Funções de Dispersão

As funções de dispersão ("hash functions") caracterizam-se por aplicarem um dado conjunto de valores (de grande dimensão) num outro conjunto de valores, não necessariamente do mesmo tipo, de menor dimensão.
Uma das utilizações para funções deste tipo é dado pela criação de índices de pesquisa, os quais permitem procuras mais eficientes em grandes conjuntos de dados.
Dado que se tem um conjunto de chegada de (muito) menor dimensão do que o conjunto de partida vão ocorrer obrigatoriamente *colisões*, os quais devem ser minimizados por uma eficiente *dispersão dos valores*.

Colisões: ocorrem quando diferentes valores de partida coincidem no mesmo valor de chegada.

Dispersão: os valores de entrada devem ser dispersos de forma a minimizar a ocorrência de colisões.

Funções de Dispersão Criptográficas

O papel das funções de dispersão na criptografia está relacionado em especial com os aspectos de verificação da *integridade da informação* e *autenticação* das mensagens.

Dado uma mensagem, uma função de dispersão produz um resultado designado por *valor de dispersão* ("hash-code").

Mais precisamente uma função de dispersão h aplica uma sequência de bits de comprimento arbitrário, em sequências de comprimento fixo n. Para o caso em que se tenha $D \xrightarrow{h} R$ com $|D| > |R|$ a função é do tipo muitos-para-um, isto é, a existência de colisões (pares de valores de entrada diferentes com o mesmo valor de saída) é inevitável.

Funções de Dispersão Criptográficas

A ideia básica da utilização de funções de dispersão em criptografia é a de utilizar o valor de dispersão como uma imagem representativa de uma dada sequência de entrada, servindo para a identificar.

É usual usarem-se os termos de *marcas digitais* ("imprints"), *impressão digital* ("digital fingerprint"), *versão compacta* ("message digest").

Funções de Dispersão Criptográficas

As funções de dispersão dividem-se em duas classes: *funções de dispersão sem chave*, as quais têm um único valor de entrada; e *funções de dispersão com chave*, as quais têm como valores de entrada, uma dada chave secreta, e a informação a ser tratada.

Um exemplo de utilização de funções de dispersão com chaves é a criação de senhas de acesso ("passwords") aos sistemas informáticos, por exemplo, sistemas de gestão de bases de dados, sistemas operativos, etc.

Definição

As funções de dispersão caracterizam-se por um dado conjunto de propriedades.

Definição (Função de Dispersão)

Uma função de dispersão (em criptografia) é uma função h que tem, no mínimo, as duas propriedades seguintes:

1. Compressão - *h aplica uma entrada x de comprimento (finito) arbitrário, numa saída $h(x)$ de comprimento fixo n (ambos os valores em bits).*
2. Facilidade de cálculo - *dado h e uma entrada x, $h(x)$ é fácil de calcular.*

Definição

A definição e classificação das funções de dispersão é melhor compreendida tendo em vista as utilizações para as mesmas:

MDCs *Códigos de Detecção de Modificações*, também referidos como MICs (códigos de integridade das mensagens). O propósito de um MDC é (informalmente) de providenciar uma "imagem" ou valor de dispersão para uma mensagem de forma que seja possível estabelecer a integridade (não manipulação) de uma mensagem recebida.

Os MDCs são uma sub-classe das funções de dispersão sem chave.

MACs *Códigos de Autenticação de Mensagens*. O propósito de um MAC é (informalmente) de assegurar, sem a utilização de mecanismos adicionais, a origem da mensagem assim como a sua integridade.

Os MACs são uma sub-classe das funções de dispersão com chave.

MDCs

Os MDCs podem ainda ser classificados em:

- *OWHFs — Funções de Dispersão de um só sentido*, para este tipo de MDCs é difícil achar o valor de entrada de um dado valor de dispersão.
- *CRHFs — Funções de Dispersão Resistentes às Colisões*, para este tipo de MDCs é difícil achar um dado par de valores de entrada que tenham o mesmo valor de dispersão.

MACs

Aplicações adicionais das funções de dispersão com chave incluem:

- Protocolos de "santo e senha", para calcular as respostas (senhas) que dependem de uma chave e de uma mensagem (santo).
- Confirmações de Chaves Secretas.

Assume-se (princípio de Kerckhoff) que o algoritmo da função de dispersão é do conhecimento público.

Temos então que, no caso dos MDCs, dado um valor de entrada, todos podem calcular o respectivo valor de dispersão.

No caso dos MACs, dado um valor de entrada, qualquer pessoa que tenha o conhecimento da chave pode calcular o valor de dispersão.

Propriedades Básicas

Além das propriedades já referidas (definição) temos ainda algumas importantes para os fins em vista:
Para uma função de dispersão h com entradas x, x' e saídas y, y':

- *resistência à pré-imagem* — para essencialmente todo o valor de saída pré-especificado, é computacionalmente inviável achar um valor de entrada cuja valor de dispersão é o referido valor de saída. Isto é, é inviável, para um dado valor de saída y do qual não se sabe o valor de entrada, achar uma pré-imagem x' tal que $h(x') = y$

Isto significaria que um adversário poderia facilmente pré-computar valores de saída para um pequeno conjunto de valores de entrada, e como tal inverter trivialmente a função de dispersão para esse valores de saída.

Propriedades Básicas

- *resistência à 2ª pré-imagem* — é computacionalmente inviável achar um segundo valor de entrada que tenha o mesmo valor de dispersão que um outro valor de entrada, isto é, dado um x achar uma segunda pré-imagem $x' \neq x$ tal que $h(x) = h(x')$.
- *resistência à colisão* — é computacionalmente inviável achar duas entradas distintas x, x' com um valor de dispersão igual, isto é, tal que $h(x) = h(x')$.
  Note-se que aqui há liberdade de escolha em ambos os valores de entrada.

resistência à 2ª pré-imagem $\doteq$ *resistência à colisão fraca*.
resistência à colisão $\doteq$ *resistência à colisão forte*.

Definições

Definição (OWHF)

Uma função de dispersão unidireccional (OWHF) é uma função de dispersão (fácil de computar e com factor de compressão) com as propriedades adicionais de resistência à pré-imagem e à 2ª pré-imagem.

Definição (CRHF)

Uma função de dispersão resistente às colisões (CRHF) é uma função de dispersão (fácil de computar e com factor de compressão) com as propriedades adicionais de resistência à 2ª pré-imagem e à colisão.

Embora, na prática, as funções de dispersão do tipo CRHF tenham a propriedade adicional de resistência à pré-imagem, por razões técnicas, esta propriedade não está incluída na definição.

Exemplo

Vejamos como algumas funções de dispersão simples respondem a estes questões:

Propriedades de Funções de Dispersão

1. Uma função de verificação de somas módulo 32 (somas de 32bits de todas as palavras de 32bits de uma mensagem), é uma função que é fácil de calcular, que comprime a entrada, mas não é resistente à pré-imagem.
2. A função $g(x) = x^2 \mod n$ é resistente à pré-imagem, mas não oferece compressão, nem é resistente à 2ª pré-imagem.
3. Dado uma cifra por blocos E, podemos definir uma função de dispersão; $f(x) = E_k(x) \oplus x$ para uma qualquer chave k fixa. Este função é resistente à pré-imagem, e à 2ª pré-imagem, não oferecendo, no entanto, compressão.

MACs

Definição (Código de Autenticação de Mensagem (MAC))

Um algoritmo, código de autenticação de mensagem, é uma família de funções h_k parametrizadas por uma chave secreta k, com as seguintes propriedades:

1. facilidade de cálculo — *para uma dada função h_k, dado a chave k e uma entrada x, então $h_k(x)$ é fácil de calcular. O resultado é designado por valor-MAC, ou MAC.*
2. compressão — *h_k aplica uma entrada x de comprimento finito mas arbitrário a uma saída $h_k(x)$ de comprimento fixo de n bits.*

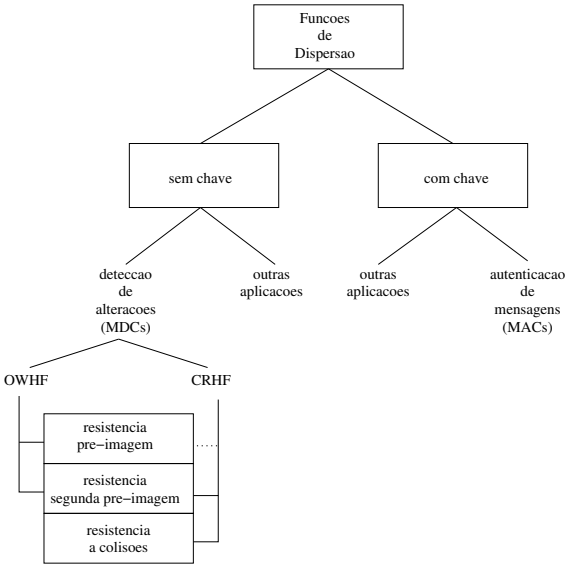
Dada a especificação de uma família de funções h, para todo o possível valor da chave secreta k, verifica-se a seguinte propriedade

3. resistência ao cálculo — *dado zero ou mais pares (texto,valor-MAC), $(x_i, h_k(x_i))$, é inviável computacionalmente calcular um qualquer par (texto, valor-MAC), $(x, h_k(x))$ para uma qualquer entrada $x \neq x_i$ (inclusive para os casos $h_k(x) = h_k(x_i)$, para algum i).*

Se esta última propriedade não se verifica um algoritmo MAC está sujeito a ser falsificado.

Enquanto a resistência ao cálculo implica a não recuperação da chave, o contrário não se verifica.

Taxonomia



Objectivos dos Adversários vs. MDCs

Os objectivos de um adversário que queira "atacar" um MDC são os seguintes:

1. *atacar um OWHF*:
   - dado um valor de dispersão y, achar a pré-imagem x tal que $y = h(x)$;
   - ou dado um par $(x, h(x))$, achar a segunda pré-imagem x' tal que $h(x') = h(x)$.
2. *atacar um CRHF*:
   - achar uns quaisquer valores de entrada x, x', tais que $h(x') = h(x)$.

Objectivos dos Adversários vs. MACs

Os objectivos de um adversário correspondentes para um algoritmo MAC são:

1. *atacar um MAC*: sem conhecimento prévio da chave secreta k, calcular um novo par (texto,valor-MAC), $(x, h_k(x))$ para um dado texto $x \neq x_i$, dados um ou mais pares $(x_i, h_k(x_i))$.

Objectivos dos Adversários vs. MACs (cont)

De forma similar às cifras existem alguns cenários de ataque que permitem melhorar as possibilidades de quebrar um MAC.

1. *ataque texto conhecido*. Um ou mais pares (texto, valor-MAC) $(x_i, h_k(x_i))$ estão disponíveis.
2. *ataque texto escolhido*. Um ou mais pares (texto, valor-MAC) $(x_i, h_k(x_i))$ estão disponíveis para x_i escolhido pelo adversário.
3. *ataque adaptativo texto escolhido*. O x_i é escolhido pelo adversário, podendo este basear novas escolhas nos resultados já obtidos até ao momento.

Do ponto de vista de certificação de um MAC este deve suportar o ataque adaptativo texto escolhido independentemente de um tal ataque poder ser, ou não, montado.

Tipos de Falsificação (selectivo, existencial)

Quando é possível falsificar um MAC, a severidade das consequências práticas diferem de acordo com o grau de controlo do adversário sobre o valor de x para o qual o MAC pode ser falsificado.

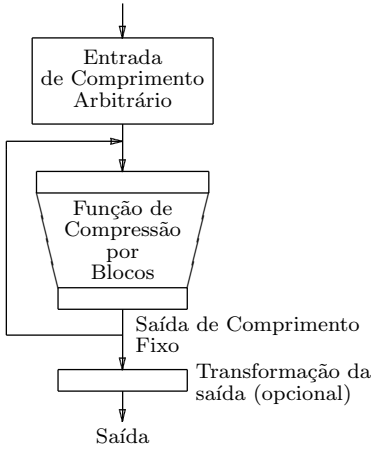
1. *Falsificação selectiva* — ataques aonde um adversário é capaz de produzir um novo par (texto, valor-MAC) para um texto da sua escolha, mesmo que só controle parcialmente essa escolha.
2. *Falsificação existencial* — ataques aonde um adversário é capaz de produzir um novo par (texto, valor-MAC), mas nos quais não tem controlo sobre o valor do texto.

A obtenção da chave secreta do MAC é o ataque mais eficaz, possibilitando, trivialmente o ataque de falsificação selectiva.

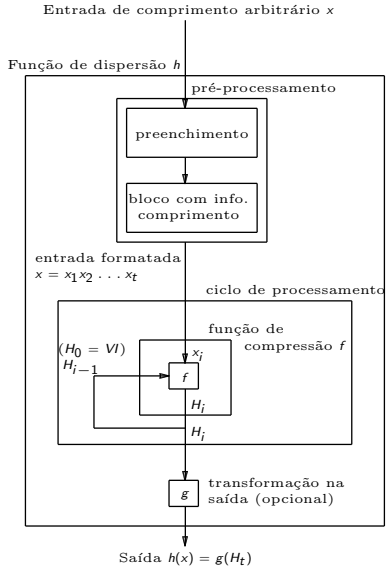
Uma falsificação de um MAC permite ao adversário falsificar um texto e fazê-lo passar por autêntico.

Modelo Genérico

O modelo genérico para funções de dispersão sem chave assemelha-se aos modos de funcionamento das cifras.



Modelo Genérico (detalhado)



Modelo Genérico

1. Uma entrada de comprimento finito arbitrário x é dividida em blocos de comprimento fixo de r-bits, este processo vai implicar um preenchimento do último bloco (métodos de preenchimento) e pode incluir, por motivos de segurança, o comprimento total (um múltiplo de r).
2. A função interna f, a função de compressão, combina então os sucessivos blocos x_i com um resultado obtido no passo anterior para obter um valor intermédio de comprimento n. É necessário definir um valor de inicialização H_0.
3. O valor de dispersão é obtido por aplicação da função de transformação g ao resultado final do passo anterior. É usual a função g ser a função identidade.

   Temos então que:

$$H_0 = VI; \qquad H_i = f(H_{i-1}, x_i), \quad 1 \leq i \leq t; \qquad h(x) = g(H_t)$$

Facto

Uma qualquer função de compressão f que seja resistente às colisões pode ser "extendida" para a forma de uma função de dispersão resistente às colisões.

Meta-Método

Meta-modo de Merkle para Funções de Dispersão

Entrada: função de compressão f, resistente às colisões;
Saída: função de dispersão sem chave h, resistente às colisões.
Suponha-se que f aplica uma entrada de $(n + r)$-bits, a uma saída de n bits (128 e 512, são valores usuais). A partir de f constrói-se uma função de dispersão h com valores de dispersão de comprimento n-bits, do seguinte modo:

1. partir a entrada x, de comprimento b-bits, em blocos $x_1 x_2 \ldots x_t$ de comprimento r-bits, preenchendo o último bloco com 0s, se necessário;
2. Define-se um bloco extra x_{t+1}, o *bloco comprimento*, no qual se coloca a representação binária de b justificada à direita (supõe-se que $b < 2^r$).
3. Designando por O^j a sequência de bits com j 0s, e $\|$ a concatenação de duas sequências de bits, define-se x, o valor de dispersão de n-bits, como sendo $h(x) = H_{t+1} = f(H_t \| x_{t+1})$ dado por:

$$H_0 = 0^n; \qquad H_i = f(H_{i-1} \| x_i), \quad 1 \leq i \leq t + 1.$$

Fortalecimento do Meta-Método

A demonstração de que a função de dispersão resultante h é resistente às colisões sempre que a função de compressão f o é necessita da inclusão de um *bloco comprimento*.
A adição desse bloco é designada por o *Fortalecimento de Merkle-Damgård*.

Fortalecimento Merkle-Damgård

Antes de se proceder ao cálculo do valor de dispersão para uma mensagem $x = x_1 x_2 \ldots x_n$ (com o blocos de comprimento r bits), tendo a mensagem um comprimento de b bits, define-se um bloco extra x_{t+1}, o *bloco comprimento*, no qual se coloca a representação binária de b justificada à direita (supõe-se que $b < 2^r$).

Funções de Dispersão sem Chave (MDCs)

Tendo em conta a sua função interna de compressão os MDCs podem ser divididos de forma grosseira em três categorias:

- MDCs baseadas em Cifras por Blocos;
- MDCs construídas de raiz;
- MDCs baseadas na aritmética modular.

MDCs Baseados em Cifras por Blocos

A ideia base é a de re-utilizar uma componente já presente na nossa "caixa de ferramentas criptográfica", construindo deste modo uma função de dispersão sem grandes custos de programação.

Dado o carácter próprio das cifras (nomeadamente a sua invertibilidade) não se sabe exactamente quais são os requisitos necessários a impor para produzir uma função de dispersão. Caso a caso tem-se o conhecimento de alguns requisitos mínimos.

Nota

Uma (n, r) Cifra por blocos é, uma função invertível de uma entrada de texto claro de n-bits, para um texto cifrado de n-bits, utilizando uma chave de r-bits. Se E é uma tal cifra, então $E_k(x)$ designa a cifração de x utilizando a chave k.

MDCs Baseados em Cifras por Blocos

As cifras por blocos usuais utilizam um bloco de 64-bits, vai-se usar esse valor como referência.
As funções de dispersão vão-se dividir naquelas que produzem um bloco simples de n-bits, e aquelas que produzem um bloco duplo de $2n$-bits.

- OWHF (bloco simples), utilizando cifras por blocos, com um bloco de (aproximadamente) $n = 64$ bits;
- CRHF (bloco duplo), utilizando cifras por blocos, com um bloco de comprimento (aproximadamente) $n = 128$ bits.

A utilização de uma função de dispersão de bloco duplo deve-se ao facto de haver muitas cifras disponíveis com um bloco de comprimento de 64-bits, sendo que um bloco desse comprimento não garante a resistência às colisões por parte da função de dispersão.

Implementação de MDCs

Uma característica importante na implementação de funções de dispersão através de cifras por blocos é o número de operações de cifração requeridas para produzir um valor de dispersão com um comprimento de bloco igual ao bloco da cifra.

Definição (Débito de uma Função de Dispersão)

Seja h uma função de dispersão construída a partir de uma cifra por blocos, com uma função de compressão f que perfaz s encriptações por blocos, para processar cada um dos sucessivos blocos de n-bits da mensagem. Então o débito *de h é* $\frac{1}{s}$.

Implementação de MDCs (cont.)

Função de Dispersão	(n, k, m)	**Débito**
Matyas-Meyer-Oseas	(n, k, n)	1
Davis-Meyer	(n, k, n)	k/n
Miyaguchi-Preneel	(n, k, n)	1
MDC-2 (com DES)	$(64, 56, 128)$	$1/2$
MDC-4 (com DES)	$(64, 56, 128)$	$1/4$

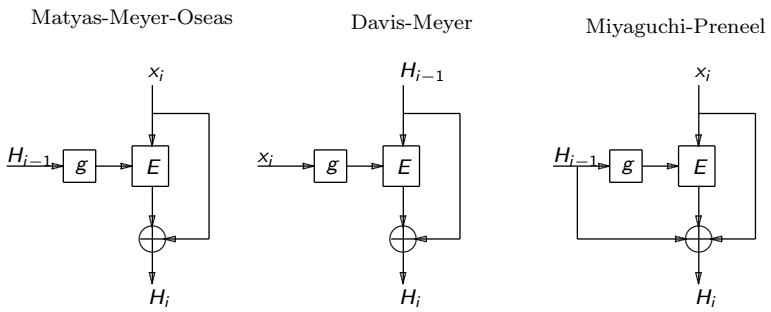
(n, k, m)
n-bits - comprimento do bloco da cifra.
k-bits - comprimento da chave da cifra por blocos.
m-bits - comprimento do valor de dispersão.

MDCs de Comprimento Simples e Débito 1

Os três primeiros algoritmos referidos na tabela têm em comum:

- uma cifra por blocos de comprimento n-bits, E_K, parametrizada por uma chave (simétrica) K;
- uma função g que aplica entradas com n-bits a chaves K, apropriadas à cifra E_K. Se as chaves K têm comprimento n-bits a função g pode ser a função identidade;
- um valor inicial VI (n-bits, usualmente), apropriado à cifra usada.

MDC — M-M-O & D-M & M-P



MDC — Matyas-Meyer-Oseas

Função de Dispersão Matyas-Meyer-Oseas

Entrada: uma sequência de bits x;
Saída: valor de dispersão, $h(x)$ com n bits.

1. A entrada x é dividida em blocos de comprimento n bits, com preenchimento se necessário. Após o pré-processamento temos um mensagem dividida em t blocos de n bits, $x = x_1 x_2 \ldots x_t$.
2. É necessário pré-definir um valor inicial VI.
3. O valor de saída H_t é definido por:

$$H_0 = VI; \quad H_i = E_{g(H_{i-1})}(x_i) \oplus x_i,\ 1 \leq i \leq t.$$

MDC — Davis-Meyer

Função de Dispersão Davis-Meyer

Entrada: uma sequência de bits x;
Saída: valor de dispersão, $h(x)$ com n bits.

1. A entrada x é dividida em blocos de comprimento k bits, aonde k é o comprimento da chave, com preenchimento se necessário. Após o pré-processamento temos um mensagem dividida em t blocos de k bits, $x = x_1 x_2 \ldots x_t$.
2. É necessário pré-definir um valor inicial, de comprimento n-bits, VI.
3. O valor de saída H_t é definido por:

$$H_0 = VI; \quad H_i = E_{x_i}(H_{i-i}) \oplus H_{i-1},\ 1 \leq i \leq t.$$

MDC — Miyaguchi-Preneel

Função de Dispersão Miyaguchi-Preneel

Entrada: uma sequência de bits x;
Saída: valor de dispersão, $h(x)$ com n bits.

1. A entrada x é dividida em blocos de comprimento n bits, com preenchimento se necessário. Após o pré-processamento temos um mensagem dividida em t blocos de n bits, $x = x_1 x_2 \ldots x_t$.
2. É necessário pré-definir um valor inicial VI.
3. O valor de saída H_t é definido por:

$$H_0 = VI; \quad H_i = E_{g(H_{i-1})}(x_i) \oplus x_i \oplus H_{i-1},\ 1 \leq i \leq t.$$

Exercício Prático 8

Implemente a função de dispersão Matyas-Meyer-Oseas (ou Davis-Meyer, ou Miyaguchi-Preneel), utilizando para tal uma das cifra por blocos já implementadas, por exemplo a cifra produto definida no exercício prático 7.

- Preenchimento não âmbiguo.
- Interface Entrada/Saída: ficheiros e linha de comando.
- Implementação em C (ou C++).

MDCs de Comprimento Duplo

As funções de dispersão MDC-2 e MDC-4 requerem 2 e 4 operações de uma cifra por blocos, respectivamente.

- Empregam duas ou quatro iterações de um esquema Matyas-Meyer-Oseas (comprimento simples) como forma de se obter uma esquema de comprimento duplo.
- Caso se use a cifra DES (concepção original) produzem uma função de dispersão de 128 bits.
- É possível usar outras cifras por blocos.

MDC-2 e MDC-4

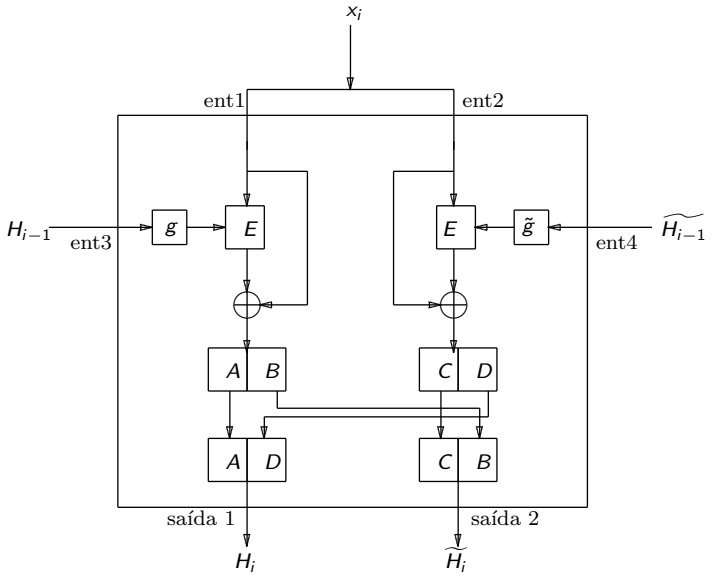
Os esquemas MDC-2 e MDC-4 fazem uso das seguintes componentes.

- A cifra DES como cifra por blocos E_K de comprimento $n = 64$bits, parametrizado por uma chave K de comprimento $k = 56$bits.
- duas funções g e $\tilde{g}$ que aplicam um valor de 64bits U para um valor de uma chave DES apropriadas (56bits) do seguinte modo.
  Para $U = u_1 u_2 \ldots u_{64}$, apagam-se todos os oitavos bits começando pelo u_8 e colocando o segundo e o terceiro bits a '10' para g e '01' para $\tilde{g}$.

$$\begin{aligned} g(U) &= u_1 10 u_4 u_5 u_6 u_7 u_9 u_{10} \ldots u_{63} \\ \tilde{g}(U) &= u_1 01 u_4 u_5 u_6 u_7 u_9 u_{10} \ldots u_{63} \end{aligned}$$

Garante-se deste modo que os valores encontrados não são chaves fracas da cifra DES.

MDC-2



MDC-2 — Algoritmo

Função de Dispersão MDC-2 (baseada em DES)

Entrada: uma sequência de bits x de comprimento $r = 64t$ bits, com $t \geq 2$.
Saída: valor de dispersão, $h(x)$ com 128 bits.

1. Dividir a entrada x em blocos de 64 bits, $x = x_1 x_2 \ldots x_t$.
2. Escolher um par de constantes (não secretas) VI e $\widetilde{VI}$, de um conjunto de valores recomendados. Por exemplo:

$$VI = 0x5252525252525252, \quad \widetilde{VI} = 0x2525252525252525.$$

3. Sejam C_i^e, C_i^d, a metade esquerda e direita (32bits) de C respectivamente. A saída $h(x) = H_t \| \widetilde{H_t}$ é dada por (com $1 \leq i \leq t$):

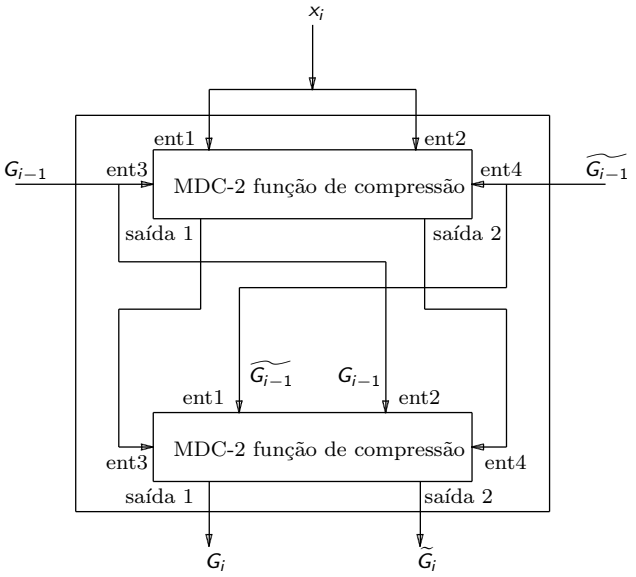
$$H_0 = VI; \quad k_i = g(H_{i-1}); \quad C_i = E_{k_i}(x_i) \oplus x_i; \quad H_i = C_i^e \| \widetilde{C_i^d}$$
$$\widetilde{H_0} = \widetilde{VI}; \quad \widetilde{k_i} = \tilde{g}(H_{i-1}); \quad \widetilde{C_i} = E_{\widetilde{k_i}}(x_i) \oplus x_i; \quad \widetilde{H_i} = \widetilde{C_i^e} \| C_i^d.$$

MDC-4

A função de dispersão MDC-4 é construída recorrendo-se às funções de compressão do MDC-2.
Uma iteração da função de compressão MDC-4 consiste de duas aplicações sequenciais da função de compressão MDC-2, aonde:

- as duas entradas de 64 bits para o primeiro MDC-2 são ambas o próximo bloco de 64 bits da mensagem;
- as chaves para o primeiro MDC-2 são derivadas das saídas da anterior iteração do MDC-4;
- as chaves para o segundo MDC-2 são derivadas das saídas do primeiro MDC-2;
- as duas entradas de 64 bits para o segundo MDC-2 são as saídas, trocando a ordem relativa da anterior iteração do MDC-4.

MDC-4



MDC-4, Algoritmo

Função de Dispersão MDC-4 (baseada em DES)

Entrada: uma sequência de bits x de comprimento $r = 64t$ bits, com $t \geq 2$.
Saída: valor de dispersão, $h(x)$ com 128 bits.

1. Dividir a entrada x em blocos de 64 bits, $x = x_1 x_2 \ldots x_t$.
2. Escolher um par de constantes (não secretas) VI e $\widetilde{VI}$, de um conjunto de valores recomendados. Por exemplo:

$$VI = 0x5252525252525252, \quad \widetilde{VI} = 0x2525252525252525.$$

3. Sejam C_i^e, C_i^d, a metade esquerda e direita (32bits) de C respectivamente. A saída $h(x) = G_t \| \widetilde{G_t}$ é dada por (com $1 \leq i \leq t$):

$$\begin{array}{lll} G_0 = VI; & \widetilde{G_0} = \widetilde{VI} & \\ k_i = g(G_{i-1}); & C_i = E_{k_i}(x_i) \oplus x_i; & H_i = C_i^e \| \widetilde{C_i^d} \\ \widetilde{k_i} = \widetilde{g}(\widetilde{G_{i-1}}); & \widetilde{C_i} = E_{\widetilde{k_i}}(x_i) \oplus x_i; & \widetilde{H_i} = \widetilde{C_i^e} \| C_i^d \\ j_i = g(H_i); & D_i = E_{j_i}(\widetilde{G_{i-1}}) \oplus \widetilde{G_{i-1}}; & G_i = D_i^e \| \widetilde{D_i^d} \\ \widetilde{j_i} = \widetilde{g}(\widetilde{H_i}); & \widetilde{D_i} = E_{\widetilde{j_i}}(G_{i-1}) \oplus G_{i-1}; & \widetilde{G_i} = \widetilde{D_i^e} \| D_i^d. \end{array}$$

Funções de Dispersão

Algoritmos construídos de raiz para servirem como funções de dispersão criptográficas:

- MD4;
- MD5;
- SHA-1;
- RIPEMD-160;

São concebidos tendo em vista uma optimização da velocidade de processamento.

MD4; SHA-1

MD4 "Message Digest Algorithm", foi concebida especialmente para máquina de 32bits, questões de segurança levaram à implementação do MD5 como uma variante (mais segura) do MD4.

SHA-1 "Secure Hash Algorithm", é baseado no MD4, foi proposto pela agência Americana de normas (NIST), para utilização governamental.

RIPEMD-160 é baseada no MD4, e pretende ser uma versão melhorada desta, tendo já em conta os conhecimentos adquiridos na análise das funções de dispersão MD4 e MD5.

Funções de Dispersão baseadas em Aritmética Modular

A ideia básica de uma função de dispersão baseada em aritmética modular é a de construir uma função de dispersão iterada usando a aritmética modular (módulo M) como base da função de compressão.

- re-utilização de programas e sistemas informáticos já existentes (sistemas de chave pública);
- A possibilidade de adaptar a cifra (números de bits) às necessidades de segurança sentidas.

O principal defeito passa por uma menor velocidade de processamento.

MASH

- MASH-1 ("Modular Arithmetic Secure Hash, algorithm 1") é uma função de dispersão baseada em aritmética modular.
- Foi proposto para inclusão numa norma ISO/IEC.
- Envolve a utilização de uma operação modular semelhante ao RSA e cujo comprimento em bits afecta a segurança da mesma.
- M deve ser díficil de factorizar, e para um dado M a segurança está baseada na dificuldade de extrair raízes modulares.
- O comprimento em bits de M determina também o comprimento dos blocos a processar, assim como o comprimento do valor de dispersão.
- Dado que é uma proposta recente a sua segurança é ainda uma questão em aberto.

MASH — Algoritmo

MASH-1 (versão de 1995)

Entrada: uma sequência de bits x de comprimento $0 \leq b < 2^{n/2}$ bits.
Saída: valor de dispersão, $h(x)$ com n bits, em que n é aproximadamente o comprimento em bits do módulo de M.

1. *Inicialização:*
   - Fixar um $M = pq$ de comprimento m bits, aonde p e q são primos escolhidos aleatoriamente (secretos) tais que a factorização de M é intratável;
   - Definir o comprimento (em bits) n do valor de dispersão como sendo maior múltiplo de 16 menor do que m ($n = 16n' < m$);
   - $H_0 = VI = 0$;
   - $A = 0xf00\ldots0$, uma constante de n bits.
2. *Preenchimento e fortalecimento-MD:* preencher x com 0-bits, se necessário, para obter uma mensagem de comprimento $t \cdot n/2$, com o menor valor possível para $t \geq 1$. Dividir a mensagem em blocos de $(n/2)$ bits, $x_1 x_2 \ldots x_t$, juntar um bloco final x_{t+1} contendo a representação de b em $(n/2)$ bits.

MASH — Algoritmo (cont.)

3. *Expansão:* expandir cada x_i para um bloco de n bits y_i, particionando-o em blocos de 4 bits e antecedendo cada um dos blocos por um bit 1, com excepção de y_{t+1} aonde a sequência de bits a inserir é 1010 (e não 1111).
4. *Função de Compressão:* para $1 \leq i \leq t+1$ aplica-se duas entradas de n bits (H_{i-1}, y_i) a uma saída de n bits do seguinte modo (com $e = 2$):

$$H_i \leftarrow ((((H_{i-1} \oplus y_i) \vee A)^e \mod M) \dashv n) \oplus H_{i-1}.$$

5. *Saída:* o valor de dispersão é o bloco de n bits H_{t+1}.

Notação: $\oplus$ é o *ou exclusivo*, $\vee$ é o *ou*, ambos em termos de bits; $m \dashv n$, é retirar dos n bits mais à direita de uma sequência de m bits.

Define-se a variante *MASH-2* fixando o valor de e (passo 4) em $e = 2^8 + 1$.

Exercício Prático 9

Implemente a função de dispersão MASH-2.

- Preenchimento e fortalecimento-MD.
- Interface Entrada/Saída: ficheiros e linha de comando.
- Implementação em C (ou C++).

Funções de Dispersão com Chave (MACs)

As funções de dispersão com chave (secreta) cujo fim específico é o de autenticação das mensagens são designados por MACs.

Um nível de segurança adequado (dependendo dos recursos computacionais disponíveis ao adversário) é dado por MACs com comprimento em bits relativamente pequeno (por exemplo 32 bits para um MAA), ou chaves pequenas (por exemplo, 56 bits para MACs baseados no DES-CBC).

Um limite superior na segurança dos MACs é dado pelo resultado seguinte.

Ataque Aniversário num MAC

Seja h um algoritmo MAC baseado numa função de compressão iterada, a qual tem uma variável interna de n bits, e é determinístico. Então pode-se falsificar o MAC utilizando $O(2^{n/2})$ pares (texto claro,MAC) mais um número v de pares (texto claro,MAC) os quais, dependendo do h, variam entre 1 e aproximadamente 2^{n-m}.

MACs baseados em Cifras por Blocos

Os algoritmos MACs mais comuns estão baseados em cifras por blocos, recorrendo a uma sequência de aplicações da cifra. Quando a cifra usada é a cifra DES tem-se $n = 64$, e a chave do MAC é a chave de 56 bits da cifra DES.

Algoritmo CBC-MAC

Algoritmo CBC-MAC

Entrada: x; cifra E; chave secreta da cifra E.
Saída: valor de dispersão (MAC) para x com comprimento n bits, sendo que esse é o comprimento do bloco da cifra E.

1. Dividir a entrada x em blocos de 64 bits, $x = x_1 x_2 \dots x_t$, fazendo preenchimento no fim (por exemplo o método sem, ambiguidade).
2. Processamento CBC. Denotando E_k o encriptar usando a cifra E com chave k calcula-se o bloco H_t do seguinte modo:

$$\begin{aligned} H_1 &\leftarrow E_k(x_1) \\ H_i &\leftarrow E_k(H_{i-1} \oplus x_i), \quad 2 \leq i \leq t \end{aligned}$$

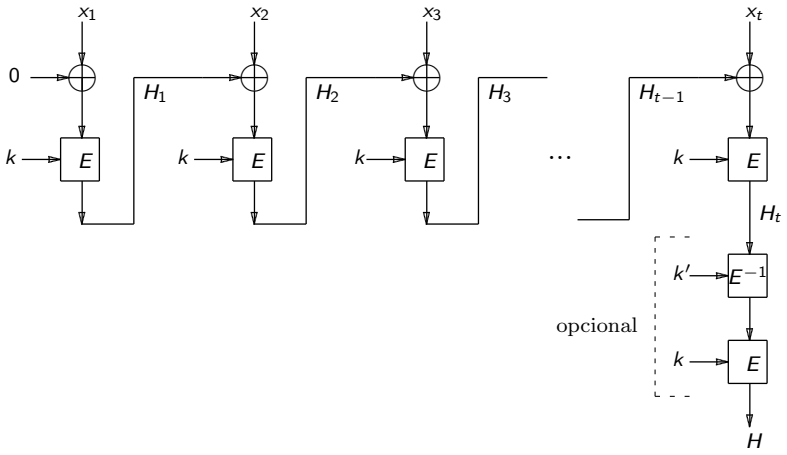
   Este é um CBC (chiper-block-chaining) normal com $VI = 0$, e descartando os blocos cifrados $C_i = H_i$.
3. (eventual) Para aumentar a segurança do MAC através de uma segunda chave secreta $k' \neq k$. Calcular:

$$H'_t \leftarrow E^{-1}_{k'}(H_t) \qquad H_t \leftarrow E_k(H'_t)$$

   Isto não é mais do que um passo de uma tripla-encriptação com duas chaves diferentes.

Algoritmo CBC-MAC (cont.)

4. O resultado final é o bloco de n bits H_t.



CBC-MAC notas

Nota (CBC-MAC fortalecimento)

A parte opcional reduz as ameaças relacionadas com uma procura exaustiva no espaço das chaves, e previne a falsificação existencial de texto escolhido, sem que haja uma perda significativa na eficiência. A opção por um esquema triplo-CBC em toda a sequência já teria um maior impacto na eficiência.
Outras alternativas para combater a falsificação incluem:

- *incluir à cabeça um bloco comprimento antes do cálculo do valor MAC;*
- *utilizar a chave k para encriptar o comprimento m, obtendo $k' = E_k(m)$, usando a seguir k' como chave para o MAC.*

Nota (Vector de Inicialização no CBC-MAC)

Enquanto a utilização de um vector de inicialização aleatório VI, é importante na encriptação, como forma de evitar um ataque "livro de códigos", ao primeiro bloco, isto não é importante na algoritmo MAC.

Construção de MACs a partir de MDCs

Uma possibilidade para a construção de um algoritmo MAC é o de o fazer a partir de um MDC com a simples inclusão de uma chave secreta k.

É de notar no entanto que as necessidades de segurança não são as mesmas.

Vejamos alguns exemplos em que tal construção não é isenta de perigos.

Método do Prefixo Secreto

Considere-se a mensagem $x = x_1 x_2 \ldots x_t$ e um MDC com função de compressão f, tal que $H_0 = VI$, $H_i = f(H_{i-1}, x_i)$, $h(x) = H_t$.

- Suponha-se que se constrói um MAC prefixando a mensagem com uma chave secreta k de tal forma que o algoritmo MAC não é mais do que o cálculo de $h(k\|x)$.

  Então, extendendo a mensagem x com um bloco arbitrário y, pode-se deduzir $M = h(k\|x\|y)$ como sendo $f(M, y)$ sem saber a chave secreta k.
- Por razões similares não é seguro construir um algoritmo MAC a partir de um MDC, utilizando a chave k como vector de inicialização.

  Se k ocupa todo o primeiro bloco, então pode-se pre-calcular $f(VI, k)$, o que ilustra que ao adversário basta achar um k' (não necessariamente k) tal que $f(VI, k) = f(VI, k')$; isto é equivalente a usar um VI secreto.

Método do Sufixo Secreto

Um método alternativo é o de usar a chave secreta como um sufixo à mensagem, temos então que um MAC de n bits de x é igual a $M = h(x\|k)$.
Neste caso, pode-se aplicar um ataque "dia do aniversário".
Um adversário que possa escolher a mensagem x (ou um prefixo dela), pode em $O(2^{n/2})$ operações, achar um par de mensagens x e x' para as quais $h(x) = h(x')$, o que pode ser feito sem conhecimento da chave k.
O que este método faz basicamente é uma aplicação da função de dispersão seguida de uma encriptação do valor final; desta forma (fraca) o valor MAC depende unicamente do último valor da cadeia, e a chave é usada num só passo.

Os exemplos anteriores sugerem que para a construção de um MAC a partir de um MDC se tem de "envolver" a mensagem.

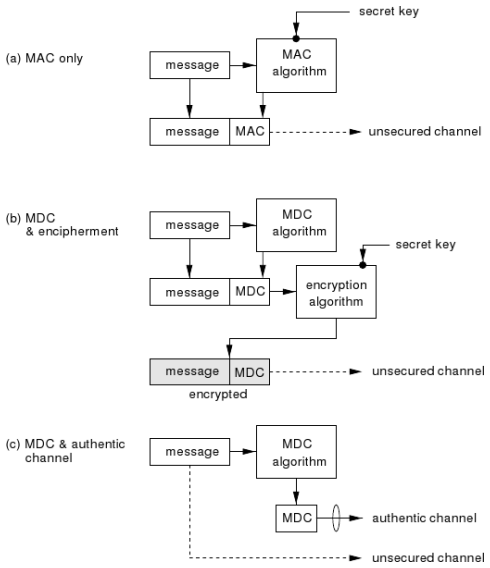
Método do Preenchimento Envolvente

Para uma chave k e um MDC h calcula-se o valor MAC da mensagem x como sendo: $h_k(x) = h(k\|p\|x\|k)$, com p uma sequência de preenchimento de k para o tamanho de um bloco, como forma de garantir que o cálculo interno envolve pelo menos duas iterações.

Por exemplo se h é o MD5 e k tem um comprimento de 128 bits, então p é uma sequência de preenchimento com 384 bits.

Este método é superior aos dois anteriores, no que diz respeito à construção de um MAC a partir de um MDC.

Autenticação de Mensagens



Ataques às Funções de Dispersão

Ataques genéricos são aqueles que podem ser aplicados a uma qualquer função de dispersão, tratando-a como uma caixa fechada cujas únicas características conhecidas sejam:

- o comprimento do bloco de saída n;
- (para os MACs) o comprimento da chave;
- o tempo gasto por operação pela função de dispersão.

Tipicamente assume-se que o valor de dispersão aproxima o valor de uma variável aleatória uniformemente distribuída.
Os ataques com estas características são:

- os baseados no comprimento (em bits) da saída;
- procura exaustiva no espaço das chaves;
- ataque dia do aniversário.

Ataques no Comprimento de um MDC

Dado uma mensagem x com um valor de dispersão, $h(x)$, de n bits, pode-se achar um valor de colisão por um "método da força bruta".

Método da Força Bruta para MDCs

Gerar uma sequência de bits x' (com um comprimento em bits limitado), e verificar se $h(x') = h(x)$.

O custo de uma tentativa é negligenciável, em termos de tempo: uma aplicação da função de dispersão; em termos de espaço: não significativo.

Assumindo-se que o valor de dispersão aproxima o valor de uma variável aleatória uniformemente distribuída, então a probabilidade de encontrar uma tal colisão é de 2^{-n}.

Uma função de dispersão deve ser desenhada tendo como objectivo de segurança o facto de este ser o melhor ataque possível.

Ataques Básicos para Funções de Dispersão

Facto

Ataques Básicos Para uma função de dispersão, $h(x)$, com n bits, é possível, por palpite, achar uma pre-imagem, ou uma segunda pre-imagem, após 2^n aplicações da função de dispersão.
Para um adversário que seja capaz de escolher mensagens um ataque dia do aniversário permite achar um par de colisão x, x', com $h(x) = h(x')$ após $2^{n/2}$ operações e com custos de memória negligenciáveis.

Ataque Dia do Aniversário

Algortimo Yuval, Ataque Dia-de-Aniversário

Entrada: mensagem legítima x_1; mensagem fraudulenta x_2; função de dispersão, h, de um só sentido de m bits.
Saída: x_1', x_2', os quais resultam de pequenas modificações de x_1 e x_2, com $h(x_1') = h(x_2')$.

1. Gerar $t = 2^{m/2}$ pequenas modificações de x_1' de x_1.
2. Calcular o valor de dispersão para cada uma das mensagens modificadas guardando esse valor, de tal forma que possam ser pesquisadas posteriormente.
3. Gerar pequenas modificações x_2' de x_2, calculando $h(x_2')$ e, para cada um desses valores, verificar se há uma correspondência com algum dos x_1' acima definidos. Continuar até que se encontre uma tal correspondência, a qual deve ocorrer após aproximadamente t candidatos x_2'.

Segurança Ideal

Definição

Segurança Ideal Uma função de dispersão sem chave de n bits tem segurança ideal se, para um dado valor de dispersão, ambas as condições seguintes se verificarem:

1. *a produção de uma pré-imagem, ou de uma segunda pré-imagem, requer aproximadamente 2^n operações;*
2. *a produção de colisões requerer aproximadamente $2^{n/2}$ operações.*

Ataques no Espaço das Chaves de um MAC

Ataque por procura exaustiva.

Com um único par (texto claro, MAC), um atacante pode calcular o valor de dispersão de n bits, para todas as chaves possíveis, comparando-o com o MAC conhecido.

Para uma chave de comprimento t bits, isto requer 2^t aplicações da função de dispersão (operações MAC),

Se se assumir que o MAC se comporta como uma função aleatória, pode-se mostrar que é espectável obter a chave única, testando as chaves candidatas usando para tal pouco mais de t/n pares (texto calor,MAC).

Idealmente uma chave MAC não será recuperável em menos de 2^t operações.

Um ataque probabilístico (tentativa à sorte) no espaço das chaves de um MAC, para uma chave de comprimento t bits e uma entrada fixa, uma tentativa aleatória tem um probabilidade de $\approx 2^{-t}$ com $t < n$, de encontrar o valor MAC respectivo.

Ataques no comprimento de um MAC

A falsificação de de MAC envolve a produção de, para um qualquer valor de entrada x, o correspondente valor MAC, sem conhecimento da chave.

Para uma função de dispersão com chave (MAC) com n bits, o adivinhar de um valor de dispersão, para um dado texto, ou uma pre-imagem para um dado valor de dispersão, tem probabilidade de sucesso de $\approx 2^{-n}$, como para os MDCs.
No entanto, para os MACs, a diferença é que os valores obtidos não podem ser verificados, a não ser que se tenham obtidos pares (texto claro, MAC) previamente, ou que de algo forma os mesmos possam ser obtidos.

Dado que a obtenção da chave permite a falsificação de uma forma trivial, esse tipo de ataque é também possível quando se trata de falsificar um MAC.

Idealmente, um adversário não será capaz de produzir novos (e correctos) pares (texto claro,MAC), (x, y) com uma probabilidade significativamente melhor do que $\max(2^{-t}, 2^{-n})$, isto é, o melhor dos dois tipos de ataque, o adivinhar a chave, ou o adivinhar do valor de dispersão.