacada. Segundo (MENEZES *et al.*, 1996), resistência à colisão implica na resistência à segunda pré-imagem da função hash (propriedade 2 da seção 3.1.1).

O ataque do aniversário não é eficiente contra a resistência à segunda pré-imagem. Assim pode parecer que a sua aplicabilidade é muito limitada, pois em aplicações como autenticação e assinaturas digitais, um adversário não estaria interessado em uma colisão com uma mensagem aleatória, mas sim em uma colisão com uma mensagem específica. Para contrariar essa ideia, o algoritmo de Yuval (algoritmo 2), mostra como o ataque do aniversário pode ser usado para encontrar colisões com efeitos importantes.

Algoritmo 2 O Algoritmo do Ataque do Aniversário de Yuval. Adaptado de (MENEZES *et al.*, 1996)

Entrada: mensagem m_1 legítima; mensagem m_2 fraudulenta; função hash H com saída de n bits

Saída: m'_1 e m'_2 resultantes de pequenas modificações em m_1 e m_2 tal que $H(m'_1) = H(m'_2)$

1: Gere $t = 2^{n/2}$ mensagens m'_1 resultantes de pequenas modificações em m_1 (modificações que não altere o sentido da mensagem original);
2: Aplique a função hash H em cada uma da mensagens modificadas m'_1 e armazene os valores hash $H(m'_1)$ juntamente com a mensagem m'_1 em uma tabela;
3: Gera mensagens m'_2 resultante de pequenas modificações em m_2; para cada mensagem m'_2 gerada aplique a função hash H e verifique se para algum m'_2 existe $H(m'_2) = H(m'_1)$ na tabela do passo anterior. Com base no paradoxo do aniversário, uma colisão será encontrada, com probabilidade de $1/2$, após t gerações de mensagens m'_2.

Observe na tabela 2 o nível de esforço computacional exigido para encontrar colisão em uma função hash com saída de tamanho n.

Tabela 2: Esforço computacional exigido para um hash de tamanho n bits

Resistência à pré-imagem	2^n
Resistência à segunda pré-imagem	2^n
Resistência à colisão	$2^{n/2}$

Fonte : (STALLINGS, 2008)

Como o tamanho do conjunto imagem de uma função hash (conjunto das etiquetas hash possíveis) é muito menor que o domínio da função, sempre haverá colisões nesta função. O que precisa ser feito para evitar o ataque do aniversário, é projetar uma função cujo o tamanho n da etiqueta hash torne o cálculo de $2^{n/2}$ computacionalmente difícil. Atualmente as funções hash utilizam etiquetas hash de até 512 bits.

Por exemplo, considere uma função hash H com saída de 256 bits, essa função é imune ao ataque do aniversário porque exige no mínimo 2^{128} operações, que é computacionalmente inviável para os dias atuais (PARDO, 2013).

3.1.4 Algoritmos de hash seguros

Nesta seção, serão descritos alguns algoritmos de hash que, dependendo do tamanho da cadeia de bits de saída, são considerados seguros ou não.

A maioria dos algoritmos de hash considerados seguros atualmente são projetados com base na construção Merkle-Damgård descrita na seção 3.1.2. Segundo o National Institute of Standards and Technology (2008), um algoritmo hash é chamado de seguro quando é computacionalmente inviável:

- encontrar uma mensagem a partir de uma dada etiqueta hash;
- encontrar duas mensagens diferentes que produzam a mesma etiqueta hash.

Hoje os principais algoritmos de hash utilizados em diversas aplicações, inclusive em sistemas criptográficos, são os algoritmos do padrão SHA (*Secure Hash Algorithm*). O SHA foi projetado em 1993 pelo NIST (National Institute of Standards and Technology) e adotado como padrão pelo governo norte-americano. Foi desenvolvido originalmente para ser utilizado no DSS (*Digital Signature Standard*), mas pouco tempo depois de sua publicação foram detectados problemas de segurança. Assim, depois de uma revisão, o algoritmo foi publicado como SHA-1

(National Institute of Standards and Technology, 1995). Desde então o SHA original passou a ser chamado de SHA-0 para evitar confusão com os demais algoritmos do padrão SHA.

O SHA-1 produz uma etiqueta hash de 160 bits e devido a sua popularização, passou as ser bastante analisado por pesquisadores. Diversos trabalhos tem sido publicados apontando métodos de gerar colisão nesse algoritmo. Em 2005, Biham et. al. publicaram *Collisions of SHA-0 and Reduced SHA-1* (BIHAM *et al.*, 2005) sobre colisões no SHA-0/SHA-1. Ainda em 2005, Wang et. al. no artigo *Finding Collisions in the Full SHA-1* introduziram um conjunto de estratégias que podem ser utilizados para facilitar a busca por colisão para o SHA-1. Devido a esses e outros trabalhos, em 2002, o NIST produziu novas versões do padrão SHA. As novas versões são os algoritmos SHA-224, SHA-256, SHA-384 e SHA-512 (National Institute of Standards and Technology, 2008) e geram respectivamente saídas de 224, 256, 384 e 512 bits. Esse conjunto de variações, ficou conhecido como SHA-2 e assim como o SHA-1, é projetado com base na construção Merkle-Damgård.

Outro importante algoritmo de hash é o MD5 - *Message Digist 5* (RIVEST, 1992) é um algoritmo de hash construído em 1991 por Ronald Rivest para substituir o MD4 (considerado vulnerável). Esse algoritmo também é construído com base no método Merkle-Damgård, o tamanho da etiqueta hash produzida é de 128 bits, esse tamanho de hash é considerado pequeno nos dias atuais. O MD5 foi amplamente utilizado, mas atualmente é considerado inseguro. Nos últimos anos vários trabalhos foram publicados explorando a resistência à colisão do MD5. Um dos primeiros foi o artigo *Collisions for the Compression Function of MD5* (BOER; BOSSELAERS, 1993).

3.2 Código de Autenticação de Mensagem - MAC

Como visto na seção anterior, as funções hash criptográficas são utilizadas para verificar a integridade de mensagens. Porém, em algumas situações, não só a integridade da mensagem que precisa ser verificada, mas também sua autenticidade.

Segundo Stallings (2008), autenticação de mensagem é um procedimento para verificar se a mensagem recebida provêm realmente da origem afirmada e se não foi alterada. Isso significa que o processo de autenticação garante também a integridade da mensagem.

Para autenticar uma mensagem, utiliza-se um tipo de função chamada MAC - Message Authentication Code (Código de Autenticação de Mensagem). Essa função tem o proposito de garantir autenticação da origem dos dados, mas não garante a privacidade deles.

MAC pode ser entendido como uma técnica de autenticação que envolve o uso de uma mensagem e uma chave secreta, para gerar um pequeno bloco de dados de tamanho fixo, que é transmitido juntamente com a mensagem (STALLINGS, 2008).

Formalmente, MAC é uma função $MAC : \{0,1\}^* \times \{0,1\}^k \rightarrow \{0,1\}^n$, que mapeia uma mensagem $m \in \{0,1\}^*$ de comprimento arbitrário e uma chave secreta $k \in \{0,1\}^k$ em uma saída de comprimento fixo $t \in \{0,1\}^n$ semelhante a uma etiqueta hash. Tal função pode ser denotada por

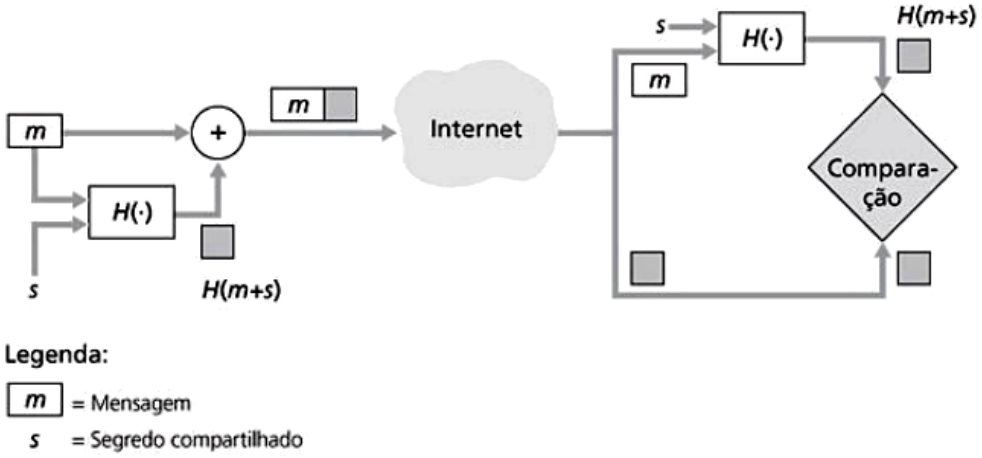
$$MAC_k(m) = t$$

MAC também pode ser entendido como o pequeno bloco de dados de tamanho fixo obtido como saída de uma função MAC, assim como as etiquetas hash retornadas pelas funções hash.

O cenário mais comum para a utilização de autenticação de mensagem é quando duas pessoas desejam se comunicar por mensagens utilizando um canal inseguro. Neste caso, o emissor e o receptor querem a garantia que a mensagem transmitida é autêntica.

Um MAC baseado em funções hash pode ser descrito genericamente da seguinte forma. Uma mensagem m de qualquer tamanho é utilizada como entrada para uma função hash H, além da mensagem m, uma chave secreta k também é utilizada para compor a entrada da função. A função hash será aplicada à mensagem juntamente com a chave secreta k, esse processo resultará em uma etiqueta $t = MAC_k(m)$. A etiqueta MAC será enviado juntamente com a mensagem. A figura 2 ilustra o conceito geral de geração de MAC com base em função hash.

Figura 2: MAC baseado em função hash



Fonte: (KUROSE; ROSS, 2010)

A função hash $H(\cdot)$ recebe como entrada a mensagem e a chave secreta $(m+s)$, em que o sinal de $+$ representa uma operação de concatenação.

Para que o destinatário autentique a mensagem recebida, basta aplicar a mesma função hash H à mensagem juntamente com a chave secreta. Se a etiqueta hash obtida for exatamente a mesma recebida significa que a mensagem está intacta e a autoria da mensagem também esta comprovada, uma vez que apenas o legítimo remetente conhecia a chave secreta. Note que

a chave secreta precisa ser previamente compartilhada com o destinatário da mensagem. A diferença da etiqueta MAC para a etiqueta hash (resumo) fica por conta de que a etiqueta hash pode ser obtida simplesmente aplicando a mesma função hash à mensagem. Porém, para obter a etiqueta MAC é necessário a mensagem original e a chave secreta. Sendo assim, aplicando-se a função somente à mensagem resultará em uma etiqueta hash totalmente diferente.

3.2.1 Construção de MAC baseado em função hash - HMAC

Uma função de hash tradicional não pode ser utilizada diretamente como MAC, já que, além da mensagem, não recebe como entrada uma chave secreta. Porém, uma função de hash segura pode ser utilizada como núcleo do algoritmo de MAC. Esta seção descreve a estrutura do algoritmo HMAC.

O algoritmo HMAC (BELLARE *et al.*, 1996a) é um esquema de autenticação de mensagem construído a partir de funções hash criptográficas. HMAC pode ser usado com qualquer função de hash criptográfica iterativa, por exemplo, MD5 ou SHA-1, em combinação com uma chave secreta compartilhada.

De acordo com a RFC2104 (BELLARE *et al.*, 1997), a construção do HMAC tem os seguintes objetivos:

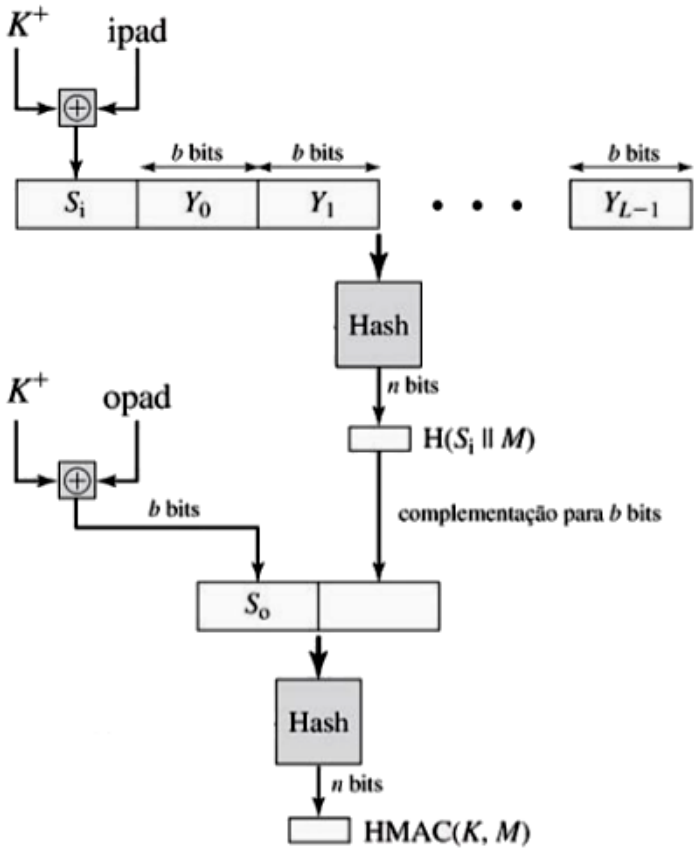
- Usar, sem modificações, as funções de hash disponíveis. Em particular, as funções de hash que têm um bom desempenho em software, e para o qual o código é livremente e amplamente disponível;
- Preservar a performance original da função hash sem incorrer em uma degradação significativa;
- Usar e gerenciar chaves de uma forma simples;
- Ter uma análise criptográfica bem compreendida da força do mecanismo de autenticação baseadas em suposições razoáveis sobre a função de hash utilizada internamente;
- Permitir facilmente a substituição da função hash embutida caso sejam encontrada funções de hash mais rápidas ou mais seguras.

Formalmente o HMAC pode ser expresso da seguinte forma:

$$HMAC(K,M) = H((K \oplus opad) \| H((K \oplus ipad) \| M)).$$

A estrutura do HMAC foi definida em (BELLARE *et al.*, 1996b). O algoritmo apresentado por Bellare *et al.* (1996b) mostra detalhes de sua operação. A tabela 3 define os termos utilizados no algoritmo.

Figura 3: Estrutura do HMAC



Fonte: (STALLINGS, 2008)

Tabela 3: Definição dos termos utilizados no algoritmo HMAC

M	Mensagem a ser autenticada
H	Função hash criptográfica
K	Chave secreta
K^+	Sequencia de bits formada pela chave K acrescentada de zeros à direita
$\|\|$	Denota concatenação
$\oplus$	Denota ou exclusivo (XOR)
b	Tamanho do bloco da função de compressão
$ipad$	Constante 00110110 (36 em hexadecimal) repetido $b/8$ vezes
$opad$	Constante 01011100 (5C em hexadecimal) repetido $b/8$ vezes
S_i	Bloco formando por K^+ XOR $ipad$
Y_i	i-ésimo bloco de M

Fonte: Adaptado de(STALLINGS, 2008)

Observe que na tabela 3, b se refere ao tamanho do bloco da função de compressão. Uma função hash iterativa quebra uma mensagem em blocos de tamanho fixo e utiliza cada um desses blocos como entrada para a função de compressão. O tamanho do bloco varia de acordo com a função de compressão utilizada (seção 3.1.2). A chave K será preenchida com zeros extras à direita até atingir o tamanho do bloco b, ou o hash da chave original se esta for maior que o tamanho do bloco.

Algoritmo 3 Alagorimo HMAC

Entrada: Mensagem m e chave secreta K
Saída: Código MAC HMAC(K,m)

1: Acrescentar zeros ao final do K para criar uma sequência de b bits K^+, em que b é o tamanho do bloco da função de compressão. (Por exemplo, se K tiver um comprimento de 160 bits e b = 512, então K será acrescido de 352 bits zeros).
2: Fazer XOR, de K^+ calculado na etapa (1) com $ipad$ formando o bloco de b bits S_i.
3: Anexar o fluxo de dado m com S_1 calculado na etapa (2).
4: Aplicar H para o fluxo gerado na etapa (3).
5: Fazer XOR de K^+ com $opad$ para produzir o bloco de b bits S_0.
6: Anexar o resultado do passo (4) a S_0.
7: Aplicar H para o fluxo gerado no passo (6) e apresente o resultado como saída.

3.2.2 Segurança do HMAC

A segurança do HMAC está diretamente relacionada com a segurança da função hash usada internamente. Um algoritmo de MAC utiliza a função hash como uma caixa preta, não há necessidade de modificar o código da função para implementar a algoritmo de MAC. Segundo Bellare *et al.* (1996b), se o HMAC falhar como um MAC seguro, é porque há fraqueza suficiente na função hash embutida que precisa ser descartada.

A segurança do MAC significa segurança contra falsificações. Um MAC é considerado quebrado se um atacante for capaz de encontrar alguma mensagem m juntamente com seu valor MAC correto, sem conhecer a chave K.

Em (BELLARE *et al.*, 1996a), é apresentado uma relação exata entre a força da função hash criptográfica e a força do MAC. Bellare mostra que se um atacante for capaz de falsificar uma função MAC, ele pode quebrar a função de hash usada internamente em uma das seguintes maneiras:

- O atacante é capaz de calcular uma saída da função de compressão mesmo quando *vetor de inicialização* é aleatório, secreto e desconhecido do atacante;
- O atacante encontra colisões na função hash mesmo quando o *vetor de inicialização* da função de compreensão é aleatório e secreto.

Se qualquer um dos ataques acima for bem sucedido, é porque as propriedades de segurança da função hash embutida não foram satisfeitas. No primeiro caso, a aplicação da função MAC, que requer uma chave secreta, é semelhante a função de compressão usada pela função hash, que requer um *vetor de inicialização* (seção 3.1.2). Se o atacante for capaz de falsificar uma função MAC, desconhecendo a chave, ele será capaz de falsificar a função de compreensão, mesmo que o *vetor de inicialização* seja aleatório. Para esse ataque seria necessário uma ataque de força bruta à chave.

No segundo caso o atacante procura encontrar m e m' tal que $H(m) = H(m')$. Esse é o ataque do aniversário, mostrado na seção 3.1.3.2 que exige um esforço computacional $O(2^{n/2})$ para uma função hash com saída de tamanho n. O ataque do aniversário, que é a base para encontrar colisões em funções hash criptográficas, pode ser aplicado para atacar também o esquemas HMAC. Mesmo assim, encontrar colisões pelo ataque do aniversario no HMAC é ainda mais difícil que encontrar colisões em uma função hash. Em uma função hash o atacante pode gerar $2^{n/2}$ operações e armazenar as mensagens testadas juntamente com as respectivas etiquetas hash encontradas. Em seguida observa se existe uma colisão.

No HMAC, o atacante não pode simplesmente gerar pares de mensagem/MAC, já que ele não conhece a chave K. Para que o ataque possa acontecer, é necessário que o atacante seja capaz de observar um certo número de pares de mensagem/MAC, em uma uma comunicação entre um emissor e o receptor. Observe que o atacante depende de um usuário legítimo que gere, no mínimo, $2^{n/2}$ pares mensagem/MAC utilizando a mesma chave. Somente com estas condições é que o ataque pode ser aplicado a essas mensagens.

Por exemplo, quando esse algoritmo de MAC utiliza uma função interna com saída de 128 bits, o atacante necessita observar 2^{64} blocos de mensagem/MAC gerados utilizando a mesma chave. Segundo (BELLARE *et al.*, 1996b), utilizando um link de 1 Gbps, seria necessário 250.000 anos para processar todos os dados necessário para que tal ataque ocorresse.

Observe que o ataque do aniversário aplicado ao HMAC é computacionalmente inviável com a tecnologia atual. É importante destacar que, segundo Bellare *et al.* (1996b), o ataque do aniversário ao HMAC é o ataque mais conhecido atualmente. Assim, se uma função hash bem projetada for utilizada pelo algoritmo HMAC, então todos os ataques a esse algoritmo não terão sucesso.

Sendo assim, para que o MAC seja considerado seguro, basta que a função hash tenha as propriedades de segurança necessárias, principalmente que seja livre de colisão. De acordo com Bellare *et al.* (1996b), a única maneira do HMAC falhar é se a função hash utilizada internamente falhar.

4 ASSINATURAS DIGITAIS

Substituir a assinatura escrita à mão não é uma tarefa fácil. É necessário um sistema que possa garantir que uma das partes escreva uma mensagem e a "assine", de modo que a outra parte tenha certeza quem é o autor da mensagem. Além disso, é preciso garantir que o documento não foi alterado desde a sua criação.

Com a popularização da Internet muitas transações comerciais e contratos estão sendo realizados por meio da rede. Em muitos casos, não existe mais o tradicional documento em papel e as assinaturas confirmando que os signatários concordam com o conteúdo do documento. Atualmente esses contratos são documentos eletrônicos, então é necessário uma forma de assinar esses documentos com as mesmas garantias que assinatura manual fornece.

A assinatura digital é um método criptográfico que permite que o criador de um documento digital seja identificado pelo receptor. Os esquemas de assinaturas digitais visam garantir a autenticidade de uma mensagem. Esses esquemas são construídos a partir de criptossistemas de chave pública.

A forma mais básica de gerar uma assinatura digital é fornecendo como entrada para um algoritmo criptográfico E uma mensagem M e a chave privada K_s do emissor da mensagem. Com isso ele pode produzir

$$C = E(K_s, M)$$

e enviar o texto cifrado C ao receptor.

Quando um algoritmo criptográfico faz uso da chave privada do remetente para cifrar uma mensagem, obrigatoriamente a chave pública do remetente deve ser utilizada para decifrar a mensagem. Essa é uma das principais características dos criptossistema de chave pública (capítulo 2, seção 2.2.5). Assim, o receptor pode obter a mensagem original

$$M = E(K_p, C).$$

Nesse esquema, qualquer pessoa de posse da chave pública do emissor pode decifrar a

mensagem. De fato, não há o objetivo de esconder a mensagem. O principal objetivo da assinatura digital é garantir a **autenticidade** e o **não repúdio**.

As assinaturas digitais dependem de duas suposições fundamentais: a primeira é que a chave privada é segura e que apenas o proprietário da chave tem acesso a ela. A segunda é que a única maneira para produzir uma assinatura digital é usar a chave privada (BURNETT *et al.*, 2001).

Suponha a seguinte situação: Bob recebe uma mensagem supostamente enviada por Alice, isto é, cifrada com a chave privada de Alice. Para confirmar a autenticidade da mensagem, Bob tenta decifrar a mensagem utilizando a chave pública de Alice. Se o processo for bem sucedido, então a mensagem realmente foi cifrada com a chave privada de Alice. Logo, constata-se que foi Alice quem enviou a mensagem, pois presume-se que apenas Alice tem acesso a sua chave privada. Considere que ambos escolheram previamente o algoritmo criptográfico a ser utilizado nas comunicações entre eles.

Esse é um cenário básico que apresenta algumas problemas. Um exemplo de problema nesse esquema é que a assinatura é a própria mensagem encriptada e será tão longa quanto a mensagem a ser assinada. Nas seções seguintes, esse e outros problemas serão resolvidos incorporando novos elementos ao processo de assinatura digital.

Como visto anteriormente, o processo de assinatura digital faz uso da criptografia de chave pública. O principal algoritmo utilizado nesse processo é o RSA, descrito a seguir.

4.1 Criptossistema RSA

Embora o trabalho apresentado por Diffie e Hellman (1976b) tenha uma relevância história, na data da publicação, os autores ainda não tinham um algoritmo criptográfico que, de fato, realizasse o processo de cifragem. Foi a partir da ideia seminal de Diffie e Hellman (1976b) que o professor do MIT, Ron Rivest, juntamente com Adi Shamir e Len Adleman começaram a trabalhar na implementação de um algoritmo que fosse capaz de criptografar a informação utilizando o conceito de chave pública. Dois anos depois, o trio publicou o algoritmo RSA (RIVEST *et al.*, 1978), assim chamado devido as iniciais dos nomes dos criadores. Deste a introdução do conceito de chave pública, alguns algoritmos surgiram, mas o RSA é o algoritmo mais utilizado até os dias atuais.

4.1.1 Geração das Chaves

Para fazer uso do RSA, cada usuário pode gerar suas próprias chaves, isto é, a chave pública e a chave privada. Todo o processo de geração de chaves e criptografia do RSA tem base na

aritmética modular.

A geração das chaves começa pela escolha de dois números primos aleatórios grandes p e q. A geração de números primos foi discutida no capítulo 2, seção 2.3.2. Com p e q escolhidos, calcula-se o produto $n = p \cdot q$. Os número primos p e q são fundamentais para a segurança do RSA, devem ter aproximadamente o mesmo tamanho, algo superior a 100 algarismos cada. Lembre-se que existe uma dificuldade histórica para fatorar números inteiros grandes em seus fatores primos. Isto significa que dado apenas n, em que n seja um número inteiro com mais de 300 algarismos, é computacionalmente difícil fatorar n. O problema da fatoração de inteiros foi discutida no capítulo 2, seção 2.3. O motivo pelo qual os primos precisam ser grandes e aproximadamente do mesmo tamanho é que existem alguns métodos de fatoração capazes de encontrar eficientemente os fatores primos pequenos de n.

O próximo passo é escolher um expoente denotado por e que seja primo relativo a $\phi(n)$, em que

$$\phi(n) = (p-1)(q-1)$$

é a função totiente de Euler. O processo continua calculando-se um inteiro d que satisfaça

$$e \cdot d \; mod \; \phi(n) = 1.$$

Isso é equivalente a dizer

$$d \equiv e^{-1} \; mod \phi(n).$$

Nesse momento os números p e q não são mais importantes. Na verdade eles devem ser descartados para evitar que vazem. Os números n, e e d serão suficientes para gerar as chaves pública e privada da seguinte forma:

- Chave pública : (e,n);
- Chave privada : (d,n);

O inteiro e é chamado de expoente de encriptação e o inteiro d é chamado de expoente de decriptação, enquanto o n é chamado de módulo.

Essas chaves são os elementos fundamentais para o processo de cifragem, decifragem ou de assinatura digital de uma mensagem. A chave privada, como o próprio nome sugere, deve ser guardada cuidadosamente para que não seja capturada por intrusos. A chave pública será divulgada diretamente pelo proprietário ou será armazenada em repositórios públicos. Na seção 4.5 será apresentada a ICP-Brasil, uma cadeia hierárquica que gerencia o uso de chaves e é responsável por emitir certificados digitais e divulgar as chaves públicas.

4.1.2 Criptografia e decriptografia com RSA

O RSA exige que a mensagem M a ser cifrada seja representada por números inteiros. Então, considera-se a cadeia de bits que representa a mensagem M e aplica-se uma divisão em blocos de k bits. De acordo com Stallings (2008), não há um tamanho padrão para o bloco, cada bloco deve ter um valor binários menor que o módulo RSA n. Em um exemplo simples, se o tamanho do bloco é k=8 e n=323, então o valor binário de cada bloco é no máximo 2^k = 2^8 = 255, portanto menor que 323.

Criptografia com RSA

Suponha que Alice queira cifrar uma mensagem para enviá-la a Bob. A mensagem M será dividida em blocos M_i de k bits, em que $i = \{1,2,...,j\}$. Para cifrar cada bloco é preciso representá-lo através de um número inteiro, neste caso será calculado qual inteiro corresponde a sequência de bits do bloco em questão.

O bloco cifrado será dados por

$$C_i = (M_i)^e \, mod \, n,$$

em que C_i é um bloco de texto cifrado, M_i é um bloco de texto claro, e é o expoente de encriptação da chave pública de Bob e n é o módulo RSA.

Como exemplo numérico, suponha que a chave pública seja **[119 , 5]** e a chave privada seja **[119 , 77]** e que o valor do bloco M_i seja "65", assim

$$C_i = 65^5 \, mod \, 119 = 46.$$

O texto claro "65" foi cifrado para "46". Na prática, o módulo RSA é consideravelmente maior que o n usado neste exemplo.

Decriptografia com RSA

Para decifrar a mensagem utiliza-se o mesmo algoritmo com a chave privada de Bob, assim o texto claro será dado por

$$M_i = (C_i)^d \, mod \, n.$$

Usando o mesmo exemplo anterior,

$$M_i = 46^{77} \, mod \, 119 = 65.$$

Obtém-se novamente o texto claro "65". Esta forma de utilizar o criptossistema oferece garantias de confidencialidade da mensagem. Observe que a única chave capaz de decifrar a mensagem cifrada com a chave pública de Bob, é a correspondente chave privada de Bob. Dessa forma, Bob é o único que tem acesso ao conteúdo da mensagem.

4.2 Assinaturas digitais com RSA

Na seção anterior, o RSA foi utilizado para cifrar e decifrar uma mensagem. Porém essa não é única maneira de usá-lo. É possível obter uma assinatura digital apenas invertendo a ordem do uso das chave. Nas transações eletrônicas, normalmente o processo de assinatura agrega outros elementos e assim é capaz de adicionar mais propriedades à assinatura. A seção 4.3 apresenta uma forma mais sofisticada de gerar assinaturas digitais.

Suponha que Bob deseja receber uma informação de Alice, mas quer a garantia que foi realmente Alice que a enviou. Isto significa que Alice não poderá repudiar a informação, ou seja, não poderá negar que é a autora da mensagem. Assim, Alice não utiliza a chave pública de Bob, Alice usa sua própria chave secreta.

O RSA somente será capaz de decifrar a mensagem utilizando a chave pública de Alice. Bob ou qualquer outra pessoa que possuir a chave pública pode decifrar a mensagem. Nitidamente não há confidencialidade, mas satisfez o que Bob queria, a garantia que Alice não pode negar a informação. Isto significa que se a mensagem foi decifrada com a chave pública de Alice, então a cifragem foi feita com a chave privada dela. Nesse criptossistema, presume-se que apenas Alice tenha acesso a sua chave privada.

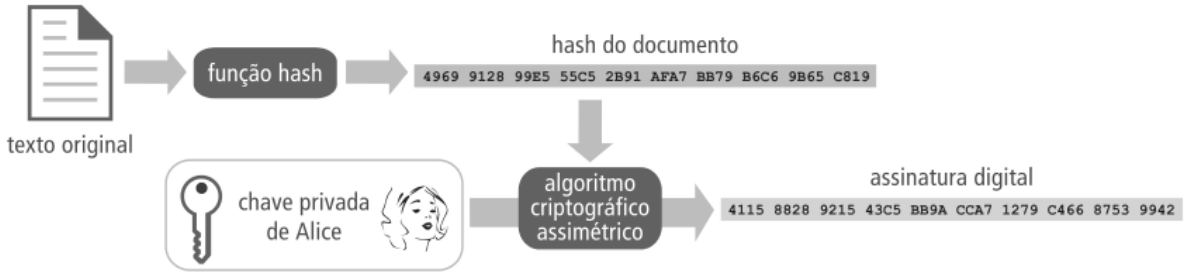
4.3 Assinaturas digitais com RSA e funções hash

A maneira de produzir uma assinatura digital apresentada na seção anterior tem alguns problemas importantes. A assinatura obtida é o próprio texto cifrado, isto significa que a assinatura é tão longa quanto o próprio texto. A cifragem e a decifragem no RSA exige uma exponenciação mod n para cada bloco cifrado ou decifrado. Isto significa que produzir uma assinatura para um documento muito grande pode gastar muito tempo.

Para resolver esse problema, pode-se gerar assinaturas digitais para documentos eletrônicos usando um elemento adicional: as funções hash. Como visto no capítulo 3, uma função hash é um método capaz de gerar uma etiqueta que forneça uma identificação unívoca para uma determinada mensagem de qualquer tamanho. Assim, é fácil perceber que para assinar um documento não há necessidade de criptografar todo o documento, já que neste caso, o objetivo não é tornar a mensagem confidencial. Para assinar tal documento basta cifrar a etiqueta hash que é a representação única do documento em questão (ver detalhes sobre colisão no capitulo 3). A vantagem da utilização de funções hash criptográficas no processo de assinatura digital é o aumento de desempenho, pois os algoritmos de criptografia assimétrica são muito lentos.

Dessa maneira a assinatura digital passa a ser a etiqueta hash que permite ao criador de um documento unir ao documento criado a etiqueta hash que funciona como autenticação.

Figura 4: Assinatura digital utilizando algoritmos de chave pública e função hash



Fonte: (Secretaria de Estado de Fazenda do Distrito Federal, 2012)

O cenário a seguir exemplifica a geração de uma assinatura digital utilizando função hash. Alice resolve enviar uma mensagem m para Bob, assim Alice aplica uma função hash H a mensagem m, este processo gera um hash $x = H(m)$. Com sua chave privada K_S, Alice cifra o hash x aplicando

$$RSA(K_S, x) = C,$$

anexa o hash cifrado C à mensagem e os transmite. Para garantir que o autor da mensagem é realmente quem diz ser e que a mensagem não foi alterada, o receptor decifra o hash com a chave pública K_P do remetente aplicando

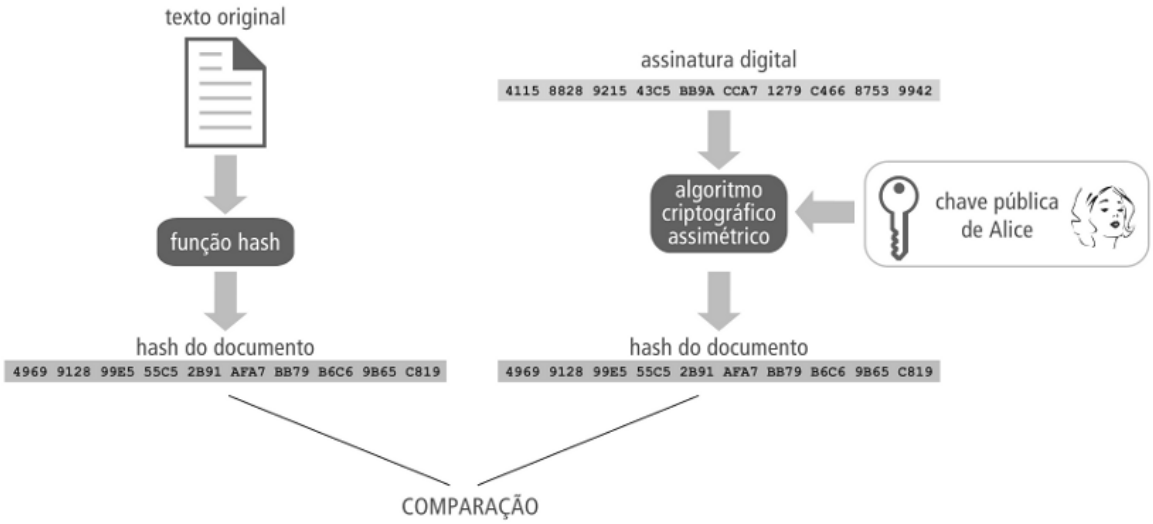
$$RSA(K_P, C) = x.$$

De posse do hash x, Bob aplica a mesma função hash $H(m)$ à mensagem m, se o hash obtido for igual ao hash enviando pelo remetente significa que a mensagem não foi alterada e o autor é realmente quem diz ser.

Observe que esse processo é capaz de garantir as seguintes propriedades:

- **Integridade**: permite verificar se o documento não foi alterado após a aplicação da assinatura, comparando a etiqueta hash enviada pelo emissor com a etiqueta hash da mensagem gerada pelo próprio receptor;
- **Autenticidade**: permite verificar se um documento realmente foi gerado e aprovado pelo emissor;
- **Não repúdio**: quando constatada a autenticidade da mensagem, impede que o emissor recuse a autoria ou conhecimento do documento.

Figura 5: Verificação da assinatura digital utilizando função hash



Fonte: (Secretaria de Estado de Fazenda do Distrito Federal, 2012)

A função hash utilizada permite verificar a integridade da mensagem. Caso a mensagem seja interceptada e modificada o receptor será capaz de detectar tal adulteração. A mensagem pode até ser lida, mas o atacante não será capaz de gerar um novo hash correspondente a mensagem alterada e reenviá-lo ao destinatário. Pois para isso, ele teria que enviar a mensagem alterada e o novo hash cifrado com a chave privada do emissor. Isto não será possível, já que o atacante não tem acesso a tal chave. Caso o atacante pense em cifrar o novo hash com sua própria chave secreta, isso facilmente será detectado, pois o receptar não será capaz de decifrar, com a chave pública do verdadeiro emissor, um hash cifrado com a chave privada do atacante.

Em resumo, a garantia de integridade é feita com base em uma característica das funções hash, que se um único bit for alterado uma nova etiqueta hash será gerada. A garantida de autenticidade é feita com base no conceito de chave pública. Se Bob conseguiu decifrar a mensagem de Alice utilizando a chave pública dela, é porque Alice cifrou a mensagem com sua chave privada, a qual somente ela conhece, portanto a mensagem é autêntica.

Todo o processo de assinatura digital começa com geração das chaves pública e privada. Os detalhes da geração de chaves foram discutidos na seção 4.1.1, porém há um problema com esse processo. Como garantir que a chave pública de um indivíduo foi realmente gerada por ele? Por exemplo, um impostor pode ter gerado um par de chaves e apresentá-lo como sendo de Alice para receber as mensagens endereçadas à Alice. Esse problema foi resolvido com a criação de um documento chamado Certificado Digital.

4.4 Certificado Digital

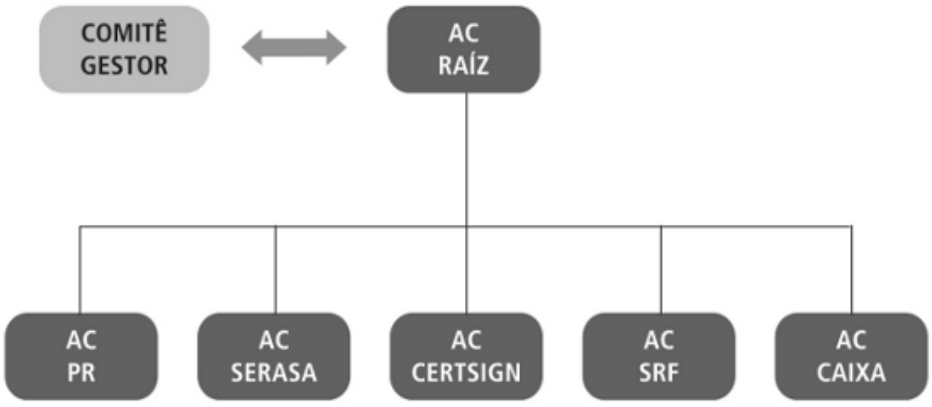
É um documento digital contendo dados do indivíduo ou entidade, como nome, CPF ou CNPJ e a chave pública. O certificado digital associa a chave pública a uma pessoa física ou jurídica e serve como meio de divulgação da chave pública. Assim como qualquer outro documento de identificação, como por exemplo carteira de identidade, o certificado digital identifica e garante que o proprietário do certificado é quem realmente diz ser nas operações eletrônicas por meio da rede.

Mesmo com o certificado digital, os problemas de identificação ainda não estão totalmente resolvidos. A questão agora é como saber se um certificado é válido. Os certificados são emitidos por orgão do governo de um país. Normalmente existe um conjunto de leis que atribuem poderes para que tais orgãos possam emitir e gerenciar os certificados e que sejam considerados legalmente válidos. Para oferecer essa garantia, o certificado digital é emitido e assinado digitalmente por uma instituição confiável denominada autoridade certificadora. A autoridade certificadora é um orgão do governo que, por foça de lei, tem o poder de atestar a identidade de uma entidade que deseja realizar transações eletrônica em sistemas computacionais. A autoridade certificadora coloca seus dados no certificado juntamente com os dados do titular e os assina com sua chave privada. Para conferir a autenticidade do certificado, basta conferir a assinatura do mesmo com a chave pública da autoridade certificadora.

Na próxima seção será apresentada a infraestrutura necessária para que assinaturas e certificados digitais sejam usados em transações de negócios por meio da Internet.

4.5 Assinaturas digitais na ICP Brasil

Figura 6: Principais componentes da ICP Brasil



Fonte: (Secretaria de Estado de Fazenda do Distrito Federal, 2012)

Infraestrutura de Chave Pública (ICP) é um conjunto de padrões envolvendo várias entidades e componentes, tais como, autoridade certificadora, autoridade de registro, certificados digitais entre outros para viabilizar o uso de chaves públicas.

Autoridade Cerificadora - AC

AC é uma entidade de confiança que emite certificados digitais para pessoas físicas, empresas ou outras AC. No topo desta cadeia de certificação esta a AC-Raiz, no Brasil é representada pelo ITI (Instituto Nacional de Tecnologia da Informação). É competência da AC-Raiz emitir, distribuir, revogar e gerenciar certificados das AC imediatamente subsequentes e manter uma lista de certificados revogados. Além disso, fiscaliza e audita autoridades certificadoras e autoridades de registros e demais membros participantes da cadeia de certificação.

Autoridade de Registro - AR

AR é uma entidade de apoio ligada a uma autoridade certificadora habilitada junto AC-Raiz. Cada AC escolhe quantas AR prestarão serviço a ela. A AC delega a função de identificação do interessado no certificado digital. A AR tem a responsabilidade de garantir a veracidade das informações fornecidas pelo interessado no certificado. A AR envia estas informações para a AC para que gere, assine digitalmente e emita o certificado digital.

4.6 Certificados X.509

X.509 (CHOKHANI *et al.*, 2003) é um padrão para certificados digitais criado em 1988 pela ITU-T (Telecommunication Standardization Sector - Setor de Normalização das Telecomunicações). Atualmente o padrão X.509 está na versão 3 lançada em 2003. Esse padrão surgiu da necessidade de todos os certificados utilizados em transações na rede estarem no mesmo formato.

4.7 Segurança do RSA

Toda a segurança do RSA depende fundamentalmente da dificuldade de calcular o expoente d, mesmo conhecendo o expoente e, que faz parte da chave pública e o módulo n, que é comum nas duas chaves. Relembre que a chave pública é constituída pelo par (n,e) e a chave privada é constituída pelo par (d,n). Observe que o único componente que não é público nessas chaves é o expoente d que pode ser calculado

$$d \equiv e^{-1} \, mod \, \phi(n),$$

usando o algoritmo euclidiano estendido. Evidentemente, para encontrar d, antes é necessário calcular

$$\phi(n) = (p-1)(q-1). \tag{4.1}$$

Este é o momento em que é possível perceber claramente que a escolha adequada dos números primos p e q contribui de forma significativa para a segurança do RSA e consequentemente para qualquer esquema de assinatura que esteja utilizando esse algoritmo internamente. Os inteiros p e q são fatores primos do módulo n. Aliás, n da função 4.1 é apenas o produto de $p \cdot q$. Considerando que a decifragem de uma mensagem é feita pela operação

$$M = C^d \, mod \, n,$$

então, aparentemente, a maneira mais trivial de atacar o RSA é tentando descobrir o expoente d. Porém, descobrir d, um número compostos por centenas e algarismos, não é uma tarefa tão fácil assim. Para descobrir d é necessário encontrar os números primos p e q. Como p e q foram escolhidos aleatoriamente durante a geração das chaves e possivelmente foram descartados logo em seguida, então a única maneira de encontrar tais elementos é tentando fatorar n. Isto leva a um problema clássico da matemática que tem despertado o interesse de estudiosos de várias épocas. Esse problema é conhecido como **o problema da fatoração de inteiros** (capítulo 2, seção 2.3).

Diversos métodos foram propostos ao longo dos anos (ver capítulo 2, seção 2.3.3), mas nenhum deles ainda é capaz de fatorar eficientemente um inteiro de mais de 300 algarismos, considerando o poder computacional atual.

O RSA foi projetado para sobreviver a evolução do poder computacional, ou pelo menos sobreviver por várias décadas. Quando o tamanho do módulo RSA n se tornar inseguro devido ao aumento da velocidade de processamento dos computadores clássicos, basta aumentar o tamanho dos primos p e q que geram n. Neste momento há um questionamento inevitável, é possível encontrar tantos números primos grandes para gerar as chaves? O matemático Euclides de Alexandria respondeu está pergunta ainda no século III a.C, em sua obra clássica *Elementos*: há infinitos números primos.

Terada (2008) observa que, até hoje, os pesquisadores não descobriram qualquer outro ponto fraco no RSA. As outras formas para recalcular a chave secreta que já foram propostas, são tão ou mais difíceis que a fatoração do módulo n.

5 RESGATE DA AUTORIA DE UMA ASSINATURA EM ANEL

Este capítulo apresenta um esquema de resgate de autoria de uma assinatura em anel (TOMAZ *et al.*, 2012). Aqui é apresentada uma nova propriedade para compor o esquema de assinatura em anel, proposto por Rivest *et al.* (2001), sem alteração nas propriedades existentes. A seção 5.1 apresenta uma introdução ao esquema de assinatura em anel e suas principais características. A seção 5.2 descreve um cenário detalhado em que o esquema de assinatura, já com a nova propriedade de resgate de autoria, poderia ser utilizado.

Os detalhes para implementar a propriedade de resgate de autoria, elemento principal deste trabalho, estão descritos nas seções 5.3 e 5.4. Essas duas seções compõe especificamente a proposta deste trabalho e mostram as ferramentas utilizadas e os protocolos necessários para acrescentar a propriedade em questão. Em seguida, a seção 5.5 estabelece semelhanças e diferenças em relação a outros trabalhos. Finalmente, a seção 5.6 identifica os elementos responsáveis por tornar este esquema confiável.

5.1 O conceito de assinatura em anel

Em 1991, Chaum e Heyst (1991) apresentaram um tipo de esquema de assinatura chamado de *assinatura em grupo*. As assinaturas em grupo são esquemas que permitem que um membro do grupo assine uma mensagem em nome do grupo sem revelar sua identidade, de tal modo que fique evidente que a assinatura foi produzida por um dos membro do tal grupo. Esse esquema define três propriedades.

1. Somente membros do grupos podem assinar mensagens.
2. Os receptores da assinatura podem verificar que tal assinatura pertence aquele grupo, mas não podem descobrir qual membro do grupo realmente produziu a assinatura.
3. Em caso de disputas posteriores, a assinatura pode ser “aberta”, com ou sem a ajuda dos

membros do grupos para revelar a identidade do verdadeiro assinante.

Os esquemas de assinatura em anel foram propostos com base nos esquemas de assinaturas em grupo, porém apresentam suas próprias características descritas a seguir.

A noção de assinatura em anel foi introduzida em (RIVEST *et al.*, 2001). Um esquema de assinatura em anel permite que um membro de um grupo divulgue uma mensagem anonimamente, de tal forma que cada um dos membros do grupo seja considerado o possível autor da mensagem. Os autores dessa ideia estabeleceram propriedades importantes que esses esquemas devem satisfazer.

1. Cada membro do grupo precisa estar associado publicamente a uma chave pública por meio de uma Autoridade Certificadora (ver capítulo 4, seção 4.5). Se, por exemplo, todos os membros desse grupo pertencem a uma mesma instituição, então isso garante a autenticidade e a credibilidade da fonte da informação.

2. O grupo não precisa ser organizado ou ter uma formação premeditada com um número exato de participantes.

3. Os demais membros do grupo não devem ser informados que suas chaves públicas serão usadas neste esquema.

4. O verificador da mensagem deve ser incapaz de dizer qual membro de um grupo de possíveis assinantes realmente produziu a assinatura.

A ideia principal de uma assinatura em anel é garantir o anonimato de quem divulgou e ainda garantir a autenticidade da informação, mostrando que a mensagem partiu de um dos membros do referido grupo.

Assinaturas em anel são semelhantes à noção de assinaturas em grupos introduzidas por Chaum e Heyst (1991). Nos esquemas de assinatura em grupo, existe um gerente que predefine certos grupos de usuários e distribui as chaves especialmente geradas para cada um deles. Cada membro pode, individualmente, usar sua chave privada e as chaves públicas dos demais membros para assinar anonimamente mensagens em nome de seu grupo. As assinaturas produzidas por diferentes membros do grupo parecem indistinguíveis para os seus verificadores, mas não para o gerente do grupo que pode revogar o anonimato do assinante. Nas assinaturas em anel, não tem gerente de grupo, não há configuração prévia e não há colaboração entre os membros do grupo. Qualquer membro pode assinar em nome de qualquer conjunto de indivíduos. Além disso, ele pode escolher um novo grupo a cada mensagem divulgada sem precisar obter o consentimento dos outros membros.

Nos esquema de assinatura em anel, normalmente é adotada a seguinte terminologia:

- *Anel*: conjunto de possíveis assinantes;
- *Assinante*: membro do anel que produz a assinatura em questão. É o delator;
- *Não assinante*: cada um dos outros membros que não produziu a assinatura da mensagem, porém teve sua chave pública participando do esquema;
- *Verificador*: indivíduo que confere a autenticidade da assinatura.
- A_s: representa o membro que produz a assinatura. Representa o delator;
- A_i, $i \neq s$: representa os membros do grupo, isto é, os membros não assinantes;
- S_s: chave privada do assinante usada para produz a assinatura;
- P_i: chave pública do i-ésimo membro do anel;
- g_{P_i}: função de criptografia assimétrica utilizando a chave pública do i-ésimo membro do anel;
- g_{S_s}: função de criptografia assimétrica utilizando a chave privada do assinante;

Uma assinatura em anel consiste em dois procedimentos principais:

- **geração da assinatura** (m, A). Dada uma mensagem m a ser assinada e um conjunto de chaves públicas $A = \{P_1, P_2, P_3 ... P_n\}$, o assinante A_s pode produzir uma assinatura em anel σ usando A e sua própria chave secreta S_s.
- **verificação da assinatura** (m, σ). Dada uma mensagem m e uma assinatura σ que inclui $A = \{P_1, P_2, P_3, ..., P_n\}$, então um verificador pode determinar se σ é uma assinatura em anel válida para m, gerada por um dos membro do grupo.

5.2 Definição do problema

Considere que os assessores de um chefe de estado usam criptografia de chave pública para garantir uma comunicação segura entre eles na rede. Portanto cada um deles possui uma chave pública P_i e uma chave privada S_i conforme o padrão criptográfico assimétrico. Cada assessor será representado por $i = \{1, 2, ..., n\}$. Cada membro desse grupo deve possuir um certificado digital adquirido conforme padrão internacional e emitido pela autoridade certificadora da ICP (Infraestrutura de chave pública) do país que habita (ver capítulo 4, seção 4.5). O certificado digital é o documento que permite a associação de um indivíduo com sua chave pública.

Agora, imagine a seguinte problemática: o atual presidente ou ditador desse país realiza pesquisas nucleares com intenções bélicas. Os assessores sabem dessa informação e um entre os *n* assessores deseja tornar pública esta informação secreta. A publicação dessa informação pode levar a uma perda de popularidade num regime democrático ou a sanções comerciais internacionais no caso de um regime ditatorial. Ao revelar esta informação, o delator provoca um conflito no governo do chefe desta nação.

Diante dessa situação, o membro delator estuda como revelar esta informação para o mundo sem ser identificado. O delator considera que a divulgação desta informação pode gerar dois possíveis cenários a seguir:

1. O chefe de estado, apesar de toda a polêmica durante o seu governo, consegue permanecer no poder. Nesse caso, o delator deseja preservar seu anonimato, pois se ele for descoberto poderá sofrer graves punições.

2. O chefe de estado não consegue se manter no poder por conta da denúncia e fica provado que realmente existiam armas nucleares sendo construídas em segredo. Devido a esta importante ação do delator em nome da paz mundial, ele poderá ser reconhecido como personalidade internacional e certamente receberá homenagens. Então, após a perda do poder desse chefe de estado, o delator poderá se revelar à comunidade internacional como o autor da assinatura.

Um método capaz de satisfazer as características do cenário 1, foi proposto em (RIVEST *et al.*, 2001). Na verdade, o esquema proposto por Rivest *et al.* (2001) é o clássico esquema de assinatura em anel, em que o delator permanece anônimo após gerar uma assinatura que possui total credibilidade devido aos membros que compõem o anel. No problema em questão, os membros do grupo de assessores do chefe de estado seriam todos devidamente identificados pelas suas chaves públicas. Todos seriam suspeitos, mas nenhum pode ser acusado de ter gerado tal assinatura. Porém, tal método não satisfaz as características do cenário 2. Isto significa, que o delator garante seu anonimato, mas jamais será capaz de provar que é o autor da divulgação das informações.

Diante dessa situação, este trabalho apresenta um esquema de assinatura em anel baseado no esquema de Rivest *et al.* (2001), em que o assinante pode, mais tarde, revogar seu anonimato apresentando valores secretos que provam que somente ele seria capaz de gerar tal assinatura. Essa propriedade representa uma expansão do conceito original de assinatura em anel e será chamada aqui de *resgate de autoria*. A propriedade de revogar o anonimato não é exatamente nova, diversos trabalho já forma publicados com essa proposta, no entanto, essa propriedade recebeu diferentes nomes. A seção 5.5 apresenta os principais trabalhos que sugerem alguma forma de adicionar a propriedade de resgatar a autoria. Alguns deles alteram em algum ponto o

protocolo original, outros mantém a ideia original. Os trabalhos relacionados serão discutidos com mais detalhes na seção 5.5. Será uma maneira de apresentar as principais diferenças em relação a este trabalho. Um trabalho importante publicado por Dong *et al.* (2012) será discutido com mais detalhes por ter semelhança na forma como a ideia foi proposta.

5.3 Ferramentas utilizadas

Este trabalho propõe uma maneira de resgatar a autoria da assinatura em anel com base na ideia apresentada em (FILHO; NASCIMENTO, 2011). Filho e Nascimento (2011) propuseram o resgate de autoria em esquemas de assinatura em anel usando o conceito de dispositivos quânticos a prova de falsificação (BOUDA *et al.*, 2008). Tal proposta necessita da existência de uma terceira parte confiável que compartilham partículas quânticas emaranhadas com a máquina do delator. Para implementar tal proposta é necessário uma tecnologia ainda não factível atualmente (memórias quânticas que armazenam dados por longo tempo).

Nesta dissertação, as técnicas utilizadas são baseadas em (TOMAZ *et al.*, 2012) e preservam o protocolo original de Rivest *et al.* (2001). Assim como na proposta de Filho e Nascimento (2011), a principal diferença em relação ao trabalho de Rivest *et al.* (2001) é apresentada antes mesmo de começar a geração da assinatura (geração e armazenamento dos estados de Bell). Nesta dissertação, os valores utilizados como entrada para a função trapdoor serão MACs gerados pelo algoritmo HMAC, um algoritmo de autenticação de mensagem baseado em função hash resistente à colisão. Essa modificação simples permitirá que, no futuro, o assinante revele-se como o verdadeiro autor da mensagem. Ele terá a garantia que a fonte da informação poderá ser autenticada devido a técnica de autenticação de mensagem amplamente conhecida e utilizada em operações computacionais.

5.3.1 Código de autenticação de mensagem - MAC

Para substituir as máquinas quânticas à prova de falsificação utilizadas em (FILHO; NASCIMENTO, 2011), propõe-se usar os códigos de autenticação de mensagens - MACs apresentados no capítulo 3. Optou-se em usar esse tipo de protocolo porque os MACs são tecnologia amplamente utilizada para autenticação de informações, além de serem capazes de detectar integridade de mensagens com base em funções hash. Os MACs são os elementos fundamentais deste esquema, pois será por meio deles que o delator poderá provar a autenticidade da mensagem divulgada.

5.3.2 Função trapdoor

O esquema de assinatura em anel original proposto por Rivest *et al.* (2001) utiliza o conceito de funções trapdoor one-way. Uma função trapdoor one-way é uma função unidirecional $f : X \to Y$ com uma propriedade adicional que, dada alguma informação extra (chamada de informações secreta ou trapdoor) torna-se possível encontrar, para qualquer $y \in Y$, um $x \in X$ tal que $f(x) = y$ (MENEZES *et al.*, 1996). Isto significa que com o segredo, é fácil calcular a inversa da função. Neste caso, a função trapdoor one-way adotada para este esquema é o RSA apresentado no capítulo 2, que será capaz de cifrar e decifrar uma mensagem *m* em tempo polinomial.

5.3.3 Cifra simétrica

Tão importante quanto a criptografia assimétrica é o conceito de cifras simétricas. Os principais esquemas de criptografia simétrica (capítulo 2, seção 2.2.4) atualmente são construídos com base em cifra de blocos.

De acordo com Knudsen e Robshaw (2011), uma cifra de bloco cifra um bloco de texto claro M em um bloco de texto cifrado C sob a ação de uma chave secreta z. Isso normalmente é denotado como

$$C = E_z(M).$$

A forma exata da transformação criptográfica será determinada pela escolha da cifra do bloco e o valor da chave z. O processo de decifragem usa a mesma chave fornecida pelo usuário. Esse processo normalmente é denotado por

$$M = D_z(C).$$

Formalmente uma cifra de bloco é uma função $F : \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$, onde para cada chave $z \in \{0,1\}^t$, a função $F_z : \{0,1\}^n \to \{0,1\}^n$ dada por $F_z(x)$ é uma permutação[1]. Nesta definição, $\{0,1\}^t$ deve ser entendido como o espaço de chave e os elementos de $\{0,1\}^n$ devem ser entendido como os blocos de texto claro e texto cifrado. A função F_z funciona como uma função de encriptação (PARDO, 2013). Note que $\{0,1\}^n$ é o conjunto de todas as sequências possíveis de tamanho n a partir do alfabeto $\{0,1\}$.

Na criptografia moderna, uma cifra de bloco geralmente é modelada como uma permutação pseudoaleatória. O motivo pelo qual é usado uma permutação pseudoaleatória é evitar criar um sistema determinístico, desse modo cria-se um sistema resistente a ataques de múltiplos textos cifrados conhecidos. Em outras palavras, em vez de usar uma única função para cifrar mensagens, muda-se a função toda vez que a o processo de cifragem for executado. Assim, se

[1] Uma permutação de um conjunto X é uma função bijetora $f : X \to X$.

$F : \{0,1\}^t \times \{0,1\}^n \rightarrow \{0,1\}^n$ é uma cifra de bloco, então para toda chave $z \in \{0,1\}^t$ a função F_z é uma permutação sobre $\{0,1\}^n$. Além disso, dado z é possível calcular $F_z{}^{-1}$ eficientemente.

Para o esquema de assinatura em anel apresentado neste trabalho, assim como em (RIVEST *et al.*, 2001), assume-se a existência de um algoritmo de cifragem simétrico definido publicamente que para qualquer chave z, a função E_z seja uma permutação sobre os b bits de uma string.

5.3.4 Função de Composição

Rivest *et al.* (2001), propuseram uma função de composição chaveada

$$C_{(z,v)}(y_1, y_2, y_3, ..., y_n), \tag{5.1}$$

que recebe como entrada uma chave z, uma valor de inicialização v e uma lista de valores arbitrários $y_1, y_2, y_3, ..., y_n \in \{0,1\}^l$, onde l é o tamanho de cada y_i. Esta função produz uma valor $q \in \{0,1\}^l$, tal que para qualquer (z,v) e qualquer valores fixos de $y_i, i \neq s$, $C_{(z,v)}$ é uma permutação sobre $\{0,1\}^l$. Rivest *et al.* (2001) definem a função $C_{(z,v)}$ baseada em um esquema de criptografia simétrico E modelada como uma permutação pseudoaleatória com uma chave z.

$$C_{(z,v)}(y_1, ..., y_n) = E_z(y_n \oplus E_z(y_{(n-1)} \oplus E_z(... \oplus E_z(y_1 \oplus v)))). \tag{5.2}$$

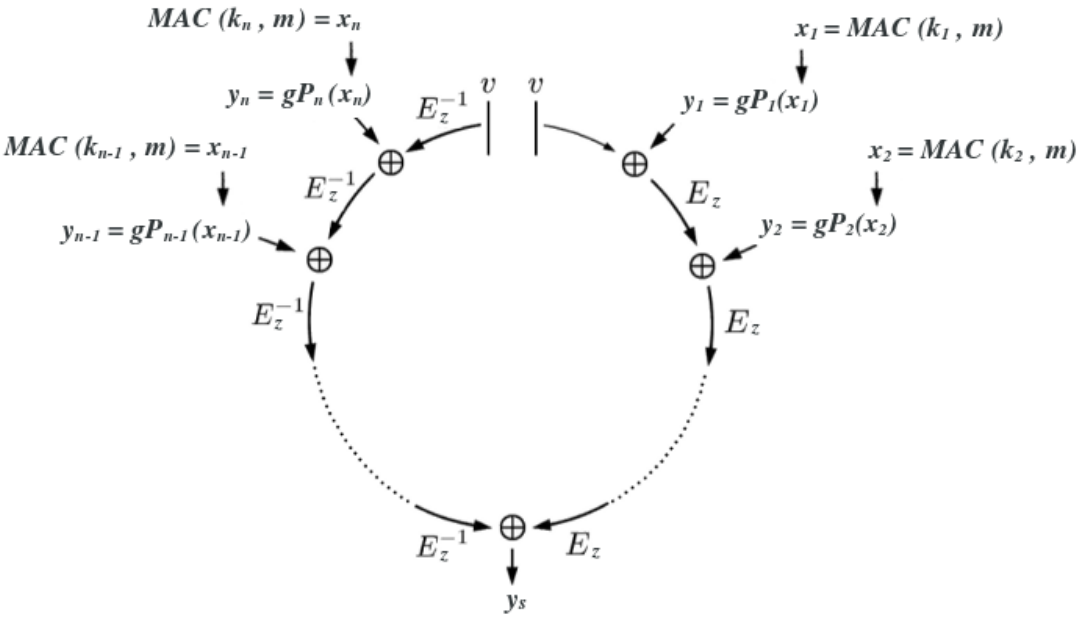
5.4 Assinatura em anel com resgate de autoria usando códigos de autenticação de mensagem

Neste trabalho propõe-se a construção de um esquema de assinatura em anel com resgate de autoria usando apenas técnicas computacionais disponíveis atualmente. Será apresentado um protocolo que gera a assinatura mantendo o anonimato do delator, mas permite que no futuro ele possa revelar a autoria da sua assinatura quebrando a propriedade de anonimato.

5.4.1 Geração da assinatura

Dada uma mensagem m a ser assinada, a chave privada S_s do assinante e um conjunto de chave públicas $A = \{P_1, P_2, ..., P_n\}$. O assinante pode gerar a assinatura conforme protocolo 1.

Figura 7: Construção da assinatura em anel



Fonte: adaptado de (RIVEST *et al.*, 2001)

Protocolo 1. *Geração da assinatura*

1. *Para iniciar a geração da assinatura, será necessário que o assinante selecione, aleatoriamente, um vetor de inicialização* $v \in \{0,1\}^l$*, em que* l *é o número de bits de* v*.*

2. *O assinante seleciona aleatoriamente* $n-1$ *valores* k_i *para serem usados como chave secreta no algoritmo HMAC. Para cada membro não assinante do anel será usado um valor secreto* k_i*, em que* $1 \leq i \leq n$ *e* $i \neq s$*.*

3. *O assinante calcula os MACs,* $x_i = HMAC_{k_i}(m)$*, em que* $x_i \in \{x_1, x_2, ..., x_n\}$*,* $1 \leq i \leq n$ *e* $i \neq s$*. O assinante é o s-ésimo membro e o seu valor* x_s *não será calculado como um MAC.*

4. *O assinante calcula um valor* $z = H(m)$ *que é o hash da mensagem m. O valor* $z \in \{0,1\}^b$ *será usado como chave para selecionar uma permutação* E_z*.*

5. *Calcula-se os valores* y_i*, exceto* y_s*, aplicando-se a função trapdoor com a chave pública de cada membro do anel*

$$g_{P_i}(x_i) = y_i. \tag{5.3}$$

6. *Para fechar a assinatura em anel, o delator precisa encontrar o valor de* y_s *que satisfaça a equação*

$$C_{(z,v)}(y_1, y_2, y_3, ..., y_n) = q, \tag{5.4}$$

onde o valor q produzido por $C_{(z,v)}$ deve ser igual ao vetor de inicialização v. O valor y_s pode ser calculada da seguinte forma:

$$\alpha = E_z(y_{s-1} \oplus ...E_z(y2 \oplus E_z(y_1 \oplus v))) \tag{5.5}$$

$$\beta = {E_z}^{-1}(y_{s+1}... \oplus {E_z}^{-1}(y_n \oplus {E_z}^{-1}(v))) \tag{5.6}$$

$$y_s = \alpha \oplus \beta. \tag{5.7}$$

7. *Encontrado y_s, então o assinante usa a chave secreta dele S_s para calcular o valor*

$$x_s = g_{S_s}(y_s) \tag{5.8}$$

8. *Assim, o assinante pode assinar a mensagem m com a tupla*

$$(P_1, P_2, ..., P_n; v; x_1, x_2, ..., x_n) \tag{5.9}$$

Observe na figura 7 que a assinatura será composta por várias iterações, uma para cada membro do anel. Cada iteração será composta pela escolha da chave k_i, o cálculo do MAC x_i e a cifragem de x_i com a chave pública P_i do membro não assinante resultando em

$$y_i = g_{P_i}(x_i). \tag{5.10}$$

Na etapa 4 do protocolo, a função E_z será utilizada para cifrar o resultado de uma operação XOR entre y_i e o resultado da função E_z da iteração anterior. Exceto na primeira iteração em que será utilizado o valor de v para compor a operação XOR com o y_1. Observe que os resultado destas iterações são denotados por

$$v = E_z(y_n \oplus E_z(y_{(n-1)} \oplus E_z(... \oplus E_z(y_1 \oplus v)))).$$

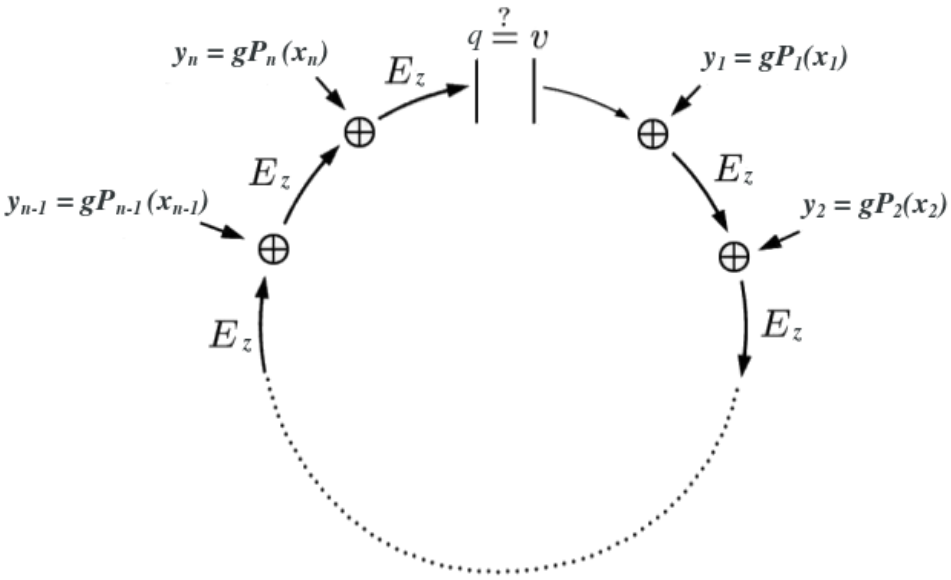
Na etapa 6 do protocolo, assume-se que, dado os valores de $y_i = g_{P_i}(x_i)$, em que $x_i \in \{x_1, ..., x_{s-1}, x_{s+1}, ..., x_n\}$ e um valor inicial v de b bits, existe um único valor para y_s que satisfaça a equação 5.4, que pode ser calculado de forma eficiente. Observe na figura 7 que o assinante A_s escolhe aleatoriamente sua posição no anel. Assim o valor α na equação 5.5 é computado no sentido horário até a posição de A_{s-1} e o valor β na equação 5.6 é computado no sentido anti-horário até a posição de A_{s+1}.

5.4.2 Verificação da assinatura

A assinatura $(P_1, P_2, ..., P_n; v; x_1, x_2, ..., x_n)$ pode ser verificada para dois casos diferentes. No primeiro caso, a verificação é feita de forma pública, onde qualquer pessoa com os dados da assinatura em anel pode verificar a autenticidade da informação. No entanto, ninguém sabe

quem gerou tal assinatura, sendo assim, o assinante permanece anônimo. No segundo caso, o autor da assinatura pode sair do anonimato apresentando uma prova que está relacionada com a geração da assinatura. De forma que todos ficam convencidos de que ele gerou a assinatura. Que no caso deste trabalho, serão apresentadas as chaves k_i.

Figura 8: Verificação da assinatura em anel



Fonte: adaptado de (RIVEST *et al.*, 2001)

Protocolo 2. *Verificação da assinatura*

1. *Para $i = 1,2,...,n$ o verificador calcula $g_{P_i}(x_i) = y_i$.*
2. *O verificador calcula $H(m) = z$ para obter a chave para E.*
3. *Finalmente o verificador confere se todo $y_i = g_{P_i}(x_i)$ satisfaz a equação*

$$C_{(z,v)}(y_1, y_2, ..., y_n) = v. \tag{5.11}$$

4. *Se a equação 5.11 for satisfeita, então o verificador considera a assinatura válida, caso contrário, o verificador rejeita a assinatura.*

Deve-se notar que a assinatura contém n chaves públicas que identificam n assessores do chefe de estado. Suponha que o assinante queira divulgar a mensagem para um jornal, numa correta abordagem, o assinante envia a mensagem juntamente com a assinatura $(P_1, P_2, ..., P_n; v; x_1, x_2, ..., x_n)$ e o nome de cada membro assessor, incluindo o seu próprio nome. O jornalista pode verificar a assinatura conferindo se os valores $y_i = g_{P_i}(x_i)$ satisfazem a função 5.11, e constatar se a assinatura, de fato, foi feita por um dos assessores.

Ainda com base no problema definido na seção 5.2, se o chefe de estado se manter no poder, então o assinante não se manifesta para preservar-se de ataques por parte do governo. Caso contrário, o chefe de estado perdendo o seu mandato e a situação política se invertendo, então o assinante poder revogar seu anonimato e provar que foi ele quem divulgou tais informações confidenciais.

5.4.3 Resgate de autoria

Para provar que é o autor da assinatura em anel, o assinante afirma que possui informações secretas $k_i = \{k_1, k_2, ..., k_{(s-1)}, k_{(s+1)}, ..., k_n\}$ capazes de produzir os respectivos valores $x_i = \{x_1, x_2, ..., x_{(s-1)}, x_{(s+1)}, ..., x_n\}$. Observe que x_s não foi gerado da mesma forma que x_i. Para convencer o verificador, o assinante revela os valores k_i e sugere ao verificador que execute os passos que constam no protocolo 3.

Protocolo 3. *Resgate da autoria*

1. *Para* $1 \leq i \leq n$ *e* $i \neq s$*, calcular o MAC*

$$x_i = HMAC_{k_i}(m).$$

2. *O assinante declara que é s-ésimo membro do anel e que* x_s *é o resultado de*

$$x_s = g_{S_s}(y_s),$$

   em que y_s *foi calculado conforme equação 5.7.*

3. *O verificador confere se os valores* $x_i \in (P_1, P_2, ..., P_n; v; x_1, x_2, ..., x_n)$*. Se o resultado for verdadeiro, então o verificador pode identificar o real assinante.*

Caso exista alguma premiação para o delator, provavelmente todos os membros do anel estariam interessados nesta recompensa. É possível que afirmem que produziram tal assinatura. Mas quando o real assinante revelar os valores secretos k_i, e utilizá-los como chave secreta para produzir os MACs $x_i = HMAC_{k_i}(m)$, todos ficam convencidos de que foi ele quem gerou a assinatura, já que somente ele conhece os valores de k_i. Assim, o assinante pode comprovar que realmente produziu a assinatura $(P_1, P_2, ..., P_n; v; x_1, x_2, ..., x_n)$.

5.5 Trabalhos relacionados

Desde a introdução do conceito de assinatura em anel em 2001, muitos trabalhos foram propostos com base nessa ideia. Alguns trabalhos alteram o protocolo de geração da assinatura,

outros adicionam novas propriedades. Uma revisão do estado da arte dos esquemas de assinatura em anel pode ser vista em (WANG *et al.*, 2008). Um problema importante que tem recebido muita atenção é como revogar o anonimato do assinante. No esquema de Rivest *et al.* (2001), o verdadeiro assinante não pode revogar seu anonimato, porque essa propriedade não foi inserida pelos autores.

Esta seção descreve alguns trabalhos que foram publicados com o objetivo de acrescentar uma propriedade capaz de revogar a anonimato do assinante. Essa propriedade tem recebido diferentes nomes, ainda não há um consenso quanto a denominação. Por não haver tal consenso, neste trabalho, esta propriedade recebeu o nome de *resgate de autoria*.

Um dos métodos que apresenta uma maneira de resgatar a autoria da assinatura em anel foi proposto em (FILHO; NASCIMENTO, 2011). Os autores propõem o resgate de autoria em esquemas de assinatura em anel usando o conceito de dispositivos quânticos a prova de falsificação (BOUDA *et al.*, 2008). Tal proposta necessita da existência de uma terceira parte confiável que compartilham partículas quânticas emaranhadas com a máquina do delator. Para implementar tal proposta é necessário uma tecnologia ainda não factível atualmente. Foi a partir da proposta de Filho e Nascimento (2011) que surgiu a ideia de procurar um método implementável de gerar os valor de x_i, de tal modo que esse método suportasse informações secretas. As informações secretas seriam utilizadas posteriormente para provar a autoria da mensagem. Com isso, optou-se por utilizar os códigos de autenticação de mensagem - MACs.

Em (LEE *et al.*, 2005) um esquema que adiciona a propriedade de revogar o anonimato é chamado de *convertible ring signature*. Este esquema permite que o verdadeiro assinante revele uma informação secreta sobre a assinatura que garante sua autoria. Neste caso, a maneira como a chave z da função de encriptação simétrica E é gerada foi alterada. A geração da chave z envolve um parâmetro secreto que será revelado quando o assinante revolver assumir a autoria da assinatura em anel.

Outro método que pretende garantir o anonimamente, como no esquema de assinatura em anel original, mas também adicionar a possibilidade de resgatar a autoria posteriormente, foi proposto em (LV; WANG, 2003). Os autores propõe um novo método de assinatura definindo uma função trapdoor própria. Este esquema foi chamado de *verifiable ring signature* e permite que a verificação da autoria da assinatura possa ser realizada com base no problema do logaritmo discreto.

Os métodos propostos, como pode-se perceber, de alguma forma alteram uma ou mais etapas do protocolo original. Uma proposta interessante foi apresentada em (DONG *et al.*, 2012), os autores sugerem nenhuma alteração no esquema original de Rivest *et al.* (2001). Nessa proposta os autores apenas propõem uma maneira diferente de gerar um dos parâmetros de entrada da assinatura. No caso, eles geram o valor inicial v por meio de uma função hash

que recebe como entrada uma lista de valores aleatório públicos concatenada com um valor secreto r. O valor de v não é mais aleatório como em (RIVEST *et al.*, 2001), agora o vetor de inicialização v é calculado de forma semelhante a maneira como (LV; WANG, 2003) calculam a chave z da função de encriptação E. Em ambos os casos os valores são resultados de uma função hash que tem como entrada um lista de valores x_i concatenados com um valor secreto r que será revelado posteriormente, se o assinante desejar.

Esta dissertação também apresenta um método simples e eficiente de tornar um esquema de assinatura em anal baseado em RSA em um esquema de assinatura em anel conversível como chamado por Lee *et al.* (2005), ou simplesmente um esquema de assinatura em anel com resgate de autoria como é tratado aqui. A importância de manter o esquema baseado em RSA é porque o RSA é uma função trapdoor popularmente utilizada em operações pela Internet. O protocolo 3 mostra que, assim como em (FILHO; NASCIMENTO, 2011; LV; WANG, 2003; DONG *et al.*, 2012), o real assinante precisa apresentar alguma informação secreta para comprovar que, de fato, é o autor da assinatura em anel. Observe que os valores x_i não são mais aleatórios como nos trabalhos citados anteriormente, tais valores são calculados como MACs. Além disso, no esquema apresentado neste trabalho não basta apresentar um único valor secreto, é necessário relevar $n-1$ valores secretos (n é o número de participantes do anel), dificultando uma falsificação por parte de um atacante.

A prova da importância dessa propriedade é a quantidade razoável de trabalhos que sugerem uma forma de obtê-la. Neste trabalho a proposta de adicionar tal propriedade preza pela simplicidade e robustez do método. Todas as ferramentas utilizadas aqui são amplamente conhecidas na literatura e amplamente utilizadas nas transações pela Internet.

5.6 Comentários sobre a confiabilidade do resgate da autoria

Há três propriedades de segurança a serem satisfeitas pelo esquema de assinatura descrito neste trabalho. As propriedades são:

- *ambiguidade de assinante*. A probabilidade de que um verificador determinar com sucesso o assinante real de uma assinatura em anel é exatamente $1/n$, onde n é o número total de membros do anel.

- *assinatura não falsificável*. Qualquer atacante não deve ter probabilidade não desprezível de sucesso em falsificar uma assinatura em anel válida para alguma mensagem m em nome de um grupo do qual ele não participe, mesmo se ele conhecer assinaturas em anel válidas para mensagens diferente de m.

- *resgate de autoria apenas pelo real assinante*. Um não assinante não deve

conseguir convencer um verificador que é o assinante real porque ele não é capaz de computar eficientemente os valores para satisfazer os valores x_i da assinatura $(P_1, P_2, ..., P_n; v; x_1, x_2, ..., x_n)$. Como visto no protocolo 1, os valores x_i são MACs que dependem da mensagem m e de uma chave secreta k_i. Neste caso, as únicas maneiras de obter x_i é por meio de força bruta ou por meio do ataque do aniversário, mas ambas são computacionalmente inviáveis, conforme mostra a seção 3.2.2.

As duas primeiras propriedades foram definidas no trabalho original de Rivest *et al.* (2001). O esquema apresentado neste trabalho adiciona a terceira propriedade ao esquema de assinatura proposto por Rivest *et al.* (2001) sem alterar o protocolo original. É possível garantir resgate da autoria sem prejudicar a segurança quanto à identidade do delator. Embora as entradas da equação do anel não sejam mais aleatórias como em Rivest *et al.* (2001), as chaves usadas para o cálculo dos MACs são aleatórias. Assim, nota-se que para cada z e v a equação do anel tem $2^{|k|(n-1)}$ soluções.

A segurança do HMAC, utilizado neste trabalho como o algoritmo de MAC, está diretamente relacionada com a segurança da função hash usada internamente. Um algoritmo de MAC utiliza a função hash como uma caixa preta. Segundo (BELLARE *et al.*, 1996b), se o HMAC falhar como um MAC seguro, é porque há fraqueza suficiente na função hash embutida que precisa ser descartada. A segurança do MAC significa segurança contra falsificações. Um MAC é considerado quebrado se um atacante for capaz de encontrar alguma mensagem m juntamente com seu valor MAC correto, sem conhecer a chave k. Uma discussão sobre a segurança dos MAC pode ser encontrada no capítulo 3, seção 3.2.2. A segurança das funções hash foi discutida na seção 3.1.3.

Nesta proposta o que vai garantir que qualquer outro membro não assuma a autoria da mensagem são as chaves secretas k_i utilizadas no algoritmo MAC. Neste caso, os interessados na recompensa farão ataques por força bruta ao algoritmo MAC com o objetivo de conseguirem descobrir as chaves k_i utilizadas durante a geração da assinatura. Isto significa que o membro oponente teria que calcular $HMAC_k(m) = x_1$ para todos os valores possíveis de k ($2^{|k|}$ em que $|k|$ é o número de bits da chave), até encontrar um MAC que combine com x_1. Esse esforço é para encontrar apenas uma das chaves, mas para conseguir assumir a autoria da mensagem ele precisaria de um total de $n-1$ chaves (o valor de x_s não foi calculado através de MAC).

Em resumo, é possível dizer que o nível de esforço computacional para conseguir cada uma das chaves k_i é de aproximadamente $2^{|k|}$ tentativas. Assim, um parâmetro que garante a credibilidade do assinante ao gerar as entradas da assinatura em anel são os tamanhos das chaves apresentadas no momento de resgatar a autoria da assinatura.

5.6.1 Motivos que tornam este esquema mais seguro que outros esquemas anteriores

O objetivo desta subseção é apresentar uma comparação mais específica entre o esquema proposto neste trabalho baseado em (TOMAZ *et al.*, 2012) e o esquema proposto em (DONG *et al.*, 2012). O motivo dessa comparação mais específica deve-se ao fato do trabalho de Dong *et al.* (2012) apresentar uma semelhança mais próxima com o método proposto aqui. Outros trabalhos relacionados já foram discutidos na seção 5.5.

A principal semelhança entre os dois esquemas comparados é a utilização de Códigos de Autenticação de Mensagens-MACs para implementar o resgate de autoria. Contudo, o método MAC utilizando em (DONG *et al.*, 2012) é menos seguro quando comparado ao algoritmo HMAC utilizando neste trabalho.

No trabalho de Dong *et al.* (2012) é utilizado um tipo de MAC conhecido como *método do sufixo secreto* proposto em (TSUDIK, 1992). Tsudik (1992) apresenta dois métodos de autenticação de mensagem utilizando função hash, os quais serão descritos a seguir.

- *Método do prefixo secreto:* neste método a chave secreta k é utilizada como prefixo para a mensagem m e a função hash H recebe como entrada uma string composta por $k\|m$, isto é,

  $$H(k\|m) = MAC_k(m),$$

  em que $\|$ denota concatenação.

- *Método do sufixo secreto:* neste método a chave secreta k é utilizada como sufixo para a mensagem m e a função hash H recebe como entrada uma string composta por $m\|k$, isto é,

  $$H(m\|k) = MAC_k(m),$$

  em que $\|$ denota concatenação.

Em (DONG *et al.*, 2012), o resgate de autoria utiliza um esquema MAC e a chave secreta será revelada posteriormente para provar quem gerou a assinatura em anel. Especificamente, o esquema apresentado por Dong *et al.* (2012) altera a forma como o valor inicial v é gerado, isto é

$$v = H(x_1\| \cdots \|x_{s-1}\|x_{s+1}\| \cdots \|x_n\|r),$$

em que r é um valor secreto escolhido aleatoriamente. Segundo os próprios autores, há apenas duas formas de um atacante forjar a assinatura, e um deles é tentar encontrar um valor r que satisfaça $v = H(x_1\| \cdots \|x_{s-1}\|x_{s+1}\| \cdots \|x_n\|r)$, mas isso seria computacionalmente inviável porque a função hash H é resistente a colisão.

Observe que o método MAC escolhido por Dong *et al.* (2012) é o método do sufixo secreto descrito acima. Segundo Oppliger (2011), o método do sufixo secreto é considerado inseguro.

Oppliger (2011) argumenta que se a função H é uma função hash iterada (seção 3.1.2), então o método do sufixo secreto tem uma fraqueza estrutural. Tal fraqueza a ser explorada depende da função hash utilizada (mais especificamente sua função de compressão). O $\text{MAC}_k(m)$ consiste em

$$\underbrace{H(H(H(\cdots H(H(m_1)\|m_2)\|\cdots)\|m_n)}_{\text{H*(m)}}\|k),$$

em que m_i representa cada um dos blocos da mensagem m e $\|$ denota concatenação. Observe que $H^*(m)$ não depende de k. Essa característica permite o *know-message attack* (ataque da mensagem conhecida), em que o adversário tenta descobrir a chave ou outras informações, desde que conheça alguns pares mensagem-MAC.

Outro motivo que torna o método do sufixo secreto mais vulnerável é que um ataque de colisão off-line pode ser utilizado para obter uma colisão na função hash. Isso significa que é possível aplicar o ataque do aniversário. Segundo Stavroulakis e Stamp (2010) um ataque de colisão pode ser transformado em um ataque *MAC forgery* (falsificação de MAC) para esse esquema de MAC. Nesse ataque duas mensagens m e m' são encontradas, tal que $m \neq m'$ e $H(m) = H(m')$. Dessa forma, a função H é aplicada a uma mensagem $m\|y$, em que y é alguma mensagem arbitrária adicionada a m, de modo que $\text{MAC}_k(m\|y) = H(m\|y\|k)$. Consequentemente, a etiqueta MAC de $m\|y$ é também a etiqueta MAC para $m'\|y$ devido a estrutura interna da função hash H. Uma colisão interna para a função MAC iterada automaticamente permite uma verificação de um MAC falsificado por meio de um ataque do *chosen-message attack* (ataque de mensagem escolhida) que exige uma única mensagem escolhida. Observe que é possível falsificar a etiqueta MAC mesmo sem conhecer a chave secreta k.

A grande vantagem do HMAC em relação ao método do sufixo secreto é que a resistência à colisão da função hash usada internamente não é necessária. Em (BELLARE, 2006) o autor prova que o HMAC é seguro mesmo que a função utilizada internamente não seja resistente à colisão, basta que a função seja uma função pseudoaleatória, que é consideravelmente mais frágil que uma função hash resistente à colisão. Assim, mesmo que as funções hash resistentes à colisão populares (como por exemplo MD5 e SHA1) sejam quebradas, o HMAC ainda continua sendo seguro. Isso mostra que se a função hash utilizada no método do sufixo secreto não for resistente à colisão ou se essas funções forem quebradas no futuro, os dois ataque apresentados acima serão bem sucedidos para falsificar um MAC.

Ainda a favor do HMAC e outros bons algoritmos MAC, esses algoritmos são projetados para fazerem os ataque acimas muito mais custoso computacionalmente. É possível aplicar o ataque do aniversário em HMAC, mas tem que ser um ataque on-line (ver seção 3.2.2), o que faz com que os ataques ao HMAC sejam muito mais difíceis de executar na prática. Essa é uma razão pela qual o HMAC é preferível em detrimento do método do sufixo secreto.

6 CONCLUSÃO E PERSPECTIVA

A proposta apresentada neste trabalho representa uma expansão do esquema de assinatura em anel proposto por Rivest *et al.* (2001). A proposta de Rivest *et al.* (2001) permite que um assinante escolha um conjunto de membros para compor um grupo temporário e assinar mensagens em nome desse grupo de forma anônima. Um verificador é capaz de identificar que a mensagem partiu daquele grupo, mas não e capaz de dizer quem é o verdadeiro assinante. No protocolo original de Rivest *et al.* (2001) esse assinante não será capaz de revogar seu anonimato, mesmo que, no futuro, seja conveniente para ele.

Este trabalho apresenta um esquema de assinatura em anel baseado no esquema de Rivest *et al.* (2001), em que o assinante pode, mais tarde, revogar seu anonimato apresentando valores secretos que provam que somente ele seria capaz de gerar tal assinatura. Outros trabalhos já foram publicados com o objetivo de acrescentar tal propriedade capaz de revogar o anonimato do assinante. Esta propriedade tem recebido diferentes nomes, ainda não há um consenso quanto a denominação, por não haver tal consenso, neste dissertação, tal propriedade recebeu o nome de *resgate de autoria*. Este trabalho preserva todos as propriedades originais do esquema de assinatura em anel, não adiciona elementos extras e utiliza apenas ferramentas computacionais populares e amplamente utilizadas na prática. A principal diferença em relação ao trabalho de Rivest *et al.* (2001) é apresentada antes mesmo de começar a geração da assinatura. Os valores utilizados como entrada para a função trapdoor serão MACs gerados pelo algoritmo HMAC, um algoritmo de autenticação de mensagem baseado em função hash resistente à colisão. Essa modificação simples permitirá que, no futuro, o assinante revele-se como o verdadeiro autor da mensagem com a garantia que a fonte poderá ser autenticada devido aos códigos de autenticação de mensagem - MACs amplamente conhecidos e utilizados em operações computacionais.

O que garante o resgate de autoria pelo real assinante é provar que os valor x_i usados como entrada para a função trapdoor não são mais aleatórios como em (RIVEST *et al.*, 2001). Estes valores são agora calculados por meio de um algoritmo MAC que recebeu como entrada chaves secretas k_i e a mensagem a ser assinada m, portando são códigos de autenticação de mensagem calculados de forma determinística. Assim, o que vai garantir que qualquer outro membro do

anel não assuma a autoria da mensagem são as chaves secretas k_i utilizadas no algoritmo MAC. Neste caso, os interessados em forjar tal assinatura para receberem uma possível recompensa terão que fazer ataques por força bruta ao algoritmo MAC com o objetivo de conseguirem descobrir as chaves k_i utilizadas durante a geração da assinatura. Isso significa que o membro oponente teria que fazer $MAC_k(m) = x_1$ para todos os valores possíveis de k ($2^{|k|}$ onde $|k|$ é o número de bits da chave), até encontrar um MAC que combine com x_1. Esse esforço é para encontrar apenas uma das chaves, mas para conseguir assumir a autoria da mensagem ele precisaria de um total de $n-1$ chaves secretas.

A proposta deste trabalho é factível experimentalmente e ainda não foi implementada. Para uma implementação, coloca-se como perspectivas futuras, uma avaliação de uma assinatura em anel com diferentes funções trapdoors. Isso atenderia as necessidades dos delatores em diferentes instituições. Instituições podem optar por usar outras funções criptográficas de chave pública além do popular RSA. Para os programadores interessados na implementação de um esquema como esse, é importante saber o percentual dos tipos de funções trapdoors usadas na Internet.

REFERÊNCIAS

AGRAWAL, M.; KAYAL, N.; SAXENA, N. PRIMES is in P. *Annals of Mathematics 160*, v. 2, p. 781–793, 2004. Versão original do artigo publicada em 2002.

ANDRESS, J. *The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice*. 1. ed. USA: Elsevier Science, 2011.

BACH, E.; SHALLIT, J. *Algorithmic Number Theory: Efficient algorithms*. 1. ed. [S.l.]: MIT Press, 1996.

BELLARE, M. New proofs for nmac and hmac: Security without collision-resistance. In: DWORK, C. (Ed.). *Advances in Cryptology - CRYPTO 2006*. [S.l.]: Springer Berlin Heidelberg, 2006, (Lecture Notes in Computer Science, v. 4117). p. 602–619.

BELLARE, M.; CANETTI, R.; KRAWCZYK, H. Keying Hash Functions for Message Authentication. In: ANNUAL INTERNATIONAL CRYPTOLOGY CONFERENCE ON ADVANCES IN CRYPTOLOGY, 16th., 1996. *Anais...* London, UK: Springer-Verlag, 1996. (CRYPTO '96), p. 1–15.

BELLARE, M.; CANETTI, R.; KRAWCZYK, H. Message Authentication using Hash Functions- The HMAC Construction. *CryptoBytes*, v. 2, 1996.

BELLARE, M.; KRAWCZYK, H.; CANETTI, R. *RFC2104 - HMAC:Keyed-Hashing for Message Authentication*. United States: RFC Editor, 1997.

BIHAM, E.; CHEN, R.; JOUX, A.; CARRIBAULT, P.; LEMUET, C.; JALBY, W. Collisions of SHA-0 and Reduced SHA-1. In: ANNUAL INTERNATIONAL CONFERENCE ON THE THEORY AND APPLICATIONS OF CRYPTOGRAPHIC TECHNIQUES, 24th., 2005. *Anais...* [S.l.]: Springer, 2005. (EUROCRYPT 2005).

BOER, D.; BOSSELAERS, A. Collisions for the Compression Function of MD5. In: WORKSHOP ON THE THEORY AND APPLICATION OF CRYPTOGRAPHIC TECHNIQUES ON ADVANCES IN CRYPTOLOGY, 1993. *Anais...* Secaucus, NJ, USA: Springer-Verlag, 1993. (EUROCRYPT '93), p. 293–304.

BONEH, D. Twenty Years of Attacks on the RSA Cryptosystem. *NOTICES OF THE AMS*, v. 46, p. 203–213, 1999.

BOUCINHA, F. C. *A Survey of Cryptanalytic Attacks on RSA*. Dissertação (Mestrado) — Universidade Técnica de Lisboa, Lisboa, 2011.

BOUDA, J.; MATEUS, P.; PAUNKOVIC, N.; RASGA, J. On the power of quantum tamper-proof devices. *International Journal of Quantum Information*, v. 6, 2008.

BRIGGS, M. E. *An Introduction to the General Number Field Sieve*. Dissertação (Mestrado) — Virginia Polytechnic Institute and State University, Blacksburg, Virginia, USA, 1998.

BUCHMANN, J. A. *Introdução à Criptografia*. 1. ed. São Paulo: Berkeley, 2002.

BURNETT, S.; PAINE, S.; SECURITY, R. *RSA Security's official guide to cryptography*. 1. ed. United States: Osborne/McGraw-Hill, 2001.

CHAUM, D.; HEYST, E. V. Group signatures. In: *Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques*. Berlin, Heidelberg: Springer-Verlag, 1991. (EUROCRYPT'91), p. 257–265.

CHOKHANI, S.; FORD, W.; SABETT, R.; MERRILL, C.; WU, S. *Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*. IETF, nov. 2003. RFC 3647 (Informational). (Request for Comments, 3647). Disponível em: <http://www.ietf.org/rfc/rfc3647.txt>. Acesso em: 19/03/2014.

COUTINHO, S. *Números inteiros e criptografia RSA*. 2. ed. Rio de Janeiro: IMPA-Instituto Nacional de Matemática Pura e Aplicada, 2005.

DAMGÅRD, I. A Design Principle for Hash Functions. In: ANNUAL INTERNATIONAL CRYPTOLOGY CONFERENCE ON ADVANCES IN CRYPTOLOGY, 9th., 1989, London, UK. *Anais...* London, UK: Springer-Verlag, 1989. (CRYPTO '89), p. 416–427.

DIFFIE, W. The first ten years of public-key cryptography. *Proceedings of the IEEE*, Institute of Electrical and Electronic Engineers, v. 76, p. 560 – 577, maio 1988.

DIFFIE, W.; HELLMAN, M. New Directions in Cryptography. *Information Theory, IEEE Transactions*, Institute of Electrical and Electronic Engineers, v. 22, p. 644–654, set. 1976.

DIFFIE, W.; HELLMAN, M. E. Multiuser Cryptographic Techniques. In: NATIONAL COMPUTER CONFERENCE AND EXPOSITION, 1976, New York. *Anais...* New York: ACM, 1976. (AFIPS '76), p. 109–112.

DONG, Q.; LI, X.; LIU, Y. Two extensions of the ring signature scheme of rivest-shamir-taumann. *Information Sciences*, v. 188, n. 0, p. 338 – 345, 2012.

FILHO, J. A. F.; NASCIMENTO, J. C. Como vazar uma mensagem de forma anônima e depois resgatar a autoria. In: Simpósio Brasileiro de Telecomunicações, 29ª., 2011, Curitiba. *Anais...* Curitiba: SBrT-Sociedade Brasileira de Telecomunicações, 2011.

GOLDREICH, O. *Foundations of Cryptography: Basic Techniques*. New York: Cambridge University Press, 2004.

GOODRICH, M.; TAMASSIA, R. *Introdução à Segurança de Computadores*. 1. ed. Porto Alegre: Bookman, 2013.

KNUDSEN, L. R.; ROBSHAW, M. J. *The Block Cipher Companion*. 1. ed. [S.l.]: Springer, 2011.

KUMAR, M. *Cryptography and Network Security*. 3. ed. India: Krishna Prakashan, 2006.

KUROSE, J.; ROSS, K. *Redes de computadores e a Internet: Uma nova abordagem top-down*. 5. ed. São Paulo: Addison Wesley, 2010.

LEE, K.; WEN, H.; HWANG, T. Convertible ring signature. *IEE Proceedings - Communications*, v. 152, p. 411 – 414, 2005.

LV, J.; WANG, X. Verifiable ring signature. In: *Proceedings of the Third International Workshop on Cryptology and Network Security*. Miami FL, USA: DMS Proceedings, 2003. (CANS'03), p. 663–665.

MENEZES, A.; OORSCHOT, P. van; VANSTONE, S. *Handbook of Applied Cryptography*. USA: Taylor & Francis, 1996.

MERKLE, R. C. *Secrecy, Authentication, and Public Key Systems*. Tese (Doutorado) — Stanford University, Stanford, CA, USA, 1979.

MERKLE, R. C. One Way Hash Functions and DES. In: ANNUAL INTERNATIONAL CRYPTOLOGY CONFERENCE ON ADVANCES IN CRYPTOLOGY, 9th., 1989, London, UK. *Anais...* London, UK: Springer-Verlag, 1989. (CRYPTO '89), p. 428–446.

MILLER, G. L. Riemann's Hypothesis and Tests for Primality. *Journal of Computer and System Sciences*, v. 13, n. 3, p. 300 – 317, 1976.

National Institute of Standards and Technology. *FIPS 180-1. Secure Hash Standard*. [S.l.], apr 1995.

National Institute of Standards and Technology. *FIPS 180-3, Secure Hash Standard, Federal Information Processing Standard (FIPS), Publication 180-3*. [S.l.], ago. 2008.

OPPLIGER, R. *Contemporary Cryptography*. 2. ed. United States: Artech House, 2011.

PARDO, J. *Introduction to Cryptography with Maple*. 1. ed. [S.l.]: Springer-Verlag, 2013.

POLLARD, J. M. A Monte Carlo method for factorization. *BIT*, v. 15, p. 331–334, 1975.

RABIN, M. Probabilistic Algorithm for Testing Primality. *Journal of Number Theory*, v. 12, n. 1, p. 128 – 138, 1980.

RIVEST, R. *RFC1321 - The MD5 Message-Digest Algorithm*. United States: RFC Editor, 1992.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Commun. ACM*, ACM, New York, NY, USA, v. 21, n. 2, p. 120–126, fev. 1978.

RIVEST, R. L.; SHAMIR, A.; TAUMAN, Y. How to leak a secret. In: INTERNATIONAL CONFERENCE ON THE THEORY AND APPLICATION OF CRYPTOLOGY AND INFORMATION SECURITY: ADVANCES IN CRYPTOLOGY, 7th., 2001, London, UK. *Anais...* London, UK: Springer-Verlag, 2001. (ASIACRYPT '01), p. 552–565. Disponível em: <http://dl.acm.org/citation.cfm?id=647097.717015>.

Secretaria de Estado de Fazenda do Distrito Federal. *O Que é Certificação Digital*. 2012. Disponível em: <www.fazenda.df.gov.br/arquivos/pdf/O_que_e_certificado_digital.pdf>. Acesso em: 19/03/2014.

SHOR, P. W. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In: ANNUAL SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE, 1994, Washington, DC, USA. *Anais...* Washington, DC, USA: IEEE Computer Society, 1994. (SFCS '94), p. 124–134.

SOLOVAY, R.; STRASSEN, V. A Fast Monte-Carlo Test for Primality. *SIAM Journal on Computing*, SIAM, v. 6, n. 1, p. 84–85, 1977.

STALLINGS, W. *Criptografia e Segurança de Redes: Princípios e Práticas*. 4. ed. São Paulo: Pearson/Prentice Hall, 2008.

STAVROULAKIS, P.; STAMP, M. *Handbook of Information and Communication Security*. [S.l.]: Springer, 2010.

TERADA, R. *Segurança de Dados: Criptografia em Redes de Computadores*. 2. ed. São Paulo: Blucher, 2008.

TOMAZ, A. E. B.; FILHO, J. A. F.; NASCIMENTO, J. C. Recuperando a autoria de uma assinatura em anel usando códigos mac. In: Simpósio Brasileiro de Telecomunicações, 30ª., 2012, Brasília. *Anais...* Brasília: SBrT-Sociedade Brasileira de Telecomunicações, 2012.

TSUDIK, G. Message authentication with one-way hash functions. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 22, n. 5, p. 29–38, out. 1992.

WANG, L.; ZHANG, G.; MA, C. A survey of ring signature. *Frontiers of Electrical and Electronic Engineering in China*, Higher Education Press, v. 3, n. 1, p. 10–19, 2008.

WHITMAN, M.; MATTORD, H. *Principles of Information Security*. 4. ed. USA: Cengage Learning, 2011.

YAN, S. *Computational Number Theory and Modern Cryptography*. 2. ed. Singapore: Wiley, 2012.

YUVAL, G. How to Swindle Rabin. *Cryptologia*, v. 3, p. 187–189, 1979.

APÊNDICE A – BASE MATEMÁTICA DO PARADOXO DO ANIVERSÁRIO E ATAQUE DO ANIVERSÁRIO

Este apêndice apresenta a base matemática para compreensão do paradoxo do aniversário e do ataque do aniversário. O desenvolvimento deste apêndice está baseado nos livros de Stallings (2008) e de Goodrich e Tamassia (2013).

A.1 Base matemática do paradoxo do aniversário

O paradoxo do aniversário diz que em um grupo de 23 pessoas escolhidas aleatoriamente, a probabilidade de pelo menos duas delas compartilharem a mesma data de aniversário é maior que 1/2. Como é mais fácil calcular a probabilidade de duas pessoas não compartilharem a mesma da de aniversário, considerando um grupo de k pessoas e um ano de 365 dias, então

$$\bar{p}(k) = \left(\frac{365}{365}\right) \cdot \left(\frac{364}{365}\right) \cdot \left(\frac{363}{365}\right) \cdots \left(\frac{365-(k-1)}{365}\right), \tag{A.1}$$

em que $\bar{p}(k)$ é a probabilidade de não haver colisão de datas. Essa probabilidade pode ser calculado por

$$\bar{p}(k) = \frac{365!}{365^k(365-k)!}. \tag{A.2}$$

O resultado obtido em A.2 pode ser mais facilmente compreendido observando os detalhe a seguir. Observe que

$$365 \cdot 364 \cdot 363 \cdots (365-(k-1)) = \frac{365!}{(365-k)!}$$

Desta forma, é possível simplificar a equação A.1 com a equação a seguir

$$\bar{p}(k) = \frac{\frac{365!}{(365-k)!}}{365^k} = \frac{365!}{365^k(365-k)!}. \tag{A.3}$$

A.2 Base matemática do ataque do aniversário

O raciocínio do paradoxo do aniversário por ser estendido para o ataque do aniversário. Nesse caso a questão é quantas entradas k para a função hash H serão necessárias para encontrar uma colisão com probabilidade de 1/2. O problema do aniversário é apenas um problema específico. Aqui o número 365 (representando a quantidade de dias do ano) será substituído por $n = 2^b$ onde b é quantidade de bits da etiqueta hash e 2^b é quantidade de etiquetas hash possíveis. Com base no paradoxo do aniversário tem-se

$$p(k) = 1 - \frac{n!}{n^k(n-k)!}. \tag{A.4}$$

Essa generalização pode ser reescrita como

$$\begin{aligned}
p(k) &= 1 - \frac{n \cdot (n-1) \cdots (n-(k-1))}{n^k} \\
p(k) &= 1 - \left[\frac{n-1}{n} \cdot \frac{n-2}{n} \cdots \frac{n-(k-1)}{n}\right] \\
p(k) &= 1 - \left[\left(1-\frac{1}{n}\right) \cdot \left(1-\frac{2}{n}\right) \cdots \left(1-\frac{k-1}{n}\right)\right]
\end{aligned}$$

Para encontrar uma expressão fechada para $p(k)$ considere

$$1 - x \approx e^{-x} \tag{A.5}$$

para x próximo de 0. Como n é grande, $1/n$ é próximo de 0, assim obtém-se

$$\begin{aligned}
p(k) &\approx 1 - \left[e^{-1/n} \cdot e^{-2/n} \cdots e^{-(k-1)/n}\right] \\
p(k) &\approx 1 - e^{-[(1/n)+(2/n)+\cdots+((k-1)/n)]} \\
p(k) &\approx 1 - e^{-(k(k-1))/2n}
\end{aligned}$$

Considerando que a probabilidade desejada é 0.5, tem-se

$$1/2 = 1 - e^{-(k(k-1))/2n} \tag{A.6}$$

$$2 = e^{(k(k-1))/2n} \tag{A.7}$$

$$ln2 = \frac{k(k-1)}{2n} \tag{A.8}$$

Para k muito grande pode-se substituir $k(k-1)$ por k^2. Desenvolvendo a equação A.8 tem-se

$$k = \sqrt{2(ln2)n} = 1.1774\sqrt{n}$$
$$k \approx \sqrt{n}$$

Observe que $n = 2^b$, em que b representa o tamanho em bits da etiqueta hash de saída. Dessa forma é fácil perceber que

$$k \approx \sqrt{2^b} = 2^{b/2}$$