Universidade Federal de Mato Grosso do Sul
Faculdade de Computação

Pedro Henrique Rocha de Oliveira

Estudo e implementação de algoritmos de criptografia

Campo Grande - MS
2 de dezembro de 2024

Pedro Henrique Rocha de Oliveira

Estudo e implementação de algoritmos de criptografia

Atividade orientada de ensino de Ciência da Computação da Faculdade de Computação da Universidade Federal de Mato Grosso do Sul.

Orientador(a): Dra. Ana Karina Dourado Salina de Oliveira

Campo Grande - MS
2 de dezembro de 2024

Resumo

Esse trabalho tem como objetivo principal apresentar conceitos fundamentais de criptografia, explorando os conceitos da criptografia simétrica, assimétrica, funções hash, cifras de bloco e assinatura digital. Serão demonstrados os principais algorítimos de cada função criptográfica apresentada durante este trabalho.

Sumário

1 Introdução **4**

2 Criptografia **4**
2.1 Criptografia simétrica 4
2.1.1 Cifras de bloco e crifas de fluxo 5
2.1.2 Algoritmo DES (Data Encryption Standard) 5
2.1.3 Algoritmo AES (Advanced Encryption Standard) 5
2.2 Criptografia assimétrica 6
2.2.1 Algoritmo RSA 6

3 Funções hash **7**

4 Assinatura digital **7**

5 Construção Esponja **8**

6 Ascon **8**
6.1 Authenticated encryption with associated data (AEAD) 9
6.1.1 Inicialização 10
6.1.2 Processamento de dados associados 10
6.1.3 Processamento de texto simples 10
6.1.4 Finalização 11
6.2 Hashing 11
6.2.1 Inicialização 12
6.3 Permutação 13
6.3.1 Adição de constantes 13
6.3.2 Camada de substituição 14
6.3.3 Camada de difusão linear 14

1 Introdução

A Segurança da Informação tem se tornado um tema cada vez mais relevante à medida que a sociedade avança para um mundo mais digital. Nesse cenário, considerando o grande volume de troca e armazenamento de dados, a demanda por melhores medidas de proteção aumenta. Uma dessas medidas é a criptografia, que segundo a Kaspersky, é a conversão de dados de um formato legível para um formato codificado com intuito de proteger sua confidencialidade e autenticidade[1]. Essa técnica tem sido poderosa para garantir sistemas mais seguros, assegurando que apenas usuários autenticados e autorizados tenham acesso aos dados enviados.

Para introduzir a criptografia, é necessário o conhecimento geral sobre segurança da informação. A segurança da informação se adapta aos requisitos específicos de cada situação. Independentemente dos agentes envolvidos, todas as partes em uma transação devem ter a certeza de que certos objetivos foram alcançados.[2] A seguir, uma lista de alguns desses objetivos:

- **Privacidade ou Confidencialidade** Manter informações em segredo, acessíveis apenas para aqueles autorizados a vê-las.
- **Integridade dos Dados** Garantir que as informações não tenham sido alteradas por meios não autorizados ou desconhecidos.
- **Disponibilidade** Assegurar que as informações e os recursos estejam acessíveis e utilizáveis quando necessários.
- **Autenticação ou Identificação de Entidade** Verificar a identidade de uma entidade.
- **Não Repúdio** Impedir a negação de compromissos ou ações anteriores.

2 Criptografia

A criptografia é definida como a prática que envolve desenvolver e usar algoritmos para proteger e ofuscar uma informação que será transmitida para outra entidade.[3] Nesse contexto, é fundamental que os algoritmos de criptografia garantam a confidencialidade e a integridade dos dados, impedindo que informações sejam acessadas ou alteradas por agentes externos.

2.1 Criptografia simétrica

A criptografia simétrica, também conhecida como criptografia de chave privada, é um método em que duas partes compartilham uma chave secreta única e utilizam-na quando pretendem enviar uma mensagem. Uma parte envia uma mensagem, ou texto simples, à outra parte, utilizando a chave secreta para criptografar a mensagem e obter o texto cifrado que é enviado ao destinatário. O destinatário utiliza a mesma chave para descriptografar o texto cifrado e recuperar a mensagem original. A simetria refere-se à utilização da mesma chave tanto para a codificação quanto para a decodificação. [4]

2.1.1 Cifras de bloco e crifas de fluxo

Os sistemas de criptografia são comumente divididos em duas categorias principais: cifras de bloco e cifras de fluxo.

- **Cifras de Bloco:** O cifrador de bloco divide o texto original em blocos de tamanho fixo. Cada bloco é uma sequência de bits, composta por 0's e 1's, com um comprimento predefinido. A cifra de bloco processa cada bloco individualmente, criptografando ou descriptografando um bloco de cada vez.

- **Cifras de fluxo:** O cifrador de fluxo utiliza uma chave para gerar um fluxo de chaves (*keystream*), que é uma sequência de bits com o mesmo comprimento do texto simples de tamanho arbitrário. A cifragem é feita realizando um XOR entre o texto simples e a *keystream*, resultando no texto cifrado. Para reverter, aplica-se novamente a operação XOR entre o texto cifrado e a *keystream*.

2.1.2 Algoritmo DES (Data Encryption Standard)

O **DES (Data Encryption Standard)** é um algoritmo de cifra de bloco que opera em blocos de dados de 64 bits. Para cada bloco de 64 bits de entrada contendo o texto original a saída é um bloco de 64 bits com texto cifrado. A chave utilizada pelo DES tem um comprimento de 56 bits (A chave é normalmente expressa como um número de 64 bits, mas cada oitavo bit é utilizado para verificação de paridade e é ignorado. Estes bits de paridade são os bits menos significativos dos bytes de chave). A chave pode ser qualquer número de 56 bits e pode ser alterada a qualquer momento.[4]

A função de rodada DES é uma rede de substituição-permutação. Mais especificamente, o cálculo de $f(k_i, R)$ com $k_i \in \{0,1\}^{48}$ e $R \in \{0,1\}^{32}$ funciona da seguinte maneira: R é expandido para um valor de 48 bits R'. Isso é realizado simplesmente duplicando metade dos bits de R.

$$R' := E(R)$$

E é chamada de função de expansão. É realizado um XOR com o valor expandido R' e k_i, que também tem 48 bits de comprimento, e o valor resultante é dividido em 8 blocos, cada um dos quais tem 6 bits de comprimento. Cada bloco é passado por uma S-box cujo a função é receber uma entrada de 6 bits e produz uma saída de 4 bits; concatenando a saída das 8 S-boxes, obtemos um resultado de 32 bits. Uma permutação de mistura é então aplicada aos bits deste resultado para obter a saída final. [4]

2.1.3 Algoritmo AES (Advanced Encryption Standard)

O algoritmo **AES (Advanced Encryption Standard)** tem um comprimento de bloco de 128 bits, e as chaves podem possuir 128 bits, 192 bits ou 256 bits. O AES é uma cifra de rodadas; o número de rodadas, denotado por N, depende do comprimento da chave. [5]

- Se o comprimento da chave for de 128 bits, então N = 10 rodadas são realizadas.

- Se o comprimento da chave for de 192 bits, então N = 12 rodadas são realizadas.

- Se o comprimento da chave for de 256 bits, então N = 14 rodadas são realizadas.

O AES, assim como o DES, é essencialmente uma rede de substituição-permutação. Durante o cálculo do algoritmo AES, uma matriz de 4 por 4 bytes chamada estado é modificada em uma série de rodadas. O estado é inicialmente definido como igual à entrada para a cifra.[4]. As operações são aplicadas ao estado em uma série de quatro estágios durante cada rodada:

1. **AddRoundKey**: Em cada rodada uma sub-chave de 128 bits é criada a partir da chave secreta e é interpretada como uma matriz de 4 por 4 bytes. A matriz de estado é atualizada fazendo um XOR com esta sub-chave.

2. **SubBytes**: Nesta etapa, cada byte da matriz de estado é substituído por outro byte de acordo com uma única tabela de pesquisa fixa S. Esta tabela de substituição (ou S-box) é uma bijeção sobre $\{0,1\}^8$

3. **ShiftRows**: Nesta etapa, os bytes em cada linha da matriz de estado são deslocados para a esquerda da seguinte forma: a primeira linha da matriz não é tocada, a segunda linha é deslocada um lugar para a esquerda, a terceira linha é deslocada dois lugares para a esquerda, e a quarta linha é deslocada três lugares para a esquerda. Todos os deslocamentos são cíclicos, de modo que, por exemplo, na segunda linha o primeiro byte se torna o quarto byte.

4. **MixColumns**: Nesta etapa, uma transformação invertível é aplicada aos quatro bytes em cada coluna. Esta transformação tem a propriedade de que se duas entradas diferem em $b > 0$ bytes, então a aplicação da transformação produz duas saídas diferindo em pelo menos 5 - b bytes.

2.2 Criptografia assimétrica

A criptografia assimétrica, também conhecida como criptografia de chave pública, é um método de criptografia de dados que usa duas chaves: uma chave pública e uma chave privada. A chave pública é usada para criptografar os dados e pode ser distribuída. A chave privada é usada para descriptografar os dados criptografados com a chave pública.[6]

A ideia da criptografia de chave pública foi concebida na década de 1970 por Diffie e Hellman. A ideia deles era que poderia ser possível criar um sistema criptográfico no qual existem duas chaves distintas. Neste sistema, a chave pública, que pode ser amplamente conhecida, é utilizada para criptografar o texto simples. Por outro lado, a chave privada, que é mantida em segredo pelo seu proprietário, é usada para decifrar o texto cifrado. Portanto, um sistema criptográfico de chave pública permitiria que qualquer pessoa criptografasse uma mensagem a ser transmitida para Bob, digamos, e apenas Bob poderia decifrar a mensagem. [5]

2.2.1 Algoritmo RSA

O mais famoso do algoritmos de chave pública é o RSA, que recebe o nome de seus três desenvolvedores: Ron Rivest, Adi Shamir e Leonard Adleman. Seus criadores eram pesquisadores no Laboratório de Ciência da Computação do MIT. O algoritmo foi anunciado pela primeira vez na coluna "Mathematical Games"de Martin Gardner na Scientific American de agosto de 1977. O artigo formal "A method for obtaining digital signatures and public-key cryptosystems"foi publicado em 1978 nas Comunicações da Associação para Maquinaria de Computação.[7]

A implementação do RSA faz uso da aritmética modular, do teorema de Euler e da função totiente de Euler. As etapas do algoritmo incluem: [8]

1. Primeiramente, o receptor escolhe dois números primos grandes p e q. O produto deles, $n = pq$, será metade da chave pública.

2. O receptor calcula $\phi(pq) = (p-1)(q-1)$ e escolhe um número e que seja relativamente primo a $\phi(pq)$. Na prática, e é frequentemente escolhido $2^{16} + 1 = 65537$, embora possa ser escolhido um número tão pequeno quanto 3 em alguns casos. e será a outra metade da chave pública.

3. O receptor calcula o inverso modular d de e módulo $\phi(pq)$. Em outras palavras, $de \equiv 1 \ (mod \ \phi(n))$. d é a chave privada.

4. O receptor distribui ambas as partes da chave pública: n e e. d é mantido em segredo.

3 Funções hash

Uma função de hash criptográfica é usada para comprimir uma mensagem de comprimento arbitrário para um texto de curto, aleatória e de comprimento fixo. A função hash é uma função pública que pode ser conhecida por qualquer pessoa. As funções de hash não podem ser usadas para criptografia, por duas razões fundamentais. Primeiro, o fato de que as funções de hash não possuem uma chave. O segundo é que as funções de hash não podem ser invertidas (elas não são funções injetoras), então um resumo da mensagem não pode ser "descriptografado"para produzir um valor de texto simples único. [5]

4 Assinatura digital

A assinatura digital é uma forma de garantir a autenticação, integridade e não repúdio dos dados, utilizando funções de hasing criptográfico e criptografia assimétrica. É possível aplicar uma assinatura digital a qualquer tipo de dado, incluindo e-mails, contratos e mensagens. Para assinar digitalmente dados, são utilizadas duas funções criptográficas[9]. O processo de assinatura digital segue os seguintes passos:

1. Gerar um hash dos dados usando uma função de hashing criptográfico.

2. Criptografar o hash gerado utilizando a chave privada do signatário, gerando a assinatura digital dos dados.

3. Enviar os dados e a assinatura para a parte verificadora.

4. Como parte do processo de verificação, a parte verificadora gera o hash dos dados usando a mesma função de hashing criptográfico.

5. Descriptografar a assinatura utilizando a chave pública da parte signatária, resultando em um hash descriptografado que só pode ser gerado pelo signatário.

6. Comparar o hash descriptografado com o hash calculado para determinar se a assinatura digital é válida.

Quando uma chave pública é usada com um certificado de chave pública (ou seja, um certificado que confirma a validade de uma chave pública e contém informações sobre o proprietário da chave), é possível identificar quem assinou o documento ou provar que ele foi assinado por um indivíduo específico.[9]

5 Construção Esponja

A função esponja, ou construção esponja, é uma função de mapeamento que recebe uma entrada de tamanho arbitrário e produz uma saída de tamanho igualmente arbitrário. Esse processo utiliza permutações de comprimento fixo e uma regra de preenchimento (padding).

A construção esponja opera em um estado de $b = r + c$ bits. O valor r é chamado de taxa de bits (*bitrate*) e o valor c é chamado de capacidade.

- **Inicialização:** A string de entrada é preenchida com uma regra de preenchimento (padding) e dividida em blocos de r bits. Em seguida, os b bits do estado são inicializados com zero.
- **Absorção:** Na fase de absorção, uma operação XOR é aplicada entre os blocos de entrada de r bits com os primeiros r bits do estado, seguido pela permutação f.
- **Extração:** Na fase de extração, os primeiros r bits do estado são retornados como blocos de saída, intercalados com a permutação f. O número de blocos de saída é escolhido livremente pelo usuário. Os últimos c bits do estado nunca são diretamente afetados pelos blocos de entrada e nunca são produzidos como saída durante a fase de extração.

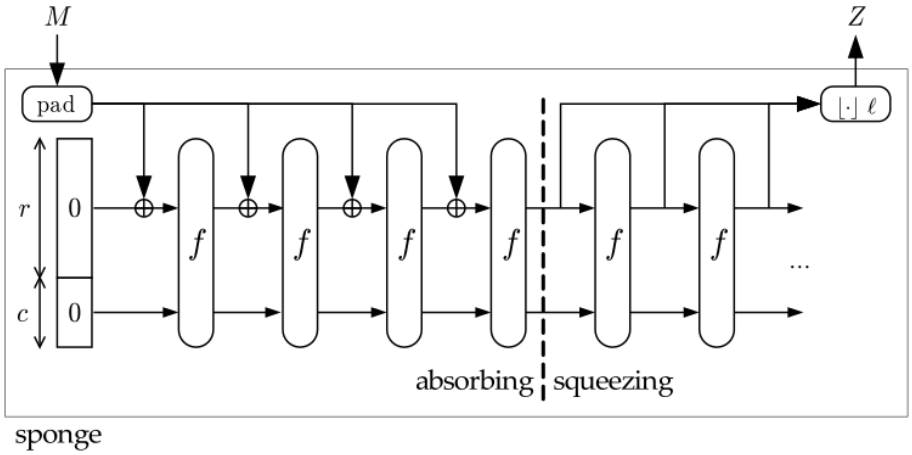

Figura 1: Função esponja [10]

6 Ascon

Ascon é um conjunto de algoritmos para autenticação e hashing projetados para ser leve e de fácil implementação. O algoritmo foi escolhido como opção primaria no portfólio

final da competição de CEASER.[11] Ascon foi projetado para ser seguro e eficiente tanto em software quanto em hardware, ocupando pouco espaço em dispositivos. [12]

Inspirado em ferramentas criptográficas amplamente analisadas e padronizadas, Ascon garante robustez e confiabilidade. É ideal para dispositivos leves que precisam realizar operações criptográficas e se comunicar com servidores de alto desempenho. Ascon é especialmente eficiente para mensagens curtas e foi projetado para resistir a ataques do mundo real, como ataques de canal lateral, garantindo que, mesmo que um invasor consiga acesso a dados internos, isso não comprometerá a chave secreta ou permitirá falsificações sem um esforço significativo. [13]

As implementações mais comum do Ascon incluem Ascon-128, Ascon-128a, Ascon-80pq e Ascon-Hash.

Tabela 1: Parâmetros dos Algoritmos de AEAD

Nome	Tamanho de bits de				Rounds	
	Chave	Nonce	Tag	Data Block	p^a	p^b
Ascon-128	128	128	128	64	12	6
Ascon-128a	128	128	128	64	12	8
Ascon-80pq	160	128	128	64	12	6

Tabela 2: Parâmetros do Algoritmo de Hash

Nome	Tamanho do hash(bits)	Datablock	p^a
Ascon-Hash	256	64	12

Todos os esquemas oferecem segurança de 128 bits e utilizam internamente a mesma permutação de 320 bits, com diferentes números de rodadas. A camada de substituição emprega um equivalente de S-box, enquanto a camada de permutação utiliza funções lineares semelhantes às do SHA-2. A permutação resultante é definida em palavras de 64 bits, usando apenas funções booleanas bit a bit (AND, NOT, XOR) e rotações dentro da palavra. [13]

6.1 Authenticated encryption with associated data (AEAD)

O Ascon, ao ser utilizado para criptografia autenticada, usa uma chave de até 160 bits. Ele define algoritmos específicos para criptografia $E_{k,r,a,b}$ e descriptografia $D_{k,r,a,b}$. Na criptografia, a entrada inclui uma chave secreta K, um nonce N de 128 bits, dados associados A e texto simples P. A saída é o texto cifrado C e uma tag de autenticação T de 128 bits.

$$(E_{k,r,a,b}(K, N, A, P) = (C, T)$$

Na descriptografia, a entrada é a chave secreta, o nonce, os dados associados, o texto cifrado e a tag. A saída é o texto simples se a tag for verificada corretamente, ou um erro $\perp$ se a verificação falhar.

$$D_{k,r,a,b}(K, N, A, C, T) \in \{P, \perp\}.$$

6.1.1 Inicialização

O estado inicial do Ascon possui 320 bits, composto pela chave secreta K de k bits, o nonce de 128 bits e um vetor de inicialização (IV). Este IV é composto pelo tamanho da chave (k), a taxa (r), o número de rodadas de inicialização e finalização (a) e o número de rodadas intermediárias (b).

$$\mathrm{IV}_{k,r,a,b} \leftarrow k\|r\|a\|b\|0^{160-k} = \begin{cases} 80400c0600000000 & \text{para Ascon-128} \\ 80800c0800000000 & \text{para Ascon-128a} \\ a0400c06 & \text{para Ascon-80pq} \end{cases}$$

$$S \leftarrow \mathrm{IV}_{k,r,a,b}\|K\|N$$

Durante a inicialização, a transformação p é aplicada a vezes no estado inicial, seguido por um XOR com a chave secreta

$$S \leftarrow p^a(S) \oplus (0^{320-k}\|K)$$

6.1.2 Processamento de dados associados

O Ascon processa os dados associados A em blocos de r bits. Ele adiciona um único bit 1 e o menor número possível de bits 0 a A para obter um múltiplo de r bits e divide em s blocos de r bits, $A_1|\ldots|A_s$. Caso A esteja vazio, nenhum preenchimento é aplicado e $s = 0$:

$$A_1, \ldots, A_s \leftarrow \begin{cases} \text{blocos de } r \text{ bits de } A\|1\|0^{r-1-(|A| \mod r)} & \text{se } |A| > 0 \\ \emptyset & \text{se } |A| = 0 \end{cases}$$

Cada bloco A_i com $i = 1, \ldots, s$ é combinado utilizando XOR com os primeiros r bits S_r do estado S, seguido pela aplicação da permutação de b rodadas p_b a S:

$$S \leftarrow p_b((S_r \oplus A_i)\|S_c), \quad 1 \le i \le s$$

Após o processamento de A_s (também se $s = 0$), uma constante de 1 bit é combinada por XOR com S:

$$S \leftarrow S \oplus (0^{319}\|1)$$

6.1.3 Processamento de texto simples

O Ascon processa o texto simples P em blocos de r bits. O processo de preenchimento (*padding*) adiciona um único bit 1 e o menor número possível de bits 0 ao texto P para que o comprimento do texto preenchido seja um múltiplo de r bits. O texto preenchido resultante é dividido em t blocos de r bits, $P_1, \ldots, P_t$:

$$P_1, \ldots, P_t \leftarrow \text{blocos de } r \text{ bits de } P\|1\|0^{r-1-(|P| \mod r)}$$

Encriptação Em cada iteração, um bloco de texto plano preenchido P_i com $i = 1, \ldots, t$ é combinado por XOR com os primeiros r bits S_r do estado interno S, seguido pela extração de um bloco de texto cifrado C_i. Para cada bloco, exceto o último, o estado interno S é transformado pela permutação p^b:

$$C_i \leftarrow S_r \oplus P_i$$

$$S \leftarrow \begin{cases} p^b b(C_i \| S_c) & \text{se } 1 \leq i < t \\ C_i \| S_c & \text{se } i = t \end{cases}$$

O último bloco de texto cifrado C_t é truncado para o comprimento do último bloco de texto simples não preenchido, de modo que seu comprimento esteja entre 0 e $r-1$ bits, e o comprimento total do texto cifrado C seja exatamente o mesmo que o do texto plano original P:

$$\tilde{C}_t \leftarrow \lfloor C_t \rfloor_{|P| \mod r}$$

Decriptação: Em cada iteração, exceto a última, o bloco de texto simples P_i é calculado realizando um XOR do bloco de texto cifrado C_i com os primeiros r bits S_r do estado interno. Em seguida, os primeiros r bits do estado interno, S_r, são substituídos por C_i. Finalmente, para cada bloco de texto cifrado, exceto o último, o estado interno é transformado pela permutação p_b:

$$P_i \leftarrow S_r \oplus C_i$$

$$S \leftarrow p^b(C_i \| S_c), \quad 1 \leq i < t$$

Para o último bloco de texto cifrado truncado $\tilde{C}_t$ com $0 \leq \ell < r$ bits, o procedimento é diferente:

$$\tilde{P}_t \leftarrow \lfloor S_r \rfloor_\ell \oplus \tilde{C}_t$$

$$S \leftarrow (S_r \oplus (\tilde{P}_t \| 1 \| 0^{r-1-\ell})) \| S_c$$

6.1.4 Finalização

A chave secreta K é combinada utilizando XOR com o estado interno e o estado é transformado pela permutação p^a usando a rodadas. O tag T consiste nos últimos 128 bits do estado combinados por XOR com os últimos 128 bits da chave K:

$$S \leftarrow p^a(S \oplus (0^r \| K \| 0^{c-k}))$$

$$T \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$$

6.2 Hashing

A função de hashing baseada em esponja e é parametrizada pela taxa r, número de rodadas a, e um limite de comprimento de saída h (h = 0 para saída ilimitada). A função de saída extensível $X_{h,r,a}$ mapeia uma mensagem M para um hash H de comprimento $\ell \leq h$.

$$X_{h,r,a}(M, \ell) = H$$

O Ascon-Hash e o Ascon-Xof usam o mesmo algoritmo, variando os parâmetros: Ascon-Hash com $h = 256$ e Ascon-Xof com $h = 0$. O parâmetro h define o comprimento de H. Assim, chamadas com os mesmos parâmetros e entradas, mas com $\ell' < \ell''$, resultam em H' e H'' com os mesmos valores para os primeiros ℓ' bits [13].

6.2.1 Inicialização

O modo de operação para hashing consiste em uma contrução esponja.

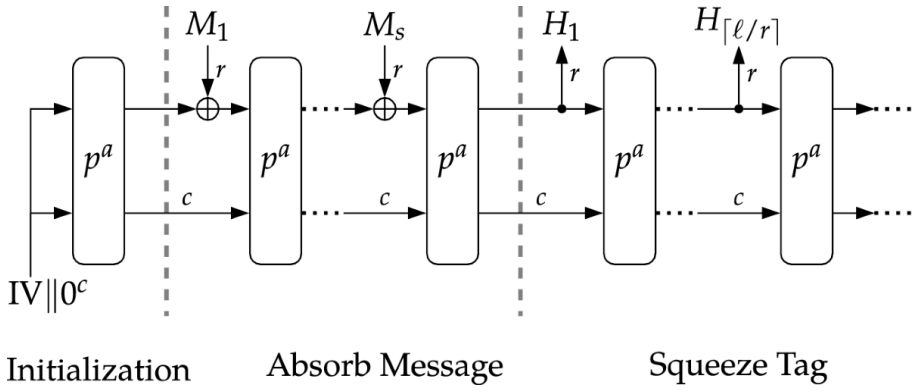

Figura 2: Modo de Hash $X_{h,r,a}$ em Ascon-Hash, Ascon-Xof

Inicialização: O estado inicial de 320 bits é definido por um IV constante que especifica os parâmetros do algoritmo em um formato semelhante ao do Ascon (incluindo $k = 0$, a taxa r e os números de rodadas a e $b = 0$, cada um sendo um inteiro de 8 bits), seguido pelo comprimento máximo de saída de h bits como um inteiro de 32 bits (com $h = 256$ para o Ascon-Hash e $h = 0$ para saída ilimitada no Ascon-Xof) e um valor zero de 256 bits. A permutação de a rodadas p_a é aplicada para inicializar o estado S:

$$IV_{h,r,a} \leftarrow 0^8\|r\|a\|0^8\|h = \begin{cases} 00400c0000000000 \text{ para Ascon-Xof} \\ 00400c0000000100 \text{ para Ascon-Hash} \end{cases}$$

$$S \leftarrow p^a(IV_{h,r,a}\|0^{256})$$

Absorção: A mensagem M é processada em blocos de r bits. O processo de preenchimento é o mesmo que para o texto simples do Ascon: ele adiciona um único bit 1 e o menor número possível de bits 0 a M para que o comprimento da mensagem preenchida seja um múltiplo de r bits. A mensagem preenchida resultante é dividida em s blocos de r bits, $M_1 \ldots M_s$:

$$M_1, \ldots, M_s \leftarrow \text{blocos de } r \text{ bits de } M\|1\|0^{r-1-(|M| \mod r)}$$

Os blocos de mensagem M_i com $i = 1, \ldots, s$ são processados da seguinte forma. Cada bloco M_i é combinado por XOR com os primeiros r bits S_r do estado S, seguido pela aplicação da permutação de p_a a S:

$$S \leftarrow p^a((S_r \oplus M_i)\|S_c), \quad 1 \le i \le s$$

Extração: A saída do hash é extraída do estado em blocos de r bits H_i até que o comprimento de saída solicitado $\ell \le h$ seja completado após $t = \lfloor \ell/r \rfloor$ blocos. Após cada extração, o estado interno S é transformado pela permutação p_a:

$$H_i \leftarrow S_r$$

$$S \leftarrow p^a(S), \quad 1 \leq i \leq t = \lfloor \ell / r \rfloor$$

O último bloco de saída H_t é truncado para $\ell \mod r$ bits (a menos que r seja divisor de ℓ) e $H = H_1 \| \ldots \| \tilde{H}_t$ é retornado:

$$\tilde{H}_t \leftarrow \lfloor H_t \rfloor_{\ell \mod r}$$

6.3 Permutação

A principal operação que ocorre durante a execução do Ascon, em ambas as variações, são as permutações de 320 bits p^a e p^b. Essas permutações, por sua vez, consistem em três etapas: p_C, p_S e p_L:

$$p = p_L \circ p_S \circ p_C$$

p^a e p^b diferem apenas no número de rodadas. O número de rodadas a e o número de rodadas b são parâmetros ajustáveis.

Para a aplicação das transformações, o estado de 320 bits S é dividido em cinco palavras de 64 bits x_i:

$$S = x_0 \| x_1 \| x_2 \| x_3 \| x_4$$

6.3.1 Adição de constantes

A etapa de adição de constante p_C adiciona uma constante de rodada c_r à palavra de registro x_2 do estado S na rodada i (veja tabela 3). Ambos os índices r e i começam do zero e usamos $r = i$ para p_a e $r = i + a - b$ para p_b:

$$x_2 \leftarrow x_2 \oplus c_r$$

p^{12}	p^8	p^6	Constante c_r	p^{12}	p^8	p^6	Constante c_r
0			00000000000000f0	6	2	0	0000000000000096
1			00000000000000e1	7	3	1	0000000000000087
2			00000000000000d2	8	4	2	0000000000000078
3			00000000000000c3	9	5	3	0000000000000069
4	0		00000000000000b4 1	10	6	4	000000000000005a
5	1		00000000000000a5	11	7	5	000000000000004b

Tabela 3: Constantes de rodada c_r usadas em cada rodada i de p^a e p^b

6.3.2 Camada de substituição

A camada de substituição p_S atualiza o estado S com 64 aplicações paralelas da S-box de 5 bits $S(x)$ a cada bit-slice dos cinco registradores $x_0 \dots x_4$ definida na figura 3.

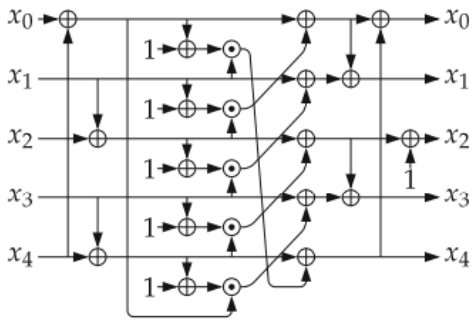

Figura 3: Ascon 5-bit S-box S

6.3.3 Camada de difusão linear

A camada de difusão linear p_L é aplicada dentro de cada palavra de registro de 64 bits x_i. Geralmente, a camada de difusão linear é implementada conforme descrito na figura 5. Ela aplica uma função linear $\Sigma_i(x_i)$ a cada palavra x_i definida na figura 4.

$$x_i \leftarrow \Sigma_i(x_i), \quad 0 \leq i \leq 4.$$

Figura 4: Camada de difusão linear do ASCON com as funcções de 64 bits $\Sigma_i(x_i)$

$$
\begin{aligned}
x_0 &\leftarrow \Sigma_0(x_0) = x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28) \\
x_1 &\leftarrow \Sigma_1(x_1) = x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39) \\
x_2 &\leftarrow \Sigma_2(x_2) = x_2 \oplus (x_2 \ggg \ 1) \oplus (x_2 \ggg \ 6) \\
x_3 &\leftarrow \Sigma_3(x_3) = x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17) \\
x_4 &\leftarrow \Sigma_4(x_4) = x_4 \oplus (x_4 \ggg \ 7) \oplus (x_4 \ggg 41)
\end{aligned}
$$

```
x0 ^= x4;    x4 ^= x3;    x2 ^= x1;
t0  = x0;    t1  = x1;    t2  = x2;    t3  = x3;    t4  = x4;
t0 = ~t0;    t1 = ~t1;    t2 = ~t2;    t3 = ~t3;    t4 = ~t4;
t0 &= x1;    t1 &= x2;    t2 &= x3;    t3 &= x4;    t4 &= x0;
x0 ^= t1;    x1 ^= t2;    x2 ^= t3;    x3 ^= t4;    x4 ^= t0;
x1 ^= x0;    x0 ^= x4;    x3 ^= x2;    x2 = ~x2;
```

Figura 5: Instruções em Pipeline para implementação bitsliced da S-box de 5 bits $S(x)$

Referências

[1] Kaspersky. *O que é criptografia de dados? Definição e explicação*. Acesso em 18 de abril de 2024. URL: https://www.kaspersky.com.br/resource-center/definitions/encryption.

[2] Alfred J. Menezes, Paul C. van Oorschot e Scott A. Vanstone. *Handbook of Applied Cryptography (Discrete Mathematics and Its Applications)*. CRC Press, 1996.

[3] IBM. *What is cryptography?* Acesso em 18 de abril de 2024. URL: https://www.ibm.com/topics/cryptography.

[4] Jonathan Katz e Yehuda Lindell. *Introduction to Modern Cryptography 2nd Edition*. Chapman e Hall/CRC, 2014.

[5] Douglas R. Stinson e Maura B. Paterson. *Cryptography : theory and practice*. Chapman e Hall/CRC, 2018.

[6] *Asymmetric encryption*. Acesso em 13 de abril de 2024. URL: https://encyclopedia.kaspersky.com/glossary/asymmetric-encryption/.

[7] Chris Christensen. *Introduction to RSA and to Authentication*. 2006. URL: https://www.nku.edu/~christensen/section%2026%20RSA.pdf.

[8] *RSA Encryption*. Acesso em 14 de abril de 2024. URL: https://brilliant.org/wiki/rsa-encryption/.

[9] Valentin Mulder et al. *Trends in Data Protection and Encryption Technologies*. Springer, 2023.

[10] *The sponge and duplex constructions*. URL: https://keccak.team/sponge_duplex.html.

[11] *The CAESAR committee, CAESAR: competition for authenticated encryption: security, applicability and robustness (2014)*. URL: https://competitions.cr.yp.to/caesar-submissions.html.

[12] *Ascon - Lightweight Authenticated Encryption and Hashing*. Acesso em 20 de junho de 2024. URL: https://ascon.iaik.tugraz.at.

[13] Christoph Dobraunig et al. "ASCON v1.2: Lightweight Authenticated Encryption and Hashing". Em: *Journal of Cryptology* (2020).