DCC–IME–USP

Routo Terada
Sala 204C Tel.: 3091 5918
e-mail rt@ime.usp.br

MAC0336/5723 - Criptografia e Segurança de Dados

Primeiro Semestre de 2023

- Prazo de entrega: veja no *e-disciplinas*
- Resolver *individualmente*. Duas soluções **idênticas** receberão **nota zero**
- Entregue no sistema UM ÚNICO arquivo contendo os arquivos seguintes, comprimidos em um único arquivo:
  - um arquivo chamado LEIA.ME (em formato .txt) com:
    - seu nome completo, e número USP,
    - os nomes dos arquivos inclusos com uma breve descrição de cada arquivo,
    - uma descrição sucinta de *como usar* o programa executável, necessariamente na linha-de-comando, i.e., SEM interface gráfica,
    - qual computador, compilador e sistema operacional foi usado (modelo, versão, etc..),
    - instruções de como compilar o(s) arquivo(s) fonte(s).
  - o arquivo MAKE, se for o caso.
  - os arquivos do programa-fonte, necessariamente em **linguagem C**
  - o programa *compilado*, i.e.,

    incluir o código executável
    (se não incluir, a nota será zero!)

  - se for o caso, alguns arquivos de entrada e saída usados nos testes: arquivos com os dados de *entrada* chamados ENT1, ENT2, etc., e arquivos com os dados de *saída* correspondentes, chamados SAI1, SAI2, etc.
- Coloque comentários em seu programa explicando o que cada etapa do programa significa! Isso será levado em conta na sua nota.

- Faça uma saída clara! Isso será levado em conta na sua nota.
- Não deixe para a última hora. Planeje investir 70 por cento do tempo total de dedicação em escrever o seu programa todo e simular o programa SEM computador (eliminando erros de lógica) ANTES de digitar e compilar no computador. Isso economiza muito tempo e energia.
- A nota será diminuida de um ponto a cada dia "corrido" de atraso na entrega.
- Os alunos "ouvintes"*não* devem entregar uma solução.

Este exercício-programa consiste em:

1. Elaborar um programa para criptografar e decriptografar um arquivo de qualquer comprimento, com o Algoritmo K128 descrito na Seção 1. Esse arquivo pode ser QUALQUER sequência de bits, qualquer tipo digital: um texto, uma imagem, uma música, programa-fonte, etc.
2. O programa deve também medir a aleatoridade do algoritmo, conforme descrito a partir da página 8.

A chave principal K de 128 bits é derivada de uma senha, como descrito a seguir (Seção 2, página 3).

3. No final deste enunciado há Quatro S-boxes que devem ser utilizados na solução deste EP.

Notação:

- $A||B$ denota concatenação de A e B.
- $|A|$ denota comprimento de A.
- $\oplus$ denota ou-exclusivo.
- $<<$ denota rotação circular.
- $\boxminus$ denota subtração $\bmod 2^{32}$.
- $\boxplus$ denota soma $\bmod 2^{32}$.

1 Definição do Algoritmo K128

Implementar o Algoritmo criptográfico K128, com chave de 128 bits, e com entrada e saída de 128 bits. Você deve **deduzir** o algoritmo inverso do K128.

O número N de iterações (*rounds*) é variável, mas $N = 12$ neste projeto. As subchaves $KR5(i,.), KM32(i,.)$ são definidas na Seção 4 na página 4. $UmaIteracao$ é definida na Seção 7 na página 4.

Algoritmo K128 de N iterações

Entrada: X de 128 bits é um bloco; subchaves $KR5(i,.), KM32(i,.)$. Observ.: X pode ser **qualquer** sequência de bits.

Saída: Y de 128 bits de bloco criptografado é a saída

1. **para** $i = 0, 1, 2...N-1$ **faça sequencialmente**{
2. $\quad X \leftarrow UmaIteracao(i, X, KR5(i,.), KM32(i,.))$ }
3. $Y \leftarrow X$

2 Senha e chave principal K

A senha A a ser digitada deve conter pelo menos **8** caracteres (1 byte de 8 bits por caractere em código ASCII), com **pelo menos** 2 letras e 2 algarismos decimais. Por exemplo:

byte	caractere
00111111	?
01000000	@
01000001	A
01000010	B
01000011	C
01100001	a
01100010	b
01100011	c
01111110	~

Geração da chave K de 128 bits a partir da senha: se a senha A digitada possuir menos que 16 caracteres (i.e., 16 bytes), a chave principal K de 128 bits deve ser derivada de A concatenando-se A com ela própria até completar 16 bytes (128 bits).

3 Chaves intermediárias K_i

Para $j = 1, 2, 3$, as funções $f_j()$, usadas nesta Seção, estão definidas na Seção 6 na página 5. As **chaves intermediárias** K_i para a iteração $i = 0, 1, 2, ...$ são geradas a partir da chave principal K, e são utilizadas para gerar as subchaves $KR5(i, .)$ e $KM32(i, .)$, definidas na Seção 4, página 4. O seu programa pode gerar cada K_i só no momento da execução da i-ésima iteração, para economizar memória.

Algoritmo de geração de chave intermediária K_i para iteração $i = 0, 1, 2, ..11$.

Entrada: 128 bits $K_i = X||Y||Z||W$, onde $|X| = |Y| = |Z| = |W| = 32$ bits, e constantes $ConstR = (10011)_2$ de 5 bits, e $ConstM = (CB3725F7)_{16}$ de 32 bits.

Observação: $K_0 = K \oplus (5A827999874AA67D657B7C8EBD070242)_{16}$, onde K é a chave principal de 128 bits do Algoritmo K128 - ver Seção 2 na página 3.

Saída: $K_{i+1} = W||Z||Y||X$ de 128 bits, onde $|X| = |Y| = |Z| = |W| = 32$ bits.

1. **para** $i = 0, 1, 2, ...11$ **faça sequencialmente**{
2. $W := W \oplus f_2(X, ConstR << [(i+2)^2] \bmod 3), ConstM << [(i+3)^2] \bmod 7));$
3. $Z := Z \oplus f_1(W, ConstR << (i+2) \bmod 3), ConstM << (i+3) \bmod 7));$
4. $Y := Y \oplus f_3(Z, ConstR << [(i+2)^3] \bmod 3), ConstM << [(i+3)^3] \bmod 7));$
5. $X := X \oplus f_2(Y, ConstR << [(i+2)^2] \bmod 3), ConstM << [(i+3)^2] \bmod 7));$
6. }

4 Geração de subchaves $KR5, KM32$

A seguir o algoritmo para gerar as subchaves $KR5(i, j)$ e $KM32(i, j)$, para iteração $i = 0, 1, 2, ...$ e $j = 0, 1, 2, 3$, a partir das subchaves K_i para $i = 0, 1, 2, ...$ O seu programa pode gerar cada $KR5(i, j)$ e $KM32(i, j)$ só no momento da execução da i-ésima iteração.

Algoritmo de geração de subchaves $KR5(i, j)$ e $KM32(i, j)$ para iteração $i = 0, 1, 2, ..11$, e $j = 0, 1, 2, 3$.

Entrada: 128 bits de chave intermediária $K_i = X||Y||Z||W$, onde $|X| = |Y| = |Z| = |W| = 32$ bits
Saída: para $j = 0, 1, 2, 3$, $KR5(i, j)$ de 5 bits e $KM32(i, j)$ de 32 bits.

1. **para** $i = 0, 1, 2, ...11$ **faça sequencialmente**{
2. $KR5(i, 0) := 5BitsDaDireita(X); KR5(i, 1) := 5BitsDaDireita(Y);$
3. $KR5(i, 2) := 5BitsDaDireita(Z); KR5(i, 3) := 5BitsDaDireita(W);$
4. $KM32(i, 0) := W; KM32(i, 1) := Z; KM32(i, 2) := Y; KM32(i, 3) = X;$
5. }

5 S-boxes

As quatro S-boxes $S_j : \{0,1\}^8 \rightarrow \{0,1\}^{32}$ são funções não-lineares, definidas e pré-calculadas em forma de tabela. As S-boxes estão no Anexo, página 11. Elas são usadas nas definições na Seção 6.

6 Funções f_1, f_2, f_3

As funções f_1, f_2, f_3 , definidas a seguir são usadas na Seção 7, na página 6. Para $j = 1, 2, 3, 4$, $S_j : \{0,1\}^8 \rightarrow \{0,1\}^{32}$ são as S-boxes definidas na Seção 5 na página 5.
Definição de $f_1 : \{0,1\}^{32} \times \{0,1\}^5 \times \{0,1\}^{32} \rightarrow \{0,1\}^{32}$
Para X de 32 bits, $K5$ de 5 bits, e $K32$ de 32 bits, $f_1(X, K5, K32) = Y$ de 32 bits é da seguinte forma:

$$\begin{aligned} \text{(1) calcular } I &: = [(K32 \oplus X) \bmod 2^{32} << K5], \\ \text{(2) dividir } I &= I_1||I_2||I_3||I_4, \text{ cada } I_t \text{ de 8 bits} \\ \text{(3) calcular } Y &: = [\ (S_1(I_1) \boxplus S_2(I_2))\ \boxminus\ S_3(I_3)\ ] \oplus S_4(I_4) \end{aligned}$$

Definição de $f_2 : \{0,1\}^{32} \times \{0,1\}^5 \times \{0,1\}^{32} \rightarrow \{0,1\}^{32}$

Para X de 32 bits, $K5$ de 5 bits, e $K32$ de 32 bits, $f_2(X, K5, K32) = Y$ de 32 bits é da seguinte forma:

$$\begin{aligned} \text{(1) calcular } I &: = [(K32 \oplus X) \bmod 2^{32} << K5], \\ \text{(2) dividir } I &= I_1||I_2||I_3||I_4, \text{ cada } I_t \text{ de 8 bits} \\ \text{(3) calcular } Y &: = [\ (S_1(I_1) \boxminus S_2(I_2))\ \oplus\ S_3(I_3)\ ] \boxplus S_4(I_4) \end{aligned}$$

Definição de $f_3 : \{0,1\}^{32} \times \{0,1\}^5 \times \{0,1\}^{32} \rightarrow \{0,1\}^{32}$
Para X de 32 bits, $K5$ de 5 bits, e $K32$ de 32 bits, $f_3(X, K5, K32) = Y$ de 32 bits é da seguinte forma:

$$\begin{aligned} \text{(1) calcular } I &: = [(K32 \oplus X) \bmod 2^{32} << K5], \\ \text{(2) dividir } I &= I_1||I_2||I_3||I_4, \text{ cada } I_t \text{ de 8 bits} \\ \text{(3) calcular } Y &: = [\ (S_1(I_1) \oplus S_2(I_2))\ \boxplus\ S_3(I_3)\ ] \boxminus S_4(I_4) \end{aligned}$$

7 Definição de $UmaIteracao(i, X, KR5(i, .), KM32(i, .))$

A seguir, a definição de $UmaIteracao(i, X, KR5(i, .), KM32(i, .))$, na qual são usadas as funções f_1, f_2, f_3 , definidas na Seção 6 na página 5, e $KR5(i, .), KM32(i, .)$, definidas na Seção 4 na página 4.
Algoritmo $UmaIteracao : \{0, 1, 2, ...11\} \times \{0,1\}^{128} \times \{0,1\}^5 \times \{0,1\}^{32} \rightarrow \{0,1\}^{128}$
Entrada $X = A||B||C||D$ de 128 bits, sendo $|A| = |B| = |C| = |D| = 32$ bits; $KR5(i, .), KM32(i, .)$
Saída $C||B||A||D$ de 128 bits

1. $C := C \oplus f_2(D, KR5(i, 0), KM32(i, 0))$;
2. $B := B \oplus f_1(C, KR5(i, 1), KM32(i, 1))$;
3. $A := A \oplus f_3(B, KR5(i, 2), KM32(i, 2))$;
4. $D := D \oplus f_2(A, KR5(i, 3), KM32(i, 3))$;

8 Execução na linha de comando

O seu programa deve ser executado na linha de comando, com parâmetros relevantes, em um dos seguintes modos: (se houver a opção -a após a senha, o programa deve gravar **brancos** no lugar do arquivo de entrada e deletá-lo, o *default* é não efetuar o apagamento)

- Modo (1) Para criptografar arquivos:
  programa -c -i <arquivo de entrada> -o <arquivo de saída> -p <senha> -a
- Modo (2) Para decriptografar arquivos:
  programa -d -i <arquivo de entrada> -o <arquivo de saída> -p <senha>
- Modo (3) Para calcular aleatoriedade pelo método descrito abaixo, Seção 11, na página 8:
  programa -1 -i <arquivo de entrada> -p <senha>

9 O que o programa deve fazer

O seu programa deve ler do disco o arquivo de entrada Entra, que pode ser QUALQUER sequência de bits (música, figura, texto, etc.), e deve gravar o arquivo de saída Sai correspondente a Entra criptografado ou decriptografado com a senha A, no modo CBC (Cipher Block Chaining, veja Seção 10), que consiste em encadear um bloco de 128 bits com o bloco anterior criptografado da maneira vista em aula, e também descrito no livro-texto Segurança de Dados (RT).

O seu programa deve também efetuar o item descrito no final deste enunciado, Seção 11, na página 8.

10 Modo CBC e testes

As regras a seguir devem ser satisfeitas:

1. No modo CBC, utilizar bits iguais a UM como Valor Inicial.
2. V. deve testar o programa com pelo menos dois arquivos Entra. Por exemplo, o seu próprio programa-fonte. Teste não só com arquivos-texto como com arquivos binários; por exemplo, com algum código executável, ou MP3, ou JPEG, etc..
3. Se o último bloco a ser criptografado do arquivo Entra não possuir comprimento igual a 128 bits, completá-lo com bits iguais a UM.
4. Acrescentar sempre um bloco extra, criptografado, no arquivo Sai com o comprimento do arquivo original Entra.
5. Verifique se o arquivo **decriptografado** Sai possui o mesmo comprimento que o arquivo original Entra.

11 Medidas de aleatoridade - entropia

Seja $VetEntra$ um vetor lido de um arquivo de entrada para a memória principal com pelo menos 512 bits (i.e., pelo menos 4 blocos de 128 bits, de modo que

$$VetEntra = Bl(1)||Bl(2)||Bl(3)||Bl(4)||...,$$

sendo cada bloco $Bl()$ de 128 bits e $|VetEntra| \geq 4 * 128 = 512$).

Para $j = 1, 2, ..., |VetEntra|$ fazer o seguinte:

1. alterar apenas na memória interna (RAM) *SÓ* o j-ésimo bit do vetor $VetEntra$ de cada vez, obtendo um **outro vetor** na memória interna, chamado $VetAlterj$, para $j = 1, 2, 3, ...$ tal que $|VetEntra| = |VetAlterj|$; isto é, $VetEntra$ e $VetAlterj$ *SÓ* diferem no j-ésimo bit, mas são de igual comprimento. No caso de apenas 4 blocos, tem-se $j = 1, 2, 3, ...512$. Por exemplo, no caso de 8 bits em cada bloco, um único bit alterado na posição $j = 2$, tem-se $Bl(1) = 01110101$, $BlAlter(1) = 00110101, ...$ e

   $$VetEntraj = BlAlter(1)||BlAlter(2)||... = 01110101||...$$

   $$VetAlterj = BlAlter(1)||BlAlter(2)||... = 00110101||...$$

   ou seja, os blocos diferem só no bit na posição 2, e então $Hamming(VetEntra, VetAlterj) = 1$.

2. seja $VetEntraC = BlC(1)||BlC(2)||BlC(3)||BlC(4)||...$ o vetor $VetEntra$ criptografado pelo K128 no modo CBC. C de Criptografado.

3. seja $VetAlterjC = BlAlterjC(1)||BlAlterjC(2)||BlAlterjC(3)||BlAlterjC(4)||...$o vetor $VetAlterj$ criptografado pelo K128 no modo CBC.

4. para $k = 1, 2, 3, ...$, medir a distância de Hamming, **separadamente**, entre **cada** bloco $BlC(k)$ e $BlAlterjC(k)$, sendo ambos de 128 bits. ($BlC(k)$ do vetor $VetEntraC$ e o correspondente bloco $BlAlterjC(k)$ do vetor $VetAlterjC$). Para 4 blocos de 128 bits, tem-se 4 medidas de distância, para cada $j = 1, 2, 3, ...$, sendo cada medida chamada, digamos, $H(k)$, para cada par de blocos $BlC(k), BlAlterjC(k)$. Ou seja, para 4 blocos tem-se $k = 1, 2, 3, 4$, e para cada $j = 1, 2, 3, ...$, $H(k)$ =Hamming$(BlC(k), BlAlterjC(k))$. Veja ilustração na página 9.

5. estas medidas de distância de Hamming $H(k)$ devem ser acumuladas em somas chamadas, digamos, $SomaH(k)$. Para 4 blocos de 128 bits, tem-se 4 somas cumulativas, sendo que:

   (a) $SomaH(1)$ acumula 128 valores de $H(1)$ correspondentes a $j = \underbrace{1, 2, 3..., 128}_{\text{bloco 1}}$ (para $j > 128$ $H(1) = 0$ pois $BlC(1) = BlAlterjC(1)$). Veja ilustração na página 10.

(b) $SomaH(2)$ acumula $2*128 = 256$ valores de $H(2)$ correspondentes a $j = \underbrace{1, 2, 3..., 128}_{\text{bloco 1}}, \underbrace{129..., 256}_{\text{bloco 2}}$ (para $j > 2*128$ $H(2) = 0$ pois $BlC(2) = BlAlterjC(2)$ e $H(1) = 0$ pois $BlC(1) = BlAlterjC(1)$). Veja ilustração na página 10.

(c) $SomaH(3)$ acumula $3 * 128 = 384$ valores de $H(3)$ correspondentes a $j = 1, 2, 3...., 384$

(d) $SomaH(4)$, acumula $4 * 128 = 512$ valores de $H(4)$ correspondentes a $j = 1, 2, 3..., 512$.

6. de forma análoga às somas $SomaH(k)$, o programa deve calcular os valores mínimo e máximo de $H(1), H(2), ...$

$VetEntra =$	$Bl(1)$	$Bl(2)$	$Bl(3)$	$Bl(4)$
Aplica algoritmo em CBC	K128(K)	K128(K)	K128(K)	K128(K)
$VetEntraC$ (criptografado)=	$BlC(1) = A$	$BlC(2) = B$	$BlC(3) = C$	$BlC(4) = D$
$VetAlterj =$	$BlAlterj(1)$	$BlAlterj(2)$	$BlAlterj(3)$	$BlAlterj(4)$
Aplica algoritmo em CBC	K128(K)	K128(K)	K128(K)	K128(K)
$VetAlterjC$ (criptografado)=	$BlAlterjC(1) = A'$	$BlAlterjC(2) = B'$	$BlAlterjC(3) = C'$	$BlAlterjC(4) = D'$
Posição do único bit alterado	$j = 1, 2, ...128$	$j = 129, ...256$	$j = 257, ...384$	$j = 385, ...512$
Distância de Hamming	$H(1) = Ham(A, A')$	$H(2) = Ham(B, B')$	$H(3)Ham(C, C')$	$H(4)Ham(D, D')$
Núm.de valores de $H(k)$	128	256	384	512
Soma acumulada de $H(k)$	$SomaH(1)$	$SomaH(2)$	$SomaH(3)$	$SomaH(4)$
Máximo e Mínimo de $H(k)$	$Max(1), Min(1)$	$Max(2), Min(2)$	$Max(3), Min(3)$	$Max(4), Min(4)$

Tabela geral de valores para as medidas de aleatoridade do algoritmo K128

$VetEntra =$	$Bl(1)$	$Bl(2)$	$Bl(3)$	$Bl(4)$
Aplica algoritmo em CBC	K128(K)	K128(K)	K128(K)	K128(K)
$VetEntraC$ (criptografado)=	$BlC(1) = A$	$BlC(2) = B$	$BlC(3) = C$	$BlC(4) = D$
$VetAlter =$	$BlAlter(1) = Bl(1)$	$BlAlter(2)$	$BlAlter(3)$	$BlAlter(4)$
Aplica algoritmo em CBC	K128(K)	K128(K)	K128(K)	K128(K)
$VetAlterjC$ (criptografado)=	$BlAlterjC(1) = A' = A$	$BlAlterjC(2) = B'$	$BlAlterjC(3) = C'$	$BlAlterjC(4) = D'$
Posição do único bit alterado	$j = 1, 2, ...128$	$j = 129, ...256$	$j = 257, ...384$	$j = 385, ...512$
Distância de Hamming	$H(1) = Ham(A, A') = 0$	$H(2) = Ham(B, B')$	$H(3) = Ham(C, C')$	$H(4) = Ham(D, D')$
Núm.de valores de $H(k)$	128	256	384	512
Soma acumulada de $H(k)$	$SomaH(1)$	$SomaH(2)$	$SomaH(3)$	$SomaH(4)$
Máximo e Mínimo de $H(k)$	$Max(1), Min(1)$	$Max(2), Min(2)$	$Max(3), Min(3)$	$Max(4), Min(4)$

Tabela, para $j = 129, ...256$, de valores para as medidas de aleatoridade do algoritmo K128

$VetEntra =$	$Bl(1)$	$Bl(2)$	$Bl(3)$	$Bl(4)$
Aplica algoritmo em CBC	K128(K)	K128(K)	K128(K)	K128(K)
$VetEntraC$ (criptografado)=	$BlC(1) = A$	$BlC(2) = B$	$BlC(3) = C$	$BlC(4) = D$
$VetAlterj =$	$BlAlterj(1) = A'$	$BlAlterj(2) = B'$	$BlAlterj(3)$	$BlAlterj(4)$
Aplica algoritmo em CBC	K128(K)	K128(K)	K128(K)	K128(K)
$VetAlterjC$ (criptografado)=	$BlAlterjC(1) = A' = A$	$BlAlterjC(2) = B' = B$	$BlAlterjC(3) = C'$	$BlAlterjC(4) = D'$
Posição do único bit alterado	nenhum bit alterado	nenhum bit alterado	$j = 257, ...384$	
Distância de Hamming	$H(1) = Ham(A, A') = 0$	$H(2) = Ham(B, B') = 0$	$H(3) = Ham(C, C')$	$H(4) = Ham(D, D')$
Núm.de valores de $H(k)$	128	256	384	512
Soma acumulada de $H(k)$	$SomaH(1)$	$SomaH(2)$	$SomaH(3)$	$SomaH(4)$
Máximo e Mínimo de $H(k)$	$Max(1), Min(1)$	$Max(2), Min(2)$	$Max(3), Min(3)$	$Max(4), Min(4)$

Tabela, para $j = 257, ...384$, de valores para as medidas de aleatoridade do algoritmo K128

No final o programa deve imprimir uma tabela contendo os valores **máximos, mínimos e médios** das distâncias de Hamming entre **cada** bloco criptografado de 128 bits $BlC(k)$ e $BlAlterjC(k)$, conforme o Algoritmo K128, no modo CBC. Para 4 blocos de 128 bits, o programa deve imprimir 4 valores máximos, 4 mínimos, e 4 médios.

Em resumo: o seu programa deve também medir a aleatoriedade (entropia) do K128 na forma descrita acima.

Anexo: Quatro S-boxes

A seguir as quatro S-Boxes não-lineares, com entrada de 8 bits (i.e., cada S-Box contém $2^8 = 256$ elementos, 8 colunas e 32 linhas) e saída de 32 bits (em notação hexadecimal). A tabelas devem ser lidas da esquerda para a direita, de cima para baixo.

S-Box S1

```
30fb40d4 9fa0ff0b 6beccd2f 3f258c7a 1e213f2f 9c004dd3 6003e540 cf9fc949
bfd4af27 88bbbdb5 e2034090 98d09675 6e63a0e0 15c361d2 c2e7661d 22d4ff8e
28683b6f c07fd059 ff2379c8 775f50e2 43c340d3 df2f8656 887ca41a a2d2bd2d
a1c9e0d6 346c4819 61b76d87 22540f2f 2abe32e1 aa54166b 22568e3a a2d341d0
66db40c8 a784392f 004dff2f 2db9d2de 97943fac 4a97c1d8 527644b7 b5f437a7
b82cbaef d751d159 6ff7f0ed 5a097a1f 827b68d0 90ecf52e 22b0c054 bc8e5935
4b6d2f7f 50bb64a2 d2664910 bee5812d b7332290 e93b159f b48ee411 4bff345d
fd45c240 ad31973f c4f6d02e 55fc8165 d5b1caad a1ac2dae a2d4b76d c19b0c50
882240f2 0c6e4f38 a4e4bfd7 4f5ba272 564c1d2f c59c5319 b949e354 b04669fe
b1b6ab8a c71358dd 6385c545 110f935d 57538ad5 6a390493 e63d37e0 2a54f6b3
3a787d5f 6276a0b5 19a6fcdf 7a42206a 29f9d4d5 f61b1891 bb72275e aa508167
38901091 c6b505eb 84c7cb8c 2ad75a0f 874a1427 a2d1936b 2ad286af aa56d291
d7894360 425c750d 93b39e26 187184c9 6c00b32d 73e2bb14 a0bebc3c 54623779
64459eab 3f328b82 7718cf82 59a2cea6 04ee002e 89fe78e6 3fab0950 325ff6c2
81383f05 6963c5c8 76cb5ad6 d49974c9 ca180dcf 380782d5 c7fa5cf6 8ac31511
35e79e13 47da91d0 f40f9086 a7e2419e 31366241 051ef495 aa573b04 4a805d8d
548300d0 00322a3c bf64cddf ba57a68e 75c6372b 50afd341 a7c13275 915a0bf5
6b54bfab 2b0b1426 ab4cc9d7 449ccd82 f7fbf265 ab85c5f3 1b55db94 aad4e324
cfa4bd3f 2deaa3e2 9e204d02 c8bd25ac eadf55b3 d5bd9e98 e31231b2 2ad5ad6c
954329de adbe4528 d8710f69 aa51c90f aa786bf6 22513f1e aa51a79b 2ad344cc
7b5a41f0 d37cfbad 1b069505 41ece491 b4c332e6 032268d4 c9600acc ce387e6d
bf6bb16c 6a70fb78 0d03d9c9 d4df39de e01063da 4736f464 5ad328d8 b347cc96
75bb0fc3 98511bfb 4ffbcc35 b58bcf6a e11f0abc bfc5fe4a a70aec10 ac39570a
3f04442f 6188b153 e0397a2e 5727cb79 9ceb418f 1cacd68d 2ad37c96 0175cb9d
c69dff09 c75b65f0 d9db40d8 ec0e7779 4744ead4 b11c3274 dd24cb9e 7e1c54bd
f01144f9 d2240eb1 9675b3fd a3ac3755 d47c27af 51c85f4d 56907596 a5bb15e6
580304f0 ca042cf1 011a37ea 8dbfaadb 35ba3e4a 3526ffa0 c37b4d09 bc306ed9
98a52666 5648f725 ff5e569d 0ced63d0 7c63b2cf 700b45e1 d5ea50f1 85a92872
af1fbda7 d4234870 a7870bf3 2d3b4d79 42e04198 0cd0ede7 26470db8 f881814c
474d6ad7 7c0c5e5c d1231959 381b7298 f5d2f4db ab838653 6e2f1e23 83719c9e
bd91e046 9a56456e dc39200c 20c8c571 962bda1c e1e696ff b141ab08 7cca89b9
1a69e783 02cc4843 a2f7c579 429ef47d 427b169c 5ac9f049 dd8f0f00 5c8165bf
```

S-Box S2

```
1f201094 ef0ba75b 69e3cf7e 393f4380 fe61cf7a eec5207a 55889c94 72fc0651
ada7ef79 4e1d7235 d55a63ce de0436ba 99c430ef 5f0c0794 18dcdb7d a1d6eff3
a0b52f7b 59e83605 ee15b094 e9ffd909 dc440086 ef944459 ba83ccb3 e0c3cdfb
d1da4181 3b092ab1 f997f1c1 a5e6cf7b 01420ddb e4e7ef5b 25a1ff41 e180f806
1fc41080 179bee7a d37ac6a9 fe5830a4 98de8b7f 77e83f4e 79929269 24fa9f7b
e113c85b acc40083 d7503525 f7ea615f 62143154 0d554b63 5d681121 c866c359
3d63cf73 cee234c0 d4d87e87 5c672b21 071f6181 39f7627f 361e3084 e4eb573b
602f64a4 d63acd9c 1bbc4635 9e81032d 2701f50c 99847ab4 a0e3df79 ba6cf38c
10843094 2537a95e f46f6ffe a1ff3b1f 208cfb6a 8f458c74 d9e0a227 4ec73a34
fc884f69 3e4de8df ef0e0088 3559648d 8a45388c 1d804366 721d9bfd a58684bb
e8256333 844e8212 128d8098 fed33fb4 ce280ae1 27e19ba5 d5a6c252 e49754bd
c5d655dd eb667064 77840b4d a1b6a801 84db26a9 e0b56714 21f043b7 e5d05860
54f03084 066ff472 a31aa153 dadc4755 b5625dbf 68561be6 83ca6b94 2d6ed23b
eccf01db a6d3d0ba b6803d5c af77a709 33b4a34c 397bc8d6 5ee22b95 5f0e5304
81ed6f61 20e74364 b45e1378 de18639b 881ca122 b96726d1 8049a7e8 22b7da7b
5e552d25 5272d237 79d2951c c60d894c 488cb402 1ba4fe5b a4b09f6b 1ca815cf
a20c3005 8871df63 b9de2fcb 0cc6c9e9 0beeff53 e3214517 b4542835 9f63293c
ee41e729 6e1d2d7c 50045286 1e6685f3 f33401c6 30a22c95 31a70850 60930f13
73f98417 a1269859 ec645c44 52c877a9 cdff33a6 a02b1741 7cbad9a2 2180036f
50d99c08 cb3f4861 c26bd765 64a3f6ab 80342676 25a75e7b e4e6d1fc 20c710e6
cdf0b680 17844d3b 31eef84d 7e0824e4 2ccb49eb 846a3bae 8ff77888 ee5d60f6
7af75673 2fdd5cdb a11631c1 30f66f43 b3faec54 157fd7fa ef8579cc d152de58
db2ffd5e 8f32ce19 306af97a 02f03ef8 99319ad5 c242fa0f a7e3ebb0 c68e4906
b8da230c 80823028 dcdef3c8 d35fb171 088a1bc8 bec0c560 61a3c9e8 bca8f54d
c72feffa 22822e99 82c570b4 d8d94e89 8b1c34bc 301e16e6 273be979 b0ffeaa6
61d9b8c6 00b24869 b7ffce3f 08dc283b 43daf65a f7e19798 7619b72f 8f1c9ba4
dc8637a0 16a7d3b1 9fc393b7 a7136eeb c6bcc63e 1a513742 ef6828bc 520365d6
2d6a77ab 3527ed4b 821fd216 095c6e2e db92f2fb 5eea29cb 145892f5 91584f7f
5483697b 2667a8cc 85196048 8c4bacea 833860d4 0d23e0f9 6c387e8a 0ae6d249
b284600c d835731d dcb1c647 ac4c56ea 3ebd81b3 230eabb0 6438bc87 f0b5b1fa
8f5ea2b3 fc184642 0a036b7a 4fb089bd 649da589 a345415e 5c038323 3e5d3bb9
43d79572 7e6dd07c 06dfdf1e 6c6cc4ef 7160a539 73bfbe70 83877605 4523ecf1
```

S-Box S3

```
8defc240 25fa5d9f eb903dbf e810c907 47607fff 369fe44b 8c1fc644 aececa90
beb1f9bf eefbcaea e8cf1950 51df07ae 920e8806 f0ad0548 e13c8d83 927010d5
11107d9f 07647db9 b2e3e4d4 3d4f285e b9afa820 fade82e0 a067268b 8272792e
553fb2c0 489ae22b d4ef9794 125e3fbc 21fffcee 825b1bfd 9255c5ed 1257a240
4e1a8302 bae07fff 528246e7 8e57140e 3373f7bf 8c9f8188 a6fc4ee8 c982b5a5
a8c01db7 579fc264 67094f31 f2bd3f5f 40fff7c1 1fb78dfc 8e6bd2c1 437be59b
99b03dbf b5dbc64b 638dc0e6 55819d99 a197c81c 4a012d6e c5884a28 ccc36f71
b843c213 6c0743f1 8309893c 0feddd5f 2f7fe850 d7c07f7e 02507fbf 5afb9a04
a747d2d0 1651192e af70bf3e 58c31380 5f98302e 727cc3c4 0a0fb402 0f7fef82
8c96fdad 5d2c2aae 8ee99a49 50da88b8 8427f4a0 1eac5790 796fb449 8252dc15
efbd7d9b a672597d ada840d8 45f54504 fa5d7403 e83ec305 4f91751a 925669c2
23efe941 a903f12e 60270df2 0276e4b6 94fd6574 927985b2 8276dbcb 02778176
f8af918d 4e48f79e 8f616ddf e29d840e 842f7d83 340ce5c8 96bbb682 93b4b148
ef303cab 984faf28 779faf9b 92dc560d 224d1e20 8437aa88 7d29dc96 2756d3dc
8b907cee b51fd240 e7c07ce3 e566b4a1 c3e9615e 3cf8209d 6094d1e3 cd9ca341
5c76460e 00ea983b d4d67881 fd47572c f76cedd9 bda8229c 127dadaa 438a074e
1f97c090 081bdb8a 93a07ebe b938ca15 97b03cff 3dc2c0f8 8d1ab2ec 64380e51
68cc7bfb d90f2788 12490181 5de5ffd4 dd7ef86a 76a2e214 b9a40368 925d958f
4b39fffa ba39aee9 a4ffd30b faf7933b 6d498623 193cbcfa 27627545 825cf47a
61bd8ba0 d11e42d1 cead04f4 127ea392 10428db7 8272a972 9270c4a8 127de50b
285ba1c8 3c62f44f 35c0eaa5 e805d231 428929fb b4fcdf82 4fb66a53 0e7dc15b
1f081fab 108618ae fcfd086d f9ff2889 694bcc11 236a5cae 12deca4d 2c3f8cc5
d2d02dfe f8ef5896 e4cf52da 95155b67 494a488c b9b6a80c 5c8f82bc 89d36b45
3a609437 ec00c9a9 44715253 0a874b49 d773bc40 7c34671c 02717ef6 4feb5536
a2d02fff d2bf60c4 d43f03c0 50b4ef6d 07478cd1 006e1888 a2e53f55 b9e6d4bc
a2048016 97573833 d7207d67 de0f8f3d 72f87b33 abcc4f33 7688c55d 7b00a6b0
947b0001 570075d2 f9bb88f8 8942019e 4264a5ff 856302e0 72dbd92b ee971b69
6ea22fde 5f08ae2b af7a616d e5c98767 cf1febd2 61efc8c2 f1ac2571 cc8239c2
67214cb8 b1e583d1 b7dc3e62 7f10bdce f90a5c38 0ff0443d 606e6dc6 60543a49
5727c148 2be98a1d 8ab41738 20e1be24 af96da0f 68458425 99833be5 600d457d
282f9350 8334b362 d91d1120 2b6d8da0 642b1e31 9c305a00 52bce688 1b03588a
f7baefd5 4142ed9c a4315c11 83323ec5 dfef4636 a133c501 e9d3531c ee353783
```

S-Box S4

```
9db30420 1fb6e9de a7be7bef d273a298 4a4f7bdb 64ad8c57 85510443 fa020ed1
7e287aff e60fb663 095f35a1 79ebf120 fd059d43 6497b7b1 f3641f63 241e4adf
28147f5f 4fa2b8cd c9430040 0cc32220 fdd30b30 c0a5374f 1d2d00d9 24147b15
ee4d111a 0fca5167 71ff904c 2d195ffe 1a05645f 0c13fefe 081b08ca 05170121
80530100 e83e5efe ac9af4f8 7fe72701 d2b8ee5f 06df4261 bb9e9b8a 7293ea25
ce84ffdf f5718801 3dd64b04 a26f263b 7ed48400 547eebe6 446d4ca0 6cf3d6f5
2649abdf aea0c7f5 36338cc1 503f7e93 d3772061 11b638e1 72500e03 f80eb2bb
abe0502e ec8d77de 57971e81 e14f6746 c9335400 6920318f 081dbb99 ffc304a5
4d351805 7f3d5ce3 a6c866c6 5d5bcca9 daec6fea 9f926f91 9f46222f 3991467d
a5bf6d8e 1143c44f 43958302 d0214eeb 022083b8 3fb6180c 18f8931e 281658e6
26486e3e 8bd78a70 7477e4c1 b506e07c f32d0a25 79098b02 e4eabb81 28123b23
69dead38 1574ca16 df871b62 211c40b7 a51a9ef9 0014377b 041e8ac8 09114003
bd59e4d2 e3d156d5 4fe876d5 2f91a340 557be8de 00eae4a7 0ce5c2ec 4db4bba6
e756bdff dd3369ac ec17b035 06572327 99afc8b0 56c8c391 6b65811c 5e146119
6e85cb75 be07c002 c2325577 893ff4ec 5bbfc92d d0ec3b25 b7801ab7 8d6d3b24
20c763ef c366a5fc 9c382880 0ace3205 aac9548a eca1d7c7 041afa32 1d16625a
6701902c 9b757a54 31d477f7 9126b031 36cc6fdb c70b8b46 d9e66a48 56e55a79
026a4ceb 52437eff 2f8f76b4 0df980a5 8674cde3 edda04eb 17a9be04 2c18f4df
b7747f9d ab2af7b4 efc34d20 2e096b7c 1741a254 e5b6a035 213d42f6 2c1c7c26
61c2f50f 6552daf9 d2c231f8 25130f69 d8167fa2 0418f2c8 001a96a6 0d1526ab
63315c21 5e0a72ec 49bafefd 187908d9 8d0dbd86 311170a7 3e9b640c cc3e10d7
d5cad3b6 0caec388 f73001e1 6c728aff 71eae2a1 1f9af36e cfcbd12f c1de8417
ac07be6b cb44a1d8 8b9b0f56 013988c3 b1c52fca b4be31cd d8782806 12a3a4e2
6f7de532 58fd7eb6 d01ee900 24adffc2 f4990fc5 9711aac5 001d7b95 82e5e7d2
109873f6 00613096 c32d9521 ada121ff 29908415 7fbb977f af9eb3db 29c9ed2a
5ce2a465 a730f32c d0aa3fe8 8a5cc091 d49e2ce7 0ce454a9 d60acd86 015f1919
77079103 dea03af6 78a8565e dee356df 21f05cbe 8b75e387 b3c50651 b8a5c3ef
d8eeb6d2 e523be77 c2154529 2f69efdf afe67afb f470c4b2 f3e0eb5b d6cc9876
39e4460c 1fda8538 1987832f ca007367 a99144f8 296b299e 492fc295 9266beab
b5676e69 9bd3ddda df7e052f db25701c 1b5e51ee f65324e6 6afce36c 0316cc04
8644213e b7dc59d0 7965291f ccd6fd43 41823979 932bcdf6 b657c34d 4edfd282
7ae5290c 3cb9536b 851e20fe 9833557e 13ecf0b0 d3ffb372 3f85c5c1 0aef7ed2
```

FIM FIM FIM FIM FIM FIM