UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Douglas Marcelino Beppler Martins

ALGORITMOS DE HASH CAMALEÃO

Florianópolis

2017

Para as mulheres da minha vida

AGRADECIMENTOS

Primeiramente agradeço a minha mãe, Maria Goreti, que me ensinou a fazer as coisas por amor acima de tudo. Agradeço a meu pai, Marcio, que me ensinou a ter determinação e disciplina na vida. Agradeço as minhas irmãs, Tamiris e Tais, que me ensinaram muito e são pessoas em que me inspiro muito. Agradeço a minha namorada, Júlia de Assis, que me ensinou que devo sonhar e batalhar para realizar todos os meus sonhos. Agradeço aos amigos Ronan Prazeres, Lucas Kretzer e Vitor Macário, que apesar de momentos distantes as amizades continuam fortes. Agradecer a meu orientador, professor Custódio ao professor Jean e ao Maurício Simões, por todas os ensinamentos ao longo dessa jornada. Agradecimento a minha equipe do Laboratório de Segurança em Computação, Gustavo Zambonin, Matheus Bittencourt e Ramna Sidharta pelo companheirismo e trabalho em grupo, agradeço aos meus colegas que ingressaram no curso junto comigo, principalmente ao Maike de Paula, por toda a parceria ao longo de inúmeras disciplinas ao longo curso e ao Matheus Teixeira, pelas conversas e a parceria. Por fim agradeço a Deus, por ter colocado todas essas e as incontáveis outras pessoas em meu caminho.

Wars come and go, but my soldiers stay eternal

2Pac

RESUMO

Resumos criptográficos têm sido usados para se verificar a integridade de mensagens e documentos eletrônicos. Os resumos criptográficos são números relativamente pequenos que representam de forma única os documentos eletrônicos. Assim, diferentes documentos eletrônicos possuem diferentes resumos criptográficos. A assinatura digital de um documento eletrônico, por exemplo, consiste do ciframento, utilizando a chave privada do signatário, do resumo criptográfico do documento. Se qualquer parte do documento assinado for modificada, o resumo criptográfico será diferente, invalidando a assinatura de todo o documento. Acontece que, em algumas situações práticas, é desejável poder-se alterar parte do documento sem invalidar a assinatura digital anteriormente produzida. É o caso do prontuário médico eletrônico assinado digitalmente por um médico. Esses, por razões de privacidade, precisam esconder ou modificar o nome do paciente, quando o prontuário precisar ser disponibilizado de forma pública. Entretanto, a modificação do prontuário, invalida a assinatura digital do médico. Para solucionar este problema, foi proposto na literatura o uso de um novo tipo de resumo criptográfico, denominado de hash camaleão. A diferença para os tradicionais hashes é que diferentes documentos podem ter o mesmo resumo criptográfico. Assim, uma mesma assinatura digital pode ser utilizada para se garantir a integridade e autenticidade de diferentes documentos. Por exemplo, duas versões do mesmo prontuário médico, um contendo o nome do paciente e o outro substituindo o nome do paciente por um nome genérico, poderiam ter a mesma assinatura digital. Neste trabalho são descritos e comparados os principais algoritmos de hash cameleão propostos na literatura. Além disso, é apresentado um software de assinatura digital utilizando hashes camaleão.

Palavras-chave: Integridade, Assinatura Digital, Hash, Hash Camaleão

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Funções de resumo criptográficos (do inglês, *hash function*), são funções amplamente utilizadas em diversas áreas da tecnologia da informação, cujo principal objetivo é mapear uma palavra de qualquer tamanho, que pode ser vista como uma sequência de bits, para uma palavra de tamanho fixo e relativamente pequena, ou seja, um número da ordem de grandeza entre 256 e 1024 bits.

Esse número pequeno, denominado de *hash*, pode ser usado para representar qualquer arquivo digital, como uma foto, um vídeo ou um documento PDF. Cada arquivo digital tem o seu próprio número *hash* que o representa. Além disso, por definição, o *hash* não fornece qualquer informação sobre a sequência de bits que o representa.

Para que possamos utilizar funções de resumo de forma segura em um contexto de assinatura digital de documentos eletrônicos, algumas propriedades necessitam ser respeitadas. A primeira delas, chamada de resistência à pré-imagem, consiste em não ser possível, a partir do *hash*, obter o documento eletrônico utilizado para se determinar esse *hash*.

Um exemplo da importância dessa propriedade é no armazenamento das senhas dos usuários em um provedor de serviços. Para evitar o armazenamento das senhas em claro, o servidor não armazena as senhas dos usuários, mas sim os *hashes* das senhas. Para se autenticar, o usuário envia sua senha para o servidor. Este determina o *hash* da senha e compara com o *hash* da senha previamente armazenado no tabela de senhas do servidor. Se for o mesmo *hash*, trata-se portanto da mesma senha e o servidor considera o usuário autenticado. O armazenamento do *hash* das senhas evita que seja possível obter a senha dos usuários a partir da tabela contendo os *hashes* das senhas.

Uma segunda propriedade importante que algoritmos de resumo precisam garantir é a resistência à colisão. Essa garante que não é possível encontrar dois documentos eletrônicos distintos que tenham o mesmo *hash*. A resistência à colisão é utilizada para assegurar a integridade das assinaturas digitais de documentos eletrônicos.

O modelo clássico de gerar a assinatura digital de um documento

eletrônico consiste em cifrar o *hash* do documento com a chave privada do signatário. Para se verificar a assinatura, o verificador decifra o *hash* com a chave pública do signatário e compara com o *hash* do documento. Caso eles sejam iguais, é possível afirmar quem é o signatário do documento e também que o documento está íntegro.

A terceira e última propriedade é a resistência à segunda pré-imagem. Dado o *hash* de um documento eletrônico, não deve ser possível encontrar outro documento diferente que tenha o mesmo *hash*.

Contudo, há situações na prática, principalmente relacionadas a requisitos de privacidade, onde é desejável manter a assinatura digital do documento realizada pelo signatário, mesmo que partes do mesmo sejam modificadas. Por exemplo, um prontuário médico de um paciente assinado por um médico, poderia ter o nome do paciente modificado, mantendo a integridade e autenticidade proporcionadas pela assinatura digital. Assim, o prontuário poderia ser disponibilizado, sem revelar o nome do paciente.

Em esquemas de assinaturas que utilizam funções de resumos criptográficos tradicionais, não é possível se realizar alterações nos documentos assinados. Porém, tem sido proposto na literatura algoritmos de *hash* onde é permitido, de forma controlada, a colisão de *hashes*. Esses novos algoritmos de *hashes* são conhecidos como *hash* camaleão (em ingles *chameleon hash*).

Hashes camaleão foram propostos por Hugo Krawczyk e Tal Rabin em 1998 (KRAWCZYK; RABIN, 1998, 2000). O algoritmo proposto por eles faz uso de um par de chaves criptográficas, ou seja, uma chave pública e a respectiva chave privada, com as seguintes características:

- Qualquer pessoa com o conhecimento da chave pública associada ao algoritmo pode calcular o *hash*;
- Para os que não conhecem a chave privada associada ao algoritmo é computacionalmente muito difícil encontrar duas mensagens diferentes das quais tenham um mesmo valor de *hash*;
- Para o detentor da chave privada é possível facilmente determinar uma segunda mensagem, diferente da primeira, que tenha o mesmo valor de *hash*.

Essas características garantem a um esquema de assinatura digi-

tal que utilize o algoritmo de *hash* camaleão como algoritmo de resumo, a possibilidade de se modificar um documento assinado de forma segura.

O presente trabalho visa realizar um estudo sobre algoritmos de *hash* camaleão, suas principais funções e propriedades. Os principais algoritmos de resumo camaleão propostos na literatura são analisados e comparados. Dois dos algoritmos foram implementados.

1.1 OBJETIVOS

1.1.1 Objetivos gerais

Este trabalho tem como objetivo principal realizar uma pesquisa, buscando os diversos tipos de construção de esquemas de resumo criptográfico camaleão, analisando a geração de chaves, a forma de como é calculado o valor de resumo e como é encontrada uma colisão. Adicionalmente, são analisadas suas principais características e propriedades visando seu uso em esquemas de assinatura digital.

1.1.2 Objetivos específicos

- Apresentar o *hash* camaleão, suas funções e suas características visando seu uso em assinaturas digitais;
- Apresentar os principais esquemas do algoritmo de resumo camaleão encontrados na literatura;
- Escolher e implementar alguns desses *hashes* camaleão;
- Comparar os esquemas apresentados, relacionando as propriedades analisadas de cada esquema;
- Elaborar um esquema de assinatura digital onde é possível utilizar o algoritmo de resumo criptográfico camaleão como algoritmo de *hash*.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 INTRODUÇÃO

Nesse capítulo são apresentados os principais conceitos criptográficos necessários para o entendimento de funções de *hash* e para se realizar construções de esquemas de resumo camaleão.

2.2 CRIPTOGRAFIA

Ao passar dos séculos as inúmeras formas de transmitir mensagens de forma secreta foram propostas. Passando pela era da criptografia clássica, onde o cifrador de César (100-44 A.C.), cifrador de Vigenère (1586) e a Maquina Enigma (1939) são uns dos principais cifradores dessa era. Com a publicação feita no ano de 1976 por Whitfield Diffie e Martin Hellman (DIFFIE; HELLMAN, 1976) se da inicio a era da criptografia moderna, onde é introduzido a criptografia de chaves públicas e novos cifradores como o DES (*Data Encryption Standard*, 1977) e o AES (*Advanced Encryption Standard*, 2001).

Em sua essência, criptografia é a prática de tornar ilegível um texto previamente legível e depois inverter tal operação, tornando possível proteger a mensagem, omitindo suas informações de entidades indesejadas. Hoje criptografia é fundamentada em quatro características da segurança da informação:

i. Confidencialidade: Manter a informação secreta de todos, exceto de quem é autorizado a vê-la.

ii. Integridade: Assegurar que a informação não foi alterada por meios desconhecidos ou não autorizados.

iii. Autenticidade: Confirmação da identidade da entidade que realizou a assinatura da mensagem.

iv. Não repúdio: Confirmação de quem mandou a mensagem.

2.3 ALGORITMOS DE RESUMO CRIPTOGRÁFICO

Funções de resumo (em inglês, *hash*) são usadas em diversas áreas da ciência da computação, como por exemplo em tabelas de espalhamento, uma estrutura de dados que associa uma chave a um dado valor, que utiliza de uma função de espalhamento (função *hash*) para computar a posição em um arranjo em que estará alocado tal valor, no âmbito de assinaturas digitais, as funções de resumo são ás responsáveis por garantir a integridade de um documento assinado.

Na criptografia, uma função de resumo criptográfico (em inglês, *cryptographic hash function*) é uma função que mapeia uma palavra de tamanho qualquer para uma palavra com um tamanho fixo $t \in \mathbb{N}$. Para ser considerada segura, uma função de resumo criptográfico $hash : X \longrightarrow \{0,1\}^t$ deve respeitar as seguintes propriedades(DANG, 2009):

i. Resistência à pré-imagem: Dado um hash $h \in \{0,1\}^t$, deve ser computacionalmente inviável encontrar uma mensagem $m \in X$ tal que $h = hash(m)$. Esse tipo de função é chamado de *one-way function*.

ii. Resistência à segunda pré-imagem: Dado uma mensagem $m_1 \in X$, deve ser computacionalmente inviável encontrar outra mensagem $m_2 \in X$ diferente de m_1 tal que $hash(m_1) = hash(m_2)$.

iii. Resistência à colisão: É computacionalmente inviável encontrar duas palavras $m_1, m_2 \in X$ com $m_1 \neq m_2$ tal que $hash(m_1) = hash(m_2)$.

As funções mais utilizadas pertencem a família SHA (*Secure Hash Algorithm*), publicada pelo *National Institute of Standards and Technology* (NIST) dos Estados Unidos. O primeiro algoritmo desta família, SHA-1, já não é mais considerado seguro, devido ao avanço da criptoanálise e no poder computacional, no ano de 2017 foi publicado a primeira colisão para este algoritmo (STEVENS et al., 2017).

2.4 CRIPTOGRAFIA DE CHAVE PÚBLICA

O conceito de um sistema criptográfico de chaves públicas foi proposto em 1976 por Whitfield Diffie e Martin Hellman, porém não foi proposto nenhuma estrutura algébrica da qual seja possível se construir um esquema de chaves públicas. Foi proposto então no ano de

1977 o algoritmo de RSA para se construir um esquema de assinatura digital de chaves públicas. Um sistema de criptografia assimétrica, é composto por um par de chaves, uma pública e outra privada, onde uma chave realiza uma operação que pode ser desfeita apenas pela outra. Por exemplo: a chave pública é utilizada para cifrar um texto plano e a chave privada para decifrar.

Em um sistema de criptografia assimétrica, deve ser computacionalmente inviável encontrar a chave privada conhecendo apenas o algoritmo e a chave pública. Para garantir essa propriedade, os algoritmos se baseiam em problemas matemáticos computacionalmente difíceis de se resolver, por exemplo o RSA, primeiro sistema criptográfico de chaves pública, que se baseia no problema da fatoração de números primos.

2.5 RSA

Desenvolvido por Ron Rivest, Adi Shamir, e Leonard Adleman e publicado em 1977, o RSA leva em seu nome as iniciais de seus autores. O algoritmo proposto foi desenvolvido para tornar possível o uso do esquema desenvolvido por Diffie e Hellman (DIFFIE; HELLMAN, 1976). Antes de explicar o algoritmo em si é necessário o entendimento da função *totiente* de Euler e do conceito de inversa multiplicativa módulo n.

A função *totiente* de Euler, $\varphi(n)$, define o número de inteiros positivos menores que n que são relativamente primos a n. A inversa multiplicativa de $x \pmod{n}$ é um número inteiro a tal que $a \times x \equiv 1 \pmod{n}$. O algoritmo RSA então pode ser explicado em três partes:

i. Geração de chaves:

   1. Compute dois números primos distintos p e q
   2. Compute $n = p \times q$ e $\varphi(n)$
   3. Escolha e tal que $1 < e < \varphi(n)$ e $gcd(\varphi(n), e) = 1$
   4. Compute d tal que $e \times d \equiv 1 \pmod{\varphi(n)}$, isto é, d é a inversa multiplicativa de e em $\mathbb{Z}_{\varphi(n)}$
   5. Chave pública $= (e, n)$, Chave privada $= (d, p, q)$

ii. Cifragem: A cifragem do RSA é feita em blocos, em virtude do tamanho das mensagens, na prática, ser maior do que n. Este

particionamento acontece de tal maneira em que as mensagens a serem cifradas sempre serão menores que n, com cada partição sendo cifrada separadamente. Para realizar a operação sobre uma mensagem m com a chave pública (n, e), onde $m < n$ basta computar

$$c = m^e \pmod{n}. \tag{2.1}$$

iii. Decifragem: Para decifrar um dado cifrado c com a chave privada (n, d) basta calcular

$$m = c^d \pmod{n}. \tag{2.2}$$

Para que o algoritmo tenha um funcionamento devido, deve ser possível encontrar valores para e, d e n que satisfaçam a seguinte igualdade

$$m^{ed} = m \pmod{n}; \tag{2.3}$$

esta equação é valida se e e d forem inversa multiplicativa em $\mathbb{Z}_{\varphi(n)}$ (RIVEST; SHAMIR; ADLEMAN, 1978).

A segurança do algoritmo RSA é baseada no problema fatoração da números inteiros (VON ZUR GATHEN, 2015). É esperado que a função de cifragem seja uma função de via única. Caso um atacante obtenha a fatoração de $n = pq$, então é possível encontrar $\phi(n) = (p-1)(q-1)$ e consequentemente a chave privada, através do processo descrito na Seção 2.5. Afortunadamente, fatoração de números parece ser tão difícil quanto determinar se um número é primo ou composto.

2.6 CURVAS ELÍPTICAS

Curva elíptica é um plano real algébrico, que define um conjuntos de pontos em um plano Euclideano, assim é possível realizar a seguinte definição das curvas de um corpo primo $\mathbb{F}_p$

$$y^2 = x^3 + ax + b \pmod{p} \tag{2.4}$$

onde p é um número primo que define o corpo finito $\mathbb{F}_p$, este número primo é escolhido de forma que a curva tenha suficientemente um grande número de pontos, para que seja considerada segura. Os parâmetros $a, b \in \mathbb{Z}_p$ são os que definem a curva estes precisam respeitar a seguinte equação (VON ZUR GATHEN, 2015):

$$4a^3 + 27b^2 \neq 0. \tag{2.5}$$

Os pontos de uma curva são as tuplas (x, y) que respeitem a equação da curva, e neles são definidas as operações de adição e multiplicação por um escalar.

2.7 GRUPOS ALGÉBRICOS

Na matemática, um grupo é um conjunto de elementos, munido de uma operação que combina quaisquer dois elementos desse conjunto. Esta operação precisa respeitar as 4 propriedades a seguir para ser considerado um grupo. Seja S um conjunto e $\cdot$ a uma operação sob o conjunto

i. Fechamento da operação:
$\forall a, b \in S, a \cdot b \in S$;

ii. Associatividade da operação:
$\forall a, b, c \in S, (a \cdot b) \cdot c = a \cdot (b \cdot c)$;

iii. Existência do elemento neutro:
$\exists! e \in S, \forall a \in S, e \cdot a = a \cdot e = a$;

iv. Existência do elemento inverso:
$\forall a \in S, \exists! a^{-1} \in S, a \cdot a^{-1} = a^{-1} \cdot a = e$.

2.7.1 Grupos cíclicos

Um grupo cíclico é um grupo que é gerado por um único elemento (European Mathematical Society, 1994), em outras palavras, todo elemento do grupo $\mathbb{G}$ pode ser gerado através do elemento g pertencente ao grupo, e este elemento é chamado de gerador do grupo.

2.7.2 Grupos cíclicos aditivos e grupos cíclicos multiplicativos

Grupos cíclicos aditivos e grupos cíclicos multiplicativos são grupos nos quais as operações que estes estão munidos são as de adição e de multiplicação.

Neste trabalho foi utilizado um grupo cíclico multiplicativo como conjunto dos números inteiros módulo n munido da operação de multiplicação, além do uso de um grupo multiplicativo.

2.8 GRUPOS BILINEARES

Grupos bilineares são grupos algébricos dos quais possuem as seguintes características:

1. $\mathbb{G}_1, \mathbb{G}_2$ e $\mathbb{G}_T$ grupos multiplicativos cíclicos de ordem prima p;
2. g_1 e g_2 são geradores de $\mathbb{G}_1$ e $\mathbb{G}_2$ respectivamente;
3. ψ é uma função de isomorfismo de $\mathbb{G}_1$ para $\mathbb{G}_2$, $\psi : \mathbb{G}_1 \longrightarrow \mathbb{G}_2$;
4. e é uma função de emparelhamento bilinear: $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$

É possível que $\mathbb{G}_1 = \mathbb{G}_2$, porém em casos gerais é permitido $\mathbb{G}_1 \neq \mathbb{G}_2$. Isto garante certa vantagem em algumas curvas elípticas para se obter assinaturas curtas (BONEH; BOYEN, 2004). Dado os grupos $\mathbb{G}_1$, $\mathbb{G}_2$ e $\mathbb{G}_T$, tal que $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T|$, a função de mapeamento bilinear $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ deve seguir a seguinte propriedade, para todo $u \in \mathbb{G}_1, v \in \mathbb{G}_2$ e $a, b \in \mathbb{Z}$:

$$e(u^a, v^b) = e(u, v)^{ab} \tag{2.6}$$

além de ser não-degenerado, ou seja:

$$e(g_1, g_2) \neq 1. \tag{2.7}$$

2.9 TRIPLA DIFFIE-HELLMAN

Seja $\mathbb{G}$ um grupo cíclico de ordem prima q com um gerador g. $\mathbb{G}$ é considerado um grupo que decide uma tripla Diffie-Hellman se existe um algoritmo eficiente que resolve o seguinte teste, que recebe de entrada uma tripla (g^a, g^b, g^c) e retorna 1 se e somente se, $c = ab \pmod{p}$. Caso o grupo $\mathbb{G}$ tenha está característica então podemos chamá-lo de um gap-DDH (GENNARO, 2004).

A tripla Diffie-Hellman pode ser resolvida a partir de um algoritmo implementando uma função de emparelhamento bilinear (Veja Seção 2.8), conforme foi demonstrado por Joux e Nguyen (JOUX; NGUYEN, 2001).

Algorithm 1 Algoritmo para resolver a Tripla Diffie-Hellman

1: **procedure** TRIPLE-DH
2: **if** $e(g^a, g^b) == e(g^c, g)$ **then**
3: **return** 1
4: **else**
5: **return** 0

2.10 STRONG DIFFIE-HELLMAN ASSUMPTION

Seja $\mathbb{G}_1$ e $\mathbb{G}_2$ dois grupos cíclicos de ordem prima p, g_1 e g_2 geradores de cada um dos grupos respectivamente, e possivelmente $\mathbb{G}_1 = \mathbb{G}_2$. Dado uma $(q+2)$-tupla $(g_1, g_2, g_2^x, g_2^{x^2}, g_2^{x^3}, ..., g_2^{x^q})$ como entrada, não é possível se obter como saída o par $(c, g_1^{\frac{1}{x+c}})$, onde $c \in \mathbb{Z}_p^*$.

2.11 ESQUEMA DE ASSINATURA DE SCHNORR

Esquemas de assinatura digitais de Schnorr são baseados na intratabilidade do problema do logaritmo discreto. Para tal esquema é utilizado um grupo cíclico multiplicativo $\mathbb{G}$ de ordem q, que possui um gerador g, além disso, o esquema faz uso de uma função de hash criptográfico H, que mapeia para os inteiros módulo q.

O par de chaves é definido como x um número aleatório sendo a chave privada do esquema e a chave pública é obtida através da seguinte equação $y = g^x$.

A assinatura pode ser gerada através do algoritmo 2.

Algorithm 2 Algoritmo para gerar uma assinatura de Schnorr

1: **procedure** SCHNORR SIGN(g, x, M)
2: Choose a randon k
3: Let $r = g^k$
4: Let $e = H(r||M)$
5: Let $s = k - xe$
6: **return** (s, e)

A verificação da assinatura pode ser realizada através do algoritmo 3.

Algorithm 3 Algoritmo para verificar uma assinatura de Schnorr

1: **procedure** SCHNORR VERIFY$(g, y, (s, e), M)$
2: Let $r_v = g^s y^e$
3: Let $e_v = H(r_v||M)$
4: **return** $e_v = e$

É possível observar a corretude do algoritmo, de modo que $r_v = g^s y^e = g^{k-xe} g^{xe} = g^k = r$, assim temos $e_v = H(r_v||M) = H(r||M) = e$, onde $||$ é uma operação de concatenação.

2.12 CONCLUSÃO

Neste capítulo foram introduzidos os conceitos fundamentais dos quais serão utilizados ao longo desse trabalho para a construção dos esquemas de resumo criptográficos camaleão. No próximo capítulo será introduzido o conceito, as operações e as propriedades do *hash* camaleão para que em seguida sejam construídos os esquemas, que utilizam conceitos fundamentados nesta seção, como os de RSA, grupos bilineares e assinaturas de Schnorr.

3 HASH CAMALEÃO

3.1 INTRODUÇÃO

Proposto inicialmente no ano de 1998 por Hugo Krawczyk e Tal Rabin (KRAWCZYK; RABIN, 1998) e amplificado por Giuseppe Ateniese e Breno de Medeiros em (ATENIESE; MEDEIROS, 2004b), o algoritmo de *hash* camaleão, permite que uma entidade confiável através de uma chave, chamada de *trapdoor*, possa determinar um segundo documento, diferente do primeiro, com o mesmo *hash*. Com isso, é possível modificar um documento previamente assinado sem invalidar a sua assinatura.

A Seção 3.2 apresenta o conceito de *hash* camaleão, suas funções e principais características para que se assemelhe a funções de *hash* tradicionais.

A Seção 3.3 introduz a função $CHash$ e seus parâmetros principais, que é a responsável pelo cômputo do resumo.

A Seção 3.4 introduz a função $CollisionFinding$ que é a que diferencia os esquemas de *hash* camaleão dos tradicionais, possibilitando encontrar uma colisão de forma segura.

A Seção 3.5 apresenta os requisitos que as funções de *hash* camaleão devem respeitar, para que o esquema seja considerado seguro para o uso em assinaturas digitais.

A Seção 3.6 introduz a ideia da utilização do algoritmo de *hash* camaleão em assinaturas digitais.

3.2 PROPRIEDADES

Uma função de *hash* camaleão está associada a um par de chaves criptográficas: uma chave pública e a respectiva chave privada. O termo "*hash* camaleão" foi escolhido pois o detentor da chave privada possui a capacidade de modificar a entrada da função de *hash* conforme desejar, mantendo a saída, ou seja, o mesmo *hash*.

Um esquema de *hash* camaleão deve garantir as seguintes três

propriedades:

i. Qualquer um que conheça a chave pública é capaz de computar o resumo de uma mensagem;

ii. Para os que não conhecem a chave privada, associada a um resumo previamente calculado para uma mensagem, deve ser computacionalmente muito difícil encontrar outra mensagem diferente que tenha o mesmo *hash*;

iii. Para aqueles que conhecem a chave privada, associada a um resumo previamente calculado para uma mensagem, existe um algoritmo para computar uma colisão, ou seja, encontrar outra mensagem que tenha o mesmo resumo.

Respeitando estas propriedades, a função de Hash Camaleão se assemelha a funções de *hashes* tradicionais. Agregando outras propriedades descritas ao final deste capítulo, é possível modificar esquemas padrões de assinaturas e adicionar a capacidade de encontrar colisões para os documentos assinados por estes esquemas.

3.3 COMPUTANDO O HASH

Para se computar um valor de Hash Camaleão h, primeiro é preciso definir um par de chaves. A chave pública é denotada por $\mathcal{PK}$ e a chave privada é denotada por $\mathcal{SK}$. Outro parâmetro necessário para a função de *hash* camaleão é um número r, escolhido de forma aleatória, para cada mensagem m que se deseja calcular o resumo. Dado esses parâmetros defini-se a função $CHash$

$$CHash(\mathcal{PK}, m, r) = h \tag{3.1}$$

Assim como as funções de resumo criptográfico tradicionais, a função $CHash$ deve respeitar o requisito de resistência a colisão, ou seja deve ser computacionalmente inviável para quem não conhece a chave privada associada a função de $CHash$ encontrar dois pares (m, r) e (m', r'), com $m \neq m'$, tal que

$$CHash(\mathcal{PK}, m, r) = CHash(\mathcal{PK}, m', r') \tag{3.2}$$

A função de *hash* camaleão deve ser uma função uniforme, ou seja, toda mensagem m induz a mesma probabilidade de distribuição

para $CHash(\mathcal{PK}, m, r)$ para r escolhido de forma aleatória. Assim, o valor do resumo criptográfico de um *hash* camaleão não revela nada sobre a mensagem da qual foi computada o resumo (KRAWCZYK; RABIN, 1998).

3.4 ENCONTRANDO COLISÕES

A principal diferença da função de *hash* camaleão para as funções de resumo criptográficos tradicionais é a possibilidade de se encontrar, através de um algoritmo eficiente, um valor de r', tal que o *hash* de m e m', utilizando a Equação 3.1, seja o mesmo.

Esse algoritmo é denominado de função $CollisionFinding$. Essa função permite determinar o valor de r' associado a um mensagem m', de tal forma que haja uma colisão de *hashes*.

A função $CollisionFinding$ faz uso da chave privada $\mathcal{SK}$, correspondente a chave pública $\mathcal{PK}$, utilizada no cômputo do *hash*. A função é assim definida

$$CollisionFinding(\mathcal{SK}, m, r, m') = r' \tag{3.3}$$

onde m e r são os valores utilizados para obter o resumo original, $\mathcal{SK}$ é a chave privada e m' é a mensagem que se deseja obter a colisão. O resultado da função $CollisionFinding$ é a variável aleatória r'.

Essa função também é conhecida como colisão *trapdoor*, uma vez que, só é possível determinar-se o valor de r' se a chave privada $\mathcal{SK}$ for conhecida. Sem essa chave privada, é computacionalmente difícil determinar-se o valor de r' e consequentemente, as colisões desejadas.

3.5 REQUISITOS DO *HASH* CAMELEÃO

Para que seja possível utilizar as funções de *hash* camaleão, em situações praticas, é necessário que os esquemas tenham algumas propriedades que garantam a sua eficácia e segurança. Essas propriedades são organizadas em duas classes. A primeira classe contém as propriedades relacionadas a função $CHash$ de cômputo do *hash*, e a segunda contém aquelas relacionadas a função $CollisionFinding$ utilizada para

encontrar as colisões.

3.5.1 Requisitos da função $CHash$

Os seguintes requisitos são pertinentes a função $CHash$.

a) **Compatibilidade** com esquemas de assinatura digital padrões já existentes. Esse requisito estabelece que deve ser possível utilizar o algoritmo de *hash* camaleão para verificar a integridade de dados nos esquemas de assinaturas tradicionais já difundidos (KRAWCZYK; RABIN, 2000);

b) **Resistência à colisão**: deve ser computacionalmente inviável se encontrar duas mensagens diferentes sem ter a posse da chave privada das quais resultem no mesmo valor de hash camaleão (KRAWCZYK; RABIN, 2000);

c) **Segurança semântica**: seja $H[X]$ a entropia de uma variável aleatória X, e $H[X|M]$ a entropia de X dado o valor de uma função aleatória Y de X. Segurança semântica é a afirmação de que a entropia de $H[m|C]$, onde m é uma mensagem e C uma valor de hash camaleão, é a mesma entropia de $H[m]$. Em outras palavras, um valor de resumo C do hash camaleão de uma mensagem m não revela nada a respeito da mensagem (ATENIESE; MEDEIROS, 2004b);

d) **Ocultação da mensagem**: Esse requisito consiste em afirmar que um atacante, mesmo que tenha sucesso em determinar uma colisão, não possui a capacidade de produzir novas colisões. Somente o detentor da chave privada tem essa capacidade. Assim, caso seja necessário, o detentor da chave privada pode mostrar que somente ele possui essa capacidade, gerando novas colisões (ATENIESE; MEDEIROS, 2004b);

e) **Uniformidade**: todas as mensagens m induzem a mesma probabilidade de distribuição para um r escolhido de forma aleatória (KRAWCZYK; RABIN, 2000).

3.5.2 Requisito da função $CollisionFinding$

A função $CollisionFinding$ deve ser livre de exposição de chave. Esse requisito estabelece que ao se produzir uma colisão para um par

(m, r), isso não implica em revelar qualquer informação relativa a chave secreta $\mathcal{SK}$. Assim, o atendimento a esse requisito garante que é possível encontrar colisões de forma segura.

3.6 UTILIZANDO HASHES CAMALEÃO EM ASSINATURAS DIGITAIS

Em um esquema de assinatura digital é possível substituir o algoritmo de resumo criptográfico pela função $CHash$. Para isso, a única alteração necessária é, além de adicionar o valor do *hash*, seria necessário armazenar o valor do parâmetro r e a chave pública $\mathcal{PK}$ do *hash* camaleão.

A chave pública do esquema de *hash* camaleão pode ser a mesma chave pública do signatário. Com isso, garante-se que o algoritmo de assinatura digital seja minimamente alterado.

Por sua vez, a função $CollisionFinding$ permite ao detentor da chave privada utilizada na função $CHash$, encontrar uma colisão para o resumo utilizado na assinatura, ou seja, permite modificar o documento assinado sem invalidar a assinatura.

3.7 CONCLUSÃO

Neste capítulo foram definido as funções do algoritmo de Hash Camaleão, seu funcionamento e suas principais propriedades para que seja possível a utilização do esquema em assinaturas digitais. No próximo capítulo sera mostrado o como realizar a implementação do algoritmo e sera analisado se as implementações do Hash Camaleão respeitam as características.

4 ESQUEMAS DE HASH CAMALEÃO

4.1 INTRODUÇÃO

Este capítulo tem como objetivo apresentar cinco dos mais referenciados esquemas de construção de algoritmos de *hash* camaleão encontrados na literatura. O processo de geração do par de chaves e da construção das funções $CHash$ e $CollisionFinding$ de cada um desses algoritmos são descritos. Também é feita uma análise de cada um desses esquemas em relação às propriedades e requisitos desejados de algoritmos de *hash* camaleão.

A Seção 4.2 apresenta um método de *hash* camaleão baseado nas primitivas do RSA.

A Seção 4.3 apresentado um esquema que faz uso da *Strong Diffie-Hellman Assumption*.

A Seção 4.4 apresenta dois esquemas de *hash* camaleão baseados em funções de emparelhamento bilinear.

A Seção 4.5 apresenta o último esquema de *hash* camaleão estudado, baseado no algoritmo de assinatura de Schnorr.

4.2 *HASH* CAMALEÃO BASEADO EM RSA

O primeiro esquema a ser analisado foi proposto por Giuseppe Ateniese e Breno de Medeiros em (ATENIESE; MEDEIROS, 2004b). Esse é baseado nas propriedades de chaves assimétricas do algoritmo RSA. O algoritmo RSA é apresentado na Seção 2.5, página 21 deste trabalho.

4.2.1 Geração de chaves

Nesse esquema de *hash* camaleão utilizamos um par de chaves RSA sem qualquer modificação em relação ao algoritmo RSA padrão. Seja esse par $(\mathcal{PK}, \mathcal{SK})$ onde a chave pública é a tupla $\mathcal{PK} = (e, n)$ e a chave privada é a tupla $\mathcal{SK} = (d, p, q)$.

4.2.2 Computando o *hash*

Seja a função de resumo criptográfico $H : \{0,1\}^* \longrightarrow \{0,1\}^t$, com entrada de qualquer quantidade de bits e saída de comprimento t. Existem muitas funções de *hash* propostas na literatura e em uso. Um exemplo é a função SHA-256 (DANG, 2015). Dado um par de chaves RSA $(\mathcal{PK}, \mathcal{SK})$, e S uma palavra identificadora, em sua essência S pode ser qualquer palavra, porém é utilizada como identificação da instância do *hash* camaleão. Após escolher a palavra S, pode-se então calcular o valor de $J = H(S)$. Então para se computar o valor de *hash* camaleão de uma mensagem m, basta escolher um valor aleatório r e em seguida calcular a seguinte expressão.

$$CHash(\mathcal{PK}, m, r) = J^{H(m)} r^e \pmod{n} \tag{4.1}$$

Assim se igualando a resumos criptográficos tradicionais em que qualquer pessoa possa verificar um valor de *hash* h para uma mensagem m e um valor aleatório r, basta computar o valor do resumo para (m, r) e verificar este é igual ao valor de h.

4.2.3 Encontrando colisões

Para encontrar uma colisão para um valor de *hash* previamente calculado basta computar um novo valor r' através da seguinte equação

$$\begin{aligned} CollisionFinding(SK, m, r, m') &= r J^{d(H(m)-H(m'))} \\ &= r' \pmod{n}. \end{aligned} \tag{4.2}$$

É possível se verificar que o par (m', r') é uma colisão para (m, r) realizando uma manipulação algébrica na formula de computar o algoritmo de *hash* camaleão, podendo se verificar a seguir:

$$\begin{aligned} CHash(\mathcal{PK}, m', r') &= J^{H(m')} r'^e \pmod{n} \\ &= J^{H(m')} (r J^{(H(m)-H(m'))d})^e \\ &= J^{H(m')} J^{(H(m)-H(m'))} r^e \\ &= J^{H(m)} r^e \\ &= CHash(\mathcal{PK}, m, r). \end{aligned} \tag{4.3}$$

4.2.4 Análise das propriedades

4.2.4.1 Propriedades da função $CHash$

Devido ao esquema de *hash* camaleão baseado em RSA utilizar apenas parâmetros públicos para computar o valor de resumo na função $CHash$, então é possível utilizar o algoritmo de *hash* camaleão baseado em RSA para verificar a integridade de documentos assinados, assim tendo **compatibilidade** com os esquemas de assinatura padronizados atualmente.

Para se encontrar uma colisão sem ter conhecimento da chave privada $\mathcal{SK}$ atrelada ao esquema, é necessário quebrar o equivalente a uma assinatura RSA em cima de J, assim, computar colisões para o esquema RSA sem o conhecimento da chave privada é computacionalmente impossível, tornando o esquema **resistente a colisão**.

Para cada mensagem m, o valor $h = CHash(\mathcal{PK}, m, r)$ é unicamente determinado por r, podemos dizer também que o valor de r é determinado por m também. Por serem variáveis aleatórias a entropia destas em relação ao valor de resumo h são iguais, assim respeitando a **segurança semântica**, além do uso de uma função de resumo criptográfico H também garantir a propriedade de **uniformidade**.

O detentor da chave privada $\mathcal{SK}$ pode computar uma colisão para qualquer par (m, r), sendo assim, pode calcular uma colisão para um suposto par (m', r'), como descrito na seção 3.5, para garantir a **ocultação da mensagem**.

4.2.4.2 Propriedades da função $CollisionFinding$

Dado duas assinaturas (m, r) e (m', r'), é possível computar α e β através do algoritmo de Euclides Estendido, tal que $\alpha(H(m) - H(m')) + \beta e = 1$. Então é possível utilizar α e β para encontrar J^d e calcular outras futuras colisões. Note que só é possível pois:

$$
\begin{aligned}
r' &= r J^{d(H(m)-H(m'))} \\
r'/r &= J^{d(H(m)-H(m'))} \\
(\tfrac{r'}{r})^{\alpha} &= (J^{d(H(m)-H(m'))})^{\frac{1-\beta e}{H(m)-H(m')}} \\
J^{d} &= (\tfrac{r'}{r})^{\alpha} J^{\beta}.
\end{aligned}
$$

Assim, o esquema de *hash* camaleão RSA não respeita a propriedade de **livre exposição de chaves**.

4.2.5 Implementação

A implementação do algoritmo de *hash* camaleão baseado em RSA e suas funções foi realizada utilizando a biblioteca Sage e está disponível no Anexo A. Neste anexo se encontra a demonstração de que ao se revelar uma colisão é possível encontrar outras colisões sem descobrir a chave privada.

4.3 HASH CAMALEÃO BASEADO NA *SDH-ASSUMPTION*

Giuseppe Ateniese e Breno de Medeiros em (ATENIESE; MEDEIROS, 2004b) propuseram um novo esquema de *hash* camaleão. Esse esquema utiliza as propriedades do *gap-DDH* e é considerado seguro se respeitar a *SDH-Assumption*.

4.3.1 Configuração

Seja $\mathbb{G}$ um *gap-DDH* de ordem prima p, com um gerador g e uma função de emparelhamento bilinear e.

4.3.2 Geração de chaves

Determine a chave privada $\mathcal{SK}$ (nessa seção utilizaremos a letra x para designar $\mathcal{SK}$), tal que $x \in \mathbb{Z}_p$, e a chave pública $\mathcal{PK}$ (nessa seção utilizaremos a letra h para designar $\mathcal{PK}$), tal que $h = g^x$.

4.3.3 O esquema de resumo

O esquema baseado na *SDH-Assumption* faz uso de uma função de resumo criptográfica tradicional $H : \{0,1\}^* \longrightarrow \{0,1\}^t$ e de uma palavra identificadora S.

Primeiramente é calculado o *hash* a partir desse identificador, sendo ele $c = H(S)$. Então dado uma mensagem m, basta gerar um número aleatório $r \in \mathbb{Z}_p$. O valor do *hash* camaleão para o par (m, g^r) é dado pela equação:

$$CHash(h = podes\mathcal{PK}, m, r) = g^{H(m)}(g^c h)^r. \tag{4.4}$$

Para verificar se a mensagem m corresponde ao valor de *hash* para um dado r, é necessário checar se $(g^r, g^c h, F)$ é uma tripla Diffie-Hellman, onde F é o valor do *hash* camaleão dividido por $g^{H(m)}$, ou seja $F = (g^c h)^r$. É possível realizar a implementação de um algoritmo que verifique a tripla DH utilizando uma função de emparelhamento bilinear, conforme descrito na Seção 2.8, ou seja, se a seguinte igualdade entre os emparelhamentos é verdadeira:

$$e(g^r, g^c h) = e((g^c h)^r, g). \tag{4.5}$$

Pode-se mostrar que, caso o *hash* seja correto, $(g^r, g^c h, F)$ é uma tripla DH, ou seja:

$$\begin{aligned}(g^r, g^c h, F) &= (g^r, g^x g^c, (g^c h)^r) \\ &= (g^r, g^{x+c}, (g^c g^x)^r) \\ &= (g^r, g^{x+c}, g^{(x+c)r}).\end{aligned} \tag{4.6}$$

Assim, o esquema baseado na *Strong Diffie-Hellman Assumption* permite que qualquer entidade possa calcular um valor de resumo para uma mensagem m, e também permite a verificação do resumo, ou seja, se uma mensagem m gera um valor de *hash* recebido.

4.3.4 Encontrando colisões

Para encontrar uma colisão m', tal que $m' \neq m$, para um valor de resumo gerado anteriormente pelo par (m, g^r) é necessário utilizar a

chave privada $\mathcal{SK}$ para calcular o valor de $g^{r'}$ de modo que o valor de *hash* camaleão para

$$\begin{aligned} CollisionFinding(x = \mathcal{SK}, m, r, m') &= g^r g^{\frac{H(m)-H(m')}{x+c}} \\ &= g^{r'} \end{aligned} \tag{4.7}$$

Para verificar que $g^{r'}$ e m' geram uma colisão, é necessário verificar se a propriedade da tripla DH ainda é respeitada:

$$\begin{aligned} (g^{r'}, g^c h, F') &= (g^{r'}, g^{x+c}, (g^{x+c})^{r'}) \\ &= (g^{r'}, g^{x+c}, (g^{r'})^{x+c}) \\ &= (g^{r'}, g^{x+c}, (g^r g^{\frac{H(m)-H(m')}{x+c}})^{x+c}) \\ &= (g^r g^{\frac{H(m)-H(m')}{x+c}}, g^{x+c}, g^{r(x+c)} g^{H(m)-H(m')}). \end{aligned}$$

É possível demonstrar algebricamente que um resumo de (m', r') é equivalente a um resumo obtido com (m, r), apesar de ser computacionalmente inviável computar r' pelo problema do logaritmo discreto. Para provar essa igualdade, primeiramente é necessário modificar a equação $CHash$, deixando ela em fatores de g^r de tal forma que a nova função $PrivCHash$ irá necessitar da chave privada para computar o valor do *hash*

$$\begin{aligned} CHash(\mathcal{PK}, m, r) &= g^{H(m)}(g^c h)^r \\ &= g^{H(m)} g^{rc} g^{rx} \\ &= g^{H(m)} g^{r(x+c)} \\ PrivCHash(\mathcal{SK}, m, g^r) &= g^{H(m)} g^{r(x+c)}. \end{aligned} \tag{4.8}$$

Assim, é possível calcular o valor de resumo a partir de g^r, m, e x. Então, pode-se provar que $(m', g^{r'})$ é uma colisão para o par (m, r), como visto abaixo

$$
\begin{aligned}
PrivCHash(\mathcal{SK}, m, g^{r'}) &= g^{H(m')} g^{r'(x+c)} \\
&= g^{H(m')} (g^r g^{\frac{H(m)-H(m')}{x+c}})^{(x+c)} \\
&= g^{H(m')} g^{r(x+c)} g^{H(m)-H(m')} \\
&= g^{H(m)} g^{r(x+c)} \\
&= g^{H(m)} g^{rc} g^{rx} \\
&= g^{H(m)} (g^c h)^r \\
&= CHash(\mathcal{PK}, m, r).
\end{aligned}
\tag{4.9}
$$

4.3.5 Análise das propriedades

4.3.5.1 Propriedades da função $CHash$

Igualmente como no esquema de *hash* camaleão baseado em RSA, o esquema do algoritmo de *hash* camaleão baseado na *Strong Diffie-Hellman Assumption* também só faz uso de atributos públicos para computar um resumo, tornando este **compatível** com esquemas de assinaturas tradicionais. Ademais, este esquema respeita as propriedades de **segurança semântica** e de **uniformidade** pelos mesmos motivos citados na análise das propriedades do esquema RSA, por fazer uso de duas variáveis aleatórias, m e r, e de uma função de *hash* tradicional H, que garante uniformidade em sua saída.

Sua segurança na parte de **resistência a colisões** está atrelada na dificuldade de se computar $g^{\frac{1}{x+c}}$, que no geral é um problema computacionalmente difícil (BONEH; BOYEN, 2004).

Como o detentor da chave privada $\mathcal{SK}$ pode computar uma colisão para qualquer par (m, g^r), então é garantida a propriedade de **ocultação da mensagem**.

4.3.5.2 Propriedades da função $CollisionFinding$

A partir da exposição de uma colisão para o esquema do *hash* camaleão baseado na *SDH-assumption*, (m, g^r) e $(m', g^{r'})$, é possível

então computar de forma eficiente $g^{\frac{1}{x+c}}$, equivalente ao problema computacional difícil sob a *Strong Diffie-Hellman Assumption*. Porém, caso seja escolhida uma palavra identificadora c diferente para cada vez da qual for retirar um resumo, obtemos um esquema onde a propriedade de **livre exposição de chave** é respeitada (ATENIESE; MEDEIROS, 2004b), pois ele se torna equivalente ao esquema demonstrado em (BONEH; BOYEN, 2004).

4.3.6 Implementação

A implementação do algoritmo de *hash* camaleão baseado na *Strong Diffie-Hellman Assumption* e suas funções foi realizada utilizando a biblioteca Sage e está disponível no Anexo B. Para tal implementação, foi utilizado como o grupo $\mathbb{G}$ uma extensão de curva (HEUBERGER; MAZZOLI, 2017). Apesar desta não apresentar segurança criptográfica, é possível observar a corretude do esquema de algoritmo de resumo criptográfico camaleão baseado na *SDH-Assumption*.

Para realizar implementações mais robustas, é necessário utilizar um grupo $\mathbb{G}$ o qual seja criptograficamente seguro (BLAKE; SEROUSSI; SMART, 1999). Porém, nem todas curvas elípticas aceitam uma função de emparelhamento bilinear; para tal, Barreto e Naehrig propuseram curvas das quais se é possível se realizar *pairing* de forma segura (BARRETO; NAEHRIG, 2005). Outra proposta de curvas são as curvas MNT, propostas em (PAGE; SMART; VERCAUTEREN, 2006). Para tal implementação do esquema foi utilizado o algoritmo de Tate Pairing (BARRETO et al., 2002).

4.4 *HASH* CAMALEÃO BASEADO EM EMPARELHAMENTO BILINEAR

Baseado apenas em funções de mapeamento bilineares sob grupos, (ZHANG; SAFAVI-NAINI; SUSILO, 2003) propõe duas formas de implementar as funções do algoritmo de *hash* camaleão. Ambas as formas são baseadas em identificadores para o detentor da chave, como em outros esquemas de *hash* camaleão, que criptograficamente podem ser tratados apenas como um número. Porém, estes identificadores agregam um certo nível de segurança semântica ao esquema (ATENIESE; MEDEIROS, 2004a).

4.4.1 Esquemas 1 e 2: Configuração

Seja $\mathbb{G}_1$ um grupo cíclico aditivo com um gerador P, tal que P tenha ordem prima q e seja $\mathbb{G}_2$ um grupo cíclico multiplicativo com a mesma ordem. Tome e uma função de emparelhamento bilinear de $\mathbb{G}_1$ para $\mathbb{G}_2$ tal que $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, $e(aP, bQ) = e(P, Q)^{ab} \ \forall \ P, Q \in \mathbb{G}_1$ e $a, b \in \mathbb{Z}$. O esquema faz uso de uma função de resumo criptográfico tradicional H, como nos esquemas anteriormente descritos e faz uso de uma função de resumo criptográfico H' tal que $H' : \{0,1\}^* \to \mathbb{G}_1^*$.

4.4.2 Esquema 1: Geração de chaves

Primeiramente, é necessário definir um ID (sugere-se algo que represente a entidade, como um e-mail ou pseudônimo). Em seguida, é preciso definir s tal que $s \in \mathbb{Z}_q^*$; este é considerado a chave privada mestre do esquema. Assim, é possível calcular a chave pública através de s, tal que $P_{pub} = sP$. Por fim, a chave privada $\mathcal{SK}$ é computada, outrora chamada de $S_{id} = sH'(ID)$. Então, o par de chaves consiste de $\mathcal{PK} = P_{pub}$ e $\mathcal{SK} = S_{id}$; é importante salientar que a chave privada do esquema está atrelada diretamente com o valor do ID supracitado.

4.4.3 Esquema 1: O esquema de resumo

Com posse da chave pública gerada anteriormente e da palavra ID, basta escolher aleatoriamente $R \in \mathbb{G}_1$ e então computar a seguinte expressão:

$$CHash(\mathcal{PK}, ID, m, R) = e(R, P)e(H(m)H'(ID), P_{pub}). \qquad (4.10)$$

4.4.4 Esquema 1: Encontrando colisões

Para encontrar uma colisão m' para uma mensagem m, e um valor aleatório $R \in \mathbb{G}_1$, do qual foi computado um valor de resumo de *hash* camaleão anteriormente, é necessário encontrar um valor de R' através da seguinte fórmula:

$$CollisionFinding(\mathcal{PK}, ID, \mathcal{PK}, m, R, m') = R' \\ = (H(m) - H(m'))S_{id} + R. \tag{4.11}$$

Para verificar se o par (m', R') é uma colisão para (m, R) é necessário observar se um valor de *hash* obtido através do primeiro par é equivalente a um resumo com o segundo par, o que pode ser verificado a seguir:

$$\begin{aligned} &CHash(\mathcal{PK}, ID, m', R') = e(R', P)e(H(m')H'(ID), P_{pub}) \\ &= e((H(m) - H(m'))S_{id} + R, P)e(H(m')H'(ID), P_{pub}) \\ &= e((H(m) - H(m'))S_{id}, P)e(R, P)e(H(m')H'(ID), P_{pub}) \\ &= e((H(m) - H(m'))sH'(ID), P)e(R, P)e(H(m')H'(ID), P_{pub}) \\ &= e((H(m) - H(m'))H'(ID), P_{pub})e(R, P)e(H(m')H'(ID), P_{pub}) \\ &= e(H(m)H(ID) - H(m')H'(ID), P_{pub})e(R, P)e(H(m')H'(ID), P_{pub}) \\ &= e(R, P)e(H(m)H'(ID), P_{pub}) \\ &= CHash(\mathcal{PK}, ID, m, R). \end{aligned} \tag{4.12}$$

4.4.5 Esquema 2: Geração de chaves

Igualmente como no esquema 1, a chave pública $\mathcal{PK}$ é calculada utilizando uma chave mestre s e o gerador de $\mathbb{G}_1$, tal que $\mathcal{PK} = P_{pub} = sP$. Por sua vez, a obtenção da chave privada é modificada, mas esta ainda depende diretamente do ID definido para a entidade. Para obter a chave privada no segundo esquema proposto, basta computar $\mathcal{SK} = S_{id} = \frac{1}{s+H(ID)}P$.

4.4.6 Esquema 2: O esquema de resumo

Para computar um valor de resumo para uma mensagem m, o usuário deve calcular aleatoriamente um valor de $R \in \mathbb{G}_1$ como no

esquema 1 e calcular o valor de resumo da seguinte maneira:

$$CHash(\mathcal{PK}, ID, m, R) = e(P, P)^{H(m)} e(H(ID)P + P_{pub}, R)^{H(m)}. \tag{4.13}$$

4.4.7 Esquema 2: Encontrando colisões

Para calcular uma colisão para um par (m, R), é necessário que o detentor da chave privada compute um novo valor de R' para o valor de m' desejado da seguinte maneira:

$$\begin{aligned} &CollisionFinding(\mathcal{SK}, ID, m, R, m') = R \\ &= H(m')^{-1}((H(m) - H(m'))S_{id} + H(m)R). \end{aligned} \tag{4.14}$$

Para verificar se o valor de R' encontrado para m' satisfaz uma colisão para o par (m, R) é necessário mostrar que ambas as tuplas produzam o mesmo valor de resumo, como demonstrado a seguir:

$$\begin{aligned} &CHash(\mathcal{PK}, ID, m', R') = e(R', P)e(H(m')H'(ID), P_{pub}) \\ &= e((H(m) - H(m'))S_{id} + R, P)e(H(m')H'(ID), P_{pub}) \\ &= e((H(m) - H(m'))S_{id}, P)e(R, P)e(H(m')H'(ID), P_{pub}) \\ &= e((H(m) - H(m'))sH'(ID), P)e(R, P)e(H(m')H'(ID), P_{pub}) \\ &= e((H(m) - H(m'))H'(ID), P_{pub})e(R, P)e(H(m')H'(ID), P_{pub}) \\ &= e(H(m)H(ID) - H(m')H'(ID), P_{pub})e(R, P)e(H(m')H'(ID), P_{pub}) \\ &= e(R, P)e(H(m)H'(ID), P_{pub}) \\ &= CHash(\mathcal{PK}, ID, m, R). \end{aligned} \tag{4.15}$$

4.4.8 Análise das propriedades

4.4.8.1 Propriedades das funções $CHash$

Para o cálculo do resumo criptográfico nos esquemas baseados em emparelhamento bilinear, são utilizados duas funções de *pairings* sobre parâmetros públicos do esquema e da mensagem, assim, tornando-o **compatível** com assinaturas digitais.

Para o primeiro esquema apresentado, o processo de extração de chave é descrito em (BONEH; LYNN; SHACHAM, 2001), o qual é seguro contra ataques que tentem forjar a assinatura, assumindo que o problema computacional de Diffie-Hellman (CDH) é difícil de ser resolvido.

O segundo faz uso do esquema de assinatura proposto em (SAKAI; KASAHARA, 2003), o qual é considerado seguro, assumindo a dificuldade de resolver o *k-weak* problema computacional Diffie-Hellman (k-wCDHP) é difícil de ser resolvido (ZHANG; SAFAVI-NAINI; SUSILO, 2003), o que torna ambos os esquemas **resistentes a colisões**.

Os esquemas 1 e 2, são ambos semanticamente seguros, pois o ID do usuário, e a mensagem não são relacionadas a variável aleatória $R \in \mathbb{G}_1$, e a não degeneração do emparelhamento bilinear faz com que a aleatoriedade de R torne os esquemas **semanticamente seguros** (ZHANG; SAFAVI-NAINI; SUSILO, 2003), além da **uniformidade** ser garantida pela função de *hash* H.

4.4.8.2 Propriedades da funções $CollisionFinding$

Pelas propriedades dos emparelhamentos bilineares, é possível realizar uma manipulação algébrica na igualdade de um valor de resumo e de uma colisão para descobrir a chave privada $\mathcal{SK}$. No primeiro esquema, manipulamos a seguinte igualdade:

$$\begin{aligned} &CHash(\mathcal{PK}, ID, m', R') = CHash(\mathcal{PK}, ID, m, R) \\ &e(R, P)e(H(m)H'(ID), P_{pub}) = e(R', P)e(H(m')H'(ID), P_{pub}) \\ &e(R - R', P) = e((H(m) - H(m'))H'(ID), P_{pub}) \\ &e(R - R', P) = e((H(m) - H(m'))S_{id}, P). \end{aligned} \tag{4.16}$$

Então a chave privada $\mathcal{SK} = S_{id}$ é obtida por

$$S_{id} = (H(m') - H(m))^{-1}(R - R'). \tag{4.17}$$

No segundo esquema baseado em emparelhamento bilinear tam-

bém é possível descobrir a chave privada, manipulando a igualdade

$$\begin{aligned} &CHash(P_{pub}, ID.m, R) = CHash(P_{pub}, ID.m', R') \\ &= e(P,P)^{H(m)} e(H(ID)P + P_{pub}, R)^{H(m)} \\ &= e(P,P)^{H(m')} e(H(ID)P + P_{pub}, R')^{H(m')} \\ &= e(P,P)^{H(m)-H(m')} e(H(ID)P + P_{pub}, H(m')R' - H(m)R). \end{aligned} \tag{4.18}$$

Então a chave privada $\mathcal{SK} = S_{id}$ é obtida por

$$S_{id} = (H(m') - H(m))^{-1}(H(m')R' - H(m)R). \tag{4.19}$$

Como provado na proposta em (ZHANG; SAFAVI-NAINI; SUSILO, 2003), o esquema de resumo criptográfico camaleão baseado em emparelhamento bilinear não é **livre de exposição de chave**.

4.5 ESQUEMA BASEADOS EM ASSINATURAS DE SCHNORR

O esquema proposto em (GAO; LI; WANG, 2009), é baseado no esquema de assinatura de Schnorr, no qual sua segurança é baseada na intratabilidade do problema do logaritmo discreto.

4.5.1 Configuração

Seja $\mathbb{G}$ um grupo cíclico multiplicativo com um gerador g, tal que g tenha ordem prima q, além disso, o esquema faz uso de uma função de resumo criptográfico tradicional H.

4.5.2 Geração de chaves

Similar ao esquema baseado na *SDH-Assumption*, para computar a chave privada $\mathcal{SK}$ basta escolher aleatoriamente $x \in \mathbb{Z}_q$, então a chave pública pode ser obtida através do gerador g e da chave privada x, tal que $\mathcal{PK} = y = g^x$.

4.5.3 O esquema de resumo

Primeiramente, para calcular o resumo camaleão de uma mensagem é necessário definir uma etiqueta L, além de definir dois números aleatórios $t_1,\ r_2 \in \mathbb{Z}_q$. Inicialmente então calcula-se r_1, c_1 e S_1 da seguinte maneira:

$$\begin{aligned} r_1 &= g^{t_1} \\ c_1 &= H(L; r_1) \\ S_1 &= r_1 y^{c_1}. \end{aligned} \tag{4.20}$$

Por fim, a equação para calcular o resumo é dada por:

$$CHash(\mathcal{PK}, L, m, r_2) = S_2 = g^m S_1^{r_2}. \tag{4.21}$$

4.5.4 Encontrado Colisões

Caso o detentor da chave privada $\mathcal{SK}$ deseje encontrar uma colisão m' para um par (m, r_2) do qual se tenha computado um valor de resumo anteriormente, primeiro é necessário encontrar o valor de s_1, que é a *ephemeral trapdoor*, como descrito na seção 2.11

$$s_1 = t_1 + xH(L; r_1) \pmod{q}. \tag{4.22}$$

Dado o valor de s_1 então é possível calcular o valor de r_2' para que m' seja uma colisão através da seguinte equação:

$$\begin{aligned} CollisionFinding(\mathcal{SK}, s_1, m, r_2, m') &= r_2' \\ &= s_1^{-1}(m - m') + r_2 \pmod{q}. \end{aligned} \tag{4.23}$$

Para verificar se (m', r_2') colide com (m, r_2), substitui-se os termos das equações até que $CHash(\mathcal{PK}, L, m', r_2') = CHash(\mathcal{PK}, L, m, r_2)$

$$
\begin{aligned}
CHash(\mathcal{PK}, L, m', r_2') &= S_2' \\
&= g^{m'} S_1^{r_2'} \\
&= g^{m'} S_1^{s_1^{-1}(m-m')+r_2} \\
&= g^{m'} S_1^{(t_1+xH(L;r_1))^{-1}(m-m')+r_2} \\
&= g^{m'} (g^{t_1+xH(L;r_1)})^{(t_1+xH(L;r_1))^{-1}(m-m')} S_1^{r_2} \\
&= g^{m'} g^{m-m'} S_1^{r_2} \\
&= g^{m} S_1^{r_2} \\
&= CHash(\mathcal{PK}, L, m, r_2).
\end{aligned}
\tag{4.24}
$$

4.5.5 Análise das propriedades

4.5.5.1 Propriedades da função $CHash$

Igualmente como em outros esquemas apresentados nesta seção, o algoritmo de *hash* camaleão baseado em assinaturas de Schnorr respeita a propriedade de **compatibilidade**, pois qualquer um com conhecimento dos parâmetros públicos do esquema tem capacidade de computar um valor de resumo para uma mensagem m.

Para o conhecedor da chave privada $\mathcal{SK}$ é possível, através da fórmula para se encontrar a *ephemeral trapdoor* e da $CollisionFinding$, encontrar uma colisão, provendo a **ocultação da mensagem**; porém, para um atacante que não tenha conhecimento da chave privada, como o par (r_1, s_1) é equivalente a uma assinatura de Schnorr sobre a etiqueta L, encontrar uma colisão é equivalente a quebrar esta assinatura (GAO; LI; WANG, 2009), tornando o esquema **resistente a colisões**.

Ao computar um valor de resumo, além do valor da mensagem m, e da variável de r_2 serem aleatórios, o esquema de *hash* camaleão baseado em assinaturas de Schnorr faz uso de uma chave efêmera, possuindo assim **segurança semântica**, pois o conteúdo das variáveis é sempre independente; ademais, este esquema também faz uso de uma função de resumo criptográfico tradicional, garantindo a característica da **uniformidade**.

4.5.5.2 Propriedades da função *CollisionFinding*

Como para um atacante sem conhecimento da chave privada revelar colisões é equivalente a quebrar o esquema de assinatura de Schnorr, o esquema é seguro contra **exposição de chaves** e de ataques ativos conhecidos (GAO; LI; WANG, 2009).

4.6 CONCLUSÃO

Este capítulo apresentou os principais esquemas de resumo criptográfico camaleão encontrados na literatura, analisando a forma com que é calculado o valor do resumo, as equações para se encontrar colisões, verificando se o valor de resumo original e de uma colisão são os mesmos e analisando as principais propriedades encontradas na literatura.

5 AVALIAÇÃO E COMPARAÇÃO DE HASHES CAMALEÃO E SEU USO PARA ASSINAR DOCUMENTOS ELETRÔNICOS

5.1 INTRODUÇÃO

Após a apresentação dos esquemas de resumo criptográfico camaleão, será feito neste capítulo uma análise comparativa entre estes e suas características. Em seguida, é demonstrado como é o funcionamento de um esquema de *hash* camaleão em um esquema de assinatura digital.

A Seção 5.2 apresenta uma comparação dos algoritmos de *hash* camaleão, analisando suas características e requisitos.

Na Seção 5.3 é proposto um esquemático de como utilizar o algoritmo de *hash* camaleão para assinar, verificar e modificar documentos assinados.

5.2 ANÁLISE GERAL DOS ESQUEMAS DE *HASH* CAMALEÃO

Foram analisados no capítulo 4 cinco esquemas de resumo criptográfico camaleão; um baseado na criptografia do RSA, um baseado na *Strong Diffie-Hellman Assumption*, dois esquemas baseado em emparelhamento bilinear e por fim um esquema baseado em assinaturas de Schnorr. Para poder padronizar a forma de realizar a análise, primeiro foram definidas as funções que um esquema de *hash* camaleão precisa ter e suas finalidades:

- Geração de chaves: responsável por definir o par de chaves utilizado no esquema;
- Computando o *hash*: onde é definido a função $CHash(\cdot)$, que é a responsável pode calcular o valor de resumo;
- Encontrando colisões: que define a função $CollisionFinding(\cdot)$, que através dela é possível se encontrar uma colisão entre duas mensagem para o esquema.

Para todos os esquemas citados, foi demonstrado que existe uma colisão entre duas mensagens m, m', tais que $m \neq m'$, ou seja, os va-

lores de resumo criados aplicando a função $CHash$ a estas mensagens são iguais.

Para analisar a viabilidade da utilização dos esquemas no âmbito de assinaturas digitais, foi feito um levantamento dos principais artigos e selecionadas as principais características que um esquema camaleão necessita respeitar, e para todas estas propriedades, descritas na seção 3.5, foi analisado em cada esquema se é respeitada, como pode ser observado pela tabela a seguir:

Tabela 1 – Comparação das propriedades dos esquemas do algoritmo de hash camaleão

	RSA	**SDH**	**Pairing 1**	**Pairing 2**	**Schnorr**
Compatibilidade	✓	✓	✓	✓	✓
Resistência a colisão	✓	✓	✓	✓	✓
Segurança semântica	✓	✓	✓	✓	✓
Ocultação da mensagem	✓	✓	✓	✓	✓
Uniformidade	✓	✓	✓	✓	✓
Livre de exposição de chave		✓			✓

A propriedade de compatibilidade diz respeito à capacidade do esquema de ser utilizado em esquemas de assinaturas digitais como uma função de resumo criptográfico; para isso, deve ser possível que qualquer pessoa possa calcular um valor de resumo para qualquer mensagem.

A resistência a colisão serve para que os esquemas sejam equivalentes a outros algoritmos de *hash* criptográficos, evitando ataques onde é possível se encontrar duas mensagens com o mesmo valor de resumo, embora esquemas de *hash* camaleão possibilitem colisões, é imprescindível que apenas o detentor da chave privada do esquema consiga obter tal colisão e que isto seja possível somente através da função $CollisionFinding$.

As propriedades de segurança semântica, ocultação da mensagem e uniformidade precisam ser respeitadas para que os esquemas tenham o mesmo comportamento dos algoritmos normais, dissociando o resumo do conteúdo da mensagem, tornando o valor final do resumo

similar a um número aleatório.

Por fim, a última propriedade analisada é a garantia da liberdade de exposição de chave, ou seja, a chave privada não será revelada no cômputo de uma colisão. O esquema baseado em RSA (Seção 4.2) e os dois esquemas baseados em emparelhamento bilinear (Seção 4.4) não garantem esta propriedade, pois é possível realizar uma manipulação algébrica na estrutura de ambos os esquemas e, consequentemente, encontrar o valor da chave privada. Devido a estas características, restam apenas dois esquemas nos quais são possíveis encontrar colisões e manter a chave privada segura: o esquema baseado na *Strong Diffie-Hellman Assumption* (Seção 4.3) e o esquema baseado nas assinaturas de Schnorr (Seção 4.5).

5.3 UTILIZAÇÃO EM ESQUEMAS DE ASSINATURAS DIGITAIS

Após a análise dos requisitos realizada, é possível utilizar um esquema de resumo criptográfico camaleão em esquemas de assinaturas digitais, desde que este respeite todas as propriedades analisadas anteriormente, tornando possível modificar o documento assinado sem invalidar a assinatura e sem a necessidade de realizar uma nova assinatura.

Para realizar uma assinatura digital do documento utilizando o algoritmo de *hash* camaleão, o assinante primeiro precisa definir o par de chaves $(\mathcal{PK}, \mathcal{SK})$ utilizado no esquema. A chave pública deve estar acessível a todos que necessitem verificar a assinatura: para isso, deve ser criado um campo específico na assinatura para armazená-la, além de outros parâmetros dos esquemas, como palavras identificadoras. Por fim, para computar o resumo, basta apenas definir o valor aleatório r, e calcular o valor da equação $CHash$ para o documento. Dado o valor de resumo criptográfico calculado, em seguida basta realizar a cifra com a chave privada do assinante e a assinatura está concluída. O esquema descrito na Figura 1 descreve o fluxo da assinatura.

Assim, a construção da assinatura mantém-se de forma similar à de esquemas de assinatura tradicionais, apenas com a ressalva de que os parâmetros do algoritmo de resumo criptográfico camaleão precisarão ser disponibilizados assinatura, para que seja possível afirmar a inte-

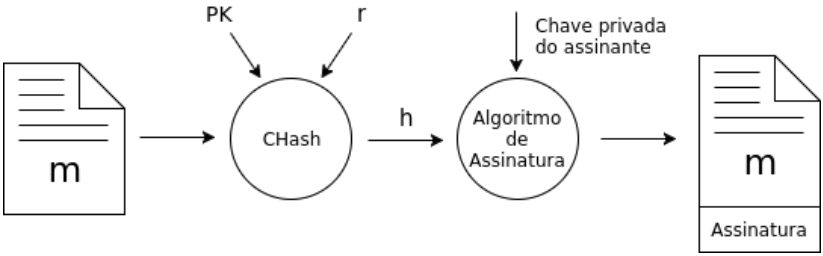

Figura 1 – Diagrama do fluxo de assinatura de um documento digital com *hash* camaleão

gridade do documento no processo de verificação da assinatura.

Caso o detentor da chave privada, associada a chave pública utilizada para se calcular o resumo original da assinatura, deseje alterar o documento, este deve passar para a função $CollisionFinding$ os parâmetros m e r que foram utilizados para gerar o primeiro resumo, além da chave privada $\mathcal{SK}$. A saída r', junto com o novo documento m', são uma colisão para o documento original m. Assim, basta armazenar junto à assinatura o novo valor de r' e a assinatura para o novo documento será válida. A Figura 2 mostra o fluxo de modificação do documento, a fim de montar a nova assinatura, sem a necessidade do uso da chave privada do assinante.

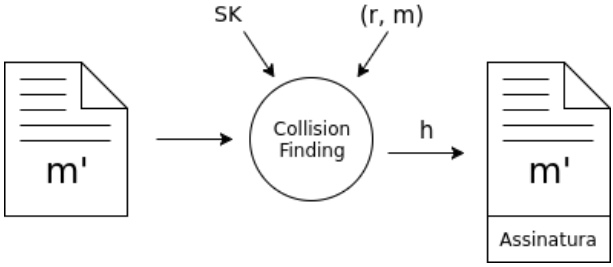

Figura 2 – Diagrama do fluxo da utilização da função encontrando colisões do *hash* camaleão

Para realizar a validação da assinatura, é preciso primeiro computar o valor do *hash* camaleão. Para isso é necessário buscar a chave pública $\mathcal{PK}$ e o valor de r na assinatura. Em seguida, é computado o valor do resumo criptográfico camaleão para o arquivo; este deve ser

igual ao valor da assinatura após ter sido decifrado pela chave pública do assinante, caso contrário a assinatura é inválida. Este processo indepente se a assinatura foi gerada com ou sem colisões. A Figura 3 mostra o fluxo de validação de uma assinatura que utiliza o algoritmo de resumo camaleão.

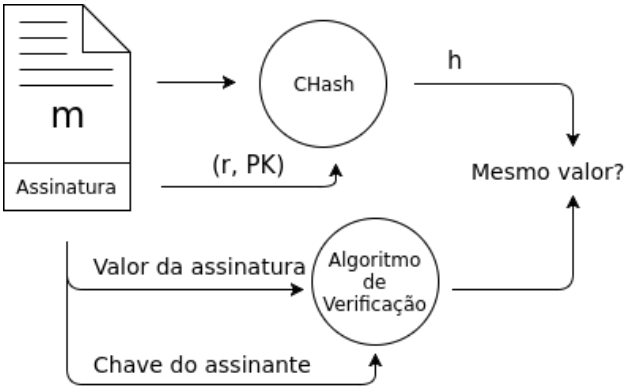

Figura 3 – Diagrama do fluxo da validação de uma assinatura digital utilizando *hash* camaleão.

5.4 CONCLUSÃO

Neste capítulo foram analisados todos os esquemas apresentados no capítulo anterior, avaliando as propriedades para que estes possam ser utilizados com segurança no âmbito de assinaturas digitais. Visto que dois esquemas podem ser utilizados como função de resumo criptográfico foi esquematizado o uso do *hash* camaleão em assinaturas digitais, mostrando como seria realizado a assinatura e a verificação, além de demonstrar como modificar o documento assinado sem invalidar a assinatura através da função encontrando colisões.

6 CONSIDERAÇÕES FINAIS

Este trabalho apresentou o algoritmo de resumo criptográfico camaleão, visando aumentar o escopo de assinaturas digitais resolvendo o problema introduzido na Seção 1, do qual um médico ao ter a necessidade de publicar um prontuário eletrônico assinado digitalmente de um paciente, ao modificar o prontuário para omitir o nome do paciente por questões de privacidade tem a necessidade de assinar novamente o prontuário, pois a modificação acaba por invalidar a assinatura. Este problema acontece devido aos algoritmos de resumo criptográficos tradicionais utilizados nos esquemas de assinatura não permitirem uma modificação do conteúdo assinado sem que se modifique a saída do algoritmo.

Para tal, foi apresentado o algoritmo de *hash* camaleão, que associa um par de chaves ao esquema de resumo criptográfico, onde qualquer um com o conhecimento da chave pública consegue calcular um valor de resumo para um documento eletrônico, e o detentor da chave privada consegue encontrar uma colisão para um documento do qual foi retirado um resumo previamente.

Inicialmente foi analisado a proposta do algoritmo, quais as funções um esquema deve ter e quais devem ser o comportamento destas funções, selecionando as principais características encontradas na literatura.

Após definir conceitualmente foram apresentados cinco formas de se construir um esquema de *hash* camaleão, demonstrando sua geração de chaves, a obtenção do resumo e da colisão, analisando a corretude da colisão e as principais características, estas voltadas a utilização destes esquemas em assinaturas digitais.

Dado a definição e a análise dos esquemas apresentados foi realizado uma comparação de todos os esquemas, levando em consideração as propriedades analisadas, concluindo que apenas dois dos cinco esquemas apresentados são efetivamente factíveis de se utilizar em um esquema assinatura digital.

Por fim foi proposto a estrutura de um esquema de assinatura digital, capaz de assinar, verificar e modificar documentos digitais, sem

a necessidade de se modificar a assinatura, assim, sendo possível de se ter dois documentos diferentes dos quais resultem na mesma assinatura digital.

6.1 TRABALHOS FUTUROS

- Fazer implementação de um protótipo para gerar assinaturas digitais que utilize o algoritmo de resumo criptográfico camaleão e o esquema de assinatura proposto para possibilitar ao usuário assinador encontrar colisões para o documento assinado, podendo assim, caso deseje, modificar a assinatura de um documento, sem que se invalide o documento e sem a necessidade de se re-assinar o documento;
- Realizar um levantamento de outros esquemas criptográficos que possam ser utilizados para se construir um esquema de *hash* camaleão;
- Construir um esquema de assinatura digital baseado inteiramente no algoritmo de *hash* camaleão, onde a chave privada do esquema e a chave privada do assinante são as mesmas;
- Prova da segurança do esquema proposto, verificando se todas as propriedades de uma assinatura digital se mantenha.

REFERÊNCIAS

ATENIESE, G.; MEDEIROS, B. D. Identity-based chameleon hash and applications. In: *Financial Cryptography*. [S.l.: s.n.], 2004. v. 4, p. 164–180.

ATENIESE, G.; MEDEIROS, B. D. On the key exposure problem in chameleon hashes. In: *SCN*. [S.l.: s.n.], 2004. v. 4, p. 165–179.

BARRETO, P. et al. Efficient algorithms for pairing-based cryptosystems. In: *Crypto*. [S.l.: s.n.], 2002. v. 2, p. 354–368.

BARRETO, P. S.; NAEHRIG, M. Pairing-friendly elliptic curves of prime order. In: *International Workshop on Selected Areas in Cryptography*. [S.l.: s.n.], 2005. p. 319–331.

BLAKE, I.; SEROUSSI, G.; SMART, N. *Elliptic curves in cryptography*. [S.l.]: Cambridge University Press, 1999.

BONEH, D.; BOYEN, X. Short signatures without random oracles. In: *Eurocrypt*. [S.l.: s.n.], 2004. v. 3027, p. 56–73.

BONEH, D.; LYNN, B.; SHACHAM, H. Short signatures from the weil pairing. *Advances in Cryptology—ASIACRYPT 2001*, p. 514–532, 2001.

DANG, Q. H. *SP 800-107. Recommendation for Applications Using Approved Hash Algorithms*. Gaithersburg, MD, United States, 2009.

DANG, Q. H. *Secure Hash Standard*. [S.l.], jul 2015. <https://doi.org/10.6028/nist.fips.180-4>.

DIFFIE, W.; HELLMAN, M. New directions in cryptography. *IEEE Transactions on Information Theory*, v. 22, n. 6, p. 644–654, Nov 1976.

European Mathematical Society. *Encyclopedia of Mathematics*. 1994. <https://www.encyclopediaofmath.org>.

GAO, W.; LI, F.; WANG, X. Chameleon hash without key exposure based on schnorr signature. *Computer Standards & Interfaces*, v. 31, n. 2, p. 282–285, 2009.

GENNARO, R. Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In: *Crypto*. [S.l.: s.n.], 2004. v. 3152, p. 220–236.

HEUBERGER, C.; MAZZOLI, M. Elliptic curves with isomorphic groups of points over finite field extensions. *Journal of Number Theory*, v. 181, p. 89–98, 2017.

JOUX, A.; NGUYEN, K. *Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups*. 2001. Cryptology ePrint Archive, Report 2001/003. <http://eprint.iacr.org/2001/003>.

KRAWCZYK, H.; RABIN, T. *Chameleon Hashing and Signatures*. 1998. Cryptology ePrint Archive, Report 1998/010. <http://eprint.iacr.org/1998/010>.

KRAWCZYK, H.; RABIN, T. Chameleon signatures. In: *The Network and Distributed System Security Symposium (NDSS)*. San Diego, California: Internet Society, 2000.

PAGE, D.; SMART, N. P.; VERCAUTEREN, F. A comparison of mnt curves and supersingular curves. *Applicable Algebra in Engineering, Communication and Computing*, v. 17, n. 5, p. 379–392, 2006.

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. ISSN 0001-0782. <http://doi.acm.org/10.1145/359340.359342>.

SAKAI, R.; KASAHARA, M. *ID based Cryptosystems with Pairing on Elliptic Curve*. 2003. Cryptology ePrint Archive, Report 2003/054.

STEVENS, M. et al. *The first collision for full SHA-1*. 2017. Cryptology ePrint Archive, Report 2017/190. <http://eprint.iacr.org/2017/190>.

VON ZUR GATHEN, J. *CryptoSchool*. [S.l.: s.n.], 2015.

ZHANG, F.; SAFAVI-NAINI, R.; SUSILO, W. Id-based chameleon hashes from bilinear pairings. *IACR Cryptology ePrint Archive*, v. 2003, p. 208, 2003.

ANEXO A – Código da implementação do esquema de resumo criptográfico camaleão baseado em RSA

Listing A.1 – "Implementação das funções do esquema baseado em RSA utilizando SageMath"

```
import hashlib

def xgcd(a,b):
    prevx, x = 1, 0; prevy, y = 0, 1
    while b:
        q = a//b
        x, prevx = prevx - q*x, x
        y, prevy = prevy - q*y, y
        a, b = b, a % b
    return a, prevx, prevy

# RSA params
e = 65537
n = 58320926802483396389125212252107500436411065096243589331109453840952980382099477479763371362482218203849372786771879820882797942704393146267906895112144911
d = 18887092031309147580188191477161444516263918787901093122411264603512306874431952839123875232453069410346972957692409167245161823370205126607956820408534411

r = 123456789

m = "msg1"
m2 = "msg2"
s = "id"

l = "label"

# j = hash(L)
j = int(hashlib.sha256(l).hexdigest(), 16)
# hm = hash(m)
hm = int(hashlib.sha256(m).hexdigest(), 16)
# jhm = j^hash(m)
jhm = pow(j, hm, n)
# r^e
re = pow(r, e, n)
# chash1 = j^hash(m) * r^e mod n
chash1 = (jhm * re) % n

# Collision Finding
# b = j^d
b = pow(j, d, n)
# hm2 = hash(m2)
hm2 = int(hashlib.sha256(m2).hexdigest(), 16)
# hmhm = hash(m) - hash(m2)
hmhm = hm - hm2
# r2 = r*b^(hash(m) - hash(m2)) mod n
r2 = (r * pow(b, hmhm, n)) % n

# jhm2 = j^hash(m2)
jhm2 = pow(j, hm2, n)
# r2e = r2^e
r2e = pow(r2, e, n)
# chash2 = j^hash(m2) * r2^e mod n
chash2 = (jhm2 * r2e) % n

chash1 == chash2

# Problem with expouse a collision
# Find alfa and beta such that 1 = (hm - hm2)*alfa + v*beta
delta = hmhm
v = e
alfa = xgcd(delta, v)[1]
beta = xgcd(delta, v)[2]

# Find B such that B = (r2/r)^alfa * J^beta
bfind = pow(r2/r, alfa, n) * pow(j, beta, n)

# Find a new colision without d (TRAPDOOR)
m3 = "mensagem3"
# hm3 = hash(m3)
hm3 = int(hashlib.sha256(m3).hexdigest(), 16)
r3 = r*pow(bfind, hm - hm3, n) % n
r3e = pow(r3, e, n)
jhm3 = pow(j, hm3, n)
chash3 = (jhm3 * r3e) % n

chash1 == chash2 == chash3
```

ANEXO B – Código da implementação do esquema de resumo criptográfico camaleão baseado na *Strong Diffie-Hellman Assumption*

Listing B.1 – "Implementação das funções do esquema baseado na SDH-Assumption utilizando SageMath"

```
p = 103; A = 1; B = 18; E = EllipticCurve(GF(p), [A, B])
P = E(33, 91); n = P.order();
print("q = {}".format(n))
k = GF(n)(p).multiplicative_order();
print("degree k = {}".format(k))
K.<a> = GF(p^k)

EK = E.base_extend(K)
P = EK(P)
Qx = 69*a^5 + 96*a^4 + 22*a^3 + 86*a^2 + 6*a + 35
Qy = 34*a^5 + 24*a^4 + 16*a^3 + 41*a^2 + 4*a + 40
Q = EK(Qx, Qy);
# multiply by cofactor so Q has order n:
h = 551269674; Q = h*Q
P = EK(P)


# Chameleon scheme parameters
print("Let g = {} a generator of order {}".format(Q, Q.order()))
# e = Integer(randrange(1, n))
e = 77
print("e = {}".format(e))


def egcd(a, b):
    '''
    Extended Euclidean Algorithm
    '''
    if a == 0:
        return (b, 0, 1)
    else:
        ge, y, x = egcd(b % a, a)
        return (ge, x - (b // a) * y, y)

def inv(a):
    '''
    Inverse mod order of group
    '''
    return egcd(a, n)[1]%n


def chash(m, r, pk, g, e):
    '''
    CHash function
    '''
    return m*g + r*(e*g + pk)


def chash_gr(m, gr, pk, g, e):
    return m*g + (gr + e*g) + (gr + pk)

def collision(m, r, m_, x, g, e):
    '''
    Collision finding function
    '''
    factor = (m-m_)*inv(x+e)
    return r*g + factor*g

def sk_chash(m, gr, x, g, e):
    '''
    Function for compute the chameleon hash with the sk and g^r
    '''
    return m*g + (x+e)*gr

def triple_dh(g, ga, gb, gc):
    '''
    Compute if (ga, gb, gc) is a Diffie-Hellman triple
    '''
    return ga.tate_pairing(gb, n, k) == g.tate_pairing(gc, n, k)

def verify_chash(m, r, pk, g, e, ch):
    '''
    Verify triple diffie-hellman
    '''
    ga = r*g
    gb = pk + e*g
    gc = r*(e*g + pk)

```

```
79
80      return triple_dh(g, ga, gb, gc)
81
82  # Public and private keys
83  x = 14
84  h = x*Q
85  print("x = {}\nh = {}".format(x, h))
86
87  # Message 1
88  m = 136
89  r = 32
90
91  # Message 2
92  m_ = 21
93
94  chm = chash(m, r, h, Q, 77)
95  print("CHash({}, {}, {}) = {}".format(m, r, h, chm))
96  gr_ = collision(m, r, m_, x, Q, 77)
97  chm
98  sk_chash(m_, gr_, x, Q, 77)
99  verify_chash(m, r, h, Q, 77, chm)
```

ANEXO C – Código da implementação do esquema de resumo criptográfico camaleão baseado em assinaturas de Schnorr

Listing C.1 – "Implementação das funções do esquema baseado em RSA utilizando SageMath"

```
import hashlib

def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m

q = 53707525850502909992456713255285669551205363375260169571216185243899217761864845446509849235319087411055813335160735000669367623083162806660733938352077168818816998273240205360944391404386800526948095047303506147348458957525332598509385971437927069169627241560124206746396361100124285189178208876073
print("q = {}".format(q))

m = 123456789
print("m = {}".format(m))

g = 1009
print("g = {}".format(g))
# SK
x = 7589
print("x = {}".format(x))
# PK
y = pow(g, x, q)
print("y = pow(g, x, q) = {}".format(y))
# Label l
l = "douglas"
print("l = {}".format(l))

t1 = 47294729
print("t1 = {}".format(t1))
r2 = 5651
print("r2 = {}".format(r2))
r1 = pow(g, t1, q)
print("r1 = pow(g, t1, q) = {}".format(r1))

c1 = int(hashlib.sha256(l+str(r1)).hexdigest(), 16) % q
print("c1 = {}".format(c1))

S1 = (r1 * pow(y, c1, q)) % q
print("S1 = {}".format(S1))

# CHASH
S2 = (pow(g, m, q) * pow(S1, r2, q)) % q
print("\n \n CHash(PK, m, r2) = S2 = {} \n \n".format(S2))


# CollisionFinding
m2 = 987654321
print("m2 = {}".format(m2))
s1 = (t1+x*int(hashlib.sha256(l+str(r1)).hexdigest(), 16)) % q
print("s1 = {}".format(s1))
s1_ = modinv(s1, q)
print("inv s1 = {}".format(s1_))
r2_ = (s1_*(m-m2) + r2 )% q
print("r2_ = {}".format(r2_))

# CHASH
S2_ = (pow(g, m2, q) * pow(S1, r2_, q)) % qprint("\n \n CHash(PK, m2, r2') = {} \n \n".format(
      S2_))
```

ANEXO D – Artigo

Algoritmos de Hash Camaleão

Douglas Marcelino Beppler Martins[1]

[1]Universidade Federal de Santa Catarina

Resumo. *Resumos criptográficos têm sido usados para se verificar a integridade de mensagens e documentos eletrônicos. Os resumos criptográficos são números relativamente pequenos que representam de forma única os documentos eletrônicos. Assim, diferentes documentos eletrônicos possuem diferentes resumos criptográficos. A assinatura digital de um documento eletrônico, por exemplo, consiste do ciframento, utilizando a chave privada do signatário, do resumo criptográfico do documento. Se qualquer parte do documento assinado for modificada, o resumo criptográfico será diferente, invalidando a assinatura de todo o documento. Acontece que, em algumas situações práticas, é desejável poder-se alterar parte do documento sem invalidar a assinatura digital anteriormente produzida. É o caso do prontuário médico eletrônico assinado digitalmente por um médico. Esses, por razões de privacidade, precisam esconder ou modificar o nome do paciente, quando o prontuário precisar ser disponibilizado de forma pública. Entretanto, a modificação do prontuário, invalida a assinatura digital do médico. Para solucionar este problema, foi proposto na literatura o uso de um novo tipo de resumo criptográfico, denominado de hash camaleão. A diferença para os tradicionais hashes é que diferentes documentos podem ter o mesmo resumo criptográfico. Assim, uma mesma assinatura digital pode ser utilizada para se garantir a integridade e autenticidade de diferentes documentos. Por exemplo, duas versões do mesmo prontuário médico, um contendo o nome do paciente e o outro substituindo o nome do paciente por um nome genérico, poderiam ter a mesma assinatura digital. Neste trabalho são descritos e comparados os principais algoritmos de hash cameleão propostos na literatura. Além disso, é apresentado um software de assinatura digital utilizando hashes camaleão.*

1. Introdução

Funções de resumo criptográficos (do inglês, *hash function*), são funções amplamente utilizadas em diversas áreas da tecnologia da informação, cujo principal objetivo é mapear uma palavra de qualquer tamanho, que pode ser vista como uma sequência de bits, para uma palavra de tamanho fixo e relativamente pequena, ou seja, um número da ordem de grandeza entre 256 e 1024 bits.

Esse número pequeno, denominado de *hash*, pode ser usado para representar qualquer arquivo digital, como uma foto, um vídeo ou um documento PDF. Cada arquivo digital tem o seu próprio número *hash* que o representa. Além disso, por definição, o *hash* não fornece qualquer informação sobre a sequência de bits que o representa.

Para que possamos utilizar funções de resumo de forma segura em um contexto de assinatura digital de documentos eletrônicos, algumas propriedades necessitam ser respeitadas. A primeira delas, chamada de resistência à pré-imagem, consiste em não ser

possível, a partir do *hash*, obter o documento eletrônico utilizado para se determinar esse *hash*.

Um exemplo da importância dessa propriedade é no armazenamento das senhas dos usuários em um provedor de serviços. Para evitar o armazenamento das senhas em claro, o servidor não armazena as senhas dos usuários, mas sim os *hashes* das senhas. Para se autenticar, o usuário envia sua senha para o servidor. Este determina o *hash* da senha e compara com o *hash* da senha previamente armazenado no tabela de senhas do servidor. Se for o mesmo *hash*, trata-se portanto da mesma senha e o servidor considera o usuário autenticado. O armazenamento do *hash* das senhas evita que seja possível obter a senha dos usuários a partir da tabela contendo os *hashes* das senhas.

Uma segunda propriedade importante que algoritmos de resumo precisam garantir é a resistência à colisão. Essa garante que não é possível encontrar dois documentos eletrônicos distintos que tenham o mesmo *hash*. A resistência à colisão é utilizada para assegurar a integridade das assinaturas digitais de documentos eletrônicos.

O modelo clássico de gerar a assinatura digital de um documento eletrônico consiste em cifrar o *hash* do documento com a chave privada do signatário. Para se verificar a assinatura, o verificador decifra o *hash* com a chave pública do signatário e compara com o *hash* do documento. Caso eles sejam iguais, é possível afirmar quem é o signatário do documento e também que o documento está íntegro.

A terceira e última propriedade é a resistência à segunda pré-imagem. Dado o *hash* de um documento eletrônico, não deve ser possível encontrar outro documento diferente que tenha o mesmo *hash*.

Contudo, há situações na prática, principalmente relacionadas a requisitos de privacidade, onde é desejável manter a assinatura digital do documento realizada pelo signatário, mesmo que partes do mesmo sejam modificadas. Por exemplo, um prontuário médico de um paciente assinado por um médico, poderia ter o nome do paciente modificado, mantendo a integridade e autenticidade proporcionadas pela assinatura digital. Assim, o prontuário poderia ser disponibilizado, sem revelar o nome do paciente.

Em esquemas de assinaturas que utilizam funções de resumos criptográficos tradicionais, não é possível se realizar alterações nos documentos assinados. Porém, tem sido proposto na literatura algoritmos de *hash* onde é permitido, de forma controlada, a colisão de *hashes*. Esses novos algoritmos de *hashes* são conhecidos como *hash* camaleão (em ingles *chameleon hash*).

Hashes camaleão foram propostos por Hugo Krawczyk e Tal Rabin em 1998 [Krawczyk and Rabin 1998, Krawczyk and Rabin 2000]. O algoritmo proposto por eles faz uso de um par de chaves criptográficas, ou seja, uma chave pública e a respectiva chave privada, com as seguintes características:

- Qualquer pessoa com o conhecimento da chave pública associada ao algoritmo pode calcular o *hash*;
- Para os que não conhecem a chave privada associada ao algoritmo é computacionalmente muito difícil encontrar duas mensagens diferentes das quais tenham um mesmo valor de *hash*;
- Para o detentor da chave privada é possível facilmente determinar uma segunda mensagem, diferente da primeira, que tenha o mesmo valor de *hash*.

Essas características garantem a um esquema de assinatura digital que utilize o algoritmo de *hash* camaleão como algoritmo de resumo, a possibilidade de se modificar um documento assinado de forma segura.

O presente trabalho visa realizar um estudo sobre algoritmos de *hash* camaleão, suas principais funções e propriedades. Os principais algoritmos de resumo camaleão propostos na literatura são analisados e comparados. Dois dos algoritmos foram implementados.

2. Objetivos

2.1. Objetivos gerais

Este trabalho tem como objetivo principal realizar uma pesquisa, buscando os diversos tipos de construção de esquemas de resumo criptográfico camaleão, analisando a geração de chaves, a forma de como é calculado o valor de resumo e como é encontrada uma colisão. Adicionalmente, são analisadas suas principais características e propriedades visando seu uso em esquemas de assinatura digital.

2.2. Objetivos específicos

- Apresentar o *hash* camaleão, suas funções e suas características visando seu uso em assinaturas digitais;
- Apresentar os principais esquemas do algoritmo de resumo camaleão encontrados na literatura;
- Escolher e implementar alguns desses *hashes* camaleão;
- Comparar os esquemas apresentados, relacionando as propriedades analisadas de cada esquema;
- Elaborar um esquema de assinatura digital onde é possível utilizar o algoritmo de resumo criptográfico camaleão como algoritmo de *hash*.

3. Esquemas de Hash Camaleão

4. *Hash* camaleão baseado em RSA

O primeiro esquema a ser analisado foi proposto por Giuseppe Ateniese e Breno de Medeiros em [Ateniese and De Medeiros 2004b]. Esse é baseado nas propriedades de chaves assimétricas do algoritmo RSA.

4.1. Geração de chaves

Nesse esquema de *hash* camaleão utilizamos um par de chaves RSA sem qualquer modificação em relação ao algoritmo RSA padrão. Seja esse par $(\mathcal{PK}, \mathcal{SK})$ onde a chave pública é a tupla $\mathcal{PK} = (e, n)$ e a chave privada é a tupla $\mathcal{SK} = (d, p, q)$.

4.2. Computando o *hash*

Seja a função de resumo criptográfico $H : \{0,1\}^* \longrightarrow \{0,1\}^t$, com entrada de qualquer quantidade de bits e saída de comprimento t. Existem muitas funções de *hash* propostas na literatura e em uso. Um exemplo é a função SHA-256 [Dang 2015]. Dado um par de chaves RSA $(\mathcal{PK}, \mathcal{SK})$, e S uma palavra identificadora, em sua essência S pode ser qualquer palavra, porém é utilizada como identificação da instância do *hash* camaleão.

Após escolher a palavra S, pode-se então calcular o valor de $J = H(S)$. Então para se computar o valor de *hash* camaleão de uma mensagem m, basta escolher um valor aleatório r e em seguida calcular a seguinte expressão.

$$CHash(\mathcal{PK}, m, r) = J^{H(m)} r^e \pmod{n} \tag{1}$$

Assim se igualando a resumos criptográficos tradicionais em que qualquer pessoa possa verificar um valor de *hash* h para uma mensagem m e um valor aleatório r, basta computar o valor do resumo para (m, r) e verificar este é igual ao valor de h.

4.3. Encontrando colisões

Para encontrar uma colisão para um valor de *hash* previamente calculado basta computar um novo valor r' através da seguinte equação

$$\begin{aligned} CollisionFinding(SK, m, r, m') &= r J^{d(H(m)-H(m'))} \\ &= r' \pmod{n}. \end{aligned} \tag{2}$$

É possível se verificar que o par (m', r') é uma colisão para (m, r) realizando uma manipulação algébrica na formula de computar o algoritmo de *hash* camaleão, podendo se verificar a seguir:

$$\begin{aligned} CHash(\mathcal{PK}, m', r') &= J^{H(m')} r'^e \pmod{n} \\ &= J^{H(m')} \left(r J^{(H(m)-H(m'))d}\right)^e \\ &= J^{H(m')} J^{(H(m)-H(m'))} r^e \\ &= J^{H(m)} r^e \\ &= CHash(\mathcal{PK}, m, r). \end{aligned} \tag{3}$$

4.4. Análise das propriedades

4.4.1. Propriedades da função $CHash$

Devido ao esquema de *hash* camaleão baseado em RSA utilizar apenas parâmetros públicos para computar o valor de resumo na função $CHash$, então é possível utilizar o algoritmo de *hash* camaleão baseado em RSA para verificar a integridade de documentos assinados, assim tendo **compatibilidade** com os esquemas de assinatura padronizados atualmente.

Para se encontrar uma colisão sem ter conhecimento da chave privada $\mathcal{SK}$ atrelada ao esquema, é necessário quebrar o equivalente a uma assinatura RSA em cima de J, assim, computar colisões para o esquema RSA sem o conhecimento da chave privada é computacionalmente impossível, tornando o esquema **resistente a colisão**.

Para cada mensagem m, o valor $h = CHash(\mathcal{PK}, m, r)$ é unicamente determinado por r, podemos dizer também que o valor de r é determinado por m também. Por

serem variáveis aleatórias a entropia destas em relação ao valor de resumo h são iguais, assim respeitando a **segurança semântica**, além do uso de uma função de resumo criptográfico H também garantir a propriedade de **uniformidade**.

O detentor da chave privada $\mathcal{SK}$ pode computar uma colisão para qualquer par (m, r), sendo assim, pode calcular uma colisão para um suposto par (m', r'), para garantir a **ocultação da mensagem**.

4.4.2. Propriedades da função $CollisionFinding$

Dado duas assinaturas (m, r) e (m', r'), é possível computar α e β através do algoritmo de Euclides Estendido, tal que $\alpha(H(m) - H(m')) + \beta e = 1$. Então é possível utilizar α e β para encontrar J^d e calcular outras futuras colisões. Note que só é possível pois:

$$
\begin{aligned}
r' &= r J^{d(H(m)-H(m'))} \\
r'/r &= J^{d(H(m)-H(m'))} \\
(\tfrac{r'}{r})^{\alpha} &= (J^{d(H(m)-H(m'))})^{\frac{1-\beta e}{H(m)-H(m')}} \\
J^d &= (\tfrac{r'}{r})^{\alpha} J^{\beta}.
\end{aligned}
$$

Assim, o esquema de *hash* camaleão RSA não respeita a propriedade de **livre exposição de chaves**.

5. Hash camaleão baseado na *SDH-Assumption*

Giuseppe Ateniese e Breno de Medeiros em [Ateniese and De Medeiros 2004b] propuseram um novo esquema de *hash* camaleão. Esse esquema utiliza as propriedades do *gap-DDH* e é considerado seguro se respeitar a *SDH-Assumption*.

5.1. Configuração

Seja $\mathbb{G}$ um *gap-DDH* de ordem prima p, com um gerador g e uma função de emparelhamento bilinear e.

5.2. Geração de chaves

Determine a chave privada $\mathcal{SK}$ (nessa seção utilizaremos a letra x para designar $\mathcal{SK}$), tal que $x \in \mathbb{Z}_p$, e a chave pública $\mathcal{PK}$ (nessa seção utilizaremos a letra h para designar $\mathcal{PK}$), tal que $h = g^x$.

5.3. O esquema de resumo

O esquema baseado na *SDH-Assumption* faz uso de uma função de resumo criptográfica tradicional $H : \{0,1\}^* \longrightarrow \{0,1\}^t$ e de uma palavra identificadora S.

Primeiramente é calculado o *hash* a partir desse identificador, sendo ele $c = H(S)$. Então dado uma mensagem m, basta gerar um número aleatório $r \in \mathbb{Z}_p$. O valor do *hash* camaleão para o par (m, g^r) é dado pela equação:

$$CHash(h = podes\mathcal{PK}, m, r) = g^{H(m)}(g^c h)^r. \tag{4}$$

Para verificar se a mensagem m corresponde ao valor de *hash* para um dado r, é necessário checar se $(g^r, g^c h, F)$ é uma tripla Diffie-Hellman, onde F é o valor do *hash* camaleão dividido por $g^{H(m)}$, ou seja $F = (g^c h)^r$. É possível realizar a implementação de um algoritmo que verifique a tripla DH utilizando uma função de emparelhamento bilinear, ou seja, se a seguinte igualdade entre os emparelhamentos é verdadeira:

$$e(g^r, g^c h) = e((g^c h)^r, g). \tag{5}$$

Pode-se mostrar que, caso o *hash* seja correto, $(g^r, g^c h, F)$ é uma tripla DH, ou seja:

$$\begin{aligned}(g^r, g^c h, F) &= (g^r, g^x g^c, (g^c h)^r) \\ &= (g^r, g^{x+c}, (g^c g^x)^r) \\ &= (g^r, g^{x+c}, g^{(x+c)r}).\end{aligned} \tag{6}$$

Assim, o esquema baseado na *Strong Diffie-Hellman Assumption* permite que qualquer entidade possa calcular um valor de resumo para uma mensagem m, e também permite a verificação do resumo, ou seja, se uma mensagem m gera um valor de *hash* recebido.

5.4. Encontrando colisões

Para encontrar uma colisão m', tal que $m' \neq m$, para um valor de resumo gerado anteriormente pelo par (m, g^r) é necessário utilizar a chave privada $\mathcal{SK}$ para calcular o valor de $g^{r'}$ de modo que o valor de *hash* camaleão para

$$\begin{aligned}CollisionFinding(x = \mathcal{SK}, m, r, m') &= g^r g^{\frac{H(m)-H(m')}{x+c}} \\ &= g^{r'}\end{aligned} \tag{7}$$

Para verificar que $g^{r'}$ e m' geram uma colisão, é necessário verificar se a propriedade da tripla DH ainda é respeitada:

$$\begin{aligned}(g^{r'}, g^c h, F') &= (g^{r'}, g^{x+c}, (g^{x+c})^{r'}) \\ &= (g^{r'}, g^{x+c}, (g^{r'})^{x+c}) \\ &= (g^{r'}, g^{x+c}, (g^r g^{\frac{H(m)-H(m')}{x+c}})^{x+c}) \\ &= (g^r g^{\frac{H(m)-H(m')}{x+c}}, g^{x+c}, g^{r(x+c)} g^{H(m)-H(m')}).\end{aligned}$$

É possível demonstrar algebricamente que um resumo de (m', r') é equivalente a um resumo obtido com (m, r), apesar de ser computacionalmente inviável computar r' pelo

problema do logaritmo discreto. Para provar essa igualdade, primeiramente é necessário modificar a equação $CHash$, deixando ela em fatores de g^r de tal forma que a nova função $PrivCHash$ irá necessitar da chave privada para computar o valor do *hash*

$$\begin{aligned} CHash(\mathcal{PK}, m, r) &= g^{H(m)}(g^c h)^r \\ &= g^{H(m)} g^{rc} g^{rx} \\ &= g^{H(m)} g^{r(x+c)} \\ PrivCHash(\mathcal{SK}, m, g^r) &= g^{H(m)} g^{r(x+c)}. \end{aligned} \tag{8}$$

Assim, é possível calcular o valor de resumo a partir de g^r, m, e x. Então, pode-se provar que $(m', g^{r'})$ é uma colisão para o par (m, r), como visto abaixo

$$\begin{aligned} PrivCHash(\mathcal{SK}, m, g^{r'}) &= g^{H(m')} g^{r'(x+c)} \\ &= g^{H(m')} (g^r g^{\frac{H(m)-H(m')}{x+c}})^{(x+c)} \\ &= g^{H(m')} g^{r(x+c)} g^{H(m)-H(m')} \\ &= g^{H(m)} g^{r(x+c)} \\ &= g^{H(m)} g^{rc} g^{rx} \\ &= g^{H(m)} (g^c h)^r \\ &= CHash(\mathcal{PK}, m, r). \end{aligned} \tag{9}$$

5.5. Análise das propriedades

5.5.1. Propriedades da função $CHash$

Igualmente como no esquema de *hash* camaleão baseado em RSA, o esquema do algoritmo de *hash* camaleão baseado na *Strong Diffie-Hellman Assumption* também só faz uso de atributos públicos para computar um resumo, tornando este **compatível** com esquemas de assinaturas tradicionais. Ademais, este esquema respeita as propriedades de **segurança semântica** e de **uniformidade** pelos mesmos motivos citados na análise das propriedades do esquema RSA, por fazer uso de duas variáveis aleatórias, m e r, e de uma função de *hash* tradicional H, que garante uniformidade em sua saída.

Sua segurança na parte de **resistência a colisões** está atrelada na dificuldade de se computar $g^{\frac{1}{x+c}}$, que no geral é um problema computacionalmente difícil [Boneh and Boyen 2004].

Como o detentor da chave privada $\mathcal{SK}$ pode computar uma colisão para qualquer par (m, g^r), então é garantida a propriedade de **ocultação da mensagem**.

5.5.2. Propriedades da função $CollisionFinding$

A partir da exposição de uma colisão para o esquema do *hash* camaleão baseado na *SDH-assumption*, (m, g^r) e $(m', g^{r'})$, é possível então computar de forma eficiente $g^{\frac{1}{x+c}}$,

equivalente ao problema computacional difícil sob a *Strong Diffie-Hellman Assumption*. Porém, caso seja escolhida uma palavra identificadora c diferente para cada vez da qual for retirar um resumo, obtemos um esquema onde a propriedade de **livre exposição de chave** é respeitada [Ateniese and De Medeiros 2004b], pois ele se torna equivalente ao esquema demonstrado em [Boneh and Boyen 2004].

6. *Hash* camaleão baseado em emparelhamento bilinear

Baseado apenas em funções de mapeamento bilineares sob grupos, [Zhang et al. 2003] propõe duas formas de implementar as funções do algoritmo de *hash* camaleão. Ambas as formas são baseadas em identificadores para o detentor da chave, como em outros esquemas de *hash* camaleão, que criptograficamente podem ser tratados apenas como um número. Porém, estes identificadores agregam um certo nível de segurança semântica ao esquema [Ateniese and De Medeiros 2004a].

6.1. Esquemas 1 e 2: Configuração

Seja $\mathbb{G}_1$ um grupo cíclico aditivo com um gerador P, tal que P tenha ordem prima q e seja $\mathbb{G}_2$ um grupo cíclico multiplicativo com a mesma ordem. Tome e uma função de emparelhamento bilinear de $\mathbb{G}_1$ para $\mathbb{G}_2$ tal que $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, $e(aP, bQ) = e(P, Q)^{ab}$ $\forall\ P, Q \in \mathbb{G}_1$ e $a, b \in \mathbb{Z}$. O esquema faz uso de uma função de resumo criptográfico tradicional H, como nos esquemas anteriormente descritos e faz uso de uma função de resumo criptográfico H' tal que $H' : \{0,1\}^* \rightarrow \mathbb{G}_1^*$.

6.2. Esquema 1: Geração de chaves

Primeiramente, é necessário definir um ID (sugere-se algo que represente a entidade, como um e-mail ou pseudônimo). Em seguida, é preciso definir s tal que $s \in \mathbb{Z}_q^*$; este é considerado a chave privada mestre do esquema. Assim, é possível calcular a chave pública através de s, tal que $P_{pub} = sP$. Por fim, a chave privada $\mathcal{SK}$ é computada, outrora chamada de $S_{id} = sH'(ID)$. Então, o par de chaves consiste de $\mathcal{PK} = P_{pub}$ e $\mathcal{SK} = S_{id}$; é importante salientar que a chave privada do esquema está atrelada diretamente com o valor do ID supracitado.

6.3. Esquema 1: O esquema de resumo

Com posse da chave pública gerada anteriormente e da palavra ID, basta escolher aleatoriamente $R \in \mathbb{G}_1$ e então computar a seguinte expressão:

$$CHash(\mathcal{PK}, ID, m, R) = e(R, P)e(H(m)H'(ID), P_{pub}). \tag{10}$$

6.4. Esquema 1: Encontrando colisões

Para encontrar uma colisão m' para uma mensagem m, e um valor aleatório $R \in \mathbb{G}_1$, do qual foi computado um valor de resumo de *hash* camaleão anteriormente, é necessário encontrar um valor de R' através da seguinte fórmula:

$$\begin{aligned} CollisionFinding(\mathcal{PK}, ID, \mathcal{PK}, m, R, m') &= R' \\ &= (H(m) - H(m'))S_{id} + R. \end{aligned} \tag{11}$$

Para verificar se o par (m', R') é uma colisão para (m, R) é necessário observar se um valor de *hash* obtido através do primeiro par é equivalente a um resumo com o segundo par, o que pode ser verificado a seguir:

$$\begin{aligned} &CHash(\mathcal{PK}, ID, m', R') = e(R', P)e(H(m')H'(ID), P_{pub}) \\ &= e((H(m) - H(m'))S_{id} + R, P)e(H(m')H'(ID), P_{pub}) \\ &= e((H(m) - H(m'))S_{id}, P)e(R, P)e(H(m')H'(ID), P_{pub}) \\ &= e((H(m) - H(m'))sH'(ID), P)e(R, P)e(H(m')H'(ID), P_{pub}) \\ &= e((H(m) - H(m'))H'(ID), P_{pub})e(R, P)e(H(m')H'(ID), P_{pub}) \\ &= e(H(m)H(ID) - H(m')H'(ID), P_{pub})e(R, P)e(H(m')H'(ID), P_{pub}) \\ &= e(R, P)e(H(m)H'(ID), P_{pub}) \\ &= CHash(\mathcal{PK}, ID, m, R). \end{aligned} \tag{12}$$

6.5. Esquema 2: Geração de chaves

Igualmente como no esquema 1, a chave pública $\mathcal{PK}$ é calculada utilizando uma chave mestre s e o gerador de $\mathbb{G}_1$, tal que $\mathcal{PK} = P_{pub} = sP$. Por sua vez, a obtenção da chave privada é modificada, mas esta ainda depende diretamente do ID definido para a entidade. Para obter a chave privada no segundo esquema proposto, basta computar $\mathcal{SK} = S_{id} = \frac{1}{s+H(ID)}P$.

6.6. Esquema 2: O esquema de resumo

Para computar um valor de resumo para uma mensagem m, o usuário deve calcular aleatoriamente um valor de $R \in \mathbb{G}_1$ como no esquema 1 e calcular o valor de resumo da seguinte maneira:

$$CHash(\mathcal{PK}, ID, m, R) = e(P, P)^{H(m)}e(H(ID)P + P_{pub}, R)^{H(m)}. \tag{13}$$

6.7. Esquema 2: Encontrando colisões

Para calcular uma colisão para um par (m, R), é necessário que o detentor da chave privada compute um novo valor de R' para o valor de m' desejado da seguinte maneira:

$$\begin{aligned} &CollisionFinding(\mathcal{SK}, ID, m, R, m') = R \\ &= H(m')^{-1}((H(m) - H(m'))S_{id} + H(m)R). \end{aligned} \tag{14}$$

Para verificar se o valor de R' encontrado para m' satisfaz uma colisão para o par (m, R) é necessário mostrar que ambas as tuplas produzam o mesmo valor de resumo, como demonstrado a seguir:

$$
\begin{aligned}
&CHash(\mathcal{PK}, ID, m', R') = e(R', P)e(H(m')H'(ID), P_{pub}) \\
&= e((H(m) - H(m'))S_{id} + R, P)e(H(m')H'(ID), P_{pub}) \\
&= e((H(m) - H(m'))S_{id}, P)e(R, P)e(H(m')H'(ID), P_{pub}) \\
&= e((H(m) - H(m'))sH'(ID), P)e(R, P)e(H(m')H'(ID), P_{pub}) \\
&= e((H(m) - H(m'))H'(ID), P_{pub})e(R, P)e(H(m')H'(ID), P_{pub}) \\
&= e(H(m)H(ID) - H(m')H'(ID), P_{pub})e(R, P)e(H(m')H'(ID), P_{pub}) \\
&= e(R, P)e(H(m)H'(ID), P_{pub}) \\
&= CHash(\mathcal{PK}, ID, m, R).
\end{aligned}
\tag{15}
$$

6.8. Análise das propriedades

6.8.1. Propriedades das funções $CHash$

Para o cálculo do resumo criptográfico nos esquemas baseados em emparelhamento bilinear, são utilizados duas funções de *pairings* sobre parâmetros públicos do esquema e da mensagem, assim, tornando-o **compatível** com assinaturas digitais.

Para o primeiro esquema apresentado, o processo de extração de chave é descrito em [Boneh et al. 2001], o qual é seguro contra ataques que tentem forjar a assinatura, assumindo que o problema computacional de Diffie-Hellman (CDH) é difícil de ser resolvido.

O segundo faz uso do esquema de assinatura proposto em [SAKAI and KASAHARA 2003], o qual é considerado seguro, assumindo a dificuldade de resolver o *k-weak* problema computacional Diffie-Hellman (k-wCDHP) é difícil de ser resolvido [Zhang et al. 2003], o que torna ambos os esquemas **resistentes a colisões**.

Os esquemas 1 e 2, são ambos semanticamente seguros, pois o ID do usuário, e a mensagem não são relacionadas a variável aleatória $R \in \mathbb{G}_1$, e a não degeneração do emparelhamento bilinear faz com que a aleatoriedade de R torne os esquemas **semanticamente seguros** [Zhang et al. 2003], além da **uniformidade** ser garantida pela função de *hash* H.

6.8.2. Propriedades da funções $CollisionFinding$

Pelas propriedades dos emparelhamentos bilineares, é possível realizar uma manipulação algébrica na igualdade de um valor de resumo e de uma colisão para descobrir a chave privada $\mathcal{SK}$. No primeiro esquema, manipulamos a seguinte igualdade:

$$
\begin{aligned}
&CHash(\mathcal{PK}, ID, m', R') = CHash(\mathcal{PK}, ID, m, R) \\
&e(R, P)e(H(m)H'(ID), P_{pub}) = e(R', P)e(H(m')H'(ID), P_{pub}) \\
&e(R - R', P) = e((H(m) - H(m'))H'(ID), P_{pub}) \\
&e(R - R', P) = e((H(m) - H(m'))S_{id}, P).
\end{aligned}
\tag{16}
$$

Então a chave privada $\mathcal{SK} = S_{id}$ é obtida por

$$S_{id} = (H(m') - H(m))^{-1}(R - R'). \tag{17}$$

No segundo esquema baseado em emparelhamento bilinear também é possível descobrir a chave privada, manipulando a igualdade

$$\begin{aligned} &CHash(P_{pub}, ID.m, R) = CHash(P_{pub}, ID.m', R') \\ &= e(P,P)^{H(m)} e(H(ID)P + P_{pub}, R)^{H(m)} \\ &= e(P,P)^{H(m')} e(H(ID)P + P_{pub}, R')^{H(m')} \\ &= e(P,P)^{H(m)-H(m')} e(H(ID)P + P_{pub}, H(m')R' - H(m)R). \end{aligned} \tag{18}$$

Então a chave privada $\mathcal{SK} = S_{id}$ é obtida por

$$S_{id} = (H(m') - H(m))^{-1}(H(m')R' - H(m)R). \tag{19}$$

Como provado na proposta em [Zhang et al. 2003], o esquema de resumo criptográfico camaleão baseado em emparelhamento bilinear não é **livre de exposição de chave**.

7. Esquema baseados em assinaturas de Schnorr

O esquema proposto em [Gao et al. 2009], é baseado no esquema de assinatura de Schnorr, no qual sua segurança é baseada na intratabilidade do problema do logaritmo discreto.

7.1. Configuração

Seja $\mathbb{G}$ um grupo cíclico multiplicativo com um gerador g, tal que g tenha ordem prima q, além disso, o esquema faz uso de uma função de resumo criptográfico tradicional H.

7.2. Geração de chaves

Similar ao esquema baseado na *SDH-Assumption*, para computar a chave privada $\mathcal{SK}$ basta escolher aleatoriamente $x \in \mathbb{Z}_q$, então a chave pública pode ser obtida através do gerador g e da chave privada x, tal que $\mathcal{PK} = y = g^x$.

7.3. O esquema de resumo

Primeiramente, para calcular o resumo camaleão de uma mensagem é necessário definir uma etiqueta L, além de definir dois números aleatórios t_1, $r_2 \in \mathbb{Z}_q$. Inicialmente então calcula-se r_1, c_1 e S_1 da seguinte maneira:

$$\begin{aligned} r_1 &= g^{t_1} \\ c_1 &= H(L; r_1) \\ S_1 &= r_1 y^{c_1}. \end{aligned} \tag{20}$$

Por fim, a equação para calcular o resumo é dada por:

$$CHash(\mathcal{PK}, L, m, r_2) = S_2 = g^m S_1^{r_2}. \tag{21}$$

7.4. Encontrado Colisões

Caso o detentor da chave privada $\mathcal{SK}$ deseje encontrar uma colisão m' para um par (m, r_2) do qual se tenha computado um valor de resumo anteriormente, primeiro é necessário encontrar o valor de s_1, que é a *ephemeral trapdoor*

$$s_1 = t_1 + xH(L; r_1) \pmod{q}. \tag{22}$$

Dado o valor de s_1 então é possível calcular o valor de r_2' para que m' seja uma colisão através da seguinte equação:

$$\begin{aligned} CollisionFinding(\mathcal{SK}, s_1, m, r_2, m') &= r_2' \\ &= s_1^{-1}(m - m') + r_2 \pmod{q}. \end{aligned} \tag{23}$$

Para verificar se (m', r_2') colide com (m, r_2), substitui-se os termos das equações até que $CHash(\mathcal{PK}, L, m', r_2') = CHash(\mathcal{PK}, L, m, r_2)$

$$\begin{aligned} CHash(\mathcal{PK}, L, m', r_2') &= S_2' \\ &= g^{m'} S_1^{r_2'} \\ &= g^{m'} S_1^{s_1^{-1}(m-m')+r_2} \\ &= g^{m'} S_1^{(t_1+xH(L;r_1))^{-1}(m-m')+r_2} \\ &= g^{m'} \left(g^{t_1+xH(L;r_1)}\right)^{(t_1+xH(L;r_1))^{-1}(m-m')} S_1^{r_2} \\ &= g^{m'} g^{m-m'} S_1^{r_2} \\ &= g^{m} S_1^{r_2} \\ &= CHash(\mathcal{PK}, L, m, r_2). \end{aligned} \tag{24}$$

7.5. Análise das propriedades

7.5.1. Propriedades da função $CHash$

Igualmente como em outros esquemas apresentados nesta seção, o algoritmo de *hash* camaleão baseado em assinaturas de Schnorr respeita a propriedade de **compatibilidade**, pois qualquer um com conhecimento dos parâmetros públicos do esquema tem capacidade de computar um valor de resumo para uma mensagem m.

Para o conhecedor da chave privada $\mathcal{SK}$ é possível, através da fórmula para se encontrar a *ephemeral trapdoor* e da $CollisionFinding$, encontrar uma colisão, provendo a **ocultação da mensagem**; porém, para um atacante que não tenha conhecimento da chave privada, como o par (r_1, s_1) é equivalente a uma assinatura de Schnorr sobre a etiqueta L, encontrar uma colisão é equivalente a quebrar esta assinatura [Gao et al. 2009], tornando o esquema **resistente a colisões**.

Ao computar um valor de resumo, além do valor da mensagem m, e da variável de r_2 serem aleatórios, o esquema de *hash* camaleão baseado em assinaturas de Schnorr faz uso de uma chave efêmera, possuindo assim **segurança semântica**, pois o conteúdo das variáveis é sempre independente; ademais, este esquema também faz uso de uma função de resumo criptográfico tradicional, garantindo a característica da **uniformidade**.

7.5.2. Propriedades da função $CollisionFinding$

Como para um atacante sem conhecimento da chave privada revelar colisões é equivalente a quebrar o esquema de assinatura de Schnorr, o esquema é seguro contra **exposição de chaves** e de ataques ativos conhecidos [Gao et al. 2009].

8. Avaliação e Comparação de Hashes Camaleão e seu uso para Assinar Documentos Eletrônicos

9. Análise geral dos esquemas de *hash* camaleão

Foram analisados no capítulo 3 cinco esquemas de resumo criptográfico camaleão; um baseado na criptografia do RSA, um baseado na *Strong Diffie-Hellman Assumption*, dois esquemas baseado em emparelhamento bilinear e por fim um esquema baseado em assinaturas de Schnorr. Para poder padronizar a forma de realizar a análise, primeiro foram definidas as funções que um esquema de *hash* camaleão precisa ter e suas finalidades:

- Geração de chaves: responsável por definir o par de chaves utilizado no esquema;
- Computando o *hash*: onde é definido a função $CHash(\cdot)$, que é a responsável pode calcular o valor de resumo;
- Encontrando colisões: que define a função $CollisionFinding(\cdot)$, que através dela é possível se encontrar uma colisão entre duas mensagem para o esquema.

Para todos os esquemas citados, foi demonstrado que existe uma colisão entre duas mensagens m, m', tais que $m \neq m'$, ou seja, os valores de resumo criados aplicando a função $CHash$ a estas mensagens são iguais.

Para analisar a viabilidade da utilização dos esquemas no âmbito de assinaturas digitais, foi feito um levantamento dos principais artigos e selecionadas as principais características que um esquema camaleão necessita respeitar, e para todas estas propriedades, foi analisado em cada esquema se é respeitada, como pode ser observado pela tabela a seguir:

A propriedade de compatibilidade diz respeito à capacidade do esquema de ser utilizado em esquemas de assinaturas digitais como uma função de resumo criptográfico; para isso, deve ser possível que qualquer pessoa possa calcular um valor de resumo para qualquer mensagem.

A resistência a colisão serve para que os esquemas sejam equivalentes a outros algoritmos de *hash* criptográficos, evitando ataques onde é possível se encontrar duas mensagens com o mesmo valor de resumo, embora esquemas de *hash* camaleão possibilitem colisões, é imprescindível que apenas o detentor da chave privada do esquema consiga obter tal colisão e que isto seja possível somente através da função $CollisionFinding$.

Tabela 1. Comparação das propriedades dos esquemas do algoritmo de hash camaleão

	RSA	SDH	Pairing 1	Pairing 2	Schnorr
Compatibilidade	✓	✓	✓	✓	✓
Resistência a colisão	✓	✓	✓	✓	✓
Segurança semântica	✓	✓	✓	✓	✓
Ocultação da mensagem	✓	✓	✓	✓	✓
Uniformidade	✓	✓	✓	✓	✓
Livre de exposição de chave		✓			✓

As propriedades de segurança semântica, ocultação da mensagem e uniformidade precisam ser respeitadas para que os esquemas tenham o mesmo comportamento dos algoritmos normais, dissociando o resumo do conteúdo da mensagem, tornando o valor final do resumo similar a um número aleatório.

Por fim, a última propriedade analisada é a garantia da liberdade de exposição de chave, ou seja, a chave privada não será revelada no cômputo de uma colisão. O esquema baseado em RSA e os dois esquemas baseados em emparelhamento bilinear não garantem esta propriedade, pois é possível realizar uma manipulação algébrica na estrutura de ambos os esquemas e, consequentemente, encontrar o valor da chave privada. Devido a estas características, restam apenas dois esquemas nos quais são possíveis encontrar colisões e manter a chave privada segura: o esquema baseado na *Strong Diffie-Hellman Assumption* e o esquema baseado nas assinaturas de Schnorr.

10. Utilização em esquemas de assinaturas digitais

Após a análise dos requisitos realizada, é possível utilizar um esquema de resumo criptográfico camaleão em esquemas de assinaturas digitais, desde que este respeite todas as propriedades analisadas anteriormente, tornando possível modificar o documento assinado sem invalidar a assinatura e sem a necessidade de realizar uma nova assinatura.

Para realizar uma assinatura digital do documento utilizando o algoritmo de *hash* camaleão, o assinante primeiro precisa definir o par de chaves $(\mathcal{PK}, \mathcal{SK})$ utilizado no esquema. A chave pública deve estar acessível a todos que necessitem verificar a assinatura: para isso, deve ser criado um campo específico na assinatura para armazená-la, além de outros parâmetros dos esquemas, como palavras identificadoras. Por fim, para computar o resumo, basta apenas definir o valor aleatório r, e calcular o valor da equação $CHash$ para o documento. Dado o valor de resumo criptográfico calculado, em seguida basta realizar a cifra com a chave privada do assinante e a assinatura está concluída. O esquema descrito na Figura 1 descreve o fluxo da assinatura.

Assim, a construção da assinatura mantém-se de forma similar à de esquemas de assinatura tradicionais, apenas com a ressalva de que os parâmetros do algoritmo de resumo criptográfico camaleão precisarão ser disponibilizados assinatura, para que seja possível afirmar a integridade do documento no processo de verificação da assinatura.

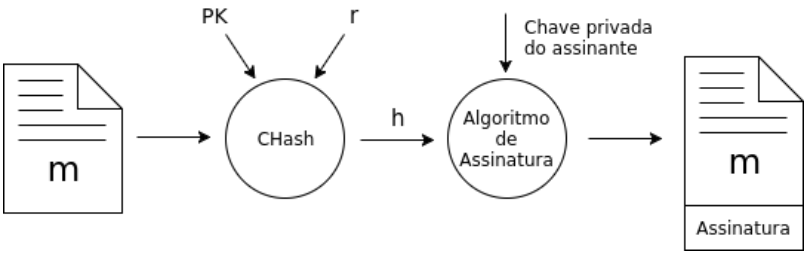

Figura 1. Diagrama do fluxo de assinatura de um documento digital com *hash* camaleão

Caso o detentor da chave privada, associada a chave pública utilizada para se calcular o resumo original da assinatura, deseje alterar o documento, este deve passar para a função $CollisionFinding$ os parâmetros m e r que foram utilizados para gerar o primeiro resumo, além da chave privada $\mathcal{SK}$. A saída r', junto com o novo documento m', são uma colisão para o documento original m. Assim, basta armazenar junto à assinatura o novo valor de r' e a assinatura para o novo documento será válida. A Figura 2 mostra o fluxo de modificação do documento, a fim de montar a nova assinatura, sem a necessidade do uso da chave privada do assinante.

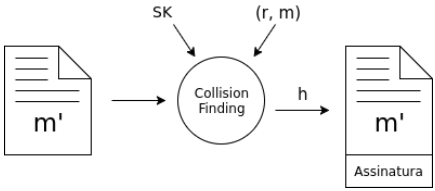

Figura 2. Diagrama do fluxo da utilização da função encontrando colisões do *hash* camaleão

Para realizar a validação da assinatura, é preciso primeiro computar o valor do *hash* camaleão. Para isso é necessário buscar a chave pública $\mathcal{PK}$ e o valor de r na assinatura. Em seguida, é computado o valor do resumo criptográfico camaleão para o arquivo; este deve ser igual ao valor da assinatura após ter sido decifrado pela chave pública do assinante, caso contrário a assinatura é inválida. Este processo indepente se a assinatura foi gerada com ou sem colisões. A Figura 3 mostra o fluxo de validação de uma assinatura que utiliza o algoritmo de resumo camaleão.

11. Considerações Finais

Este trabalho apresentou o algoritmo de resumo criptográfico camaleão, visando aumentar o escopo de assinaturas digitais, do qual um médico ao ter a necessidade de publicar um prontuário eletrônico assinado digitalmente de um paciente, ao modificar o prontuário para omitir o nome do paciente por questões de privacidade tem a necessidade de assinar novamente o prontuário, pois a modificação acaba por invalidar a assinatura. Este

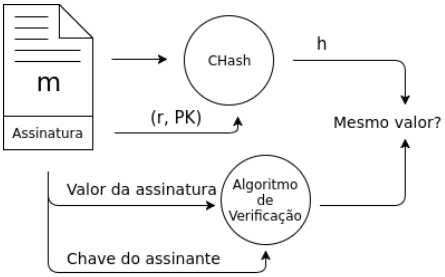

Figura 3. Diagrama do fluxo da validação de uma assinatura digital utilizando *hash* camaleão.

problema acontece devido aos algoritmos de resumo criptográficos tradicionais utilizados nos esquemas de assinatura não permitirem uma modificação do conteúdo assinado sem que se modifique a saída do algoritmo.

Para tal, foi apresentado o algoritmo de *hash* camaleão, que associa um par de chaves ao esquema de resumo criptográfico, onde qualquer um com o conhecimento da chave pública consegue calcular um valor de resumo para um documento eletrônico, e o detentor da chave privada consegue encontrar uma colisão para um documento do qual foi retirado um resumo previamente.

Inicialmente foi analisado a proposta do algoritmo, quais as funções um esquema deve ter e quais devem ser o comportamento destas funções, selecionando as principais características encontradas na literatura.

Após definir conceitualmente foram apresentados cinco formas de se construir um esquema de *hash* camaleão, demonstrando sua geração de chaves, a obtenção do resumo e da colisão, analisando a corretude da colisão e as principais características, estas voltadas a utilização destes esquemas em assinaturas digitais.

Dado a definição e a análise dos esquemas apresentados foi realizado uma comparação de todos os esquemas, levando em consideração as propriedades analisadas, concluindo que apenas dois dos cinco esquemas apresentados são efetivamente factíveis de se utilizar em um esquema assinatura digital.

Por fim foi proposto a estrutura de um esquema de assinatura digital, capaz de assinar, verificar e modificar documentos digitais, sem a necessidade de se modificar a assinatura, assim, sendo possível de se ter dois documentos diferentes dos quais resultem na mesma assinatura digital.

Referências

Ateniese, G. and De Medeiros, B. (2004a). Identity-based chameleon hash and applications. In *Financial Cryptography*, volume 4, pages 164–180.

Ateniese, G. and De Medeiros, B. (2004b). On the key exposure problem in chameleon hashes. In *SCN*, volume 4, pages 165–179.

Boneh, D. and Boyen, X. (2004). Short signatures without random oracles. In *Eurocrypt*, volume 3027, pages 56–73.

Boneh, D., Lynn, B., and Shacham, H. (2001). Short signatures from the weil pairing. *Advances in Cryptology—ASIACRYPT 2001*, pages 514–532.

Dang, Q. H. (2015). Secure hash standard. Technical report, NIST.

Gao, W., Li, F., and Wang, X. (2009). Chameleon hash without key exposure based on schnorr signature. *Computer Standards & Interfaces*, 31(2):282–285.

Krawczyk, H. and Rabin, T. (1998). Chameleon hashing and signatures. Cryptology ePrint Archive, Report 1998/010.

Krawczyk, H. and Rabin, T. (2000). Chameleon signatures. In *The Network and Distributed System Security Symposium (NDSS)*, San Diego, California. Internet Society.

SAKAI, R. and KASAHARA, M. (2003). Id based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054.

Zhang, F., Safavi-Naini, R., and Susilo, W. (2003). Id-based chameleon hashes from bilinear pairings. *IACR Cryptology ePrint Archive*, 2003:208.